UNIVERSITY OF CALIFORNIA
Los Angeles

Data-Driven Optimization for Modeling in Computer Graphics and Vision

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Lap-Fai Yu

2013

ABSTRACT OF THE DISSERTATION

Data-Driven Optimization for Modeling in Computer Graphics and Vision

by

Lap-Fai Yu

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Demetri Terzopoulos, Chair

In view of the immense and rapidly increasing quantity of user-created 3D content and real-world scene data publicly available on the internet, as well as the widespread popularity of data acquisition devices such as low-cost depth cameras, it has become convenient to acquire or access data that can potentially be utilized for modeling. In this thesis, we explore how data-driven optimization can be adapted to the essential task of modeling, both from the computer graphics and computer vision perspectives.

We first discuss the conceptual innovations inherent to model synthesis through data-driven optimization, along with the advantages of and considerations in its application. We then tackle various challenging modeling problems within our novel framework. In the context of computer graphics, we devise data-driven optimization methods for virtual world modeling, virtual character modeling, and interactive scene modeling. In the context of computer vision, we devise data-driven optimization methods for 3D surface reconstruction from images.

The dissertation of Lap-Fai Yu is approved.

Joseph M. Teran

Song-Chun Zhu

Stanley J. Osher

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2013

To my parents . . .

who planted seeds of creativity,

curiosity and enthusiasm in my mind.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Acknowledgments

I am very grateful to many people who kindly offered help and support throughout my PhD study.

It is one of the greatest fortunes in my life to have been advised by Professor Demetri Terzopoulos during my PhD program. He has always been very supportive of my thoughts and ideas, placing his confidence in me to carry out the many research plans we have had over the years, which eventually led to various successes. The Make-it-Home idea came to my mind in September 2010. Although the idea was unusual and I had little research experience at the time, Demetri was completely supportive and encouraged me to pursue it. Highly motivated, I passionately conducted the research under his guidance. Several months later, it spawned a very well-received SIGGRAPH 2011 paper, my first successful publication. Demetri is a great educator. He encouraged me often, which greatly strengthened my confidence and aroused my interest in scientific pursuits. There were countless moments of frustration and hardship throughout my PhD journey, but Demetri always provided me with professional and kind advice, which helped me to remain strong and overcome the challenges. I learned not only from his research expertise, but also benefited from his standards of research excellence and his very nice character as a respectable mentor—personal qualities that a successful scientist ought to have. My years of study at UCLA have been both fruitful and fun. I felt at home in Demetri's group and will forever think of our lab as a place to visit from time to time.

I would also like to thank my parents for they have dedicated so much to my education. Not only have they taught me to strive for excellence, but they also ignited my passion for knowledge and aroused my curiosity in the wonders of nature. Since my early childhood, I was given complete freedom to conduct my own scientific projects and explore outside the classroom. I feel especially grateful to my father who spent countless hours

educating and discussing with me. In fact, many of the ideas in this thesis originated from the conversations we had while having fun many years ago. In 2009, at the critical moment of deciding whether I should pursue a PhD degree, it was my father who persuaded me till dawn that I should follow my heart and go, even though he understood that I would have to be away from home for years. I feel immeasurably grateful to my parents.

I greatly thank Professor Tony F. Chan for his mentorship and care throughout my PhD study. I enjoyed every discussion with him. He is such an inspiring and motivational mentor that every time after speaking with him I felt even more passionate about research. Despite his very busy schedule, Professor Chan always provided me immediate assistance when I was in need; for example, writing a strong recommendation letter for my Dissertation Year Fellowship application. I am very grateful for his kindness and assistance, and from him I learned what it takes to be a very successful and reputable scientist and leader.

I am very grateful to Professor Sai-Kit Yeung for introducing me to the exciting fields of computer graphics and vision research; for providing me with close guidance and sharing with me his valuable experience in performing successful research and striving for excellence; for the many days and nights we worked hard together at UCLA and Singapore before the SIGGRAPH deadlines; for being very encouraging in discussing various creative research ideas, which led to success after success; and for his heartful words of kindness and encouragement throughout these years. I feel so proud of the successes that we achieved together and I look forward to even more to come.

I would like to thank Dr. Stephen Lin for his great care and mentorship during my internship at Microsoft Research Asia. His insight and leadership was excellent and I enjoyed every research meeting we had. Under his guidance, I successfully published my first paper in the CVPR conference. This was an important milestone of my intellectual endeavors in computer vision research, for which I thank him very much.

My thanks go to Professor Yu-Wing Tai for his excellent advice and guidance both in my research and career. I also thank Li Chen and Jaesik Park for the various fruitful interactions we had at MSRA, where we learned and had so much fun together. I would like to thank Shuochen Su for his kind help on the CVPR project. I would like to thank Dr. Shawn Singh for his very generous assistance on the Make-it-Home project. I also thank Dr. Alex Vasilescu for her important instruction on research discipline, which I always keep in mind and teach to my future students. I am also grateful to Professor Vladlen Koltun for his mentorship on the Clutterbrush project. I also feel very grateful to Professor Dit-Yan Yeung who inspired me greatly on data-driven approaches during my M.Phil. study, which laid the foundations of many of the ideas that I had in my PhD work. I would also like to thank Professor Chi-Keung Tang for his encouragement from time to time since my undergraduate study, and for his professional help and guidance in the Make-it-Home project. My thanks also go to Professors Dit-Yan Yeung, Chi-Keung Tang, Quan Long, and Pedro V. Sander for writing me strong recommendation letters, which supported my admission into the CS PhD program at UCLA.

I would also like to thank Professor Song-Chun Zhu for serving on my PhD thesis committee and for his inspiring lectures on statistical inference, where I got greatly motivated in how the techniques could be used in computer graphics. I am grateful for his advice on my research directions during my oral qualifying examination, which were insightful and great to pursue. I would also like to thank Dr. Wenze Hu for his help on the formulation of the Make-it-Home project. I am grateful to Yibiao Zhao for the various interesting discussions we had about research and for his generous help on the usability study of the Clutterbrush project. I sincerely hope to collaborate with them in the future.

It was my great honor to work with Professor Stanley J. Osher in the Make-it-Home project, and I would like to thank him for inviting me to present my work to his research group and for serving on my PhD thesis committee. I would like to thank Professor

Joseph M. Teran for teaching me computational linear algebra, which I found highly useful in my research and for serving on my PhD thesis committee.

I would like to thank my roommate, Lap-Fai Lee, for his professional advice on the user studies and on many other things. I enjoyed the various discussions we had. I also thank many friends, including Lowell Bander, Kristen Aw, Li Ma, Anh Do, Wenjia Huang, Weiguang Si, and many others who volunteered to help with our user studies.

I sincerely thank Ka Keung Lau, who encouraged me both in research and in many other things over the years. I also thank the Cheung family, who took care of me with love throughout my childhood and provided me with great chances to explore computers and technology. Many of the research ideas I have had in computer graphics originated from the countless hours of fun and interaction with video games that I had with the Cheung family, something I could have never imagined. This dissertation is also written in memory of my grandmother who lovingly took care of me in my childhood.

It has been my great pleasure to work with Noah Duncan in my fourth year of PhD study. I very much enjoyed the time we brainstormed interesting research ideas together and I look forward to our exciting projects in the years to come. I also thank Luz Orozco for her kindness and help in the early years of my PhD study. I would also like to thank Professor Michael S. Brown for his valuable advice on my research directions and his great help on narrating my video demos. Teresa Wan also assisted in narrating a demonstration video. I also thank Professor Hongbo Fu for the various research discussions we had. My thanks also go to Jan Adamec who kindly modified his room arranger program to enable my initial experimentation in the Make-it-Home project.

I sincerely appreciate the Sir Edward Youde Memorial Fellowship, which generously provided me financial support throughout my PhD study. I especially thank Lady Youde for her kindness and warm greetings each year. I also thank Ms. Fung and Ms. Christine Cheng at the Sir Edward Youde Memorial Fund Council for their kind support. My

thanks also go to Ms. Era Wong for her financial support throughout the years. I also greatly appreciate the UCLA Dissertation Year Fellowship that supported me in the final year of my PhD program.

Last but not least, I am very thankful to an anonymous Culver City bus driver; had he not been driving on time, I would have missed the most important person that I have met throughout my PhD journey—my wife, Lin.

VITA

2007	B.Eng. (Computer Science), HKUST.
2007	B.BA. (General Business Management), HKUST.
2009	M.Phil. (Computer Science), HKUST.
2009–2010	Instructional Assistant, Department of Computer Science, HKUST.
2010–2011	Research Assistant, Computer Science Department, UCLA.
2012–2013	Research Summer Intern, Microsoft Research Asia.
2013–2014	Teaching Assistant, Computer Science Department, UCLA.

PUBLICATIONS

Yu, L.-F., Yeung, S.-K., Koltun, V., and Terzopoulos, D. (2013). "Clutterbrushing". Technical Report UCLA-CSD-130015, UCLA Computer Science Department. To be submitted to the *ACM SIGGRAPH 2014 Conference*.

Yu, L.-F., Yeung, S.-K., Tai, Y.-W., and Lin, S. (2013). "Shading-based shape refinement of RGB-D images". In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Portland OR, June, 1–8.

Yu, L.-F., Yeung, S.-K., Tai, Y.-W., Terzopoulos, D., and Chan, T.F. (2013). "Outdoor photometric stereo". In *Proc. IEEE International Conference on Computational Photography (ICCP)*, Cambridge, MA, April, 1–8.

Yu, L.-F., Yeung, S.-K., Terzopoulos, D., and Chan, T. F. (2012). "Dressup!: Outfit synthesis through automatic optimization". *ACM Transactions on Graphics*, 31(6):134:1–14. (*Proc. ACM SIGGRAPH Asia 2012 Conf.*, Singapore, November, 2012.)

Yu, L.-F., Yeung, S. K., Tang, C.-K., Terzopoulos, D., Chan, T.F., and Osher, S. (2011). "Make it home: Automatic optimization of furniture arrangement". *ACM Transactions on Graphics*, 30(4):86:1–11. (*Proc. ACM SIGGRAPH 2011 Conf.*, Vancouver, Canada, August, 2011.)

CHAPTER 1

Introduction

In recent years, computer-based three-dimensional (3D) modeling has given rise to some amazing applications. To list just a few, several motion pictures, such as “Avatar”, are either in whole or in part situated in virtual worlds populated by virtual characters and creatures, some of the latest video games feature entire virtual cities, and Google, Inc., is steadily creating a 3D digital archive of the entire globe (Google Earth). On the whole, the opportunities of modeling technology are endless and its ongoing and future impact is exciting.

Considering the complexity of the real world, however, the state-of-the-art in creating content for most virtual world applications such as video games, motion pictures, and online applications remains woefully simple-minded. It is cumbersome and time-consuming to create detailed 3D content using today’s modeling technology. This contributes to the fact that the production of games and movies requires tremendous investments of time and money, and that one still cannot create 3D models as easily as one can draw with a pen. The difficult long-range goal confronting us is to enable digital content creators, whether they be experienced digital artists or mere amateurs, to quickly represent and synthesize virtual worlds that are as rich as the real world. To this end, much research remains to be done to improve both 3D content-synthesis capability and speed.

In this thesis, we delineate and tackle a variety of important and challenging 3D modeling research problems. In particular, we investigate the application of data-driven

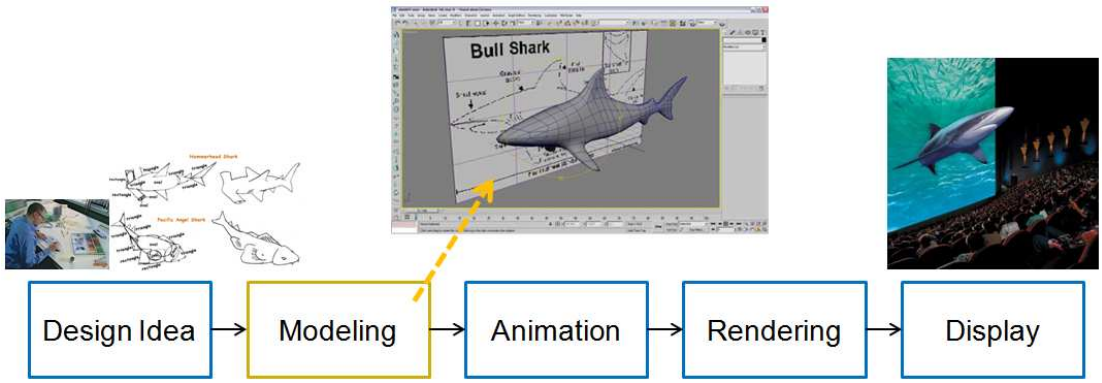


Figure 1.1: *The 3D computer graphics pipeline.*

optimization to the essential task of modeling, both in the context of computer graphics and computer vision. The objective from the graphics perspective is to design powerful and intuitive modeling tools to assist digital artists in creating 3D content. The objective from the vision perspective is to reconstruct realistic 3D models of objects and scenes from image data.

1.1 The Modeling Problem

Modeling in computer graphics refers to the process of creating mathematical models to represent 3D objects in the virtual environment. Figure 1.1 depicts the role of modeling in the 3D computer graphics pipeline. After creation, 3D models are typically rendered, which takes into account the interaction between the illumination in the virtual environment with the geometry and reflectance properties of the 3D model, resulting in a high quality synthetic image of the virtual scene if, as is typical, the goal is to achieve photorealism. Such virtual 3D models can also be animated. For example, 3D virtual characters used in motion pictures are usually animated in a way similar to how real human actors would act in order to advance a storyline. Furthermore, such virtual 3D models can be fabricated in order to produce a 3D physical model in the real

world—the recent trend of 3D printing in computer-aided design. Overall, 3D models serve as important prerequisites in various graphics and design applications.

A large component of modeling research aims to make innovations in the virtual 3D model creation process; for example:

- How can we generate photorealistic 3D models that closely mimic objects of interest in the real world?
- How should modeling tools be designed to enable fast and easy modeling?
- Is it possible to generate a variety of realistic 3D models automatically?

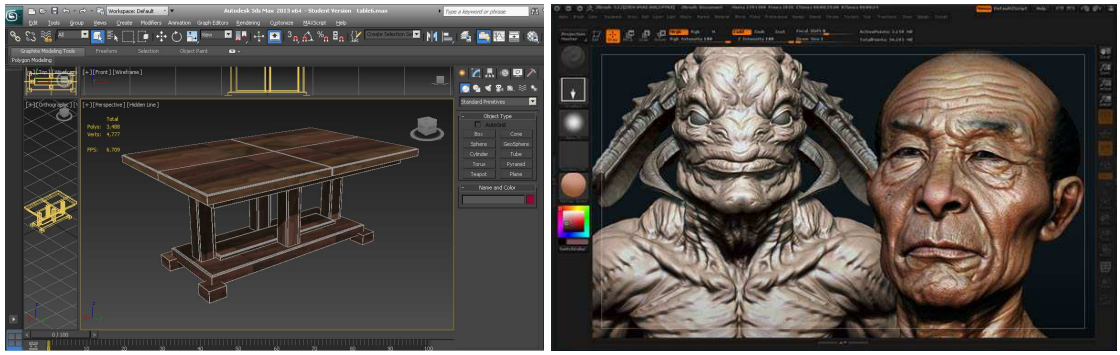
1.2 Common Modeling Approaches

This section describes modeling approaches that have been widely adopted for graphics production (e.g., for movies and games).

1.2.1 Manual Model Creation

The most common way to create 3D models is by manual means. Polygonal modeling and digital sculpting are popular manual modeling methods, and the choice between them depends on the type of the 3D model to be created. For modeling man-made objects such as furniture and buildings, polygonal modeling is preferred because the polygonal mesh representation is generally consistent with the shapes of man-made objects. For modeling organic objects such as virtual human characters, digital sculpting is preferred due to its added flexibility in the mesh modeling process.

Manual 3D modeling is commonly done via interactive modeling software (e.g., 3ds Max, Maya, ZBrush) and Figure 1.2 shows some modeling interfaces. Commercial



Polygonal modeling Digital sculpting

Figure 1.2: *Common methods used in the manual creation of 3D models. Left: Polygonal modeling in 3ds Max; Right: Digital sculpting in ZBrush.*

modeling software has reached a level of sophistication that their interfaces, which provide a wide range of modeling control, enable the creation of highly realistic and detailed 3D models. However, the control complexity poses quite a steep learning curve, so that 3D modeling has become the privilege of professional digital artists. But even for professionally-trained experts, the creation of detailed 3D models still requires a lot of time and effort, which often accounts for the tremendous investment of time and money in movie and game production. For example, in the 2009 movie "Avatar", which was filmed in a virtual world, the production alone cost about $300 million (Wikipedia, 2013) and took several years of intensive labor. For the latest games and movies, which usually demand a vast quantity of high-quality 3D models, the investment is similar or even higher. This severely limits the speed of production and the number of annual releases. In constrast to the situation in the 1980's when a video game usually involved simple 2D graphics and the entire production could be carried out by a small team, the current barrier to production in the video game and movie industry is huge.

While the manual approach to creating 3D models is perhaps the most straightforward for an artist, he or she can readily benefit from the development of more powerful and intelligent modeling tools and interfaces. Indeed, there have been ongoing efforts in

Figure 1.3: *Procedural modeling methods are used to automatically generate various types of 3D models. Left: Plants synthesized by an L-system (Prusinkiewicz and Lindenmayer, 1996); Middle: Procedurally-generated buildings (Müller et al., 2006); Right: A procedurally-generated city (Parish and Müller, 2001).*

computer graphics research to enhance interactive modeling methods; e.g., (Gal et al., 2009), (Igarashi et al., 1999), (Bokeloh et al., 2011), and (Chaudhuri et al., 2011a), among others, were introduced to assist modelers.

1.2.2 Automatic Model Generation

When the 3D models of interest involve highly repetitive patterns, one may resort to rule-based approaches to automatically generate them. One successful application of rule-based approaches is for the generation of plants, such as trees. (Prusinkiewicz and Lindenmayer, 1996) proposed the use of formal grammars to encode the generative rules of plants. This has proven to be successful and the modeling technique has been widely adopted, such as in the level design components of common game engines (e.g., Unity) and in gaming applications where plants are synthesized to populate the terrain. There are also interesting approaches to apply formal grammars to describe and encode the generation of city layouts (Parish and Müller, 2001) and buildings (Müller et al., 2006), which enable virtual cities to be procedurally synthesized, and this has successfully led to the development of modeling software such as the CityEngine. Figure 1.3 shows some procedurally-generated 3D models.

The major advantages of rule-based procedural modeling approaches is the fast modeling speed and the fact that the entire model synthesis task can be automated once given the correct generative grammar. This makes it an attractive choice for the generation of a vast amount of similar 3D models, such as trees, to populate a virtual world. Furthermore, the generation can be done on the fly; for example, trees can be generated in real time as the user navigates through a virtual forest. The elegant description of the object type as a grammar also results in a small storage size, which is favorable for interactive applications.

However, the automatic, rule-based approaches have significant limitations. First, while the special case of plant synthesis can be described algorithmically by a grammar, this may not be the case for the generation of other types of objects that do not have a repetitive pattern in their 3D forms; e.g., the generation of virtual human characters. It can be difficult to devise a suitable grammar which yields desirable generation. Also, it can be difficult for the user to control the generation process, and the resulting models may not be what he or she desires. There have recently been research efforts to tackle these problems (Stava et al., 2010; Talton et al., 2011); for example, by trying to learn a descriptive grammar from data, or constraining the generative process using data such that the user can exercise better control over it.

1.3 Thesis Objective and Contributions

The present thesis can be stated as follows:

> Based on useful relationships and features extracted from data, which can be created manually or acquired using sensors, optimization approaches can be devised to generate realistic 3D models automatically and to facilitate the manual 3D modeling task.

Accordingly, the objective is to achieve the following goals:

- Devise novel data-driven optimization approaches to automatically synthesize realistic 3D models. Specifically, we will demonstrate such approaches in virtual world modeling and virtual character modeling.

- Devise novel data-driven optimization approaches to facilitate interactive modeling tasks. Specifically, we will demonstrate how such approaches can provide useful modeling suggestions in virtual character modeling and interactive scene modeling.

- Devise novel data-driven optimization approaches to enable realistic 3D surface reconstruction from real-world data. Specifically, we will demonstrate the application of such approaches to perform outdoor photometric stereo and to refine 3D shapes from intensity and depth data.

- Verify the effectiveness of the devised approaches through various qualitative and quantitative experiments, perceptual studies, and usability tests.

Our specific contributions are grouped into the following five projects:

Figure 1.4: *Top: Initial layout where furniture pieces are placed arbitrarily. Bottom: Two synthesized furniture arrangements optimized to satisfy ergonomic criteria, such as unobstructed accessibility and visibility, required of a realistic furniture configuration.*

Make it Home: Automatic Synthesis of Furniture Arrangement (Yu et al., 2011) In Chapter 2, we develop a system that automatically synthesizes indoor scenes realistically populated by a variety of furniture objects (Figure 1.4). Given examples of sensibly furnished indoor scenes, our system extracts, in advance, hierarchical and spatial relationships for various furniture objects, encoding them into priors associated with ergonomic factors, such as visibility and accessibility, which are assembled into a cost function whose optimization yields realistic furniture arrangements. To deal with the prohibitively large search space, the cost function is optimized by simulated annealing using a Metropolis-Hastings state search step. We demonstrate that our system can synthesize multiple realistic furniture arrangements and, through a perceptual study, investigate whether there is a significant difference in the perceived functionality of the automatically synthesized results relative to furniture arrangements produced by human designers.

Figure 1.5: *Outfit optimization with different input dress codes. Left: An input human body with hair color, eye color, and skin color specified, plus a wardrobe of clothing items. Right: Optimized outfits for dress code* Casual *and* Business.

DressUp: Outfit Synthesis Through Automatic Optimization (Yu et al., 2012) In Chapter 3, we develop an automatic optimization approach to outfit synthesis (Figure 1.5). Given the hair color, eye color, and skin color of an input human body model, plus a wardrobe of clothing items, our outfit synthesis system suggests a set of outfits subject to a particular dress code. We introduce a probabilistic framework for modeling and applying dress codes that exploits a Bayesian network trained on example images of real-world outfits. Suitable outfits are then obtained by optimizing a cost function that guides the selection of clothing items to maximize the color compatibility and dress code suitability. We demonstrate our approach on the four most common dress codes: *Casual*, *Sportswear*, *Business-Casual*, and *Business*. A perceptual study validated on multiple resultant outfits demonstrates the efficacy of our framework.

Figure 1.6: *A living room before (left) and after clutterbrushing (right).*

The Clutterbrush: Interactive Scene Modeling In Chapter 4, we introduce the Clutterbrush, an interactive tool for enhancing indoor scenes with small-scale details (Figure 1.6). When the user points to a location in the scene, the Clutterbrush suggests detail items for that location. In order to present appropriate suggestions, the Clutterbrush is trained on a dataset of images of real-world scenes, annotated with support relations. Our experiments and user study demonstrate that the adaptive suggestions presented by the Clutterbrush increase modeling speed and that clutterbrushing enhances the realism of indoor scenes.

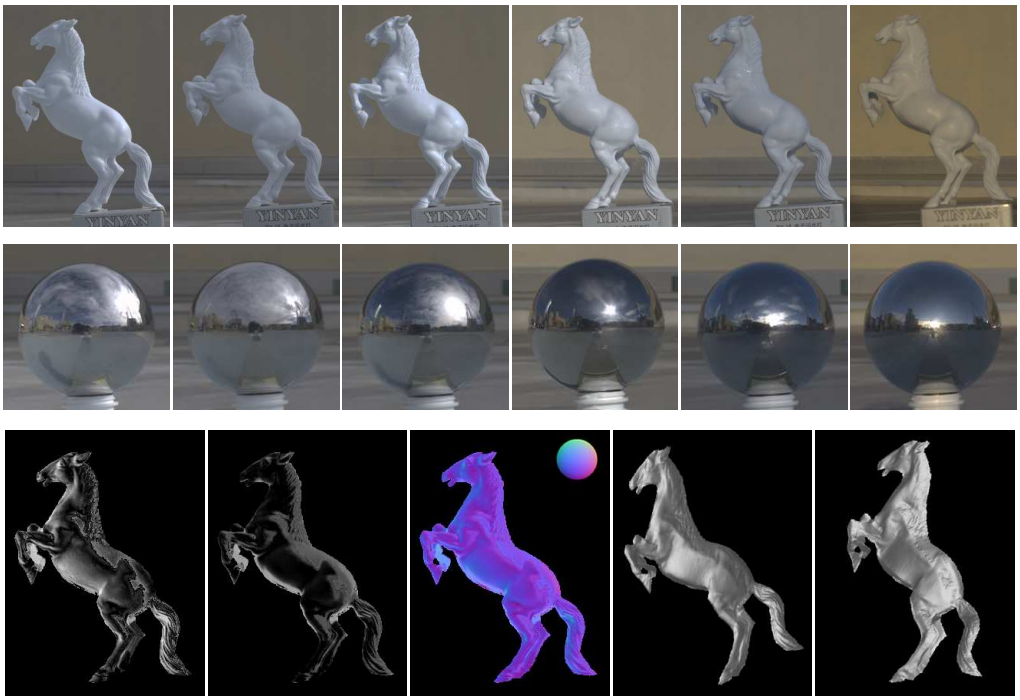
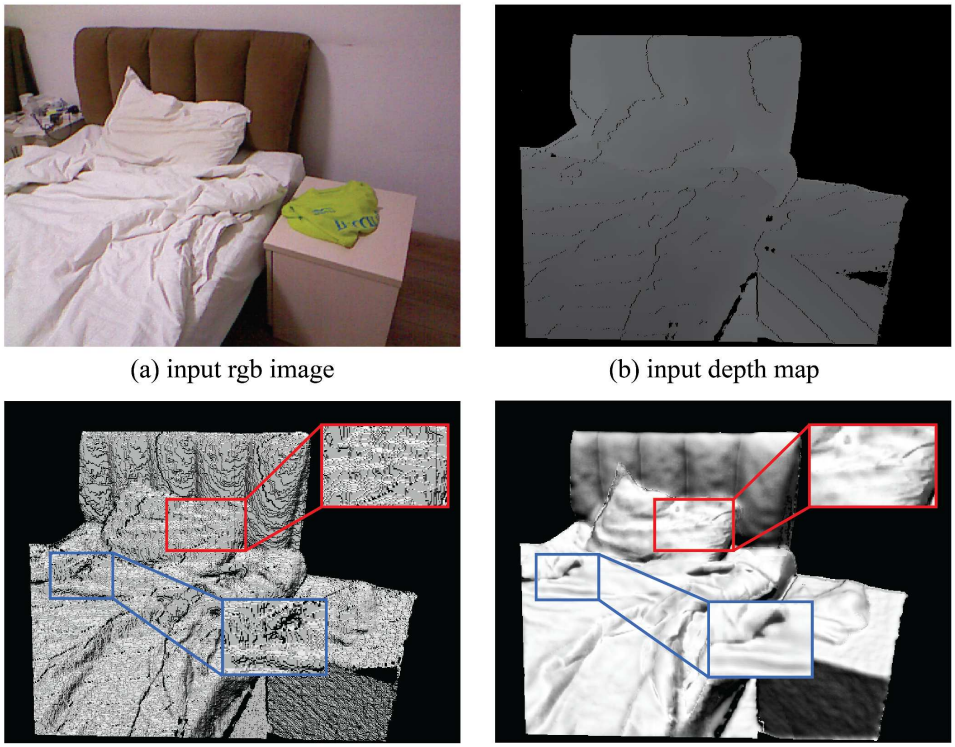


Figure 1.7: *Horse (Sunlight). Top row: Captured input images. Middle row: Environmental illumination captured in a mirror sphere. Note the variance among the input images under different illumination conditions. Bottom row: The left two images show the normal maps* $\mathbf{n}$ *displayed as* $\mathbf{n} \cdot \mathbf{l}$ *with* $\mathbf{l} = [-1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}]^T$ *and* $\mathbf{l} = [1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}]^T$*, respectively. The third image shows the color-coded normal map. The fourth and the fifth images show two different views of the reconstructed 3D surface of the horse.*

Outdoor Photometric Stereo (Yu et al., 2013b) In Chapter 5, we introduce a framework for outdoor photometric stereo utilizing natural environmental illumination (Figure 1.7). Our framework extends beyond existing photometric stereo methods intended for laboratory environments to encompass robust outdoor operation in the real world. We motivate our framework, describe the components of its processing pipeline, and assess its performance in synthetic experiments as well as in natural experiments including objects in outdoor environments with complex real-world illuminations.



(a) input rgb image (b) input depth map

(c) shaded normal map of raw Kinect data (d) our result

Figure 1.8: *Shading-based shape refinement deals with shape/reflectance ambiguities and enhances surface normals computed from raw Kinect data.*

Shading-Based Shape Refinement of RGB-D Images (Yu et al., 2013a) In Chapter 6, we develop a shading-based shape refinement algorithm which uses a noisy, incomplete depth map from the Kinect sensor to help resolve ambiguities in shape-from-shading (Figure 1.8). In our framework, the partial depth information is used to overcome bas-relief ambiguity in normals estimation, as well as to assist in recovering relative albedos, which are needed to reliably estimate the lighting environment and to separate shading from albedo. This refinement of surface normals using a noisy depth map leads to high-quality 3D surfaces. The effectiveness of our algorithm is demonstrated through several challenging real-world examples.

We will next discuss several fundamental issues common to the aforementioned projects.

1.4 Data-Driven Optimization Approaches for Modeling

We are witnessing the rapidly increasing public availability of real-world data and user-created content on the internet. The wide availability of real-world scene sensors has made data acquisition much easier. In addition to high-quality digital cameras, which have become commonplace over the last decade, in recent years low-cost depth cameras (e.g., the Microsoft Kinect) have been growing in popularity. This has made publicly available on the internet a tremendous quantity of RGB color intensity and RGB/depth (RGB-D) data. For example, there exist RGB datasets of nearly every common scene and object type (Xiao et al., 2010; Deng et al., 2009) and RGB-D datasets of many common indoor scenes (Silberman et al., 2012).

These trends have made data-driven modeling a potentially powerful approach, as we will demonstrate in this thesis by devising several data-driven optimization methods for modeling. These novel methods utilize data to assist with the modeling task. The data can be of various types, depending on the problem setting and may or may not be of the same form as the final 3D models that are synthesized by our methods. In particular,

- in the *Make-it-Home* project, the data comprise examples of virtual 3D content that have been manually created by users;
- in the *DressUp* project, the data comprise many annotated fashion images;
- in the *Clutterbrush* project, the data comprise many real-world images annotated with support relations;
- in the *Outdoor Photometric Stereo* project, the data comprise RGB images of the object-of-interest captured under different outdoor conditions;
- in the *Shading-Based Shape Refinement* project, the data comprise RGB-D images captured by a depth camera.

Within our data-driven modeling framework, we identify and represent relevant interrelationships between different entities in the data. The learnt knowledge is usually modeled statistically using a machine learning model, which can differ depending on the problem. For example, in the *DressUp* project, a Bayesian network is applied to statistically represent the learnt probabilities of different clothing combinations in the data. One can then utilize the learnt statistics to assist with the modeling process. In the *Make-it-Home* project, the learnt statistics are used to guide an optimizer to automatically synthesize new furniture arrangements. In the *Outdoor Photometric Stereo* project, shading variation statistics are used to constrain the optimization of the surface normals of the object of interest, thereby enabling a high-quality 3D surface reconstruction of the object.

1.4.1 Advantages

The major advantage of our data-driven approach is that the modeling process is effectively guided by exploiting inherent relationships present in the data that may be difficult to represent explicitly a priori. For example, in the *DressUp* project it seems evident that there are inherent statistical relationships guiding how different clothing items should be worn together, as observed in daily life (we generally do not dress ourselves with clothes randomly selected from our wardrobes). However, it can be difficult to write down all the "dressing rules" and to determine the probabilities of how different clothing items should be worn together, given that there can be hundreds of clothing items and a tremendous variety of possible combinations. By contrast, we can efficiently bring to bear the inherent statistics of clothing combinations by capturing the co-occurrence probabilities of clothing items in a Bayesian network.

Subsequent to the data learning stage, the process of creating 3D models can be formulated as an optimization, wherein the learnt statistics serve as constraints guiding the optimizer in its search for a solution, which is taken as the resulting 3D model. For

example, in the *DressUp* project, the learnt clothing statistics are used to guide an optimizer to synthesize virtual character models which are realistically dressed. There are advantages to formulating modeling as an optimization problem, including:

Explicit Control: Users can have explicit control over the modeling process, and hence the expected optimization results, by adjusting the cost terms used in the optimization. For example, in the *Make-it-Home* project, users can in principle include any interior design / ergonomics factor to be applied in optimizing layouts, as long as it can be formulated mathematically as a cost term. In the *DressUp* project, users can control the variety of the outfits synthesized by controlling the functional form of the style cost term. This is an attractive and important feature of optimization-based model synthesis, especially when the techniques are used in automatic mass production of virtual content (e.g., a building containing hundreds of room layouts, or a crowd composed of thousands of virtual characters).

Automation: Users can automate the entire modeling process. This results in a tremendous saving of work and time because the tedious routine of modeling is instead accomplished by the optimizer through automatic synthesis.

Guided Modeling: Users can semi-automate the modeling process while having close, interactive control. For example, in the *Clutterbrush* project, an optimization-based suggestive modeling interface facilitates indoor scene modeling. During the modeling process, the suggestion engine instantly provides optimized object suggestions to assist the user in quickly embellishing the scene in a realistic manner.

Multiple Design Exploration: Users can obtain multiple, realistic modeling results. The optimizer in general stops once it reaches a local optimum, which it returns as a

modeling result. As there can usually be multiple local optima with respect to the cost terms, the optimizer can be run for multiple trials, where it generates a feasible result in each trial. This way, multiple realistic modeling results can be obtained, which may all be selected for the modeling task or serve as modeling inspirations for the user. For example, in the *Make-it-Home* project, the optimizer is run multiple times to generate different optimized layouts for a factory scene. These generated layouts can all be used to populate rooms inside a factory.

1.4.2 Challenges

The data-driven optimization approach to modeling is a promising direction, yet it poses various research challenges:

What data to use? The data to be used is specific to the modeling task. In some cases, such as in the *Make-it-Home* project, the data (example room layouts) and the modeling result (synthesized layouts) are of the same form. A top-down approach is usually adopted in which the desired model type (e.g., house models or plant models?) is first determined. Then we identify and generalize the common characteristics among the instances of the desired model type. Finally, we seek or capture the data that contains the necessary inherent relationships in order to synthesize the desired model type.

How to obtain the data? The data may be captured from the real world by data acquisition devices; for example, the *Clutterbrush* is trained using RGB-D data captured by a depth camera in real-world indoor scenes. The data may also be completely synthetic; for example, in the *Make-it-Home* project, the room layouts that serve as training data are manually created by users. In many cases, the training data must be annotated by humans when it involves features that are difficult to be accurately identified by automated classification techniques. For example, in the *DressUp* project, human users

helped to annotate the clothing items that are present in the fashion images. In such scenarios, a recent trend is to leverage collaborative "personpower" over the internet to perform the annotation; for example, the annotation services provided by the Amazon Mechanical Turk. Nevertheless, this routine process can be a challenge to data-driven approaches especially when the dataset is a large-scale one.

How to mitigate noise in the data? Another problem is that data captured in the real world are often noisy and suffer from various problems due to the limitations of the data acquisition devices. For example, the RGB-D data captured by low-cost depth cameras such as the Kinect are usually noisy and incomplete, or that data-capture can only be performed indoors. Therefore, the quality of 3D models captured by the Kinect remains far from satisfactory for immediate use. Various denoising techniques have been researched to address this problem. Our work in this thesis on *Shading-Based Shape Refinement* of RGB-D images can be regarded as a useful postprocessing technique, inasmuch as it exploits shading information to improve the raw depth data acquired by common depth cameras, so that a high-quality 3D model of the scene can be obtained as the final output.

What relationships to learn from the data? This refers to identifying the useful features to learn from the data, which are to be used for guiding the modeling task in the subsequent stage. Note that the same data may be utilized differently in different scenarios, depending on the modeling objective, and that different types of features may be extracted accordingly. For example, while both the *Clutterbrush* project and the *Shading-Based Shape Refinement* project make use of RGB-D scene data from the Kinect, support relations were used in the former while shading details were used in the latter.

How to represent the learnt relationships? The relationships learnt from the data must be encoded in a machine learning model, which supports queries in the subsequent modeling (optimization) stage. For example, the *DressUp* project makes use of a Bayesian network to encode the co-occurrence probabilities of different clothing items. Note that the choice of the learning model is largely determined by how the learnt relationships will be used in the modeling stage. For example, using a Bayesian network, we can readily query various conditional and joint probabilities in the modeling stage, thus providing flexibility in making arbitrary probabilistic queries in the modeling interface and, hence, increasing convenience to the user.

How to optimize with respect to the learnt relationships? This refers to the choice of optimizer. The optimization algorithm searches for a local optimum which will be taken as the modeling result. First, one needs to encode the desired relationships from the data as cost terms whose minimization should give a desirable modeling result. Hence, the choice of cost terms directly affects the quality of the result. For example, one can use robust functions in formulating the cost terms in order to alleviate bad effects due to noise in the data. Second, one must choose an optimization algorithm to optimize the objective function comprising the cost terms. Many factors must be considered in this choice, which usually depend on the complexity of the optimization landscape posed by the cost terms. For example, for a simple quadratic cost function, one may obtain an optimal solution by a simple least-squares method. If the optimization landscape is highly non-convex, however, one may need to resort to a stochastic optimization method to obtain optimal solutions; for example, the Markov chain Monte Carlo method. There are other practical challenges; for example,

- How should the optimization be initialized?
- Is the optimization result sensitive to the initialization?
- How to avoid bad local minima?

- How long does the optimization take?
- When should the optimization stop?

For example, in cases where the optimization result is sensitive to the initialization, one must determine how to obtain a good initialization from the input data. For interactive applications, long computational times are impractical, and one must either adopt a strategy to speed up the optimization (e.g., by parallel processing) or use simpler functional forms for the cost terms that can be more efficiently optimized.

The following sections detail the use of data-driven optimization approaches for modeling both from the computer graphics and computer vision perspectives and further explain some of their attractive features.

1.4.3 The Computer Graphics Perspective

This section discusses the role of data-driven 3D modeling in the context of graphics. This refers to the top-down modeling process: Given the common characteristics of certain categories of 3D models, what tools can we devise to facilitate the modeling process? Examples include the modeling of virtual buildings, furniture objects, and human characters. There are two main streams of techniques in graphics modeling, (i) interactive modeling techniques and (ii) automatic modeling techniques.

The first category, interactive modeling techniques, focuses on providing convenience and semi-automation (e.g., providing useful suggestions) to assist the user in the modeling process. For example, intelligent suggestions generated using Bayesian networks may be provided to enable assembly-based modeling (Chaudhuri et al., 2011a) and an interactive tool may be devised to assist interior modeling following interior design guidelines (Merrell et al., 2011a). These kinds of techniques are particularly desired in subjective, artistic creation, as they can enhance the ease of modeling while still pro-

viding artists the necessary freedom to control the creative process. Moreover, these kinds of tools can greatly ease the difficulty faced by layman modelers by providing useful initial modeling suggestions for them. The convenience they offer makes them an attractive choice for integration in future modeling interfaces intended for general users; e.g., the virtual character / interior layout modeling interfaces present in video games such as “The Sims”.

Our work on outfit synthesis in the *Dressup* project partly falls into this first category. Trained by clothing relationships gleaned from real-world fashion images, our framework can act as an interactive suggestion engine for character modeling. For example, one can fix a clothing item (e.g., sport-shoes) and ask for outfit suggestions. As the probabilistic inquiry and outfit synthesis process is almost instant, one can modify modeling attributes (e.g., clothing color) and get useful suggestions on the fly. This is highly practical; e.g., in the character modeling engine of gaming applications, which nowadays usually feature many user-created clothing items.

Our work in the *Clutterbrush* project is also aligned with the data-driven, interactive modeling direction. Our approach is built on the novel idea of using real-world scene data to assist with virtual world modeling. The Clutterbrush is an interactive scene modeling tool that is trained on the support relations present in real-world scene data. It encodes the learnt knowledge into a probabilistic framework such that when the modeler models different parts of a virtual room, the Clutterbrush will quickly analyze the virtual environment and provide instant, useful object suggestions that can be directly chosen by the modeler. We will show in the usability tests that, using the Clutterbrush, users can in general model an indoor scene much faster, while the resulting scene shows richer scene details, which we refer to as clutter objects.

The second category, automatic modeling techniques, focuses on providing full-automation in the modeling process; hence, they are suitable for generating multitudes of realistic models in a timely-manner with minimal human intervention. Recently, there have

been efforts on the automatic generation of realistic indoor environments; e.g., floor-plans (Merrell et al., 2010) and furniture arrangement (Yu et al., 2011). The general advantage of automated techniques is the capability to quickly model a large-scale virtual world, and those methods can potentially be incorporated in gaming/map-navigation applications to generate realistic models on the fly.

Our *Make-it-Home* project on automatic furniture arrangement falls into this category. Useful relationships among furniture objects and the scene are learnt from positive examplar scenes, which are subsequently enforced in the model synthesis phase. The approach can be used to generate a large number of different furniture layouts, which are often needed in virtual world applications such as to populate the rooms in virtual building in video games. This work is a breakthrough in the direction of automatic level design.

In our *DressUp* project, different realistic outfits under the same dressing style can be automatically generated by a MCMC sampler, which can be used to clothe a large number of virtual characters automatically in real-time, in order to synthesize crowd of characters exhibiting realistic variety. This can greatly enhance realism in video games as well. For example, non-player characters (NPC) can be dynamically generated, each exhibiting a different dressing style. This eliminates the artifactual repetitiousness resulting from the commonly-used work-around of cloning and reusing clothing textures on multiple virtual characters.

1.4.4 The Computer Vision Perspective

This section discusses the role of data-driven 3D modeling from the computer vision perspective. This refers to the bottom-up modeling process: Given real-world data, such as RGB and depth values, how can we reconstruct a realistic 3D model? This has been an active research area for decades, and various methods have been proposed;

Figure 1.9: *A view of the virtual San Francisco in Google Earth.*

e.g., shape from shading, photometric stereo, multi-view stereo, and structure-from-motion. Each method is based on some assumptions, for example, Lambertian surface reflectance, controlled lighting environment, uniform albedo, convex surfaces, etc., and each method has its own limitation; for example, multi-view stereo performs poorly on textureless objects and the reconstructed point cloud is usually sparse, while shape-from-shading uses only one image but generally suffers from ambiguity.

The main advantage of vision-based 3D modeling is that it can allow realistic 3D reconstruction of real-world scenes and objects in a fast and faithful manner. For example, the Google Earth project is an ambitious project to reconstruct a 3D model of the entire world (Figure 1.9 shows a view of the virtual San Francisco in Google Earth). The scale of the problem seems to be infeasible for an approach that would attempt to manually create a realistic 3D model for every building. Hence, a vision-based approach—in particular, an extensive spectrum of stereo-based 3D reconstruction techniques—is more promising than the graphics-based approach, since it is easier to acquire the vision data needed (RGB images) than to model every building in the world manually.

A major challenge with the vision-based 3D modeling approach is to deal with the many different variations in surface properties among different objects in the real world. As a result, specific approaches have been devised to tackle the modeling of specific categories of objects, each based on different assumptions. For example, there are vision-based approaches specifically tailored to the modeling of plants (Quan et al., 2006), trees (Tan et al., 2007), hair (Chai et al., 2012), clothes (Wang et al., 2011a), etc.. However, the goal remains the same—given the input data, to realistically reconstruct the object of interest in 3D.

Due to the different challenges and problems in the vision-based 3D modeling pipeline, numerous opportunities arise for computer vision research. In particular, the growing popularity of low-cost RGB-D (e.g., Kinect) and light-field cameras will make possible a new wave of 3D reconstruction and indoor scene modeling approaches, which should be accessible to general users, thereby making a significant impact. In fact, there have been active research efforts to reconstruct indoor scene using low-cost depth cameras (Izadi et al., 2011).

The computer vision portion of this thesis focuses on easing the 3D reconstruction process and improving the quality of the reconstructed 3D surfaces. In particular, we devise approaches to make 3D reconstruction more accessible to general users collecting data under natural indoor/outdoor environment rather than in controlled, laboratory environments.

In our *Outdoor Photometric Stereo* project, we demonstrate for the first time that high-quality 3D reconstruction by photometric stereo can be performed outdoors with the use of a mirror sphere; for example, by using natural sunlight over a day. The same technique can also be applied under natural indoor illumination, hence relaxing the conventional assumption of a darkroom setting during data-capture, thus making photometric stereo much more accessible to general users.

In our work on *Shading-Based Shape Refinement* of RGB-D images, we aim at repairing the Kinect depth-map and refining the 3D shapes obtained. The Kinect provides an economical solution to indoor scene reconstruction; however, the depth maps that it captures usually suffer from a considerable amount of missing values, resulting in breakages in the reconstructed 3D models. Various Kinect-based applications usually fill in such holes by simple smoothing. In this project, in order to repair the depth-map and refine the shape obtained, we make intensive use of illumination estimation and shading cues present in the corresponding RGB image captured. The developed technique promises to benefit a wide range of Kinect-based 3D applications.

1.5 Thesis Synopsis and Organization

To summarize the thesis at hand, in view of the large amount of abstract relationships inherent among 3D models and the growing amount of 3D model data that are freely available (e.g., 3D models in Trimble 3D Warehouse, Kinect indoor scene dataset), data-driven approaches show great promise for semantic labeling, relationship extraction, object search, and model synthesis. Our data-driven modeling approach generally comprises learning and synthesis phases. Abstract relationships are initially learnt from real-world / human-provided training data, in order to train a generative statistical model that supports inquiries in the subsequent optimization-based model synthesis process. It is exciting to explore creative applications to the modeling problem in vision and graphics, and vision and graphics techniques can ultimately be fused together to reinforce the modeling process.

The remainder of the thesis is organized as follows: Aiming at demonstrating the data-driven optimization approaches for modeling, we will present five projects in which we devise novel, data-driven optimization approaches for model synthesis. These will be discussed in the five chapters that follow—in Chapters 2, 3, and 4, we tackle three

modeling problems from the computer graphics perspective, whereas in Chapters 5 and 6, we tackle two modeling problems from the computer vision perspective. In each chapter, we will begin with a problem statement and introduction, followed by a review of related work relevant to the problem, a presentation of the technical details of our novel data-driven optimization method suited to the modeling task at hand, followed by experimental validations, discussion of results, and suggestions for future work. Chapter 7 concludes the thesis.

CHAPTER 2

Virtual World Modeling: Make-it-Home

2.1 Introduction

Whereas in recent years numerous publications have appeared demonstrating the automatic modeling of building exteriors and facades, the automatic generation of realistic indoor configurations has not yet received the attention that it deserves. With the growing popularity of social virtual worlds and massively-multiplayer online games that feature large quantities of realistic environmental content, automated procedural methods for synthesizing indoor environments are needed, as it would be too tedious and impractical to model every indoor scene manually. Currently, such indoor modeling is usually simplified or even ignored, which severely limits the realism of many virtual environments.

A realistic indoor scene is typically populated by several different kinds of furniture objects, but only a few of the many possible spatial arrangements of these objects are functional and livable. For example, the front of a television or computer screen should not be blocked, since it is supposed to be visible. Furthermore, most of the objects in the scene should be accessible to human habitants. On the other hand, one object is often placed on top of another object, such as a vase on a table, so there exists a hierarchical relationship among the two objects if we regard the carrier object as the parent and the supported object as its child.

While the aesthetic and creative process of interior design would best be done by professional interior designers, our goal is to create software capable of automatically generating furniture arrangements for complex indoor scenes that are optimized to respect important ergonomic factors. This technique would be useful in multiplayer online games and other graphics applications requiring fully automatic interior design with a high degree of realism. The system that we present achieves this goal in two stages.

First, our system extracts spatial relationships on the placement of furniture pieces from user-supplied exemplars of furnished indoor scenes. This step is done only once, in advance. The acquisition of examples and subsequent extraction of spatial relationship should not be costly, given that many virtual worlds feature user-created content and collaborative design. A scene is then initialized with furniture pieces randomly placed at arbitrary positions and orientations. Here, the furniture placement is almost always unlivable, with objects that are wrongly-located (e.g., a bookshelf is placed at the center of the room rather than against a wall) or wrongly-oriented (e.g., a television screen is facing the wall), and furniture is usually blocking pathways between doors.

Given an arbitrary initial arrangement, such as the one shown in Figure 1.4(left), optimizing a furniture arrangement subject to human ergonomics is not an easy task, since the search space can be prohibitively large. To address this issue, in the second stage, the initial layout will be adjusted iteratively by minimizing a cost function that accounts for factors, such as human-accessibility, visibility, pairwise object relationships, and so forth, wherein the spatial relationships extracted from the exemplars are encoded as prior cost terms. We demonstrate that the overall cost function can be optimized by simulated annealing with a Metropolis-Hastings state-search step. From the random initial arrangement in Figure 1.4, the optimization produces the two synthesized example scenes shown in the figure.

We furthermore perform a perceptual study that adopts a subjective forced-choice approach to investigate whether people have a preference based on perceived functionality

between our synthesized results and arrangements produced by human designers.

2.2 Related Work

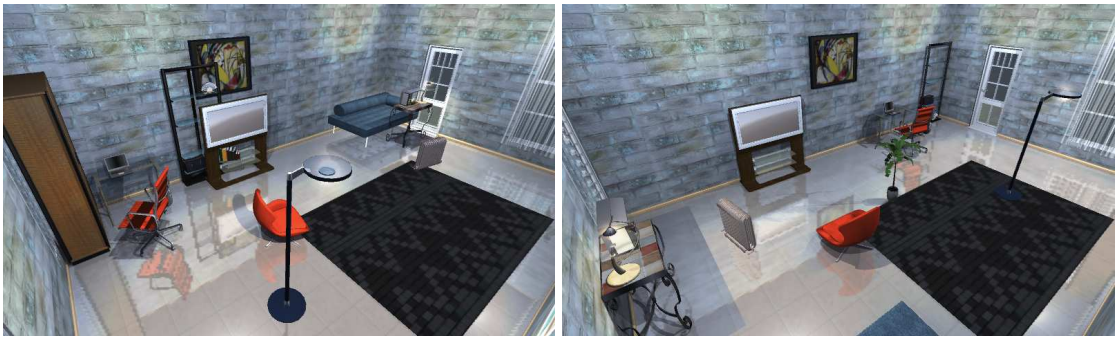
We will first discuss the scarce existing research on generating functional furniture arrangements and then review other relevant work ranging from floor-plan generation and interior design metrics to applications requiring (rapid) generation of livable indoor scenes, such as virtual worlds and artificial life.

2.2.1 Furniture Arrangement

Previous systems that generate furniture arrangements (Kjlaas, 2000; Akazawa et al., 2005; Germer and Schwarz, 2009; Larive et al., 2004; Sanchez et al., 2003) require manual control or intervention, or do not adequately consider ergonomic factors, which makes them susceptible to generating uninhabitable arrangements. Figure 2.1 shows an unsatisfactory arrangement where ergonomic factors are neglected in the interior design.

To generate a furniture layout, (Kjlaas, 2000) represents a given room as a nested hierarchy of rectangular templates, which are swapped by eight predetermined mutation functions. Empty boxes are placed in front of doors and windows to represent free space. However, the approach is limited to rectangular rooms, and each template plus the set of corresponding parameters must be carefully designed.

Akazawa et al. (2005) use a semantic database to explicitly store furniture spatial relationships in order to synthesize new arrangements. Our furniture representation is similar in its use of furniture object interrelationships with parent-child hierarchies where each object is represented as a bounding box. Unlike our approach, however, their inter-object contact constraints must be manually specified in the database.



No human ergonomics With human ergonomics

Figure 2.1: *Examples of furniture arrangement. Left: An unsatisfactory spatial arrangement resulting from the neglect of human ergonomics considerations; note that the furniture objects are packed together near the upper-left corner and are blocking the door. Right: A satisfactory arrangement with realistically positioned furniture objects that are accessible, do not obstruct the door, and include a television that is readily viewable from a well-positioned chair.*

Germer and Schwarz (2009) take a similar approach, regarding each furniture object as an agent seeking to attach itself to a parent object. Since the parent-child relationships of each object must be manually defined, however, this task will become prohibitive as the number of objects grows. Furthermore, as ergonomic factors such as good accessibility and visibility are disregarded, unrealistic and uninhabitable configurations suffering undesired physical or visual blocking are unavoidable.

Recently, Yeh et al. (2012) described an approach that uses stochastic optimization to generate open worlds. The approach is based on manually specified relationships between objects. Fisher et al. (2012) describes a probabilistic approach to synthesize 3D object arrangements using a large dataset of virtual scenes modeled by human users.

2.2.2 Floor Plan Generation

Relevant floor plans can be generated before planning the layout of furniture pieces. Recently, Merrell et al. (2010) proposed a data-driven method to generate residential building layouts. In their work, 120 examples of architectural programs are used to train a Bayesian network that captures the relationships among different rooms. Given certain user requirements as priors, the Bayesian network can then be used to generate a floor plan, which is iteratively modified to incorporate desirable human factors. The result is a synthesized floor plan for a residential building.

We also note the work of Chun and Lai (1997), which encapsulates architectural design knowledge into an expert system. The system modules can be used to evaluate floor plan and furniture arrangement according to government regulations and interior design guidelines, providing suggestions for changes.

2.2.3 Relationship with Interior Design Practices

Although interior design involves creative solutions that can be fairly subjective, a set of quantifiable design criteria has long been accepted in the industry. Specifically, such criteria determine whether the design is functional and suitable for human inhabitants. Panero and Zelnick (1979) conducted a detailed study on human dimensions and ergonomics, by carefully defining metrics such as height, width, reachable-range, and visibility, which are believed to be conducive to functional and comfortable designs. For example, a television should maintain a certain distance from the normal viewing area (e.g., a sofa) depending on the dimensions of its screen. Viewing from an oblique angle should also be avoided for the sake of the viewer's comfort. Note that (Ching and Binggeli, 2005) describes these human factors as the "prime determinants" of interior design, emphasizing that while average measurements should be used, flexibility should be exercised to satisfy specific user needs. In optimization terms, such guide-

lines can be interpreted as soft constraints. In (Mitton and Nystuen, 2007; Ching and Binggeli, 2005), the importance of accessibility in furniture placement is noted, which is a common consideration in decorating rooms with different purposes. Ching and Binggeli (2005) illustrate how pathways connecting doors may affect human movement and interior furnishing. In general, a pathway connecting doors should be a short path that facilitates movement while leaving considerable floor areas for furnishing. The width of a pathway should depend on the habitant's body width, with possible amendments when designing homes for the physically-challenged to allow wheelchair movement.

2.2.4 Related Applications

The realistic synthesis of spatial arrangements of objects can tremendously facilitate virtual world modeling. For example, (Shao and Terzopoulos, 2007) demonstrated a large-scale virtual model of a train station populated by numerous autonomous pedestrians. The mobile human agents can perceive the environmental objects they encounter and respond to them appropriately. An automatic means for properly placing various different kinds of environmental content in the scene would be useful in this context.

Collaborative design spaces have been used to assist object modeling (Talton et al., 2009), and they are commonly used in constructing virtual worlds, such as Second-Life and many massively-multiplayer online games. The first stage of our approach entails the extraction of spatial and hierarchical relationships from positive furnished examples, which is a more practical approach compared to the manual specification of such relationships, especially for scenes where there are hundreds of different kinds of objects.

There are numerous efforts in the modeling and synthesis of cities and building exteriors. In (Chen et al., 2008; Müller et al., 2006; Parish and Müller, 2001; Vanegas

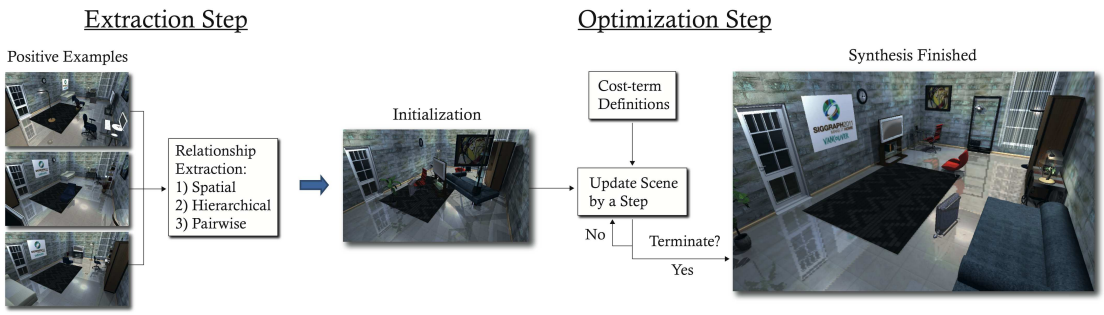


Figure 2.2: *Overview of the Make-it-Home system.*

et al., 2012) a procedural modeling approach was used to realistically synthesize streets, buildings, and cities, by which parameters such as height and age can be specified to guide the synthesis. The result is a realistic city model populated with buildings. Other efforts (Müller et al., 2007; Xiao et al., 2008; Bao et al., 2013a) also employ image-based approaches to model facades. Such techniques can be adopted in Google Earth, Bing Maps, and other applications that enable users to zoom into street views and navigate the exteriors of texture-mapped buildings in 3D.

2.3 Overview of the Make-it-Home System

We have developed a software system that can extract important relationships from examples of interior design, and then automatically rearrange furniture objects in a scene to synthesize different new configurations. This is an important aspect of virtual world modeling of interior environments.

Figure 2.2 provides an overview of our approach, which is divided into two stages: (1) the extraction of spatial, hierarchical, and pairwise relationships from positive examples and (2) the synthesis of novel furniture arrangements through optimization. The following section describes the first stage and the subsequent one describes the second.

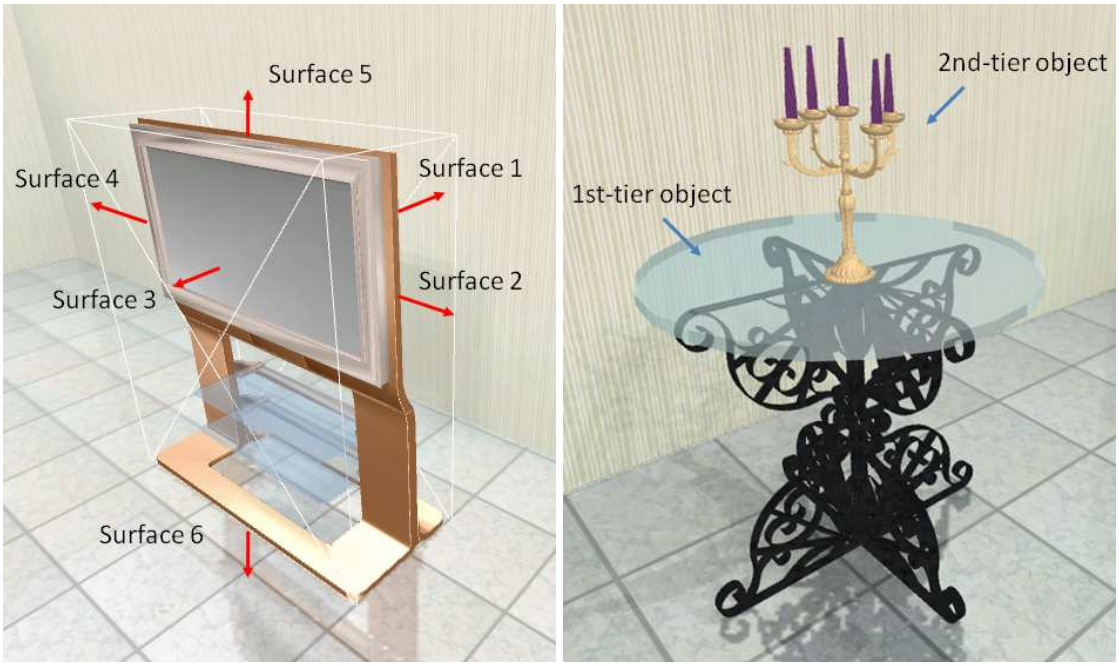


Figure 2.3: *Left: A television, its bounding box, and six surfaces; Right: A candelabrum on a table; the table is a first-tier object and the candelabrum is a second-tier object.*

2.4 Furniture Relationship Extraction

2.4.1 Object Representation

Optimizing furniture arrangement into a realistic and functional indoor configuration involves considerable complexity, taking into account various interacting factors, such as pairwise furniture relationships, spatial relationships with respect to the room, and other human factors. An effective representation that captures the necessary spatial relationships is needed.

Bounding surfaces: Similarly to (Germer and Schwarz, 2009; Kjlaas, 2000), each object in the scene is represented by a set of bounding surfaces (it can be a simple rectangular bounding box or a convex hull to deal with more complex spatial arrange-

ments). Figure 2.3 shows an example object (television) represented by a bounding box whose six surfaces are labeled 1 to 6. Apart from the top and bottom surfaces, we search for the "back" surface of every object, which is the surface closest to any wall. Other surfaces are labeled as "non-back" surfaces. The back surface is used to define a reference plane for assigning other attributes.

Center and orientation: Figure 2.4(a) shows the key attributes of an object—center and orientation, denoted by (p_i, θ_i), where p_i denotes the (x, y) coordinates and θ_i is the angle relative to the nearest wall (defined as the angle between the nearest wall and the back surface). An optimized furniture arrangement $\{(p_i, \theta_i)\}$ involving all objects i is one that minimizes our cost function defined in the next section.

Accessible space: For each surface of the object, we assign a corresponding accessible space (see Figure 2.4(b)). We define a_{ik} to be the center coordinates of accessible space k of object i. The diagonal of the region is measured by ad_{ik}, which is used to measure how deep other objects can penetrate into the space during optimization. The size of the accessible space is set from available examples or given as input related to the size of a human body. If the space is very close to the wall in all the examples, the corresponding surface need not be accessible; otherwise, we set it to be the dimension of an average-sized adult if such a measurement is not given.

Viewing frustum: For some objects, such as the television and painting, the frontal surface must be visible. We assign a viewing frustum to this particular surface. Given an object i, its viewing frustum is approximated by a series of rectangles with center coordinates v_{ik}, where k is the rectangle index. vd_{ik} is the diagonal of the rectangle, which is useful in defining the penetration cost akin to that for the accessible space. Figure 2.4(c) provides an example.

Other attributes: Other attributes are involved in the optimization process. Referring again to Figure 2.4(a), the distance from p_i to its nearest wall is defined as d_i; the diagonal from p_i to the corner of the bounding box is defined as b_i (the current implementation is a rectangle). We also record the z-position z_i of the object.

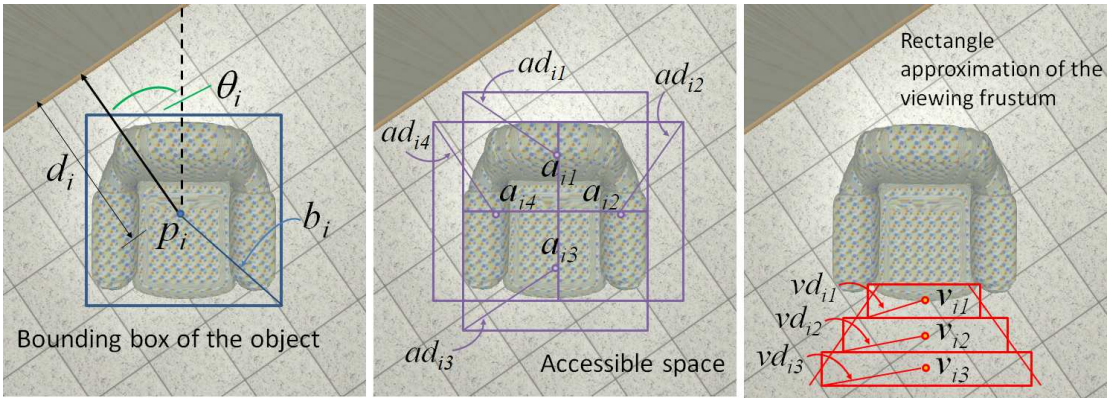
Note that to simplify the optimization process, the translation step considers the (x, y)-space only. In other words, an object's z-position is fixed as the z-position of the surface of its first-tier parent. Nevertheless, the z-position can still change in the swapping step, when a second-tier object changes its first-tier parent and is placed on a different surface. Possible collisions in the z-dimension will still be considered when evaluating accessibility and visibility costs. For example, an overlap between a chair and a bed in the (x, y) space is penalized, while that between a wall clock and a bed is not, as the former involves collision in the z-dimension but the latter does not. Thus, the chair tends to move away from the bed in the (x, y) space, whereas the wall clock does not.

2.4.2 Learning Prior Relationships

Given the above object representation, the following furniture relationships are extracted automatically from positive input examples.

Spatial relationships: The key prior relationships are the distance of an object to its nearest wall $\bar{d}_i$ and its relative orientation to the wall $\bar{\theta}_i$. They are respectively estimated as the clustered means of input examples, where we can assign one of the clustered means as $\bar{d}_i$ and $\bar{\theta}_i$ respectively for object i during optimization. The number of clusters can be preset or estimated by (Grunwald, 2007).

Hierarchical relationships: Given two objects A and B, object A is defined as the parent of B (and B as the child of A) if A is supporting B by a certain surface. Fig-



(a) (b) (c)

Figure 2.4: *An example object i. (a) Length d_i measures the distance of the object center p_i to its nearest wall. Angle θ_i is the orientation of the object relative to the nearest wall (or the tangent plane if the wall is nonplanar). Length b_i gives the diagonal of the bounding box. (b) The object has 4 accessible spaces centered at a_{i1}, a_{i2}, a_{i3}, and a_{i4} respectively. (c) A viewing frustum associated with the object is represented by 3 rectangles centered at v_{i1}, v_{i2}, v_{i3}. Quantities ad_{ik} and vd_{ik} denote the corresponding diagonal lengths of the rectangles.*

ure 2.3 shows a candelabrum on top of a table. The table is hence the parent of the candelabrum, and the candelabrum is the child of the table.

Suppose an example room populated by furniture objects is given. With the room itself regarded as the root, all objects directly supported by the floor or the wall are defined as "first-tier objects" (e.g. bed, table, clock on the wall). All objects supported by a surface of a first-tier object (e.g., a vase on top of a cupboard) are defined as "second-tier objects". A room configuration is thus represented by a hierarchy of relationships. For simplicity, our optimization considers only first-tiers and second-tiers, which should cover most objects of interest.

Pairwise relationships: Certain objects, such as a television and a sofa or a dining table and chairs, interact with each other in pairs subject to pairwise orientation and distance constraints. Each pairwise relationship can be set by clicking the corresponding objects in the UI, after which the mean relative distance and angle are extracted from the examples for use as pairwise constraints.

2.5 Furniture Arrangement Optimization

Given the spatial relationships extracted as described above, our goal is to integrate this information into an optimization framework with a properly defined cost function quantifying the quality of the furniture arrangement. Given an arbitrary room layout populated by furniture objects, the synthesized arrangement should be useful for virtual environment modeling in games and movies, interior design software, and other applications.

The search space of our problem is highly complex as objects are interdependent in the optimization process. The furniture positions and orientations depend on numerous

factors, such as whether the object should be visible or accessible. It is very difficult to have a global optimization scheme or a closed-form solution that yields a unique optimum.

To tackle this problem, we resort to stochastic optimization methods, specifically, simulated annealing (Kirkpatrick, 1984) with a Metropolis-Hastings state-search step (Metropolis et al., 1953; Hastings, 1970) to search for a good approximation to the global optimum. Note, however, that given a room, a set of furniture objects, and the prior spatial and hierarchical relationships, numerous acceptably-good configurations will be possible. This is the rationale for finding a good approximation in a reasonably short time, rather than searching exhaustively over the complex search space in order to find the global optimum of the cost function. The evaluation of interior decoration results can be subjective; hence, we will perform a perceptual study to validate the realism of our synthesized results.

2.5.1 Simulated Annealing

Simulated annealing is a computational imitation of the (physical) annealing process, which gradually lowers the temperature of a heat bath that controls the thermal dynamics of a solid in order to bring it into a low-energy equilibrium state. Theoretically, the algorithm is guaranteed to reach the global minimum at a logarithmic rate given a sufficiently slow cooling schedule (Geman and Geman, 1984). Using such a slow cooling schedule is impractical, however. Nevertheless, it has been widely used to find quasioptimal configurations in circuit design, operations, and many scientific problems. As in the work on floor-plan generation (Merrell et al., 2010), we found that simulated annealing with the simple Metropolis criterion (Chib and Greenberg, 1995) is effective in our problem of optimizing configurations in the space of possible furniture arrangements. For additional details about the simulated annealing method, refer to (Schneider and Kirkpatrick, 2006; Liu, 2008; Aarts and Korst, 1989).

By analogy, the furniture objects in our application are regarded as the atoms of a metal being annealed—they are initially "heated up" to allow flexible rearrangement, and refine their configuration as the temperature gradually decreases to zero. At each temperature, the Metropolis criterion is used to determine the transition probability. It employs a Boltzmann-like objective function

$$f(\phi) = e^{-\beta C(\phi)}, \tag{2.1}$$

where the state of the system $\phi = \{(p_i, \theta_i) | i = 1, \ldots, n\}$ represents a furniture configuration comprising the positions p_i and orientations θ_i of each of the n furniture objects, C is the cost (energy) function, which will be defined in Section 2.5.3, and β is inversely proportional to the temperature, increasing over the iterations as the system anneals from a high temperature to a low temperature. At each iteration, a new furniture configuration ϕ', or "move", is proposed, and it is accepted with probability

$$\alpha(\phi'|\phi) = \min\left[\frac{f(\phi')}{f(\phi)}, 1\right] \tag{2.2}$$

$$= \min\left[\exp(\beta(C(\phi) - C(\phi')), 1\right]. \tag{2.3}$$

Note that the Metropolis criterion can accept moves that increase the cost, which enables the method to avoid becoming stuck at local minima.

Figure 2.5 depicts an example furniture optimization process. We typically initialize the furniture objects in random positions and orientations, a configuration that typically has very high energy. The supplemental videos include animations of the optimization process.

2.5.2 Proposed Moves

To explore the space of possible arrangements effectively, the proposed move $\phi \to \phi'$ involves both local adjustment, which modifies the current arrangement, and a global reconfiguration step that swaps objects, thereby altering the arrangement significantly.

Initialization 1000 iterations 5000 iterations 15000 iterations 25000 iterations

Figure 2.5: *Furniture arrangement optimization from a random initial configuration (left). As the optimization process proceeds, the furniture configuration is iteratively updated until it achieves an optimized final arrangement ϕ^* in 25,000 iterations (right).*

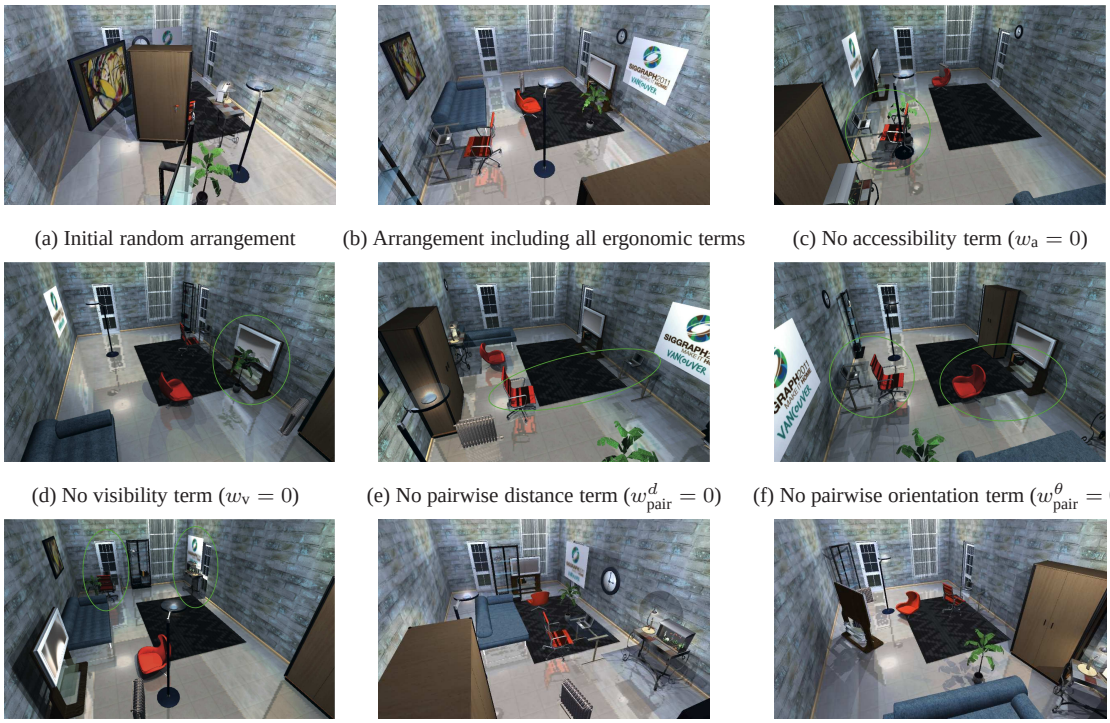
Translation and Rotation: The basic move of the optimization modifies the position of an object and its orientation. For the purposes of the furniture arrangement problem, 2D translation and rotation transformations suffice to configure objects into practicable arrangements, since in most cases furniture objects stand upright on the floor due to gravity. In addition, we found in practice that performing translation and rotation separately gives a more stable optimization. In mathematical terms, an object i or a subset of objects is selected and updated with the move $(p_i, \theta_i) \rightarrow (p_i + \delta p, \theta_i)$ or $(p_i, \theta_i) \rightarrow (p_i, \theta_i + \delta\theta)$, where $\delta p \sim [\mathcal{N}(0, \sigma_p^2)\ \mathcal{N}(0, \sigma_p^2)]^T$ and $\delta\theta \sim \mathcal{N}(0, \sigma_\theta^2)$, with $\mathcal{N}(\mu, \sigma^2) = (2\pi\sigma^2)^{-1/2} e^{-(x-\mu)^2/2\sigma^2}$ a normal (Gaussian) distribution of mean μ and variance σ^2. The variances σ_p^2 and σ_θ^2, which determine the average magnitude of the moves, are proportional to the temperature.

Swapping Objects: To enable a more rapid exploration of the arrangement space and avoid becoming stuck in local minima, a move involving the swapping objects in the existing arrangement may be proposed. Two objects of the same tier are selected at random and their positions and orientations are interchanged: $(p_i, \theta_i) \leftrightarrow (p_j, \theta_j)$ for objects i and j. Object swapping usually changes the cost significantly, thereby leading to considerable rearrangement of the configuration.

Figure 2.6: *Left: A pathway connecting doors. Right: A pairwise constraint between the television and the sofa.*

Moving Pathway Control Points: Given two doors, multiple pathways are possible. By moving the control points of the pathway, which is represented as a cubic Bezier curve, the pathway can change its course to avoid colliding with furniture objects. As shown in Figure 2.6, the free space of a pathway is represented by a series of rectangles along the curve. Thus, pathways can also be regarded as "furniture objects" whose control points may be modified, and a move can be defined as the translation of a pathway control point in a certain direction.

With the aforementioned moves, given a floor-plan and a fixed number of furniture objects that define the solution space, the configuration of a furniture object (p_i, θ_i) has a positive probability to move to any other configuration (p'_i, θ'_i). Given the annealing schedule, the solution space is explored more extensively with larger moves early in the optimization, and the furniture configuration is more finely tuned with smaller moves towards the end.



(a) Initial random arrangement (b) Arrangement including all ergonomic terms (c) No accessibility term ($w_{\mathrm{a}} = 0$)

(d) No visibility term ($w_{\mathrm{v}} = 0$) (e) No pairwise distance term ($w^{d}_{\mathrm{pair}} = 0$) (f) No pairwise orientation term ($w^{\theta}_{\mathrm{pair}} = 0$)

(g) No pathway term ($w_{\mathrm{path}} = 0$) (h) No prior distance term ($w^{d}_{\mathrm{pr}} = 0$) (i) No prior orientation term ($w^{\theta}_{\mathrm{pr}} = 0$)

Figure 2.7: *The effect on the automatic arrangement (b) of the furniture in (a) resulting from the omission of individual cost terms: Disregarding human ergonomics results in unrealistic synthesized arrangements that that are not livable in several ways; e.g., (c) the furniture objects are colliding, (d) a potted plant is blocking the television and the armchair, (e) the work-chair is too far from the desk, (f) the armchair is facing away from the television, (g) the desk and work-chair are blocking the door, (h) furniture objects are too far from the wall, (i) objects are randomly oriented.*

2.5.3 Cost Function

The goal of the optimization process is to minimize a cost function that characterizes realistic, functional furniture arrangements. Although it is often difficult to quantify the "realism" or "functionality" of a furniture arrangement, the following basic criteria should not be violated.

Accessibility: A furniture object must be accessible in order to be functional (Mitton and Nystuen, 2007; Ching and Binggeli, 2005). In Section 2.4.1, we defined for every face of an object an accessible space determined from prior examples and the dimensions of the human body (see Figure 2.4). To favor accessibility, the cost increases whenever any object moves into the accessible space of another object. Suppose object i overlaps with the accessible space k of object j, the accessibility cost is defined as

$$C_{\mathrm{a}}(\phi) = \sum_i \sum_j \sum_k \max\left[0, 1 - \frac{\|p_i - a_{jk}\|}{b_i + ad_{jk}}\right]. \tag{2.4}$$

Note that we simplify the move by dropping the optimization of orientation θ_i, only measuring the relative distance. Our experiments revealed that this simplification suffices to ensure accessibility and more easily prompts the overlapping object to move away.

Visibility: Some objects, such as a television or a painting, impose strict requirements on the visibility of their frontal surfaces, since their fundamental functionality is compromised if their fronts are blocked by another object. For every such object that must be visible, we associate with it a viewing frustum (see Figure 2.4). Similar to the accessibility constraint, whenever another object moves into some object's viewing frustum, the cost increases in order to discourage the move. As discussed in Section 2.4.1, for an object j with a viewing frustum we approximate the frustum by a series of rectangles whose center coordinates are defined as v_{jk}. If object i overlaps with the visibility

approximation rectangle k of object j, the visibility cost is defined as

$$C_{\text{v}}(\phi) = \sum_i \sum_j \sum_k \max\left[0, 1 - \frac{\|p_i - v_{jk}\|}{b_i + vd_{jk}}\right]. \tag{2.5}$$

Note that it is similar to the accessibility cost C_{a}, where the accessible space of object j is replaced by the viewing frustum.

Pathway Connecting Doors: Another important criterion involves pathways between doors (Ching and Binggeli, 2005). The placement of furniture objects such that they block doors should obviously be inhibited. However, a room configuration with circuitous and narrow pathways should also be avoided. To strike a balance, we assume that a pathway in a typical living environment should be smooth, and we define its locus by a cubic Bezier curve, where the free space of the pathway is approximated by a series of rectangular objects, as shown in Figure 2.6. Thus, the movement of furniture objects into the rectangles is penalized. Apart from moving furniture objects, the pathway itself can be adjusted by translating the control points of the Bezier curve. Because a pathway should be free of obstacles and thus visible, the pathway cost C_{path} can be defined similarly as C_{v} defined in Eq. (2.5), and applied to the series of rectangles along the pathway.

Prior: The prior cost controls the similarity between the new configuration and configurations seen in the examples. According to Section 2.4.2, we extract for each furniture object its prior distance and orientation to the nearest wall $(\bar{d}_i, \bar{\theta}_i)$. Alternatively, for any new furniture object that is absent from the positive examples, the user can manually assign the prior. Given a new room layout, the current furniture arrangement will be compared with the prior by

$$C^d_{\text{pr}}(\phi) = \sum_i ||d_i - \bar{d}_i|| \tag{2.6}$$

$$C^\theta_{\text{pr}}(\phi) = \sum_i ||\theta_i - \bar{\theta}_i||, \tag{2.7}$$

where d_i and θ_i can be computed from the current p_i; i.e., finding the distance and relative angle to the nearest wall.

Pairwise Constraint: The pairwise constraint is applied between two furniture objects with a specific pairwise relationship; e.g., the television should be facing the sofa as shown in Figure 2.6, and a bedside table should be close to a bed. It thus encodes the natural affinity of certain furniture objects in the optimized result. We define the pairwise constraint $C^d_{\text{pair}}(\phi)$ and $C^\theta_{\text{pair}}(\phi)$ by simply replacing the distance and orientation to the wall in the prior cost as defined by Eqs. (2.6) and (2.7), with the desired distance and orientation between the pair of objects.

Given the above costs, we define the overall cost function as

$$\begin{aligned} C(\phi) = w_{\text{a}} C_{\text{a}}(\phi) + w_{\text{v}} C_{\text{v}}(\phi) + w_{\text{path}} C_{\text{path}}(\phi) \\ + w^d_{\text{pr}} C^d_{\text{pr}}(\phi) + w^\theta_{\text{pr}} C^\theta_{\text{pr}}(\phi) \\ + w^d_{\text{pair}} C^d_{\text{pair}}(\phi) + w^\theta_{\text{pair}} C^\theta_{\text{pair}}(\phi). \end{aligned} \tag{2.8}$$

The w coefficients determine the relative weighting between the cost terms; in practice, we set $w_{\text{a}} = 0.1$, $w_{\text{v}} = 0.01$, $w_{\text{path}} = 0.1$, $w^d_{\text{pr}} = w^d_{\text{pair}} = [1.0, 5.0]$, and $w^\theta_{\text{pr}} = w^\theta_{\text{pair}} = 10.0$. The effect of omitting individual terms is depicted in Figure 2.7.

The optimization formulation can be readily extended to second-tier objects—optimization is performed to move second-tier objects on the supporting surfaces provided by their first-tier counterparts in the same way that furniture objects move over the floor space of a room, which is regarded as the root in the hierarchy. However, second-tier objects will attach to their first-tier parents if they are not already attached when the optimization begins.

	Number of Objects	Number of Iterations	Total Time (sec)
Living Room	20	20000	22
Bedroom	24	20000	48
Restaurant	54	25000	219
Resort	30	42000	126
Factory	51	42000	262
Flower Shop	64	22000	376
Gallery	35	18000	88

Table 2.1: *Computation times are measured on a 3.33GHz Intel Xeon PC. Spatial and hierarchical relationships are extracted automatically from positive examples.*

2.6 Results

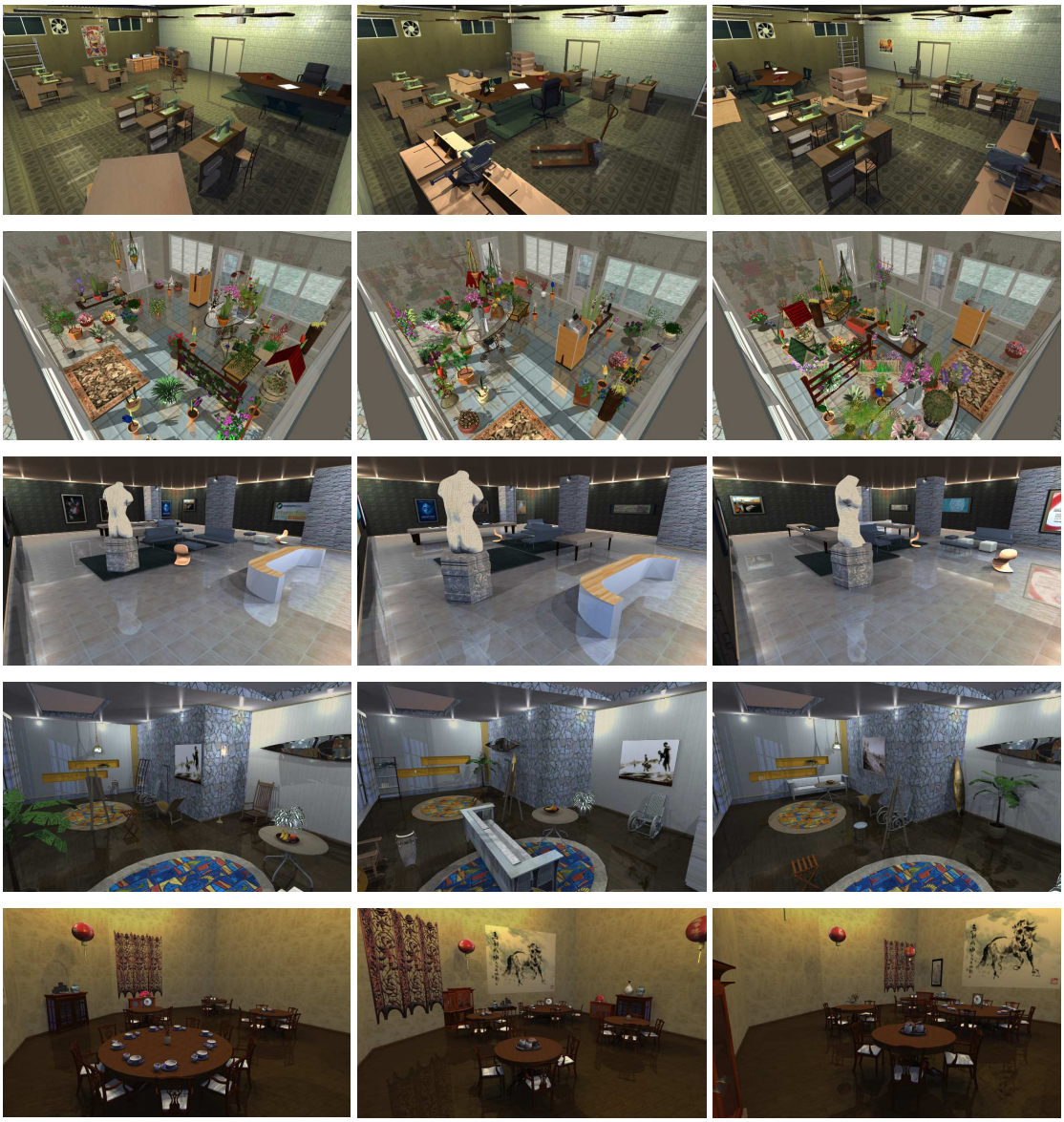
Figure 2.8 shows typical input exemplars that serve in extracting furniture relationships. For each scene, we build five exemplars which cover the most common types of furniture objects. The furniture objects used in the input exemplars for relationship extraction may differ in appearance from those used in the synthesis, although they are of the same type.

To demonstrate the efficacy of our optimization approach, we tested it on seven different scenes, the *Living Room* and *Bedroom* examples shown in earlier figures, plus the five additional scenes, *Factory*, *Flower Shop*, *Gallery*, *Resort*, and *Restaurant*, shown in Figure 2.9. Table 2.1 tabulates the computational complexity, running time, and the number of iterations in each scene. Note that the respective positions and orientations of the windows, doors, and ceiling fans are fixed and not updated during the optimization unless otherwise stated.

For each scene, we synthesized three different furniture arrangements; the same view

Living Room *Bedroom* *Factory* *Flower Shop* *Gallery* *Resort* *Restaurant*

Figure 2.8: *Typical input examples for different scenes.*



Synthesis 1 Synthesis 2 Synthesis 3

Figure 2.9: *Selected views of our synthesized results. Top to bottom: Factory, Flower Shop, Gallery, Resort, Restaurant.*

of each synthesized arrangement is shown in Figure 2.9 for comparison. Two additional views of each synthesized arrangement are included in the perceptual study, which will be detailed in the next section.

The *Factory* scene in the figure shows the efficacy of the pairwise constraint. By modifying the weights of the pairwise distance and orientation terms, different groupings of work desks and chairs are obtained. The accessibility and visibility constraints acting together prevent the door and poster from being blocked. The *Flower Shop* provides a striking example of the effect of the pathways constraint, which maintains a clear path between the doors despite the dense coverage of the remainder of the room by flowers. For this scene, we change the position of the main door in each synthesis to illustrate different path generation solutions. The accessibility constraint also prevents the cashier from being blocked. We modeled the *Gallery* scene based on an image of the Yale University Art Gallery. The scene consists of a non-rectangular room supported by numerous pillars. Our synthesis result suggests a new interior arrangement for the gallery, where optimizing visibility and accessibility helps avoid obstruction of the pictures and information counter. The *Resort* provides another example of a non-rectangular room. Using pairwise constraints between the easel and the stool, our system automatically generated a area dedicated to painting. We choose a different mean position for the sofa in Synthesis 2 so that the optimized location is farther from the wall. The *Restaurant* example illustrates the significance of the pairwise relationship on both first-tier and second-tier objects. With the use of a concentric spatial relationship between the chairs and table extracted from the exemplars, different numbers of chairs are correctly oriented and evenly distributed around their respective tables and each dish-set is near and properly oriented to its corresponding chair.

2.7 Perceptual Study

We performed a perceptual study to evaluate the realism and functionality of the furniture arrangements synthesized by our interior design system. Our null hypothesis H_0 was that users perceive no significant differences in the functionality of the synthesized arrangements relative to those produced by a human designer given the same rooms and sets of furniture objects. The alternative hypothesis H_1 was that users did perceive significant differences. Our experiment was conducted using a subjective, two-alternative, forced-choice preference approach patterned after the one reported in (Jimenez et al., 2009).

2.7.1 Participants

25 volunteer participants were recruited who were unaware of the purpose of the perceptual study. This number of participants was comparable with similar studies in which 16 users were recruited (Jagnow et al., 2008; Jimenez et al., 2009). The participants included 18 males and 7 females whose ages ranged from 20 to 60. All the subjects reported normal or corrected-to-normal vision with no color-blindness and reported that they are familiar with the indoor scenes to be tested in the study. 14 subjects reported that they did not have any expertise in interior design.

2.7.2 Data

The synthesis results shown in Figure 2.9 were compared against furniture arrangements designed by humans. To assess the significance of priors and pairwise constraints, we produced two additional synthesis results by respectively setting $w^d_{\text{pr}} = w^d_{\text{pair}} = 0$ and $w^\theta_{\text{pr}} = w^\theta_{\text{pair}} = 0$. Figure 2.10 shows selected views of the two additional synthesized examples of the five scenes. Note that the positions of objects mounted on the walls, such as paintings and posters, are fixed in the examples.

Figure 2.10: *Synthesis results obtained without enforcing a selected constraint. Left: No distance constraint; e.g., in the Living Room, the couch is not placed against the wall and, in the Factory, some work-chairs are placed far from their respective work desks. Right: No orientation constraint; e.g., in the Living Room the television is oriented at an awkward angle against the wall and, in the Factory, some work-chairs are oriented arbitrarily.*

Figure 2.11: *A screenshot used in our perceptual study. Each participant was shown 70 pairs comprising a synthesized arrangement and an arrangement created by a human designer given the same room and set of furniture objects. Left: Overhead and 2 different views of a synthesized furniture arrangement. Right: Corresponding views of a human-designed furniture arrangement.*

2.7.3 Procedure

The study was conducted in a manner similar to the traditional practice adopted in industry, where interior decorators present their design alternatives to customers and request their preference. It involved static 2D image viewing rather than 3D scene navigation so as to eliminate differences due to varying degrees of skill among the participants in using navigation software. The viewing of video was avoided because, as our preliminary experiments showed, repeated video viewing easily causes fatigue.

Figure 2.11 shows a screenshot used in our perceptual study for pairwise comparison. The left and right color plates respectively show three views of a furniture arrangement, one synthesized by our system and the other created by a human designer. Each participant viewed a total of 70 trials (5 paired comparisons $\times$ 7 scenarios $\times$ 2 trials).

Participants were encouraged to ask any questions prior to the study. After completing a consent form and questionnaire, they were given a sheet indicating the task description:

> "This test is about selecting a color plate from a pair of color plates, and there are 70 pairs in total. Each plate shows three views of a furniture arrangement. You will be shown the plates side-by-side with a grey image displayed between each evaluation.
>
> Your task in each evaluation is to select the arrangement in which you would prefer to live, stay, work, visit, etc., depending on the primary function of the room, by clicking on the color plate. You can view the test pair for an unlimited amount of time, but we suggest that you spend around 15 seconds on each set before making your selection."

The color plates were presented to each participant in a different random order. Counterbalancing was used to avoid any order bias—each paired comparison was assessed twice by each participant, where in half of the trials the synthesized arrangement is displayed as the left plate and as the right plate in the other half.

2.7.4 Outcome and Analysis

The primary goal of the experiment was to validate the quality of the furniture arrangements synthesized by our system relative to that of arrangements designed by humans. If human-designed arrangements are not clear winners over the synthetic ones, then our system may be considered successful.

The collected preference outcomes were analyzed to determine if any statistically significant trend exists. To this end, we first adopted the Chi-square nonparametric analysis technique. A one-sample Chi-square includes only one dimension, such as is the case in our perceptual study. The obtained (A_E/A_1,A_E/A_2,A_E/A_3,A_E/A_4,A_E/A_5) frequencies were compared to an expected 25/25 (50 for each comparison) result to ascertain whether this difference is significant. The Chi-square values were computed and then tested for significance. Table 2.2 tabulates the survey results. Overall, they indicate that

the furniture arrangements created by humans are *not* clearly preferred over the arrangements A_1, A_2, and A_3, when all the cost terms participate in the optimization that synthesizes the furniture arrangements. For the A_E/A_1, A_E/A_2 and A_E/A_3 pairs, among the 21 synthesized arrangements, only 3 showed a significant difference ($p < 0.05$) inasmuch as most of the participants were able to identify the human-designed arrangement in these cases.

Second, we adopted a Bayesian analysis (Gallistel, 2009; Rouder et al., 2009) to determine whether the number of participants who selected the synthesized layout was what would be expected by chance, or if there was a preference pattern. For each scene, we assumed that the participant had a probability P of picking the human-designed arrangement, and that the results of different trials of the same scene were independent of each other. Based on these assumptions, we used a binomial distribution to model the results, where the only parameter was P. Then H_0 has $P = 0.5$ and H_1 has $P = [0, 1]$. We computed the odds O on H_0 over H_1. According to (Rouder et al., 2009), $O > 3$ shows evidence favoring H_0 whereas $O < 1/3$ shows evidence favoring H_1, while other odds values are inconclusive.

Table 2.3 tabulates the odds computed. For the A_E/A_1, A_E/A_2, and A_E/A_3 pairs among the 21 synthesis results, 10 favor H_0 indicating the lack of a significant perceived difference between the furniture arrangements synthesized by our system and the human-designed arrangements, 6 favor H_1 indicating a significant difference, and 5 are inconclusive.

Orientation vs distance: Most users chose the human-designed arrangement when the distance or orientation constraint was inhibited, and it was easier for users to detect the difference when we inhibited the orientation term than when we inhibited the distance term. Omitting orientation constraints yields bad results in practice, which suggests that a greater weight can be applied in penalizing orientation deviation during

Scene	A_E/A_1		A_E/A_2		A_E/A_3		A_E/A_4		A_E/A_5	
	χ^2-value	p-value	χ^2-value	p-value	χ^2-value	p-value	χ^2-value	p-value	χ^2-value	p-value
Living Room	1.210	0.271	0.010	0.920	0.010	0.920	**5.290**	**0.021**	**10.89**	**0.001**
Bedroom	0.810	0.368	**7.290**	**0.007**	0.010	0.920	**4.410**	**0.036**	**20.09**	**0.000**
Factory	0.490	0.484	1.690	0.194	0.810	0.368	**13.69**	**0.000**	**20.25**	**0.000**
Flower Shop	0.090	0.764	**9.610**	**0.002**	**6.250**	**0.012**	0.090	0.764	**10.89**	**0.001**
Gallery	0.250	0.617	3.610	0.057	0.090	0.764	1.690	0.194	3.610	0.057
Resort	0.010	0.920	2.890	0.089	0.090	0.764	**9.610**	**0.002**	**12.25**	**0.000**
Restaurant	3.610	0.057	0.250	0.617	1.690	0.194	**8.410**	**0.004**	2.890	0.089

Table 2.2: *Chi-square analysis (degrees of freedom = 1, level of significance = 0.05). A_E, A_1, A_2, A_3 are, respectively, the example arrangement and synthesis results 1, 2, and 3 in Figure 2.8 and Figure 2.9. A_4 and A_5 are the respective synthesis results without distance and orientation considerations. Values shown in boldface indicate significant differences.*

optimization.

2.8 Summary, Discussion, and Future Work

In this chapter, we introduced a framework for the automatic synthesis of furniture layouts, avoiding manual or semi-automated interior design approaches that are impractical in graphics applications requiring full automation. We believe that our work is the first to consider in a comprehensive manner human factors, among them accessibility, visibility, pathway constraints, and so forth. We have demonstrated the effectiveness of our automated interior design approach in generating arrangements for various scenarios, and our results have been deemed by human observers to be perceptually valid in functionality compared to arrangements generated by human designers.

Although our framework espouses optimization as a means of synthesizing realistic furniture arrangements, it provides users the flexibility to control furniture placement

Scene	A_E/A_1 odds	A_E/A_2 odds	A_E/A_3 odds	A_E/A_4 odds	A_E/A_5 odds
Living Room	(1.377)	**5.506**	**5.506**	0.016	0.000
Bedroom	(2.135)	0.002	**5.506**	0.042	0.000
Factory	**3.050**	(0.818)	(2.135)	0.000	0.000
Flower Shop	**4.894**	0.000	0.005	**4.894**	0.000
Gallery	**4.020**	0.102	**4.894**	(0.818)	0.102
Resort	**5.506**	0.223	**4.894**	0.000	0.000
Restaurant	0.102	**4.020**	(0.818)	0.000	0.223

Table 2.3: *Odds on the null hypothesis H_0 over the alternative hypothesis H_1. Values shown in boldface favor H_0, indicating no significant difference; values shown in parentheses are inconclusive; other values favor H_1.*

that respects furniture functionality and interior design aesthetics. For instance, the pairwise constraint promotes the even distribution of chairs around a circular table in the *Restaurant* example, which is a typical case of radial balance or symmetry (Ching and Binggeli, 2005; Malnar and Vodvarka, 1992). Our framework also demonstrates its effectiveness in a "tight fit" scenario, where many functional groupings of furniture (e.g., work desks and chairs) are possible as in the *Factory* example, as well as in a "loose fit" scenario, where the placement is more flexible and furniture types are more diverse, as in the *Resort* example. The framework is also flexible enough to cater to specific needs related to human factors, which may be readily encoded into the accessibility and pathway terms in order to generate livable furniture arrangements. Note that we make the implicit assumption that the perimeter of a room is long enough to accommodate all the furniture objects that ought to be placed against walls. Violating this assumption may lead to local suboptima or failure cases, where in the resulting layout some furniture objects that should be positioned against walls may be placed at a distance from the nearest wall that is already occupied by another object.

Given our automatic tool for synthesizing furniture arrangements plus existing methods for synthesizing floor-plans, buildings, and cities, we can potentially create and model virtual worlds much faster and with much less human effort. The resulting interior enrichment would enhance the level-of-detail and, therefore, the realism of large-scale buildings in virtual worlds, which are becoming increasingly ubiquitous in motion pictures and interactive games.

The future extension of our framework will focus on functional issues, including consideration of interior lighting design and the acoustic qualities of a synthesized furniture configuration, as well as on subjective, aesthetic issues, among them the selection of furniture styles and colors consistent with design concepts such as balance, harmony, and emphasis (Ching and Binggeli, 2005; Malnar and Vodvarka, 1992). Furthermore, pairwise relationships between objects in exemplars should be analyzed in a more sophisticated manner from the perspective of unsupervised machine learning.

Future work can also investigate how sophisticated human factors, such as pathways and the interaction of humans with their environment, can be taken into account in the automatic generation of interior layouts. Other sophisticated factors, such as lighting and aesthetics, can also be considered in the layout optimization process. This will not only lead to higher realism of the generated layouts, but may also help in creating smart homes where the generated layouts may be employed by robots to automatically arrange the given rooms into pleasant configurations.

CHAPTER 3

Virtual Character Modeling: DressUp

3.1 Introduction

As you awaken each day, there is a simple question that you may need to answer: How should I dress today? Your wardrobe contains various kinds of clothes, such as dress shirts, dress pants, jeans, sweaters, suits, and different types of shoes. What combination of clothing will have you most appropriately dressed for the day's activities, thereby making you most visually appealing? Perhaps you would like multiple suggestions that best coordinate with the new tie that you received from your daughter as a birthday gift. The outfit selection problem also occurs in computer graphics modeling, especially in movie and game production: How should one appropriately dress a large number of human characters with an eye to functionality while avoiding visual awkwardness and repetitiveness? The manual specification of clothing is obviously tedious and it may be prohibitive on a large scale.

We demonstrate that the task at hand, of selecting appropriate subsets of clothing items from a wardrobe, can be addressed formally as a combinatorial optimization problem. Figure 1.5 shows an overview of our approach. A suitable outfit requires jointly combining a variety of clothing items to satisfy functional and certain visual criteria. We do not generally wear a pair of sandals with dress pants to the office, nor do we wear a red dress shirt with a green suit for a business meeting. In addition, to put a wardrobe into full use, we would like to explore as many good solutions as possible, so that we

can exhibit sartorial variety. A similar, but much larger-scale problem comes up with regard to online boutique websites, where shoppers can select among many clothing items. Usually it is not difficult for a shopper to locate a desired clothing item; the non-trivial question is how this clothing item should be matched in terms of style and color with other clothing items from the same or different shops or from one's wardrobe at home.

There is no single universal rule that satisfies both the relevant *functional* and *visual* criteria. People generally categorize outfits into *dress codes*, which represent different functionalities. These can range from strictly regulated ones such as *White Tie*, suitable for formal events, to relatively unrestricted ones such as *Casual*, suitable for many everyday activities. Without restriction, one can define a particular clothing requirement for an event and consider it a dress code. Different religions, societies, and cultural practices adhere to different dress codes; for example, in some formal occasions, Scottish men wear a kilt, a form of dress not commonly worn by men elsewhere. The visual criteria involve numerous factors, from human body attributes such as skin color, eye color, hair color, and body shape, which are model-specific, to aspects of the clothing items such as the clothing color, cutting, style, and fabric texture. The rules vary across national and cultural boundaries and historic timelines. Even when one has satisfied all the applicable rules, whether one is dressed in a visually pleasing manner is still a rather subjective question.

In tackling the clothes matching problem, we enforce functional and visual criteria through the two most important factors—*dress code* and *color*. While color is an obvious visual factor (Jackson, 1987; Zyla, 2010; Nicholson, 2003), to a certain extent it is also related to functionality, which in turn depends on culture. For example, people in China usually dress in red for festivals and in white for funerals. On the other hand, the dress code is a broader guideline that pertains more to the combination of clothing items. Some dress codes also have strict requirement for the colors of particular items,

but how different colors coordinate is not their main concern.

Given a specific dress code favoring various combinations of clothing items and a human body, our outfit synthesis framework optimizes the color compatibility between the human body and the suggested items in order to realize both the functional and visual criteria. We employ four of the most common dress codes *Sportswear*, *Casual*, *Business-Casual*, and *Business*, which cover the main functionalities of daily life in much of the world. These dress codes are encoded in our system within a probabilistic framework, via a Bayesian network. The Bayesian network is trained on real image data and it associates any particular clothing item combination with an observed probability distribution under any specific dress code. Additional dress codes and other matching criteria can be trained and included in the same manner. As is common practice in the fashion industry (Jackson and Lulow, 1984; Jackson, 1987; Henderson and Henshaw, 2008), our system classifies the color type of the human subject as 'warm' or 'cool' based on his or her skin, hair, and eye colors. This is automatically accomplished by a classifier that is pre-trained on a database of images of people. After assigning the user color type, our system will suggest a preferable color palette for the subject and this color palette will serve as a soft constraint during the optimization, which automatically searches for clothing items guided by the dress code while satisfying color compatibility criteria subject to the suggested color palette.

In summary, outfit selection is a common everyday problem; however, the nature of this problem is very broad and it involves a considerable amount of visual and social factors that can be implicit and abstract. Our main contributions are as follows:

1. The introduction of a novel topic area to computer graphics and a first attempt to tackle the automatic outfit synthesis problem through a data-driven approach.

2. The encoding of implicit, probabilistic clothing matching relationships on real-world data through Bayesian Networks that support conditional queries and in-

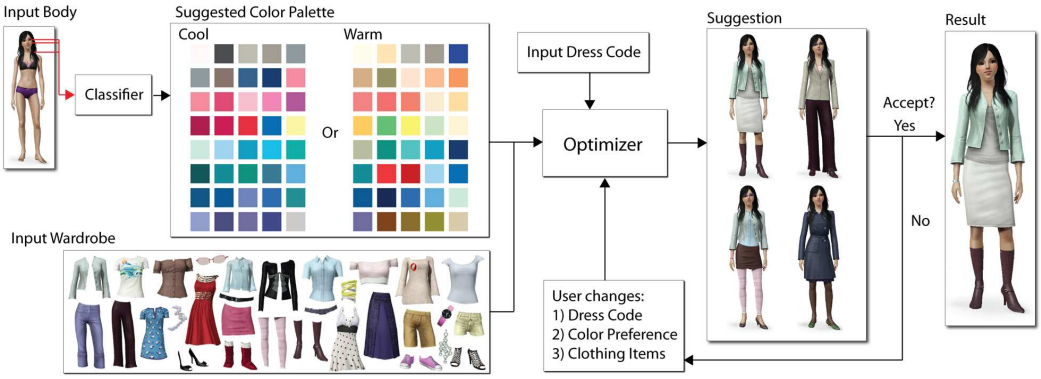


Figure 3.1: *Our optimization framework. Inputs include body color attributes, an input dress code, and a wardrobe of clothing items. The optimizer generates optimal suggestions according to cost terms defined by the dress code, the suggested color palette, and color compatibility. In response to a user's change of style, color preference, or specification of particular clothing items, our system automatically synthesizes new outfits.*

corporate a Support Vector Machine classifier of body color tone that applies subjective evaluation criteria common in the fashion industry.

3. The formulation of outfit synthesis as an optimization problem that takes into account the style and color compatibility of clothing combinations, and that is flexible and easily extensible through the modification of the Bayesian networks and cost terms of the formulation.

4. The application of our novel approach in different practical scenarios; e.g., as an outfit advisor, as a suggestion engine in shopping/boutique websites, or as part of the character modeling engine for games/virtual world applications.

5. The validation of the efficacy of our approach through a gender-specific perceptual study.

3.2 Related Work

We will begin by discussing related work on clothing and virtual character modeling in computer graphics and then review other relevant work on fashion and color.

3.2.1 Clothing in Computer Graphics

Modeling, animating, and rendering visually realistic clothing has been an area of interest in computer graphics for decades (Terzopoulos et al., 1987; Terzopoulos and Fleischer, 1988; Provot, 1995; Baraff and Witkin, 1998) and it has received much attention in recent years in movies and games especially for dressing large numbers of human characters. Researchers have been putting significant effort into the realistic modeling (Kaldor et al., 2008; Volino et al., 2009; Kavan et al., 2011; Wang et al., 2011b; Umetani et al., 2011; Guan et al., 2012) and/or animation (Kaldor et al., 2010; de Aguiar et al., 2010; Wang et al., 2010; Feng et al., 2010) of clothing, and their efforts have enabled computer animated clothing to blend seamlessly with the clothing worn by real actors.

Tools are now available to help artists interactively design virtual garments, which is adequate for highly-detailed, small scale production, e.g., for motion pictures. However, manual approaches become too tedious on a large scale, such as when there is a need to clothe numerous virtual humans in a virtual city. While our work does not concern the physically-realistic deformation of clothes meshes over virtual bodies, we are not aware of any research on automatic outfit synthesis in computer graphics; i.e., given a set of clothing items and a human body model, automatically suggest a clothing combination for a general or particular scenario.

Tsujita et al. (2010) conducted a user survey that pointed out the difficulty that people have in selecting suitable outfits from their wardrobe. They proposed the simple heuristic of not repeating outfits on consecutive days, and installed a camera system

in a user's wardrobe that can acquire and upload pictures of clothes to the internet so that the user can solicit outfit selection advice from friends. Our data-driven approach captures clothing combination "advice" implicit in example fashion images, but we automate the suggestion and synthesis process.

3.2.2 Human Modeling

Human characters are an important aspect of creating virtual worlds (McDonnell et al., 2006; Dobbyn et al., 2006; McDonnell et al., 2008, 2009; O'Sullivan, 2009). While realistic human animation and rendering can be critical (Tecchia et al., 2002), variety in human appearance is equally important when considering a large group of people. Ulicny et al. (2004) describe a system that enables the interactive creation of virtual humans with variety. The importance of appearance variation in realistic human perception is nicely summarized with an extensive perceptual study in the work of O'Sullivan et al. (2008; 2009; 2009).

For the most part, existing human modeling software requires substantial manual intervention. However, researchers have proposed approaches to mass-produce various characters by automatically modifying the texture, color, and geometry of different body parts in order to create crowds that exhibit some natural variation (McDonnell et al., 2006; Dobbyn et al., 2006; Thalmann et al., 2007). However, the goal of prior approaches is to enhance the realism of the crowd as a whole, rather than specific concern as to whether any individual in the crowd is dressed properly or in a visually pleasing manner. The lack of a fast, highly automated approach to this problem limits variation in the style of human characters, leading, in particular, to repetitive sartorial patterns that greatly reduce realism.

Casual Sportswear Business-Casual Business

Figure 3.2: *Example images of dress codes from Google Images.*

3.2.3 Color in Clothing

Recently, techniques for combining colors in a scene to make it look, say, "harmonious" or "peaceful" have been gaining interest (Cohen-Or et al., 2006; O'Donovan et al., 2011). Color coordination is a core consideration in clothes matching (Zyla, 2010; Gilchrist, 2011; Nicholson, 2003). Fashion and make-up professionals usually regard color coordination as person-specific, mostly dependent on the person's intrinsic color tones, in particular, the skin, eye, and hair colors (Jackson, 1987). A basic approach is to first classify individuals as suited to a 'warm' or 'cool' color palette, from which they should choose the colors for their clothes. As there is no definitive classification rule, subjective evaluation is usually performed, and a common test is to have observers evaluate whether the individual looks best wearing gold or silver accessories, respectively (Jackson, 1987). There are other variations of classification which are more subtle and abstract—e.g., in accordance with the season (Jackson and Lulow, 1984), or according to "light/deep/clear/soft" (Henderson and Henshaw, 2008). However, the basic principle is still the same—suggesting a color palette for clothing items based on the classification result.

3.2.4 Dress Codes

A dress code is a set of rules governing what garments may be worn together and in what setting. Such rules are commonly agreed upon among people, usually dependent on events and occasions. Common dress codes nowadays include *Sportswear*, *Casual*, *Business-Casual*, *Business*, and *Formal*. Figure 3.2 shows typical example images. Some of the aforementioned dress codes also constrain the color of the items; for example, *Business* clothing tends to be darker, while there is not much restriction in *Casual* or *Sportswear*. Pattern, fabric weight, and texture are also relevant to the dress code (Gilchrist, 2011).

The dress code is important in governing the functionality of the clothing (Schoeffler and Gale, 1973; Fischer-Mirkin, 1995; Flusser, 2002; Sondag, 2011; Gilchrist, 2011). However, the main objective of a dress code is to convey a message through the combination of various clothing items. For example, dressing without a tie for a job interview will convey a less formal and more relaxed impression, while donning a suit, dress shirt, and tie to the beach will create an unusual scene. Without a strict definition, the perception of some dress codes can be ambiguous and personal; e.g., some *Business Casual* outfits may be regarded as *Business* or *Casual*.

3.3 Data-driven Approaches

Figure 3.1 shows an overview of our optimization approach for automatic outfit suggestion. The inputs comprise a human body model, a specific dress code, and a predefined wardrobe. The output is one or more optimized outfit suggestions. Before presenting the technical details of our optimization framework and developing the objective function in the next section, let us consider the information required to define our cost functions.

Example node	Example state
Dress Code	Casual, Sportswear, Business-Casual, Business
Chest 1	t-shirt, dress shirt, sleeveless
Chest 2	tank, sweater, vest, long t-shirt
Chest 3	suit jacket, jacket, hoodie, open sweater
Hip	jeans, shorts, dress pants
Foot	slippers, dress shoes, boots
Neck	necklace, scarf, tie, bow tie

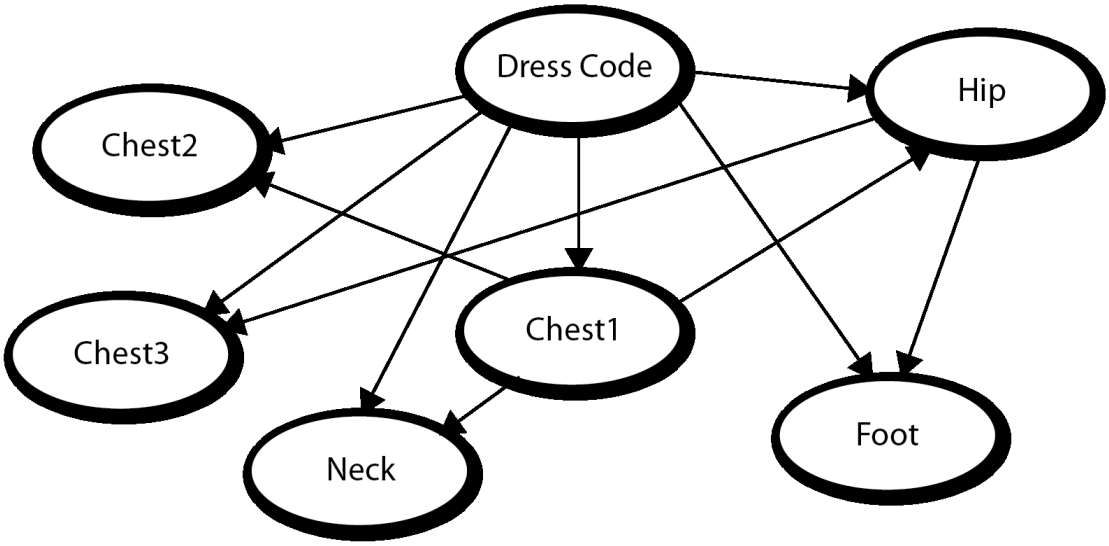


Figure 3.3: *Representing distributions of clothing items combinations with a Bayesian network. Top: A table showing the major example nodes with some of their states. Bottom: A part of an example Bayesian network for men, trained using labeled fashion images. Refer to our supplementary materials for the complete graphs of the Bayesian networks. Note that each node, except for the dress code node, has a state 'none'.*

There are two preprocessing steps before the optimization process—encoding the clothing relationship and classifying the color tone of the subject body. We must quantify the relationships among different clothing items so that we can define compatible costs among them; for example, what should or should not be worn based on the selected dress code and some already selected clothing items. As we have discussed, the dress code involves various factors and can change from time to time. For example, a dress shirt usually goes with dress shoes if dress pants are worn, but there could be more flexibility if jeans are worn.

An expedient way to generate outfit variety is to randomly select among predefined rules to combine clothing items. However, the question of how to define the rules, which is critical to synthesis quality, is susceptible to subjective bias. It is difficult to consider all possible combinations, and the rules quickly become intractable to maintain as the types of clothing items grow. Restricting to a small subset of possible outfits may avoid awkward synthesis, but it will result in limited variety and common artifacts such as "repeated" characters that are noticeable in virtual scenes. The lack of conditional query support has also prohibited the use of such approaches in practical scenarios (e.g., shopping websites).

One possibility to encoding various relationships and defining compatible costs between clothing items is to adopt a data-driven approach based on observational data. Data driven approaches have recently proven to be successful in problems involving abstract semantic relationships; for example, in architectural design, furniture layout, assembly-based 3D modeling, and color compatibility applications (Merrell et al., 2010; Yu et al., 2011; Chaudhuri et al., 2011a; O'Donovan et al., 2011). Since our goal is to match different clothing items in a sensible manner, and with natural variety conforming to real world observations, a probabilistic machine learning framework trained by real world data is appropriate to encode the matching cost, such that the higher the probability of a particular clothes combination, the lower is its matching cost.

An important issue in establishing the probabilistic relationships between different clothing items relates to their *conditional dependencies*. For example, the frequent occurrence in the data of a jeans-sandals combination and a dress shirt-jeans combination could lead to a dress shirt-jeans-sandals combination style being generated, which should have very low likelihood. Therefore, simply encoding the observed probability of a clothing item and any combination between it with other items is prone to error.

Probabilistic graphical models, in particular, Bayesian networks, are an elegant and efficient choice (Pearl, 1988; Koller and Friedman, 2009) for learning the implicit relationships among different clothing items consistent with their conditional dependencies. Our trained Bayesian networks effectively encode the probability distributions in the space of clothing combinations. An important feature of the Bayesian network is its ability to support conditional query, which is frequently needed in clothes matching. The values of any subset of a clothing combination can be fixed and the probabilities of the remaining clothes can be calculated. For example, given the *Business-Casual* dress code, one may constrain the upper body to be clothed in a t-shirt and blazer and query the probability of the lower body being clothed in jeans according to the trained distribution. This allows better flexibility to recommend clothing items under different user-specified conditions or scenarios.

3.3.1 Bayesian Networks for Clothing Relationships

To make the scope of our problem tractable, we train separate Bayesian networks for men's and women's wear and exclude color from the training process. In our current system, we train these networks on four dress codes: *Sportswear*, *Casual*, *Business-Casual*, and *Business*. Figure 3.3 shows part of the Bayesian network for men. The network for women is similar, with differences in some of the node states; e.g., having state *dress* in node *Chest 1*. The complete networks can be found in our supplementary materials.

The nodes of the Bayesian networks correspond to different body regions on which a clothing item can be worn, and each node state represents the type of clothing item being worn. For example, the node *foot* has states *dress shoes*, *slippers*, *boots*, and so on. Except for the node *dress code*, each node also has a state *none*, which is used when the node does not carry any clothing item; e.g., *foot* = *none* when no shoe is worn. While state choices can be easily modified to suit specific domain needs, as a general case, we follow common classification in boutique websites such as "H&M" and "eBay".

To enable us to handle more complicated situations where there is layering of clothing, we permit a body region to be represented by multiple nodes that correspond to multiple clothing layers. For example, the chest has nodes *Chest 1*, *Chest 2*, and *Chest 3*, with *Chest 1* corresponding to the innermost layer (e.g., a t-shirt), *Chest 2* to the middle layer (e.g., a vest), and *Chest 3* to the outer layer (e.g., a jacket).

Usually a reasonable quantity of input training data is required. For example, 120 architecture programs were used to train the networks in (Merrell et al., 2010). In our case, we downloaded around 3000 images for the four dress codes for men and women from Google Images[1]. Since some of the downloaded images are not useful, and determining whether the images belong to the dress code is a subjective process, we hired three fashion school students to manually label the attributes of each instance in the network, who used their judgment to disregard inappropriate images. In total, around 2000 labeled data sets for men and women were used to train the Bayesian networks. Labeling each image took about 15–20 sec, and the whole labeling process took 4 hours. Example training images for *Business* and *Sportswear* are shown in Figure 3.4. Variety arises when multiple item combinations occur under the same dress code. The images, labeling program, and labeled data are included in the supplementary materials.

[1]Example keywords we used for the image search: 'Casual wear for men', 'Sportswear for men', 'Business-Casual wear for men', Business wear for men', and similarly for women.

Figure 3.4: *The top row shows typical example images from Google Images. The bottom row shows the corresponding labeled data used for Bayesian network training. Note that some images may have occluded items (e.g., shoes are not visible in the second image), but partially labeled data is still usable in training the Bayesian network.*

Figure 3.5: *Top: An example fashion image and its corresponding 5-color palette. (Image courtesy of COLOURLovers.)*

Two attributes should be assigned to two different nodes if the corresponding two clothing items can coexist; e.g., shirt and suit jacket. Otherwise, they should be put under the same node; e.g., sandals and lace-up shoes, since it is not possible to wear both at the same time. The important point here is to capture the relationships among different clothing items and their conditional dependencies. Using the labeled data, we learnt the Bayesian network structures for men and women respectively, by the Tree Augmented Naive Bayes method (Friedman et al., 1997) which maximizes conditional mutual information between attributes. The conditional probability tables are trained by the Expectation-Maximization algorithm, which can learn the probabilities even if some training data are only partially labeled. Notice that other methods such as maximum likelihood estimation could also be adopted. We found that the results generated using the learnt networks faithfully reproduce our human perceptual requirement for the four dress codes considered. Table 3.1 shows some example queries based on the probabilities captured.

3.3.2 Body Color Tone Classifier

After encoding the probabilistic relationships among the clothing items, the next step is to inform the optimization process of a color guide. It is a common practice in fashion to first classify a person's body color tone and then suggest a suitable color palette for matching clothes for them. There are multiple ways for color tone classification such as subjective evaluation tests (Jackson, 1987) or by "guidelines" or "rules"[2]. However, as shown in Table 3.2, the classification "guidelines" can be very obscure and cumbersome, arguably uninterpretable by a general user. There is obviously no unique one-to-one correspondence between body color attributes and color tone classification for users to follow.

To this end, we train a classifier to predict the body color tone of a target person consistent with human preferences. This has two major advantages. First, we integrate subjective evaluation tests commonly adopted in fashion (Jackson, 1987) into a machine learning framework by capturing the subjective evaluation from a number of people. Second, after the classifier is trained, it is intuitive at the user's end—a user simply inputs his/her body colors (e.g., by a few clicks on his/her face photo) and automatically obtains a color tone classification result, instead of interpreting obscure descriptions.

We acquired from Google Images a training dataset comprising 1000 facial images after discarding images with strong illumination effect, including both males and females. For each image, we manually extracted the RGB values of the eye, skin, and two locations in the hair (to encode hair color variation). In accordance with common practice in the fashion industry, we matched each image with a set of silver accessories and then with a set of gold accessories, from which a test subject was asked to choose which one they preferred, thus indicating 'cool' and 'warm' color tone, respectively. We recruited 40 volunteer participants, including 20 males and 20 females whose ages ranged from

[2]http://www.askandyaboutclothes.com/Tutorials/CindyBuschColorAnalysis.htm

20 to 60, to evaluate the 1000 face images. Evaluation took about 5-10 sec per image.

We trained a Support Vector Machine (SVM) classifier (Cristianini and Shawe-Taylor, 2000) and performed cross-validation by randomly choosing 900 data for training and 100 data for testing, achieving a prediction rate of about 77%. Given a previously unseen human body model with specific skin, hair, and eye colors, the trained classifier predicts the body color tone, thereby recommending either a 'cool' or 'warm' color palette to be used in the optimization. Each suggested color palette consists of 40 colors, as in (Jackson, 1987). While the evaluation is by its nature subjective and ambiguous, we find that in general people with brownish/reddish hair and brownish/greenish eyes are usually classified as 'warm', whereas those with light-colored hair and dark/bluish eyes are classified as 'cool'. The labeled training data and labeling program are included in the supplementary materials.

3.3.3 Color Compatibility Predictor

Figure 3.5 shows example images from fashion websites such as "Wear Palettes" and "COLOURLovers", which are usually accompanied by a representative 5-color palette that supports the color matching idea. Akin to this practice, at each iteration of the optimization, our optimizer extracts a representative 5-color palette from an outfit and evaluates the color compatibility of the palette based on the regression model from (O'Donovan et al., 2011), which is trained by a large number of user-rated color palettes. The trained regression model can take a 5-color palette as input, and predict a user preference rating (see (O'Donovan et al., 2011) for the details of the training dataset, prediction result evaluation, and analysis).

3.4 Outfit Optimization

In performing the optimization phase, our system exploits the trained Bayesian networks, body color tone classifier and color compatibility predictor described in the above sections. Given a human model, a wardrobe of clothing items and a dress code as inputs, our system suggests multiple outfits whose colors are adjusted desirably such that they are compatible to each other guided by the color palette.

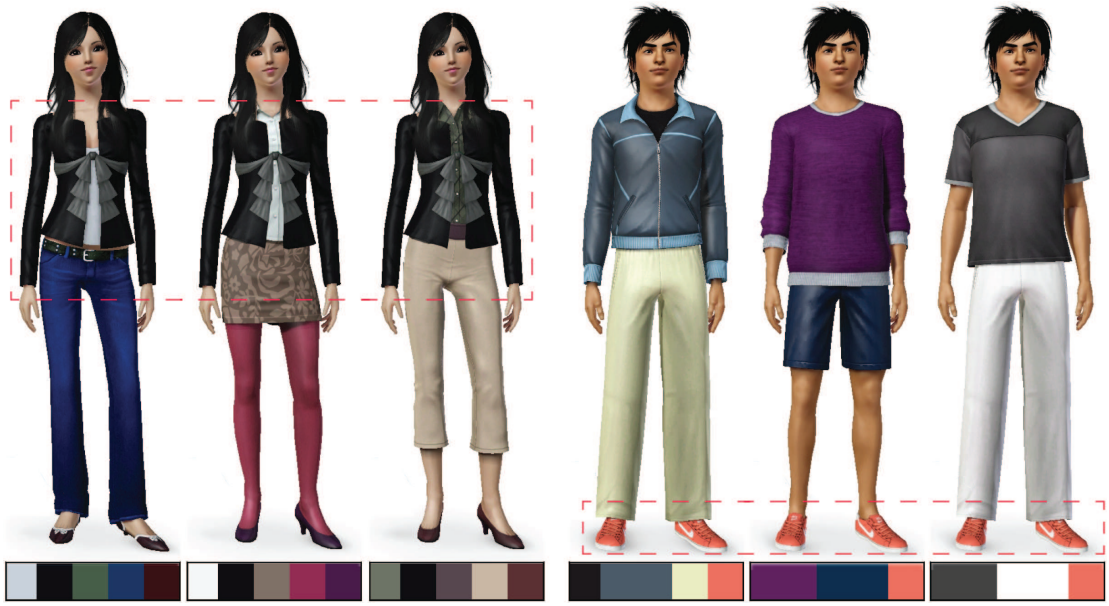
To achieve our goals we must solve a combinatorial optimization problem. Denoting the wardrobe as W, which is a set containing all clothing items, the state of our system is a subset of W, which we refer to as an *outfit*, $\phi = \{\theta_i | i = 1, \ldots, T\}$, where each $\theta_i = (c_i, n_i, s_i)$ is a 5-value tuple representing a selected clothing item. The term $c_i = (r_i, g_i, b_i)$ contains the RGB values of the clothing item, which are quantized from 0 to 255, n_i is the node of the Bayesian network to which this clothing item belongs, and s_i is the corresponding node state. For example, $n_i =$ *foot* and $s_i =$ *dress shoes* means that selecting the clothing item θ_i corresponds to setting the node *foot* of the Bayesian network to state *dress shoes*.

Note that the total number T of selected clothing items is a variable that can be changed. Thus, the dimension of the input space is a variable. Our goal is to obtain an outfit ϕ that minimizes an objective function described in the next section.

3.4.1 Objective Function

We now describe the cost terms constituting our overall objective function.

Style Cost: In order to obtain the matching cost between different clothing items, at each iteration, we must determine every node state of the Bayesian network. Suppose the network has N nodes (excluding the root node *dress code*) denoted by $x_1, \ldots, x_N$.



(a) 3.44 (b) 3.35 (c) 3.35 (d) 3.57 (e) 3.48 (f) 3.50

Figure 3.6: *Results with specific clothing items being fixed. (a)–(c) fixed black sweater. (d)–(e) fixed orange shoes. Color ratings are shown at the bottom.*

Given an outfit ϕ, every node x_k is instantiated to state $S(x_k)$ by:

$$S(x_k) = \begin{cases} s_i & x_k = n_i \\ \text{none} & x_k \neq n_i, \forall i \end{cases} \tag{3.1}$$

The style cost term has two components $C^{\text{indv}}_{\text{style}}$ and $C^{\text{joint}}_{\text{style}}$. Given *dress code* $= d \in \{$*Casual*, *Sportswear*, *Business-Casual*, *Business*$\}$, then $C^{\text{indv}}_{\text{style}}$ encodes the conditional probability of each clothing item. It guides the optimizer by penalizing the selection of clothing items that do not fit dress code d. On the other hand, $C^{\text{joint}}_{\text{style}}$ defines the conditional joint probability of the clothing item combination:

$$C^{\text{indv}}_{\text{style}}(\phi) = 1 - \frac{1}{N}\sum_k P(S(x_k)|\textit{dress code} = d). \tag{3.2}$$

$$C^{\text{joint}}_{\text{style}}(\phi) = 1 - P(S(x_1), \ldots, S(x_N)|\textit{dress code} = d). \tag{3.3}$$

To evaluate these costs, our framework makes queries over the Bayesian network to

provide the conditional and conditional joint probabilities in (3.2) and (3.3).[3] In case the user fixes one or multiple node states, the fixed node states will become the given conditions. Figure 3.6 shows two examples with specific items being fixed.

Color Rating Cost: Similar to the convention in fashion images, we use a 5-color palette to represent a clothing combination ϕ which comprises T selected clothing items, based on a heuristic:

1. Each clothing item is represented by the color of its largest surface area.

2. Select 5 colors:

   If $T = 5$, select colors from all clothing items.

   If $T > 5$, sort clothing items by their surface areas. Select colors from the 5 clothing items with the largest surface areas.

   If $T < 5$, sort clothing items by their surface areas. Duplicate colors of the $5 - T$ clothing items with the largest surface areas. Select the $5 - T$ duplicated colors and the colors of the T clothing items.

3. Sort the 5 selected colors according to their physical position on the body, from top to bottom.

In practice, we assume the outfit comprises at least 2 clothing items, i.e., $T >= 2$. Denoting these ordered 5 colors as $\lambda_1, \ldots, \lambda_5$, this is the 5-color palette representing outfit ϕ. The color compatibility cost is

$$C_{\text{color}}^{\text{comp}}(\phi) = 1 - [R(\lambda_1, \ldots, \lambda_5) - 1]/4. \tag{3.4}$$

In (3.4), $R \in [1, 5]$ is the regression model from (O'Donovan et al., 2011), which

[3] To illustrate the effectiveness of $C_{\text{style}}^{\text{indv}}$, suppose the dress code is *Business*, the initialized outfit is "shirt, jeans, slippers" and another outfit "shirt, dress pants, slippers" is sampled. Although $C_{\text{style}}^{\text{joint}}$ will evaluate both outfits as unlikely, $C_{\text{style}}^{\text{indv}}$ will favor the latter, hence effectively guiding the synthesis towards a *Business* outfit.

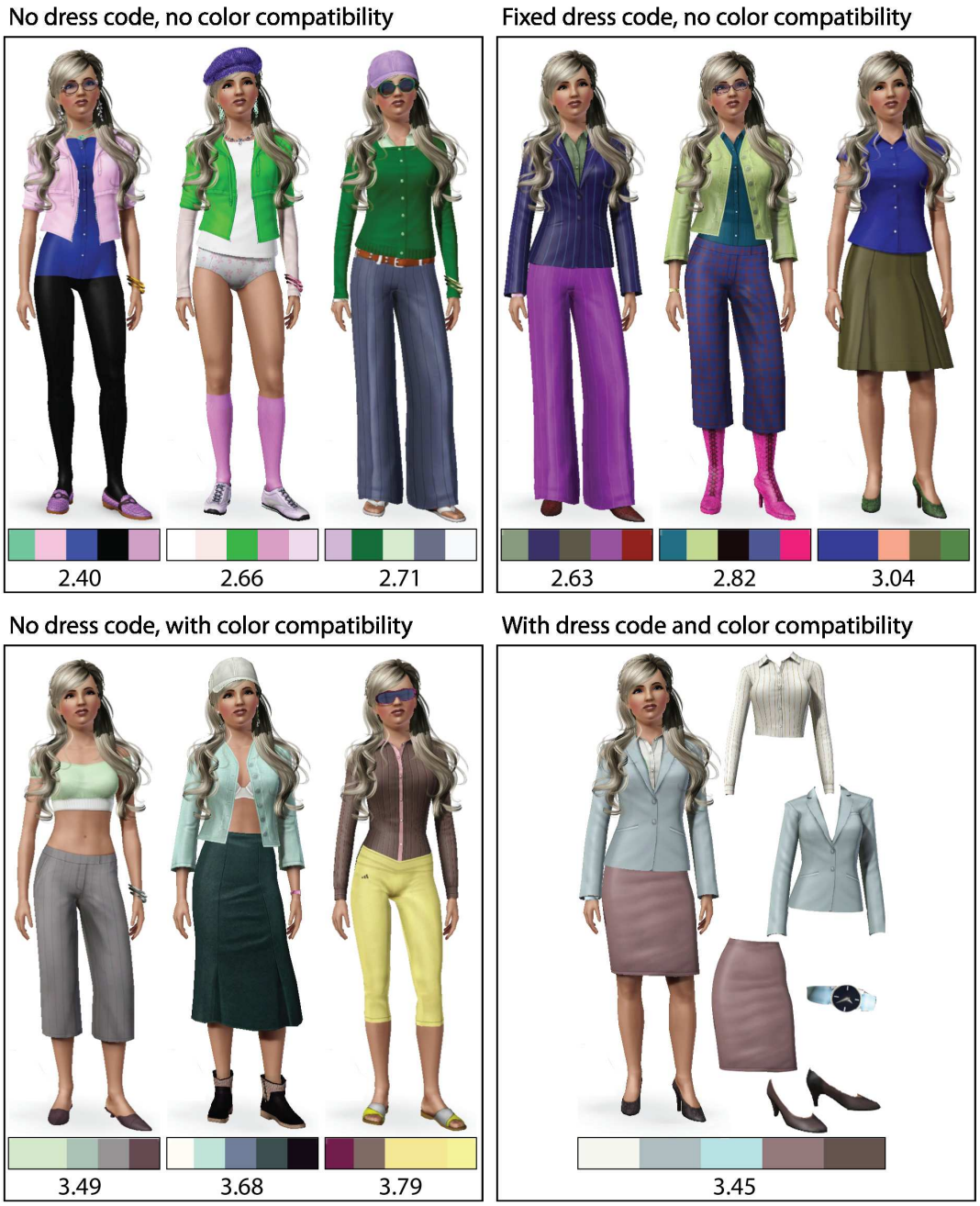


Figure 3.7: *Effects of omitting individual cost terms. Top left: No dress code and no color optimization. Top right: Fixed dress code, no color optimization. Bottom left: No dress code but with color optimization. Bottom right: Fixed dress code with color optimization. The color ratings are shown at the bottom.*

Figure 3.8: *Results with two different color palettes.*

predicts the user rating of a 5-color palette, with a higher rating implying higher user preference. The cost in (3.4) is normalized accordingly.

Color Palette Cost: To keep the clothing item colors close to the suggested color palette, the system calculates the distance of each clothing item's color c_i to each color c_j in the suggested color palette, and penalizes it if the nearest distance is larger than a threshold h. The color palette cost term is defined as

$$C_{\text{color}}^{\text{palette}}(\phi) = \frac{1}{T\sqrt{3Z^2}} \sum_i \max(\min_j \|c_i - c_j\| - h, 0) \tag{3.5}$$

,where $Z = 255$ is the maximum quantized RGB value.

The total cost function is the weighted sum of the above cost terms:

$$C(\phi) = w_{\text{style}}^{\text{indv}} C_{\text{style}}^{\text{indv}}(\phi) + w_{\text{style}}^{\text{joint}} C_{\text{style}}^{\text{joint}}(\phi) + \\ w_{\text{color}}^{\text{comp}} C_{\text{color}}^{\text{comp}}(\phi) + w_{\text{color}}^{\text{palette}} C_{\text{color}}^{\text{palette}}(\phi) \tag{3.6}$$

The w coefficients determine the relative weighting between the cost terms; in practice, we set $w_{\text{style}}^{\text{indv}} = 1.0$, $w_{\text{style}}^{\text{joint}} = [5.0, 10.0]$, $w_{\text{color}}^{\text{comp}} = 1.0$, and $w_{\text{color}}^{\text{palette}} = 1.0$. Figure 3.7 shows the effect of omitting the style and color cost terms. Figure 3.8 illustrates the effect of using different color palettes.

3.4.2 Reversible Jump Markov Chain Monte Carlo

Since our optimization problem is combinatorial and the number of combination items can vary (e.g., a jacket can be added or removed), it is difficult to define a closed-form solution. In fact, as in the real world, we wish to obtain multiple optimal solutions (outfits) from the same wardrobe instead of a single global optimum. This motivates the generation of candidate solutions by sampling a density function defined over the space of possible outfits. The density function is defined using idealized analytical formulations. Sampling is performed using a Markov chain Monte Carlo sampler. Figure 3.9 shows multiple optimal outfits generated.

One of the difficulties for our optimization problem is that its dimensionality may change; i.e., the number of clothing items may be altered during the optimization process. To deal with this complication, we adopt the Reversible Jump MCMC (RJMCMC) framework (Green, 1995) which can be considered a generalization of the original Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970). RJMCMC works by supplementing the parameter-changing diffusion moves of MH with an additional set of dimension-altering jump moves, which allow the chain to move between subspaces of different dimension. RJMCMC has been successfully applied to other graphics and vision problems such as procedural modeling (Talton et al., 2011) and image segmentation (Tu and Zhu, 2002).

3.4.3 Annealing

To efficiently explore the solution space, we apply the simulated annealing technique (Schneider and Kirkpatrick, 2006) in the optimization process. We define a Boltzmann-like objective function:

$$f(\phi) = \exp(-\beta C(\phi)), \tag{3.7}$$

where β is a constant inversely proportional to the temperature of the annealing process. At the beginning of optimization, β is set to a low value, equivalent to setting a high temperature, which allows the sampler to more aggressively explore the solution space. Then β is gradually increased throughout the optimization. Near the end, β attains a large value, equivalent to setting the temperature near zero, thereby allowing the sampler to refine the solution. Figure 3.10 shows the iterative optimization process.

3.4.4 Proposed Move

We adopt the *dimension matching* strategy to allow reversible jumps across subspaces of different dimension or within the same subspace. At each iteration of our optimization, a move $m' \in \{m_a, m_r, m_s, m_m\}$ is chosen with probability $p_{m'}$. Associated with the move is a move-specific proposal distribution $q_{m'}(.)$, which is a function of an auxiliary variable U'. As move m' is chosen, a sample of the auxiliary variable U' is drawn from $q_{m'}(U')$, which modifies the current outfit ϕ to a proposed new outfit ϕ' by a deterministic function $\phi' = h(\phi, U')$. We also need to compute the reverse move m, which reverts ϕ' back to ϕ, by sampling U from $q_m(U)$ such that $\phi = h^*(\phi', U)$. The proposed outfit ϕ' is then accepted with probability

$$\alpha(\phi'|\phi) = \min(1, \frac{p_m}{p_{m'}} \frac{q_m(U)}{q_{m'}(U')} \left| \frac{\partial(\phi', U)}{\partial(\phi, U')} \right| \frac{f(\phi')}{f(\phi)}), \tag{3.8}$$

where $|\partial(\phi', U)/\partial(\phi, U')|$ is the Jacobian of the diffeomorphism from (ϕ, U') to (ϕ', U). Defining $\phi' = h(\phi, U') = U'$ and $\phi = h^*(\phi', U) = U$, the Jacobian is unity (Godsill, 2001). For further detail on RJMCMC, refer to (Green, 2003; Andrieu et al., 2003).

Based on the RJMCMC formulation, we follow the natural strategy to define the jump moves as adding/removing a clothing item to/from the outfit, which induce a dimension change, and diffusion moves as swapping items or modifying an item's color, which involve no dimension change, as follows:

Figure 3.9: *Multiple outfit recommendations. The dressed models and the corresponding items. Top:* Sportswear. *Bottom:* Business-Casual. *The recommendations from left to right are arranged in descending matching cost value.*

Initialization Iteration 20 Iteration 50 Iteration 100 Iteration 180 Iteration 250

Figure 3.10: *Outfit optimization from a random initial configuration (left) for dress code* Business. *As the optimization process proceeds, the clothing items are iteratively updated until the outfit converges to the desired clothing item combination with coordinated color.*

Figure 3.11: *Outfit synthesis results for the models, associated items, and the 5-color palette. From top to bottom: "Mag" (Female, Cool), "Eddie" (Male, Cool), "Ce" (Female, Warm), "Jacen" (Male, Warm). From left to right:* Casual, Sportswear, Business-Casual *and* Business.

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

Figure 3.12: *Close-up views of populated virtual scenes with and without outfit consideration. (a) Outfits synthesized randomly; (b)–(c) Outfits synthesized under dress code* Business*; (d) Outfits re-synthesized after changing dress code to* Sportswear*. An unnatural appearance clearly results in the absence of a proper dress code.*

Adding an Item (m_a): Randomly pick an available clothing item θ_j from wardrobe W and add it to outfit ϕ, so that $\phi' = \phi \bigcup \{\theta_j\}$.

Removing an Item (m_r): Randomly remove a selected clothing item θ_i from outfit ϕ, so that $\phi' = \phi \setminus \{\theta_i\}$.

Swapping Items (m_s): Randomly pick a selected clothing item θ_i from outfit ϕ, and swap it with an available clothing item θ_j from wardrobe W, so that $\phi' = \phi \setminus \{\theta_i\} \bigcup \{\theta_j\}$.

Modifying an Item Color (m_m): Randomly pick a selected clothing item θ_i from outfit ϕ and change its color c_i. Hence, θ_i is updated as: $\theta_i' = (c_i + \delta c_i, n_i, s_i)$, where $\delta c_i \sim [\mathcal{N}(0, \sigma_c^2)\, \mathcal{N}(0, \sigma_c^2)\, \mathcal{N}(0, \sigma_c^2)]^T$ and, with $\mathcal{N}(\mu, \sigma^2) = (2\pi\sigma^2)^{-1/2} e^{-(x-\mu)^2/2\sigma^2}$, a Gaussian distribution of mean μ and variance σ^2. The variance σ_c^2, which determines the average magnitude of the change, is proportional to the temperature.

The acceptance probabilities of the proposed RJMCMC moves are:
Adding an Item (m_a):

$$\alpha(\phi'|\phi) = \min(1, \frac{p_r}{p_a} \frac{\frac{1}{|\phi'|}}{\frac{1}{|W \setminus \phi|}} \frac{f(\phi')}{f(\phi)}) \tag{3.9}$$

$$= \min(1, \frac{p_r}{p_a} \frac{|W \setminus \phi|}{|\phi'|} \frac{f(\phi')}{f(\phi)}). \tag{3.10}$$

Removing an Item (m_r):

$$\alpha(\phi'|\phi) = \min(1, \frac{p_a}{p_r} \frac{\frac{1}{|W \setminus \phi'|}}{\frac{1}{|\phi|}} \frac{f(\phi')}{f(\phi)}) \tag{3.11}$$

$$= \min(1, \frac{p_a}{p_r} \frac{|\phi|}{|W \setminus \phi'|} \frac{f(\phi')}{f(\phi)}). \tag{3.12}$$

Swapping Items (m_s):

$$\alpha(\phi'|\phi) = \min(1, \frac{p_s}{p_s} \frac{\frac{1}{|\phi'|}\frac{1}{|W|}}{\frac{1}{|\phi|}\frac{1}{|W|}} \frac{f(\phi')}{f(\phi)}) \tag{3.13}$$

$$= \min(1, \frac{f(\phi')}{f(\phi)}). \tag{3.14}$$

Modifying an Item Color (m_m):

$$\alpha(\phi'|\phi) = \min(1, \frac{p_m}{p_m} \frac{p(\theta_i|\theta_i')}{p(\theta_i'|\theta_i)} \frac{f(\phi')}{f(\phi)}) \tag{3.15}$$

$$= \min(1, \frac{f(\phi')}{f(\phi)}). \tag{3.16}$$

In our implementation, we simply set the prior distribution uniformly over the moves as $p_a = p_r = p_s = p_m = 0.25$.

3.5 Results

To demonstrate the efficacy of our optimization approach, we tested it on six different virtual human models, three males and three females. Figure 3.11 depicts two males and two females. The remaining characters were used in our perceptual study and can be found in the supplementary material. For the males, “Thor” has white skin and dark brown hair, “Eddie” has yellow skin and black hair, and “Jacen” has black skin and black hair. For the females, “Fiona” has white skin and blonde hair, “Mag” has yellow skin and black hair, and “Ce” has dark brown skin and black hair.

We synthesized all four test dress codes *Sportswear*, *Casual*, *Business-Casual* and *Business* for all the models. We optimized the male and female model outfits using the

Bayesian networks learned for males and females, respectively. The clothing items are also segregated into male and female wardrobes. Each wardrobe contains about 10 clothing items for each of the 40 states in the Bayesian network, so there are about 400 clothing items in total. We simply used a budget of 250 optimization iterations for each outfit synthesis, which takes about 1-2 second per synthesis on a 3.33GHz Intel Xeon PC.

The final optimized outfits with the corresponding selected items are shown in Figure 3.11. We also show the corresponding 5-color palette alongside with the items. Mag and Eddie are classified as 'cool' and a 'cool' color palette was assigned to them prior to the optimization. Meanwhile Ce and Jacen are classified as 'warm'. The 'cool' and 'warm' color palettes are shown in Figure 3.1. For all the generated results, the color ratings are greater than 3.3.

The *dress code* as the root node determines the style of synthesis; i.e. what clothing items should be chosen and how they should combine. For example, in the 3rd row showing the synthesis for Ce, the same sweater is chosen for *Casual* and *Business*. However, the sweater is worn alone in *Casual*, but with a suit jacket in *Business*.

When we designed our Bayesian networks, we defined more than one node for the chest to permit the coexistence of different items. Several generated results reflect this property, which is important for creating variation. It happens more often for the dress code *Business*; for example, Eddie in the 2nd row wears a dress shirt, a vest, and a suit jacket for his upper body outfit, a combination which is occasionally observed in the *Business* training data.

Our outfit optimization can lead to two potential applications:

Outfit Suggestion Engine: The outfit suggestions can readily assist shoppers in boutique websites or fitting rooms, in which case the clothing items are those available in

Figure 3.13: *Populating virtual scenes. Our approach can automatically suggest appropriate outfits to a large number of virtual characters. Dress codes* Sportswear *and* Casual *were used in accordance with the virtual beach scene.*

the store; or it can be used as a personal outfit advisor, in which case the clothing items are those available in the user's wardrobe. The support of efficient, arbitrary probabilistic queries can handle scenarios commonly encountered in the clothes matching process. For example, conditional queries allow one to fix one or multiple clothing items and ask for multiple matching suggestions. Refer to Figure 3.6 for two examples. One can also change the preferred color palette, after which the optimizer will update the suggestion accordingly, as shown in Figure 3.8. As a personal outfit advisor, given a dress code, it can automatically suggest many decent outfits out of the user's wardrobe, thereby making full use of it. Refer to Figure 3.9 for two examples.

Virtual Character Modeling: Our approach is also useful for dressing human-like characters in large-scale virtual worlds, in which case the artist can specify dress codes and allow the computer to synthesize coordinated clothing combinations for each char-

acter in a fully automated manner. This can be readily incorporated on top of character modeling engines in gaming applications, which commonly support automatic clothes meshing on virtual characters,[4] but lack support for reasoning about the many possible outfits out of the massive amount of clothing items available.

Figure 3.12 shows virtual scenes with and without outfit consideration. One can easily see that the scene appears unnatural if the characters are not properly dressed; e.g., donning a suit jacket, or wearing a dress in a gym, or dressing in sportswear in the office. Figure 3.13 shows a beach scene populated by approximately 100 virtual characters automatically dressed up in *Sportswear* and *Casual* dress codes. With our optimization, the characters are appropriately dressed in multiple ways to create variety suitable to the scene.

While we demonstrated our approach based on the four dress codes that are common nowadays, our framework offers the flexibility to cope with specific clothing styles matching a theme. An interesting example is for a massively-multiplayer online game featuring the Medieval Fantasy, in which case the node states can be replaced by medieval clothes and specific nodes such as "weapon" may be added. In this case, training examples may be collected directly from the player-created game characters, and our trained framework can be used to provide outfit suggestions in the character modeling engine used by new players, or for the automatic, realistic synthesis and dressing of non-player characters.

3.6 Perceptual Study

We performed a perceptual study to evaluate the functional and visual appearance of our outfit synthesis framework. Since comparisons of outfits are inherently subjective, one

[4]Examples include Playstation Home, XBox 360 Avatars (http://marketplace.xbox.com/en-US/AvatarMarketplace), Second Life, etc.

Figure 3.14: *Example images in Experiment 2 of the perceptual study. Left: Outfits synthesized with the corresponding dress codes. Right: Outfits synthesized without dress code consideration. Note that all syntheses considered the color cost terms.*

possible way is to evaluate our synthesis results against comparable results produced by human fashion designers. However, assessing metrics and performing pairwise comparisons is very difficult when there are significant differences, and they may not lead to meaningful conclusions. For example, a particular subject may be fond of some particular skirt and be biased in favor of women wearing this skirt.

The goal of our system is to synthesize visually reasonable or pleasing outfits under certain dress codes. To evaluate the efficacy of our approach, we must demonstrate that the clothing items enforce the selected dress code and that their colors are nicely coordinated. Since color coordination was extensively evaluated in (O'Donovan et al., 2011) and crowd perception as such was studied comprehensively in (O'Sullivan, 2009), our perceptual study was focused on whether the matched clothes are functionally sound individually. We attempted to verify the following two conditions, by two experiments: First, a classification experiment to testify the outfit recommendations that our system produces successfully reflect the dress code and, hence, validate our Bayesian network training. Second, a discrimination experiment to verify that the incorporated dress code yields a benefit over outfit synthesis results obtained in its absence.

Similar to those of other authors (Jagnow et al., 2008; Jimenez et al., 2009; Yeung et al., 2011a; Yu et al., 2011), our experiments were conducted using a subjective, five-alternative/two-alternative, forced-choice preference approach. In Experiment 1, our null hypothesis H_0 was that users cannot recognize the dress code of the syntheses for each category; i.e., recognition rate is at chance level. In Experiment 2, our null hypothesis H_0 was that users show no preference among the syntheses with and without dress code consideration.

Participants: 32 volunteer participants were recruited who were unaware of the purpose of the perceptual study. This number of participants was comparable with similar studies in which 16 users were recruited (Jagnow et al., 2008; Jimenez et al., 2009). The participants included 16 males and 16 females whose ages ranged from 20 to 60.

All the subjects reported normal or corrected-to-normal vision with no color-blindness and reported that they are familiar with the dress codes to be tested in the study. 29 subjects reported that they did not have any expertise in fashion design.

Data: We picked 4 virtual models to cover both genders: Thor and Jacen are male, Fiona and Mag are female. For each virtual model, we synthesized 20 outfits (5 per dress code) with the complete objective function, and 20 outfits with an objective function lacking the style cost term. Figure 3.11 depicts example matching results with their associated items used in the user study. For the pairwise comparison, examples are shown in Figure 3.14. With multiple outfits per dress code we can create variety in the comparisons. The images used in perceptual study are listed for visual inspection in the supplementary materials.

Procedure: The study was conducted in two experiments. Participants were encouraged to ask any question prior to the study. After completing a consent form and questionnaire, they were given a sheet detailing the task descriptions.

Experiment 1 (Classification): The main goal was to test whether our generated results reflect the corresponding dress code faithfully and, hence, verify our Bayesian network encoding. To achieve this, we asked the subject whether the synthesized clothing combinations fall into any of our encoded dress codes:

> "This experiment involves selecting a dress code from an image of a dressed model. There are 80 images.
>
> Your task in each evaluation is to select one of the following dress codes which you feel best describes the outfit shown in the image: *Sportswear*, *Casual*, *Business-Casual*, *Business*, and *Other* if the image does not match any of the previous four. You can view the test image for an unlimited amount of time, but we suggest that you spend around 15 seconds on each image before making your selection."

Experiment 2 (Discrimination): The main goal was to evaluate if incorporating the style cost term really shows a significant preference on the functionality of the outfit compared to outfits synthesized without consideration of a dress code:

> "This experiment involves selecting a dressed model from a pair of images, and there are 160 pairs in total. You will be shown the images side-by-side with a grey image displayed between each evaluation.
>
> Your task in each evaluation is to select the model based on their outfit in which you would prefer to dress for a particular occasion which is depicted in the top of the image pairs: *Casual*, *Sportswear*, *Business-Casual*, or *Business*. You can view the test pair for an unlimited amount of time, but we suggest that you spend around 15 seconds on each pair before making your selection."

Each participant viewed a total of 160 trials (4 models $\times$ 4 dress codes $\times$ 5 pairs $\times$ 2 trials). Each pair comprises a full objective result and a result randomly chosen among those synthesized without considering the style cost term. The pairs were presented to each participant in a different random order. Counterbalancing was used to avoid any order bias—each paired comparison was assessed twice by each participant: In half of the trials the full objective result was displayed on the left side and in the other half on the right.

Results and Analysis: Figure 3.15 shows the correct recognition rates of Experiment 1. We display the results by gender of the participants versus the gender of the syntheses. Overall, the correct recognition rates are: *Casual* (83.125%), *Sportswear* (66.875%), *Business-Casual* (67.969%), *Business* (76.25%). The detailed recognition rates tabulated by gender can be found in the supplementary materials.

Figure 3.15 also shows some interesting observations. While all correct recognition rates were significantly above chance, *Sportswear* and *Business-Casual* have lower

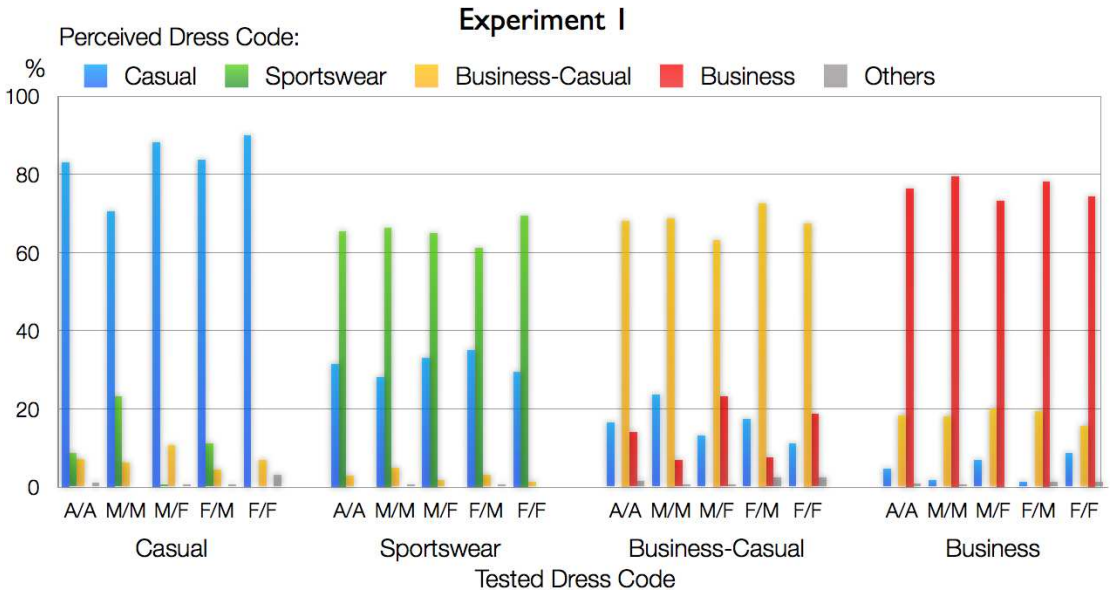


Figure 3.15: *Recognition rates of Experiment 1: Perceived dress code versus tested dress code. "A/A": All participants perceiving all syntheses. "M/F": Male participants perceiving female syntheses. Similar for "M/F", "F/M" and "F/F". All recognition rates are significantly above chance level.*

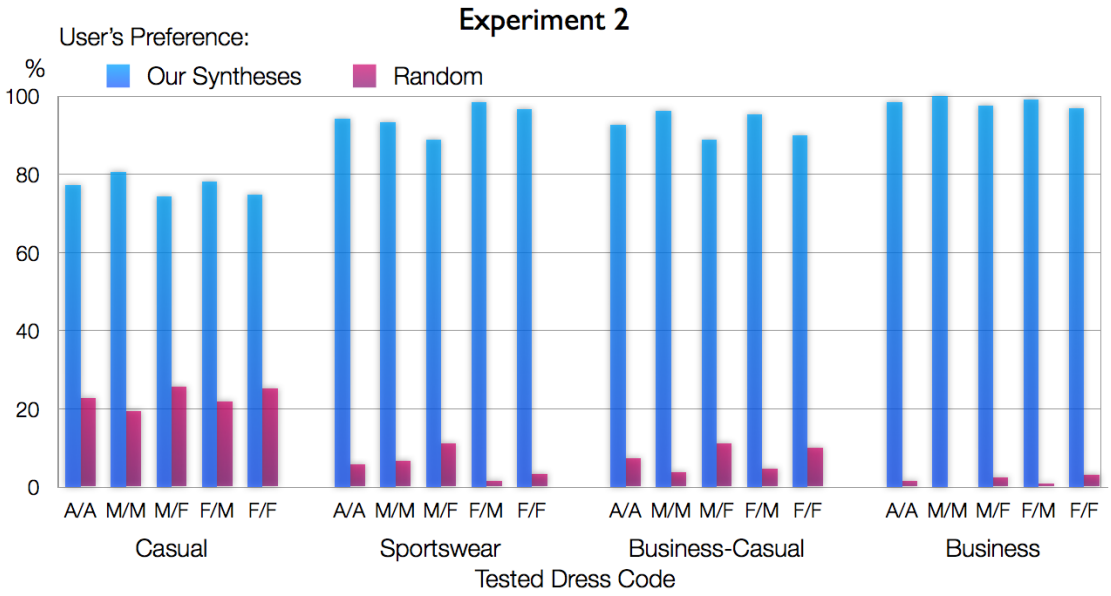


Figure 3.16: *User's preference of Experiment 2: Our syntheses versus random syntheses. The rates of picking our syntheses are significantly above chance level.*

recognition rates. A certain portion of *Sportswear* was perceived as *Casual*, while a certain portion of *Business-Casual* was perceived as *Business* and *Casual*, respectively. This is probably because in reality, the perception of different dress codes can be ambiguous and may overlap; e.g., some people may regard a subset of *Sportswear* and a subset of *Business-Casual* also as belonging to *Casual*, which tends to be more frequently chosen as a result and received higher recognition rates. This also accounts for the recognition rates of *Business* that have minor portions perceived as *Business Casual*, which if added up together should give rates over 90% for both genders.

With respect to gender difference, we note that men's *Business* tend to be more definitive than women's *Business*, with slightly higher correct recognition rates on men's *Business* syntheses and less men's *Business-Casual* syntheses being perceived as *Business*. On the contrary, men's *Casual* tend to have more overlap with *Sportswear* perceptually. The *Casual* plot shows a certain portion of men's *Casual* being perceived as *Sportswear*, while this is rarely the case for women's *Casual*. Finally, we note that male and female participants tend to give similar response trends in classification.

Figure 3.16 depicts the results of Experiment 2 by comparing the rates of choosing our synthesized outfit and random syntheses. In all the cases our syntheses are much more preferable than random syntheses. Notice that the relatively lower recognition rates on dress code *Casual*, which is not surprising due to its less restrictive nature. To ascertain that our results are significant, we performed t-tests against chance in both experiments. Figure 3.17 summarizes the p-values. In all cases, we have p-values less than 0.00001, which are very small. Therefore, we reject the null hypothesis H_0 in both experiments. For Experiment 1, this concludes that subjects can correctly recognize the dress code of the syntheses as one of the 4 encoded dress codes. For Experiment 2, this concludes that subjects also prefer the syntheses that include dress code consideration.

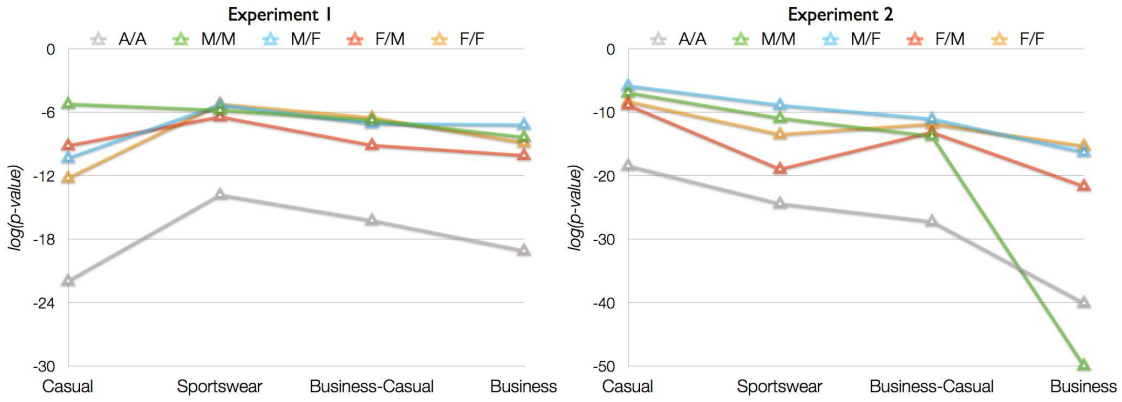


Figure 3.17: *Results of t-tests against chance for Experiment 1 (left) and Experiment 2 (right) shown as log(p-value). Notations are the same as in Figure 3.15. Test for A/A has* $d.f. = 31$. *Other tests have* $d.f. = 15$. *All tests have log(p-value)* < -5 *which is equivalent to p-value* < 0.00001.

3.7 Summary, Discussion, and Future Work

In this chapter, we introduced an automated framework for outfit synthesis, which is a highly practical topic both in daily life and computer graphics. Our approach optimizes outfits in a way similar to real-world situations. The body color tone classifier automates the classification pre-process in fashion practices, avoiding cumbersome, obscure, manual classification. From the user's perspective, our framework is highly intuitive in practical use. On the one hand, if one fixes item colors and permits only addition, removal, or swapping moves during optimization, one is mimicking the scenario of a fixed wardrobe, and the optimizer jointly considers style and color when synthesizing outfits out of the available clothing items. On the other hand, if one permits the changing of certain clothing item colors, this is similar to buying new clothes, and it is particularly useful for populating virtual worlds with characters that exhibit realistic sartorial variety.

Currently, we have incorporated four different dress codes into our outfit synthesis system, but our learning and synthesis framework is flexible enough to accommodate ad-

ditional criteria such as season, texture pattern, clothing shape, age, body proportion, or even the association of outfits with multiple dress codes during training. For simplicity, we assumed that each clothing item is represented by its dominant color. More sophisticated representations, such as representing each clothing item with an arbitrary number of colors (e.g., one color for a plain shirt and two colors for a checkered shirt), can be readily handled by our RJMCMC formulation, which flexibly allows varying the number of dimensions. On the other hand, the color palette suggestion is motivated from the fashion literature, and it is easy for users to change according to their own preference; e.g., using a more colorful palette for a festive occasion, or replacing the color palette with one tailor-made by a fashion professional for a specific client, or trained by large-scale commercial datasets. Our framework is novel inasmuch as it formulates the seemingly abstract fashion matching problem as a combinatorial optimization problem in which style and color are jointly considered. Speeding up the automated outfit synthesis process for crowds through parallel computation, and a comprehensive human perceptual study on different outfits are additional interesting avenues for future work.

(a) *Conditional probabilities*:	
P(bracelet | *dress code* = Business)	0.0691
P(bracelet | *dress code* = Business Casual)	0.2456
P(bracelet | *dress code* = Casual)	0.4591
P(bracelet | *dress code* = Sportswear)	0.1023
(b) *Joint conditional probabilities*:	
P(dress | *dress code* = Business Casual)	0.1706
P(dress | *dress code* = Business Casual, *Foot* = legging)	0.6710
P(sweater | *dress code* = Business Casual)	0.0504
P(sweater | *dress code* = Business Casual, *Chest 1* = dress shirt)	0.1095
(c) *Conditional joint probabilities*:	
P(sport pants, sport shoes, tank | *dress code* = Sportswear)	0.1181
P(sport pants, sport shoes, long t-shirt | *dress code* = Sportswear)	0.0353
P(shorts, sport shoes, sweater | *dress code* = Sportswear)	0.0007

Table 3.1: *Example probabilistic queries supported by Bayesian Networks. (a) Simple queries by conditional probabilities. (b) Increased joint conditional probabilities effectively reflect common matching styles such as "legging, dress", "dress shirt, sweater" from the training data. (c) Conditional joint probabilities for more complicated combinations. The major advantage of Bayesian Networks is that they support instant, arbitrary queries.*

Body Attribute Description	Classification
(a) teal blue eyes, dirty blonde hair, peach skin	Autumn
(b) dark brown eyes, oyster white hair, ivory skin	Autumn
(c) black brown eyes, white hair, beige skin	Winter

Table 3.2: *Example classification guidelines for four-season body color tone. To obtain a classification result, the user must first determine his/her body attributes according to the description. The description and classification can be obscure to interpret: (a) and (b) have different descriptions, but are classified as the same, while (b) and (c) have similar descriptions, but are classified as different. (Courtesy of AskAndyAbout-Clothes.com; refer to website for the full table.)*

CHAPTER 4

Interactive Scene Modeling: The Clutterbrush

4.1 Introduction

Visual realism is one of the defining goals of computer graphics. While realism is often approached in terms of rendering fidelity, it is also a modeling problem (Newell and Blinn, 1977). Realism calls for creating synthetic environments that display a convincing level of detail, on par with typical real-world scenes.

Consider the kitchens in Figure 4.1. There is stark contrast between the real-world scenes and the synthetic ones. Without the odds and ends that populate real-world scenes, the synthetic environments appear eerily barren, devoid of traces of life. This lack of small-scale detail objects is characteristic of synthetic environments and commonly undermines their realism. As observed by Xu et al. (2002), “computer graphics scenes are often unrealistically simple or overly tidy."

The difficulty of populating a scene with detail objects is due in part to the large number of such objects that can appear in a typical scene. A realistic indoor environment can easily contain over a hundred detail items: books, stationery, and computing equipment in an office; dinnerware, cookware, and food items in a kitchen; clothes, bedding, and decor in a bedroom. Searching for each individual item becomes tedious at this scale. The mere identification of items that could fit well at a particular place in the scene becomes a chore when it needs to be repeated hundreds of times.

Real-world kitchens Synthetic kitchens

Figure 4.1: *Real-world (left) and synthetic kitchens (right). The images of real-world scenes were obtained by searching Flickr Creative Commons with the keyword "kitchen"; the synthetic scenes were obtained by searching the Trimble 3D Warehouse with the same keyword. The synthetic scenes appear barren in comparison.*

To make the enhancement of synthetic indoor scenes with detail objects faster and easier, we have developed the Clutterbrush, an interactive tool that assists modelers in enriching their scenes. When the user points to a location in the scene, the Clutterbrush suggests appropriate detail items for that location. The user thus retains control over the content of the scene, but the laborious search for appropriate clutter objects is automated. The Clutterbrush identifies appropriate clutter items, which are presented to the user. A scene can thus be rapidly populated by repeatedly pointing the Clutterbrush and picking a suggested object until a sufficient level of detail is reached. As an example, Figure 1.6 shows a virtual living room before and after Clutterbrushing.

In order to suggest appropriate objects, the Clutterbrush must be trained on data, since manually codifying the dependencies between hundreds of types of clutter objects, furniture objects, and scenes would be impractical. The need for data on which the Clutterbrush can be trained presents us with a circular dependency: since creating realistically detailed scenes using current modeling tools is tedious and time-consuming, there are few such scenes in the public domain. We overcome this difficulty by training the Clut-

terbrush on real-world imagery. Specifically, the Clutterbrush is trained on a dataset of images of real-world indoor scenes, annotated with object types and support relations. This yields a scalable training pipeline that transfers the semantics of real-world scenes to 3D modeling.

We evaluated the utility of the Clutterbrush in a set of experiments with participants who modeled different types of indoor scenes, and independent evaluators who assessed relative scene realism. The experimental results demonstrate that the suggestions presented by the Clutterbrush speed up modeling time, and that Clutterbrushing enhances the realism of indoor scenes.

4.2 Related Work

The difficulty of modeling realistic indoor scenes has been recognized for some time. Early work focused on assisted placement and arrangement of objects. Bukowski and Séquin (1995) developed a system for assisted placement of furniture and other objects in indoor scenes. The system helped align objects by associating them to each other: keeping bookshelves against walls, for example, or tables on floors. The associations between different types of objects were specified manually. Xu et al. (2002) described a system for automatic placement of a large number of objects in an indoor scene. The system is guided by a set of manually specified constraints, which describe relationships between different types of objects. Following this line of work, Merrell et al. (2011b) described a system that assists furniture arrangement in indoor scenes. The system relies on manually encoded interior design guidelines. Relationships between furniture types are again specified manually. The reliance on manual constraint specification limits the scalability of these approaches. For example, in order to handle a new scene type, such as a classroom or a hotel lobby, all relevant relationships between all types of objects would need to be specified in detail. In contrast, our pipeline is designed to

deal with the numerous small and diverse detail items that populate indoor scenes. As we demonstrate, it is highly scalable.

Yu et al. (2011) described a procedural furniture arrangement system that is trained on data. For each type of scene, five example scenes were manually constructed to illustrate relationships between furniture types. This limits the scalability of approach in dealing with new scene types and with a large number of clutter objects. Fisher and Hanrahan (2010) described a context-aware search engine for 3D models that is trained on data. Given an incomplete indoor scene and a bounding box specified by the user, the system retrieves objects that fit the scene and the bounding box, based on pairwise relationships and individual object descriptors extracted from the data. Fisher and Hanrahan assume that the training data consists of complete 3D scenes and rely on fine-grained geometric properties of and spatial relationships between objects in the dataset. In our setting, this assumption is unrealistic due to the dearth of realistically cluttered scenes in publicly available 3D scene repositories. For this reason, our approach is trained on images of real-world scenes.

Fisher et al. (2011) describe an approach to comparing synthetic 3D scenes based on geometric properties of and spatial relationships between objects in the scenes. As with the prior work of Fisher and Hanrahan, this relies on the availability of a dataset of complete synthetic scenes that demonstrate proper relationships between objects. In contrast, our approach decouples the dataset of clutter objects from the dataset of images that demonstrate relationships between objects. This allows our two datasets to scale independently: additional clutter instances broaden the range of objects available to the modeler and additional training images broaden the system's knowledge about object relationships. Clutter instances and real-world training images can be added independently.

Yeh et al. (2012) describe an optimization approach that procedurally arranges a large number of objects in a scene. This approach relies on manually specified relationships

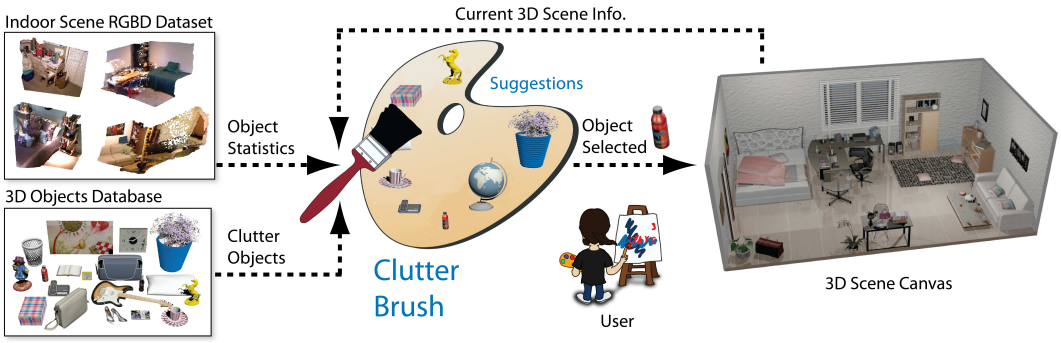


Figure 4.2: *An overview of our approach.*

between objects. Fisher et al. (2012) present an approach that synthesizes complete object arrangements that are similar to given input arrangements. Xu et al. (2013) present a sketch-based modeling system for indoor scenes. These approaches are not trained on real-world data and do not support interactive enhancement of scene detail.

Our interface builds on the idea of suggestive interfaces, proposed by Igarashi and Hughes (2001). In such interfaces, the authoring tool reasons about operations that the user is likely to undertake and presents suggestions that anticipate the user's choices. The user can pick one of the suggestions or disregard them. In this way, content creation is made faster and easier. Chaudhuri and collaborators (Chaudhuri and Koltun, 2010; Chaudhuri et al., 2011b) describe data-driven suggestive interfaces for modeling individual 3D objects. Lee et al. (2011) develop a suggestive interface for freedhand drawing. Chajdas et al. (2010) describe a suggestive interface for assigning textures to surfaces in 3D scenes and Jain et al. (2012) present an interface for assigning materials to parts of 3D objects. Umetani et al. (2012) develop a suggestive interface for designing physically valid furniture and Bao et al. (2013b) describe an interface for exploring building layouts. Our work develops a suggestive interface for enhancing the level of detail of indoor scenes.

4.3 System Overview

Figure 4.2 shows an overview of our approach.

4.3.1 Interaction

We assume that the user begins with a room in which basic furniture has been arranged. This can be achieved with any of the existing approaches to furniture layout (Bukowski and Séquin, 1995; Xu et al., 2002; Merrell et al., 2011b; Yu et al., 2011). The user then enhances the scene with the Clutterbrush. When the user points to a location in the scene, the Clutterbrush presents the types of objects that are most likely to appear there. The likelihood computation is based on the type of the scene (kitchen, bedroom, office, etc.), the type of the supporting object (floor, wall, desk, bed, etc.), and the clutter items that are already present on this supporter. This computation uses statistics extracted from a dataset of real-world images of indoor scenes. The dataset is described in the second part of this section.

The most likely clutter types are presented in order, as shown in Figure 4.3. For each clutter type, a randomly chosen object of this type is shown in situ, in the context of the scene. The user can quickly place the presented object in the scene by double-clicking on it. To see additional instances of the presented type (e.g., other bottles, other jars, other cups, etc.), the user can click on the corresponding object once: this brings up a secondary panel that shows additional instances. This hierarchical selection scheme affords greater flexibility in choosing a particular object instance of the suggested type, for example out of personal preference for the object's color or style.

Throughout the modeling process, the user can also freely reposition and delete objects, after which the new scene setting will automatically be taken into account and appropriate new suggestions will be presented with the next stroke of the Clutterbrush. The

Figure 4.3: *Interaction with the Clutterbrush. Left: the user brushes the counter and the Clutterbrush presents objects that fit the brushed location. Right: the same scene after one of the suggested objects was selected.*

interface is further demonstrated in the supplementary video.

4.3.2 Data

The suggestive functionality of the Clutterbrush relies on the computation of the likelihood of a given clutter type appearing at the brushed location. This likelihood computation is described in detail in Section 4.4. It relies on conditional probabilities that relate types and quantities of clutter objects to each other and to the type of the scene and the type of the supporting object on which the clutter resides. In our approach, these probabilities are estimated by extracting empirical statistics from real-world observations of cluttered, lived-in indoor scenes. For the sake of scalability, we do not require these observations to be detailed or complete. Each observation needs to provide only the following information: the type of the scene (e.g., kitchen), a list of observed supporting objects (e.g., floor, wall, counter), and a list of clutter objects on each supporting

object (e.g., bottle on counter, another bottle on counter, cup on counter, clock on wall, garbage bin on floor).

As a source of observations of real-world scenes, we use the NYU Depth dataset (Silberman et al., 2012). This is a dataset of 1449 RGBD images of 464 indoor scenes from 3 cities. The scenes appear in their natural messy condition. To provide data for training and evaluating scene understanding algorithms, Silberman et al. used Amazon Mechanical Turk to annotate individual objects in the images as well as support relationships between objects. We use this information for training the Clutterbrush. Although the NYU dataset contains both color and range images, we do not use the range data. The range data was also not used for obtaining the annotations: the Amazon Mechanical Turk system was provided only with the color images. A dataset of analogously annotated color images can thus be used to provide additional training data for the Clutterbrush.

A sample image from the NYU dataset is shown in Figure 4.4. We trained the Clutterbrush on five scene types from the dataset: bedrooms, kitchens, living rooms, offices, and classrooms. For each scene type, we identified the most common clutter types (e.g., book, bottle, box, etc.). As shown in Table 4.1, several dozen clutter types were identified for each scene type. Some of the object types annotated in the NYU dataset were not used due to lack of available 3D models. For each clutter type that was used, we collected 4 or 5 object instances from the Trimble 3D Warehouse and from other online sources of free 3D models. In total, our 3D object database contains 176 clutter object types and 786 object instances.

4.4 Suggestion Generation

When the user brushes a location in the scene, the Clutterbrush needs to determine which clutter types to suggest and in what order. In essence, it needs to answer the

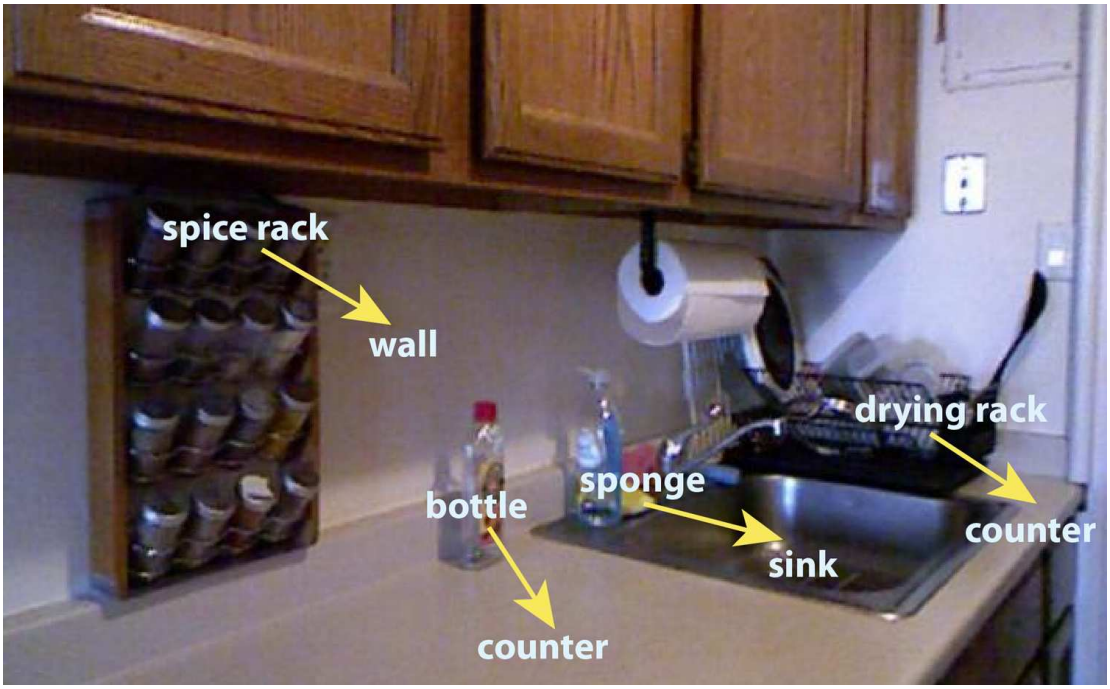


Figure 4.4: *An image from the NYU dataset, with some of the annotated support relations.*

following question: "If there was an additional clutter item at this location, what would it be?" This is done by evaluating the likelihood that a given clutter type would appear at the identified location. The types with the highest likelihood are presented to the user, in order.

The Clutterbrush evaluates the likelihood of a given clutter type conditioned on the type of the scene, the type of the supporting object, and the clutter objects that are already present on this supporter. Specifically, the brush evaluates the following probability for each clutter type Y^i:

$$P\big(x = Y^i | w, s, \{n^j\}\big), \tag{4.1}$$

where x is the type of the hypothetical new clutter item, w is the scene type ('kitchen', 'bedroom', etc.), s is the type of the supporting object ('floor', 'desk', 'wall', etc.), and n^j is the number of clutter objects of type Y^j that are already present on s, for each type Y^j for which $n^j > 0$. Note that j may be equal to i, for example when the

model considers the probability of adding a second monitor on a desk. To evaluate the probability (4.1) in terms of empirical statistics we can extract from the training data, we use the naive Bayes model. Specifically,

$$
\begin{aligned}
&P\big(x = Y^i | w, s, \{n^j\}\big) \\
&\quad = \frac{P(x = Y^i) P(w, s, \{n^j\} | x = Y^i)}{P(w, s, \{n^j\})} && (4.2) \\
&\quad \propto P(x = Y^i) P(w, s, \{n^j\} | x = Y^i) && (4.3) \\
&\quad = P(x = Y^i) P(w | x = Y^i) P(s | x = Y^i) \prod_j P(n^j | x = Y^i). && (4.4)
\end{aligned}
$$

(4.2) follows by Bayes' theorem. (4.3) follows because we are only interested in the relative ordering of the likelihoods for different clutter types Y^i, and $P(w, s, \{n^j\})$ is constant as a function of i. (4.4) follows by the naive Bayes assumption. We now describe how each of the probabilities in (4.4) is computed. All the estimates that are based on the training data are computed during a preprocessing stage. We use uniform small-sample correction in all empirical estimates to avoid zeroing out the probabilities.

Prior $P(x = Y^i)$. There are a number of ways to approximate the prior, the simplest being a uniform approximation. We use a more informed empirical prior. To estimate $P(x = Y^i)$, we enumerate all clutter instances from all classes in all input scenes and take the fraction of these instances that belong to class Y^i.

	# training images	# clutter types	# support relations
Bedroom	383	85	5843
Kitchen	225	67	4067
Living room	221	66	3712
Office	78	53	1245
Classroom	49	62	1315

Table 4.1: *Real-world images used for training the Clutterbrush.*

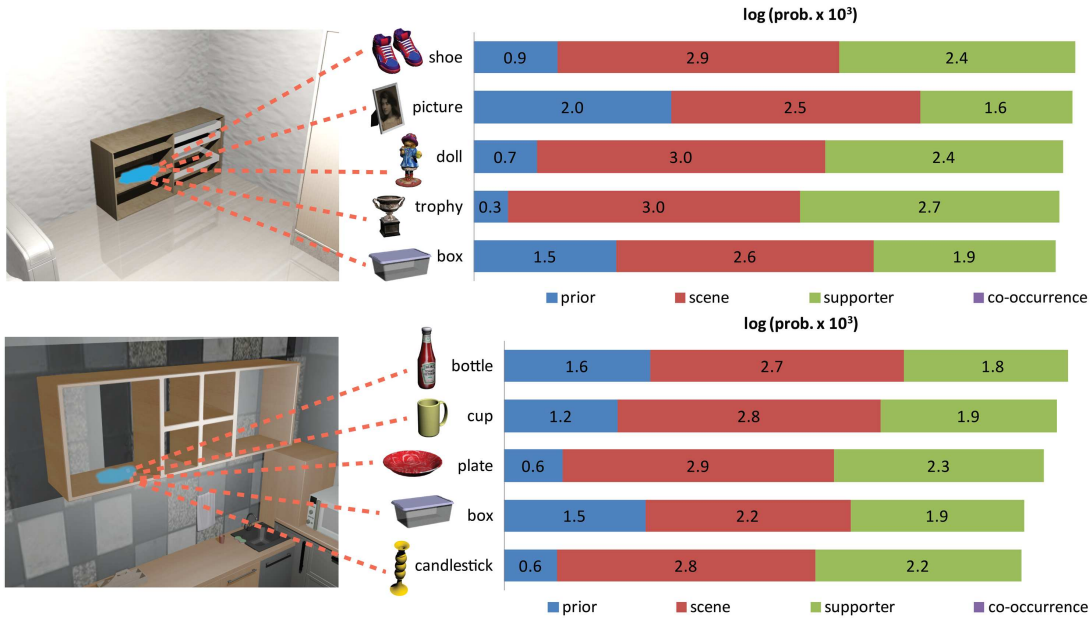


Figure 4.5: *The effect of scene probability. Brushing a shelf in a bedroom (top) and a kitchen (bottom) brings up different suggestions.*

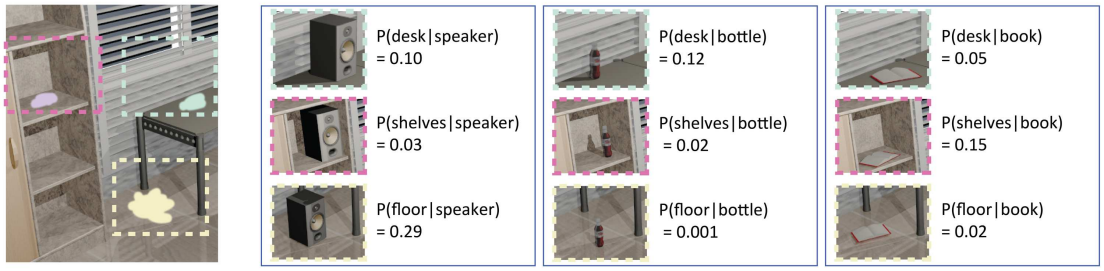


Figure 4.6: *The effect of supporter probability. Clutterbrushing three different supporters in the scene yields different supporter probabilities for the different clutter objects. The speaker has a preference for floor, the bottle has a preference for desk, and the book has a preference for shelves.*

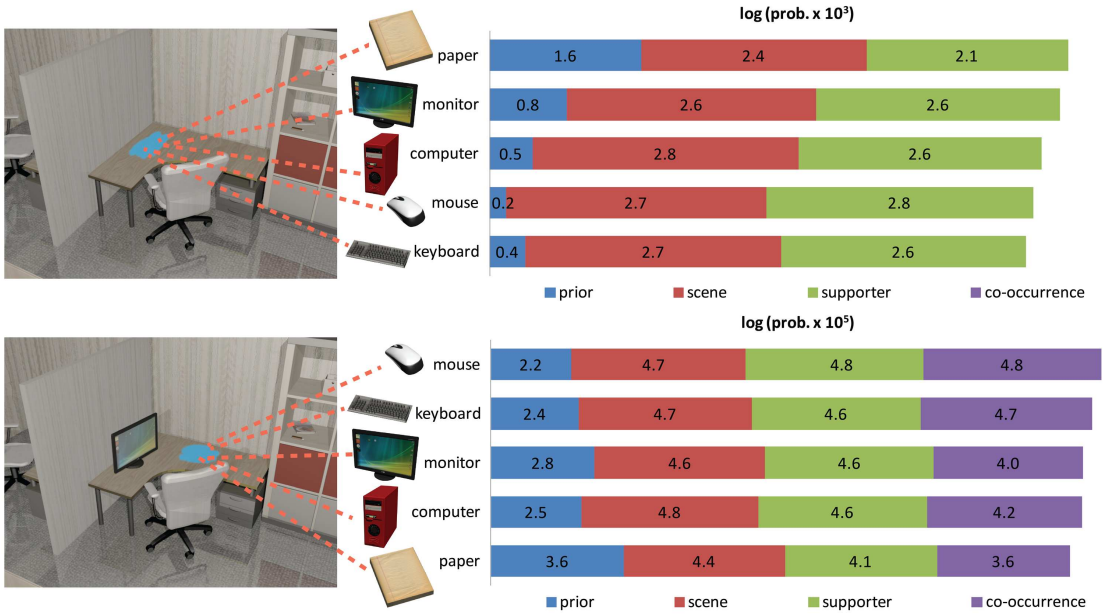


Figure 4.7: *The effect of co-occurrence probability. When the desk is first brushed (top), paper and monitor are the most likely clutter types. After a monitor is placed (bottom), mouse and keyboard become the most likely clutter types to be added.*

Conditional scene probability $P(w|x = Y^i)$. We estimate the conditional scene probability by enumerating all clutter instances from class Y^i in the training set and taking the fraction of these instances that occur in scenes of type w:

$$P(w|x = Y^i) = \frac{P(w, x = Y^i)}{P(x = Y^i)}.$$

Figure 4.5 illustrates the effect of the scene probability on the presented suggestions.

Conditional supporter probability $P(s|x = Y^i)$. Similarly, we estimate the conditional supporter probability by enumerating all clutter instances from class Y^i in the training set and taking the fraction of these instances that occur on supporters of class s:

$$P(s|x = Y^i) = \frac{P(s, x = Y^i)}{P(x = Y^i)}.$$

Figure 4.6 illustrates the effect of the supporter probability.

Co-occurrence probability $P(n^j|x = Y^i)$. This is the probability that there are currently at least n^j instances of class Y^j on the given supporter, conditioned on x being a new instance of class Y^i. Note that j can equal i. In this case, the instance x is not included in the count n^i, thus n^i is the count of other instances from class Y^i on the supporter. Note also that $P(n^j|x = Y^i)$ is interpreted as the probability that there are at least n^j instances of class Y^j on the current supporter (not counting x), because when we consider adding x to the scene, we do not know what other clutter objects will be added by the user subsequently. Figure 4.7 illustrates the effect of the co-occurrence probability.

We can evaluate $P(n^j|x = Y^i)$ by enumerating all supporters in which at least one instance of Y^i appears, and taking the fraction of these supporters in which at least n^j instances of Y^j appear. In the special case of $j = i$, we take the fraction of the scenes in which at least $n^i + 1$ instances of Y^i appear. We use a distance threshold of 1.5 meters such that only pairs of clutter objects within this distance from each other are considered in the co-occurrence estimation (both at runtime and during training). This is to deal with larger supporters, such as floors, which can cover a large area with numerous clusters of clutter.

To alleviate data fragmentation, we quantize n^j to three possible values: 'zero', 'some', and 'many'. To determine the quantization boundaries we collect, in a preprocessing step, the numbers of objects of class Y^j that occurred together on individual supporters in all scenes in the training set. Let m^j be the median of this set. We set n^j to 'zero' if there are no items of class Y^j on the current supporter, to 'some' if there are between 1 and m^j such items, and to 'many' if the number of such items is strictly greater than m^j.

4.5 Results and Evaluation

The Clutterbrush was implemented in C++ and experiments were performed on a 3.33GHz Intel Xeon machine with 12GB of RAM. All empirical statistics were computed in a preprocessing stage. At runtime, a probability of the form (4.1) is evaluated in less than 0.1 milliseconds. Suggestions were thus presented almost instantaneously when the Clutterbrush was used.

To evaluate the utility of the Clutterbrush, we recruited 20 university students and staff to model indoor scenes. There were 12 male and 8 female participants. All participants were frequent computer users and most reported some experience with using simple visual authoring interfaces intended for the general public, such as those used in video games or in photo editing applications.

Each participant was given a brief 5-10 minute tutorial on the modeling interface. We then let the participants experiment freely with the interface until they felt comfortable with the functionality. Our simple interface is demonstrated in the accompanying video: it allows users to add, delete, move, and rotate clutter objects. All participants reported being comfortable with the functionality within 15 minutes.

Each participant was assigned one of the scene types: bedroom, kitchen, living room, office, or classroom. The participant was shown several images of real-world scenes of this type, collected through Google Images. The participant was then asked to use the modeling interface to create a realistic scene of the given type. For each scene type, we manually created an initial scene with the basic furniture. The initial scenes are shown in Figure 4.8(a). These were given to the participants as input. The participants then used the modeling interface to bring the scene to a level of realism that they were satisfied with. There was no time limit, each participant decided on their own when they were done. Modeling times ranged from 6 to 25 minutes.

(a) Input scene (b) After Clutterbrushing

Figure 4.8: *Modeling experiments. (a) Initial scenes given to the participants. (b) Scenes produced by the participants using the Clutterbrush.*

Each participant was randomly and without their knowledge assigned to one of two conditions. In the first condition, the Clutterbrush presented suggestions ranked according to the model described in Section 4.4. In a separate tab, the participants could also browse an alphabetical list of available clutter types and choose objects using that list. The participants could easily toggle between the two tabs. In the second condition, the Clutterbrush only presented the alphabetical list. In this condition, the suggestion engine described in Section 4.4 was not used.

The participants were not aware of the different conditions. 10 scenes were produced in each of the two conditions. The number of scenes of each type is the same in the two sets. The scenes produced in the first condition (suggestions+alphabetical) are shown in Figure 4.8(b).

4.5.1 Usage Pattern and Modeling Time

In the first condition (suggestions+alphabetical), users added 34.1 objects to the scene on average. In the second condition (alphabetical only), users added 30.8 objects on average. In the first condition, the average time between object addition was 22.25 seconds. In the second condition, the average time between object addition was 33.17 seconds. The suggestion engine thus resulted in a 33% improvement in modeling speed.

Figure 4.9 provides a detailed analysis of the time spent using the adaptive suggestions versus the time spent using the alphabetical list, for each of the 10 users in the suggestions+alphabetical condition. The analysis demonstrates that users had a strong preference for the adaptive suggestions. On average, the suggestions were used 87% the time, while the alphabetical list was used only 13% of the time.

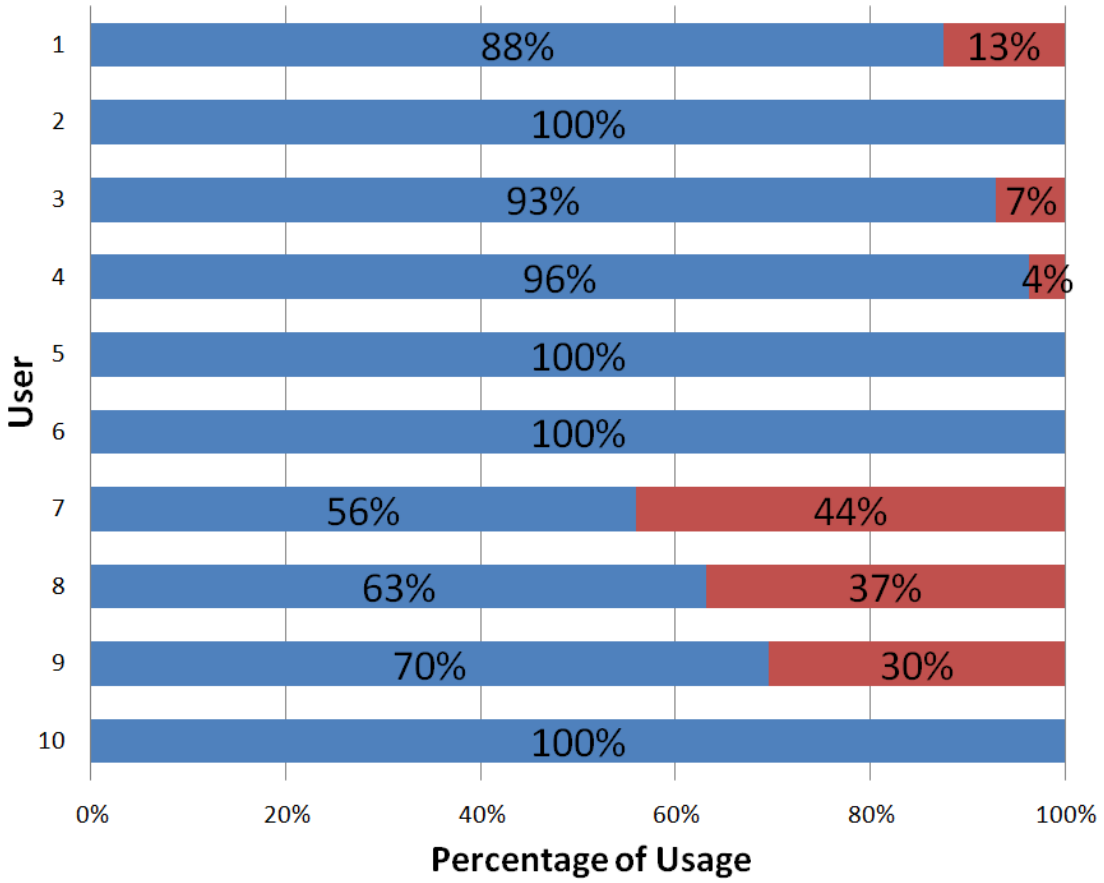


Figure 4.9: *Time spent using the suggestions (blue) versus time spent using the alphabetical list (red), for each user in the first condition, in which both the adaptive suggestions and the alphabetical list were available.*

4.5.2 Scene Realism

We performed two experiments to evaluate the relative realism of the initial scenes, the scenes produced by users in the first condition, and the scenes produced by users in the second condition.

In the first experiment, our goal was to evaluate the relative realism of the scenes before and after Clutterbrushing. To this end, we recruited a separate group of 25 evaluators. This group was recruited through social media. The evaluators were shown pairs of images. For each pair, the evaluator had to indicate which of the presented images is more realistic, or to indicate lack of preference. The two images in each pair were shown side by side and the left-right order was randomized. Each pair showed images of the same scene type. One image was of the scene in its initial condition (Figure 4.8(a)). The other image was of one of the scenes produced by participants in the first condition. The evaluators were not told how the scenes were produced and in what way they were related to each other. In total, 500 pairwise comparisons were performed.

Figure 4.10 summarizes the results of this experiment. The evaluators overwhelmingly preferred the scenes produced by Clutterbrushing. For bedrooms, kitchens, and offices, this preference was exclusive: all users voted that the scenes were more realistic after Clutterbrushing. For living rooms, 97% of the comparisons were in favor of the cluttered scenes. For classrooms, three quarters of the votes were in favor of the cluttered scenes. In informal exit interviews, the evaluators who voted for the uncluttered scenes remarked that empty classrooms are often encountered in the real world, for example when class is not in session.

We also evaluated the relative realism of the scenes produced in the first condition (suggestions+alphabetical) and scenes produced in the second condition (alphabetical only). For this experiment, we recruited a separate group of 32 evaluators, distinct from the prior groups of participants and evaluators. This group was also recruited through

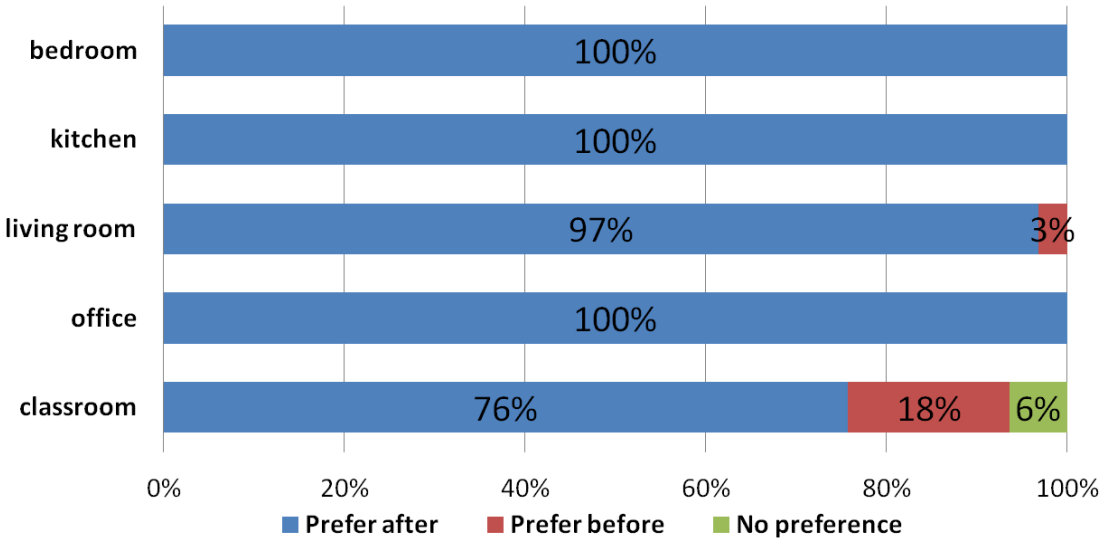


Figure 4.10: *Results of the first experiment. Independent evaluators compared the realism of scenes before and after Clutterbrushing.*

social media. In this experiment the evaluators also performed randomized pairwise comparisons, akin to the first experiment. In each pair, one of the images was of a scene created in the first condition and the other was of a scene created in the second condition. (Each pair showed scenes of the same type.) 1280 pairwise evaluations were performed.

Figure 4.11 summarizes the results. For bedrooms, offices and classrooms, the evaluators has statistically significant preference for the scenes created in the first condition ($p < 0.05$), according to a two-tailed t-test. For kitchens and living rooms, there was no significant preference for either of the two conditions.

4.6 Summary, Discussion, and Future Work

In this chapter, we presented the Clutterbrush, an interactive tool for enhancing the level of detail of indoor scenes. Our experiments demonstrate that people find scenes enhanced with detail objects to be substantially more realistic than scenes prior to detail

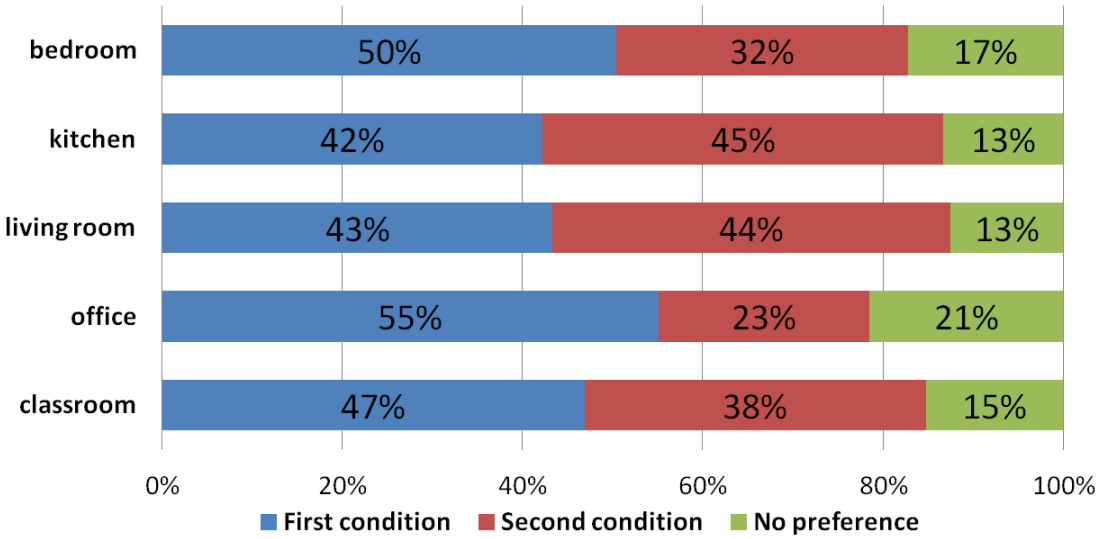


Figure 4.11: *Results of the second experiment. Independent evaluators compared the realism of scenes created in each of the two conditions.*

enhancement. The adaptive suggestion generation functionality of the Clutterbrush was found to improve modeling time and to facilitate the creation of realistic scenes. We expect the Clutterbrush to be useful in a variety of scene modeling applications, including level design interfaces in game engines, home modeling and remodeling applications, and simulation games with user-generated content.

One of the key ideas in our work is the training of the Clutterbrush on real-world images. We expect this idea to have broader applications in settings where semantics of real-world scenes and objects can be used to train intelligent visual authoring tools.

There are many possibilities for further advancing interactive tools for scene detail enhancement. Our approach operates on a purely semantic level and does not take the appearance of objects into account. It also does not reason about detailed placement of objects. This enables our approach to be trained on sparsely annotated images, which are easy to acquire for a broad range of domains. It would be interesting to investigate scalable approaches to training models that reason about finer-grained geometric properties.

CHAPTER 5

Outdoor Photometric Stereo

5.1 Introduction

Photometric stereo is a technique for inferring 3D surface shape from pixel intensities in ordinary 2D images (Woodham, 1980; Horn, 1986). In the conventional photometric stereo setting, multiple images of an object are captured under different illumination conditions. By fitting different reflectance models to the observed pixel intensities, pixel-dense surface normals are estimated (Coleman and Jain, 1982; Tagare and de-Figueiredo, 1991; Solomon and Ikeuchi, 1996; Barsky and Petrou, 2003; Basri et al., 2007; Sunkavalli et al., 2010; Lu et al., 2010a; Yeung et al., 2011b; Chandraker et al., 2011). The focus in the majority of prior photometric stereo work was to tackle technical challenges such as reflectance model assumptions, self-shadowing, specularities, noise and outliers, and so on; accordingly, experiments were normally conducted in well-controlled laboratory environments, usually dark rooms with single localized light sources, which facilitated empirical evaluation while minimizing uncertainties.

The objective of our work is to free photometric stereo from its laboratory confines and help make it a useful tool for computational photography. To this end, we develop an outdoor photometric stereo framework that can deal with real-world objects subject to natural illumination conditions. Figure 1.7 illustrates the application of our framework to a horse statuette in a sunlit outdoor environment. By analyzing the variation of pixel intensities among a set of input images (top row) acquired under different

Method	Surface Assumption	Calibration Object	Capturing Environment	#Images
Ours	Lambertian; varying albedo	mirror sphere	natural illumination	6-10 (theoretical: 4)
Johnson et al.	Lambertian; uniform albedo (painted to match calibration sphere)	same-material sphere	natural illumination	1
Basri et al.	Lambertian; varying albedo	none	dark room; general unknown lighting, fixed intensity	32-64 (theoretical:27)
Ackermann et al.	mixtures of 2-3 fundamental materials	none	natural illumination	30-50 from more than 20k over a year
Oxholm et al.	isotropic BRDF	mirror sphere	natural illumination	1

Table 5.1: *Comparisons between our work and previous works.*

environmental illuminations whose associated environment maps are acquired using a mirror sphere (middle row), our algorithms can estimate the surface normals using our generalized reflectance model and reconstruct the 3D shape of the horse statuette surface (bottom row). To deal with outliers such as shadows, highlights and/or small misalignment errors across input images, we apply low rank matrix completion (Lin et al., 2009) to preprocess the input images. Our shape estimation approach is then formulated as an optimization method which alternates between normal estimation and estimation of environmental illumination that contributes to pixel intensities. Finally, total variation regularization (Rudin et al., 1992; Bresson and Chan, 2008) is applied to refine the estimated normals by reducing ambiguities and noise in preparation for 3D surface reconstruction.

After reviewing related prior work (Section 5.2), we will present the generalized reflectance model employed in our outdoor photometric stereo framework along with the key steps of its processing pipeline, including our acquisition system, preprocessing step, main algorithm, and postprocessing step that yields high-quality pixel-dense

surface normal estimation in natural environments (Sections 5.3–5.4). We will also discuss implementation considerations relevant to these steps. Then, we will demonstrate the effectiveness of our framework through a variety of experiments including synthetic data, different background scenes, indoor scenes with different combinations of light sources, and outdoor scenes with varying sunlight (Section 5.5). Finally, we will present our conclusions and suggest avenues for future work (Section 5.6).

5.2 Related Work

There is a substantial amount of literature on photometric stereo, with representative papers including (Woodham, 1980, 1994; Horn, 1986; Coleman and Jain, 1982; Solomon and Ikeuchi, 1996; Barsky and Petrou, 2003; Wu et al., 2006; Wu and Tang, 2010; Basri et al., 2007; Wu et al., 2010; Shi et al., 2010; Sunkavalli et al., 2010; Lu et al., 2010a; Yeung et al., 2011b; Chandraker et al., 2011).

Much of this prior work is based on the Lambertian surface reflectance model, which requires at least three illumination directions to solve the surface normal estimation problem (Woodham, 1980, 1994). For more than three input images, the surface normal vector at every pixel may be obtained using least squares fitting (Wu et al., 2006; Wu and Tang, 2010), robust low rank minimization (Wu et al., 2010), or subspace clustering (Sunkavalli et al., 2010). Several authors have relaxed the Lambertian model assumption; for instance, Tagare and deFigueiredo (1991) employed an m-lobed reflective map, Solomon and Ikeuchi (1996) used the Torrance-Sparrow reflectance model, Hertzmann and Seitz (2003) used a reference object of the same material and a known shape to compute surface normals through analogy, Nayar *et al.* (1990) used a hybrid reflectance model (Torrance-Sparrow and Beckmann-Spizzichino), Goldman *et al.* (2010) optimized the shape and the BRDFs alternatively by assuming a set of basis materials according to the isotropic Ward model, and Yeung *et al.* (2011b) applied

orientation consistency to estimate normals for transparent objects.

To our knowledge, there are only a handful of papers that consider general/natural illumination conditions. Johnson and Adelson (2011) describe a shape-from-shading algorithm under natural illumination; however, their requirement of a calibration sphere of the same material BRDF as the captured object limits its practicality. Oxholm and Nishino (2012a) relax this restriction by using a mirror sphere to calibrate the illumination, while Yu *et al.* (2013a) utilize information obtained from depth cameras to constrain the problem. Since only a single input image is used, such shape-from-shading methods share some common limitations, including the fact that the estimated surface normals can easily be corrupted by outliers. Basri *et al.* (2007) used low-order spherical harmonics to model general illumination, akin to the environment mapping representation (Ramamoorthi and Hanrahan, 2001a). Their model includes 27 variables and thus requires significantly more input images compared to conventional photometric stereo. The prior work that is most relevant to ours is by Ackermann *et al.* (2012) who captured over twenty thousand outdoor webcam images throughout the year. The robustness of their photometric stereo method can be attributed to the large amount of data and a smart data selection process. However, acquiring this much image data is not an easy task. Very recently, Abrams et al. (2012) proposes a variant of photometric stereo which also works on a time lapse of outdoor images. The approach regards the sun as a distant light source and estimates the lighting direction using GPS information.

We offer an approach to outdoor photometric stereo that strikes an attractive balance between the amount of image data required (about 6 to 10 input images) and the type of calibration object needed (a mirror sphere). Compared to previous methods, our method is both practical and accurate. Table 5.1 compares our work to the aforementioned efforts.

5.3 Environment Light Photometric Stereo

In this section, we describe the basic model of our environment light photometric stereo and our image data acquisition process.

5.3.1 Basic model

In the Lambertian surface model (assuming a linear camera response function), the intensity of a pixel I depends on the surface albedo ρ, the illumination direction $\mathbf{l}$, and the surface normal $\mathbf{n}$ according to

$$I(x) = \rho(x)\mathbf{l}(x) \cdot \mathbf{n}(x), \tag{5.1}$$

where x is the image coordinate. The common photometric stereo setting presumes a distant, directional light source. Thus, $\mathbf{l}$ is spatially invariant and it can easily be estimated from the input images (Shi et al., 2010) or using a calibration object (Wu et al., 2006). To solve the surface normal $\mathbf{n}$ in (5.1), we capture multiple images each taken with a different illumination direction. Hence, we obtain more observations than unknowns in (5.1) and $\mathbf{n}$ can be solved effectively using methods presented in (Woodham, 1980; Wu et al., 2006, 2010; Sunkavalli et al., 2010).

When we have multiple directional light sources, we can extend (5.1) by summing the contribution of each light source to the pixel's intensity, as follows:

$$I(x) = \rho(x) \sum_{i=1}^{K} c_i \mathbf{l}_i \cdot \mathbf{n}(x), \tag{5.2}$$

where K is the number of light sources in the scene, and c_i is the strength of light source i with direction $\mathbf{l}_i$. In the case where the illumination comes from all directions, we can describe the image intensity using (5.2) with K tending to infinity; i.e., an integral. Note that in (5.2) the number of unknowns for $\mathbf{n}$ remains the same as in (5.1) and (5.2) remains a linear equation when ρ and $\mathbf{l}_i$ are known.

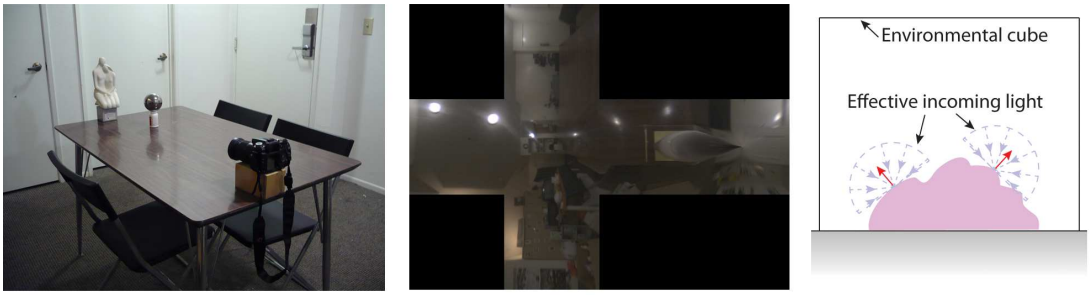


Figure 5.1: *Left: Our simple setup for data acquisition. A mirror sphere is placed near the object. Middle: From the image of the mirror sphere, we estimate the illumination environment map using the method in (Debevec, 1998). Right: For each pixel, we must estimate the illumination directions that contribute to the pixel intensity, which depends on the orientation of the associated surface normal.*

5.3.2 Data acquisition

In the conventional photometric stereo setting, images are acquired in a darkroom and the light source directions can be well calibrated with fixed light sources. By contrast, we wish to use natural environmental illumination. Hence, a major component in our environment light photometric stereo approach is the estimation of the light source directions $\mathbf{l}_i$ in (5.2).

Figure 5.1 shows our experimental setup for data acquisition. We use a mirror sphere to capture the strength of the incoming light from all directions in the form of an environment map, a common method in graphics rendering for representing the effect of distant light sources illuminating object surfaces (Debevec, 1998), and adopt it for surface normal estimation in our environment light photometric stereo. We put the mirror sphere near the object of interest and capture images of both.

Once we have acquired the environment map, we sample a number of directions in the illumination hemisphere using an icosahedron with sub-division (Ballard and Brown, 1982), and average the illumination environment map over these directions; i.e., ac-

cording to (5.2),

$$I(x) = \frac{\rho(x)}{K} \sum_{i=1}^{K} c_i \mathbf{l}_i \cdot \mathbf{n}(x), \tag{5.3}$$

where K is now the number of directions. In our implementation, we sample 2562 directions in the environment map to approximate the illumination. Note that when $\mathbf{l}_i \cdot \mathbf{n}(x) \leq 0$, then $\mathbf{l}_i$ does not contribute to the image intensity $I(x)$ in (5.3). Hence, we also need to estimate the lighting directions (Figure 5.1 (Right)) that contribute to the pixel intensity, as described in the next section.

5.4 Normal Estimation Algorithm

We will now describe our surface normal estimation algorithm. We first present our preprocessing step based on low-rank matrix completion. Then, we present our method for normal and light contribution refinement. Finally, we describe how to include total variation regularization (Rudin et al., 1992; Bresson and Chan, 2008) to postprocess surface normals using spatial support.

5.4.1 Preprocessing via low-rank matrix completion

Our input data is subject to different sources of error—e.g., shadows, highlights, and even pixel misalignment across different image captures—which can affect the performance of our algorithm. To deal with these errors, we adopt the low-rank matrix completion technique (Lin et al., 2009; Wu et al., 2010). We assemble our n input images I^j for $j = 1, \cdots, n$ into a data matrix

$$\mathbf{D} = [\text{vec}(I^1) \cdots \text{vec}(I^n)], \tag{5.4}$$

where vec(I^j) = [I^j(1), $\cdots$, $I^j(m)]^T$ is the vectorized input image and m is the number of pixels in the object mask. Since our environment light model in (5.2) is linear and the

dimension of $\mathbf{n}$ is 3, the rank of matrix $\mathbf{D}$ is in principle at most 3. However, due to the various errors, we observe in practice that the rank of $\mathbf{D}$ is greater than 3. As observed in (Wu et al., 2010), the errors related to photometric stereo are usually sparse; hence, we can isolate them by formulating the problem as a matrix rank minimization:

$$\min_{\mathbf{A},\mathbf{E}} ||\mathbf{A}||_* + \lambda||\mathbf{E}||_1 \quad \text{such that} \quad \mathbf{D} = \mathbf{A} + \mathbf{E}, \tag{5.5}$$

where $\mathbf{A}$ is a rank 3 matrix, $\mathbf{E}$ is the matrix of error residuals, $||\cdot||_*$ and $||\cdot||_1$ are the nuclear norm and L_1-norm, respectively, and $\lambda > 0$ is a weighting parameter. We use the Accelerated Proximal Gradient (Lin et al., 2009) to solve (5.5). The clean low-rank matrix $\mathbf{A}$, computed separately for each color channel, will be used as input data in the subsequent steps.

The above preprocessing step offers us three major advantages: First, by isolating the errors, such as specular highlights and shadows, our normal estimation algorithm is robust. Second, although we fix our captured object and camera physically, small misalignment errors during image data capture are inevitable and the preprocessing step makes our method robust to small misalignment errors by repairing the misaligned pixels with proximal values, which ensures the rank 3 property of matrix $\mathbf{A}$. Third, the low-rank matrix completion allows us to relax the strict requirement of a Lambertian surface reflectance model in comparing with the method in (Johnson and Adelson, 2011), as long as the non-Lambertian aspect can be factorized into the sparse residual matrix $\mathbf{E}$. Figure 5.2 compares results without and with low-rank matrix completion.

5.4.2 Normal refinement using least squares

We will now present our method for normal refinement. We will first assume that we know the lighting directions that contribute to the intensity of a pixel. In the next subsection, we will describe how to refine the contribution of each lighting direction given the surface normal. Hence, the steps for normal and lighting direction refinement

will be performed in an alternating optimization fashion.

In order to deal with surface albedo, we follow the procedures in (Wu et al., 2006) to choose a denominator image I^d and to estimate the surface normals from ratio images

$$\frac{I^j}{I^d} = \frac{\sum_i^K c_i^j \mathbf{l}_i^j \cdot \mathbf{n}}{\sum_i^K c_i^d \mathbf{l}_i^d \cdot \mathbf{n}} \tag{5.6}$$

where I^j, $j = 1, \cdots, n-1$ are the input after low-rank matrix completion. From (5.6), we can re-write (5.3) as

$$\mathbf{A}\mathbf{n} = \mathbf{0} \tag{5.7}$$

where $\mathbf{A} = [I^j \sum_i^K c_i^d \mathbf{l}_i^d - I^d \sum_i^K c_i^j \mathbf{l}_i^j]$. The least square solution of $\mathbf{n}$ can be obtained by singular value decomposition (SVD), which explicitly enforces $||\mathbf{n}|| = 1$.

5.4.3 Illumination contribution refinement

Given the estimated normal direction, we now wish to refine the contribution of illumination direction that affects the pixel intensity. Without self-occlusion, this can be achieved by fitting a hemisphere of directions such that $\mathbf{l} \cdot \mathbf{n} > 0$ (Figure 5.1 (Right)). We propose a simple heuristic method to evaluate self-occlusion: If the normal direction between neighboring pixels forms a concave shape and the current normal direction is closer to $[0, 0, 1]^T$, it is likely that the incoming light from the neighboring directions is being occluded. Hence, we give a smaller weight to the light from that projected lighting direction. We evaluate self-occlusion for all directions within the local neighborhood and finally obtain a weighted mask for the contribution of illumination directions, which represents the relative contributions of the light from different directions.

We initialize the normal direction and the corresponding hemisphere of environment illumination using exhaustive search, which minimizes the errors from the input images. The exhaustive search algorithm generally provides a good initialization, but it is

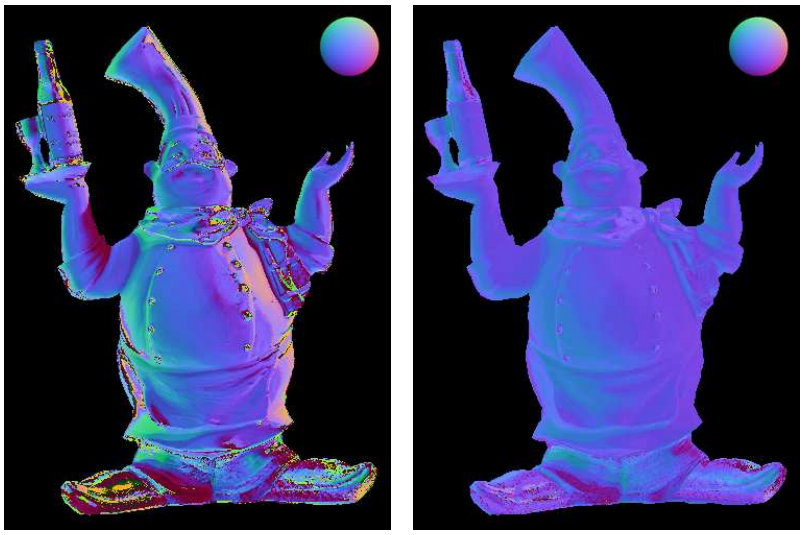


Figure 5.2: *Self-comparisons of our results without (left) and with (right) low-rank matrix completion.*

slow if the search space is large. Therefore, we sample only 42 different normal directions for the exhaustive search initialization; i.e., an icosahedron subdivided once. This yields a good balance between accuracy and efficiency for our alternating optimization (AO) approach.

5.4.4 Spatial refinement using TV regularization

Thus far, our normal estimation method processes each pixel individually. As demonstrated in several previous works (Wu et al., 2006; Goldman et al., 2010; Shi et al., 2010), spatial regularization is useful in error correction as well as in improving the overall accuracy of the estimated surface normals. In our postprocessing step, we employ L_1-norm vectorial total variation (TV) regularization to refine the estimated surface normals $\mathbf{n}^*$ obtained from the previous section, as follows:

$$\mathbf{n}^\star = \arg\min_{\mathbf{n}} ||\mathbf{n}^* - \mathbf{n}||^2 + \lambda|\nabla_\Omega \mathbf{n}|, \tag{5.8}$$

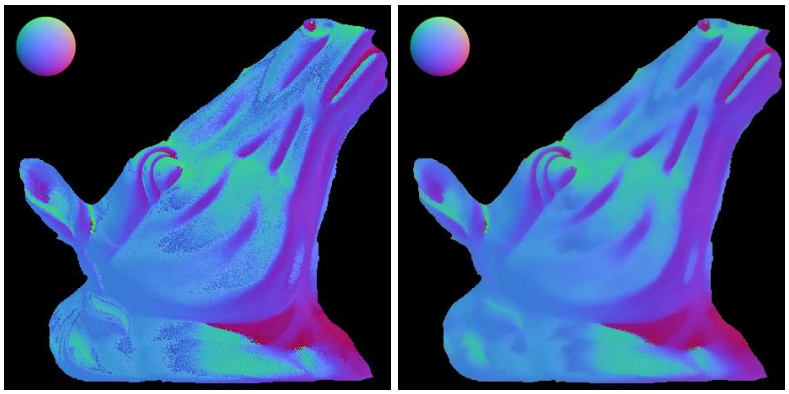


Figure 5.3: *Self-comparisons of our results without (left) and with (right) TV regularization.*

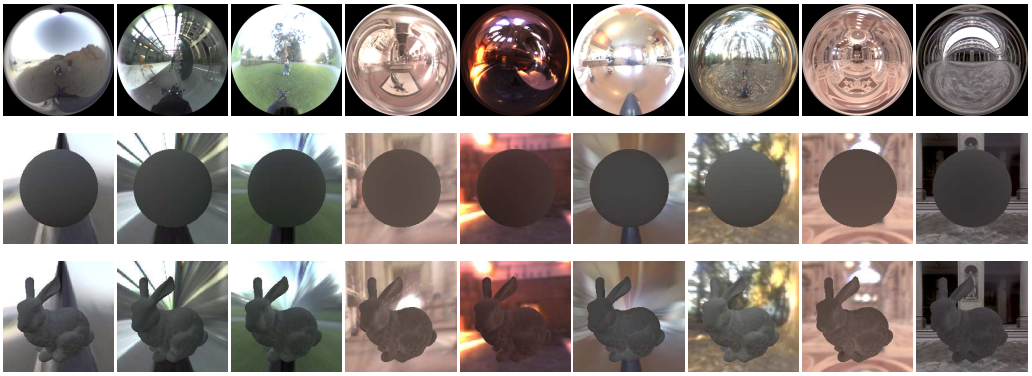


Figure 5.4: *Input environments and images for the synthetic examples Sphere and Bunny.*

where $\nabla_\Omega \mathbf{n}$ is the vectorial first derivative of $\mathbf{n}$ defined over a local neighborhood in Ω and $\lambda = 0.1$ is the regularization weight (see (Bresson and Chan, 2008) for the details).

Figure 5.6 shows intermediate results during the AO iterations and Figure 5.3 compares the result without and with spatial refinement with TV regularization. The TV regularization postprocessing step produces our final normal estimation results $\mathbf{n}^\star$, and we use the technique from (Wu et al., 2008) to reconstruct the 3D surface from $\mathbf{n}^\star$.

5.5 Results

We will now validate the efficacy of our proposed method in experiments using both synthetic and real objects.

5.5.1 Quantitative evaluation with synthetic images

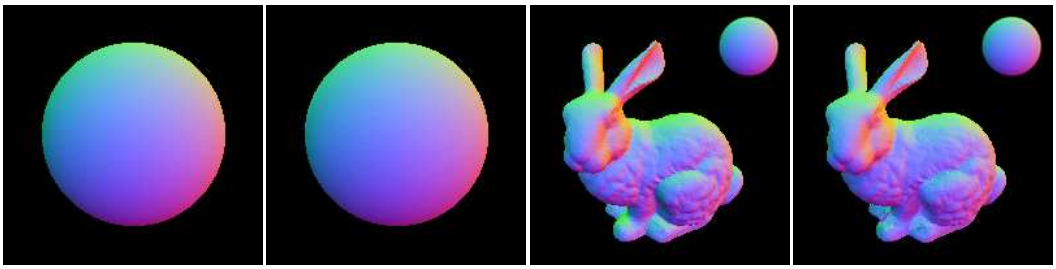
Our first experiment evaluates uses synthetic input images for which ground truth normal maps are available, and we analyze the effect of the number of input images and the convergence of our AO.

Two synthetic examples Sphere and (Stanford) Bunny were used for quantitative evaluation. We use the environment maps from (Debevec, 1998) to render the synthetic input images as shown in Figure 5.4. We show the color coded ground truth normal maps and estimated normal maps in Figure 5.5 for qualitative comparison. Our approach faithfully estimates the surface normals, which closely approximate the ground truth normals with RMS error of 0.0099 and 0.1051 for the Sphere and Bunny, respectively.

To evaluate the robustness of our method, we plot the RMS error of the estimated normals with different numbers of input images in Figure 5.6. As expected, the RMS error decreases as the number of images increases, with less significant decreases forthcoming after more than 5 input images. Figure 5.6 plots the RMS error against the number of iterations, which shows that our approach converges in 4 to 5 iterations for both examples.

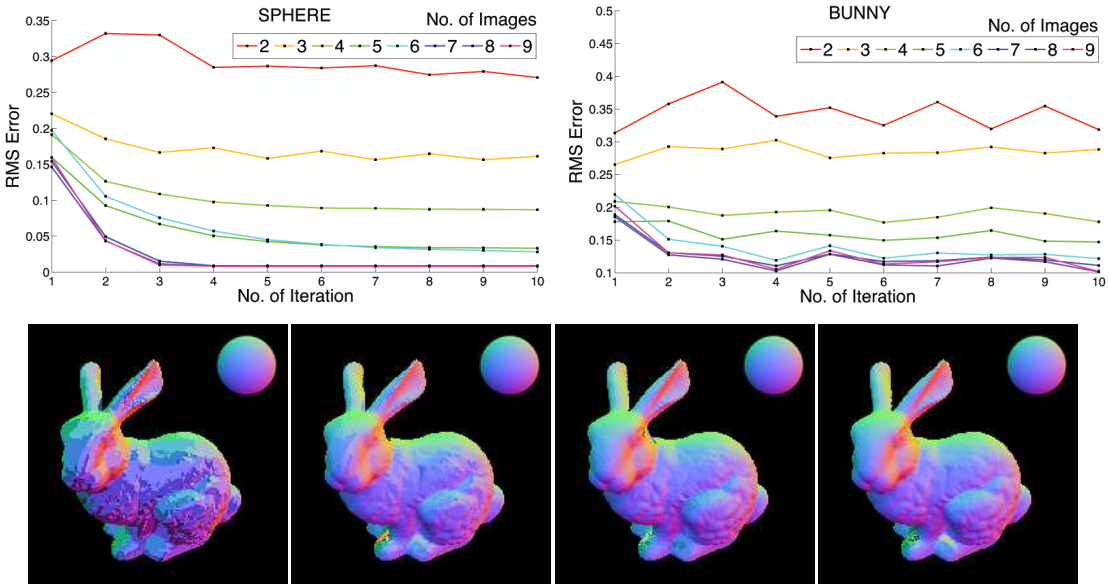
5.5.2 Qualitative evaluation with real images

Next, we evaluate our framework on various real world examples under different illumination conditions, including different background scenes, an indoor scene with



(a) Ground truth (b) Result (c) Ground truth (d) Result

Figure 5.5: *Comparison between ground truth and normal maps obtained using nine environments. The results are obtained after four iterations of the AO process.*



Iteration 0 Iteration 1 Iteration 2 Iteration 3

Figure 5.6: *Convergence analysis of our alternating optimization framework. Top: RMS error of the estimated normals versus the number of iterations, using different numbers of input images. Bottom: qualitative illustration of our intermediate results for Bunny. Iteration 0 is the result after the exhaustive search initialization.*

	#images	size	time (sec)
Sphere	9	200×200	40.3
Bunny	9	200×200	27.9
Couple	10	300×420	450.0
Mother&Baby	10	230×500	334.0
HorseHead (Indoor)	7	600×600	575.7
Chef (Indoor)	7	371×514	468.7
Shoe (Indoor)	7	496×234	309.2
Horse (Sunlight)	8	260×347	140.8
Chef (Sunlight)	8	371×503	466.9

Table 5.2: *The running times of our Matlab R2009b implementation were measured on a 3.33GHz Intel Xeon PC.*

Figure 5.7: *Input environments and images for Couple and Mother&Baby*

different indirect light sources, and an outdoor scene with sunlight direction varying throughout a day. The running times are indicated in Table 5.2.

Different background scenes. Our first real-world experiment closely mimics the synthetic experiments, by capturing the object in different illumination environments. We used 10 input images for the examples Couple and Mother&Baby. The input images are shown in Figure 5.7 and Figure 5.10 depicts the results. The reconstructed normal maps and surfaces appear faithful. We show the $\mathbf{l} \cdot \mathbf{n}$ images under two different lighting conditions to reveal the shading, and we show the images of the real objects alongside the reconstructed surfaces captured from a similar viewpoint. The corresponding closeup view illustrates the details preserved in the reconstructed surfaces. For example, the arms and legs of the Couple are clearly estimated and we can clearly see that the mother is holding her baby in Mother&Baby.

Indoor scene with different illumination conditions. Next, we consider in an indoor scene with different illumination conditions, by turning on/off different light sources in a room. Note the presence of ambient light and the use of indirect light sources (e.g., table lamp, floor lamp). We captured illumination environments and images for Shoe (Indoor), HorseHead (Indoor) and Chef (Indoor), some of which are shown in Figure 5.8. Figure 5.10 shows the results. As can be seen from the closeup image, our results are very good under these conditions and subtle details such as the textures of the shoe are faithfully reconstructed.

Outdoor scene with moving sunlight. Our final experiments were conducted in an outdoor environment using sunlight for reconstruction, which is the main goal of this project. We captured images of the Horse (Sunlight) and Chef (Sunlight) objects every hour from 10am to 5pm, obtaining eight input images per object. The input images and results are shown in Figure 1.7, Figure 5.9, and Figure 5.10. The results of Chef (Indoor) are shown alongside to facilitate comparisons. We find that although the results for the outdoor environment are reasonably good, they are not as good as those for the

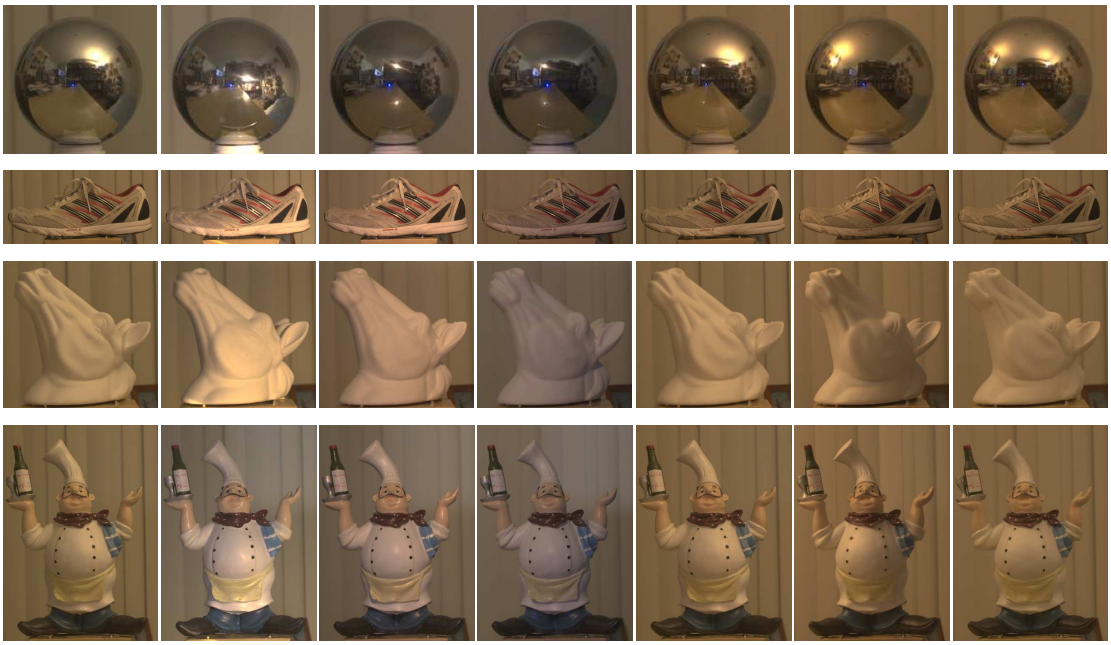


Figure 5.8: *Input environments and images for Shoe (Indoor), HorseHead (Indoor) and Chef (Indoor)*

Figure 5.9: *Input environments and images for Chef (Sunlight).*

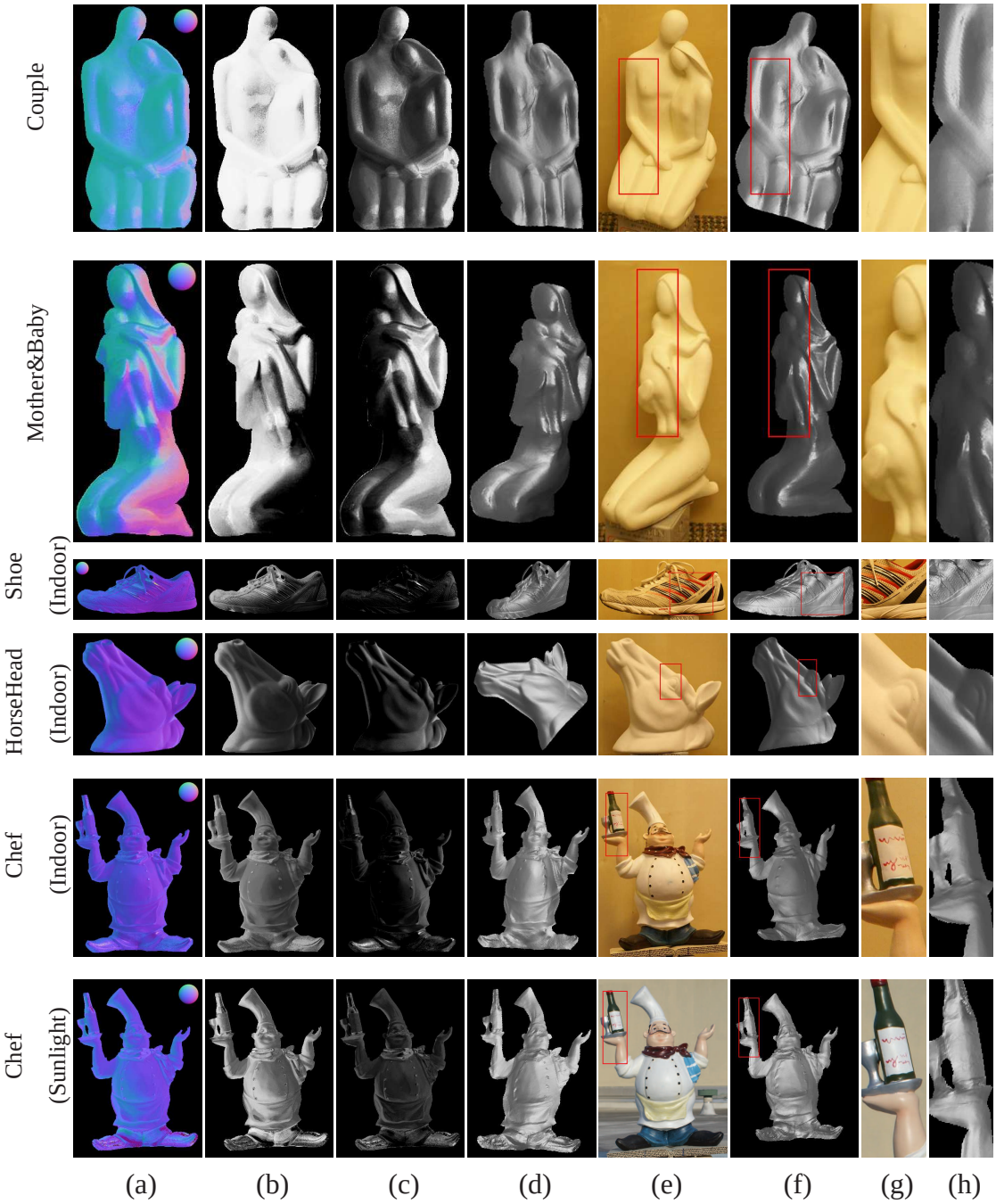


Figure 5.10: *Real-world results. (a) Color-coded normal map. (b–c) Normal map shaded by* $\mathbf{l} \cdot \mathbf{n}$ *with* $\mathbf{l} = [-1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}]^T$ *and* $\mathbf{l} = [1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}]^T$*, respectively. (d) Novel view of the reconstructed surface. (e-h) Closeup view comparing the reconstructed surface with the real object from a similar viewpoint.*

indoor environment. Part of the reason is the relatively modest variation in sunlight as the sun moves along its trajectory, whereas the light sources in the indoor environment are well distributed in different directions.

5.6 Summary, Discussion, and Future Work

In this chapter, we presented a photometric stereo framework that employs natural environmental illumination, demonstrating the feasibility of practical outdoor photometric stereo. Featuring a simple setup for data capture with an optimization framework for dense object surface normal estimation, our system achieves high quality normal estimation even for complex indoor and outdoor scenes with natural illumination.

By exploiting low-rank matrix completion and total variation regularization techniques, our framework is robust to small object misalignment, shadows, and highlights. We believe that our framework has effectively mitigated the limitations of conventional photometric stereo algorithms, among them the need for a controlled environment for image data capture.

In the future, we plan to extend our framework to incorporate non-Lambertian surface reflectance models. We also aim to integrate our framework with multi-view structure-from-motion algorithms in order to reconstruct high quality models of 3D objects in their entirety.

CHAPTER 6

Shading-Based Shape Refinement of RGB-D Images

6.1 Introduction

Shape-from-shading (SfS) is a challenging problem because of the considerable ambiguity in its solution. For the simplest case of Lambertian reflectance and known albedo, the derived solution suffers from bas-relief ambiguity (Horn and Brooks, 1989; Zhang et al., 1999; Durou et al., 2008). When albedo is unknown, the range of possible solutions expands significantly. To resolve these ambiguities, an obvious solution is to utilize a set of input images under different lighting conditions, which transforms the SfS problem into that of photometric stereo (Woodham, 1980; Yu et al., 2013b). However, such additional input data is often inconvenient to obtain in practise. Recent techniques for SfS (Johnson and Adelson, 2011; Oxholm and Nishino, 2012b) estimate shape from a single input image under natural illumination, but deal with uniform-albedo objects and require a special calibration target to measure lighting.

We propose a shading-based shape refinement algorithm that utilizes Microsoft Kinect to address the ambiguities that exist among lighting, normals and albedo. The Kinect records each RGB image together with a depth map. Although the depth map is noisy and typically contains holes,[1] we present a method that effectively utilizes this information to improve the performance of SfS for scenes with unknown reflectance variation

[1] Depth map holes result from scene areas in a Kinect depth image outside the depth sensing range or occluded from the infrared light projections, since the infrared projection and sensing directions are not the same.

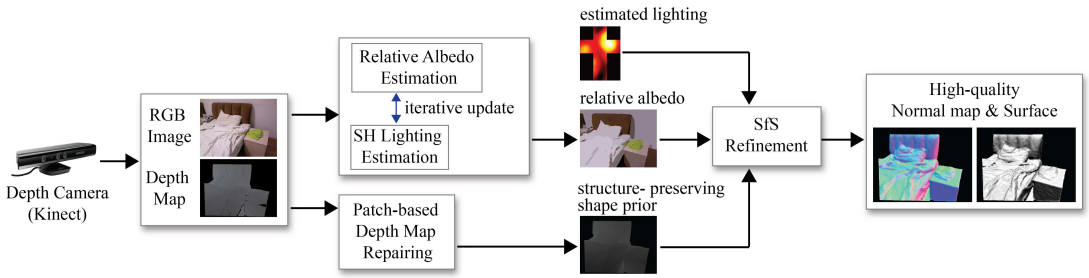


Figure 6.1: *Flowchart of our approach.*

and lighting. The depth information not only helps to resolve bas-relief ambiguity, but also aids in clustering pixels with similar normal directions. Such grouping allows us to effectively estimate relative albedos and the environment illumination in terms of spherical harmonics. To handle the holes in a depth map, we use edges from the RGB image to guide a structure-preserving hole filling process and create a reliable depth map proxy for our shading-based shape refinement algorithm. The utilization of a noisy, incomplete depth map in our approach leads to high-quality 3D scene reconstruction, as exemplified in Figure 1.8.

6.2 Related Work

Our work is related to SfS and depth map enhancement. Recent advances in SfS aim to relax strict assumptions about lighting and reflectance. In (Johnson and Adelson, 2011), Johnson and Adelson show that the inherent complexity of natural illumination actually benefits shape estimation instead of introducing greater ambiguity. Their work uses a reference sphere with the same reflectance properties as the target object to model the object's shading under the environment illumination (Ramamoorthi and Hanrahan, 2001b). Instead of assuming Lambertian reflectance, Oxholm and Nishino consider arbitrary isotropic BRDFs (Oxholm and Nishino, 2012b), with the illumination environment acquired using a reflective sphere. Recently, Barron and Malik (2012) proposed an SfS approach that enforces multiple priors on shape, albedo and illumina-

tion in estimating those properties. Our approach differs from these recent techniques in that it employs an RGB-D camera, but does not require a calibration target, an assumption of uniform albedo, or reliance on smoothness and entropy constraints which may be unsuitable for the given scene. This makes our approach more general and robust in practice. For a more extensive review of SfS, we refer readers to various surveys (Horn and Brooks, 1989; Zhang et al., 1999; Durou et al., 2008).

Apart from single image approaches, another direction is to use a shape prior to constrain the solution space of SfS. Huang and Smith (2011) interpolate the boundary normals of an object to obtain a rough shape prior to constrain SfS. Wu et al. (2011) use multiview stereo to obtain rough but reliable geometry and use it to resolve the local ambiguity of SfS. After that, the SfS solution is used to enhance the multiview stereo geometry by integrating subtle details from SfS. Such a solution, however, cannot be directly applied with an RGB-D image that contains substantial noise and holes. Their method also does not handle objects with reflectance variation.

With regard to depth map enhancement, recent advances use an additional RGB image to denoise and upsample a depth map (Yang et al., 2007; Dolson et al., 2010; Park et al., 2011). With an RGB image that has a higher resolution and signal-to-noise ratio than the depth map, a direct approach is to apply a joint bilateral filter (Yang et al., 2007; Dolson et al., 2010) using the RGB image to define a neighborhood smoothness term. In (Park et al., 2011), Park et al. formulate this as an optimization problem and show that with a small amount of user interaction, the depth map can be greatly improved. But while these depth map enhancement methods can reduce noise and increase resolution, they also lose fine depth details during the smoothing process. By contrast, our approach recovers fine depth details even if they are not captured in the initial noisy depth map, by making greater use of the RGB image through an analysis of its shading.

6.3 Depth-Assisted SfS Approach

To facilitate SfS, our approach utilizes partial depth information to separate shading from albedo, aid illumination estimation, and resolve surface normal ambiguity. No assumptions are made on the incident illumination or surface geometry, while the reflectance in the scene is taken to be Lambertian.

6.3.1 Overview

Figure 6.1 displays a flowchart of our algorithm. From the input RGB image and depth map, our method first computes a normal map from the captured depth map and segments the RGB image into regions of piecewise smooth color. Through alternating optimization (AO), the relative albedos among the different regions are calculated, and the environment illumination is estimated from the albedo-normalized image. After that, we estimate normals over the whole image using SfS with the help of a normal map computed from Kinect as a shape prior to resolve bas-relief ambiguity. For regions that lack depth map values from Kinect, we use a constrained texture synthesis to fill in the missing depth values prior to applying our normal estimation algorithm. As shown in Figure 1.8, the output of our method is a refined normal map without the shape and reflectance ambiguities of SfS nor the noise and holes of the Kinect range data.

6.3.2 Relative Albedo and Lighting Estimation

The input from Kinect consists of an RGB image $I = \{I_i\}$, $I_i = [I_{i,r}, I_{i,g}, I_{i,b}]^T$ where i is the pixel index, and a depth map. From the point cloud determined from the depth map, we calculate a rough normal map $N = \{n_i\}$, where $n_i = [n_{i,x}, n_{i,y}, n_{i,z}]^T$ is the unit normal at pixel i, obtained by a simple cross-product of the neighboring points. For pixels with missing depth values, or whose neighboring pixels have any missing

depth values, no initial normal is computed.

6.3.2.1 Relative Albedo from Common Normals

We first perform a mean-shift clustering on the RGB image, with a minimum region size of 500 pixels. Suppose this forms a set of S clusters $\mathbb{C} = \{C_u, u = 1, ..., S\}$. Each cluster contains a set of pixels and a corresponding set of normals. Under consistent environment lighting, any two pixels a and b with same normal direction in two different clusters have the same shading, and thus the differences between their pixel values are due only to differences in their relative albedos, $p_{a,k}$ and $p_{b,k}$:

$$\frac{I_{a,k}}{I_{b,k}} = \frac{p_{a,k}}{p_{b,k}}$$

where $k = 1, 2, 3$ respectively index the RGB channels. With this property, we solve for the relative albedos between different clusters using pixel-pairs of common normals from different clusters. We note that intensity ratios have also been used as an illumination invariant for object recognition (Funt and Finlayson, 1995; Nayar and Bolle, 1996).

6.3.2.2 Data Structure

To facilitate normal direction comparisons among clusters, we quantize all possible normal directions to vertices on an icosahedron, which provides a uniformly-distributed set of $T = 642$ normal directions over a sphere. The normals in an image are stored in a data structure $B_{u,j,k}$ which we refer to as *bins*, where $u = 1, ..., S$ denotes the cluster index, $j = 1, ..., T$ denotes the normal directions as sampled from the icosahedron, and $k = 0, ..., 3$ with $B_{u,j,0}$ as an indicator bit of whether the j-th normal direction exists within cluster C_u, and $[B_{u,j,1}, B_{u,j,2}, B_{u,j,3}]$ store the RGB values corresponding to the j-th normal direction in cluster C_u.

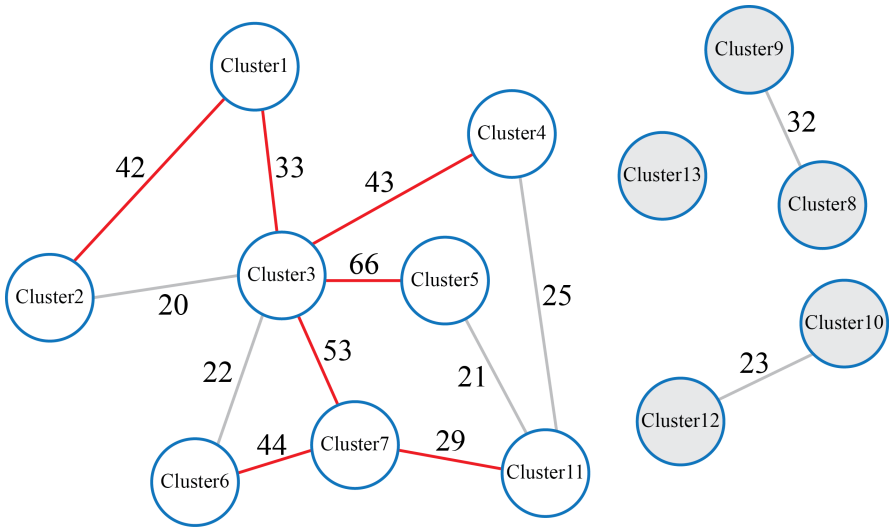


Figure 6.2: *Graph of common-normal-direction relationships among clusters, with the maximum spanning tree indicated by red edges.*

All the bin values are initialized to zeros. Then, for each cluster C_u, each normal n falls into a bin $B_{u,t,k}$, where n has the smallest dot-product with the t-th normal direction among all the T normal directions on the icosahedron. We set $B_{u,t,0} = 1$ to indicate that this bin is utilized. Then we fill in $B_{u,t,k}$, where $k = 1, 2, 3$, with the RGB values of the pixel with normal n. If there are multiple pixels having normals that fall into the same bin $B_{u,t,k}$, the median of their RGB values is used.

6.3.2.3 Graph Representation

After the data structure is built, we represent the common-normal-direction relationships between different clusters as a graph, $\mathbb{G} = \{\mathbb{C}, \mathbb{E}\}$. Each cluster C_u is represented as a node. An edge E_{u_1,u_2} exists between cluster C_{u_1} and C_{u_2} only if there are more than λ common normal directions between clusters C_{u_1} and C_{u_2}, with $\lambda = 20$ in our experiments. The edge is given a score equal to the number of common normal directions. Refer to Figure 6.2 for an example graph.

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

Figure 6.3: *Relative albedos estimated by our alternating optimization. (a) Input image, (b) & (c) Relative albedos estimated at the 1st and 5th iteration, (d) & (e) Corresponding shading images at the 1st and 5th iteration. Clusters without relative albedos in the 1st iteration are simply filled by original RGB values in (b).*

In estimating a globally consistent set of relative albedos, we utilize the maximum spanning tree (MST) algorithm to determine a cycle-free set of links that maximize the number of common normal directions between clusters. As the graph may be disconnected, a forest of trees may be formed. After the MST is found, we calculate the relative albedos between all of its clusters in a depth-first search order along the tree.

The relative albedo between two clusters is computed by first determining the common bins (corresponding to common normals) utilized in clusters C_{u_1} and C_{u_2}, denoted by $Q = \{q : B_{u_1,q,0} = 1 \text{ and } B_{u_2,q,0} = 1\}$. Then we obtain an estimate of relative albedo of C_{u_2} over C_{u_1} for each of the RGB channels:

$$p_{u_2,k} = \frac{B_{u_2,q,k}}{B_{u_1,q,k}}.$$

Among all common bins, we run RANSAC to obtain the relative albedo estimates in a manner robust against outliers. Pseudocode of this relative albedo estimation procedure is provided in the supplementary material.

6.3.2.4 Lighting Estimation

The estimated relative albedos are highly useful. By normalizing the albedos in different regions, we can then jointly use their rich variety of normal directions to more reliably estimate the environment lighting.

Suppose there are R pixels whose relative albedos are estimated from the MST, and let $\hat{n_i} = [n_i^T \ 1]^T$. We estimate the lighting in terms of 2nd order spherical harmonics (SH) for each RGB channel $k = 1, 2, 3$:

$$\hat{n_i}^T M_k \hat{n_i} = \frac{I_{i,k}}{p_{i,k}} \tag{6.1}$$

where $i = 1, ..., R$ and M_k depends on the SH coefficients for the k-th RGB channel (Ramamoorthi and Hanrahan, 2001b). Using the RGB image I and initial normal map N computed from Kinect, M_k in (6.1) can be estimated up to a scale factor by linear least-squares minimization.

6.3.2.5 Refinement by Alternating Optimization

With the estimated lighting, we refine the relative albedos and calculate the relative albedos of those clusters not yet estimated. For each cluster, an estimate of relative albedo for each RGB channel k is obtained for *each normal* n_i in the cluster as:

$$p_{i,k} = \frac{I_{i,k}}{\hat{n_i}^T M_k \hat{n_i}}. \tag{6.2}$$

RANSAC is again run on these estimates to obtain an updated relative albedo for each cluster. Using the updated relative albedos of the MST clusters, we re-estimate the SH coefficients by (6.1). This alternating optimization process is repeated until the change falls below a small value. In practice, convergence is obtained in 3-5 iterations. An example of the improvements gained through iterative optimization is shown in Figure 6.3. We note that despite the noisy normals of the depth map, the relative albedos

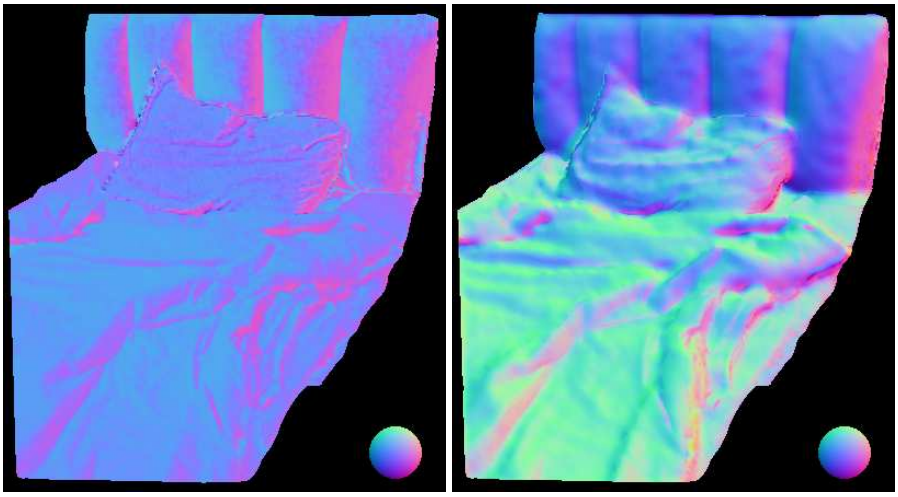


Figure 6.4: *Effect of prior normals on handling bas-relief ambiguity. Left: without prior normals, the bed is roughly co-planar with the backboard. Right: by accounting for prior normals, the bed normals are correctly pointing upward.*

between two regions can be reliably determined when they have many normals in common, as is the case for connected nodes in the MST. Moreover, the environment lighting can also be dependably recovered when the number and range of noisy normals is large, as again is the case with the MST.

6.3.3 Geometry Estimation

6.3.3.1 Structure-Preserving Shape Prior

Shape-from-shading on a single image is an ill-posed problem that suffers from bas-relief ambiguity (Belhumeur et al., 1999) (see Figure 6.4) unless a shape prior is enforced. In our work, we exploit the Kinect RGB-D data to obtain a structure-preserving shape prior, in the form of prior normals to be used later in a normal refinement step.

Kinect depth maps, however, frequently contain holes where there is no depth information for directly computing surface normals. Rather than filling the holes by smooth interpolation, which tends to lose sharp edges and corners (see Figure 6.7), we estimate

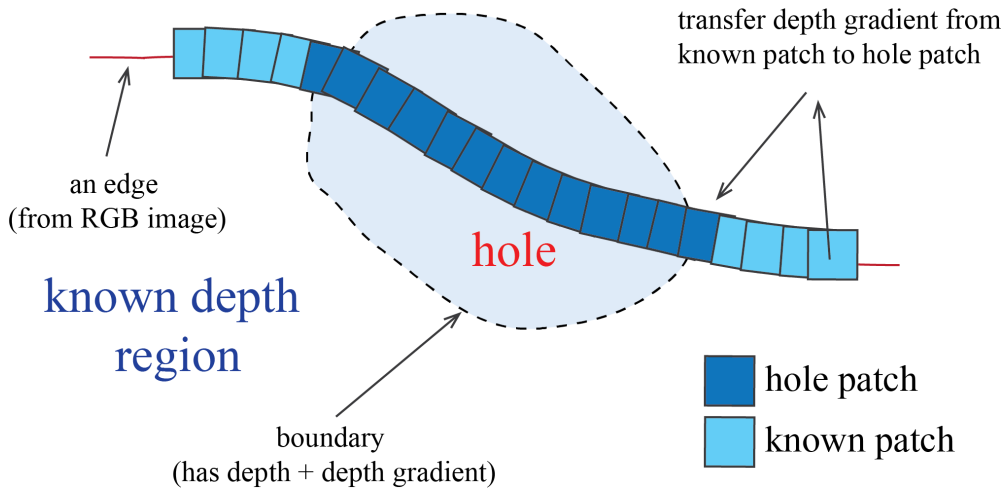
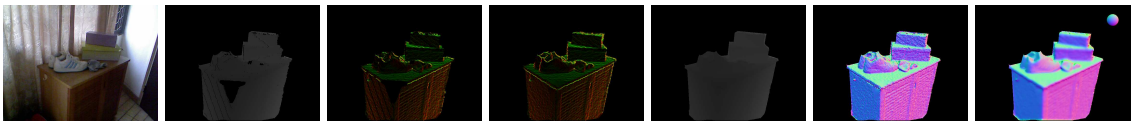


Figure 6.5: *Illustration of patch-based repairing of a structural hole.*



(a) (b) (c) (d) (e) (f) (g)

Figure 6.6: *Example of repairing a depth map hole. (a) Input RGB, (b) Input depth, (c) Depth gradient map, (d) Depth gradient map after patch repair, (e) Depth map after patch repair and poisson integration, (f) Prior normal map, (g) Resulting normal map after SfS.*

the missing data in a structure-preserving manner, similar in spirit to (Sun et al., 2005) but with different considerations due to our different problem setting.

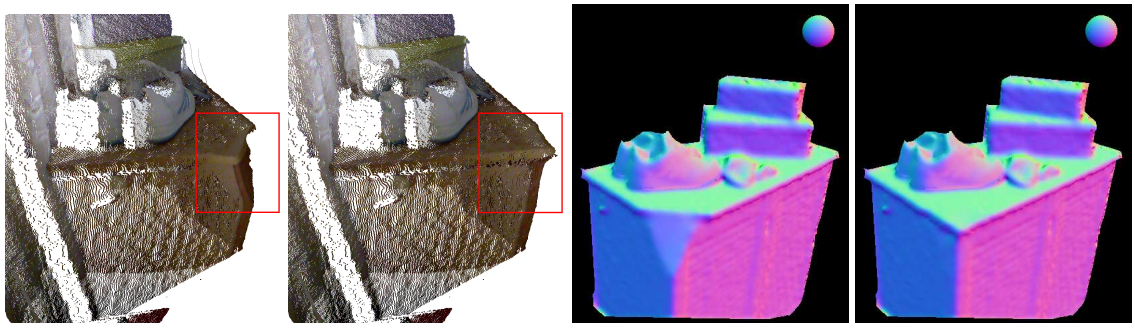
Though holes may exist in the depth image, they do not appear in the corresponding RGB image. We thus take advantage of the RGB image as a guide for depth completion in the hole region. First, a Canny edge detector is applied to the RGB image. We then identify RGB edges that pass through a hole, referred to as a *structural hole*, in the depth image. Along the edge, we generate *hole patches* which contain hole pixels whose depths need to be obtained, and *known patches* which contain no hole and are used for repairing the hole patches. Figure 6.5 shows an illustration.

The goal is to transfer the depth gradients from the known patches to the hole patches, after which the depth of the hole can be filled in by poisson integration while preserving the structure along the edge. This structural propagation is formulated as an MRF which is solved by belief propagation. The MRF total cost function for the set of hole patches $\mathbb{H}$ is defined as:

$$\begin{aligned}\mathcal{C}_{BP}(\mathbb{H}) =& w_{D_{rgb}}\mathcal{C}_{D_{rgb}}(\mathbb{H}) + w_{S_{rgb}}\mathcal{C}_{S_{rgb}}(\mathbb{H}) + \\ & w_{D_{dg}}\mathcal{C}_{D_{dg}}(\mathbb{H}) + w_{S_{dg}}\mathcal{C}_{S_{dg}}(\mathbb{H})\end{aligned} \tag{6.3}$$

where $\mathcal{C}_{D_{rgb}}(\mathbb{H})$, $\mathcal{C}_{S_{rgb}}(\mathbb{H})$, $\mathcal{C}_{D_{dg}}(\mathbb{H})$ and $\mathcal{C}_{S_{dg}}(\mathbb{H})$ are respectively the RGB data cost, RGB smoothness cost, depth gradient data cost and depth gradient smoothness cost. We set $w_{D_{rgb}} = 1.0$, $w_{D_{dg}} = 1.0$, $w_{S_{rgb}} = 0.1$ and $w_{S_{dg}} = 0.1$ in our experiments. Each cost term is detailed as follows.

Denote the set of hole patches as $\mathbb{H} = \{\mathbf{H}_l\}$ and the set of known patches as $\mathbb{K} = \{\mathbf{K}_m\}$. $\mathbf{H}_l$ is itself a set containing all pixels that lie within the patch, with each pixel indexed by local patch coordinate $\mathbf{p}$. For notational convenience, we also define $H_l(\mathbf{p})$ as the pixel location in image coordinates, such that $I(H_l(\mathbf{p}))$ is the RGB intensity of the pixel, and likewise for $I(K_m(\mathbf{p}))$. Also, H_l^{-1} returns the corresponding known patch's index, such that $\mathbf{K}_{H_l^{-1}}$ is the patch that repairs $\mathbf{H}_l$.



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

Figure 6.7: *Example of patch-based repairing versus smoothing to obtain a structure-preserving shape prior for the example scene in Figure 6.6. Patch-based repairing allows propagation of existing structure to the hole region. (a) Shape prior using poisson smoothing. (b) Shape prior using patch-based repairing. (c) & (d) Resultant normals using shape prior from (a) & (b).*

RGB Data Cost: Let $Z_{D_{rgb}}$ denote the number of pixels covered by hole patches. The RGB data cost is defined so that the selected known patch closely matches the hole patch in the RGB image:

$$\mathcal{C}_{D_{rgb}}(\mathbb{H}) = \frac{1}{3Z_{D_{rgb}}} \sum_{l} \sum_{\mathbf{p} \in \mathbf{H}_l} \|I(H_l(\mathbf{p})) - I(K_{H_l^{-1}}(\mathbf{p}))\|^2. \tag{6.4}$$

Depth Gradient Data Cost: Let $Z_{D_{dg}}$ be the number of non-hole pixels covered by hole patches, and D' as the depth gradient image. Since these pixels have depth values, their depth gradients can be calculated. The depth gradient data cost favors solutions in which the computed depth gradients closely match the original depth gradients for the non-hole pixels:

$$\mathcal{C}_{D_{dg}}(\mathbb{H}) = \frac{1}{2Z_{D_{dg}}} \sum_{l} \sum_{\mathbf{p} \in \mathbf{H}_l} \alpha(H_l(\mathbf{p})) \|D'(H_l(\mathbf{p})) - D'(K_{H_l^{-1}}(\mathbf{p}))\|^2 \tag{6.5}$$

where

$$\alpha(H_l(\mathbf{p})) = \begin{cases} 1 & H_l(\mathbf{p}) \text{ has a depth gradient} \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

RGB Smoothness Cost: Suppose $\{\mathbf{H}_{l1}, \mathbf{H}_{l2}\}$ is a pair of overlapping hole patches, and

$\mathbf{K}_{H_{l1}^{-1}}$ and $\mathbf{K}_{H_{l2}^{-1}}$ respectively denote their repairing known patches. Suppose also that pixel $\mathbf{p}_a$ of $\mathbf{K}_{H_{l1}^{-1}}$ coincides with pixel $\mathbf{p}_b$ of $\mathbf{K}_{H_{l2}^{-1}}$, when $\mathbf{K}_{H_{l1}^{-1}}$ and $\mathbf{K}_{H_{l2}^{-1}}$ are pasted onto $\mathbf{H}_{l1}$ and $\mathbf{H}_{l2}$. With Z_{ov} being the number of pixels in the overlapping regions of hole patches, we penalize solutions where the overlapping RGB values are inconsistent:

$$\mathcal{C}_{S_{rgb}}(\mathbb{H}) = \frac{1}{3Z_{ov}} \sum_{\{\mathbf{H}_{l1},\mathbf{H}_{l2}\}} \sum_{\{\mathbf{p}_a,\mathbf{p}_b\}} \|I(K_{H_{l1}^{-1}}(\mathbf{p}_a)) - I(K_{H_{l2}^{-1}}(\mathbf{p}_b))\|^2. \tag{6.7}$$

Depth Gradient Smoothness Cost: Similar to the RGB smoothness cost, we have a corresponding cost for the depth gradient image:

$$\mathcal{C}_{S_{dg}}(\mathbb{H}) = \frac{1}{2Z_{ov}} \sum_{\{\mathbf{H}_{l1},\mathbf{H}_{l2}\}} \sum_{\{\mathbf{p}_a,\mathbf{p}_b\}} \|D'(K_{H_{l1}^{-1}}(\mathbf{p}_a)) - D'(K_{H_{l2}^{-1}}(\mathbf{p}_b))\|^2. \tag{6.8}$$

After belief propagation is performed to minimize $\mathcal{C}_{BP}(\mathbb{H})$, depth gradients of pixels within hole patches are replaced by depth gradients from the assigned known patches. With the transferred depth gradients and the known depth values along the hole boundary as boundary conditions, poisson integration (Pérez et al., 2003) is used to fill in the depth values of the hole. Figure 6.6 illustrates this process.

6.3.3.2 Surface Normal Refinement

The estimated relative albedos, lighting and shape prior serve as useful inputs for normal refinement over the whole scene. Suppose there are in total Z_{total} pixels. The surface normal refinement is formulated as a non-linear optimization using the total energy function:

$$\begin{aligned}\mathcal{E}(N) = w_{sfs}\mathcal{E}_{sfs}(N) + w_{prior}\mathcal{E}_{prior}(N) + \\ w_{smooth}\mathcal{E}_{smooth}(N) + w_{norm}\mathcal{E}_{norm}(N).\end{aligned} \tag{6.9}$$

$\mathcal{E}_{sfs}(N)$ is the shape-from-shading cost represented using 2nd order spherical harmonics. It constrains the normal according to the shading observed in the RGB image:

$$\mathcal{E}_{sfs}(N) = \frac{1}{Z_{total}} \sum_{i} \sum_{k=\{1,2,3\}} (I_{i,k} - p_{i,k} \hat{n_i}^T M_k \hat{n_i})^2 \tag{6.10}$$

To resolve bas-relief ambiguity, $\mathcal{E}_{prior}(N)$ constrains the normals to be similar to the prior normals computed from the repaired Kinect depth map (see Figure 6.4). Denote the prior normal as n'_i:

$$\mathcal{E}_{prior}(N) = \frac{1}{Z_{total}} \sum_{i} \|n_i - n'_i\|^2. \tag{6.11}$$

$\mathcal{E}_{smooth}(N)$ is a smoothness term with respect to 1st-order neighbors. For the set of 1st-order neighbors, $\{i_1, i_2\}$, we have:

$$\mathcal{E}_{smooth}(N) = \frac{1}{Z_{total}} \sum_{\{i_1,i_2\}} \|n_{i_1} - n_{i_2}\|^2. \tag{6.12}$$

Finally, $\mathcal{E}_{norm}(N)$ is the norm regularization which constrains the normals to be of unit length:

$$\mathcal{E}_{norm}(N) = \frac{1}{Z_{total}} \sum_{i} (n_i^T n_i - 1)^2. \tag{6.13}$$

The total energy function $\mathcal{E}(N)$ is a weighted sum of the four energy terms, with the weights fixed to $w_{sfs} = 1.0$, $w_{prior} = 0.1$, $w_{smooth} = 0.05$ and $w_{norm} = 0.05$. The total energy function, which is non-linear in terms of normals n_i, is optimized by the trust-region-reflective algorithm. We initialize the normals to $[0, 0, 1]^T$, facing the camera.

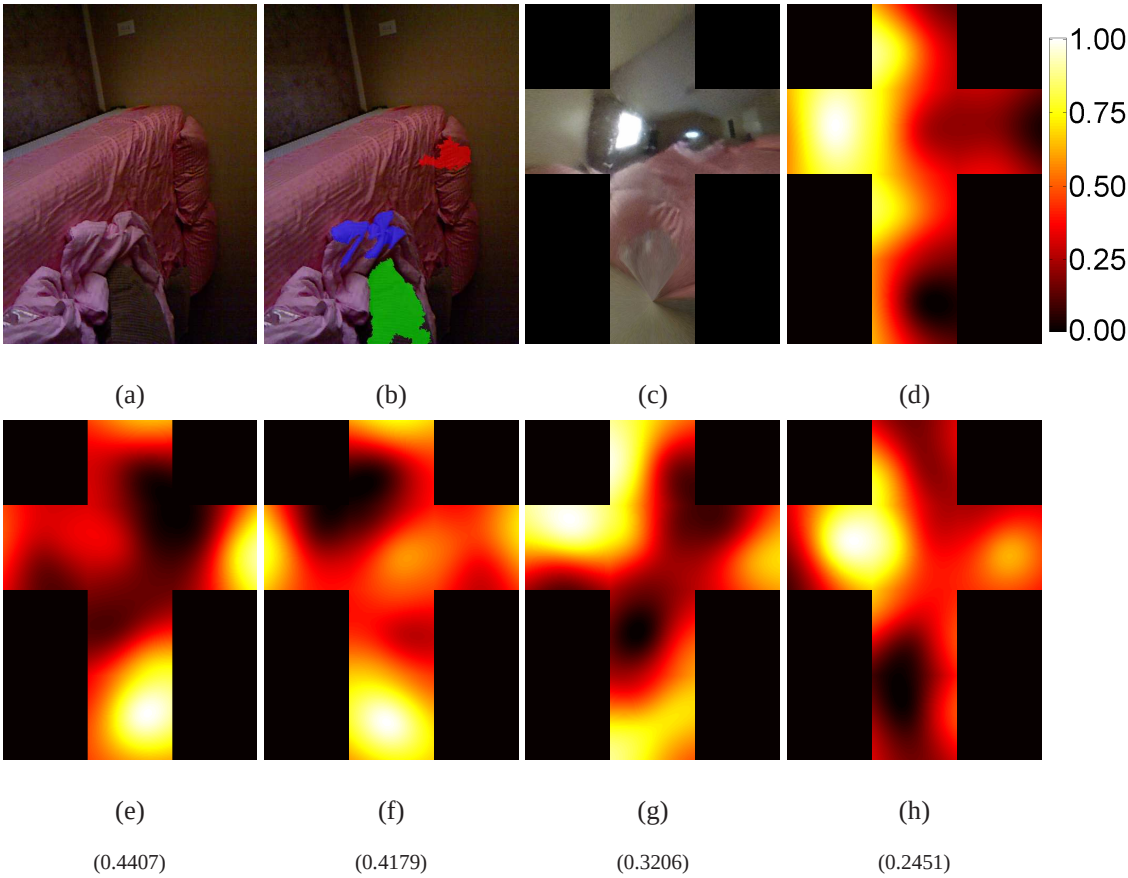


Figure 6.8: *Light estimation experiment. (a) Input scene. (b) Clusters colored for illustration. (c) Ground truth environment map. (d) Ground truth 2nd-order SH. (e) Estimation by red cluster in (b). (f) Estimation by green cluster. (g) Estimation by blue cluster. (h) Estimation by all regions in the MST. Bracketed numbers show RSME.*

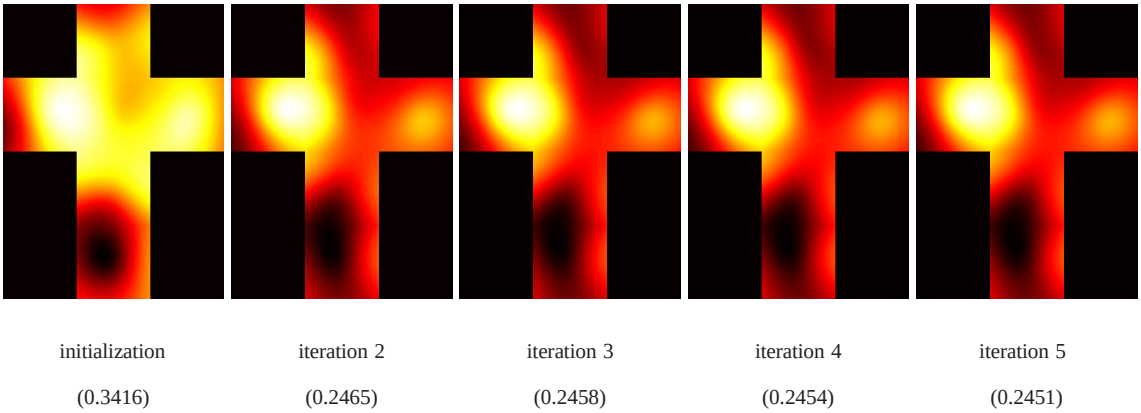
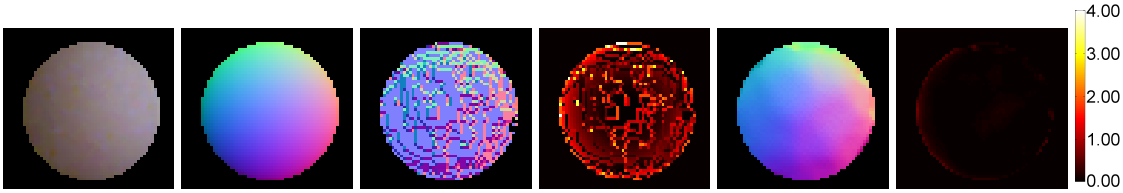


Figure 6.9: *Iterative refinement of light estimation throughout AO. Bracketed numbers show RMSE, which is converging.*



(a) (b) (c) (d) (e) (f)

Figure 6.10: *Normal estimation of a Lambertian ball in the scene,* jeans. *(a) Input image. (b) Ground-truth normal map. (c) Raw normal map. (d) Squared error map of raw normals (RMSE=0.5178), (e) Estimated normal map, (f) Squared error map of estimated normals (RMSE=0.1401).*

6.4 Results

6.4.1 Lighting Estimation

In Figure 6.8, we investigate our approach's ability to estimate environment light in an indoor scene, by comparing it to ground truth obtained using a mirrored sphere convolved with 2nd-order spherical harmonics. It can be observed that using more clusters and normals, which is made possible by the relative albedo estimation, leads to more accurate and robust light estimation. As the normals throughout the MST are used, the major light directions and intensity resemble that obtained from the mirrored sphere. Figure 6.9 also shows iterative refinement of light estimation throughout the alternating optimization process. We note that inconsistency in the environment light across the scene due to non-distant light sources will contribute to error.

6.4.2 Ground Truth Comparison

Next we validate our approach by conducting an analytical experiment in which we estimate normals of a Lambertian ball in an indoor scene (named *jeans* in the supplement).

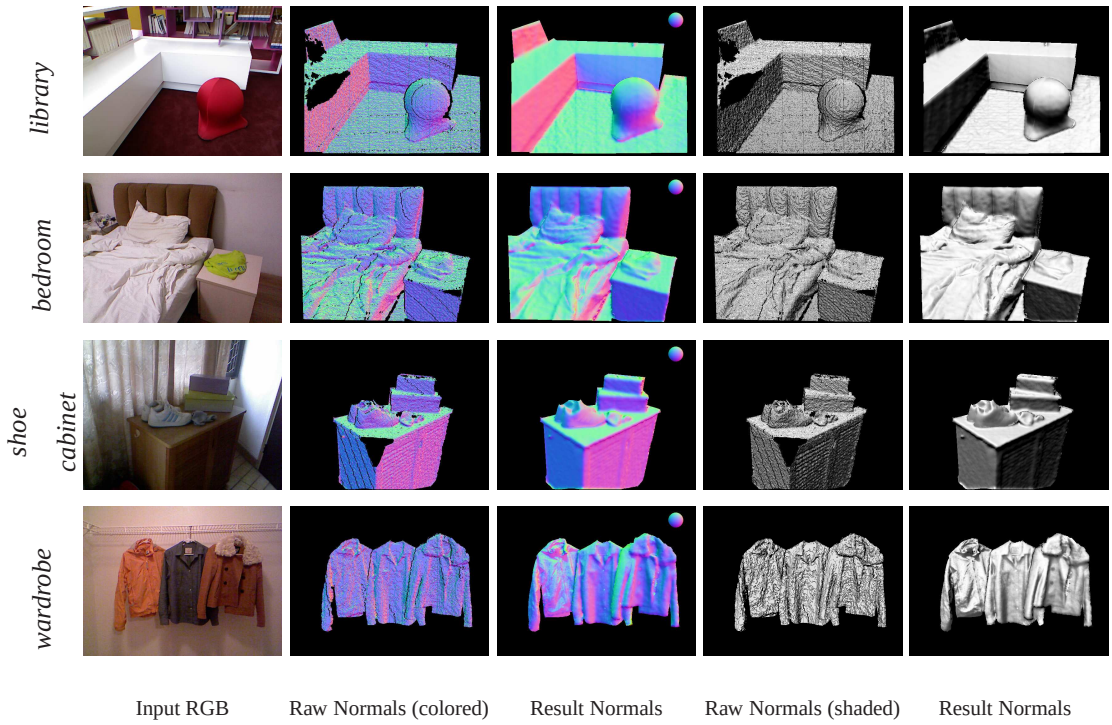


Figure 6.11: *Kinect scenes repaired by our approach.*

Figure 6.10 shows the results of our approach in refining the raw normals computed directly from the depth map. The RMSE is improved from 0.5718 to 0.1401[2]. The more apparent error along the sphere boundary is due to the greater noise in Kinect RGB images near object boundaries.

6.4.3 Repairing Kinect Scenes

We tested our approach on four indoor scenes captured by Kinect, namely, *library*, *bedroom*, *shoe cabinet* and *wardrobe*. These are common indoor scenes with shading detail that our approach can make use of to refine the reconstructed surface. Figure 6.11 shows the results. In *library*, the *structural holes* on the books and shelf are repaired by the propagated patches, and the round surface of the stool is well reconstructed by

[2]While the RMSE of relative light intensity is in the range $[0, 1]$, the RMSE of normals is in the range $[0, 2]$, as the squared error of normals is in range $[0, 4]$. For example, normals $[0, 0, 1]^T$ and $[0, 0, -1]^T$ result in a maximum squared error of 4.

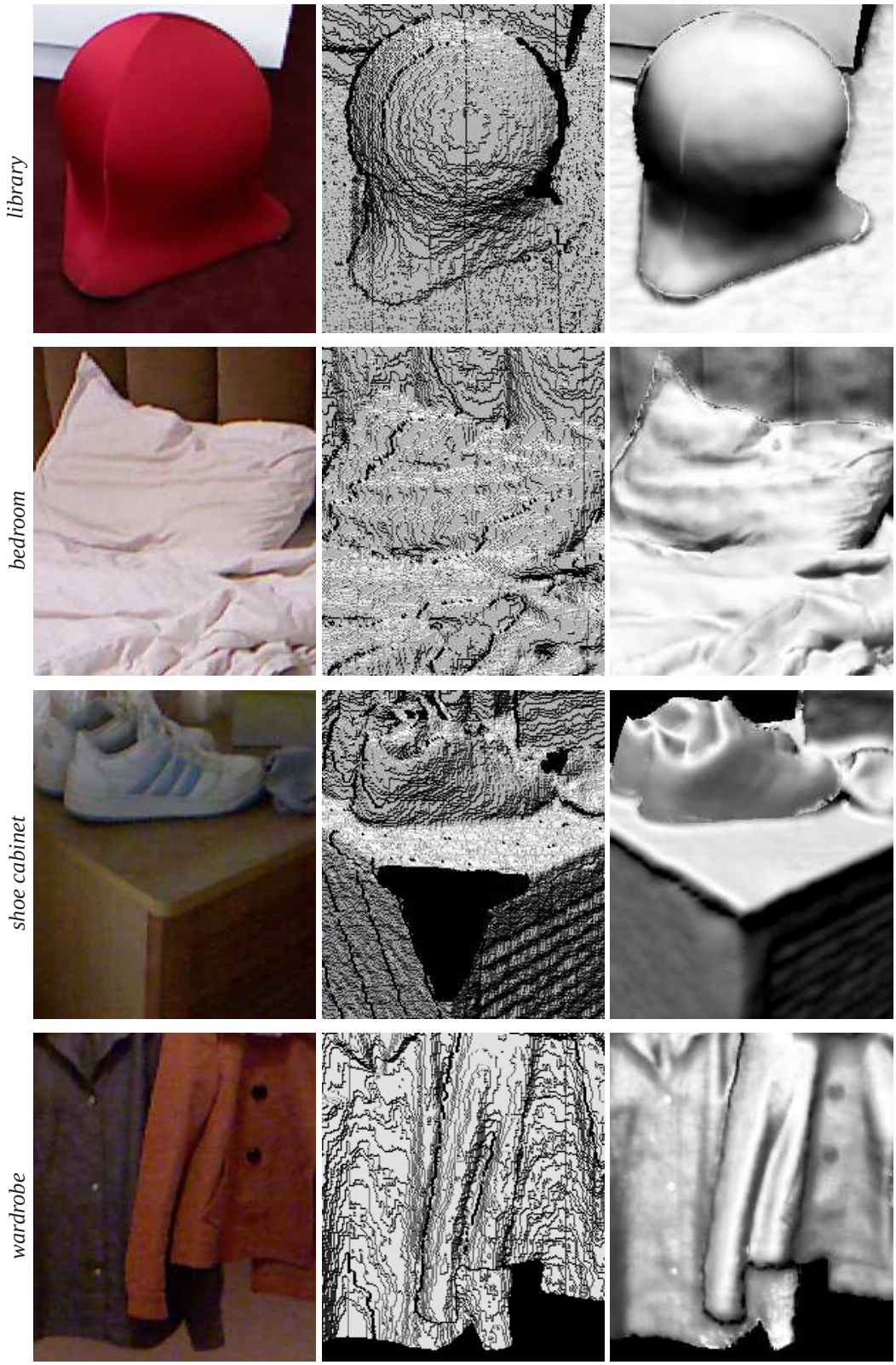


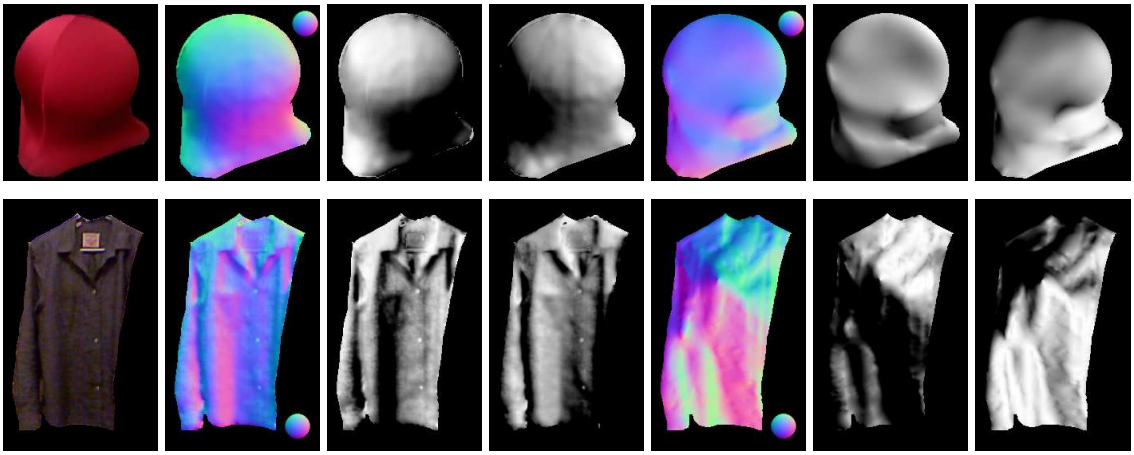
Figure 6.12: *Zoom-in Views.*

shading despite the presence of noise and holes in the input depth and normal map. In *bedroom*, details of the pillow are faithfully reconstructed, e.g., the crease at the top-right corner. In *shoe cabinet*, structural propagation enables the proper repair of the hole at the corner, which provides a correct shape prior compared to smoothing (see also Figure 6.7). To this, shading adds further details, e.g., the marks on the shoe. Finally, in *wardrobe*, shape-from-shading significantly improves the surface where very fine details such as the folded collar and button regions can be clearly seen. Please refer to the supplementary materials for three additional results.

6.4.4 Comparison with Other Methods

To demonstrate the possible improvements obtainable with noisy Kinect depth data in our method, we compare our depth-assisted approach with a state-of-the-art shape-from-shading algorithm (Barron and Malik, 2012), which operates with only an RGB image using generic albedo and illumination priors. As shown in Figure 6.13, our depth-assisted method achieves significantly better surface normal reconstructions. We believe that the priors used in (Barron and Malik, 2012) may be more appropriate for single objects than for full scenes that are captured by a Kinect. In this comparison, we used the code provided in (Barron and Malik, 2012) with the default parameters. Our approach uses only the regions with the highest-confidence relative albedos (from the MST) for lighting estimation, rather than the entire image. Our supplement contains additional results.

Figure 6.14 compares our albedo normalization result with the state-of-the-art intrinsic image separation technique of (Lee et al., 2012), which also makes use of Kinect depth data. The result of (Lee et al., 2012) was provided to us by the authors. Their work assumes the input to be a nearly flawless depth map obtained from video streams of a moving Kinect, and does not operate as well with a noisy depth map available from a single Kinect image. In contrast, our technique performs more effective albedo nor-



(a) (b) (c) (d) (e) (f) (g)

Figure 6.13: *Comparison to SfS technique of (Barron and Malik, 2012). (a) Input RGB image. (b-d) Our recovered normals and two normal maps* $\mathbf{N}$ *shaded as* $\mathbf{N} \cdot \mathbf{L}$ *with* $\mathbf{L} = (-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})^T$ *and* $\mathbf{L} = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})^T$*. (e-g) Recovered normals and shaded images of (Barron and Malik, 2012) using generic albedo and illumination priors.*

malization because the relative albedos are obtained with the help of estimated lighting. This results in more refined shading details, e.g., on the bed.

High-quality normals are vital prerequisites for different practical applications. Figure 6.15 shows a point cloud significantly refined with our resultant normals using the method of (Lu et al., 2010b). In addition, the resultant normals enable realistic relighting and high-quality 3D surface reconstruction.

6.5 Summary, Discussion, and Future Work

In this chapter, we presented a useful postprocessing method to improve the quality of surface normals obtained from Kinect. When used with the latest Kinect, which has higher resolution in RGB than in depth, the proposed method could also be utilized for the problem of depth map denoising and upsampling (Yang et al., 2007; Dolson et al.,

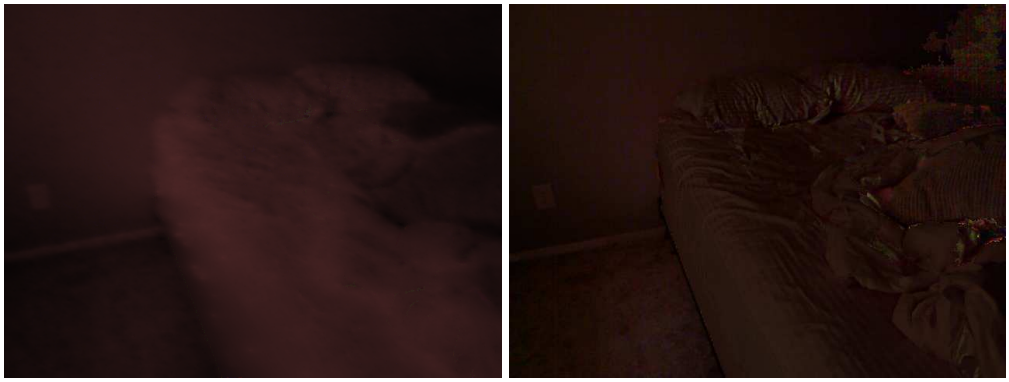


Figure 6.14: *Left: albedo normalization result of Figure 6.8 by (Lee et al., 2012). Right: our result.*

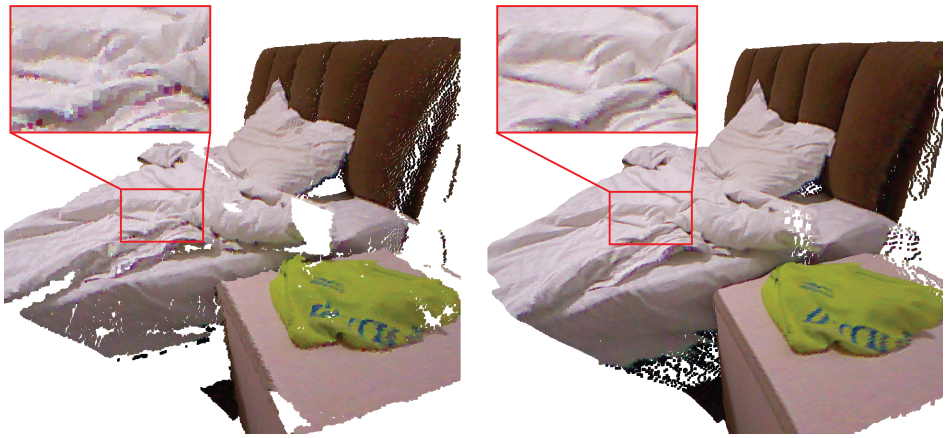


Figure 6.15: *Left: raw point cloud. Right: point cloud refined with our resultant normals.*

2010; Park et al., 2011), since the geometry is solved at the RGB image resolution and its use of shading significantly reduces the effects of depth sensor noise.

Like other patch-based image completion methods, the effectiveness of our patch-based hole repairing step is subject to the quality and compatibility of the surrounding known patches. While the RGB data is in general of higher quality than the depth data, its noise can still affect the quality of shape-from-shading. For scenes with local light sources, the environment light may differ significantly in different parts of the scene. This issue could potentially be addressed by solving for the environment light separately among local regions.

In future work, we plan to consider the lighting visibility of scene points based on the depth map, as this should improve the estimation of lighting, relative albedos, and shape-from-shading.

CHAPTER 7

Conclusion

7.1 General Summary

This dissertation has demonstrated how data-driven optimization approaches can be applied to tackle the modeling problem. From the graphics perspective, novel approaches have been proposed for virtual world modeling, virtual character modeling, and interactive indoor scene modeling. The proposed approaches share the advantage that useful relationships are learnt from real-world or human-annotated data, which can then be used to guide the optimization, or provide useful suggestions in the user interface, in order to accomplish the modeling task. This results in the generation of different realistic models that can be used for 3D graphics applications. From the vision perspective, novel approaches were proposed to enable accurate 3D surface reconstruction from real-world data. The approaches enable photometric stereo to be performed outdoors, and the use of a single RGB-D image captured with a low-cost depth camera to obtain a high quality 3D surface showing subtle details through shading analysis.

7.2 Future Trends in Data-Driven 3D Modeling

Recent R&D advances in 3D scanning, 3D display, and 3D printing have made these technologies increasingly accessible to the general public. Through data-driven modeling approaches, such data can be readily utilized to create high-quality 3D models

for various applications. With a 3D display, one can visualize 3D models like real-world physical objects. Advances in haptic devices may even allow us to touch and feel these virtual 3D objects. Furthermore, one can often easily fabricate physical 3D objects using a 3D printer, so that everyone can become a product designer as well as a manufacturer. This possibility is very exciting as it will open up many opportunities for research into and applications of data-driven 3D modeling.

Low-cost depth cameras (e.g., Microsoft's Kinect, which comes with the Xbox game console) have already become widely available. Equipped with such devices, one can readily acquire 3D data at home. In the near future every household will likely be equipped with a 3D scanner, a 3D display, and a 3D printer as well. This will probably transform human-computer interaction, as well as home entertainment and manufacturing.

These trends foretell an even stronger future role for data-driven 3D modeling. The widespread popularity of 3D data acquisition devices means that we will have more and more 3D data, analogous to how we now have copious quatities of 2D image data thanks to the widespread availability of digital cameras. This provides good grounds for data-driven modeling approaches. On the other hand, the created 3D models will no longer reside solely in the 2D display screen. The demand to visualize, touch, and feel the created 3D objects will lead to a higher expectation on the quality of the 3D models and additional factors to be considered. For example, the fabrication of 3D models must satisfy important physical, material, and mechanical properties to sustain physical and usability requirements in the real-world, which are not normally considered in conventional 3D geometric modeling. This is exciting as it opens up many opportunities for research (Prévost et al., 2013; Skouras et al., 2013; Coros et al., 2013; Zhu et al., 2012). Our research on data-driven modeling interfaces intended for general users, which ease the 3D modeling process, also closely aligns with the trend of personalized product design and home manufacturing. Such promising trends will also give impetus

to more sophisticated creative activities at the household and small business levels; e.g., to create video games, movies, products, and other 3D applications.

BIBLIOGRAPHY

Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, New York, NY. 38

Abrams, A., Hawley, C., and Pless, R. (2012). Heliometric stereo: Shape from sun position. In *European Conference on Computer Vision (ECCV)*. 121

Ackermann, J., Langguth, F., Fuhrmann, S., and Goesele, M. (2012). Photometric stereo for outdoor webcams. In *CVPR*, pages 262–269. 121

Akazawa, Y., Okada, Y., and Niijima, K. (2005). Automatic 3D scene generation based on contact constraints. In *Proc. Conf. on Computer Graphics and Artificial Intelligence*, pages 593–598. 28

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Science*, 50(1):5–43. 79

Ballard, D. and Brown, C. (1982). Computer vision. In *Prentice Hall*. 123

Bao, F., Schwarz, M., and Wonka, P. (2013a). Procedural facade variations from a single layout. *ACM Trans. Graph.*, 32(1):8:1–8:13. 32

Bao, F., Yan, D.-M., Mitra, N. J., and Wonka, P. (2013b). Generating and exploring good building layouts. *ACM Trans. Graph.*, 32(4). 102

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA. ACM Press. 61

Barron, J. T. and Malik, J. (2012). Color constancy, intrinsic images, and shape estimation. In *ECCV (4)*, pages 57–70. 137, 154, 155

Barsky, S. and Petrou, M. (2003). The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows. *IEEE Trans. on PAMI*, 25(10):1239–1252. 118, 120

Basri, R., Jacobs, D. W., and Kemelmacher, I. (2007). Photometric stereo with general, unknown lighting. *IJCV*, 72(3):239–257. 118, 120, 121

Belhumeur, P. N., Kriegman, D. J., and Yuille, A. L. (1999). The bas-relief ambiguity. *IJCV*, 35(1):33–44. 144

Bokeloh, M., Wand, M., Koltun, V., and Seidel, H.-P. (2011). Pattern-aware shape deformation using sliding dockers. *ACM Trans. Graph.*, 30(6):123:1–123:10. 5

Bresson, X. and Chan, T. F. (2008). Fast dual minimization of the vectorial total variation norm and applications to color image processing. Technical report, UCLA CAM Report. 119, 124, 128

Bukowski, R. W. and Séquin, C. H. (1995). Object associations: A simple and practical approach to virtual 3D manipulation. In *Symposium on Interactive 3D Graphics*. 100, 103

Chai, M., Wang, L., Weng, Y., Yu, Y., Guo, B., and Zhou, K. (2012). Single-view hair modeling for portrait manipulation. *ACM Trans. Graph.*, 31(4):116:1–116:8. 23

Chajdas, M. G., Lefebvre, S., and Stamminger, M. (2010). Assisted texture assignment. In *Symposium on Interactive 3D Graphics*. 102

Chandraker, M., Bai, J., and Ramamoorthi, R. (2011). A theory of differential photometric stereo for unknown BRDFs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2505–2512. 118, 120

Chaudhuri, S., Kalogerakis, E., Guibas, L., and Koltun, V. (2011a). Probabilistic reasoning for assembly-based 3D modeling. *ACM Transactions on Graphics*, 30(4). 5, 19, 66

Chaudhuri, S., Kalogerakis, E., Guibas, L. J., and Koltun, V. (2011b). Probabilistic reasoning for assembly-based 3D modeling. *ACM Trans. Graph.*, 30(4). 102

Chaudhuri, S. and Koltun, V. (2010). Data-driven suggestions for creativity support in 3D modeling. *ACM Trans. Graph.*, 29(6). 102

Chen, G., Esch, G., Wonka, P., Müller, P., and Zhang, E. (2008). Interactive procedural street modeling. *ACM Trans. Graph.*, 27(3). 31

Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335. 38

Ching, F. D. and Binggeli, C. (2005). *Interior Design Illustrated, 2nd Ed.* Wiley, New York, NY. 30, 31, 43, 44, 55, 56

Chun, H. W. and Lai, E. M.-K. (1997). Intelligent critic system for architectural design. *IEEE Trans. Knowl. Data Eng.*, 9:625–639. 30

Cohen-Or, D., Sorkine, O., Gal, R., Leyvand, T., and Xu, Y.-Q. (2006). Color harmonization. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 25(3):624–630. 63

Coleman, Jr., E. and Jain, R. (1982). Obtaining 3-dimensional shape of textured and specular surfaces using four-source photometry. *CGIP*, 18(4):309–328. 118, 120

Coros, S., Thomaszewski, B., Noris, G., Sueda, S., Forberg, M., Sumner, R., Matusik, W., and Bickel, B. (2013). Computational design of mechanical characters. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):to appear. 159

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press. 72

de Aguiar, E., Sigal, L., Treuille, A., and Hodgins, J. K. (2010). Stable spaces for real-time clothing. *ACM Trans. Graph.*, 29(4). 61

Debevec, P. E. (1998). Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH*, pages 189–198. 123, 129

Deng, J., Dong, W., Socher, R., jia Li, L., Li, K., and Fei-fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *In CVPR*. 13

Dobbyn, S., McDonnell, R., Kavan, L., Collins, S., and O'Sullivan, C. (2006). Clothing the Masses: Real-Time Clothed Crowds With Variation. In *Eurographics Short Papers (EG'06)*, pages 103 – 106. 62

Dolson, J., Baek, J., Plagemann, C., and Thrun., S. (2010). Upsampling range data in dynamic environments. In *CVPR*. 138, 155

Durou, J.-D., Falcone, M., and Sagona, M. (2008). Numerical methods for shape-from-shading: A new survey with benchmarks. *CVIU*, 109(1):22–43. 136, 138

Feng, W.-W., Yu, Y., and Kim, B.-U. (2010). A deformation transformer for real-time cloth animation. *ACM Trans. Graph.*, 29(4). 61

Fischer-Mirkin, T. (1995). *Dress Code: Understanding the Hidden Meanings of Women's Clothes.* Clarkson Potter, New York, NY. 64

Fisher, M. and Hanrahan, P. (2010). Context-based search for 3D models. *ACM Trans. Graph.*, 29(6). 101

Fisher, M., Ritchie, D., Savva, M., Funkhouser, T. A., and Hanrahan, P. (2012). Example-based synthesis of 3D object arrangements. *ACM Trans. Graph.*, 31(6). 29, 102

Fisher, M., Savva, M., and Hanrahan, P. (2011). Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph.*, 30(4). 101

Flusser, A. J. (2002). *Dressing the Man: Mastering the Art of Permanent Fashion.* HarperCollins. 64

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163. 70

Funt, B. V. and Finlayson, G. D. (1995). Color Constant Color Indexing. *PAMI*, 17(5):522–529. 140

Gal, R., Sorkine, O., Mitra, N. J., and Cohen-Or, D. (2009). iwires: an analyze-and-edit approach to shape manipulation. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 33:1–33:10, New York, NY, USA. ACM. 5

Gallistel, C. R. (2009). The importance of proving the null. *Psychological Review*, 116(2):439–453. 53

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741. 38

Germer, T. and Schwarz, M. (2009). Procedural arrangement of furniture for real-time walkthroughs. *Computer Graphics Forum*, 28(8):2068–2078. 28, 29, 33

Gilchrist, R. A. (2011). *The Encyclopedia of Men's Clothes*. 63, 64

Godsill, S. J. (2001). On the relationship between markov chain monte carlo methods for model uncertainty. *Journal Of Computational And Graphical Statistics*, 10(2):1–19. 79

Goldman, D. B., Curless, B., Hertzmann, A., and Seitz, S. M. (2010). Shape and spatially-varying BRDFs from photometric stereo. *IEEE Trans. on PAMI*, 32(6):1060–1071. 120, 127

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732. 78

Green, P. J. (2003). *Highly structured stochastics systems*, chapter Trans-dimensional Markov Chain Monte Carlo, pages 179–196. Oxford University Press, Oxford, UK. 79

Grunwald, P. (2007). *The Minimum Description Length Principle*. MIT Press. 35

Guan, P., Reiss, L., Hirshberg, D. A., Weiss, A., and Black, M. J. (2012). Drape: Dressing any person. *ACM Trans. Graph.*, 31(4):35. 61

Hastings, W. (1970). Monte Carlo samping methods using Markov chains and their applications. *Biometrika*, pages 97–109. 38, 78

Henderson, V. and Henshaw, P. (2008). *Color Me Confident: Change Your Look - Change Your Life!*. Hamlyn, London, UK. 59, 63

Hertzmann, A. and Seitz, S. (2003). Shape and materials by example: a photometric stereo approach. In *CVPR*, pages I: 533–540. 120

Horn, B. (1986). *Robot Vision*. McGraw-Hill. 118, 120

Horn, B. K. P. and Brooks, M. J. (1989). *Shape from shading*. MIT Press, Cambridge, MA, USA. 136, 138

Huang, R. and Smith, W. A. P. (2011). Shape-from-shading under complex natural illumination. In *ICIP*, pages 13–16. 138

Igarashi, T. and Hughes, J. F. (2001). A suggestive interface for 3D drawing. In *UIST*. 102

Igarashi, T., Matsuoka, S., and Tanaka, H. (1999). Teddy: A sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co. 5

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. (2011). Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559–568, New York, NY, USA. ACM. 23

Jackson, C. (1987). *Color Me Beautiful.* Ballantine Books, New York, NY. 58, 59, 63, 71, 72

Jackson, C. and Lulow, K. (1984). *Color for Men.* Ballantine Books, New York, NY. 59, 63

Jagnow, R., Dorsey, J., and Rushmeier, H. (2008). Evaluation of methods for approximating shapes used to synthesize 3D solid textures. *ACM Trans. Appl. Percept.*, 4(4):1–27. 49, 89

Jain, A., Thormählen, T., Ritschel, T., and Seidel, H.-P. (2012). Material memex: automatic material suggestions for 3D objects. *ACM Trans. Graph.*, 31(6). 102

Jimenez, J., Sundstedt, V., and Gutierrez, D. (2009). Screen-space perceptual rendering of human skin. *ACM Trans. Appl. Percept.*, 6(4):1–15. 49, 89

Johnson, M. K. and Adelson, E. H. (2011). Shape estimation in natural illumination. In *CVPR*, pages 2553–2560. 121, 125, 136, 137

Kaldor, J. M., James, D. L., and Marschner, S. (2008). Simulating knitted cloth at the yarn level. *ACM Trans. Graph.*, 27(3). 61

Kaldor, J. M., James, D. L., and Marschner, S. (2010). Efficient yarn-based cloth with adaptive contact linearization. *ACM Trans. Graph.*, 29(4). 61

Kavan, L., Gerszewski, D., Bargteil, A. W., and Sloan, P.-P. (2011). Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph.*, 30(4):93. 61

Kirkpatrick, S. (1984). Optimization by Simulated Annealing: Quantitative Studies. *Journal of Statistical Physics*, 34(5):975–986. 38

Kjlaas, K. A. H. (2000). Automatic furniture population of large architectural models. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge, MA. 28, 33

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press. 67

Larive, M., Roux, O. L., and Gaildrat, V. (2004). Using meta-heuristics for constraint-based 3D objects layout. In *Proc. Conf. on Computer Graphics and Artificial Intelligence*, pages 11–23. 28

Lee, K. J., Zhao, Q., Tong, X., Gong, M., Izadi, S., Lee, S. U., Tan, P., and Lin, S. (2012). Estimation of intrinsic image sequences from image+depth video. In *ECCV*, pages 327–340. 154, 156

Lee, Y. J., Zitnick, C. L., and Cohen, M. F. (2011). Shadowdraw: real-time user guidance for freehand drawing. *ACM Trans. Graph.*, 30(4). 102

Lin, Z., Ganesh, A., Wright, J., Wu, L., Chen, M., and Ma, Y. (2009). Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. Technical Report UILU-ENG-09-2214, UIUC. 119, 124, 125

Liu, J. S. (2008). *Monte Carlo Strategies in Scientific Computing*. Springer, New York NY. 38

Lu, Z., Tai, Y.-W., Ben-Ezra, M., and Brown, M. S. (2010a). A framework for ultra high resolution 3D imaging. In *CVPR*. 118, 120

Lu, Z., Tai, Y.-W., Ben-Ezra, M., and Brown, M. S. (2010b). A framework for ultra high resolution 3D imaging. In *CVPR*. 155

Malnar, J. M. and Vodvarka, F. (1992). *The Interior Dimension: A Theoretical Approach to Enclosed Space*. Wiley, New York, NY. 55, 56

McDonnell, R., Dobbyn, S., and O'Sullivan, C. (2006). Crowd Creation Pipeline for Games. In *International Conference on Computer Games (CGames'06)*, pages 183 – 190. 62

McDonnell, R., Larkin, M., Dobbyn, S., Collins, S., and O'Sullivan, C. (2008). Clone attack! Perception of crowd variety. *ACM Trans. Graph.*, 27(3). 62

McDonnell, R., Larkin, M., Hernández, B., Rudomín, I., and O'Sullivan, C. (2009). Eye-catching crowds: saliency based selective variation. *ACM Trans. Graph.*, 28(3). 62

Merrell, P., Schkufza, E., and Koltun, V. (2010). Computer-generated residential building layouts. *ACM Trans. Graph.*, 29:181:1–181:12. 21, 30, 38, 66, 68

Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011a). Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87. 19

Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011b). Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4). 100, 103

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092. 38, 78

Mitton, M. and Nystuen, C. (2007). *Residential Interior Design: A Guide to Planning Spaces*. Wiley, New York, NY. 31, 43

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3):614–623. 5, 31

Müller, P., Zeng, G., Wonka, P., and Van Gool, L. (2007). Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3):85. 32

Nayar, S., Ikeuchi, K., and Kanade, T. (1990). Determining shape and reflectance of hybrid surfaces by photometric sampling. *IEEE Trans. on Robotics and Automation*, 6(4):418–431. 120

Nayar, S. K. and Bolle, R. M. (1996). Reflectance based object recognition. *IJCV*, 17(3):219–240. 140

Newell, M. E. and Blinn, J. F. (1977). The progression of realism in computer generated images. In *Proceedings of the 1977 Annual Conference*, ACM. 98

Nicholson, J. (2003). *Dressing smart for women: 101 mistakes you can't afford to make and how to avoid them*. Impact Publications, Manassas, VA. 58, 63

O'Donovan, P., Agarwala, A., and Hertzmann, A. (2011). Color compatibility from large datasets. *ACM Trans. Graph.*, 30(4):63. 63, 66, 72, 75, 89

O'Sullivan, C. (2009). Variety is the spice of (virtual) life. In *MIG*, pages 84–93. 62, 89

Oxholm, G. and Nishino, K. (2012a). Shape and reflectance from natural illumination. In *ECCV (1)*, pages 528–541. 121

Oxholm, G. and Nishino, K. (2012b). Shape and reflectance from natural illumination. In *ECCV (1)*, pages 528–541. 136, 137

Panero, J. and Zelnick, M. (1979). *Human Dimension and Interior Space: A Source Book of Design Reference Standards*. Watson-Guptill, New York, NY. 30

Parish, Y. I. H. and Müller, P. (2001). Procedural modeling of cities. In *Proc. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pages 301–308, New York, NY. ACM. 5, 31

Park, J., Kim, H., Tai, Y.-W., Brown, M., and Kweon, I. (2011). High quality depth map upsampling for 3D-TOF cameras. In *ICCV*. 138, 157

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 67

Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318. 148

Prévost, R., Whiting, E., Lefebvre, S., and Sorkine-Hornung, O. (2013). Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):to appear. 159

Provot, X. (1995). Deformation constraints in a mass–spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pages 147–154. 61

Prusinkiewicz, P. and Lindenmayer, A. (1996). *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA. 5

Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., and Kang, S. B. (2006). Image-based plant modeling. *ACM Trans. Graph.*, 25(3):599–604. 23

Ramamoorthi, R. and Hanrahan, P. (2001a). An efficient representation for irradiance environment maps. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2001)*, 20(3):497–500. 121

Ramamoorthi, R. and Hanrahan, P. (2001b). An efficient representation for irradiance environment maps. In *SIGGRAPH*. 137, 143

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., and Iverson, G. (2009). Bayesian t-tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin and Review*, 16:225–237. 53

Rudin, L., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268. 119, 124

Sanchez, S., Roux, O., Luga, H., and Gaildrat, V. (2003). Constraint-based 3D-object layout using a genetic algorithm. In *Proc. Conf. on Computer Graphics and Artificial Intelligence*. 28

Schneider, J. J. and Kirkpatrick, S. (2006). *Stochastic Optimization (Scientific Computation)*. Springer-Verlag, Berlin. 38, 78

Schoeffler, O. E. and Gale, W. (1973). *Esquire's encyclopedia of 20th century men's fashions.* McGraw-Hill, New York, NY. 64

Shao, W. and Terzopoulos, D. (2007). Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274. 31

Shi, B., Matsushita, Y., Wei, Y., Xu, C., and Tan, P. (2010). Self-calibrating photometric stereo. In *CVPR*. 120, 122, 127

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*. 13, 105

Skouras, M., Thomaszewski, B., Coros, S., Bickel, B., and Gross, M. (2013). Computational design of actuated deformable characters. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):to appear. 159

Solomon, F. and Ikeuchi, K. (1996). Extracting the shape and roughness of specular lobe objects using four light photometric stereo. *IEEE Trans. on PAMI*, 18(4):449–454. 118, 120

Sondag, G. R. (2011). *Anything Other Than Naked - A guide for men on how to dress properly for every occassion.* Two Harbors Press. 64

Stava, O., Benes, B., Mech, R., Aliaga, D. G., and Kristof, P. (2010). Inverse procedural modeling by automatic generation of L-systems. *Computer Graphics Forum*, 29(2):665–674. 6

Sun, J., Yuan, L., Jia, J., and Shum, H.-Y. (2005). Image completion with structure propagation. *ACM Trans. Graph.*, 24(3). 146

Sunkavalli, K., Zickler, T., and Pfister, H. (2010). Visibility subspaces: Uncalibrated photometric stereo with shadows. In *ECCV*, pages 251–264. 118, 120, 122

Tagare, H. and deFigueiredo, R. (1991). A theory of photometric stereo for a class of diffuse non-lambertian surfaces. *IEEE Trans. on PAMI*, 13(2):133–152. 118, 120

Talton, J. O., Gibson, D., Yang, L., Hanrahan, P., and Koltun, V. (2009). Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics*, 28(5):167:1–167:10. 31

Talton, J. O., Lou, Y., Lesser, S., Duke, J., Mech, R., and Koltun, V. (2011). Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):11. 6, 78

Tan, P., Zeng, G., Wang, J., Kang, S. B., and Quan, L. (2007). Image-based tree modeling. *ACM Trans. Graph.*, 26(3). 23

Tecchia, F., Loscos, C., and Chrysanthou, Y. (2002). Image-based crowd rendering. *IEEE Computer Graphics and Applications*, 22:36–43. 62

Terzopoulos, D. and Fleischer, K. W. (1988). Deformable models. *The Visual Computer*, 4(6):306–331. 61

Terzopoulos, D., Platt, J. C., Barr, A. H., and Fleischer, K. W. (1987). Elastically deformable models. In *SIGGRAPH*, pages 205–214. 61

Thalmann, D., O'Sullivan, C., Yersin, B., MaÃ¯m, J., and McDonnell, R. (2007). EG 2007 Course on Populating Virtual Environments with Crowds . pages 23–123. 62

Tsujita, H., Tsukada, K., Kambara, K., and Siio, I. (2010). Complete fashion coordinator: a support system for capturing and selecting daily clothes with social networks. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 127–132, New York, NY, USA. ACM. 61

Tu, Z. and Zhu, S.-C. (2002). Image segmentation by data-driven markov chain monte carlo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:657–673. 78

Ulicny, B., Ciechomski, P. d. H., and Thalmann, D. (2004). Crowdbrush: Interactive authoring of real-time crowd scenes. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 243–252, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. 62

Umetani, N., Igarashi, T., and Mitra, N. J. (2012). Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4). 102

Umetani, N., Kaufman, D. M., Igarashi, T., and Grinspun, E. (2011). Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.*, 30:90:1–90:12. 61

Vanegas, C. A., Garcia-Dorado, I., Aliaga, D. G., Benes, B., and Waddell, P. (2012). Inverse design of urban procedural models. *ACM Trans. Graph.*, 31(6):168:1–168:11. 31

Volino, P., Magnenat-Thalmann, N., and Faure, F. (2009). A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph.*, 28(4). 61

Wang, H., Hecht, F., Ramamoorthi, R., and O'Brien, J. F. (2010). Example-based wrinkle synthesis for clothing animation. *ACM Trans. Graph.*, 29(4). 61

Wang, H., O'Brien, J. F., and Ramamoorthi, R. (2011a). Data-driven elastic models for cloth: modeling and measurement. *ACM Trans. Graph.*, 30(4):71:1–71:12. 23

Wang, H., O'Brien, J. F., and Ramamoorthi, R. (2011b). Data-driven elastic models for cloth: modeling and measurement. *ACM Trans. Graph.*, 30(4):71. 61

Wikipedia (2013). Avatar (2009 film) — Wikipedia, the free encyclopedia. [Online; accessed 5-June-2013]. 4

Woodham, R. (1980). Photometric method for determining surface orientation from multiple images. *Opt. Eng.*, 19(1):139–144. 118, 120, 122, 136

Woodham, R. (1994). Gradient and curvature from the photometric-stereo method, including local confidence estimation. *JOSA-A*, 11(11):3050–3068. 120

Wu, C., Wilburn, B., Matsushita, Y., and Theobalt, C. (2011). High-quality shape from multi-view stereo and shading under general illumination. In *CVPR*, pages 969–976. 138

Wu, L., Ganesh, A., Shi, B., Matsushita, Y., Wang, Y., and Ma, Y. (2010). Robust photometric stereo via low-rank matrix completion and recovery. In *ACCV*, pages 703–717. 120, 122, 124, 125

Wu, T.-P., Sun, J., Tang, C., and Shum, H. (2008). Interactive normal reconstruction from a single image. *ACM Trans. Graph.*, 27(5). 128

Wu, T.-P. and Tang, C.-K. (2010). Photometric stereo via expectation maximization. *IEEE Trans. on PAMI*, 32(3):546–560. 120

Wu, T.-P., Tang, K.-L., Tang, C.-K., and Wong, T.-T. (2006). Dense photometric stereo: A markov random field approach. *IEEE Trans. on PAMI*, 28(11):1830–1846. 120, 122, 126, 127

Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., and Quan, L. (2008). Image-based façade modeling. *ACM Trans. Graph.*, 27(5):1–10. 32

Xiao, J., Hays, J., Ehinger, K., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3485 –3492. 13

Xu, K., Chen, K., Fu, H., Sun, W.-L., and Hu, S.-M. (2013). Sketch2scene: Sketch-based co-retrieval and co-placement of 3D models. *ACM Trans. Graph.*, 32(4). 102

Xu, K., Stewart, J., and Fiume, E. (2002). Constraint-based automatic placement for scene composition. In *Graphics Interface*. 98, 100, 103

Yang, Q., Yang, R., Davis, J., and Nistï¿œer, D. (2007). Spatial-depth super resolution for range images. In *CVPR*. 138, 155

Yeh, Y.-T., Yang, L., Watson, M., Goodman, N. D., and Hanrahan, P. (2012). Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Trans. Graph.*, 31(4). 29, 101

Yeung, S. K., Tang, C.-K., Brown, M. S., and Kang, S. B. (2011a). Matting and compositing of transparent and refractive objects. *ACM Transactions on Graphics*, 30(1):2. 89

Yeung, S.-K., Wu, T.-P., Tang, C.-K., Chan, T. F., and Osher, S. (2011b). Adequate reconstruction of transparent objects on a shoestring budget. In *CVPR*, pages 2513–2520. 118, 120

Yu, L.-F., Yeung, S.-K., Tai, Y.-W., and Lin, S. (2013a). Shading-based shape refinement of RGB-D images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. 12, 121

Yu, L.-F., Yeung, S.-K., Tai, Y.-W., Terzopoulos, D., and Chan, T. F. (2013b). Outdoor photometric stereo. In *Proc. IEEE International Conf. on Computational Photography (ICCP)*, pages 1–8. 11, 136

Yu, L.-F., Yeung, S. K., Tang, C.-K., Terzopoulos, D., Chan, T. F., and Osher, S. (2011). Make it home: Automatic optimization of furniture arrangement. *ACM Transactions on Graphics*, 30(4):86:1–11. Proc. ACM SIGGRAPH 2011 Conf., Vancouver, Canada, August, 2011. 8, 21, 66, 89, 101, 103

Yu, L.-F., Yeung, S.-K., Terzopoulos, D., and Chan, T. F. (2012). Dressup!: Outfit syn-

thesis through automatic optimization. *ACM Transactions on Graphics*, 31(6):134:1–14. Proc. ACM SIGGRAPH Asia 2012 Conf., Singapore, November, 2012. 9

Zhang, R., Tsai, P.-S., Cryer, J. E., and Shah, M. (1999). Shape from shading: A survey. *IEEE PAMI*, 21(8):690–706. 136, 138

Zhu, L., Xu, W., Snyder, J., Liu, Y., Wang, G., and Guo, B. (2012). Motion-guided mechanical toy modeling. *ACM Trans. Graph.*, 31(6):127:1–127:10. 159

Zyla, D. (2010). *The Color of Style*. Dutton Adult, New York, NY. 58, 63