Generating High-quality 3D Assets from Easy-to-access 2D content

Yangtuanfeng Wang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

April 30, 2019

I, Yangtuanfeng Wang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

In the context of content creation, there is an increasing demand for high-quality digital models including object shape, texture, environment illumination, physical properties, etc. As the design and preview presentations get exclusively digital, the need for high-quality 3D assets has grown sharply. The demand, however, is challenging to meet as the process of creating such digital 3D assets remains mostly manual – heavy post-processing is still needed to clean-up captures from commercial 3D capturing devices or models have to be manually created from scratch. On the other hand, low-quality 3D data is much easier to obtain, e.g., modeled by hand, captured with a low-end device, or generated using a virtual simulator. In this thesis, we develop algorithms that consume such low-quality 3D data and 2D cues to automatically create enriched 3D content of higher-quality. Specifically, with the help of low quality underlying 3D geometry, we explore (i) how to grab 3D shape from 2D images while factorizing camera motion and object motion in a dynamic scene; (ii) how to transfer texture and illumination from a captured 2D image to 3D shapes of the same category; (iii) how to decompose 360 environment map and BRDF material from photos and reduce ambiguity by joint observation; and (iv) how to model 3D garment shape and its physical properties from a 2d sketch or image.

Impact Statement

This thesis discusses several topics related to information transfer from 2D image to 3D shape and design parameters. The proposed solution of 3D reconstruction of a dynamic scene, texture transfer from 2D image to the related 3D surface, illumination/material recovery from photo collections and garment modeling from 2D sketch extend the content of the current research community and may inspire future research programmes.

In terms of methodology, this work develops an automated pipeline that lifts information of 2D photograph into a 3D shape surface. This method may boost a 3D shape dataset with realistic texture/illumination so that machine learning algorithms may acquire better performance on several tasks when dealing with real-world data. This work also develops a workflow using a trained neural network inside an objective function in an optimization problem. As the neural network itself is differentiable, the gradient of the objective function can be calculated efficiently. Therefore, the optimization converges fastly.

The pipeline of texture/illumination transfer from an image to a 3D surface can be used in many AI companies to improve the accuracy of their service in the real world scenario. The outcome of our illumination/material recovery project can be a powerful tool for indoor design or online property view. The garment modeling system is jointly developed with Adobe Inc. The method bridges several parts of the apparel industry and may be helpful to the next generation of garment manufactory pipeline.

Acknowledgements

I want to thank my supervisor, Niloy J. Mitra, for his kindness, wisdom, and most of all, his patience. I am grateful for his generosity on his time, and I appreciate how much he has taught me about how to set up expectations and achieve to the best of my ability. I could not show more gratitude towards how he has also shared his many perspectives on many things which makes my Ph.D. so much more than just doing research.

My gratitude to the colleagues in the Smart Geometry Processing Group at University College London: Martin Kilian, Bongjin Koo, Aron Monszpart, Moos Hueting, James Hennessey, Paul Guerrero, Dan Koschier, Carlo Innamorati, Robin Roussel, Yushiang Wong, Chi-Han Peng, Nicolas Mellado, and Tom Kelly. It was a privilege to be able to work with these bright and hard-working people. Thank you for all the seminars, discussions, jokes, and having lunch together.

I would also like to thank my collaborators: Duygu Ceylan, Jovan Popovic, Tobias Ritschel, Hao Su, Qixing Huang, Leo Guibas, Pushmeet Kohli, Dongming Yan, Yongjin Liu, Ying He, Xin Tong, Weiming Wang, and Zhouwang Yang. Without their contribution and advice, this thesis could not have been completed. And special thanks to Ligang Liu for getting me prepared for this wonderful journey.

Lastly, I am very grateful to my family for helping me get through the difficult times.

一番番春秋冬夏，
一场场酸甜苦辣；
敢问路在何方?
路在脚下。
—《西游记》

Year after year,
with sorrow and happiness;
Where is the road ahead?
It's under our feet!
—*Journey to the West*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid development of computer graphics and computer vision has been fueled by the easy availability of rich 3D geometric data which, in turn, inspired researchers to process, analyze and understand collections of 3D data during the past decades. In the recent years, repositories of 3D data also stimulated the evolution of data-driven approaches heavily relying on the availability of large and high-quality dataset. Furthermore, the study on the acquisition of 3D content itself has produced newer and richer 3D content (e.g., texture, materials, physical parameters) that in turn has resulted in new data-driven applications.

In the context of computer graphics, a realistic virtual 3D scenario is the combination of geometry, illumination, texture, and material (as represented by BRDF). To faithfully capture or model a 3D scenario, the extraction of such 3D content plays a very crucial role. However, directly acquiring high-quality 3D content remains difficult. Even for a high-end 3D acquisition device, the output still needs heavy post-processing and clean-up before the data can be used. In this thesis, we explore the problem of generating high-quality 3D data from easy-to-access 2D content, i.e., photograph, sketch, etc. In addition to utilizing 2D content, we also develop algorithms to marry knowledge learned from existing low-quality shape collections and realistic physically-based virtual simulations.

We start by developing an algorithm to obtain 3D information of a real-world scene by lifting it from multiple captured 2D images. The problem of estimating motion of camera and geometry structure of a captured scene, which is known as

the Structure-of-Motion problem, has been well studied over decades. However, previous work relies on the assumption that the target scene is static during imaging. In the first part of this thesis, we propose a method that extracts the 3D structure of each rigid part and the relative motion between camera and objects from a pair of images captured from a dynamic scene. Starting from a pre-boosting step, we establish rich correspondences between the image pair. The correspondences are then grouped and 3D structure is extracted by a continuous optimization framework. Finally, a post-boosting step is proposed to generate high-quality dense 3D point cloud for each rigid part in the scene. To the best of our knowledge, this work is the first to investigate the reconstruction problem of a dynamic scene with rigid segments from only a pair of images.

Next, we observe that texture attached on a 3D shape is a very important feature in many applications, especially in the context of scene understanding. Although 3D shape with high-quality geometric details continues to get richer and richer, it is still hard to efficiently create 3D content with acceptable texture appearance. A general observation on the relationship between geometry and texture indicates that they are not one-to-one corresponded. In the second part of this thesis, we discuss the problem that aims to semantically color a 3D shape by utilizing information from a given captured image of a similar textured object. The problem is challenging as an object's texture as seen in a photograph is distorted by many factors, including pose, geometry, and illumination. These geometric and photometric distortions must first be factorized before the pure underlying texture to a new object can be transferred to produce a texture-consistent 3D model. Instead of relying on hard-to-obtain dense correspondences, we factorize the problem into the reconstruction of a set of base textures (materials) and an illumination model for the object in the image. By exploiting the estimated geometry of a similar 3D model, we reconstruct reliable texture regions and correct for the illumination, from which a full texture map can be recovered and applied to the model.

Next, we recall that the appearance of an object is the result of the complex reaction of both illumination, material, and texture. From a single observation,

such as an image captured under uncontrolled settings, faithfully factorizing the appearance is an ill-defined problem. However, among the real-world objects, a large portion of them are non-textured and similar proxy geometry can be retrieved from a 3D shape database. Furthermore, although every single observation is captured in an uncontrolled way, the same object could have been captured multiple times, or multiple objects might be captured in one scenario provide a weak form of regularization. Such a setting can be exploited to estimate the illumination and material with joint observations of non-textured objects. In the third part of the thesis, we propose to make use of a set of photographs to jointly estimate the non-diffuse materials and environment lighting in an uncontrolled setting. Our key observation is that seeing multiple instances of the same material under different illumination (i.e., environment), and different materials under the same illumination provide valuable constraints that can be exploited to yield a high-quality solution (i.e., specular materials and environment illumination) for all the observed materials and environments. Similar constraints also arise when observing multiple materials in a single environment, or a single material across multiple environments. Technically, we enable this by a novel scalable formulation using parametric mixture models that allows for simultaneous estimation of all materials and illumination directly from a set of (uncontrolled) Internet images. The core of this approach is an optimization procedure that uses two neural networks that are trained on synthetic images to predict good gradients in parametric space given observation of reflected light.

3D shape collections are getting ubiquitous nowadays. The above assumptions, however, is not satisfied for categories of non-rigid objects, such as the shape of clothes draped over the human body. When estimating 3D content associated with such categories of objects, 3D shape database will be insufficient to directly provide a reasonable geometry proxy with enough details. As designing real and virtual garments is extremely demanding with rapidly changing fashion trends and increasing need for synthesizing realistically dressed digital humans for various applications, we need simple and effective workflows to facilitate authoring sewing patterns customized to garment and target body shapes to achieve desired looks.

Traditional workflow involves a trial-and-error procedure wherein a mannequin is draped to judge the resultant folds and the sewing pattern iteratively adjusted until the desired look is achieved. This requires time and experience. Instead, we present a data-driven approach wherein the user directly indicates desired fold patterns simply by sketching while our system estimates corresponding garment and body shape parameters at interactive rates. The recovered parameters can then be further edited and the updated draped garment previewed. Technically, we achieve this via a novel shared shape space that allows the user to seamlessly specify desired characteristics across multimodal input *without* requiring to run garment simulation at design time.

We present and evaluate the algorithms developed for producing various editable 3D assets from low-quality 2D assets.

In Chapter 2, we review the existing literature related to this thesis.

In Chapter 3, we present an approach that reconstructs both the 3D structure of each rigid part as well as their motions between two images captured from a dynamic scene. This consist of material that has been published as [3].

In Chapter 4, we present an automated pipeline capable of transporting texture information from images of real objects to 3D models of similar objects. This consist of material that has been published as [4].

In Chapter 5, we propose a method that makes use of a set of photographs in order to jointly estimate the non-diffuse materials and sharp lighting in an uncontrolled setting. This consist of material that has been published as [5].

In Chapter 6, we present a share latent space learning technique that bridge the 2D sketch space, parameter space, and 3D geometry space, together with an interactive fashion design workflow. This consist of material that has been published as [6].

In Chapter 7 we conclude with discussion about future research directions based on the limitations of our methods.

Chapter 2

Background and Literature Review

Acquisition 3D content from a target scene is a long-studied topic in both computer graphics and computer vision. In this chapter, the most relevant research into structure-motion analysis, shape analysis, image decomposition, texture synthesis, material-illumination modeling, photo-realistic image editing and garment capturing and modeling are discussed. The research related to structure reconstruction and motion understanding in Chapter 3 is discussed in Section 2.1, the research related to image and shape analysis and texture manipulation in Chapter 4 is discussed in Section 2.2, the research related to material and illumination estimation in Chapter 5 is discussed in Section 2.3 and the research related to garment modeling in Chapter 6 is discussed in Section 2.4.

2.1 Structure Reconstruction and Motion Understanding

Analyzing acquired scenes remains a central topic in computer graphics and vision. The goal involves establishing correspondences, performing motion segmentation, and detecting changes in the scenes. The complexity of the problem varies greatly based on how much the objects move, how often recordings are made (i.e., isolated images versus video sequence), and how reliably correspondence can be extracted. On one hand, a static scene can be reliably reconstructed from a set of unorganized images using SfM; while, on the other hand, advanced methods exist to perform motion segmentation from video sequences by tracking correspondence. In Chapter 3,

we study the problem in the context of dynamic environments recorded with only single pair of images.

Motion factorization. Majority of the methods assume spatio-temporal coherence and access to video sequences as input. Based on the factorization of a trajectory matrix into motion and structure components, motion factorization, originally proposed by Tomasi et al. [7], remains the method of choice for multi-body motion segmentation [8, 9]. However, in real world settings with noisy inputs the method can produce only partial (tracked) trajectories. Gruber et al. [10, 11] use *Maximum Likelihood Estimation* to extend factorization to handle uncertainty and missing data. More recently, advanced methods investigate the problem as selection from a family of models while balancing between model complexity and modeling accuracy. This results in a unified formulation [12, 13] that works robustly on a variety of real-world video sequences. We consider [14] representing the state-of-the-art in motion segmentation. Their system uses a cluster of 480 carefully synchronized cameras to continuously capture changing scenes allowing tracking thousands of points and handling non-rigid objects. The above methods, however, rely on access to densely sampling video sequences, and hence are not applicable in our setting. As shown in [14] (Figure 5.), the tracking accuracy drops significantly when less number of views are used.

Motion grouping. The grouping problem has also been investigated as an instance of subspace clustering, and algebraic methods such as GPCA are extended to deal with missing data [15] and outliers [16, 17]. However, without the hypothesis of spatio-temporal coherence (i.e., access to video sequences), the methods quickly become impractical due to the exponential complexity with respect to both the dimension of the ambient space and the number of moving objects in the scene. More recently, analysis has been restricted to sparse, low-dimensional subspace representations to encode trajectory data. [18] achieve impressive performance in terms of accuracy on the Hopkins155 motion segmentation database [19]. With the assumption of sparse observations, algorithms based on geometric model fitting provide better potential. As claimed by [20], PEARL [20, 21] is largely considered

as the state-of-the-art method for few-view motion segmentation. By injecting the idea of model refitting, PEARL achieves higher accuracy on problems with a small number of input frames. However, in real world scenes, PEARL's α-expansion step is adversely affected by outliers especially under large view changes. Please refer to Section 3.5.3 for comparison with our method.

Sceneflow estimation. Scene flow is the dense or semi-dense 3D motion field of a dynamic scene that moves completely or partially with respect to a camera. There are numerous potential applications such as autonomous navigation [22], visual odometry [23], etc. Such techniques perform robust motion estimation of the surrounding objects with a continuous input stream which provides sufficient local information for corresponding search and field estimation, and hence are not applicable in our sparse observation setting.

Prior knowledge. Relying on additional information, for example to recover the 3D structure of a scene while performing motion segmentation has been shown to improve performance. For example, in the case of articulated bodies, Fayad et al. [24] optimize a single cost function to jointly solve the problem of segmentation and 3D reconstruction using an input set of point tracks. The approach has been extended to handle non-rigid objects [25]. These methods, however, require multiple frames from a video to obtain a good initialization and are not applicable in our setting.

Correspondence. The central challenge to the problem is to establish point correspondences between image pairs with large changes of viewpoint introducing severe distortions to object texture. The most common approach is to use the SIFT feature extractor [26]. However, the detected correspondences are very unreliable under large scene changes. [27] extend the concept of SIFT to create an affine invariant descriptor (see Figure 3.12 for comparison).

However, none of the above methods are designed for *only* using a single pair of input images. In Chapter 3, we directly work on a pair of images with large camera motions and object displacements, which makes correspondence detection on the image level difficult. We show that by *simultaneously* optimizing both the object

structure (i.e., 3D point locations) and object motion, we can detect scene changes and establish good point-level correspondences.

2.2 Image and Shape Analysis, Texture Manipulation

In Chapter 4 we study how to efficiently transfer texture information from 2D images to 3D shapes. This evolves the correct understanding of texture that is undistorted from shading, normal direction from underlying geometry and view from the camera.

Image decomposition. Extracting shape, illumination, and reflectance from shading is a long standing problem in computer graphics and vision. We refer to [28] for a state-of-the-art result on this topic. However, when applying these methods to object images, the results are far below the quality needed in graphics applications, due to the complexity in how different factors that affect the image formation process are interweaved. In Chapter 4, we position single point light sources over a sphere around retrieved geometry and render shading samples from an estimated view. These rendered shading samples form the basis of image shading layer. We subsequently apply an optimization based interpolation to recover the coefficient of each shading sample, which successfully decompose the input images without requiring strong assumption on pre-defined priors.

Joint image-shape analysis. Our work is also motivated by a recent line of efforts on joint analysis of image and shape collections, which aim at aggregating and propagating the complementary information contained in images and shapes. Most of these prior efforts have focused on transferring shape attributes to images. Representative works in this domain include shape-driven object detection [29], shape-driven object pose estimation [30–32], symmetry detection [33], depth reconstruction [34, 35], data-driven image-based modeling [36], and image-driven shape segmentation [37]. In contrast, transferring image attributes to shapes has received far less attention. In particular, existing works only focus on transferring global-scale attributes such as part segmentations [37] and material [38]. In Chapter 4, we focus on transferring texture attributes, which exhibit rich information at both fine and coarse scales.

Novel view prediction. Another relevant research topic is predicting the appearance of an object from novel views. In [39], Su et al. propose a probabilistic framework for inferring feature representations of unseen views. Such a framework, however, is unsuitable for the synthesis of fine and subtle texture elements. Recently, there have been efforts to predict novel views using convolutional neural networks [40, 41]. However, due to the characteristics of convolutional filters, these methods tend to smooth out local textural details. In contrast, we apply synthesis to generate texture details.

Texture synthesis. In contrast to texture mapping, texture synthesis focuses on propagating texture elements across the surface to generate realistic visual appearance. This domain has been studied extensively, and we refer to [42] for a survey. Instead of designing new texture synthesis techniques, in this paper we focus on how to adopt existing texture synthesis techniques, i.e., by computing the texture element to be synthesized on each part.

2.3 Material and Illumination Estimation

In Chapter 5, our goal is to perform an advanced *intrinsic image* decomposition (as a factorization into materials and illumination) using an *image collection* with application in *photo-realistic image manipulation*.

Materials and illumination estimation from images. The classic intrinsic image decomposition problem [43] is highly ambiguous as many shapes, illuminations, and reflectances can explain one observation made. When geometry and material for the objects in an image are known, finding the illumination is a problem linear in a set of basis images [44]. Reflectance maps [45] can also be used to map surface orientation to appearance, allowing for a limited range of applications, such as novel views [46]. In absence of such information, alternatives regularize the problem using statistics of each component such as texture [47], or exploit user annotations on Internet images [48] to develop a CRF-based decomposition approach, The latter method is widely considered to be the state-of-the-art for uncontrolled observations.

Haberet al. [49] used observations of a single known geometry observed in a

small set of images to estimate a linear combination of basis BRDFs and pixel-basis lighting. Aittalaet al. [50] capture texture-like materials by fitting SVBRDF using texture statistics to regularize a non-linear optimization on single image capture. An alternate recent trend is to use machine learning to solve inverse rendering problems. Deep learning of convolutional neural networks CNNs (cf., [51]) has been used to decompose Lambertian shading [52, 53], albedo in combination with other factors [54], intrinsics from rendered data [55], decompose images into rendering layers [56], or multiple materials under the same illumination [57].

A complementary but related problem is shape-from-shading, where again many shapes can explain a given image [45]. Alternately, illumination estimation has made use of shadows on diffuse receivers with known geometry [58–60]. Another option is to assume access to outdoor illumination following a parametric sky model [61]. Approaches to jointly solve for shape, reflectance and illumination in a single image or multiple materials under the same illumination using optimization with priors were suggested [62, 63]. Our method, being applicable to direct observation of specular materials under uncontrolled illumination, is more general and works on a large variety of materials in many illuminations.

Image-based rendering (see monograph [64]) can be used to re-create view-dependent appearance when a sufficiently dense set of images is provided. It then can produce novel views but fails to transfer to novel shapes or novel illuminations.

Image and shape collections. Visual computing has made increasing use of data, particularly image and/or shape collections with the aim to exploit cross observations. Starting from illumination [65] and its statistics [66], measurements of BRDFs [67], we have seen models of shape [68], appearance [69], object pose estimate [70], object texture [71], object attributes [30] made possible by discovering correlation across observations in image and/or 3D model collections. In the context of shape analysis, mutual constraints of instances found across images or 3D scenes in the collection have been used to propose room layouts [72], material assignments [73], or scene color and texture assignments [74]. Instead, we directly estimate materials and illumination, rather than solving an assignment problem.

Photo-realistic image manipulation. While the rendering equation [75] explains image formation to the largest part with advanced signal processing perspective for the forward theory of light transport [76], practical solutions are still lacking for the inverse problem, *i.e.,* estimating object materials and illumination directly from (uncontrolled) photographs. Instead, specialized user interfaces can assist this process [77], or multiple images of the same (static) scene can be used to improve relighting and material estimation [49]. Several manipulations of images are possible, even without knowing the decomposition into shape, illumination and material due to perceptual effects [78].

In terms of state-of-the-art image manipulations, 3-Sweep [79] propose a generalized cylinder-based interactive tool to efficiently generate part-level models of man-made objects along with inter-part relations, and use them to enable a diverse variety of non-trivial object-level interactions; Kholgadeet al. [80] align stock 3D models with input images to estimate illumination and appearance, and use them for impressive object-level image manipulations; while Karschet al. [81] estimates a comprehensive 3D scene model from a single, low dynamic range photograph and uses the information to insert virtual objects into the scene.

2.4 Garment Modeling

In Chapter 6, we focus on the problem of garment modeling from either 2D fashion drawing or parameter configuration.

Garment modeling. Traditional garment design is a complex process. Starting with initial 2D sketches of the desired garment, the designer creates the necessary flat sewing patterns which are then stitched together into a garment. Physical draping or physics-based simulation is used to infer the final shape of the garment. While there exist many professional tools (e.g., Optitex [82], Marvelous Designer [83]) to assist designers with the process, there has been a significant effort in the research community to propose alternative computational tools.

Many of the previous works have specifically focused on bridging the gap across two different design spaces. For example, [84] focus on automatically parsing 2D

garment patterns to 3D, while other methods have focused on modeling 3D garments via input sketches. Several sketch-based interfaces [85–87] have been proposed where silhouette and fold edges in an input sketch are analyzed to create a 3D garment. However, they assume the input sketch is provided with respect to a given 3D mannequin and use the body shape of the mannequin to lift the sketch to 3D. The recent approach of [88] provides a more general tool for modeling 3D developable surfaces with designed folds but require multi-view sketch input (e.g., frontal, side, and optionally top). The freeform surface modeling tool, BendSketch [89], is capable of modeling plausible 3D garments from user sketches but has no notion of the corresponding 2D sewing patterns. Recently, [90] present a modeling system where the user can sketch different types of strokes on an existing 3D draped garment to denote different types of desired folds. The system then extends and optimizes the 2D garment patterns such that the draped garment exhibits the desired folds. While this system provides impressive results, it assumes an initial 3D draped garment to be provided. In contrast, our system automatically infers the parameters of the garment and the body from an input sketch and maps them to the final 3D draped shape of the garment. We achieve this goal by proposing a deep learning based solution which draws inspiration from recent methods that use deep learning to infer 3D shapes from 2D sketches [91] or to interactively control shape variability [92]. Once the garment parameters are predicted by our method from a single sketch, the method of [90] or the rule-free method of [93] can be used to refine cuts on the resulting 2D garment patterns.

In another line of work, researchers have proposed to use other types of input such as images or RGBD scans to model garments. Given an input image, [94] first estimate the 3D body shape in a semi-automatic fashion and then lift extracted silhouette edges of the garments to 3D. [95] use an image of a garment on top of a mannequin as input and detect silhouette edges and landmark points to create a 3D garment. [96] model garments from RGBD data as a combination of 3D template components based on a set of rules. Given a database of garment templates, [2] propose a relatively complex pipeline to determine the parameters of these garments

to match an input image. More recently, [97] present a deep learning based method which predicts the deformation of a garment from a reference 3D garment given an input image. While the proposed approaches provide plausible garments, they often fail to capture the details, i.e., the exact fold patterns observed in the input. In contrast, our goal is to be able to reconstruct such folds to enable realistic design and editing of garments.

Garment editing and retargeting. In addition to modeling garments from scratch, several methods have been proposed for editing the shape and appearance of them. [98] propose an interactive editing system that enables bi-directional editing between 2D garment patterns and 3D draped forms. [99] explore apparel resizing with flexible shape control, [100] investigate stylizing tight-fitting garments. [101] present a method to map 3D edits to a garment to plausible 2D garment patterns. In contrast, we support a multi-modal design and editing paradigm, specifically focusing on modeling the desired folds and silhouettes of a garment.

Retargeting the style of an existing garment to body shapes with different proportions is a specific form of editing that has attracted special attention. [102] formulate a constrained optimization framework to transfer garments across different 3D characters. Other approaches use data-driven methods [103, 104] to replace the expensive physical cloth simulation process and present retargeting examples. In our work, we perform garment retargeting via a novel optimization procedure that directly operates at the joint embedding of different garment design spaces and can be used to transfer across 100s of shape variations while ensuring that desired fold characteristics are preserved.

Garment capture. While garment modeling approaches aim to generate realistic garments, capture methods focus on faithfully reconstructing the garment observed in the input data. They accomplish this task often by utilizing more complex capture setups. [105] present one of the earlier approaches where the geometry of a garment is reconstructed from a stereo image pair. Several follow up works have instead used a multi-view capture setup to reconstruct garments with color-coded patterns [106, 107]. [108] have eliminated the need for a color-coded pattern and presented a multi-

view markerless motion capture system for garments. Their follow-up work [109] aims to add details due to wrinkles and folds to the coarse meshes captured from the multi-view video input. [110] solve a similar problem of augmenting coarse 3D garment geometry with wrinkles using a data-driven approach. In a separate line of work, given a 2D pattern and 3D contour curve, [111] interpolates the curve in a procedural manner to generate folded paper geometry. The more recent approach of [112] analyzes a set of 3D scans to learn a deformation model for human bodies and garments where garments are represented as the residual with respect to the body shape. Finally, the ClothCap [113] system takes a 3D scan sequence as input and captures both the body and the garment shape assuming weak priors about where a specific type of garment is expected to be with respect to the body. All these methods perform a faithful reconstruction of the garments including the fold and wrinkle geometry but rely on multi-view input or alternate 3D information. In contrast, our method performs a similar prediction using a single sketched image as input and allows for subsequent multi-modal refining in the garment domain and/or the mannequin body shape.

Chapter 3

Dynamic SfM: Detecting Scene Changes from Image Pairs

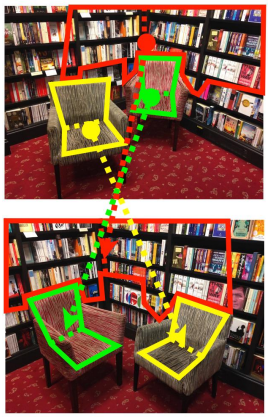
Detecting changes in scenes is important in many scene understanding tasks. In this chapter, we pursue this goal simply from a pair of image recordings. Specifically, our goal is to infer what the objects are, how they are structured, and how they moved between the images. The problem is challenging as large changes make point-level correspondence establishment difficult, which in turn breaks the assumptions of standard Structure-from-Motion (SfM). We propose a novel algorithm for dynamic SfM wherein we first generate a pool of potential corresponding points by hypothesizing over possible movements, and then use a continuous optimization formulation to obtain a low complexity solution that best explains the scene recordings, i.e., the input image pairs. We test the algorithm on a variety of examples to recover the multiple object structures and their changes.

The work presented in this chapter was developed and written in a collaboration with multiple parties, and published as [3].

3.1 Introduction

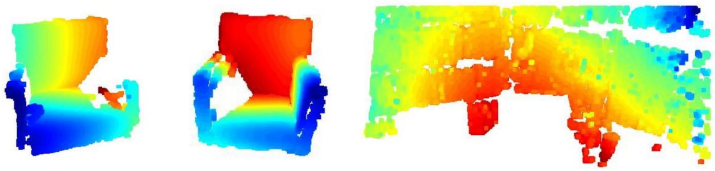
We live in a dynamic world where objects regularly move or are moved around. Understanding such a world naturally amounts to detecting what changes and what does not. This constitutes a fundamental goal in scene analysis and understanding. In the context of time-coherent acquisition, e.g., using a video feed, advanced methods

exist to reliably *track* objects to detect changes. However, limited options exist for uncontrolled settings with sparse measurements.



(a) From an input image pair, we detect the scene changes.

(b) Objects rendered in color.



(c) Objects rendered by depth. For visualization, we color code the estimated depth with values increasing from blue to red.

Figure 3.1: Using the input image pair (a), our algorithm automatically performs change detection between the two images and calculates the motion of each rigidly moving part (b) while simultaneously estimating their 3D structure to enhance performance (c). Note that due to two-view ambiguity, we have no information about the absolute depth of each object. Instead, we show the depth maps for each object separately.

In this paper, we investigate the problem of detecting scene changes from only a pair of input images. By *change*, we focus on what objects moved (i.e., segmentation), how the objects are structured (i.e., their 3D shape), and how the objects moved (i.e., motion parameters). To make our framework compatible with uncontrolled capturing setup, we allow the images to be captured by different devices from different point of views. To compensate this ego-motion, our solution is to fix the camera position and treat the background of the scene to be a moving object. We assume throughout the paper, that all objects as well as the background are moving rigidly during the *change*. When the scene is static, a seemingly natural option is to use structure from motion (SfM) to reconstruct the 3D scene from the input images. However, such an approach fails in a dynamic scenario as it simply ignores moving objects which lead to the point-to-point correspondence search fails. This is particularly so in situations as in our setting, where the input images are assumed to capture large scene changes. For example, in Figure 3.2, only (part of)

the background is recovered and the changes are completely missed. Due to the large scene changes and sparse observations, representing the 3D scene by non-rigid deformation is intractable, therefore non-rigid extension for Structure from Motion [114] is not a feasible solution here as well. Other alternative methods such as motion segmentation also falls short for different reasons (see Section 2.1).

We propose a solution based on two main steps: First, starting from a superset of candidate correspondences (i.e., including false positive matches) between the input image pairs, solving the above problems amounts to correctly *grouping* the correspondences based on the (unknown) motion models. To this end, we propose a continuous grouping formulation to simultaneously solve for segmentation, object structure, and object motion. Second, it is possible to generate a superset of candidate correspondences by pre-warping one of the input images to simulate the effect of possible homographies relating (near) planar surfaces in the two images. We realize this pre-boosting step to capture the correspondence pairs that are easily missed by direct analysis of the input image pairs. Herein we specifically make use of the structure of the scene to solve the dynamic SfM problem. Finally, in the dense reconstruction step, we improve the coarse correspondence obtained at the end of the grouping optimization to create the final output. Figure 3.1 shows a typical output of our method.

Figure 3.2: Result of directly running Structure-from-motion on images of a dynamic scene (using [115]). Note, how the algorithm misses moving objects due to its static scene assumption.

We evaluate the proposed algorithm on a range of test inputs of varying complexity. We also perform quantitative analysis on simulated test scenes with access to groundtruth and evaluate the effects of different parameter settings.

3.2 Overview

Our goal is to detect changes in indoor scenes from a pair of image recordings. This requires answering the following: (i) what are the moving parts, i.e., obtain point clouds for the moving parts of the scene; (ii) how did they move, i.e., estimate the movement for the respective objects between frames; and (iii) what are the camera parameters for the two (uncalibrated) input images.

There are two main problems: (i) direct structure-from-motion (SfM) computation on the input image pairs fails as the scene is not static (see Figure 3.2); and (ii) obtaining good quality point correspondences is challenging in presence of large scene changes as in our setting.

To address the first problem, we observe that if sufficient number of good correspondence pairs are available, then the problem reduces to grouping the correspondences according to the (unknown) moving parts. Specifically, if we have the correspondence pairs correctly grouped, we can simply perform SfM for each individual group under the additional constraint that each image has a common calibration. Hence, we formulate continuous energy minimization to group the created dense feature point matches into different rigid motion trajectories, estimate the 3D object positions and identify outlier samples (see Section 3.3.1). Note that the continuous formulation allows to take advantage of the additional information contributed by the scene structure.

To address the second problem, we integrate correspondence boosting and camera model hypothesis generation with the multi-hypothesis grouping. Essentially, we increase the set of potential correspondence pairs and only later recover the subset of correct correspondences. First, we initialize our algorithm with a sparse set of high-quality feature correspondences using any feature descriptor (SIFT in our case). Then, in a critical step, we boost the set of available correspondences

by hypothesizing part motions (or equivalently camera motions) as described in Section 3.3.2.

Finally, in Section 3.3.3 we describe how we employ a patch-based correspondence post-boosting strategy using the optimized motion grouping to generate an even denser point cloud, that can be used as input to other applications.

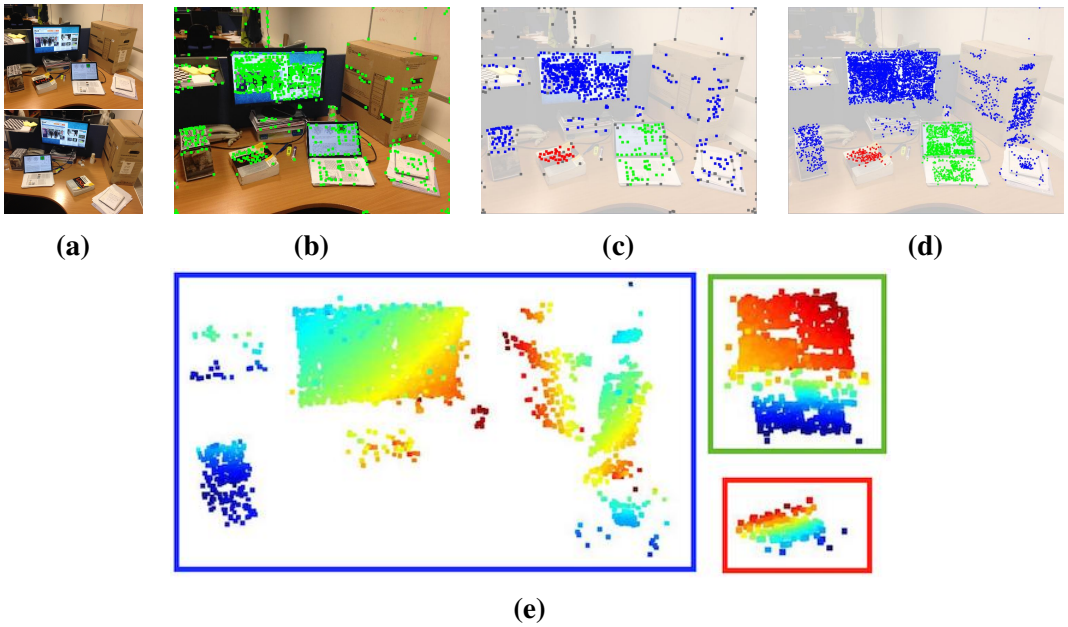


Figure 3.3: Algorithm Pipeline. From two images (a) of a dynamic scene with multiple objects undergoing rigid motions. We first generate a set of candidate correspondence pairs using our pre-boosting strategy indicated by green dots in (b). In this example, 1046 correspondence pairs were generated, instead of only 554 using SIFT directly. Next, we use continuous optimization to simultaneously recover the motion of each rigid part (c) along with their coarse 3D structure (colors show assignment to the different motion groups). Finally, we use the grouping result to obtain a denser set of correspondence pairs (d). Here, 5281 correspondence pairs are generated from 832 inlier correspondence pairs obtained from pre-boosting. We show the structure of each object (e) color coded by estimated depth, distances increasing from blue to red.

3.3 Algorithm

In this section, we first formulate the dynamic SfM as a grouping problem, wherein we categorize candidate correspondence pairs into different motion groups while identifying outlier correspondences (false positive feature matches). We then describe how to initialize the grouping optimization, that is, how to create a sufficiently rich set of correspondence pairs from two images containing significant scene changes.

Finally, we describe how the grouped correspondences can be used to obtain dense structures (i.e., point clouds) for the different moving objects in the scene. Figure 3.3 shows an overview of our method.

3.3.1 Multi-body structure and motion

At the core of our method is an energy-based continuous optimization that recovers both the 3D structure and motion of each rigid part in a dynamic scene. We seek to extract a low complexity explanation of the scene in terms of objects and their motion, that best explains the observations, i.e., the input pair of images. We observe that the problem amounts to robustly grouping a set of candidate correspondences into motion groups. Later, in Section 3.3.2, we describe how to initially extract such a set of candidate correspondences, possibly containing a significant amount of outliers. We pose the grouping problem as minimizing the reprojection error, outlier penalty, group complexity penalty, and non-smoothness by labeling the correspondences into different groups, while simultaneously estimating their 3D positions.

Let there be M^* possible moving objects, and $M > M^*$ motion model candidates captured by the corresponding camera motions L_i with $i \in 1:M$ for each image. The goal is to assign each correspondence pair to one of these (unknown) motions,

Table 3.1: List of symbols

M	Number of motion candidates
$L_i,\ i \in 1:M$	Motion model candidate, holds one set of camera parameters for each image
N	Number of feature correspondences
$\mathbf{d}_k,\ k \in 1:N$	3D position implied by a correspondence k
α_i^k	Element in label vector representing assignment likelihood of $\mathbf{d}_k$ to motion model L_i
$\|\cdot\|_{L_i}$	Operation representing the sum of reprojection errors for motion L_i in both images
δ_k	Likelihood of $\mathbf{d}_k$ being an inlier match
SN_k^j	Neighborhood of corresp. k in image j
$\|\cdot\|$	Set cardinality
β	Sparsity coefficient
$\theta_{p,q}$	Consistency weight for corresp. $p \leftrightarrow q$
ω_1	Complexity penalty weight
ω_2	Outlier penalty weight
ω_3	Consistency penalty weight

or mark it as an outlier. We encode this grouping as an $N \times M$ label matrix, with row vectors α^k for each correspondence. For each inlier correspondence, we also maintain the respective (unknown) 3D position as $\mathbf{d}_k$.

For each correspondence, we use selection variables α_i^k, the i-th $(i \in 1:M)$ component of α^k, to capture the likelihood of a correspondence belonging to motion described by the motion model L_i. Note that $\alpha_i^k \in [0,1]$ and each correspondences can at most be assigned to one model, as captured by

$$\sum_{i=1}^{M} \alpha_i^k \leq 1 \quad \forall k. \tag{3.1}$$

We parametrize the target energy via motion models $\{L_i\}$, 3D points $\{\mathbf{d}_k\}$, and label vectors $\{\alpha^k\}$. Specifically, the energy estimate consists of four terms:

$$\begin{aligned} &E(\{L_i\},\{\mathbf{d}_k\},\{\alpha^k\}) := \\ &\quad E_{data} + E_{complexity} + E_{outlier} + E_{consistency}. \end{aligned} \tag{3.2}$$

Note that the formulation takes full advantage of (unknown) structure in the motion segmentation problem by estimating a 3D position for each correspondence. This is in contrast to representing correspondence only as a pair of 2D feature point positions in two images and subsequently measuring geometric error using for example squared Sampson's distance [116].

The data term E_{data} captures the sum of reprojection errors in the two images, weighted by the assignment likelihood α. Specifically,

$$E_{data}(\{L_i\},\{\mathbf{d}_k\},\{\alpha^k\}) = \sum_{k=1}^{N}\sum_{i=1}^{M} \alpha_i^k \parallel \mathbf{d}_k \parallel_{L_i}, \tag{3.3}$$

where $\| \cdot \|_{L_i}$ in Equation 3.3 is the sum of reprojection errors of 3D point $\mathbf{d}_k$ to the two input images under camera motion L_i. We use a perspective camera model in our implementation and the reprojection error is calculated as sum of squared distances similarly to [116].

The complexity term $E_{complexity}$ penalizes having too many separate groups to

describe the motions. In other words, this term exists in pursuit of the sparsity of label vectors α^k with respect to model k. Specifically,

$$E_{complexity}(\alpha) = \omega_1 \cdot \sum_{i=1}^{M} \left(\sum_{k=1}^{N} \alpha_i^k \right)^{\beta} \tag{3.4}$$

where, β is an exponent close to zero. (Alternately, one can use a reweighted L1 formulation here.) The term weight ω_1 can be considered as a threshold of minimum number of correspondences in a group, since points in any group having less than $\lceil (\omega_1/\omega_2)^{1/\beta} \rceil$ correspondences will be identified as outliers.

For any correspondence $\{\mathbf{d}_k\}$, if it is too costly to fit using every model candidate, we consider it as an outlier by allowing α^k to tend to zero for all i. However, to avoid the trivial assignment of marking all the correspondences as outliers, we introduce the outlier term:

$$E_{outlier}(\alpha) = \omega_2 \cdot \left(\sum_{k=1}^{N} \delta_k (1 - \sum_{i=1}^{M} \alpha_i^k) \right). \tag{3.5}$$

Here, δ_k is a pre-calculated coefficient for each correspondence to indicate how much we want to penalize the k-th correspondence if it is an outlier. A simple case is to set $\delta_k = 1$ for all k. However, as illustrated in Figure 3.4, for the two-view situation, the reprojection error is not fully reliable. Unfortunately, sometimes an outlier might have very low reprojection error w.r.t a particular motion model solely by chance.

In order to address this issue, we use δ_k to account for the possibility of the k-th correspondence being an outlier. A simple criteria is how much the two groups of neighbors of the feature points in two images overlap. Specifically, we set $\delta_k = \max(1 - 0.2 \cdot |SN_k^1 \cap SN_k^2|, 0)$, where $SN_k^{1,2}$ is the set of 10 nearest neighbors of the correspondence k in the two images, measured in image space and $|\cdot|$ here denotes the cardinality of the intersection set. The weight ω_1 can also be considered as an outlier threshold as correspondences with reprojection error larger than ω_1 will finally be labeled as outliers when the optimization converges.

The consistency term $E_{consistency}$ regularizes false positive matches caused by

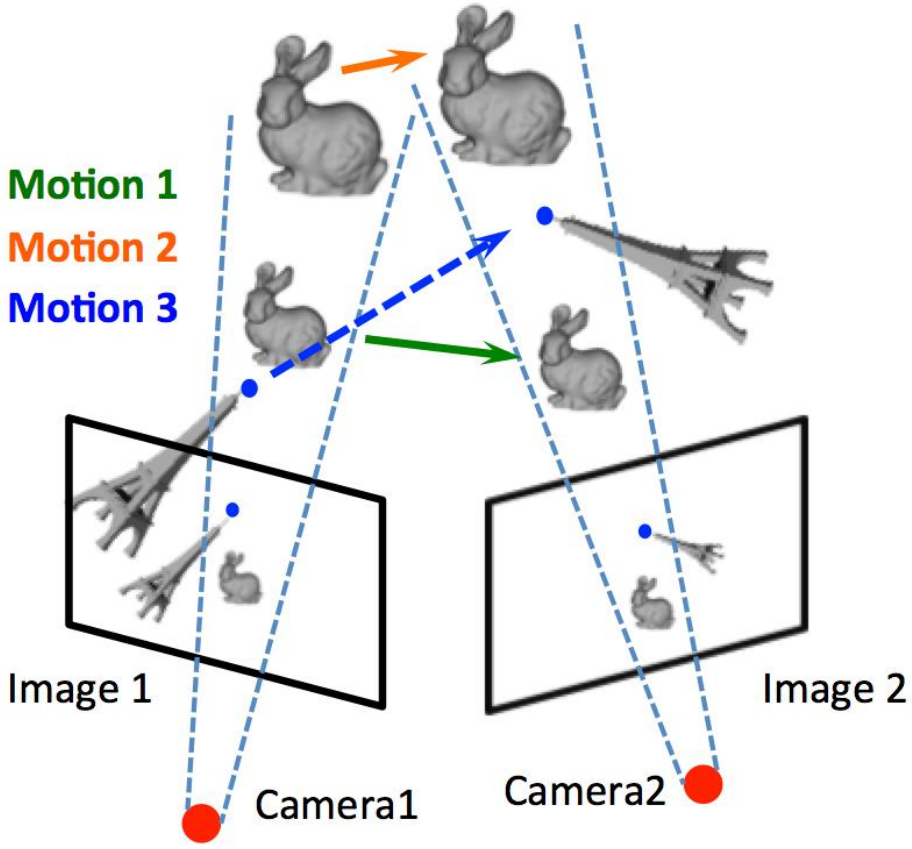


Figure 3.4: There are two types of 2-view ambiguities preventing us from estimating the relative depth of each moving object from only two viewpoints. For example, the bunny can be quite small and have quite large displacement (Motion 1) or can be very large with a relatively small displacement (Motion 2). For some particular cases, a correspondence could fit well into several different motion models, as for example the top of the tower. The correspondences in the images marked by blue dots fit Motion 1 well, although they actually belong to Motion 3.

two-view ambiguity (the tower case in Figure 3.4). Specifically,

$$E_{consistency}(\alpha) = \omega_3 \cdot \sum_{(p,q)\in DN} \theta_{p,q} \parallel \alpha^p - \alpha^q \parallel. \tag{3.6}$$

We use Delaunay triangulation in our implementation to create the neighborhood DN of each data point. The prior weight $\theta_{p,q}$ captures that correspondence p and correspondence q should belong to the same group. We use the inverse of the average squared distance between two feature points p, q on the two images to estimate $\theta_{p,q} = \left((dist_{img1}(p,q)^2 + dist_{img2}(p,q)^2)\right)^{-1}$.

We minimize $E(\{L_i\}, \{\mathbf{d}_k\}, \{\alpha^k\})$ with the constraints that the label vector is valid $\alpha_i^k \in [0,1]$ and Equation 3.1 to solve the dynamic SfM problem.

Generating motion model candidates. In order to initialize the above optimization, we need a good initial set of motion model candidates. Since the camera intrinsic parameters is unknown to us, we choose fundamental matrix instead of essential matrix to represent our motion model. Note that having a good set of correspondences allows direct generation of motion models by using the 8-points algorithm for evaluating the fundamental camera matrix (cf. [20]). Since the original set of input correspondences has both inliers and outliers, we use RANSAC [117] to create a superset of motion candidates, with some of them being correct. This is sufficient for the grouping optimization described above to extract the suitable motion models.

Domain problem. Generally, candidate models can be generated by running RANSAC until each rigid part is covered. Unfortunately, as shown in [20], the theoretical estimate of necessary iterations is always too large to be practical (in their example, 3 objects with 20/24/56 inliers can only achieve 0.92 confidence, even when sampling 10^6 times). In practice, for some lucky situations, a small number of samples may also be sufficient.

However, in real world scenarios RANSAC still falls short due to problems caused by large differences (i) in the size and (ii) feature richness of image regions that move rigidly together. This will lead to differences in the magnitudes of number of correspondences generated from different image regions. A static background or a colorful information poster will generate proportionally more feature points than the rest of the image parts. They will act as a *domain* to smaller regions, hence we refer to this as the *domain problem*. Given such proportional differences, RANSAC faces difficulties finding the smaller, rigidly moving regions in the images.

In this paper, we use a *reweighted RANSAC* strategy to get reliable motion models even from a very small fraction of good candidates. Specifically, we lower the weight of data points that have been considered as inliers by multiplying by a weight decrement factor, in order to boost the selection probability of points from other parts of the scene (see Algorithm 1 for detail). This is particularly effective in

our case, when dealing with multiple moving objects. For example, in Figure 3.3, we obtained three rigid parts with 40, 82 and 710 (domain part) inlier correspondences and 114 outliers. Our optimization converges to the correct solution after that only 5-10 candidates have been generated by our reweighted RANSAC algorithm. The behavior was similar in the other examples presented in this paper.

Algorithm 1 *reweighted* RANSAC

// Initialization
Weight decrement factor $\mu = 0.2$
Weight $w_k = 1, k \in 1:N$
Average sample times $T = 1.5$▷ each point is expected to contribute as inlier on average T times
// Reweighted RANSAC
while $\sum_{k=1}^{N} w_k > \mu^T \cdot N$ **do**
    $nBest = 0$;
    for $i = 1 : nIterations$ **do**
        Randomly sample 8 points $p_{1...8}$
        $tmpF =$ Compute fundamental matrix from $p_{1...8}$
        $tmpIniliers =$ index of inliers under fundmental matrix $tmpF$
        $S = \sum_{tmpIniliers} w_k$
        if $S > nBest$ **then**
            $fBest = tmpF$
            $nBest = S$
            $inliers = tmpIniliers$
        end if
    end for
    $w_{inliers} = \mu \cdot w_{inliers}$
    Output fundmental matrix $fBest$.
end while

Generating initial point locations. After generating a finite set of motion candidates, we estimate the initial 3D positions $\{\mathbf{d}_k^{initial}\}$ by triangulating each correspondence using the camera model that gives the smallest reprojection error. We then initialize the assignments α_i^k proportional to the inverse of the re-projection error of $\{\mathbf{d}_k^{initial}\}$ to all camera models L_i as

$$\alpha_i^k = \| \mathbf{d}_k^{initial} \|_{L_i} / \sum_{p=1}^{M} \| \mathbf{d}_k^{initial} \|_{L_p}.$$

Optimization. We use MATLAB's interior-point solver for the optimization using the following parameter settings in our experiments: $\omega_1 = 1000$, $\omega_2 = 500$ (means outlier threshold adds up to $\sqrt{500}$ pixels), $\omega_3 = 500$, and $\beta = 0.4$. As discussed in Equation 3.2, we optimize for the variables $\{L_i\}, \{\mathbf{d}_k\}, \{\alpha^k\}$. Once the optimization converged, we round α_i^k to 1 if it is greater than 0.9 and to 0 otherwise. This assigns each correspondence to a single camera model as we experimentally found $\alpha_i^k \simeq 0.99$ at convergence, indicating that $\mathbf{d}_k$ was linked to L_i.

3.3.2 Correspondence pre-boosting

The main difficulty in matching feature points between images, where the camera view points are far apart is that the orientation of the surfaces we are interested in vary a lot relative to the cameras. The change of viewpoint results in distortion of texture, and feature descriptors generated from the same image locations will change significantly as a consequence (as shown in Figure 3.6). Simply extracting feature points from two images and matching them based on the distance metric will only

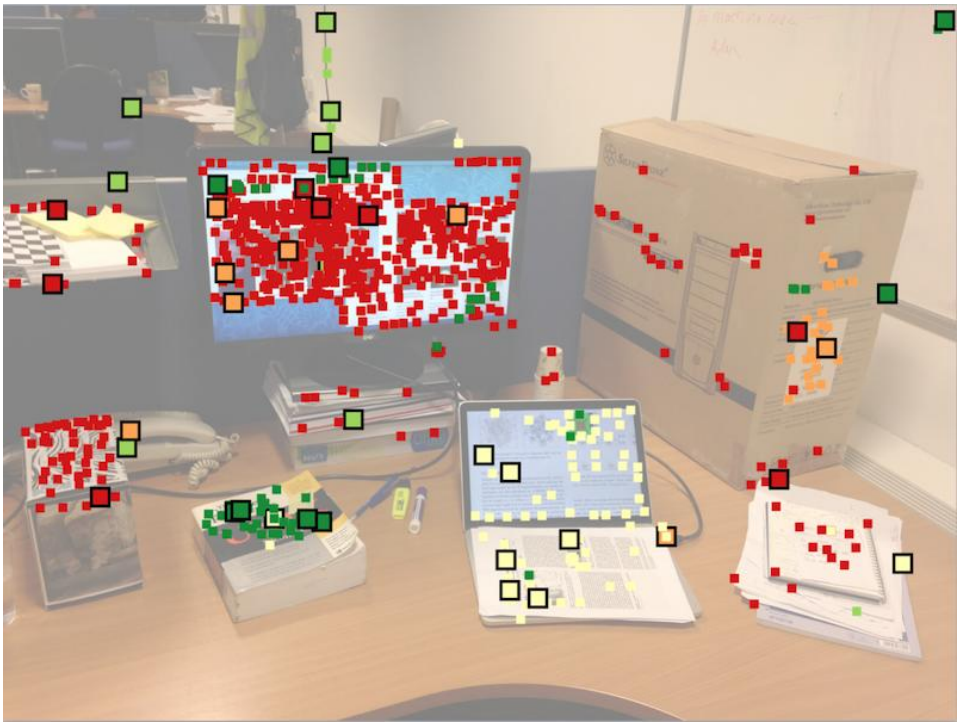


Figure 3.5: This figure shows the first five candidate models generated by *reweighted RANSAC*. Sampled sets of 8 points are shown as larger squares with black borders. Smaller squares with the same colour show inliers under the corresponding generated camera model.

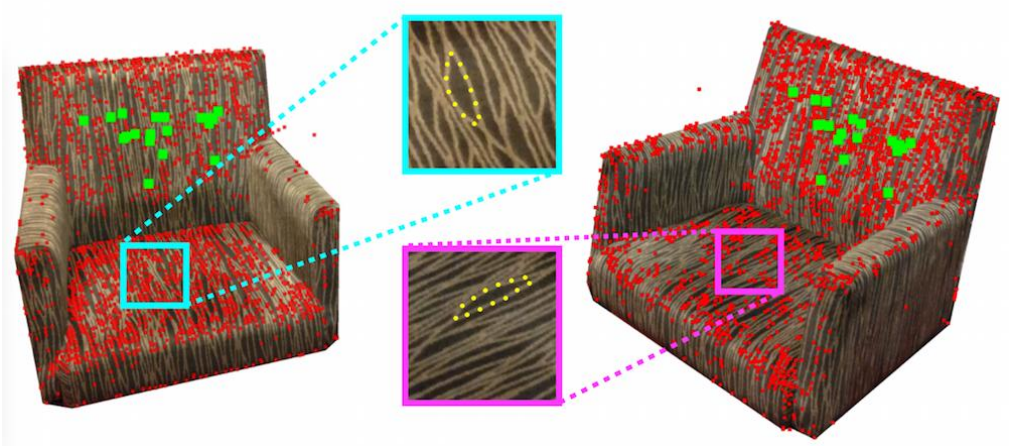


Figure 3.6: We directly extract SIFT feature points and run feature matching [26, 118] on a pair of images of a chair. Red dots are generated feature points and green dots are matched correspondences. The close-up views show the difference in the observed texture of the same patch on a surface perpendicular to the image plane. We conclude, that the failure of feature matching is mainly caused by texture distortion. A set of yellow dots are marked manually to visualise the same pattern. This motivates us to have a pre-boosting step to recover such correspondences from distortion.

reveal correspondences between image parts, where the perspective did not change a lot, in most of the cases surfaces parallel to the image plane.

The first step of our pipeline is to as much as possible boost the number correspondences obtained from areas, where the surface texture underwent significant change in distortion due to the change of camera viewpoint. These areas are initially very sparsely covered by high confidence matches. We keep one image fixed, and 'rotate' the other image in 3D, as shown in Figure 3.7. Note that this step implicitly guesses a potential motion or alternately a homography between corresponding (near) planar parts in the scene. The intuition being that if the guess is correct, then the corresponding moving parts are likely to pick up correct correspondence pairs. We extract feature points in each rotated image and perform correspondence matching w.r.t. the fixed image. Rotating the image in 3D simply allows us to change the image plane normal. We approximately create S rotated image copies by sampling a

half-sphere uniformly with the parametrization described as:

$$\begin{aligned} u_i &= \arcsin\left(1 - \frac{2i-1}{2S}\right) \quad v_i = u_i\sqrt{2\pi S} \\ \mathbf{n}_i &= [\cos(u_i)\cos(v_i);\ \cos(u_i)\sin(v_i);\ \sin(u_i)] \end{aligned} \tag{3.7}$$

where, $i = 1, 2, \ldots, S$. Warping an image by spatial rotations allows us to compensate for the difference in texture caused by the change of viewpoint. It allows us to match more feature points and results in a more complete coverage of the scene as shown in Figure 3.8. Simultaneously, more mismatches are also generated when performing pre-boosting. However, this can be handled well by our grouping optimization using the mismatch-aware outlier penalty. The idea of our pre-boosting method is similar to [27], but ours performs better due to a more uniform sampling of warping rotations. Note that this warping can also provide us with information about the structure of the

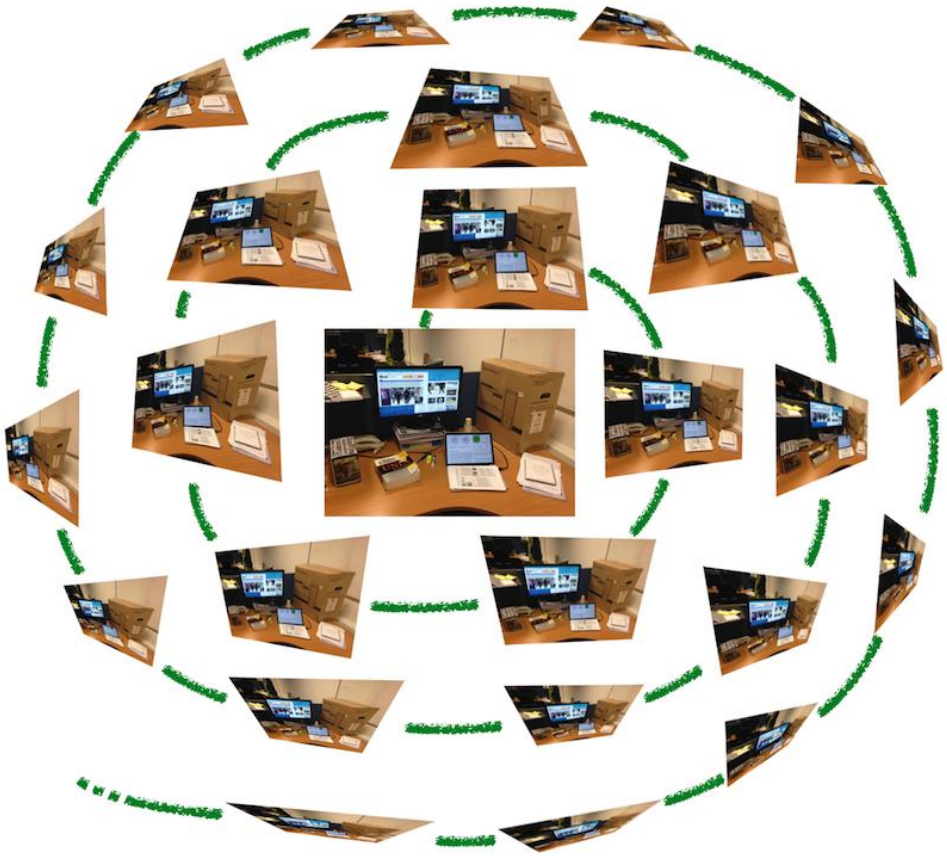


Figure 3.7: Image 1 is warped by rotating the normal of the image plane according to the parametric Equation 3.7.

scene. However, we carefully avoid to explicitly rely on this heuristic when sampling homographies, because our continuous optimization recovers the same knowledge with much higher reliability.

Implementation details. We use VLFeat's implementation of SIFT extraction and matching [118] in our experiments. We set the parameters *"peak threshold"* $= 6$ and *"edge threshold"* $= 10$. As uniqueness threshold during matching, we use a quite strong value of 0.45 to allow us to select correspondences with significant confidence and reliability. Setting $S = 30$ was enough during our experiments. We used a density threshold $d = 30$ *pixel* in order to prevent the generation of too many repetitive correspondences during matching over the warped images. This guarantees, that the minimum distance between any two matched features in the original image exceeds d pixels.

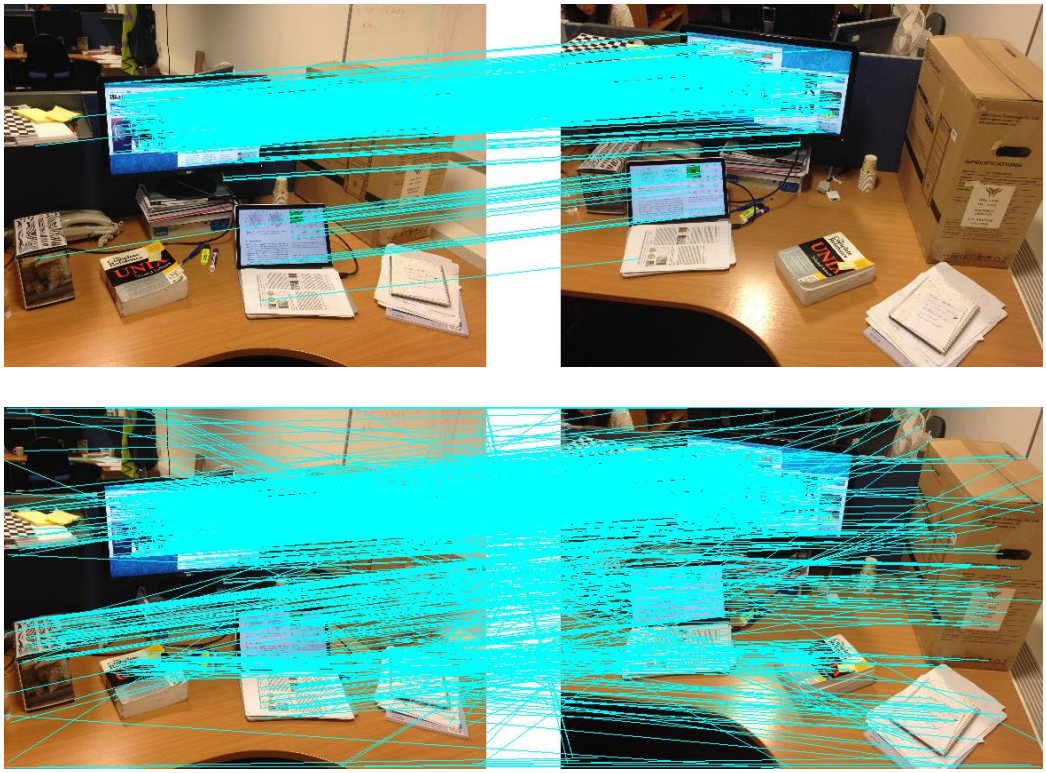


Figure 3.8: Top figure shows the feature point correspondences (cyan lines) matched between the two original images. Bottom figure shows, that after pre-boosting, the number correspondences is significantly increased, especially in areas with upward pointing surface normals. Note that a substantial amount mismatches (outliers) are generated as well, which we robustly handle in our optimization.

3.3.3 Correspondence post-boosting

Finally, we generate a denser 3D point cloud from the structure and motion recovered by the continuous optimization to utilize the recovered information in a more efficient manner. We designed our patch-based method to take advantage of the warping strategy described along with pre-boosting (Section 3.3.2), which gives us a competitive edge compared to general dense reconstruction.

Patch-based boosting. Our goal is to use our optimized, high-quality correspondences to generate a denser output point cloud using a local smoothness prior. Many methods employ the spatial regularity assumption (e.g., [119]) to locally propagate information around recovered feature point matches. We propose a hierarchical feature matching algorithm. In patch-based boosting, we perform a second round of feature point matching on each pair of patches, each connected by an optimized output correspondence. These matches were missed in the initial correspondence search, because they had several possible matches in the global scene, and were labeled as insignificant. We take advantage of our pre-boosting step by looking up between features in all warped versions of the corresponding patch.

Correspondences are stored in a *FIFO*-queue and we propagate them using a *breadth-first* strategy. Specifically, each time a correspondence is picked from the front of the queue, we extract a square patch with 80×80 pixels from one input image. We then find the corresponding patch in each of the warped images, and perform SIFT feature point extraction similarly to the pre-boosting step, using the same parameters. These feature points are then matched locally between the warped patches and the resulting correspondences are appended to the end of the queue. Each newly generated correspondence is assigned to the same motion model and group of correspondences as its parent. Similar to [119] we also applied a match density threshold set to $d = 6$ in our case, to control the blooming of correspondences, see Figure 3.3d.

3.4 Application

3.4.1 Dense reconstruction

The output of our pipeline is a motion model and 3D point cloud for each rigidly moving part of the dynamic scene. This can naturally be achieved using any dense stereo reconstruction method, i.e., CMVS/PMVS [119], CMPMVS [120], MVE [121] and SURE [122]. We apply Furukawa's PMVS algorithm (implemented in [115]), which takes a 3D point cloud of a rigid object and two 2D views of it as input. For each motion group segmented by our algorithm, we use the relative camera positions and densely matched features associated with this motion to initialize PMVS. The advantage of our result (Figure 3.9) over running structure from motion directly on each, manually segmented rigid part is that our pre- and post-boosting steps efficiently increases the covered area in the images, from which 3D structure can be recognized in our two view setting.

Figure 3.9: Classic dense reconstruction (left) and dense reconstruction based on our post-boosting step (middle) using the chair scene from Figure 3.6. Our pre- and post-boosting methods (right, rendered with green dots) enables the reconstruction of larger areas of the input images.

3.4.2 Motion interpolation

We calculate a motion in our method for each rigidly moving part of the dynamic scene, interpretable as the motion of the part from the first image to the second. As an application of this motion information, we can interpolate the dynamic scene between the input two images to generate a possible set of trajectories for the rigidly moving parts. We perform the interpolation as described in [123], see Figure 3.10.

Figure 3.10: Motion interpolation with our dense reconstruction from only a pair of input images (overlaid on first/last columns).

3.4.3 Working with multi-view

Our two-view based pipeline can be naturally extended to multi-view cases. Assume that we have Π images as input and D_k is 3D points for the k-th trajectory. The energy is formulated in the same way as in Equation 3.2, with the natural modification of some of the terms. Specifically, $\|\cdot\|_{L_i}$ in the data term is the sum of reprojection error w.r.t. all Π images, the outlier prior δ_k in the outlier penalty term will be based on all image pairs as well as neighborhood DN and inlier likelihood $\theta_{p,q}$ in the consistency term.

3.5 Results

3.5.1 Performance

We tested our algorithm on several input cases and evaluated our performance with respect to the correctness of our motion grouping output.

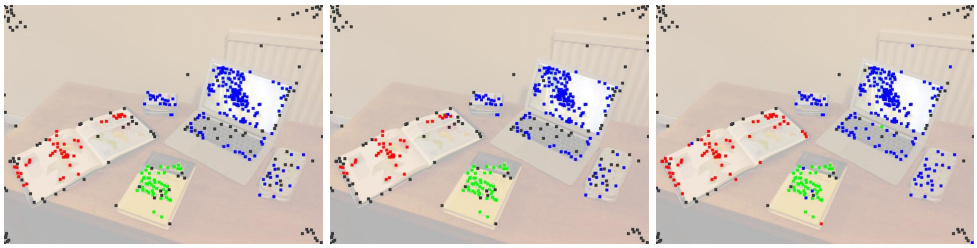
As shown in Figure 3.11, our algorithm generates accurate grouping results and overperforms comparable state-of-art methods, i.e., PEARL. Our motion interpolation (Figure 3.10) allows us to qualitatively inspect the high quality structure generated by our method. The runtime of our optimization depends on the number of correspondences and motions. For the presented scenes, our algorithm takes 30 minutes to converge on average (1 hour in worst case).

3.5.2 Evaluation

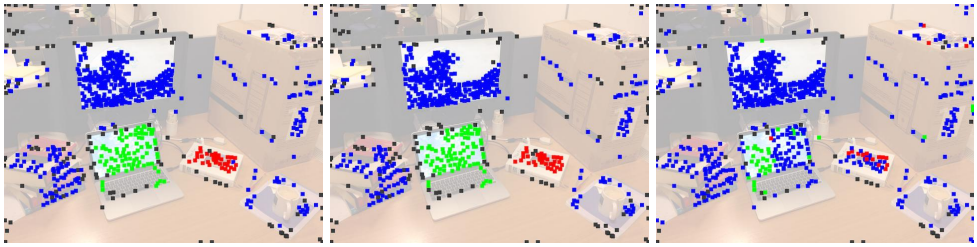
Ground truth. We performed experiments to evaluate our labeling performance on real data, and correctness of structure on a synthetic scene. To create the ground truth labeling, we first manually select correct correspondences within the same motion and label them into the same motion groups. We then estimate the groundtruth transformation for each motion with the selected correct correspondences. With groundtruth transformation, we can perform a raw grouping of correspondences and an annotation of outliers. We evaluate our output point cloud structure later in this section by comparing our reconstruction results to a synthetic scene, where the ground truth structure is known.

Pre-boosting. As shown in Figure 3.8, our pre-boosting method increases the number of correspondences between the image pair from 554 to 1046, with an outlier (mismatch) ratio 10.9% (114/1046). We define the outlier ratio as the ratio of the number of mismatch correspondences with respect to the number of all correspondences. Our measurements showed, that after pre-boosting we on average over our test scenes arrive to an outlier ratio of 10%, which, as later shown, can be handled well by the initialization of our optimization. In comparison, Figure 3.12 shows the result of ASIFT [27] and we compare favorably in terms of consistency of the match orientations.

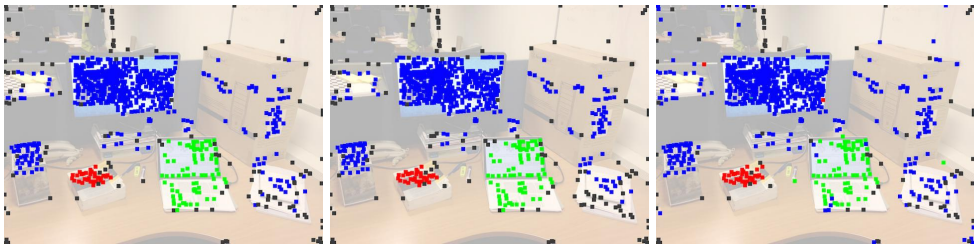
Initialization. Our pre-boosting effectively generates a lot more correspondences, however, with a moderate ratio of outliers. This is directly related to the reweighted RANSAC strategy we apply to calculate camera poses for initialization. In Figure 3.13, we show that our initialization is insensitive to the outlier ratio of the input correspondences given a single, universal parameter (weight decrement factor = 0.2



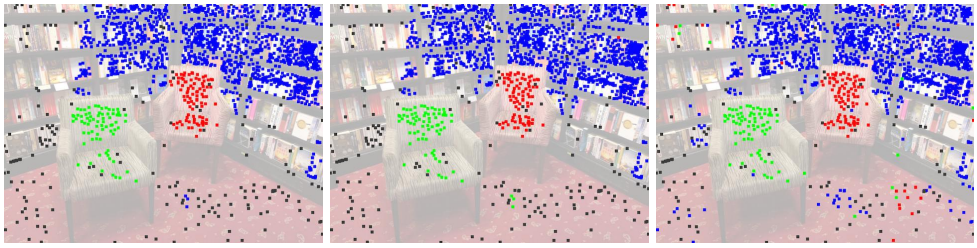
(a) GT, 3, 467 **(b)** Ours, 97.4% **(c)** PEARL, 88.2%



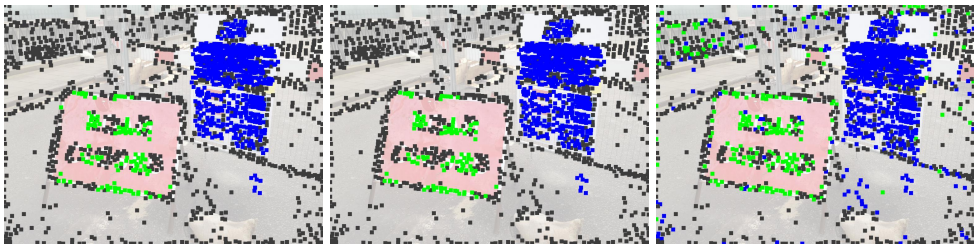
(d) GT, 3, 861 **(e)** Ours, 97.4% **(f)** PEARL, 80.7%



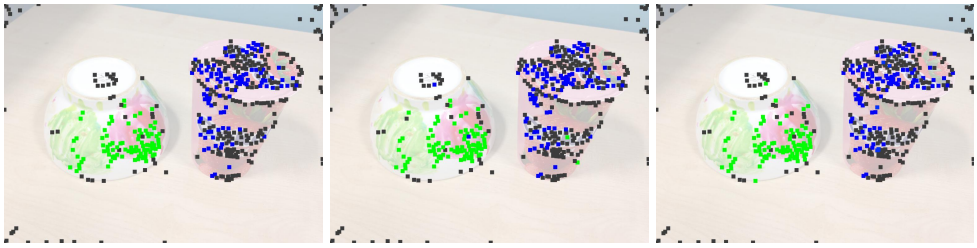
(g) GT, 3, 1046 **(h)** Ours, 98.5% **(i)** PEARL, 93.4%



(j) GT, 3, 1867 **(k)** Ours, 99.6% **(l)** PEARL, 89.1%



(m) GT, 2, 2283 **(n)** Ours, 99.0% **(o)** PEARL, 86.7%



(p) GT, 2, 497 **(q)** Ours, 98.0% **(r)** PEARL, 95.8%

Figure 3.11: Algorithm performance. Groundtruth with number of motions and number of correspondences (inlier and outlier), our result and PEARL's result with labeling correctness. Note that the PEARL was provided with our robust initialization.

during our experiments). For evaluation, we manually delete outliers or add random correspondences to the output of the pre-boosting step to simulate the change in outlier ratio. We successfully show, that our initialization is able to generate high-quality camera model candidates with very limited redundancy at different levels of outlier ratios, both with and without the domain problem.

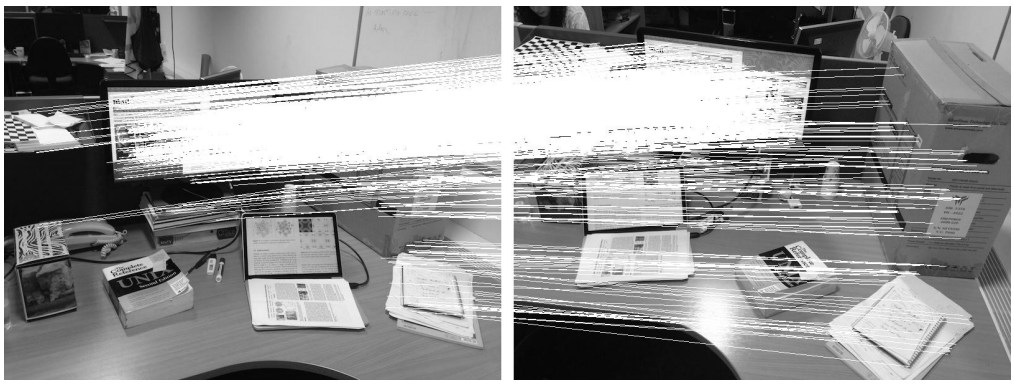


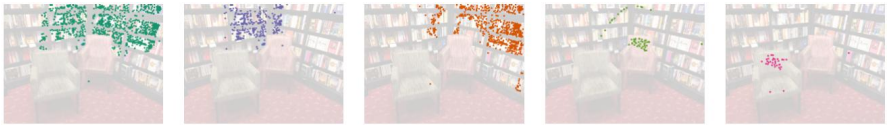
Figure 3.12: ASIFT result on input presented in Figure 3.8.

Correspondence grouping. The validation of our optimization consists of two aspects: correctness of group labeling and reconstruction quality. Group labeling is more essential as we can apply any structure from motion method consecutively to our output segmented correspondences. We first show the results with and without the adaptive outlier weight. Since we only use two input views, there might be some outliers (mismatches), that reproject well in the two images and therefore cannot be penalized by our data term. In these cases, our algorithm relies on the mismatch penalty whilst producing the reliable outputs. We then investigate the effect of our complexity and consistency terms. Our consistency term is most effective in resolving problems caused by two-view ambiguity as described in Section 3.3.1. We demonstrate the utility of the different energy terms in Figure 3.14. Omitting the consistency term results in neighboring points getting mislabeled, and prevents us from benefiting from redundancy residing in local context. Similarly, if the quality of correspondences in the local neighborhoods is not taken into account when identifying outliers (Figure 3.14c), camera estimates become less accurate due to a higher rate of false-positive matches being used or more reliable contributions being culled.

Point cloud structure. One of the main reasons, why we generate structure simultaneously with the motion segmentation is, that point clouds with correct structure will further enhance the segmentation of correspondences to consistent motions. To evaluate our reconstruction quality, we ran our algorithm on a synthetic scene (Figure 3.15). We measured reconstruction quality with respect to average depth error and groundtruth depth for each point. Our output point cloud (after post-boosting) has 5% depth difference ($\frac{\text{average depth error}}{\text{average groundtruth depth}}$) on average over the four objects. This allows us to enhance the motion segmentation and enables the further application of our outputs.

(a) No outlier correspondence.

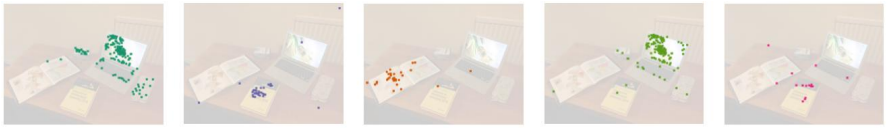
(b) With originial outliers generated by our pre-boosting.



(c) Add 20% outliers

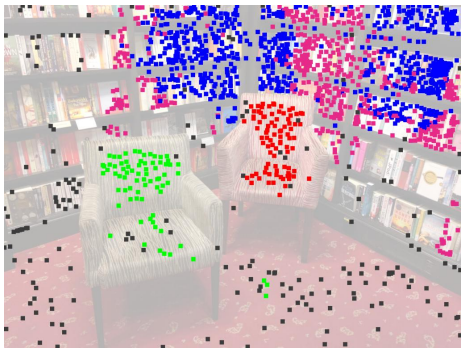
(d) No outlier correspondence.

(e) With originial outliers generated by our pre-boosting.

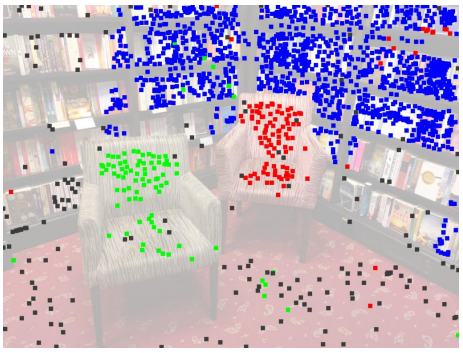


(f) Add 20% outliers

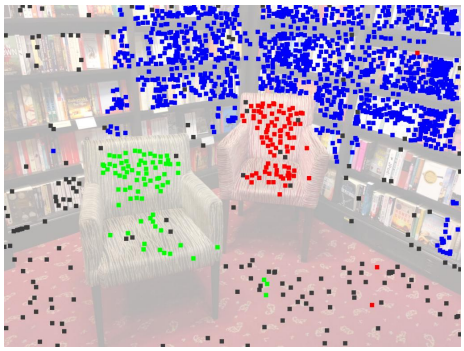
Figure 3.13: Initialization with different outlier ratios in different types of scenes (with and without domain problem).



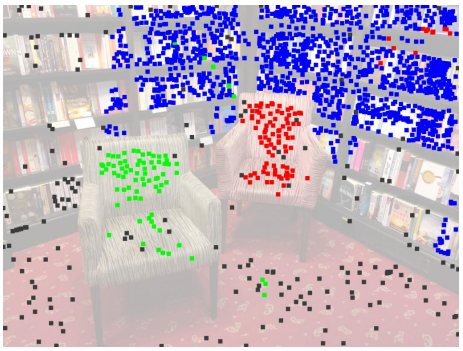
(a) no complexity term, 4 groups are generated.



(b) uniform outlier penalty without consistency, correctness 97.4%



(c) uniform outlier penalty, correctness ratio 98.8%



(d) no consistency term, correctness ratio 98.3%

Figure 3.14: Evaluation of our different energy terms. Note that the correctness ratio achieved using our full formulation is 99.6%, as shown in Figure 3.11.

3.5.3 Comparison

We compared our method to the state of the art method PEARL [20]. In Figure 3.16, we ran PEARL based on our pre-boosting result but with PEARL's initialization and based on direct SIFT matching results. In Figure 3.11, we show the performance of an improved PEARL version, that is based on our initialization. This example shows well, that although PEARL performs well on a manually annotated benchmark dataset, it actually falls short in real world cases, where the ratio of outliers is not strictly zero and/or the domain problem complicates the inference. Our method, on the other hand can be applied in real life scenarios, since it can cope with the above problems arising.

3.5.4 Evaluation on *Hopkins155*

To better evaluate the labeling efficiency, we ran our algorithm on a scene from *1R2RC* in the *Hopkins155* dataset [19]. We test our method using the first and the

last frames of the sequence as the input image pair. For fair comparison with other methods, we designed the experiment to evaluate labeling correctness only, i.e., we used the feature point locations and correspondences provided by the dataset as input. Note that the omitted pre-boosting and post-boosting steps represent a significant part of our contribution, rendering our method more general then the algorithms attempting to solve the Hopkins155 dataset. Figure 3.17 shows how our method performs with a correctness ratio of 98.7%. Note, that the input of *Hopkins155* dataset is manually annotated. Hence, it does not suffer from outliers or sensor noise, which contradicts our data assumptions. Further, in this specific example, the relative viewpoint of the rotating box (marked by green dot) changes very little between the first and the last frames, which does not provide our method with enough information to perform the structure estimation.

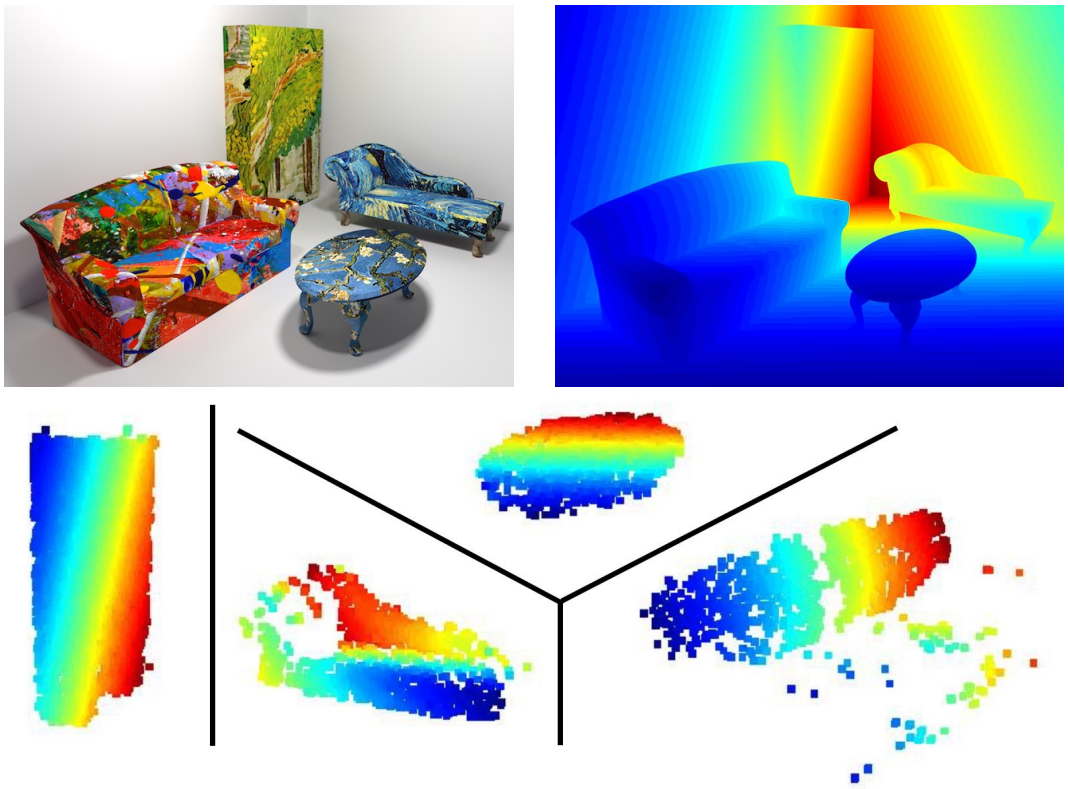


Figure 3.15: Depth map to show our output structure. Upper-left: synthetic scene; upper right: depth ground truth, from blue (close) to red (far); lower row: depth map for each object we detect after post-boosting. Due to two-view ambiguity, we actually have no information about the relative depth between objects. Therefore, we only show the depth map individually for each object.

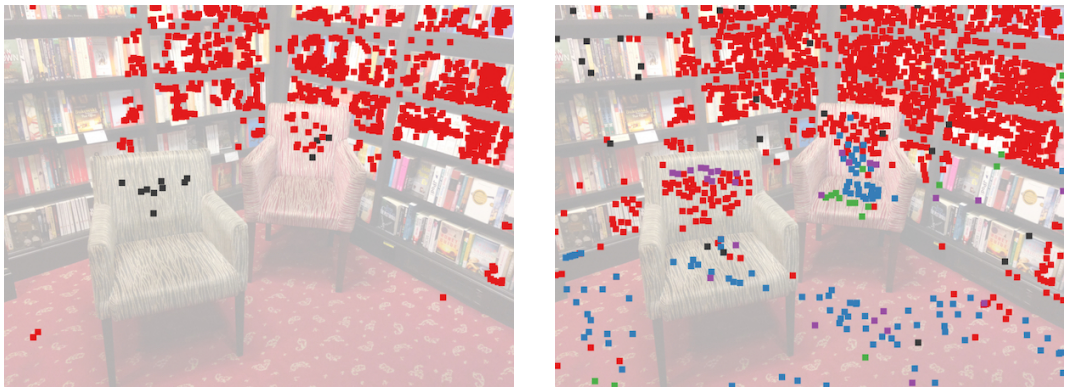


Figure 3.16: Comparison with PEARL. Left: running PEARL after extracting and match SIFT features directly. Right: running PEARL based on our pre-boosting strategy. Dark dots indicate outliers. There are three parts in the scene with 1482/83/101 inlier correspondences and 201 (10.7%) outliers. In this case, we show that PEARL fails in a real world scenario when outliers and the domain problem exists.

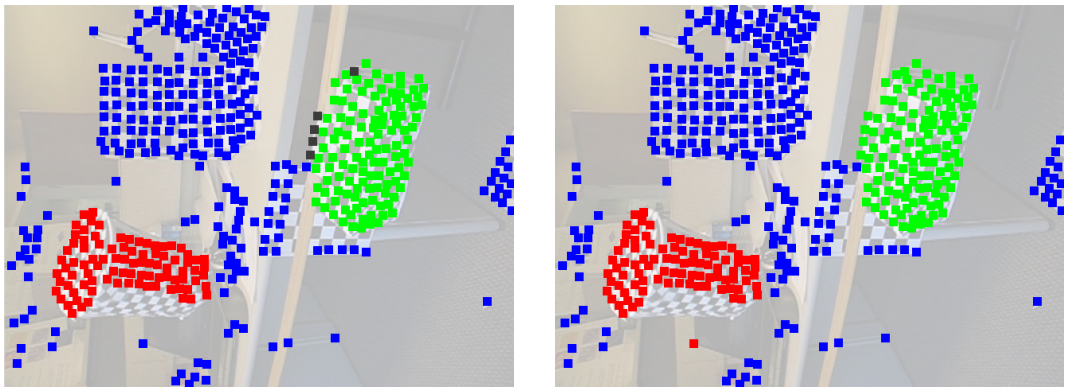


Figure 3.17: Results based on scene *1R2RC* from the Hopkins155 dataset. Input was the first and last frames of the video sequence. Left: our result with a correctness ration of 98.7%. Right: PEARL's result with a correctness ration of 99.5%. Note, that the input correspondences here are perfectly free of noise, i.e. no outlier correspondences are present, which is not a realistic real-life scenario.

3.6 Limitations

Our method has two main limitations. First, as our algorithm is based on feature points extracted directly from the images, objects with less texture cannot be recognized well. Therefore those objects will be ignored. Second, as we estimate the structure from a pair of images, it is necessary, that the relative pose of an object in the two images is sufficiently different to ensure the robustness of the 3D reconstruction, otherwise there is simply not enough information in the raw input. Empirically, we have established, that the relative view angle w.r.t. the objects should change more than 10 degrees, as confirmed by [119].

Figure 3.18: Typical failure scenarios. Left: low number of initial correspondences due to lack of texture usually causes our algorithm to miss some of the important structures in the scene. Middle and right (*car10* in *Hopkins155*): although the bus is moved, the change of perspective of the bus is barely noticeable leading to less robust structure estimation.

Chapter 4

Automated Texture Transfer from Images to Model Collections

Large 3D model repositories of common objects are now ubiquitous and are increasingly being used in computer graphics and computer vision for both analysis and synthesis tasks. However, images of objects in the real world have a richness of appearance that these repositories do not capture, largely because most existing 3D models are untextured. In this work we develop an automated pipeline capable of transporting texture information from images of real objects to 3D models of similar objects. This is a challenging problem, as an object's texture as seen in a photograph is distorted by many factors, including pose, geometry, and illumination. These geometric and photometric distortions must be undone in order to transfer the pure underlying texture to a new object — the 3D model. Instead of using problematic dense correspondences, we factorize the problem into the reconstruction of a set of base textures (materials) and an illumination model for the object in the image. By exploiting the geometry of the similar 3D model, we reconstruct certain reliable texture regions and correct for the illumination, from which a full texture map can be recovered and applied to the model. Our method allows for large-scale automated production of richly textured 3D models directly from image data, providing plausible virtual objects for 3D scene design or photo editing applications, as well as a wealth of data for training machine learning algorithms for various inference tasks in graphics and vision.

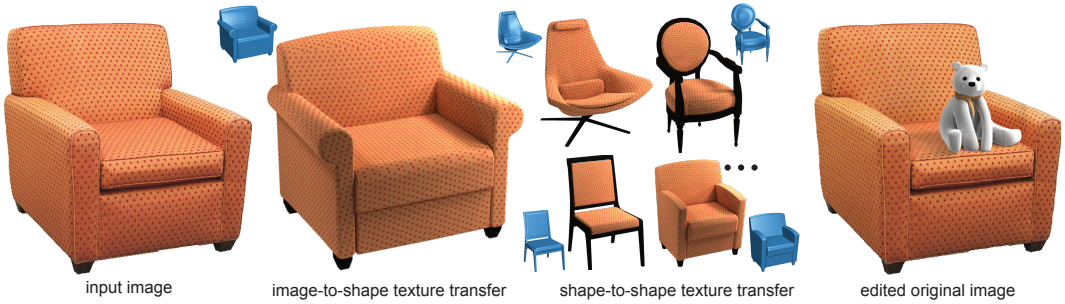


Figure 4.1: Starting from an input image and a 3D model collection, we propose a scalable method free from human intervention for image-to-shape and shape-to-shape texture transfer. The method also allows novel object insertion to the original image (right). The method exploits approximate geometry priors to factorize both geometric and illumination effects. Corresponding original 3D models are shown in blue.

The work presented in this chapter was developed and written in a collaboration with multiple parties, and published as [4].

4.1 Introduction

Synthesizing realistic 3D objects and scenes remains a central goal of computer graphics. With the growing availability of 2D image and 3D model collections, such as ImageNet [124] and ShapeNet [125], significant efforts have been made in recent years to *jointly* harness the complementary nature of the two collection types. Images capture detailed appearance information and provide object context in real world scenes, but lack depth and information about occluded areas. On the other hand, 3D models have rich full-object geometry, but often lack realistic textures needed for high-quality renderings. While significant progress has been made in transferring information across the two collections for pose estimation [29, 30], depth estimation [34], and image-driven shape segmentation [126, 127], the task of marrying 2D image textures with 3D geometry has remained elusive.

In this paper we study how to efficiently transfer texture information from 2D images to 3D shapes. Our focus is the simplest version of the problem: *given a single 2D image of an object and a part-level segmented 3D model of a similar but not necessarily identical object, how can we transfer texture information from the image to the model?* The eventual goal is to generate a fully textured model whose appearance agrees with the image in the matching view. Given the abundant

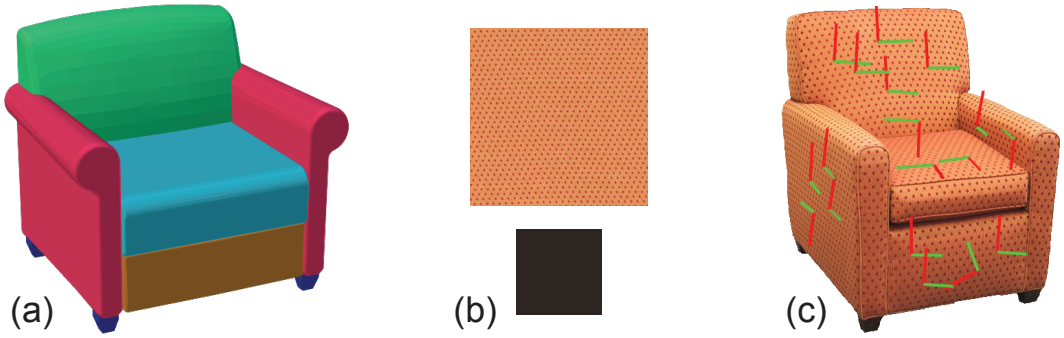


Figure 4.2: The retrieved part-level segmented model (a) for Figure 4.1, the extracted base texture patches (b), and the recovered orientation field shown as crossfield on the 3D model.

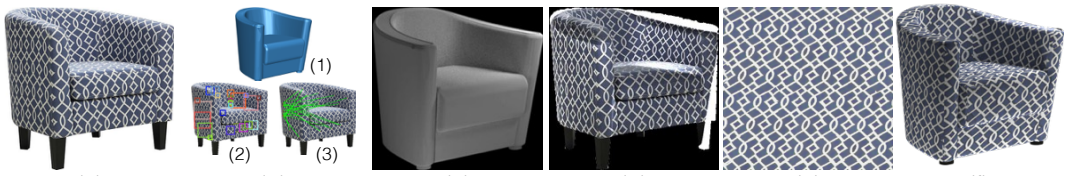


Figure 4.3: Pipeline overview. Our system takes an input image (a). A geometrically similar shape is then retrieved (b1). According to the estimated geometry we find large patches on the image (b2) and detect their correspondences (b3). The geometry also helps to factor out shading (c) and reflectance (d), so that base textures (e) are extracted with little distortion and homogeneous lighting. Final texture transfer can be applied to the retrieved model (f).

availability of image data, this immediately enables the generation of large quantities of high-quality 3D models with realistic textures, in a human intervention free setting. Synthesized models can then directly be used in computer graphics content creation applications such as object insertion in images, or be used to provide extensive training data as required by various machine learning-based algorithms for 2D or 3D inference tasks (e.g., classification, depth estimation, shape segmentation, etc.).

This problem may seem relatively straightforward, as there are now good tools for aligning images with similar 3D models. The obvious approach would be to try to establish dense correspondences between image and model pairs. However, the imaged real object and the approximating 3D model often exhibit significant difference in shape, both in terms of geometry and topology, making such correspondences difficult to obtain or even define. Furthermore, the input image provides only partial texture information for the 3D model, as occluded areas are not visible.

At a more fundamental level many challenges remain. Firstly, even in visible

areas the image texture is distorted by the interaction between the unknown real 3D object geometry and its projection into 2D image space. Furthermore, beyond geometry, complex lighting and shading effects in the image formation process also make it non-trivial to transfer the pure underlying texture from the image to the 3D model – explicit analysis is required to factor out such lighting effects. In other words, we have to decouple the texture distortions arising from object geometry and perspective projection from those due to illumination effects, and correct for both.

Our key observation is that images of many man-made objects, even those with complex textures, can be described by appropriately factorized low complexity models, separating texture and illumination information. Specifically, the appearance of an imaged object can be explained in terms of a small number of base texture patches and their orientation in different parts of the object, as well as an illumination model for the object. Both the base textures and the lights are unknown and have to be simultaneously estimated. We demonstrate how to solve the above difficult inverse problem with the help of a 3D model, which only needs to be *approximately* similar to the imaged object. This (proxy) 3D model provides enough of a geometry base to allow us to decouple geometric distortions from illumination effects, and recover both at the same time, so that their effects on the pure underlying texture can be removed. More specifically, in the analysis stage, we detect flat regions from 3D proxy model and use that to extract texture patch candidates from input image. We use flat area because the accuracy of normal direction is less sensitive to the misalignment for flat regions. The 3D proxy model also regularize the solution of shading decomposition, therefore shading (including shadow) can be removed from texture patches. Once the pure texture patch is extracted, a large texture image will be synthesized in 2D using standard texture synthesis approaches. The retrieved pure base texture can then be used to transfer appearance to other similar 3D models. In Figure 4.1 for example, given an image I, base texture and illumination are extracted (shown in Figure 4.2) with geometric guidance from a roughly similar model M_1. The extracted texture is then used to realistically texture model M_1, which in turn helps transfer texture to other models such as M_2, M_3, etc. Finally, we can synthetically

add 3D models to the original input image (a teddy in this example).

We evaluate our method on three different image-model datasets, namely chairs, tables, and cushions. A user study indicates that users found it difficult to consistently distinguish between the synthesized and real images. In another test, we use the synthesized images to boost training data for machine learning algorithms. In particular, we train deep neural networks for depth estimation and texture-guided image retrieval. For both tasks we observe significant performance improvement by using the enriched shape dataset with more diversified textures. As an application, we show the suitability of our method for novel view synthesis and object insertion.

In summary, we formulate and solve the problem of object texture transport from images to 3D models by factorizing out geometric and perspective distortions from illumination effects and compensating for both. We extensively evaluate the performance of the method and demonstrate the utility of the results for different computer graphics and computer vision tasks.

4.2 Overview

In this paper we focus on clean textured object images, such as product images on the web that are now widely available. We also use publicly available 3D model repositories, such as ShapeNet [125]. More formally, our method makes three key assumptions: (i) a clean image of the textured object is available with the background removed, (ii) the texture pattern on the object is homogeneous within object parts and each texture pattern can be built from a texture element of a size that is small when compared to the full part size, and (iii) a similar 3D model is available that is segmented into parts. Given such a product image of a textured object and a similar untextured segmented 3D model, our basic objective is to transfer the underlying textures visible in the image to the corresponding parts of the model. We also aim to further propagate such texture information to many other related models in the same class. Since textured object images are very common on the web, this enables a novel automated pipeline that can vastly enrich the set of available textured 3D models.

Our algorithm deeply exploits the geometry of the 3D model in both correcting illumination artifacts (e.g., shading) on the object and in rectifying the underlying texture information that may have been distorted by the imaging process. We aim to recover the underlying ideal texture directly, undoing the geometric and photometric distortions mentioned above. In this fashion we avoid the need to establish dense image-model correspondences, a difficult task which is not very well defined in our setting since the 3D models and the imaged object are only *approximately* similar. As a byproduct, our algorithm also builds an illumination model for the object in the image, which can be utilized for image editing tasks.

In our approach (see Figure 4.3) we seek reliable texture patches in the image which can be geometrically 'unwrapped' and photometrically corrected so as to recover the true underlying texture for the corresponding object part. After aligning the 3D model to the object image, we rely on the proxy geometry provided by the 3D model for the texture patch unwrapping. We extract multiple texture patches from the image and attempt to align them based on their periodic structure by detecting corresponding points on the wnwrapped image plane. This allows us both to select the most reliable patches that best describe the base textures elements to be used, as well as to make decisions about material groups (which texture is to be painted on which shape part). We synthesize a texture image for each material group using the largest extracted reliable texture patches and also estimate the texture orientation (indicated by cross-fields) in each object part.

For the illumination correction we again use the 3D model. We sample point light sources over a sphere around the model and generate many grayscale shading images of the model from which we approximate the lighting configuration used in the image (approximated by a discrete distribution over the sampled points). This step extracts the intrinsic image and undoes illumination effects. As the 3D proxy model is not exact, this step is an approximation. However, the extracted illumination is sufficiently correct as preparation for our subsequent texture extraction step. Note that since our goal is to extract a set of base textures, we can only focus on the near planar regions and ignore the regions near boundaries.

For the final texture transfer step, we use a global texture synthesis step in the *uv* texture domain, after some local hole filling. We prefer a global method to avoid spreading artifacts that may still be present in the extracted texture patch. Furthermore, we need to respect the orientation of the texture relative to the object part geometry, as recovered from the image. This is crucial not only for the initial texture transfer to the proxy model, but also for transferring the texture to other 3D models.

The rest of the paper is organized as follows: Section 4.3 describes the core method for extracting appearance (texture and illumination) from image I; transferring this information from image I to shape S; Section 4.4 describes how to transfer texture information from shape S to other similar models, Section 4.5 presents evaluation results and finally Section 4.6 discusses applications.

4.3 Image to Shape Texture Transfer

The input to the image to shape texture transfer stage is (i) a given image I with clear background, (ii) a similar 3D object S with part-level segmentation, and (iii) an estimated camera pose $\mathbf{V}$ of I with respect to the coordinate system associated with S. In this paper, we use the method described in [36] to retrieve a similar shape S from a shape collection and to estimate the camera pose $\mathbf{V}$ (Figure 4.4). This task can also be accomplished by other methods, for example [31]. The remainder of this section describes the details.

4.3.1 Geometry-guided Patch Extraction

We assume that the imaged object in I has a low complexity appearance model, i.e., the object is covered with a repeating texture pattern and imaged under an (unknown) illumination setting. The key challenge is to factor out the projection and illumination artifacts in order to extract a set of simple base textures and an illumination model. We use the geometry information available in the form of shape S to guide both these tasks. First, we detect repeating elements in the input image and then use them to group parts of the input shape S to guide subsequent image decomposition.

The repeating elements consist of patches of the input image each of which has a salient regular texture pattern. Extracting such regular texture patterns directly is

Figure 4.4: Shape retrieval and alignment. Here we show 5 closest results of original and blurred image (to reduce texture effects). We use the closest shape (second column) for further estimation. For each image, we use the estimated view corresponding to the model in the second column.

difficult because the texture may be distorted by the geometry of the captured shape. To address this issue, we use the geometry provided by S to unwrap each texture pattern into a common image plane. As the object in the input image and the shape S are similar, we found that it is sufficient to simply overlay the input image object and the rendered image for depth and normal transfer as explained in the following.

Patch initialization. To identify the patches, we uniformly sample a 50×50 grid of points on the input image (Figure 4.5(f)). From each grid point, we start a region growing step to find the largest square flat region on S centred at that point. The criteria for "flat region" is that within such a region, the normal difference and gradient of depth is less than a certain threshold. In this way we grow a patch around

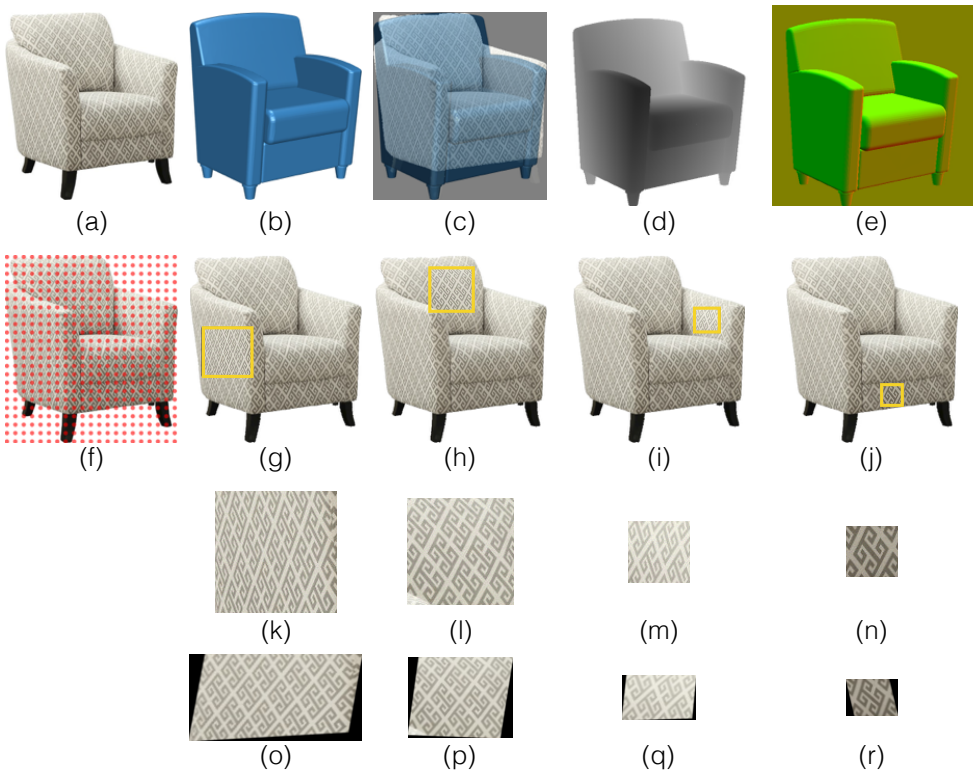


Figure 4.5: (a): Input image I; (b) retrieved shape S; (c) overlaid I and S for illustration; (d) depth map corresponding to #b; (e) normal map corresponding to #b; (f) uniform grid sampling; (g)-(j): 4 examples of cropped patch from I; (k)-(n) patches directly cropped from I corresponding to highlighted area in (g)-(j); (o)-(r) unwrapped patches into a common image plane to remove perspective distortions.

each grid point, extending it from pixel i to pixel j when

$$\left.\begin{array}{l} \angle_{i,j}(\mathbf{N}_i, \mathbf{N}_j) \leq \eta \\ \max_i(\nabla D_i) \leq \xi \end{array}\right\}, \tag{4.1}$$

where $\mathbf{N}_i$ and $\mathbf{N}_j$ denote normals on the mesh S corresponding to pixels i and j, ∇D_i denotes difference in depths on points of S around i, and η and ξ are threshold margins ($\eta = 5^\circ$ and $\xi = 3\%$). We keep all patches that cover more than 400 pixels. We denote these initial patches as $\{P^1, P^2, \dots\}$, which are allowed to overlap each other.

Patch rectification. The appearance of the initial patches may be distorted due to the geometry of the imaged object. Hence, next we use the proxy geometry in the form of S to correct the patches. Specifically, to recover from the distortion of $\{P^1, P^2, \dots\}$, we unwrap them into a common image plane. We denote the unwrapped patches as $\{P_1, P_2, \dots\}$. Unwrapping from $P^i to P_i$ is straightforward since P^i is on an almost flat region by construction. While this can be done by a local parameterization approach, we found a much simpler option to be sufficient as explained next.

For a patch P^i , let the four corners in clockwise direction be $\{q^1, q^2, q^3, q^4\}$. Correspondingly, let the unknown four corners of the unwrapped patch P_i be $\{q_1, q_2, q_3, q_4\}$. We simply flatten the patch into a plane by 'unfolding' it to a flat configuration while best keeping orginal lengths/areas. We fix q_1 to the origin on a 2D plane, set edge $\overline{q_1 q_2}$ to be the X-axis, then unfold $\triangle q^1 q^2 q^4$ into $\triangle q_1 q_2 q_4$, and finally $\triangle q^2 q^3 q^4$ into $\triangle q_2 q_3 q_4$. We keep the scale ratios of the form $\|q_i q_j\| / \|q^i q^j\|$ for triangle edges to be fixed for different patches. Since the original patch is flat, this step rarely leads to any foldover. Thus, we obtain rectified texture from different areas of the shape as shown in Figure 4.5.

Patch correspondences. We now establish correspondences across the rectified initial patches to extract a global repeating structure. Moreover, the extracted correspondence across the different patches indicates whether they capture the same texture pattern (or are outliers) and will later help us to group parts based on material (i.e., texture) assignment and to estimate the illumination environment.

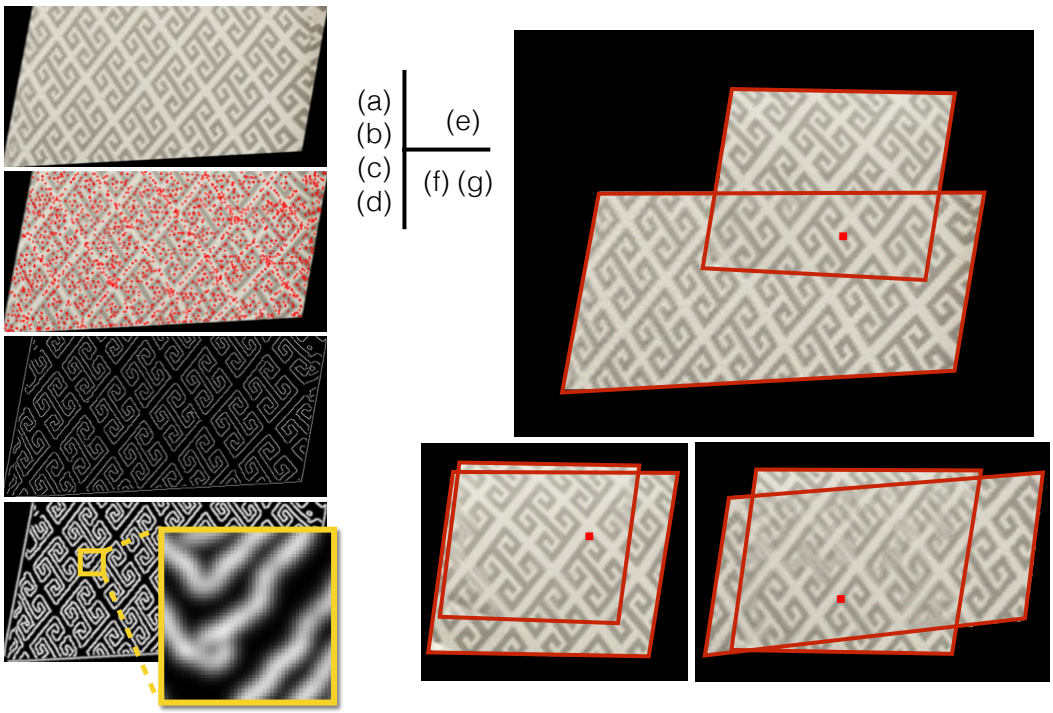


Figure 4.6: Patch registration between a pair of unwrapped patches. (a) unwrapped patch; (b) SIFT feature points; (c) Canny edges; (d) distance field of Canny edges; (e-g) examples of registration with red dot denoting corresponding feature point.

We start by registering a pair of unwrapped patches P_i and P_j. We first compute SIFT features [26]) on P_i and P_j (we denote the patch with fewer SIFT feature points as P_j). Let the m-th feature point of patch P_i be denoted f_m^i. For each such feature point, we have the location (x,y), an orientation r, and a 128-dimensional descriptor d. When registering $P_j toP_i$, we simply pick a matching pair of feature points $(f_{k_1}^i, f_{k_2}^j)$ and perform a rigid transformation on P_j, where the difference between the feature locations gives the translation and the difference between the feature orientations gives the rotation.

Since the patches can overlap, the SIFT features in $(f_{k_1}^i, f_{k_2}^j)$ are not helpful for alignment in regions of overlap. Hence, in order to robustly register the two patches, we select the 20 best matches for each f_k^j and among these $20 \times |f^j|$ matches, we pick the one with minimum registration energy $E_r(f_{k_1}^i, f_{k_2}^j)$. This energy gives low values when: (i) the Canny edge of two patches agree, and (ii) the overlap region of

the two patches are large. Specifically,

$$\begin{aligned} E_r(f^i_{k_1}, f^j_{k_2}) &= \|G(P_i) - G(T_{k_1k_2}(P_j)\|_2 \\ &+ \lambda * A(M(P_i, T_{k_1k_2}(P_j)), \end{aligned}$$

where $T_{k_1k_2}(\cdot)$ is the transformation guided from $f^j_{k_2} to f^i_{k_1}$, $M(\cdot,\cdot)$ is the mask of the union of two patches, $A(.)$ denotes the area of the overlapping region, and $G(\cdot)$ is a distance field of the extracted Canny edge (using OpenCV implementation). In our tests, we set $\lambda = -0.1$ and the distance field is generated by performing a Gaussian blur over Canny edge with kernel size of 15. Figure 4.6 shows some examples.

After two patches are registered with minimum registration energy, we select a 6×6 pixel square around the matched feature points on both P_i and $T_{k_1k_2}(P_j)$ denoted as ST_i and ST_j. We then determine whether the registration is successful based on the similarity between ST_i and ST_j. Since, at this stage, there might be an illumination effect, we simply account for that using a shading intensity ratio k and the similarity between ST_i and ST_j is defined by $\min_k(\|ST_i - k \cdot ST_j\|_2)$. Finally, if the similarity value is under a certain threshold, we mark the patch P_i and P_j to be sharing the same texture after a rigid transformation $T_{k_1k_2}$. Otherwise, they are marked to be different.

4.3.2 Material-Guided Patch Grouping

Directly performing part level texture transfer is difficult for two reasons: (i) parts can be totally occluded in image I and (ii) even for the partially visible parts, the texture information extracted from image I could be limited thus preventing high quality texture synthesis. However, for most of the real-world examples, many parts share the same texture (or plain color), thus providing redundancy. We exploit this redundancy to solve the above problems by grouping the original shape parts, with parts within the same group being assigned the same texture. We assume the models to have symmetric parts to be grouped.

We group the parts based on texture correspondence, thus producing material-based groups.Specifically, we test all patch pairs (P_i, P_j) if they are from different groups and merge the two groups if P_i and P_j are registered successfully (as described

before). After all patch pairs are examined, some parts can still remain untouched with no associated texture patch. We heuristically complete the material grouping as: (i) for thin parts with no associated textures, we treat them as plain color and use the average pixel value projected on those parts after SIFT flow [128] is applied on the image I (see Figure 4.7); and (ii) for larger parts, we link them into the group with the largest texture patch generated. For example, if a part at the back of the chair is completely occluded, we assign it to the largest texture group, which typically is the one associated with the seat cover. Next, we will synthesize one texture image for each material group based on the notion of a trustable region from the largest associated texture patch.

4.3.3 Image Decomposition

The patch correspondences can also guide the extraction of illumination from input image I. In order to perform intrinsic image decomposition, we decompose the captured photograph I into reflectance I_r and shading I_s, since the render equation tells us $I = I_r \cdot I_s$ for each channel of each pixel. In our setting, this implies that for corresponding patches, the I_r component at the corresponding feature points should be the same because they share the same texture location. In other words, when the (unknown) groundtruth shading of this image is factored out from I, I_r for

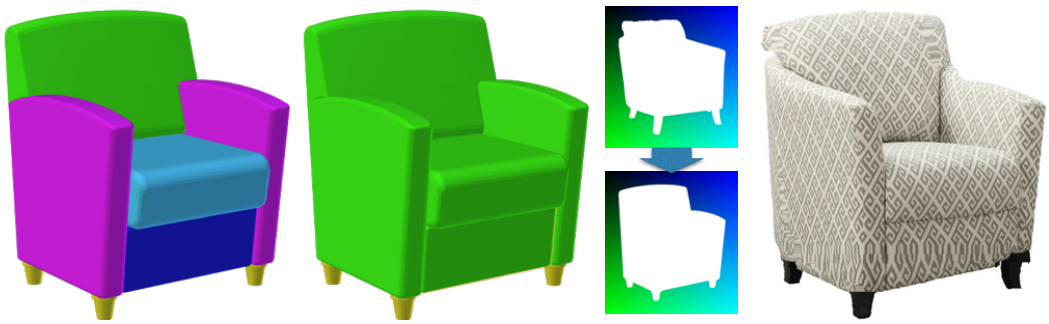


Figure 4.7: For the image/shape pair on Figure 4.5: (a) geometry-based grouping; (b) material-based grouping; (c) calculated SIFT flow from silhouette of image I to silhouette of shape S rendered from the estimated view; (d) apply SIFTflow to image I.

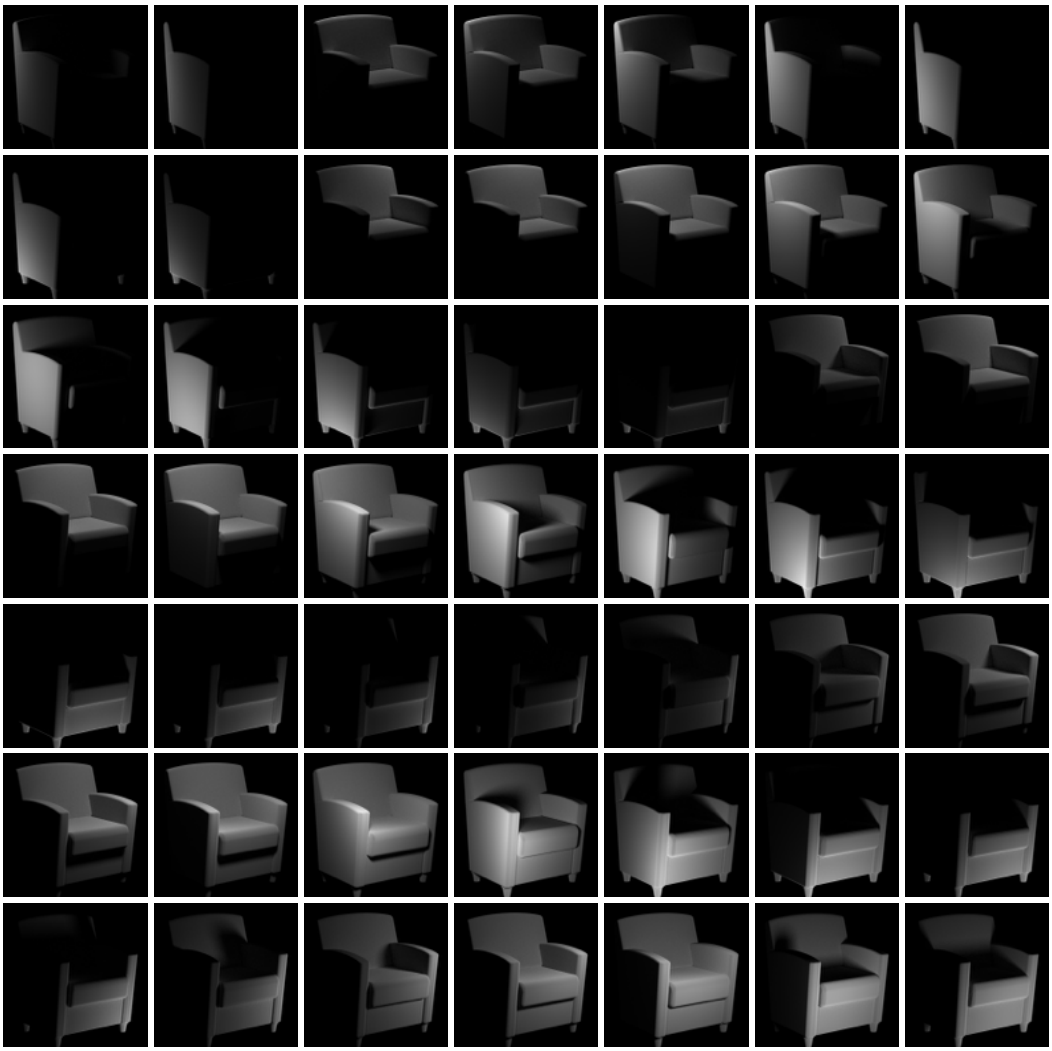


Figure 4.8: Shading samples using retrieved 3D model geometry. Only a selection of samples are shown.

corresponding locations should be the same. Thus we arrive at,

$$I_s := \underset{I_{s^*}}{\operatorname{argmin}} \sum_{(i,j)\in\Pi} \|I_{r^*}(l_i) - I_{r^*}(l_j)\|, \tag{4.2}$$

where $I_{r^*} = I/I_{s^*}$, Π denotes corresponding pairs with matched feature at location l_i and l_j on the image I, and $\|\cdot\|$ denotes 2-norm of the RGB channels. Note that I_s has RGB channels since we found environmental illumination often to be not pure white.

In the raw form, Equation 4.2 is underconstrained. As such, we further regularize I_{s^*} with shading priors. We again use the proxy geometry to drastically restrict the degrees of freedom. Specifically, we use the shading image of shape S to approx-

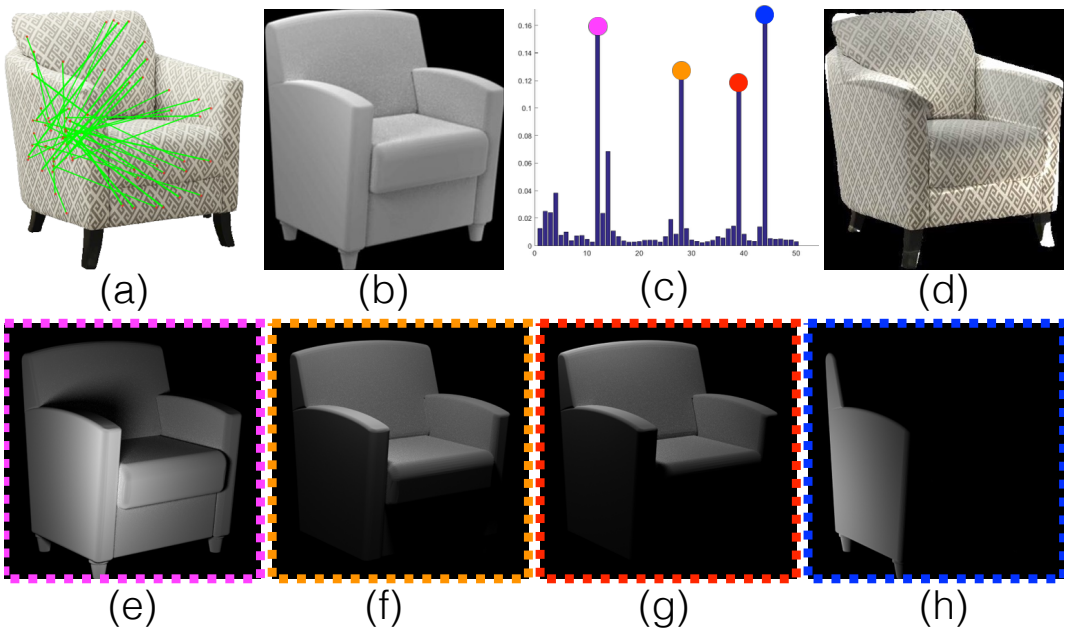


Figure 4.9: (a) Corresponded point pairs; (b) interpolated shading; (c) interpolation coefficients for the different shading samples; (d) shading factorized from the input image I. (e)-(h) shows the selected shading samples.

imate I_{s^*}. Exploiting the additive nature of illumination, we can decompose I_{s^*} into separate parts $I^1_{s^*} + I^2_{s^*} + \ldots$. To this end, we sample single point light sources on a sphere around S and render grayscale shading images $\{I_{s^1}, I_{s^2}, \ldots\}$ and approximate

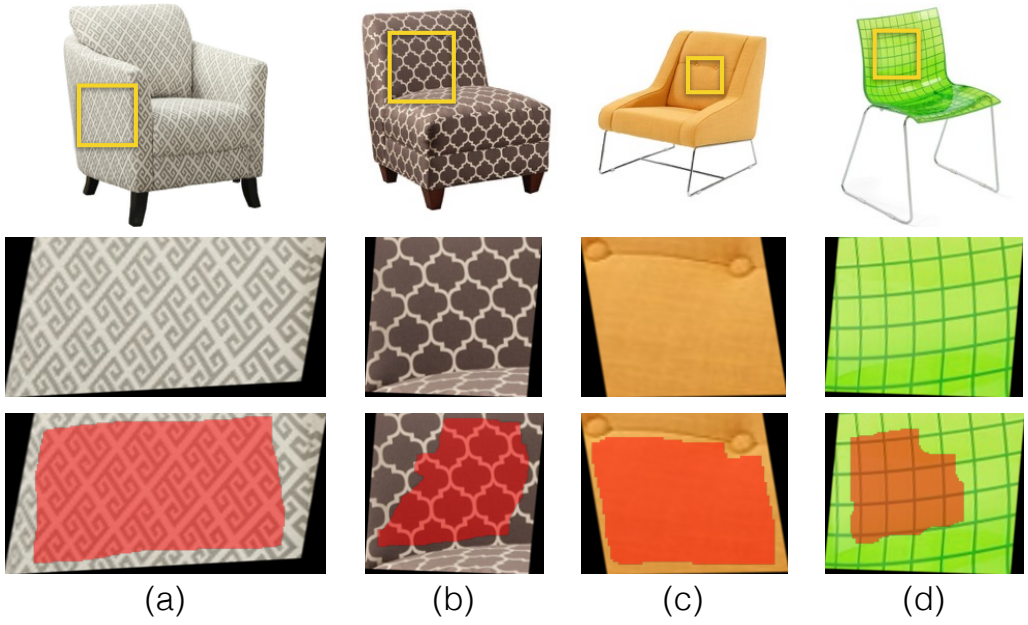


Figure 4.10: Some examples of artifacts captured by the largest texture patch and the trustworthy region after patch alignment.

I_{s^*} by interpolating the sample shading images over the RGB channel. Specifically, we place 100 individual point lights at locations $(\cos(u)\cos(v), \cos(u)\sin(v), \sin(u))$ where $u = \arcsin(1 - \frac{2k-1}{100})$, $v = u\sqrt{100\pi}$, and $k = 1, \ldots, 100$. For each point light, we render a grayscale shading image as shown in Figure 4.8. We ignore all fully black shading images as they do not contribute to the decomposition. Note that in this step, we preprocess the shading images to recover the effect of gamma correction. Therefore, via linear interpolation we have $I_{s^*} = \sum_{k=1,..,100} (c_k^{r,g,b} \cdot I_{s^k})$. Thus, illumination optimization amounts to,

$$\underset{\{c_k^t\}}{\operatorname{argmin}} \sum_{(i,j)\in\Pi} \|I_{r^*}(l_i) - I_{r^*}(l_j)\|, \tag{4.3}$$

where $I_{r^*} = I / \sum_{k=1,..,100} (c_k^{r,g,b} \cdot I_{s^k})$ and the coefficient vector regularized as $\sum_{k=1,..,100} c_k^t = 1, t = \{r,g,b\}$.

Instead of providing an intrinsic decomposition with fine detail, I_{r^*} only results in reliable reflectance in large flat areas as shown in Figure 4.9. This is because shape S is not exactly the same as the object captured in image I, especially around the boundaries and image-model mismatch regions.

The above decomposition, however, is still suitable for our goal. First, we mainly care about the texture on the large areas where patch texture could be generated. Second, we do not have to rely on other prior information as in state-of-the-art intrinsic decomposition methods [28, 129] (see comparison in Section 4.5). Finally, the recovered coefficients can easily be converted to recreate the illumination by placing light sources with extracted coefficient as intensity for each channel.

4.3.4 Texture Transfer

Once we factor out the effect of illumination, we focus on reflectance I_r. We regenerate the unwrapped texture patches $\{P_1, P_2, \ldots\}$ based on I_r. As we assume per-part texture to be homogeneous, the largest patch usually captures all necessary texture information. However, it can locally suffer from artifacts (see Figure 4.10) due to image-shape inconsistency, unexpected patterns, high distortion around boundaries, etc. Hence, directly using the largest patch easily results in unwanted artifacts.

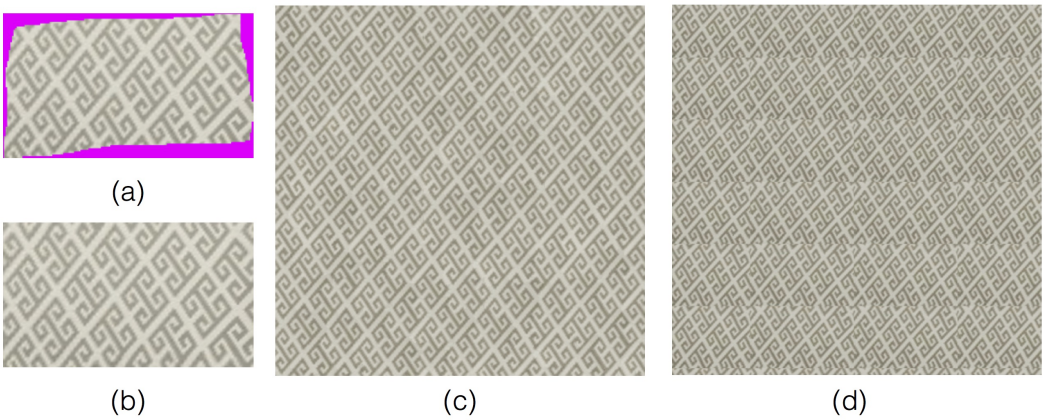


Figure 4.11: (a) Extracted trustable region; (b) hole filling by [130]; (c) texture synthesis by [131]; (d) directly repeating texture causes visible artifacts.

In this step, we extract useful texture information from $\{P_1, P_2, \dots\}$ so that we can synthesize high-quality textures for each material group. To this end, we adopt a simple but effective method: we introduce the notion of *trustable* region from the largest texture patch of each material group. The criteria for being a trustable region is whether the location texture appears more than once on image I_r. Specifically, for each material group, we perform pairwise patch registration between each patch and the largest patch. After registration, we examine all the corresponding 20×20 pixel patches on $ST_{i,j}$. We mark a region as trustable on ST_i if the sum of pixel value differences on the 20×20 pixel patch is less than 10%.

Texture synthesis. In this step, we generate a texture image for each material group based on the trustable region extracted from the last step. First, we perform hole

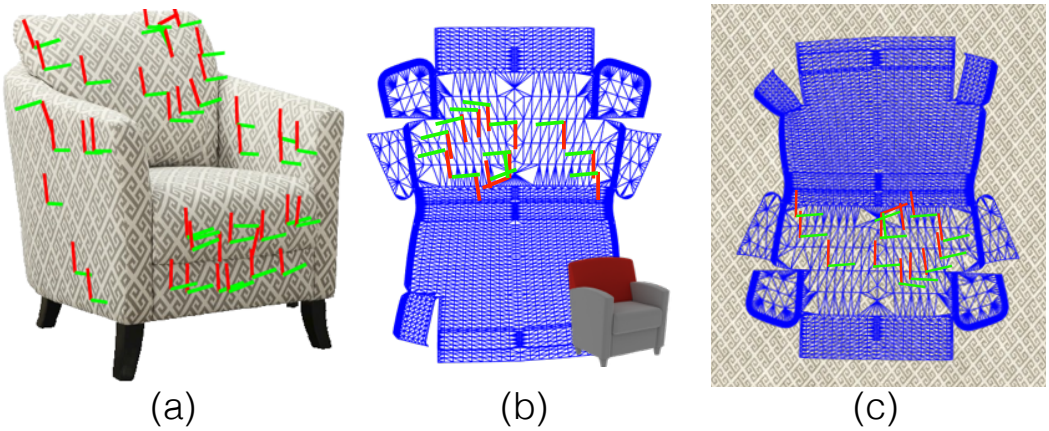


Figure 4.12: (a) Estimated orientation visualized on image I; (b) orientation frame projected on to uv plane; (c) uv-layout after rescaling and rotation.

filling using [130] to regularize the boundary of the trustable region. Then, we simply use [131] to synthesis a large size texture image. Note that we should preserve the scale of the texture as it is on image I when transferring to shape S. That means we cannot arbitrarily rescale the uv layout when parameterizing each part of S. In order to ensure good quality, we synthesize a large texture image to cover the uv layout instead of directly spatially repeating small texture examples. In our experiment, we found 2048×2048 as a reasonable size with acceptable synthesis time.

We apply the method in [132] (using Blender's built-in version called *Smart UV Project*) on each part of S for uv parameterization. By default, uv coordinates are scaled to $[0,1]$. Hence, we rescale the uv coordinates to ensure the number of pixels in texture space covered by the parameterization of the mesh is the same as the number of pixels in image I covered by the mesh when it is projected back to the image plane.

We can now transfer orientation of the uv layout to S by following image I. This is straightforward since the correspondence between a patch and the largest patch specifies a common orientation frame. We project the orientations (up and right vector) of all patches onto uv space and pick a dominant up-direction using RANSAC. Specifically, for all the orientations $\{r_1, r_2, \dots\}$, we set 20° as threshold and pick the orientation with the most number of inliers. We set the dominant up-direction as the mean orientation of these inliers. We align the dominant up-direction to the up-direction of the texture image. We flip the right direction on uv layout, if necessary based on texture registration score. See Figure 4.12 for clarification.

4.4 Shape → Shape Texture Transfer

In the previous section, we described how to transfer texture from an object in image I to a similar shape S. We now describe how to further diffuse the texture information to other similar 3D models M in the model collection. Note that we assume the model collection to be coaligned [133]. The main observation is that although the imaged object in I is too different from M to reliably generate base texture elements, it still provides valuable information as to how to *orient* the (extracted) base textures

from I to M. Intuitively, S acts as a bridge to transfer orientation information from image I to the target shape M, while the base textures are obtained based on the I-S analysis. Thus, the main task is to simply map orientation of uv layout.

Similar to part-level parameterization of S, we first parameterize each part of shape M. Given a part and its uv layout, for each face, we project the up direction in 3D space at the face center onto the surface and map the projected direction onto 2D uv space. If a face is nearly perpendicular to the up direction, we use the forward direction instead. After that, we use weighted RANSAC to compute the dominant up-direction based on the projected orientations, weighted by face area. This will give us a dominant up-direction for the corresponding parts S_* and M_* from S and M. By aligning the up-direction from M_* to S_*, we naturally get the texture orientation. This completes the texture transfer from the image I to the retrieved shape M.

4.5 Evaluation

In this section, we discuss comparison results with baseline methods, state-of-the-art alternatives, and also report evaluation results assessing the importance of each stage of the pipeline. Code and data is available online[1].

Datasets. We tested our pipeline with 70 'chair' images download by searching with keywords *chair, fabric chair, wood chair, etc.*. As shape collection, we used dataset for 'chairs' from the ShapeNet database. We used a sample of 1000 of them for our tests as they already provide enough geometry variance for this task. We also tested with two other classes, 'cushion' and 'table.'

4.5.1 Result gallery

We show a sample of texture transfer results for 42 images in Figure 4.15 (please zoom to see the results). For each image, we retrieve the closest shape and transfer texture from the image to shape as shown in the diagonal entry. After that, we perform shape-to-shape texture transfer from the closest shape to the other 41 shapes. The images are rendered from a novel view. Images rendered with estimated view are in the supplementary material. For the cushions and tables datasets, we show

[1] `http://geometry.cs.ucl.ac.uk/projects/2016/texture_transfer/`

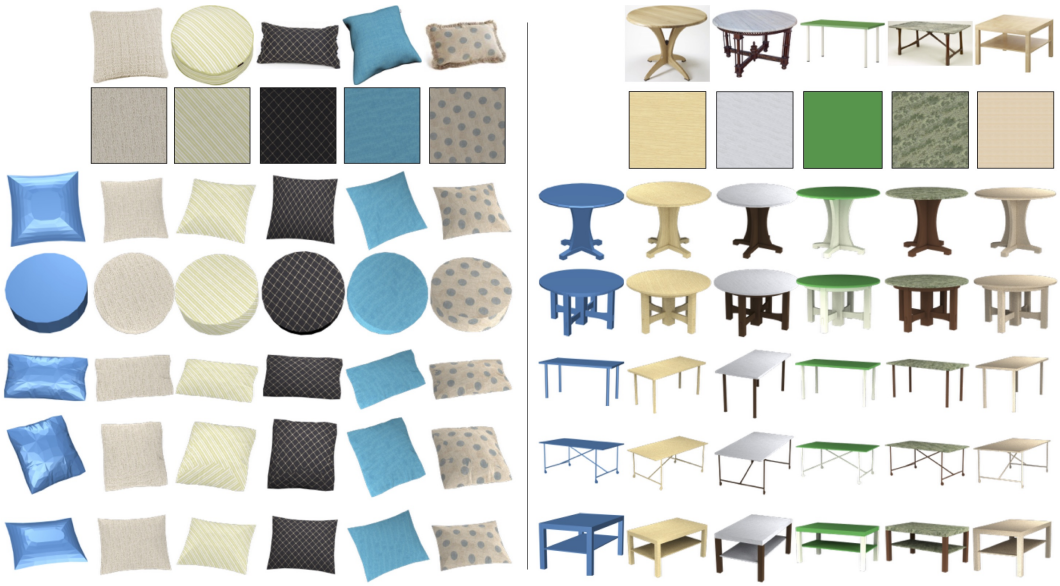


Figure 4.13: Image-to-shape and shape-to-shape texture transfer results for 'cushion' and 'table' classes.

results with 5 images for each category in Figure 4.13. We evaluate generality of our approach on different categories (cap, mug, vase, aeroplane) of objects in Figure 4.14. In this case we select 5 models from ShapeNet for each category.

4.5.2 Comparison with baseline methods

Baseline #1: SIFTflow-based projection. One simple baseline method for transferring texture from image to shape is to compute a dense correspondence between the image and a retrieved 3D model and then simply transfer texture information from the image to the projected model. As shown in Figure 4.7, we use SIFTflow technique to build such a dense correspondence between the silhouette of the image and the silhouette of the shape rendered from the estimated view. After applying SIFTflow, we project the pixels directly onto shape using the estimated dense correspondence. Note the original image is not corrected for shading effect. Figure 4.16 shows some example results.

Baseline #2: Shapenet textures. Another baseline is simply to take the texture information (when present) from the ShapeNet database. Please note that the quality of texture from ShapeNet is very variable. For example, in the chair category, about 50% of the shapes have meaningless textures, i.e., simple color assignment. On the other hand, a handful of the models (about $5-10\%$) come with high-quality texture,

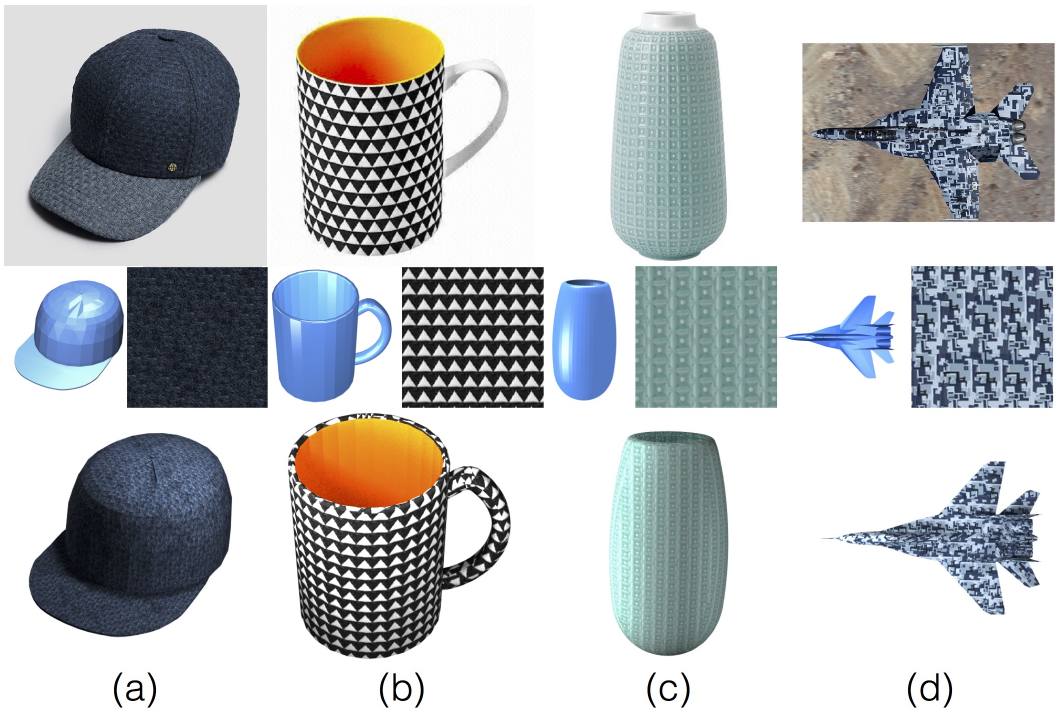


Figure 4.14: Texture transfer results for other categories. First row: input image. Second row: retrieved shape and extracted texture patch. Last row: re-rendered image with our texture transfer output and estimated illumination. Please note that the proxy shapes differ from the input images, such as the brim of the cap, handle of the mug, top of the vase, and aeroplane wings. Since our 'flat patch' is defined based on a threshold of normal difference, we can still extract a small but useful patch from a non-planar object such as the vase.

probably hand-curated by some professional.

User study. We conducted a user study on Amazon Mechanical Turk (AMT) to compare the results of our human intervention free method against the two baseline methods and also real images (product photographs). We tested on 4 datasets: (i) 70 real images; (ii) transfer texture of each such real images to the closest (retrieved) shape using our approach and rendered from the estimated view under estimated illumination; (iii) SIFTflow-based projection from the real images to the closest (retrieved) shape and rendered using our estimated illumination in a slightly rotated view from our estimation; and (iv) for each retrieved closest shape, we apply the associated texture from ShapeNet and render from the estimated view with our estimated illumination. Examples of the dataset are shown in Figure 4.16.

The user study was designed as follows: we combined all the 70×4 images and randomly selected 3000 pairs from these images. For each pair, we ask 3

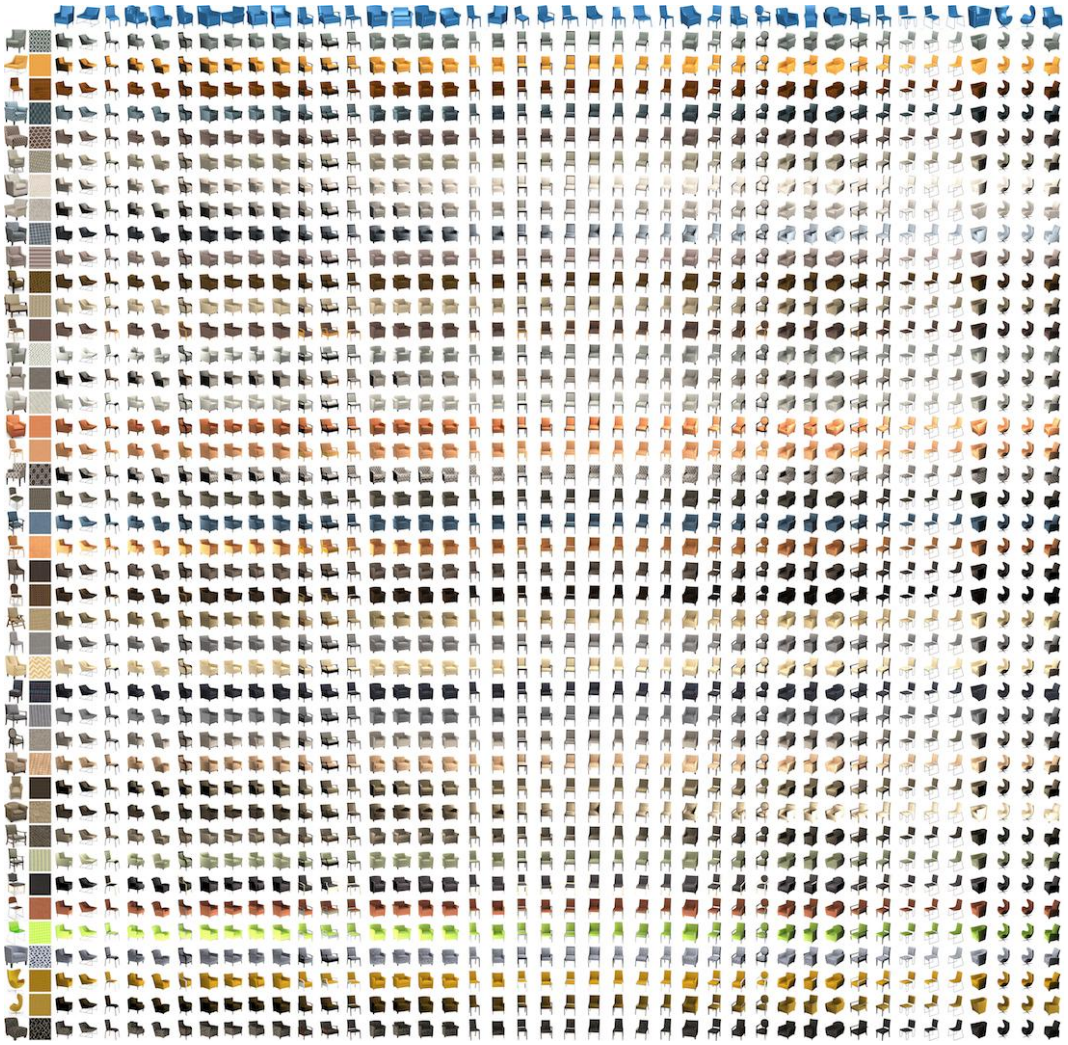


Figure 4.15: Appearance transfer on 'chair' dataset. Image-to-shape (diagonal entries) and shape-to-shape appearance transfer results. This is a high resolution figure, please zoom in to view details. The results (3D textured models along with recovered illumination setting) are also available for download as supplementary material.

different users to select the object they judged to be more realistic or plausible. (They were forced to choose one of the two images.) We analyzed the result using the Bradley-Terry model [134] to robustly predict the probability of each image *winning* against each other image. Figure 4.17 shows that the real images had the highest plausibility, only slightly better than our automatically synthesized texture transfer results. Results from the two baseline methods were easily identified as unrealistic.

Figure 4.16: Baseline examples. First row: input image (left), after applying SIFT-flow (right); second row: our results; third row: SIFTflow + projection; fourth row: ShapeNet textures.

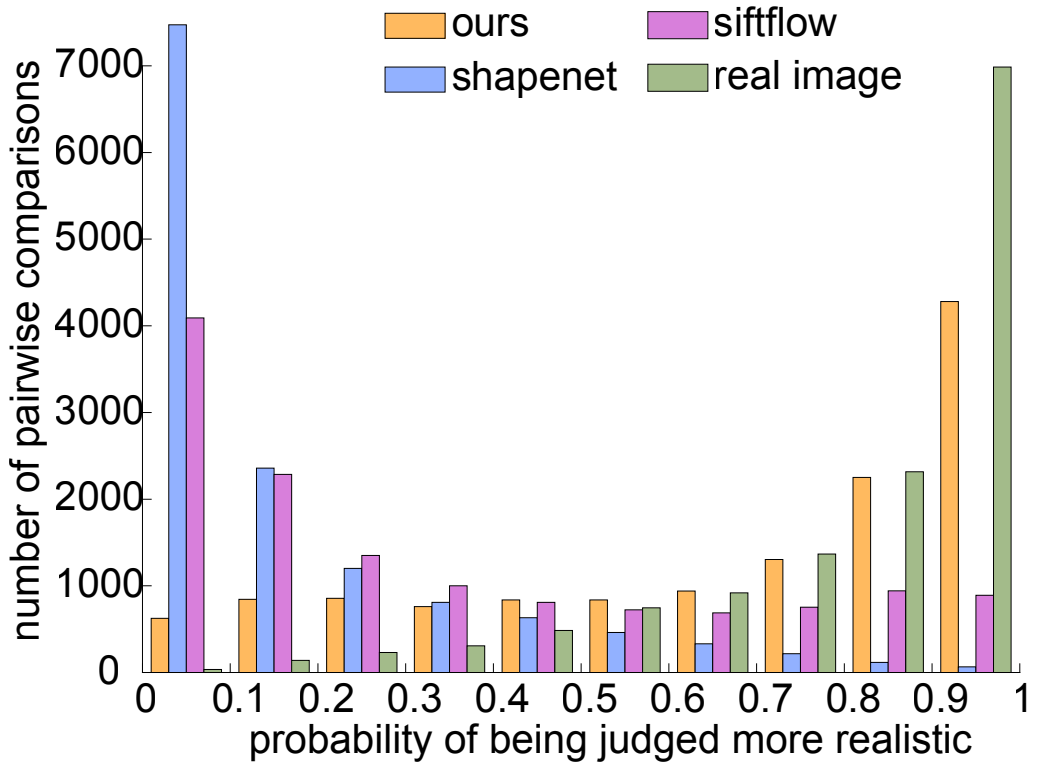


Figure 4.17: Probability of rating each image *winning* against each other image as computed based on the Bradley-Terry model [134]. Note that our results are consistently rated as realistic by the users.

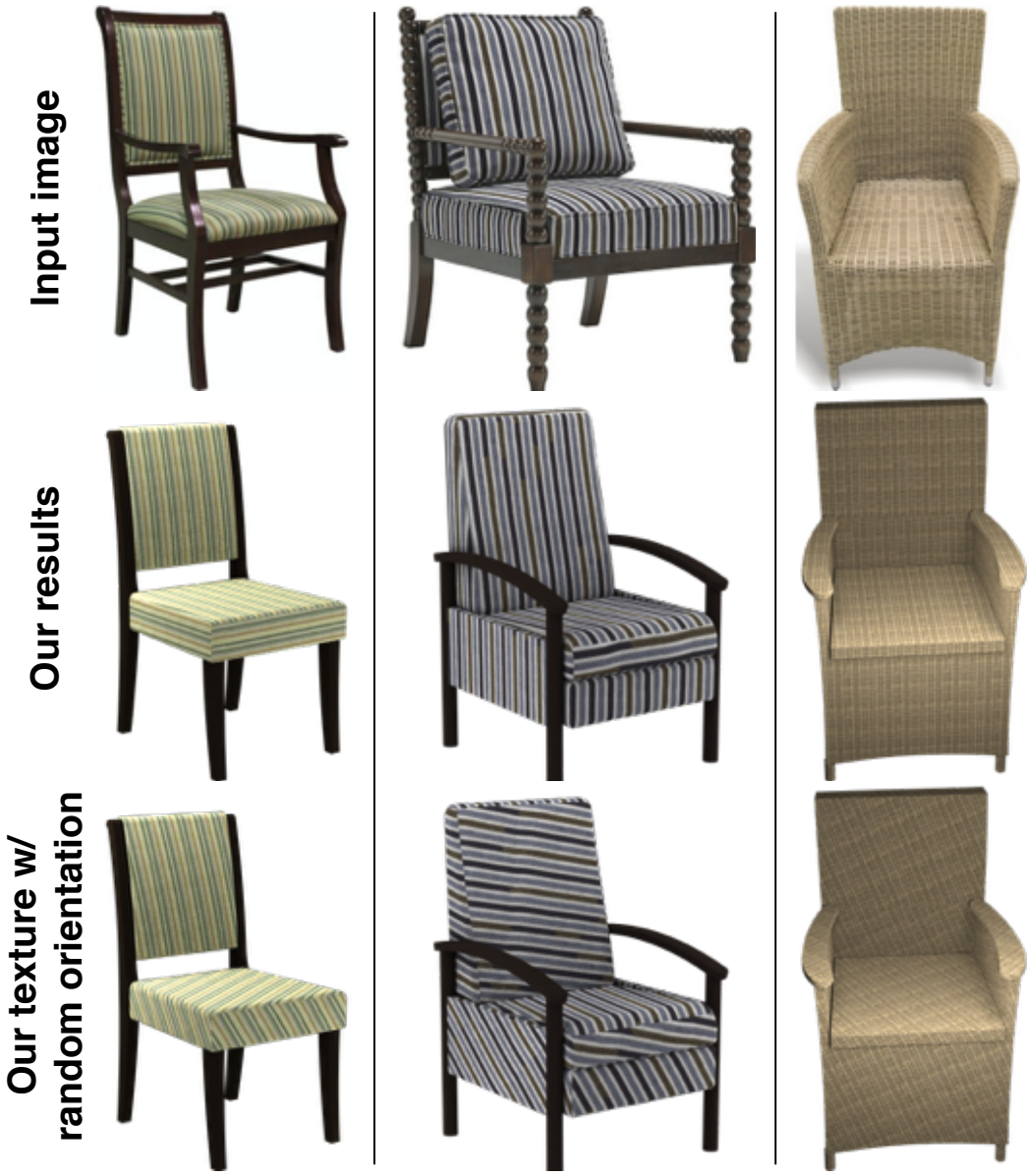


Figure 4.18: Comparison against random texture orientations: (top) input images, (middle) our appearance transfer results, (bottom) texture mapping with randomly oriented textures (base texture extracted using our method).

4.5.3 Effect of texture orientation

In Figure 4.18, we compare our results against a naive approach of assigning random orientation to the texture patches. In both case, we use the base textures extracted by our method. Random orientations easily break realism as in real-world texture patterns (e.g., fabric, wood grains) are carefully laid out with respect to the part features of the objects.

4.5.4 Effect of patch alignment

In Figure 4.20, we show the effect of our patch alignment step on a representative chair example with irregular texture. We formulate our patch alignment step based on the assumption of repetitive texture pattern but not necessary to have regular texton structure.

4.5.5 Evaluation of illumination estimation

Comparison with intrinsic decomposition methods. The goal of our illumination estimation step is essentially different from classical intrinsic decomposition. During illumination estimation, we make use of geometric information in the form of the retrieved model, which strongly regularizes the solution. Remove shading effect from the image, in turn results in the selection of large flat regions from which trustable regions are extracted.

We compare our illumination estimation result with state-of-the-art intrinsic decomposition algorithms (see Figure 4.19): IIW [129] and SIRFS [28]. Because both approaches use smoothness and parsimony priors, fine texture details can easily be removed during shading. For illumination estimation, the difference between our

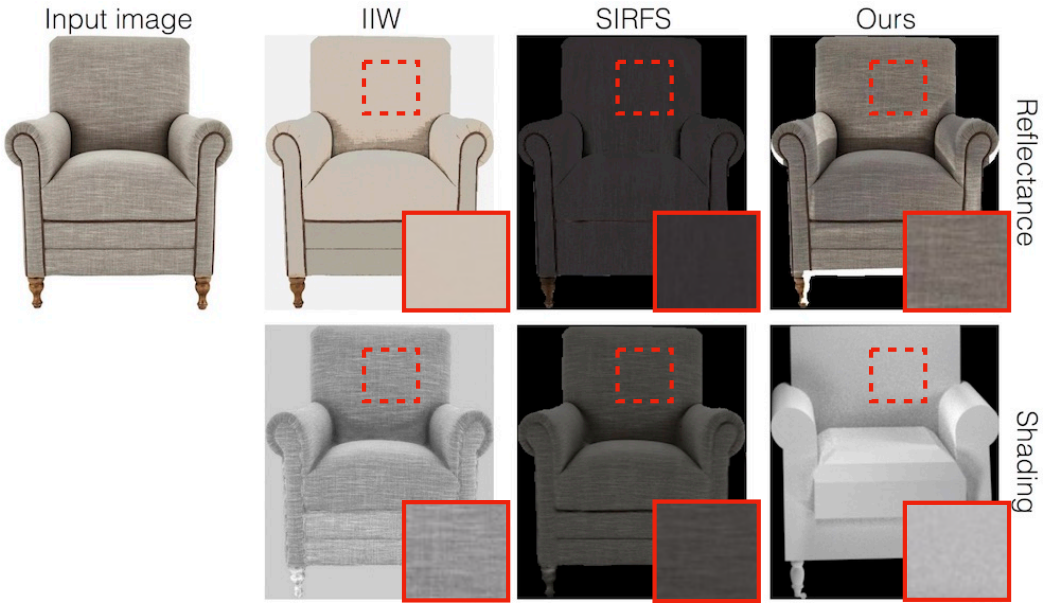


Figure 4.19: (a) Input image, (b) IIW [129] result, (c) SIRFS [28] result, (d) our result. For each result, the upper row shows reflectance while the lower row shows shading. The boxes show the zoom-in of the same region from reflectance and shading layer of different decomposition methods.

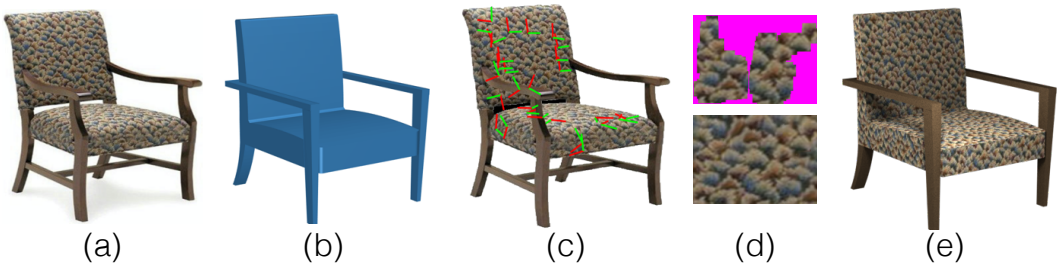


Figure 4.20: Irregular texture transfer. (a) input image, (b) retrieved shape, (c) frame orientation, (d) trustable region and hole filling result, (e) appearance transfer result.

method and SIRFS is twofold: (i) most 3D object cannot be formulated as a continuous depth map, therefore SIRFS cannot provide a reliable geometry estimation, which leads to an approximate illumination even when the combination of reflectance and shading agree with the input image; and (ii) for optimization efficiency, SIRFS has to rely on a simple rendering engine that only consider normal of geometry. In contrast, we can handle effects due to self-shadowing, ambient occlusion, etc. Since our shading samples are rendered offline (for basis computation), we can easily handle complex shading effects.

We also tested the robustness of our illumination estimation approach by changing the number of shading samples and the number of correspondences used. Figure 4.21 shows our result with 36 correspondences from image down-sampled to 18 and 9 correspondences. The number of shading samples is reduced from 50 to 25.

We use the extracted illumination to remove shading from input image to improve the brightness consistency between different parts. Therefore, after illumination correction, we obtain larger trustable regions, which in turn results in improved transfer as shown in Figure 4.22.

4.5.6 Effect of proxy shape

In Figure 4.23 we show the robustness of image-to-shape texture transfer under different choices of proxy shapes. Since we only rely on rough geometry provided by the retrieved shape, our transfer result is robust to the choice of the closest shape the initial 2D-3D correspondence.

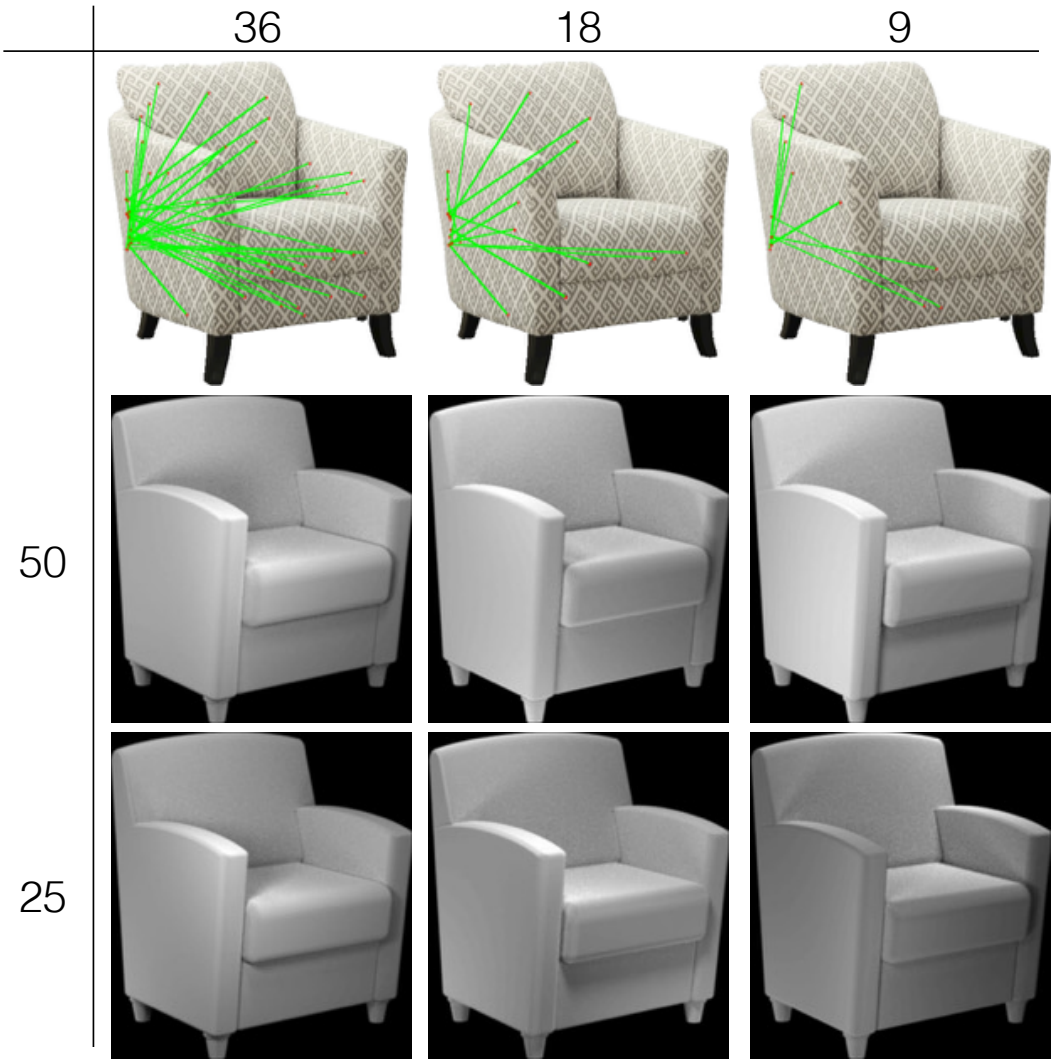


Figure 4.21: Effect of number of shading samples. (Top) Correspondences used, (middle) estimated shading with 50 shading samples, (bottom) estimated shading with 25 shading samples.

4.5.7 Comparison with TILT

We show the advantage of using geometry proxy during prespective correcting by comparing with state-of-the-art texture rectification methods. We used TILT [1], which makes use of low rank prior, to rectify a user-cropped texture patch. As shown in Figure 4.24, low rank assumption is not sufficient to remove ambiguity, while our geometry-based rectification leads to a more intuitive result.

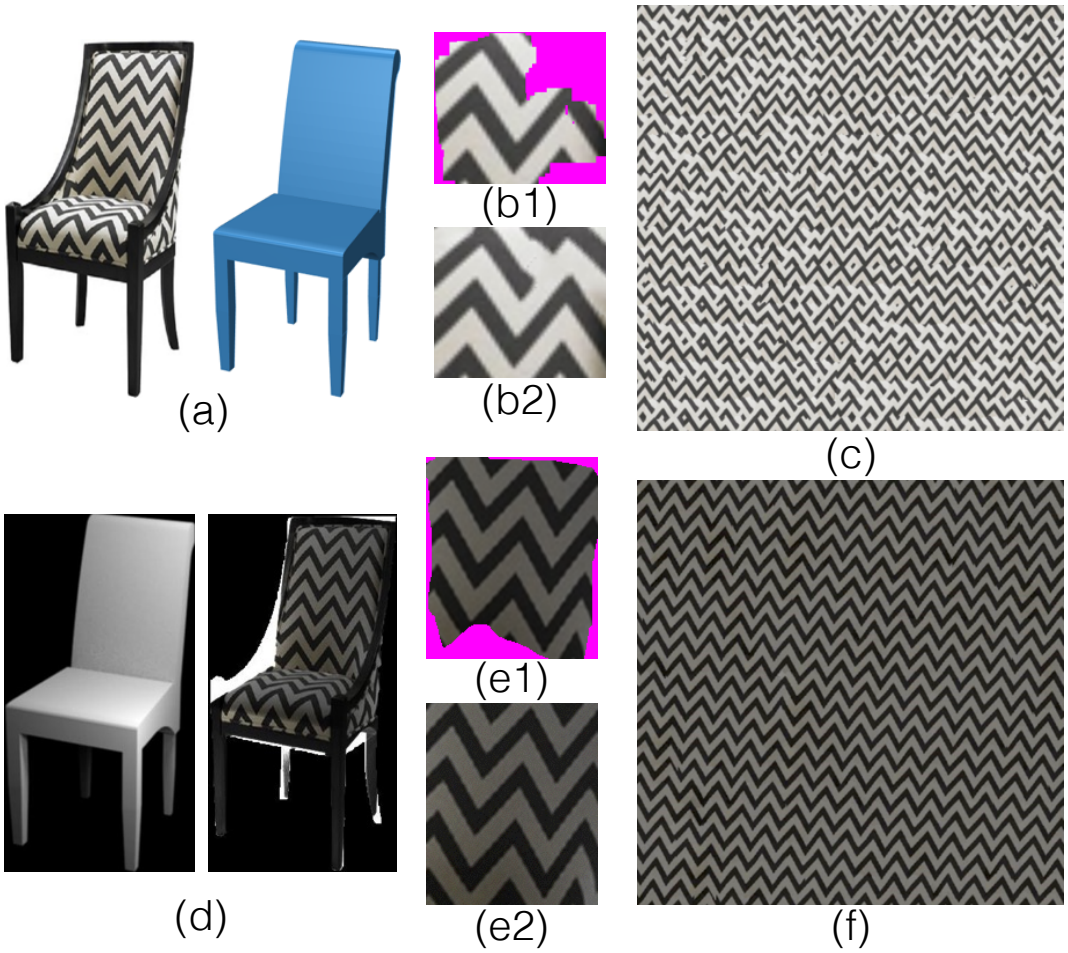


Figure 4.22: Effect of illumination correction. (a) input image and retrieved shape, (b1) extracted trustworthy region with patch alignment on patches cut from input image, (b2) hole filling results, (c) texture synthesis results. (d)estimated shading and illumination correction, (e1,e2) extracted trustworthy region and hole filling result, (f) final texture synthesis result.

4.6 Application

4.6.1 Image editing

We use the estimated illumination to realistically insert novel objects into the input image. Again, geometry from the retrieved shape S helps to estimate shadowing effects. We use our method to estimate view angle for S, texture patterns, and illumination. We render a shading image $I_S{}^*$ with the to be inserted 3D object but set the object to be invisible. Let I_S be the shading image of retrieved object. We then obtain the shading effect of the additional object as $C = I_S{}^*/I_S$. Next, we apply the shading effect C on input image as $I' = I * C$. We render H in the scene with S set to

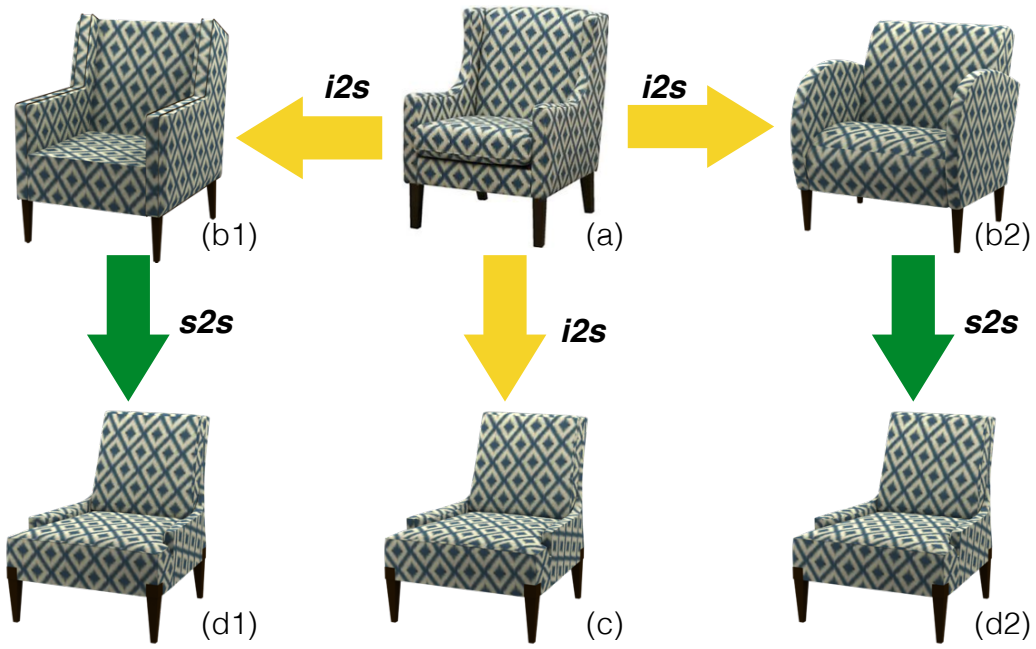


Figure 4.23: (a) Input image, (b1,b2) image-to-shape transfer texture for two closest shapes, (c) directly transferring texture to another shape, (d1,d2) indirectly transferring texture via the closest shape using shape-to-shape transfer.

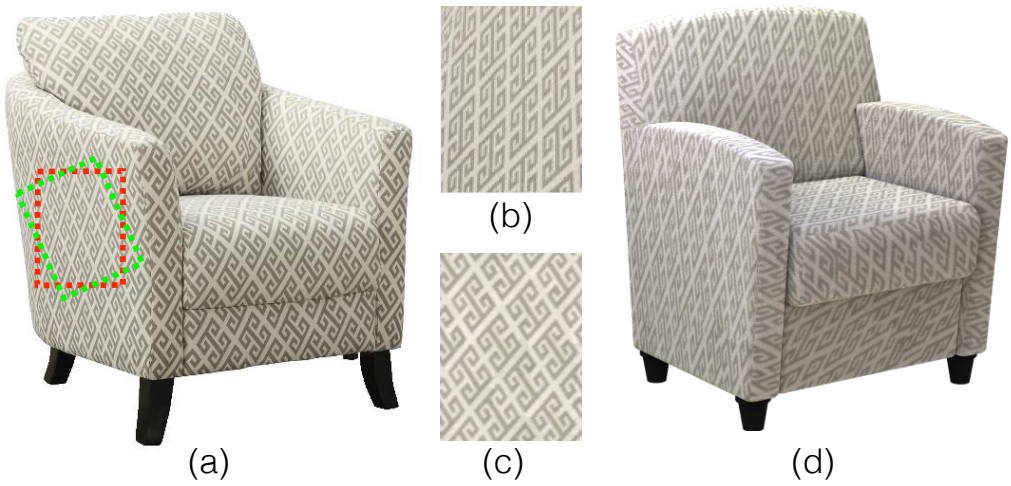


Figure 4.24: Comparison with TILT [1]: (a) a red rectangle is selected by the user on the input image and rotated to the green one using TILT for texture rectification; (b) shows the texture patch in green rectangle; for comparison, (c) is the texture patch extracted using our approach; (d) applys result of TILT texture rectification, rendered with our orientation and illumination.

be invisible. Hence, S only contributes shading effect on H due to ambient effect and shadow. Finally, we copy H to I'. Figure 4.25 shows a few examples. Note that unlike state-of-the-art object insertion methods [38, 135], the above workflow only requires the user to specify the position of the inserted object on S, while the rest of

the steps are automatic.

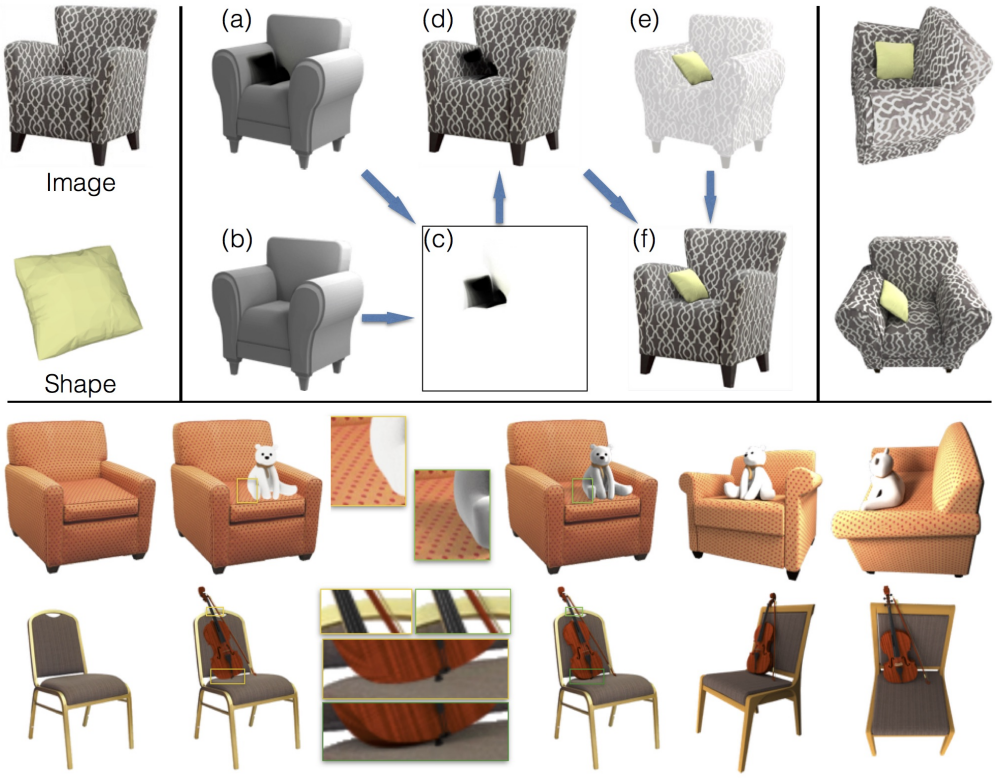


Figure 4.25: Image editing.

4.6.2 Novel view synthesis

Our texture synthesis pipeline enables us to infer the appearance of invisible parts of an object. In Figure 4.15, we present our texture transfer results rendered from novel views different from the estimated viewpoint of the image.

4.6.3 Boosting 3D model repositories

Since our system is scalable, it allows fully automated texture transfer from real product images to models in large-scale collections such as ShapeNet. Such model repositories are important for training machine-learning algorithms. Su et al. [31] uses millions of rendered images to train deep neural networks for computer vision tasks and obtains state-of-the-art results on *real-world* test data. We demonstrate that such 3D shape repositories enriched by our method can provide valuable boost to learning tasks, compared with the original textures. We conduct experiments on two tasks: single image depth estimation and texture-guided image retrieval.

(i) Single-image depth estimation. Obtaining large-amount of training data with groundtruth is not easy for training depth estimators. We train a deep neural network using our textured synthetic data. Our network takes a single image with foreground mask as input and predicts the relative depth of every pixel. This depth is in a canonical frame with fixed range, i.e., we make the assumption that objects have diameter 1 and the camera is placed at a fixed distance 3 to the center of the object. Similar to [136], we use a network with an encoding stage (a stack of convolutional layers) and a decoding stage (a stack of deconvolutional layers). For more details of the network please refer to the supplementary material.

We use synthetic data rendered from 3D models to train the network, since groundtruth depth information are obtained for free in the rendering process. Specifically, our training set is rendered from 51 3D chair models sampled from the ShapeNet, with 3 different settings of textures. As baselines, in the first setting, we render 3D shapes with no textures, and in the second setting we render them with original textures from artists. As for our own textured rendering, we transfer textures from 51 images to each 3D model. This produces 51×51 shape-texture combinations. For each texture setting, we render $80K$ images. The rendering parameters are set according to [31].

We train 3 different neural networks for the 3 texture settings from scratch and compare their performance in Figure 4.26. We evaluate on a test dataset containing 6000 images with depth information, rendered from 100 ShapeNet chair models at random viewpoints. The evaluation protocol is pixel-wise mean square depth error. We observe that the system trained with our textures is significantly better than the one trained without textures. Surprisingly, it is even better than the original textures by human artists. We hypothesize that this is because our texture transfer system allows more than one set of textures for each 3D model, resulting in improved diversity of training data, a desired property to prevent overfitting in estimating the deep learning model parameters.

(ii) Texture-guided image retrieval. We also train an image retrieval system focusing on texture similarity but agnostic to other nuisances. We choose to train a siamese

neural network for this task. A siamese network takes a pair of images as input and embeds them in a common space such that the distance in this embedding space reflects their dissimilarity for some desired property, such as texture or geometry. More details of our siamese network can be found in supplementary material. To train this network, we need both positive pairs that are objects with similar textures as well as negative pairs that are objects with dissimilar textures. Obtaining such pairs from real-world images is not easy. Again, we generate synthetic training data by rendering textured 3D models. Similar to the depth estimation experiment, we transfer textures from each of 51 2D images to each of 51 3D chair models. This allows us to sample pairs of rendered images and use them as positive pair if their textures are transferred from the same image. In total we sample 8 million such pairs, with 10% being positive. We compare with three baselines. For the first baseline, we train the same siamese network using rendered 3D models with textures from human artist (ShapeNet textures). Since each texture only presents on a single 3D models in this setting, a pair is positive only if the two images are rendered from the same 3D model. For the second baseline, we train a network by renderings without textures. Lastly, we also include distance from HoG image features as a baseline.

We evaluate all the methods by an image retrieval experiment using the Euclidean distance in the embedding space (or feature space for HoG features). Since real-world benchmark test dataset is hard to obtain, we create a synthetic test set similar to how we generate the training set. Specifically, we transfer textures from

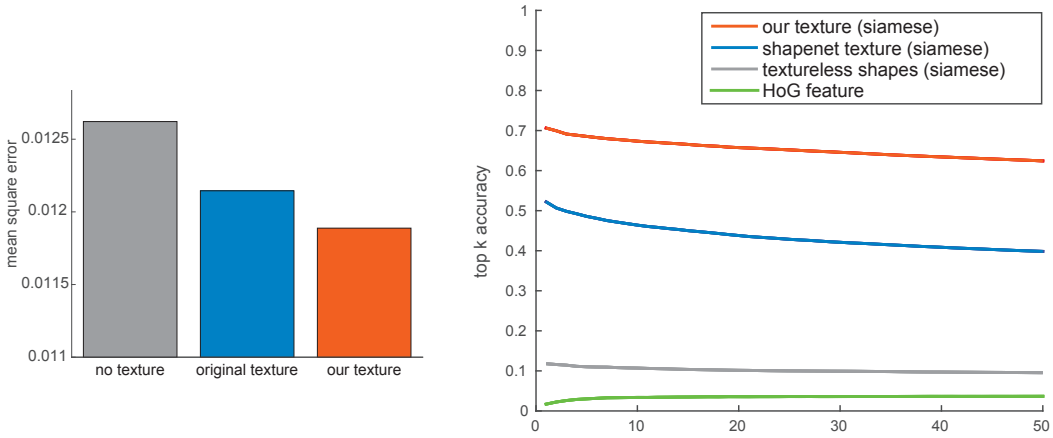


Figure 4.26: CNN-based single image depth estimation (chairs). Texture-guided image retrieval (chairs).

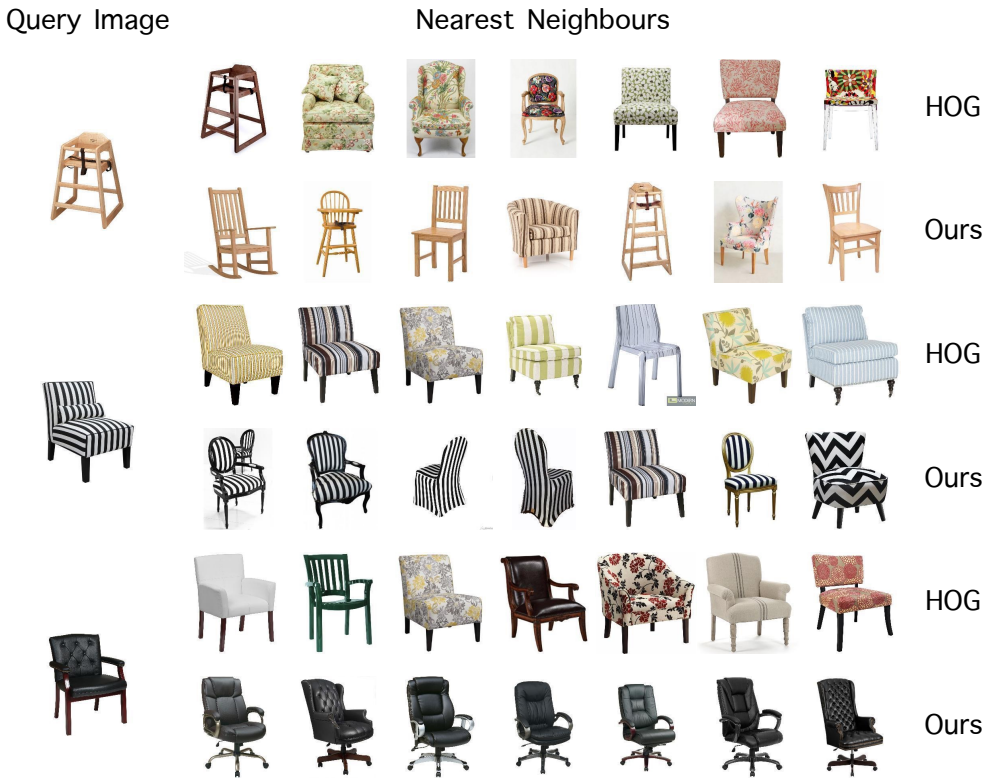


Figure 4.27: Examples of texture-guided image retrieval.

30 images to 30 shapes. The evaluation protocol is the top K retrieval accuracy for each image in the test set, i.e., how many images from the top K retrieval results have the same texture as the query. Figure 4.26 shows that our method comfortably outperforms all the baselines.

This system generalizes well to real images, though trained on synthetic images. We downloaded 757 real chair images using the Bing search engine and conducted a simple texture-guided retrieval experiment. We qualitatively compare the retrieval results using the embedding of siamese network from our textured data and HoG features in Figure 4.27. Results show that the system trained by our textured data is able to focus on texture variation while being agnostic to changes such as viewpoint, geometry, and lighting condition. More results can be found in supplementary material.

4.7 Limitations

The main limitation of our approach is that we assume the object to have homogeneous part-level textures. While this assumption allows us to reliably extract base texture from raw input images, it limits handling images where this assumption is violated (e.g., hand-painted irregular patterns on a wooden chair). Note that with limited information included in the generated patches, texture synthesis can produce unwanted artifacts in the synthesized textures by repeating partial structures.

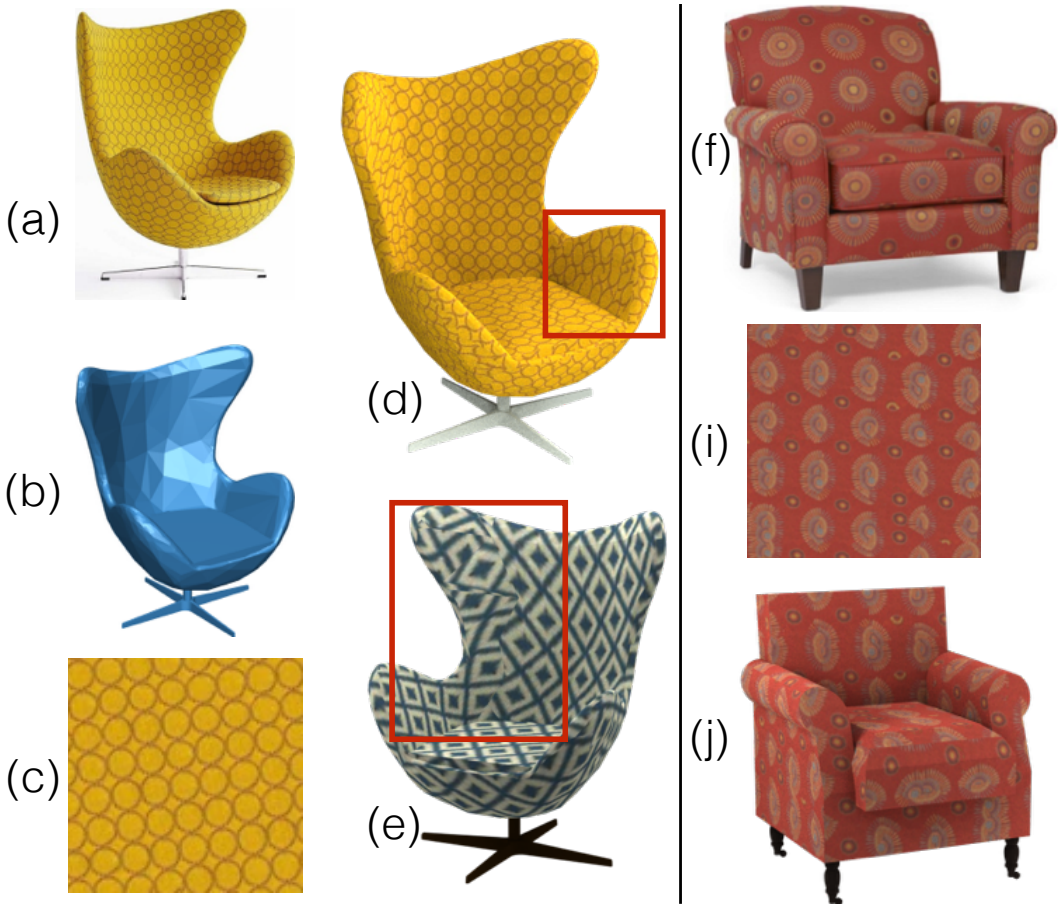


Figure 4.28: Limitations: (i) with sphere-like geometry as in (a,b), even when we synthesize texture (c) from a limited flat region, we may still get large texture distortions (d,e, cropped region); (ii) if the texture element is large compared to the size of the containing object part (f), our patch-based approach may fail to capture enough texture information, leading to problematic texture synthesis and visual artifacts.

A second limitation is that we need to unwrap textures in images according to the estimated shape normals. This approach works quite well when the surfaces are relatively flat. Although our algorithm is designed for piecewise planar shapes, it is

applicable to nonplanar shapes (i.e., Fig 4.14) as long as there are locally flat regions presenting sufficient texture information. However, for highly folded surfaces such as folds and pleats on dresses and curtains, our current method will fail. One interesting future direction will be to perform texture analysis and synthesis directly on the proxy geometries to relax this restriction.

Chapter 5

Joint Material and Illumination Estimation from Photo Sets in the Wild

Faithful manipulation of shape, material, and illumination in 2D Internet images would greatly benefit from a reliable factorization of appearance into material (i.e. diffuse and specular) and illumination (i.e. environment maps). On the one hand, current methods that produce very high fidelity results, typically require controlled settings, expensive devices, or significant manual effort. To the other hand, methods that are automatic and work on 'in the wild' Internet images, often extract only low-frequency lighting or diffuse materials. In this work, we propose to make use of a set of photographs in order to jointly estimate the non-diffuse materials and sharp lighting in an uncontrolled setting. Although spatially varying bidirectional reflectance distribution function (SVBRDF) [50] is proven to be the state-of-the-art model for high realistic surface capturing and rendering, here we assume non-spatially varying model to reduce the complexity since our observations are sparse at instance level. Our key observation is that seeing multiple instances of the same material under different illumination (i.e., environment), and different materials under the same illumination provide valuable constraints that can be exploited to yield a high-quality solution (i.e., specular materials and environment illumination) for all the observed materials and environments. Similar constraints also arise when

Figure 5.1: Joint Estimation: Input and output
We factor a set of images *(left)* showing objects with different materials (red, yellow, black, white plastic, rendered with Uffizi envmap [137]) under different illumination into per-image illumination and per-object material *(top right)* that allows for novel-*x* applications such as changing view, illumination, material, or mixed illumination/material (red chair in the left-bottom imaged environment) *(bottom right)*.

observing multiple materials in a single environment, or a single material across multiple environments. Technically, we enable this by a novel scalable formulation using parametric mixture models that allows for simultaneous estimation of all materials and illumination directly from a set of (uncontrolled) Internet images. The core of this approach is an optimization procedure that uses two neural networks that are trained on synthetic images to predict good gradients in parametric space given observation of reflected light. We evaluate our method on a range of synthetic and real examples to generate high-quality estimates, qualitatively compare our results against state-of-the-art alternatives via a user study, and demonstrate photo-consistent image manipulation that is otherwise very challenging to achieve.

Figure 5.2: Comparison to alternatives (projective texturing, average RGB of intrinsic images [138]). We see that only a proper separation into specular materials and natural illumination can predict appearance in novel views. Other approaches miss the highlight, even in the original view (average of intrinsic), or does not move under view changes (projective texturing). Please refer to the accompanying video to judge the importance of moving highlights under view changes.

5.1 Introduction

Estimating realistic material (i.e., reflectance) and illumination along with object geometry remains a holy grail of shape analysis. While significant advances have been made in the recent years in predicting object geometry and pose from 'in the wild' Internet images, estimation of plausible material and illumination has remained elusive in uncontrolled settings and at a large scale.

Successful material and illumination estimation, however, will enable unprecedented quality of AR and VR applications like allowing realistic 'transfer' of objects across multiple photographs, or inserting high-quality replicas of virtual objects into Internet images. For example, in Figure 5.1, imagine transferring the red chair from one image to another. Currently, this task is challenging as we neither have access to the (red) chair's material, nor the illumination in the target scene.

The naive solution of simply copying and pasting a 2D cutout is unsatisfactory as it easily leads to low fidelity results (*e.g.,* unrealistic highlights), and more importantly, does not allow for pose adjustments (see Figure 5.2) or relighting.

In this paper, we investigate the problem of material and illumination estimation *directly* from 'in the wild' Internet images. The key challenge is that material and illumination are never observed independently, but only as the result of the convolving reflection operation with (estimated) normal direction and view direction (assuming access to rough geometry and pose estimates). Thus, in absence of further assumptions, we *cannot* uniquely recover material or illumination from single observations (*i.e.,* images). Instead we reply on *linked* observations. We observe that often Internet images record the same objects in different environments (*i.e.,* illuminations), or multiple objects in the same environments. Such *linked* observations among all the materials and illuminations forms a (sparse) *observation matrix* providing critical constraints among the observed materials and illumination parameters (for example Figure 5.3). We demonstrate that such a special structure can be utilized to robustly and accurately estimate all the material and illumination parameters through a global optimization.

We choose a formulation based on the basic rendering equation in combination

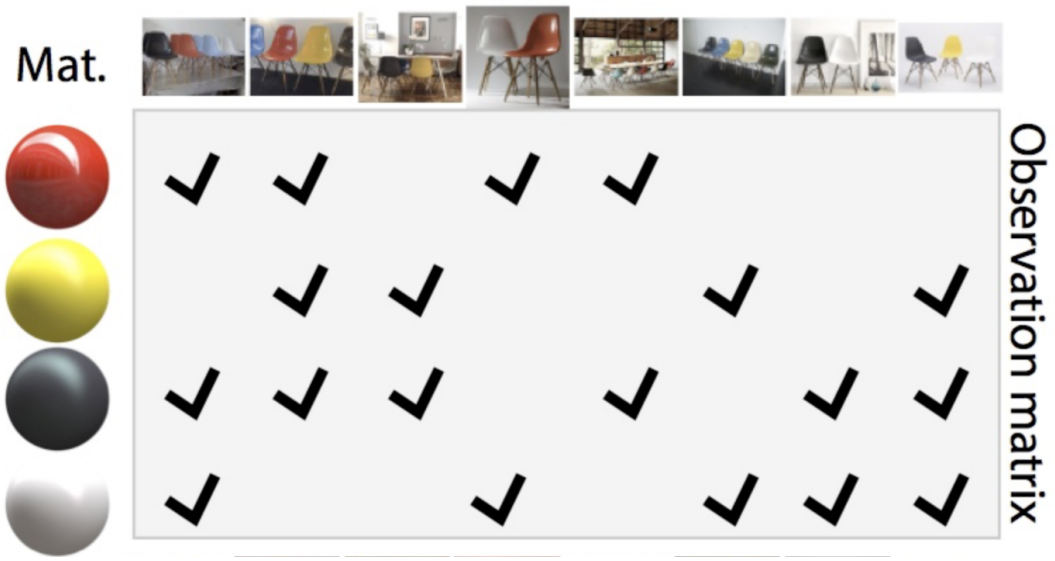


Figure 5.3: An example of the observation matrix from a set of linked photographs.

with available per-pixel geometry estimation. However, there are multiple challenges: (i) access to only approximate proxy geometry for the scene objects with rough pose estimates leads to inaccurate normal estimates; (ii) partial observations due to view bias (*e.g.,* chair backs are photographed less often) and sparsely observed normal directions (*e.g.,* flat regions in man-made objects); (iii) working with the rendering equation when updating material and illumination parameters in an inverse problem setup is inefficient in a standard physically-based rendering pipeline; and finally, (iv) access to limited data due to sparsely observed joint material-illumination pairs.

In order to overcome the above challenges, we propose a novel formulation using parametric mixture models. We propose to approximate the reflection operator and its derivative with respect to material and illumination in terms of Isotropic Spherical Gaussians (see [139]) that can be efficiently utilized to jointly optimize for the materials and illumination at a large scale (*i.e.,* involving multiple materials and illuminations). The main advantage of using Isotropic Spherical Gaussian representation is its capacity to capture high-frequency signal which is common in HDR illumination such as light source (light bulb or sun), while other linear basis based representations, i.e. Spherical Harmonic basis [28] cannot handle it well. This optimization is driven by two neural networks that were trained on a large set of materials and illuminations to predict the gradient the optimization will follow. For

example, in Figure 5.1, we observe 4 different colored (*i.e.,* material) chairs under 8 different illuminations (only 6 images shown in the teaser figure) with linked observations. Only using these limited observations, our algorithm extracts high-quality material and illumination estimates, which can then be used for non-trivial image manipulation.

We extensively evaluate our method on both synthetic and real data, both quantitatively and qualitatively (using a user study). We demonstrate that increasing the amount of linked material-illumination observations improves the quality of both the material and illumination estimates. This, in turn, enables novel image manipulations previously considered to be very challenging. In summary, our main contributions are: (i) proposing the problem of coupled material and illumination estimation from a set of Internet images; (ii) formulating an efficient and scalable algorithm that allows high-quality material and illumination estimation from a set of images; (iii) using a neural network to approximate the complicated gradient of reflected light with respect to material and illumination parameters; and (iv) utilizing the estimations to enable realistic photo-realistic image manipulations.

5.2 Overview

Starting from a set of linked photographs (*i.e.,* multiple objects observed in different shared environments), our goal is to retrieve object geometry with pose predictions and estimate per-object materials and per-environment illuminations. The estimated information can then be used to faithfully re-synthesize original appearance and more importantly, obtain plausible view-dependent appearance. Figure 5.2 shows baseline comparisons to alternative approaches to assign materials to photographed objects. We observe that even if the geometry and light is known (we give all the approaches access to our estimated environment maps, if required), the highlights would either be missing (using intrinsic image [138] for estimating average albedo), or not move faithfully (*e.g.,* with projective texturing) under view changes.

As input, we require a set of photographs of shared objects with their respective masks (see Figure 5.4). In particular, we assume the materials segmentation to be

consistent across images. As output, our algorithm produces a parametric mixture model (PMM) representation of illumination (that can be converted into a common environment map image) for each photograph and the reflectance parameters for every segmented material. We proceed in three steps.

First, we estimate object geometry and pose, and convert all the input images into an unstructured reflectance map for each occurrence of one material in one illumination in Section 5.3.1. Since we work with very few images collected from the wild, our challenge is that this information is very sparse, incomplete, and often contradict each other.

Second, we solve for illumination for each image and reflectance model parameters for each material in Section 5.3.4. This requires combining a very large number of degrees of freedom, as fine directional lighting details as well as accurate material parameters to be estimated. The challenge is that a direct optimization can easily involve many variables non-linearly coupled and lead to a cost function that is highly expensive even to evaluate as it involves solving the forward rendering equation, *e.g.,* [63]. For example, representing images and illumination in the pixel basis leads to an order of 10^4-10^5 variables (*e.g.,* $128 \times 256 \times$number-of-environment-maps). At the same time, evaluating the cost function for every observation pixel would amount to gathering illumination by iterating all pixels in the environment map, *i.e.,* an inner loop over all 128×256 environment map pixels inside an outer loop across all the $640 \times 480 \times$number-of-images-in-the-collection observations. This quickly becomes computationally intractable.

Instead, we introduce a solution based on parametric mixture-model (PMM) representation of illumination to *inverse* rendering, which has been successfully applied to forward rendering [139–143]. Our core contribution is to take PMM a step further by introducing the parametric mixture reflection operator and an approximation of its gradient, allowing to solve the optimization in a scalable fashion involving many materials and environments. The gradient approximation uses a neural network to map from observed reflected light, light and material parameters to changes of light and material parameters. It is trained on a set of synthetic images

rendered from many illuminations and many materials.

Third, the estimated material and illumination information can directly be used in standard renderers. The challenge in such applications is to capture view-dependent effects such as moving highlights. In Section 5.4.3, we show applications to manipulating images, changing the illumination and/or material and/or view, transferring materials to objects in other images, or inserting objects into new illumination (see also supplementary materials).

5.3 Algorithm

We now explain our approach in details.

5.3.1 Acquiring Geometry and Reflectance Maps

We start from a set of images with the relevant materials segmented consistently across the image collection. Designer websites (*e.g.,* Houzz) and product catalogs (*e.g.,* Ikea) regularly provide such links. Here we assume that the links are explicitly available as input. First, we establish a mapping between illumination material-pairs and observed appearance.

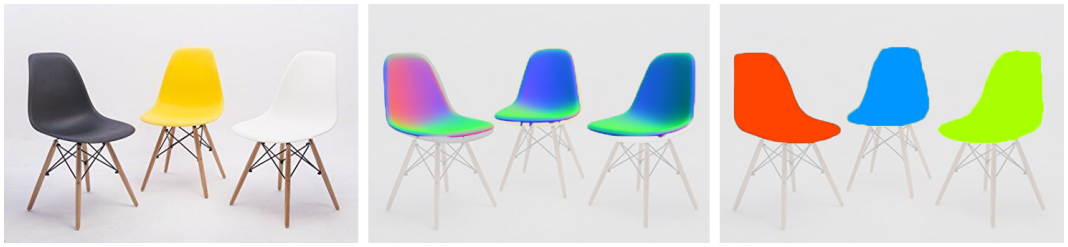


Figure 5.4: RGB, normal, and instance segmentation of a typical input image.

Per-pixel labels. For the input images, we used per-pixel orientation (screen-space normals) (Figure 5.4) obtained using render-for-CNN [144] trained on the ShapeNet to retrieve object geometry and pose estimates. We found this to provide better quality normal predictions than those obtained via per-pixel depth [145] and normal [146] estimation.

Reflectance maps. The rendering equation [75] states that

$$L_{\mathrm{o}}(\mathbf{x},\mathbf{n},\omega_{\mathrm{o}}) = \underbrace{L_{\mathrm{e}}(\mathbf{x},\omega_{\mathrm{o}})}_{\text{Emit}} + \int_{\Omega} \underbrace{f_{\mathrm{r}}(\mathbf{x},\omega_{\mathrm{i}},\omega_{\mathrm{o}})}_{\text{BRDF}} \underbrace{L_{\mathrm{i}}(\mathbf{x},\omega_{\mathrm{i}})}_{\text{Incom.}} \underbrace{<\mathbf{n},\omega_i>^{+}}_{\text{Geometry}} \mathrm{d}\,\omega_{\mathrm{i}}, \qquad (5.1)$$

where $\mathbf{x}$ is the position, $\mathbf{n}$ the surface normal at location $\mathbf{x}$, ω_{o} the observer direction, L_{o} is the observed radiance, L_{e} is light emission, L_{i} is the incoming illumination, and f_{r} the bi-directional reflectance distribution function (BRDF) [147].

We assume a simplified image formation model that allows for using a slightly generalized variant of reflectance maps [45]: (i) distant illumination, (ii) convex objects, *i.e.,* no shadows or inter-reflections, (iii) *spatially invariant* BRDFs, and (iv) no emission. Note that we do *not* assume a distant viewer as typical reflectance map does. This simplifies Eq. 5.1 to

$$L_{\mathrm{o}}(\omega_{\mathrm{o}},\mathbf{n}) = \int_{\Omega} f_{\mathrm{r}}(\omega_{\mathrm{i}},\omega_{\mathrm{o}}) L_{\mathrm{i}}(\omega_{\mathrm{i}}) <\mathbf{n},\omega_i>^{+} \mathrm{d}\,\omega_{\mathrm{i}}. \qquad (5.2)$$

A classic reflectance map is parameterized either by normal $\mathbf{n}$ or by the observer direction ω_{o}. Instead of making such a split, we take a less structured approach tailored to our problem: an *unstructured reflectance map* (URM) denoted by $\mathscr{O}$ that uses a list that holds in each entry a tuple of (i) normal o_{n}, (ii) half-angle vector $\mathbf{h}$, (iii) observed radiance o_L (cf. Figure 5.5), and (iv) indices o_{m} and o_{i} of the material and illumination, respectively. We denote $\mathbf{h}$ as the half-angle vector for front ($-z$) and observer direction, $\mathbf{h} := (<2\mathbf{n},o_\omega>\cdot\mathbf{n} - o_\omega + (0,0,-1))/2$. This parametrization will provide a more convenient way to index information. An example visualization of the URM by projecting the $\mathbf{n}$ as well as the $\mathbf{h}$ coordinate using latitude-longitude is seen in Figure 5.5.

To acquire the URM from an image with given per-pixel position and orientation, we apply inverse gamma correction such that o_L is in physically linear units. Note, that although we do not know the absolute scale inside each photo, we do not need it for most applications. Further, we do not differentiate between objects and consider only their materials (*i.e.,* an object with two material parts are essentially treated as two materials).

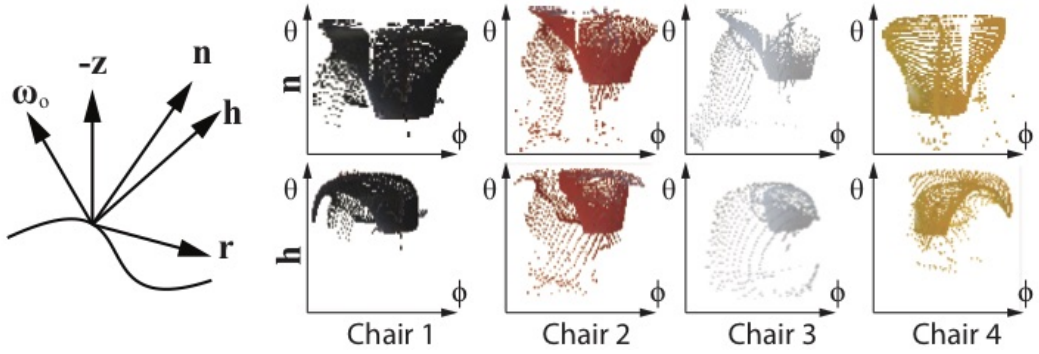


Figure 5.5: Schema and actual Unstructured Reflectance Maps of the chairs in the first column of Figure 5.1. Each point is an observed color for a specific surface orientation **n** and half-angle vector **h**.

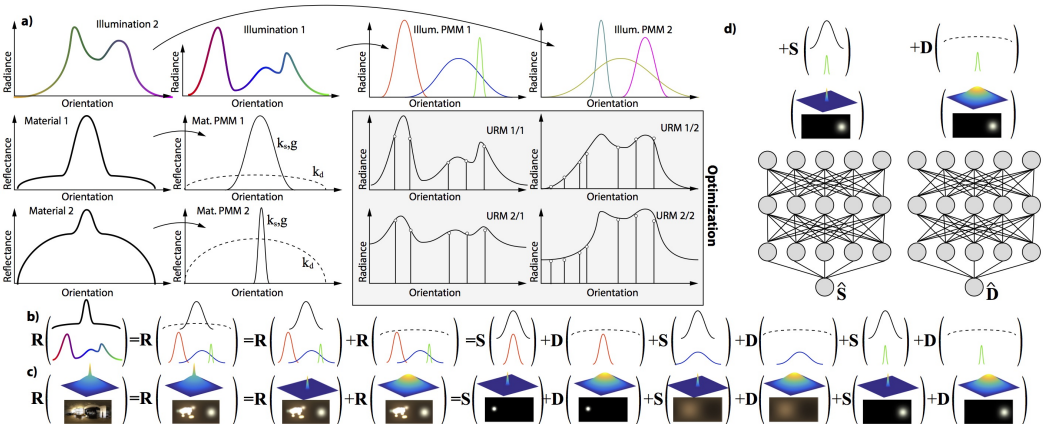


Figure 5.6: The three main ideas to enable large-scale optimization: (a) approximating illumination as parametric mixture models and the BRDF as a sum of a diffuse and a specular component; (b, c) expressing reflection as a sum of diffuse and specular reflections of individual lobes; and (d) approximating derivative of diffuse and specular reflection of ISGs using corresponding neural nets.

5.3.2 Representation

Illumination. We use Parametric Mixture Models (PMMs) to represent illumination. PMMs have been used for pre-computed light transport [140, 141, 148], BTF compression [142], interactive rendering [149], importance sampling [143], or even in caustic design [150]. A PMM encoded as

$$g(\omega|\Theta) := \sum_{l=1}^{n_\mathrm{p}} p(\omega|\Theta_l) \approx L(\omega) \tag{5.3}$$

is a sum of n_p *lobe* functions $p(\omega|\Theta_l)$ that depend on a parameter vector Θ_l to approximate, in our setting, the incoming or outgoing light function $L(\omega)$. All

parameter vectors Θ_l of one PMM are combined in a parameter matrix Θ. In our case, the domain of g is the sphere Ω parameterized using latitude-longitude representation $\omega = (\theta, \phi) \in [0, 2\pi) \times [0, \pi)$.

As mode functions, we employ Isotropic Spherical Gaussians (ISGs) [140, 143, 148]. An ISG lobe has the form

$$p(\omega|\Theta) := w \cdot \exp(-\sigma(\omega - \mathbf{z})^2),$$

where $w \in \mathbb{R}^+$ is the weight of the lobe, σ is its variance and $\mathbf{z}$ the *mean* direction. Consequently, a lobe is described by parameter vector $\Theta = (w, \sigma, \mathbf{z})$. To work with RGB values all weight components $\mathbf{w}$ in this paper are vector-valued, but the variance parameter σ is scalar. For each image, we use an ISG PMMs with $n_{\mathbf{p}} = 32$ components to represent unknown illuminations.

Material. We assume the material to be of the form

$$f_{\mathrm{r}}(\omega_{\mathrm{i}}, \omega_{\mathrm{o}}|\rho) = \underbrace{k_{\mathrm{d}} f_{\mathrm{d}}(\omega_{\mathrm{i}}, \omega_{\mathrm{o}})}_{\text{Diffuse}} + \underbrace{k_{\mathrm{s}} f_{\mathrm{s}}(\omega_{\mathrm{i}}, \omega_{\mathrm{o}}|r)}_{\text{Specular}}, \tag{5.4}$$

a parametric model that can be split into the weighted sum of a diffuse and a specular component f_{d} and f_{s} with weights k_{d} and k_{s}, repectively. We choose Lambertian as the diffuse model and GGX [151] that has a single roughness parameter r as the specular model. The material parameters are therefore a tuple $\rho = (k_{\mathrm{d}}, k_{\mathrm{s}}, r) \in \mathbb{R}^7$ of RGB diffuse and specular reflectance and a scalar roughness parameter. We denote the BRDF parameter vector of material j as $\rho^{(j)}$. Note that we do not need to represent f_{r} using a PMM, which would introduce unnecessary approximation error.

5.3.3 Reflection

Using standard notation [152] for light transport, we express reflection as an operator $\mathbf{R}$, mapping the function of incoming light L_{i} to a function of reflected outgoing light

L_o :

$$L_o(\omega_o) = \mathbf{R}(L_i|\rho)(\omega_o) = \int_\Omega \underbrace{L_i(\omega_i)}_{\text{Illumination}} \underbrace{f_r(\omega_i, \omega_o|\rho)}_{\text{BRDF}} \mathrm{d}\omega_i. \tag{5.5}$$

When using an ISG to represent the illumination, we suggest to use a *parametric reflection operator* $\mathbf{R}(\Theta|\rho)$ that maps from a single illumination ISG lobe Θ and a material ρ to a reflected light. As we assume the BRDF to be a sum of a diffuse and a specular part, we can similarly define $\mathbf{D}$ and $\mathbf{S}$ that are respectively the diffuse and the specular-only reflection and $\mathbf{R} = \mathbf{D} + \mathbf{S}$. So, finally we have

$$L_o(\omega_o) = \sum_{l=1}^{n_l} \mathbf{D}(\Theta_l|\rho) + \mathbf{S}(\Theta_l|\rho). \tag{5.6}$$

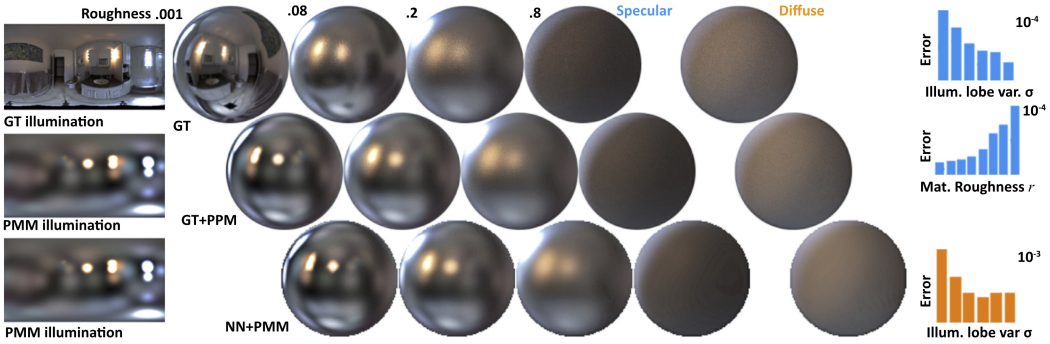


Figure 5.7: Evaluation of the neural network. The first row shows GT renderings with a GT envmap. The second row shows again GT rendering, but using the GMM fit to the envmap. This is an upper bound on the NN quality, as it works on the GMM representation. The third row shows the NN result. In the horizontal direction, specular results of increasing roughness are followed by the diffuse result in the rightmost column. The plots on the right below show the error distribution as a function of different parameters.

5.3.4 Formulation

Our task is to find a set of illuminations and a set of materials that explain the acquired observations (see the previous section). Next, we describe how to represent reflectance and illumination as well as introduce the parametric reflectance operator, its derivative with respect to material and illumination, and an approximation method for efficient joint optimization for material and illumination given the observations (see Figure 5.6).

Cost function. Our main objective function quantifies how well a set of materials and illuminations explain the input observation. It should be fast to evaluate and allow for an effective computation of its gradient with respect to illuminations and materials in order to be useful in an optimization. We formulate the objective as:

$$c(\Theta,\rho|\mathscr{O}) := \underbrace{\sum_{\mathbf{o}\in\mathscr{O}} \left\| o_{\mathrm{L}} - \sum_{l=1}^{n_{\mathrm{p}}} \mathbf{R}\left(\Theta_l^{(o_{\mathrm{i}})}|\rho^{(o_{\mathrm{m}})}\right)(o_\omega)\right\|^2}_{\text{Data}} + \underbrace{\lambda p(\Theta)}_{\text{Prior}}. \tag{5.7}$$

The gradient of this function with respect to the illumination and material comprises of evaluating $\mathbf{R}$, which involves convolving an illumination lobe with the BRDF. This is both costly to compute and we need to find its derivative. To this end, we will employ a learning-based approach, as described next.

Neural network. The input to this neural network (NN) is the parameters of a single illumination lobe, the material parameters, and the observation direction ω. We call this approximation $\hat{\mathbf{R}}$. The output is an RGB value. We keep the NNs for the diffuse and specular components to be separate and independently process the RGB channels. The corresponding approximations using NNs are denoted as $\hat{\mathbf{D}}$ and $\hat{\mathbf{S}}$, respectively. The network architecture is shown in Figure 5.8. The input to the network is a 12-dimensional vector and differs between diffuse and specular NNs. Both consume the parameters of a single illumination lobe (direction and variance). However, the diffuse net consumes the normal while the specular net consumes the half-angle. All layers are full convolutional with 288 units in each layer. The networks are trained from 200k samples from renderings of spheres produced using Blender. An evaluation of this architecture is found in Figure 5.7.

Prior. As reflectance is typically more chromatic than illumination is, our prior penalizes the variance of the illumination lobe colors *i.e.,* their RGB weights, as in $p(\Theta) = \mathrm{V}(q(\Theta)) = \mathrm{E}(q(\Theta)^2) - \mathrm{E}(q(\Theta))^2$, where $q(\Theta) = \sum_{\Theta\in\Theta}\sum_{i=1}^{n_{\mathrm{p}}} \Theta_{\mathrm{w},i}$. In other words by, first computing the average color of all lobes $q(\Theta)$ and second the variance $\mathrm{V}(q(\Theta))$ of those three channels.

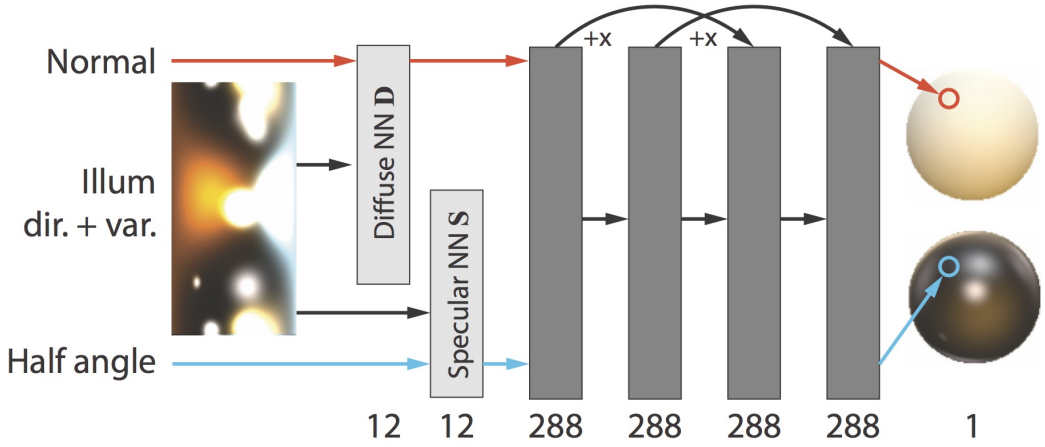


Figure 5.8: Our diffuse (*orange*) and specular (*blue*) neural network architecture, that consumes either normal and a single illumination lobe, or half-angle (*left*) and a lobe to produce a color (*right*).

Optimization. Armed with a fast method (see above) to evaluate the cost function, we employ LBFGS [153] in combination with randomization. As the full vector $\mathscr{O}$ does not fit into memory, we use a randomized subset that fits GPU memory in each iteration and dynamically change the randomization set across iterations. We stop our optimization when each observation on average has been sampled 5 times.

5.3.5 Rendering

For rendering, the result of the optimization is simply converted into an HDR environment map image by evaluating the estimated PMM for each pixel. Such an environment map along with estimated diffuse/specular parameters are then used with standard offline and online rendering applications as shown in our results.

5.4 Results

In this section, we evaluate our approach on synthetic and real image collection data (Section 5.4.1), compare to alternatives (Section 5.4.2) and give examples of potential applications (Section 5.4.3). We use L-BFGS solver for all the experiments. The complexity of our optimization in terms of the number of variables is $(7m+6n_pn)$ and hence is linear in terms of the number of input entries in the material$\times$environment observation matrix. For example, for a five-photo, five-material matrix dataset, it costs about 30 minutes using a NVIDIA Titan X GPU. Pre-training the reflection

Figure 5.9: Results on the INTERNET-LAPD dataset of four images of police cars with two materials. The first row shows the input images. The second row the reflectance maps. The observed ones are marked with black circles. In this example, all are observed. When an RM is not observed, it is re-synthesized. The third row shows our estimated illumination. Recall, that it is defined in camera space. The fourth row contains a re-synthesis using our material and illumination. Please note, that such a re-synthesis is not possible using a trivial factorization as all images have to share a common material that sufficiently explains the images. The last row shows a re-synthesis from a novel view, as well as a rendering of the materials in a new (Uffizi) illumination.

Figure 5.10: Results on the INTERNET-DOCKSTA dataset of six images of IKEA furniture with seven materials in the protocol of Figure 5.9.

Figure 5.11: Results on INTERNET-EAMES dataset of six images of a celebrated Eames chair with four materials in the protocol of Figure 5.9.

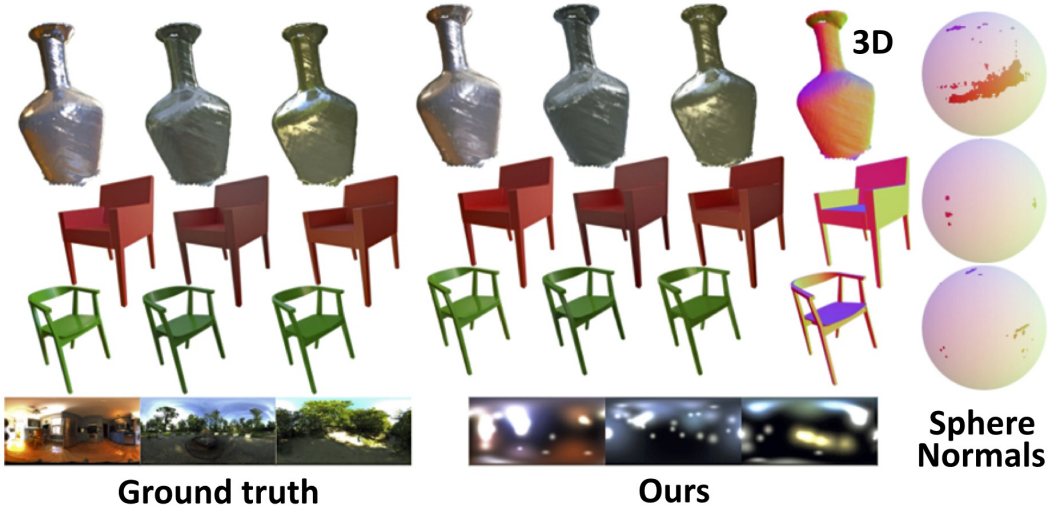


Figure 5.12: Here, we show the effect of Gauss-sphere coverage: Even for non-round objects with flat areas that have a bad coverage of the Gauss sphere the reconstruction (left) is similar to the reference (right).

operator **R**, both diffuse and specular components, takes about three hours on the same specification.

5.4.1 Evaluation

5.4.1.1 Datasets

We evaluated our method using three types of data sets, each holding multiple image collections acquired in different ways. The full resolution images and result images/video are included in the supplementary material.

The first comprises of SYNTHETIC images rendered using Mitsuba [154] and a collection of HDR environment maps. Note that here we have access to ground-truth per-pixel normals and material labels. Here, we have rendered 3 objects in 3 different

scenes with both spheres and real-world shapes that allow synthetic re-combination in an arbitrary fashion. This allows us to evaluate the proposed approach under different input variants, validating its scalability.

The second data set consists of real images collected from the INTERNET. We have manually selected the images (using iconic object name-based Google search) and masked the image content. This dataset has three sets of photographs: the LAPD car (INTERNET-LAPD), the Docksta table (INTERNET-DOCKSTA), and the Eames DSW chair (INTERNET-EAMES). For geometry, we estimated used coarse quality meshes available from ShapeNet. Images are good for qualitative evaluation but do not allow to quantify the error, especially in novel views.

The third dataset contains PHOTOS we have taken from designer objects we choose under illumination conditions (in our labs and offices). We have 3D-scanned these objects (using Kinect) to acquire their (rough) geometry. The photos are taken in five different environments and 7 materials are considered.

5.4.1.2 Qualitative evaluation

Visual quality. We show results of our approach in Figures 5.9, 5.10, 5.11, and 5.18. We evaluate the main objective of our approach, *i.e.,* estimating illumination and reflectance from a photo collection. In each figure, we show the input images, rendering of all objects' materials from original view (with the background from input images) and a novel view as well as visualizations of the material and illumination alone. Input images are shown on the top with the outputs we produce on the bottom (see supplementary for full images). Observed reflectance maps are shown encircled in black. The objects are rendered from an identical novel view, which is more challenging than rendering from the original view. The material is shown by re-rendering it under a new illumination. The exposure between all images is identical, indicating that the reflectance is directly in the right order of magnitude and can transfer to new illuminations. While the illumination alone does not appear natural, shadow and shading from it produce plausible images, even of unknown materials or new objects. Recall that large parts of the objects are not seen in any of the input images and hence large parts of the environment maps are only estimated

from indirect observation. Recall that our method does not use any data-driven prior to regularize the results.

The INTERNET-LAPD in Figure 5.9 shows a single object made from multiple materials. Figure 5.10 shows many chairs in many photos with one common 'linking' object INTERNET-DOCKSTA: the chair with material M_1, that is used to calibrate all the other objects, which are only observed sometimes. Figure 5.11 shows multiple Eames chairs from a set of photos INTERNET-EAMES. All show plausible highlights and colors, albeit only observing a fraction of the combinations. Please see the supplemental video for an animation of the highlights under changing view or object rotations. Figure 5.18 shows photos taken for this work, with all objects in all illumination conditions. Note, that both vases are made of multiple materials. The geometry of all objects in this part (except the chairs) is very approximate and acquired by a depth sensor. Still a good result quality is possible.

Prediction. Using the INTERNET-EAMES data sets, we are able to test the predictive ability of our approach by *leaving out* one image form the collection and compare it to the acquired ground truth. This is seen in Figure 5.13, where starting from the first image, we predict the red chair in the second image and the yellow chair in the third image.

Figure 5.13: Estimating materials and illumination using all chairs except the rendered one. Left: reference image; Middle: the red chair is rendered; Right: the yellow chair is rendered.

Progressive estimation. A key property of our approach is to consolidate information from multiple images to disambiguate material and illumination. This characteristic implies that adding more images to the photo set should reduce the error. Figure 5.14 confirms the rise in performance as more images are added to the linked set.

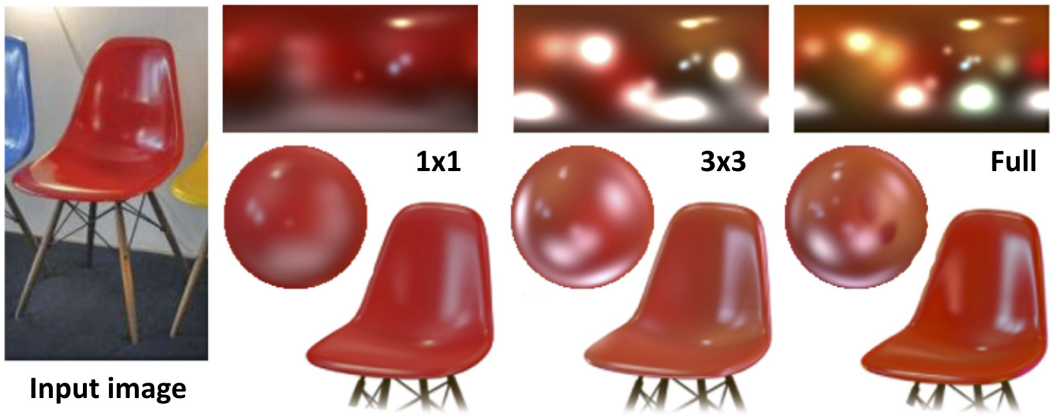


Figure 5.14: Progression of quality from left to right. Every row shows, for a selected material what the additional images can add to the quality in terms of re-rendering, material, and illumination.

5.4.1.3 Quantitative evaluation

We evaluate the effect of certain aspects of our method on the end-result (Figure 5.15). The error is quantified as DSSIM [155] structural image distance (smaller distance indicates better match) between a reference image rendered with known illumination and material compared to another rendering using our estimated material and illumination. Images were gamma-corrected and linearly tone-mapped before comparison.

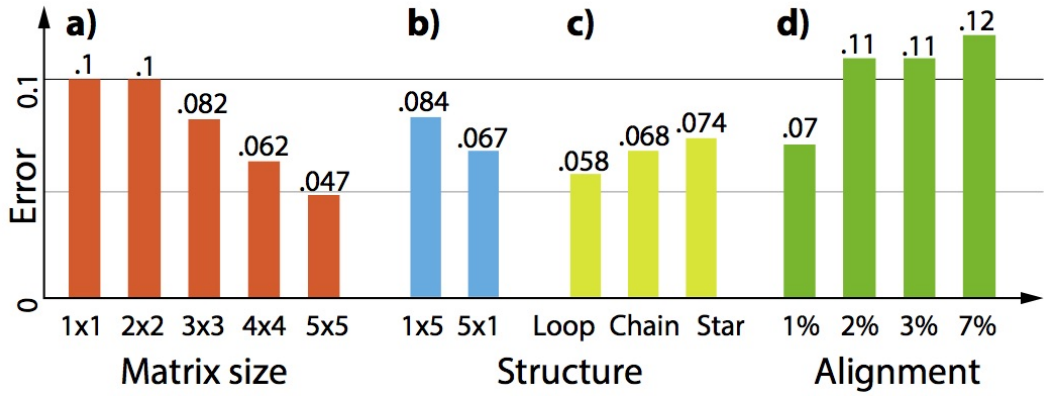


Figure 5.15: Effect of different input properties *(horizontal)* on the quality of our approach in terms of the DSSIM error *(vertical, less is better)*: matrix size, structure, and alignment error.

Matrix size. In Figure 5.15a, we show the effect of increasing matrix size on the error of predicting the entries for the SYNTHETIC data set. Here, the matrix $\mathcal{O}$

is complete, *i.e.,* all material-illumination pairs are observed. We see, that with increasing size, the estimation for all entries gets more correct while the task being solved in some sense is also bigger (more different illuminations). Note that the total error residue can go up, but the estimation gets more accurate (compared to the ground truth).

When the matrix is reduced to a single row or column (Figure 5.15b) our approach can still estimate illumination and material. For a 1×5 matrix, which estimates a single material form multiple illuminations, the approach does well; but slightly degrades for a 5×1 setting, where multiple objects are seen under the same illumination.

Label quality. We assume the input images to have per-pixel normal and material labels. In Figure 5.15d, we study the effect of incorrect normal estimates by adding a different label of uniform noise to the normal. We see that good normal fair better with the error in the order of one percent, while larger errors produce an error that saturates still at a low total value. Also, note that for the INTERNET and the PHOTOS datasets, the geometry models are coarse and/or noisy. But in absence of ground truth, we could not measure estimation error.

5.4.2 Comparison

We compare possible alternatives to our approach as shown in Figure 5.2 and supplementary material. A simple approach could be using image-based rendering based approaches [64], however, these approaches require either flat geometry or a high-enough number of images to reproduce specular appearance, neither of which is available in our input images that show a single image of one condition only. Effectively, IBR would amount to projective texturing in our setting, that is compared to our approach in Figure 5.2a. An alternative could be to run a general intrinsic image approach [138] on the input images and use the average pixel color of the albedo k_d image as the diffuse albedo. The specular could then be the color that remains. While this would provide a specular value k_s, it is not clear how to get a glossiness value g (see Figure 5.2b).

Figure 5.16: Various photo-realistic image manipulations (*e.g.,* object insertion) made possible using our estimated material and environment parameters (see Section 5.4.3 for details).

5.4.3 Application

A typical application of our approach is photo-realistic manipulation of objects in Internet images as shown in Figure 5.16. Having estimated the material and illumination parameters from all the images, we can insert virtual replica into the image (Figure 5.16b, 5.16d), transfer reflectance estimated from other Internet images to new scenes (Figure 5.16a and Figure 5.16c), or introduce new object with material under the estimated illumination. Please note that the estimated environment maps were used to render object shadows on (manually added) ground planes (Figure 5.16a, 5.16c, and 5.16d).

5.4.4 User Study

We have compared our approach to the similar approach (SIRFS) that extract intrinsic images and lighting in a user study. When asking $N = 250$ subjects if one of five animated turn-table re-renderings using our material information or the model of SIRFS is preferred when showing both in a space-randomized 2AFC the mean preference was 86.5 % in our favor (std. error of the mean is 2.1 %). The chart of the user response, their mean, the exact sample counts and standard errors for individual images are presented in Figure 5.17.

5.5 Limitations

Our approach has three main limitations: First, we use a distant light model, which results in estimation errors in large rooms with interior lights. Second, although our environment map estimates lead to photo-realistic back projections and predictions, in absence of any data-driven regularization the illumination estimates themselves may look unnatural. This is primarily due to the limited samplings we have in our input measurements. Further, when objects are in close proximity, they may ‘show up’ in environment map estimations. For example, in Figure 5.11, we see the chair’s present red shading. This is because the incoming light for the white chair is distorted by the reflection of the red chair as shown in P4 (please refer to the video in the supplementary material).

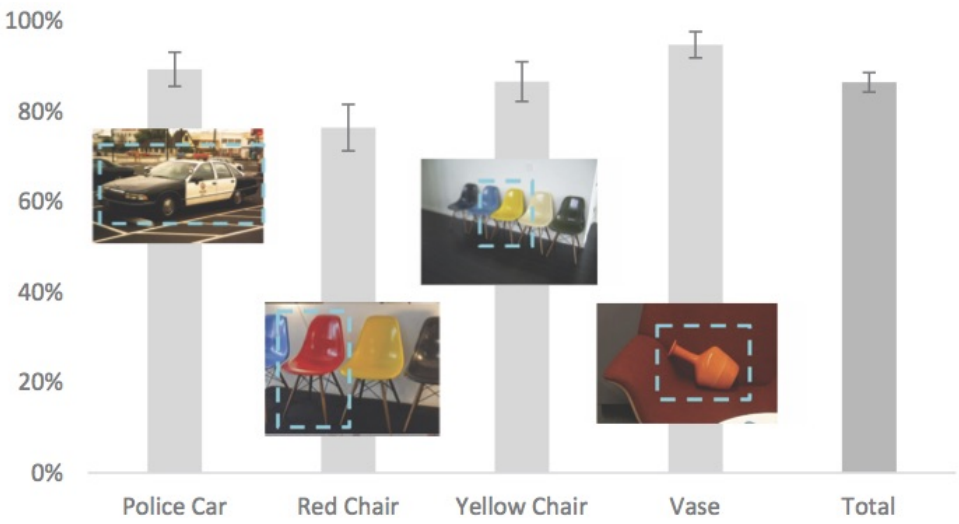


Figure 5.17: User study results. the vertical axis is the preference for ours, so more is better. Kinks are standard error of the mean, where small means certainty about the outcome.

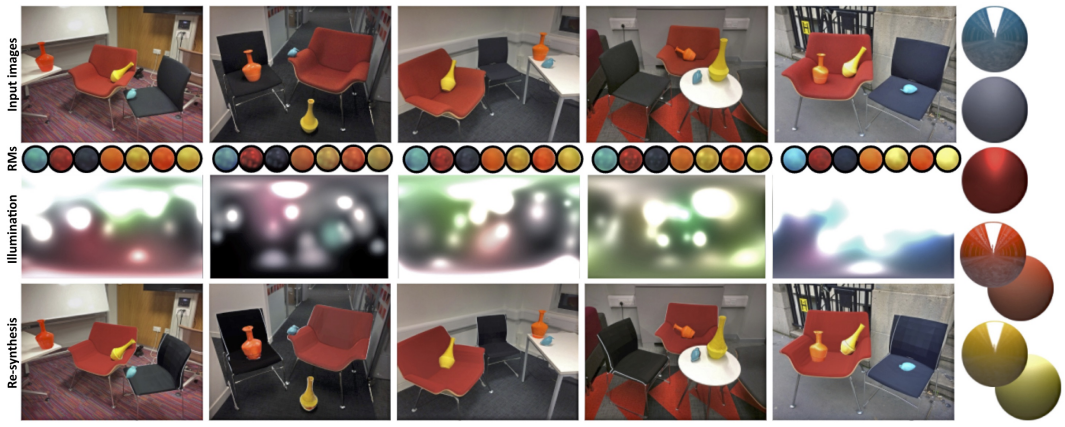


Figure 5.18: Results on the PHOTOS of five images of furniture and objects with seven materials in the protocol of Figure 5.9.

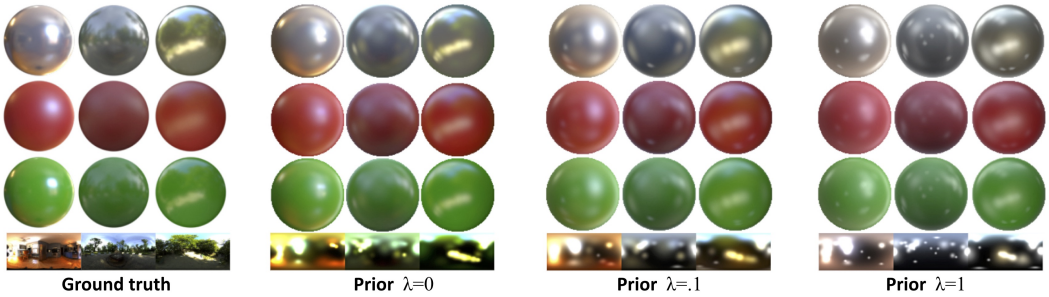


Figure 5.19: Effect of increasing prior. The top three rows show a 3×3 input. The next row shows illumination. Following the GT left, we increase the prior weight left to right. We note that illumination chromaticity decreases with increasing weight and that a good trade-off is likely at 0.1.

Chapter 6

Learning a Shared Shape Space for Multimodal Garment Design

6.1 Introduction

Developing effective tools for designing both real and virtual garments is becoming increasingly crucial. In today's digital age, consumers are a single-click away from online clothing stores, with an increasing appetite for new fashion styles. Similarly, virtual garment design attracts increasing interest from the entertainment industry since it is a significant component of creating realistic virtual humans for movies, games, and VR/AR applications. Both of these trends are creating a demand for fast, effective, and simple tools to design, edit, and adapt garments to various body shapes.

Traditionally, designing real and virtual garments has been a complex, iterative, and time-consuming process consisting of multiple steps. First, the designer sketches the look and feel of a garment or alternatively drapes fabric on a physical dress form. Then a professional pattern maker creates the 2D garment patterns referred to as *sewing patterns*. A sample garment is made from the sewing patterns and tested by draping on a real or simulating on a virtual mannequin. Often the garment parameters need to be iteratively adjusted followed by redraping or resimulation until the desired look is achieved. Finally, in order to ensure an operational outfit, the mannequin is animated to see how the garment drapes across various body poses. Furthermore,

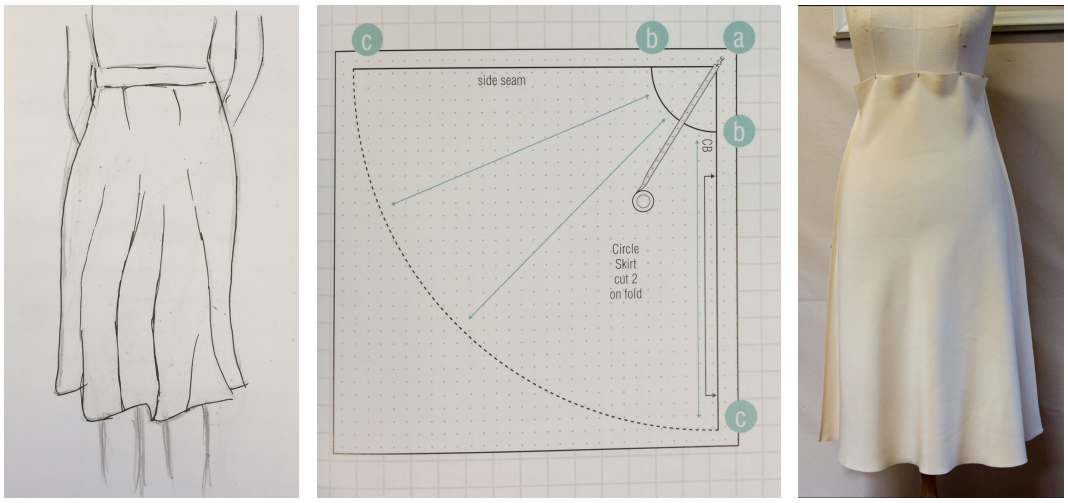


Figure 6.1: Garment designers often practice with different modalaties including sketch, parameter domain (e.g., parameters of 2D sewing patterns), and draped garments in 3D.

the same style garment needs to be adapted for different body proportions through a process called *pattern grading*. This essentially requires the complex process of garment design to be repeated multiple times.

Garment design is a complex process mainly due to the fact that it operates across three different spaces, namely 2D sketches for initial design, 2D sewing patterns and material selection for parameterized modeling, and 3D garment shape to model the interaction between garments and subject bodies producing *on-body garments* (see Figure 6.1). Much of pattern making involves using various rule-based specialized cuts and stitches (e.g., darts, pleats, yokes) to achieve desired folds on the final draped garment. Note that a particularly challenging scenario is designing free flowing garments where the characteristic patterns arise due to the interaction between fabric *hidden* behind the creases and boundary conditions induced by the underlying body shape. To aid with such challenging scenarios, an ideal computational tool should allow the designer to freely navigate across the three design spaces and effectively capture the interaction between them.

Designing such a unified treatment of the three spaces has remained elusive due to several challenges. First, interpreting 2D garment sketches requires a good understanding of what shapes and folds are possible in 3D. Although the problem appears to be badly ill-conditioned, as humans, we regularly use our experience

of typical garment folds and looks to 'regularize' the problem and interpret artist sketches. Second, the relation between the 2D garment parameters and the final 3D shape of the garment is highly non-linear depending not only on the shape of the garment itself but also its material properties, the pose, and the shape of the 3D body it is being draped on. This necessitates a computationally heavy cloth simulation process to visualize patterns arising out of garment folds, creases, and ruffles. Finally, modeling in 3D needs to ensure the physical plausibility of the garment by mapping the 3D designs to (near-) developable 2D sewing patterns.

We present a data-driven approach that overcomes the above challenges by learning a *shared shape space* that, for the first time, unifies 2D sketches; parameters of 2D sewing patterns, garment materials, and 3D body shapes; and the final draped 3D garments. We achieve this by *jointly* training multiple encoder-decoder networks that each specializes at linking pairs of representations (e.g., recovering garment and body parameters from a sketch or recovering the draped garment shapes from the parameters) while operating at a common embedding. To train these networks, we create a large scale synthetic dataset. Specifically, we first define a set of parameterized garment types (`shirt`, `skirt`, and `kimono`) and generate different garments by sampling this representation. Then, we simulate each garment on a set of 3D body shapes sampled from a deformable body model. Finally, for each of these simulated examples, we generate 2D sketches using non-photorealistic rendering. Thus, each examplar triplet in our dataset includes (i) a 2D sketch, (ii) garment and body parameters, and (iii) the resultant draped 3D shape of the garment. Subsequently, by jointly training multiple encoder-decoder networks via a novel multimodal loss function, we learn a common embedding that can be queried using any of the different modalities. Note that our network assumes that the 2D pattern topology remains fixed and does not allow transitioning across different pattern families. This remains a shortcoming of the proposed method and an open challenge to explore in future research.

The learned shared shape space enables several applications by linking the different design spaces. For example, starting from an input 2D sketch, we can (i) automatically infer garment and body shape parameters; (ii) predict the resultant

3D garment shapes from these parameters *without* going through an expensive cloth simulation process; (iii) directly texture the final garments using the linked 2D sewing pattern parameterization; (iv) sample from or interpolate in the latent space to generate plausible garment variations; or (v) retarget 2D garment patterns to new body shapes such that the resultant draped garments retain the original fold characteristics. At any stage of the design and retargeting process, the garment and body parameters inferred by our method can be provided to a cloth simulator to generate the physically accurate shape of the garment on the 3D body. Figure 3.1 shows several examples.

We qualitatively and quantitatively evaluate our approach against groundtruth test data, and demonstrate our interactive garment sketching for various applications. We also provide comparisons with alternative methods and show favorable performance (see Section 6.4). In summary, our main contributions are: (i) a method that learns a joint embedding of different design spaces for a given garment type; (ii) inferring garment and body parameters from single sketches; (iii) estimating the 3D draped configurations of the garments from the garment and body parameters to enable an interactive editing workflow; and (iv) facilitating fold-aware pattern grading across different body shapes via a novel garment retargeting optimization.

Code and data are available on our project webpage at
`http://geometry.cs.ucl.ac.uk/projects/2018/garment_design`.

6.2 Approach

6.2.1 Overview

Traditional garment design workflows involve interacting with one or more design domains, namely: (a) the 2D *design sketch* $\mathbb{S}$ that can be used to indicate a desired look and feel of a garment by specifying silhouette and fold lines; (b) the *parameter domain* $\mathbb{P}$ that allows the designer to specify both pattern parameters (i.e., size/material of sewing patterns) and body parameters (i.e., shape of the human body); and (c) the 3D *draped configuration* $\mathbb{M}$ (i.e., the 3D mesh) that captures the final garment shape on the target human body with a garment sized according to its

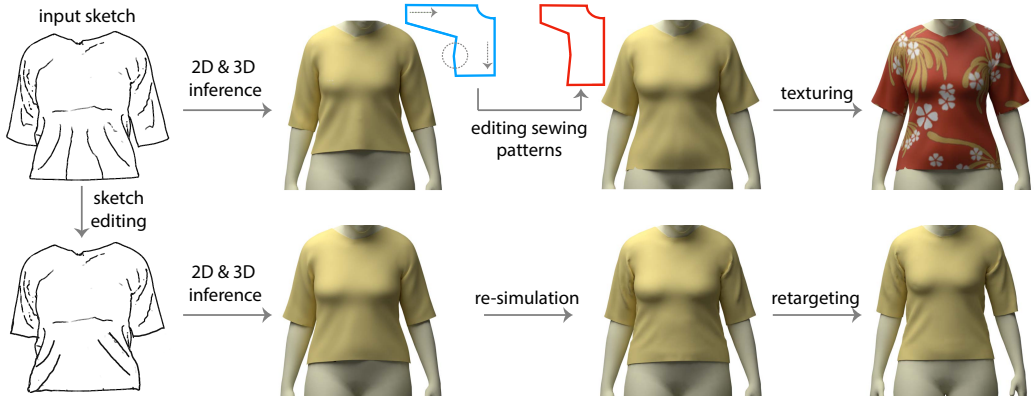


Figure 6.2: Given an input sketch, our network infers both the 2D garment sewing patterns (in blue) and the draped 3D garment mesh together with the underlying body shape. Edits in the 2D sewing patterns (e.g., shorter sleeves, longer shirt as shown in red) or the sketch (bottom row) are interactively mapped to updated 2D and 3D parameters. The 3D draped garment inferred by our network naturally comes with uv coordinates, and thus can be directly textured. The network predictions can be passed to a cloth simulator to generate the final garment geometry with fine details. Finally, the designed garment can be easily retargeted to different body shapes while preserving the original style (i.e., fold patterns, silhouette) of the design.

2D pattern parameters.

The above-mentioned domains have complementary advantages. For example, sketches provide a natural option to indicate visual characteristics of the folds such as density of folds, silhouette, etc.; parameters are effective to indicate direct changes to garment edits and/or specify target body shapes; while, the draped shape helps to generate previews under varying texture patterns, camera and/or illumination settings. By providing the designer with the ability to indicate target specifications via multiple modalities, we want to exploit the complementary advantages offered by the different domains to enrich the design process.

The above spaces, however, have very different dimensions making it challenging to robustly transition from one domain to another, or accept inputs across multiple modalities. In other words, a traditional data-driven method can easily overfit to the example datasets and result in unrealistic results in case of new test data. For example, a learned network to help transition from $\mathbb{S} \rightarrow \mathbb{M}$ easily leads to over-fitting as seen on test sketch input in Figure 6.3 (see Section 6.4). More importantly, such an approach does not give the designer access to the pattern and/or

body parameters to edit.

Instead, we propose to learn a shared shape, i.e., latent, space by jointly learning across the three domains using a novel cross-modal loss function (Section 6.2.3). Our key observation is that the shared latent space *regularizes* the learning problem by linking the different domains. From a usage point of view, the artist enters the design space via a sketch, and then continue making further changes by directly editing the inferred pattern and/or body parameters.

The designer can now create garments via a multimodal interface by seamlessly indicating sketch behavior, garment or body parameters, or retexturing (see Figure 6.2). A garment, thus designed, can then be easily remapped to a range of other body shapes, facilitating pattern grading. To ensure the original sketched folding behaviors do not get lost in the new draped garments adapted for the different body shapes, we present a novel retargeting method that formulates an optimization in the shared latent space (Section 6.2.4). Before describing the methods in detail, we next introduce the specific representation choices used in this work.

6.2.2 Parameter Space

In this work, we tested on three different garment types namely `shirts`, `skirts`, and `kimonos`. We parameterize each garment type using a small number of key dimensions (e.g., length and width of sleeve for `shirt`, length of the waistline for `skirt`) and material properties, i.e., stretch, bend, and shear stiffness (Figure 6.4).

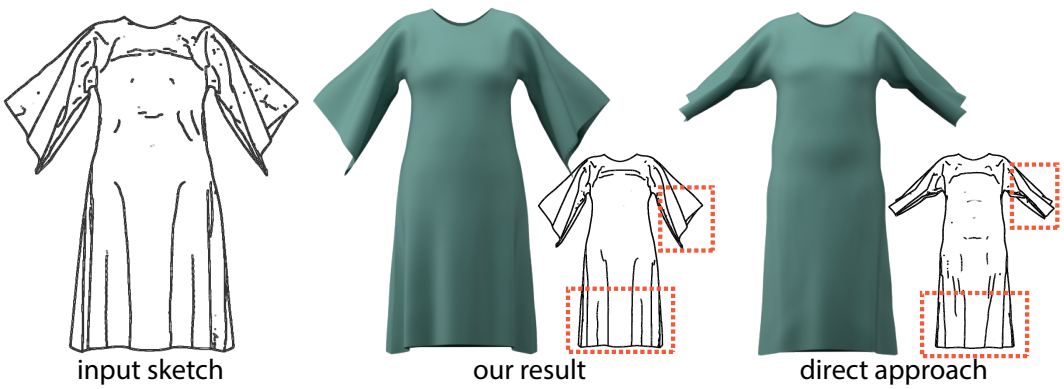


Figure 6.3: While a network trained directly to infer the draped garment from an input sketch overfits to training data, learning a joint (latent) shape space across different modalities leads to better generalization.

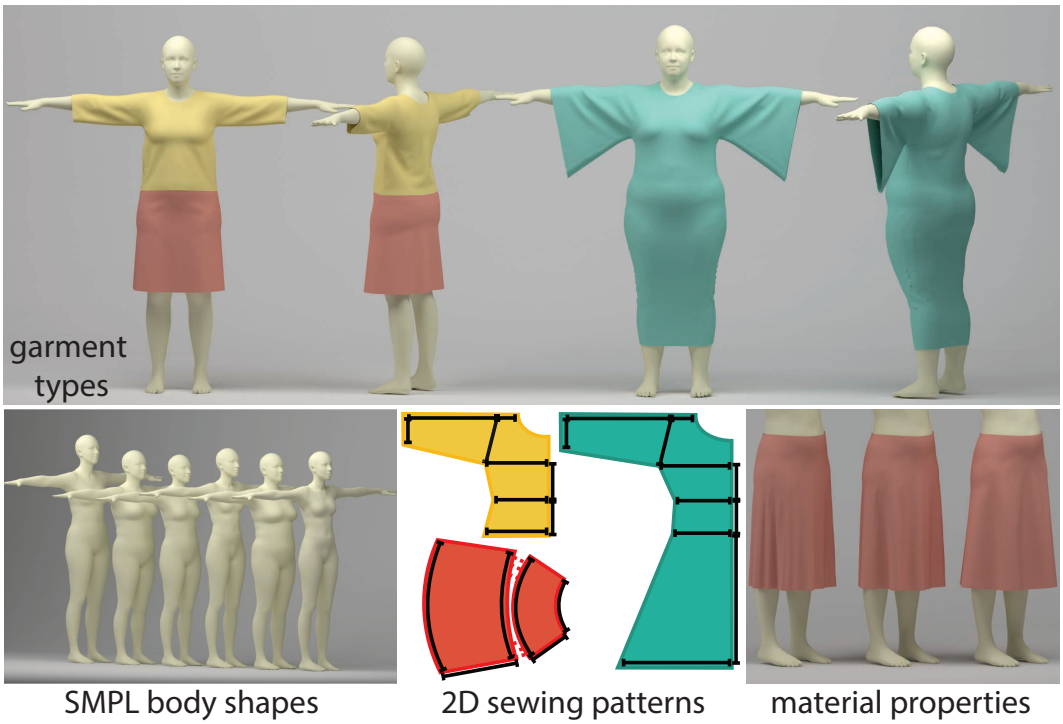


Figure 6.4: Our dataset includes three types of garments (`shirt`, `skirt`, and `kimono`). Each garment is parameterized by a small number of 2D sewing pattern dimensions as well as material properties including stretch, bend, and shear stiffness parameters. We sample different garment parameters and simulate on different 3D body shapes sampled from the parametric SMPL body model.

Specifically, the number of parameters for each garment types were: 4 for `kimono`, 9 for `shirt`, and 11 for `skirt`, respectively. We collect all the garment parameters (including dimension parameters and material parameters) in a vector $\mathbf{G}$. We adopt the SMPL [156] model for the underlying 3D body representation. Given a specific pose, SMPL provides a 10-dimensional vector $\mathbf{B}$ to describe the body shape variation. Note that the pattern parameters are encoded relative to body size, i.e., vertical parameters are related to body height and horizontal parameters are related to chest circumference or arm length. We denote the combined body and garment parameter space as, $\mathbf{P} = (\mathbf{G}, \mathbf{B}) \in \mathbb{P}$.

In order to generate a dataset of training examples for a given garment type, we first randomly sample garment parameter instances from $\mathbf{G}$ to generate the 2D patterns. With a target pose, we then sample one of the 2k female body shapes in the FashionPose dataset [157] to generate samplings of $\mathbb{P}$ resulting in a total of

8000 combinations. We then simulate combinations of garment samples over body samples using the cloth simulator in FleX [158] (see Figure 6.7). This results in draped garment meshes, which we refer to as the mesh $\tilde{\mathbf{M}}$. Given a collection of such meshes $\{\tilde{\mathbf{M}}\}$ corresponding to the same type of garment, we obtain a compressed representation by performing Principal Component Analysis (PCA) and represent each garment mesh using the first k ($k = 200$ in our tests) basis vectors, which we denote as $\mathbf{M}$. Finally, we render each $\mathbf{M}$ from a frontal viewpoint using Suggestive Contours [159] to approximate the corresponding 2D sketch. We denote the rendered sketch image as $\tilde{\mathbf{S}}$ and apply the convolutional layers of DenseNet [160] to generate a 2208-dimensional descriptor $\mathbf{S}$. Thus, for each instance (parameter combination) in our dataset, we have a 3-element set $(\mathbf{P}, \mathbf{M}, \mathbf{S})$. Given this dataset, our goal is to learn a joint shape space shared between the 3 modalities.

6.2.3 Shared Shape Space

Given the 3 different modalities $(\mathbf{P}, \mathbf{M}, \mathbf{S})$ for each example, our goal is to learn a common K-dimensional shared shape space (or a latent space), $\mathbf{L}$, that will enable a multimodal design interface (in our experiments, $K = 100$). We achieve this goal by learning the following mapping functions: (i) sketch descriptor to latent space ($F_{S2L} = \mathbf{S} \rightarrow \mathbf{L}$), (ii) parameter space to latent space ($F_{P2L} = \mathbf{P} \rightarrow \mathbf{L}$), (iii) latent

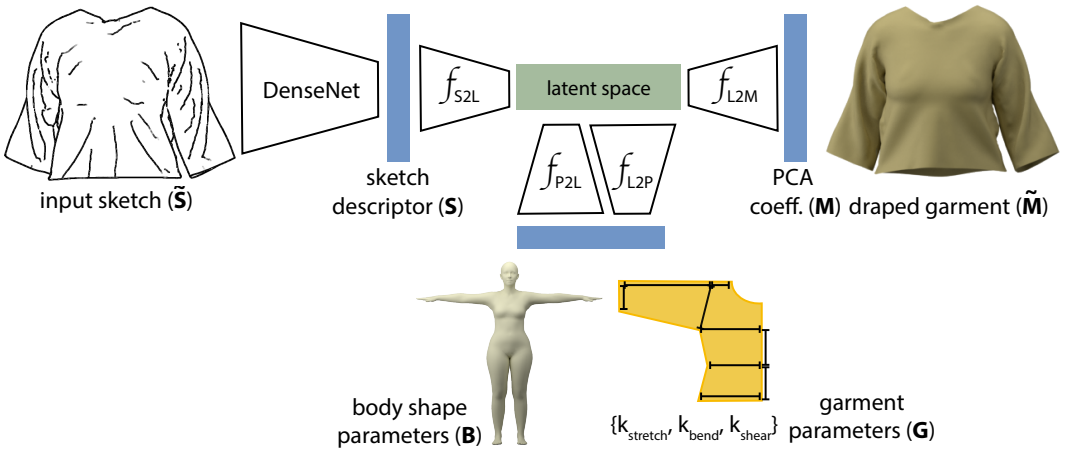


Figure 6.5: We learn a shared latent shape space between (i) 2D sketches, (ii) garment and body shape parameters, and (iii) draped garment shapes by *jointly* training multiple encoder-decoder networks.

space to parameter space ($F_{L2P} = \mathbf{L} \rightarrow \mathbf{P}$), and (iv) latent space to the draped garment shape ($F_{L2M} = \mathbf{L} \rightarrow \mathbf{M}$). We learn these mappings by jointly training four encoder-decoder neural networks (i.e., sketch-to-parameter, sketch-to-3D garment shape, parameter-to-3D garment shape, and parameter-to-parameter) that share a common embedding space (see Figure 6.5). We describe the network architecture with more details in section 6.3.

We define a loss function that jointly captures the intention of each of the encoder-decoder networks. Specifically, we penalize (i) the error in estimating garment and body shape parameters from a sketch, (ii) the error in estimating the draped garment shape from a sketch or a parameter sample, and (iii) the reconstruction error of a parameter sample from itself in an auto-encoder fashion. Thus, the combined loss function is defined as:

$$\begin{aligned}\mathscr{L}(\mathbf{P},\mathbf{M},\mathbf{S}) = \omega_1 \|P - f_{L2P}(f_{S2L}(S))\|_2 + \omega_2 \|M - f_{L2M}(f_{S2L}(S))\|_2 \\ + \omega_3 \|M - f_{L2M}(f_{P2L}(P))\|_2 + \omega_4 \|P - f_{L2P}(f_{P2L}(P))\|_2,\end{aligned} \tag{6.1}$$

where $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ denote the relative weighting of the individual errors. We set these weights such that the average gradient of each loss is at the same scale (in our experiments $\omega_1 = \omega_2 = 40\omega_3 = 40\omega_4$). Empirically the consistency terms (the last three terms) make a significant difference on the quality of the network prediction on test data.

6.2.4 Garment Retargeting

One of the most common tasks in real or virtual garment design is retargeting, i.e., adapting a particular garment style to various body shapes. Given a garment $\mathbf{G}$ designed for a particular body shape $\mathbf{B}$, the goal of the retargeting process is to identify a new set of garment parameters $\mathbf{G}'$ for a new body shape $\mathbf{B}'$ such that the look and feel of the draped garments on both body shapes are similar. Naively using the same set of garment parameters on a new body shape does not preserve the original style as shown in Figure 6.15. On the other hand, deforming the draped garment in 3D to match the desired style does not ensure a mapping to a valid configuration of sewing patterns. Instead, we propose a novel optimization process

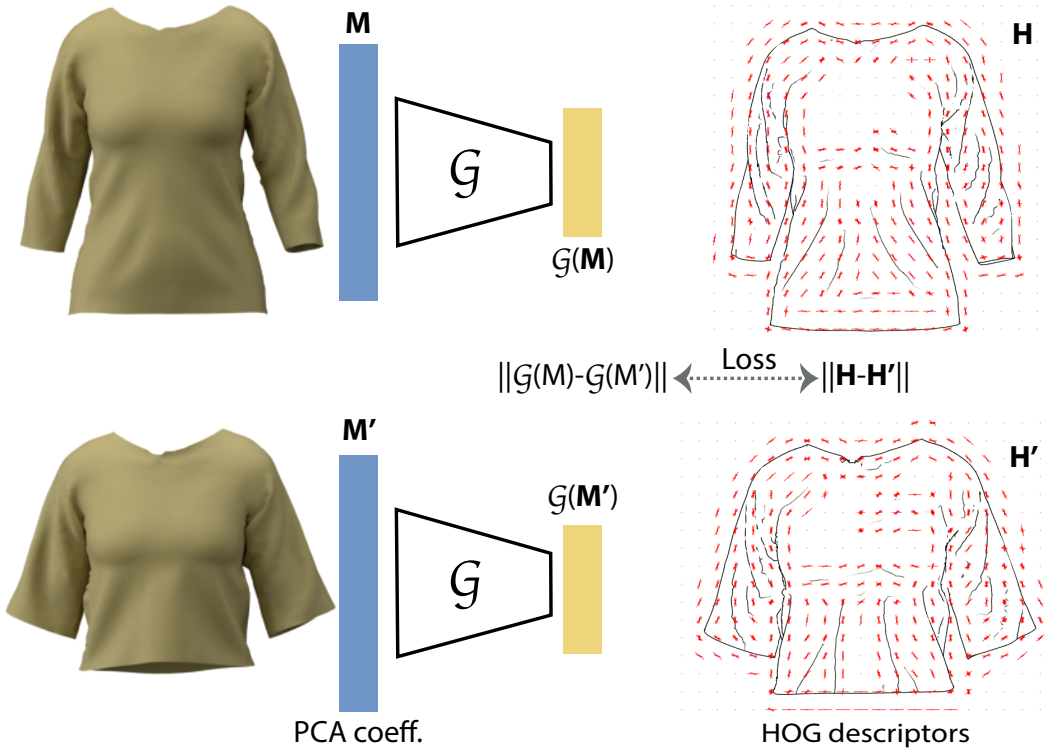


Figure 6.6: We train a Siamese network $\mathscr{G}$ that learns an embedding of draped garment shapes $\{\mathbf{M}\}$. The distance between a pair of draped garments $(\mathbf{M},\mathbf{M}')$ in this learned embedding space is similar to the distance between the HOG features of the corresponding sketches. Once trained, the loss function can be differentiated and used for retargeting optimization.

that utilizes the shared shape space presented in the previous section.

As a key component of our optimization, we learn a style-aware distance metric between draped garments. Specifically, given two sets of garment-body instances $(\mathbf{G},\mathbf{B})$ and $(\mathbf{G}',\mathbf{B}')$, our goal is to learn a distance measure between their corresponding draped garments $(\mathbf{M},\mathbf{M}')$ that is similar to the distance between the sketches of the draped garments, $(\mathbf{S},\mathbf{S}')$. We achieve this goal by learning an embedding of draped garments, $\mathscr{G}(\mathbf{M})$.

Given pairs of $(\mathbf{M},\mathbf{S})$ and $(\mathbf{M}',\mathbf{S}')$, we train a Siamese network such that $\|\mathscr{G}(\mathbf{M})-\mathscr{G}(\mathbf{M}')\|$ is similar to the distance between $(\mathbf{S},\mathbf{S}')$ (see Figure 6.6). We measure the distance between two sketches as the distance between their HOG features.

Given the learnt embedding $\mathscr{G}(\mathbf{M})$, we define an objective function for garment retargeting. For a pair of $(\mathbf{B},\mathbf{G})$ and a new body shape $\mathbf{B}'$, we optimize the following

energy function:

$$\mathscr{E}(\mathbf{G}'|\mathbf{G},\mathbf{B},\mathbf{B}') = \|\mathscr{G}(\mathbf{f_{L2M}}(\mathbf{f_{P2L}}(\mathbf{G},\mathbf{B}))) - \mathscr{G}(\mathbf{f_{L2M}}(\mathbf{f_{P2L}}(\mathbf{G}',\mathbf{B}')))\|. \quad (6.2)$$

Since both $\mathscr{G}$ and the mappings f_{L2M}, f_{P2L} are learned via differentiable networks, we can efficiently compute the gradient of the objective function during the optimization. In our experiments, we used an L-BFGS solver to optimize the above energy.

6.3 Implementation Details

In the following, we provide details about the data generation process and the network architecture.

Data generation. . When generating our training dataset, for a given garment type, i.e., `shirt`, `skirt`, or `kimono`, we first sample a set of garment parameters and generate an isotropic triangle mesh within the silhouette of sewing patterns using centroidal Voronoi triangulation. We then simulate each sampled garment

Figure 6.7: To generate synthetic data, we first shrink the body of the mannequin in rest pose into its skeleton and let the triangle mesh of the 2D sewing patterns drape naturally over it. After the draping converges, we stitch the boundary and inflate the skeleton back to its original shape. We further animate the mannequin into the target body pose to generate our final simulation result.

on varying body shapes. Note that we discarded any pairs of sampled clothes and bodies whenever the draped garment was too tight or too loose after simulation. The cloth simulator we use, i.e., FleX, is particle-based and is sensitive to the edge lengths of the mesh being simulated. Thus, we fix the average edge length across all samples of the garment, leading to meshes with varying topology and face count. In order to ensure a consistent mesh topology to perform PCA on draped garments, we use one of the simulated examples as a reference mesh topology and remesh the remaining examples. Specifically, we locate the vertices of the reference template in its associated triangle in a common sewing pattern space and compute its position in every other simulated example via barycentric coordinates. In order to balance efficiency and accuracy of simulations, we used a mesh template with about 27.5k vertices for the `kimono` dataset, 12.3k vertices for the `shirt` dataset, and 5k for the `skirt` dataset. Emperically, we found these resolutions to consistently produce fine detail wrinkles during simulation. The simulation of 8000 samples for each garment type took about 60 hours to generate with 2 Nvidia Titan X GPUs.

Once we have a set of simulated, i.e., draped, garments we generate the corresponding sketches using Suggestive Contours [159] referred as NPR (non-photorealistic rendering) in this paper. We only render the sketch from the frontal view as this is the most common practice in fashion industry. We perform data augmentation in the resulting sketches by removing small line segments, curve smoothing, adding Gaussian blur, etc. All sketches are centered and cropped into a 224×224 square patch. We extract a 2208-dimensional feature vector for each patch via DenseNet [160] (we use the DenseNet-161 architecture provided in the TorchVision library [161] and use the output of the fourth dense block as our feature vector). Note that while we used DenseNet to provide feature descriptor for the input sketch, other descriptors such as VGG [162] or ResNet [163] can alternatively be used in our setup.

Network architecture. . The encoder and decoder networks we train to learn a shared shape space, i.e., latent space, are composed of *linear blocks* which are linear layers followed by Rectifying Linear Unit (RELU) activations and batch

normalization. Specifically, the encoder, F_{S2L}, takes as input a 2208-dimensional feature vector of a sketch and maps it to the $K = 100$ dimensional latent space with 10 linear blocks (the output dimension size is kept fixed in the first 6 blocks and gradually decreased to 1000, 500, 200, and 100 in the remaining 4 blocks). The encoder, F_{P2L}, takes as input a p-dimensional parameter vector representing the garment and the body shape ($p = 22$ for `shirt`, $p = 17$ for `skirt`, $p = 24$ for `kimono` where 3 material parameters and 10 body shape parameters are consistent across the different garment types) and maps it to the latent space with 6 linear blocks (the output dimension size is kept fixed in the first block and increased to 100 in the second block). The decoder, F_{L2M}, takes as input the $K = 100$ dimensional latent space vector and maps it to the 200-dimensional PCA basis that represent the draped garment shape. This decoder consists of 6 linear blocks (the output size of the first two blocks is 100 and the output size of the remaining blocks are 200). Finally, the decoder, F_{L2P}, takes as input the $K = 100$ dimensional latent space vector and maps it to the parameters of the garment and the body shape. This decoder consists of 6 linear blocks, where the first 5 blocks keep the output dimension size fixed and the last block changes the output size based on the garment type.

We jointly train the encoder-decoder architectures for 20k epochs with a learning rate of 0.1 and batch size of 64. We use stochastic gradient descent for network backpropagation. We used pyTorch for implementation and the proposed network takes about 3 hours to train on each dataset with one Nvidia Titan X GPU.

Retargeting. . For retargeting garments across different body shapes, we train a Siamese network that learns an embedding of the draped garments such that the distance between two draped garments is similar to the distance between their corresponding sketches. This Siamese network takes as input a 200-dimensional PCA vector and maps it to an 100-dimensional embedding space with 6 linear blocks (the output dimension size is kept fixed in the first 5 blocks and decreased to 100 for the last block). We train this network for 20k epochs with a learning rate of 0.03 and batch size of 64. We use stochastic gradient descent for network backpropagation.

We refer the reader to the supplementary material for a detailed network archi-

Figure 6.8: For each reference image, we ask users to draw the corresponding sketch that we provide as input to our method. Our method generates draped garments that closely resemble the reference images.

tecture configuration.

6.4 Evaluation

We evaluate our method qualitatively on real and synthetic data and quantitatively on synthetic data. We split our synthetic dataset into training (95%) and testing sets (5%) such that no garment and body parameters are shared across the splits. For evaluations on the synthetic data, we use 200 samples from the test set for each garment type. We note that given an input sketch, our method estimates the corresponding garment and body shape parameters as well as the shape of the draped garment. In a typical design workflow, once satisfied with the design, we expect the designer to perform a cloth simulation using the estimated garment and body shape parameters to obtain the final and accurate look of the garment. The draped garment shape predicted by our network closely resembles the simulation result performed by using the predictions of the garment and body parameters. Thus, we only provide

Figure 6.9: For each sketch, we show our output with/without texture; NPR rendering of our output and the nearest sketch retrieved from our database. As highlighted in orange, our result plausibly captures the folds provided in the input sketch.

the final simulated results. Note that since we used a particle-based simulator, the simulation output used a certain safety margin between objects even when surfaces were supposed to be 'touching' each other. In our setup, this safety margin was 4mm with respect to an average body height of 1.8m. In practice, we observed some minor interpenetrations which were removed via resimulation with estimated parameters as shown in Figure 6.12. In Figure 6.9, for each input synthetic sketch, we show the simulated results with the garment and body shape parameters predicted

Figure 6.10: Our method generates consistent uv parameters across different instances of the same garment type. In this example, the alignment between the texture patterns can be best seen in the neck region.

by our method. Since our network can predict the corresponding 2D sewing pattern parameters as well, we can easily generate uv coordinates for the draped garment and texture it. To show that our network does not learn to memorize the training examples, we also show the nearest neighbors retrieved from our training dataset using the DenseNet features of sketches. As highlighted in the figure, while the NPR renderings of our results closely resemble the input sketch, the nearest neighbors fail to capture many folds present in the input sketch.

In Figure 6.8, we provide examples on real images to test the generality of our approach. For 6 different reference images, we ask different users to draw the corresponding sketch that is provided as input to our method. As shown in the figure, our method is able to generate the draped garments that closely follow the reference image and the input sketches.

We demonstrate the consistency of the uv parameters generated by our method by texturing the same garment type with varying parameters draped on varying body shapes (see Figure 6.10). Our method generates uv parameters that are free of distortions compared to generic solutions.

Quantitative evaluations. . We quantitatively evaluate the performance of the different mappings learned by our network. Specifically, starting either with an input sketch (or a set of garment and body shape parameters), we first map the input to the latent space via f_{S2L} (or f_{P2L}). Then, we measure the error in the predicted draped garment mesh (the output of f_{L2M}) and the estimated garment and body shape parameters (the output of f_{L2P}). For the draped garment, we report the average L2 error both on the PCA coefficients and the vertex positions, similarly, for the body shape, we report the average L2 error both on the SMPL shape parameters and the

vertex positions. We normalize all the parameters and the garment and body shapes to the range $[0, 1]$ and report the percentage errors. Table 6.1 provides results when a sketch or a set of garment and body parameters are provided as input. In Figure 6.11, we show the loss curve on test set during training.

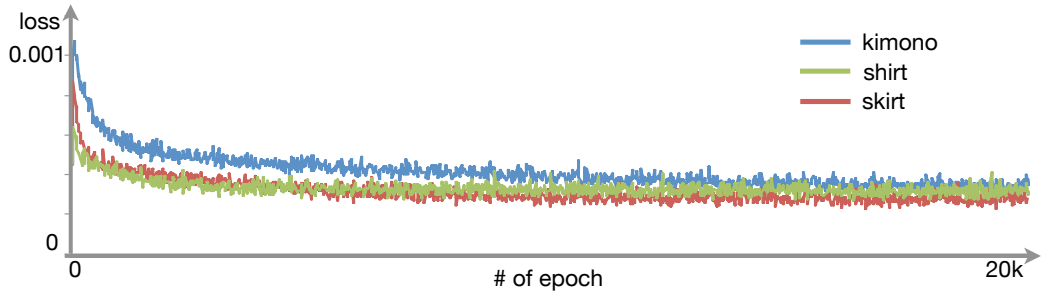


Figure 6.11: We show the combined loss (Equation 6.1) on test set during training for different garment types.

Joint shape space evaluation. . In order to demonstrate the benefit of jointly training a shared latent space across three modalities, we train an alternative single encoder-decoder network composed of the mappings f_{S2L} and f_{L2M}. As shown in Figure 6.3, this direct mapping overfits to the training data and is not able to generalize. In contrast, jointly training for additional mappings regularizes the problem and leads to better performance during test time.

Our learned shared latent space is compact and smooth, as shown in Figure6.13.

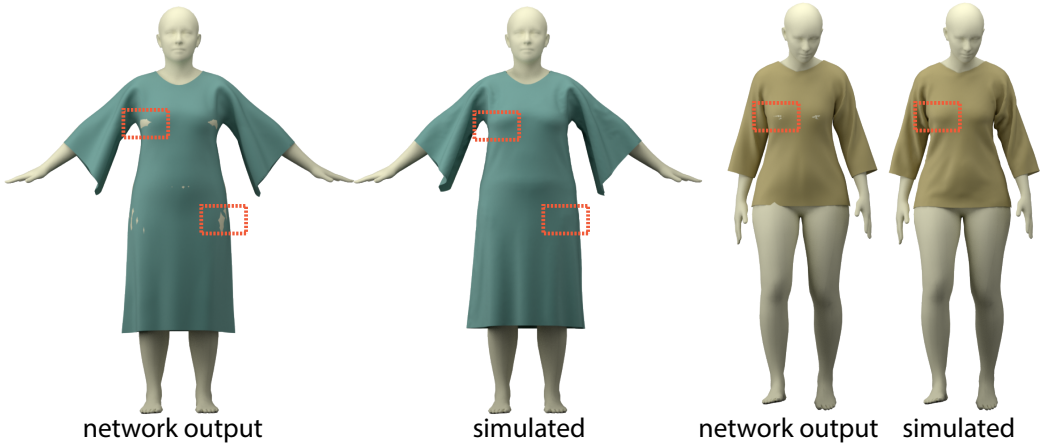


Figure 6.12: Draped garment shape predicted by our network closely resembles the simulation result using predicted garment/body parameters, and hence can be used to optionally fix draped garment and body intersections.

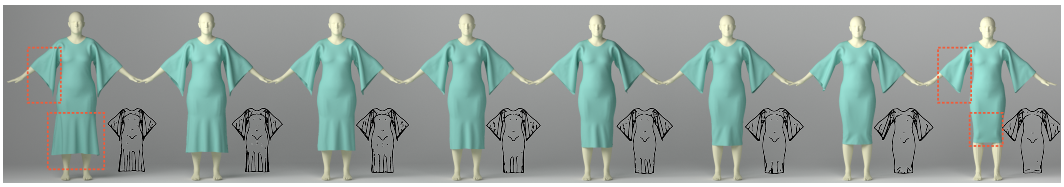


Figure 6.13: Given two kimono designs shown on the left and the right, we interpolate between their corresponding descriptors in the shared latent space. For each interpolation sample, we show the 3D draped garments inferred by our network and the NPR rendering of the simulated mesh from the corresponding sewing parameters. Our shared latent space is continuously resulting in smooth in-between results.

When we linearly interpolate between two samples in the latent space, we obtain intermediate results that change smoothly both in terms of 3D predict mesh and garment sewing parameters.

Comparison. . In Figure 6.14, we compare our approach with one example from Figure 8 of [2]. Given the reference image in this example, we ask a user to provide the corresponding sketch that is provided as input to out method. The estimate draped garment shape by our method is of similar quality but is generated at interactive rates once the network is trained by our skirt dataset in contrast to the instance-wise computation-heavy approach of [2].

Figure 6.14: We test our method on one of the examples provided in Figure 8 of [2]. We get results that are similar quality at interactive rates once the network is trained by our skirt dataset compared to their instance-wise computationally-heavy approach.

Retargeting evaluation.. As shown in Figure 6.15, draping the same garment on different body shapes do not preserve the style of the garment. Our retargeting optimization, in contrast, identifies a new set of garment parameters that would

preserve the style of the original garment on a body shape.

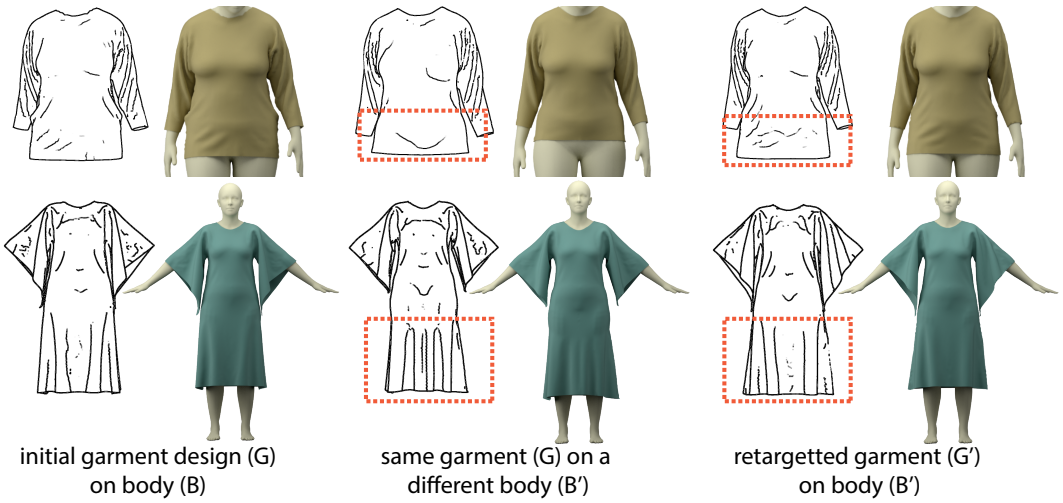


Figure 6.15: Given an initial garment design, draping the same garment on a different body shape does not preserve the style of the design. In contrast, our retargeting method optimizes for a new set of garment parameters that result in the same look of the garment on a new body shape.

Our retargeting optimization uses a distance measure between draped garments based on the Siamese network introduced in Section 6.2.4. We show a tSNE visualization [164] of the embedding learned by this network in Figure 6.16. For each embedding, we provide the sketches corresponding to various random samples showing that similar sketches get clustered.

Interactive user interface. . Based on the shared shape space we learn across different design modalities, we implement an interactive user interface (see Figure 6.18). In this interface, the user can interactively edit the input sketch, the garment or the body shape parameters, or the uv coordinates and interactively visualize the corresponding draped garment. We refer to the supplementary video which shows each of these editing options.

User study.. Finally, we conduct a user study to evaluate how closely our results resemble the input sketches. Specifically, given two input sketches (S_i, S_j), we show the corresponding draped garment shapes (M_i, M_j) predicted by our network and ask the user to pair the sketches to the garments (e.g., S_i should be paired with M_i). If the network output successfully captures the folds in the input sketch, we expect the accuracy of such pairings to be high. However, if S_i and S_j are similar to begin

with, the pairing becomes ambiguous. We generate 400 pairing queries for each garment type and ask 13 Amazon Mechanical Turk users to answer each query. We plot the accuracy of the pairings vs. the similarity of the input sketches (in terms of the L2 distance between DenseNet features) in Figure 6.19. Since we expect the simulation result (performed using the parameters predicted by our network) to capture more details, we repeat the same user study using the simulation result as the draped garment shape. As shown in the plots, the users are accurate unless the input sketches are very similar. The accuracy slightly increases when simulation results are used validating that the generated draped garments perceptually capture the fold characteristics in the input sketches.

6.5 Discussion and Limitations

The proposed joint network enables us to transfer between different domains in a freestyle. However the user must ensure their editions stay in the feasible area, i.e. 1) the sketch needs to follow the style and scale in the training set and presented in frontal view, and 2) the parameters provided by the user have to be in the same range as during the training set generation (this can be restricted in the UI design).

Our method has certain limitations that open up interesting future directions. For each garment type, we have a pre-defined set of 2D sewing parameters. Thus, we cannot represent garment shapes that are not covered by this set as in the inset example. Additionally, the choice of garment simulator also limits the pattern complexity and material variance that can be generated in the dataset. As a result, the network learns to produce pattern and material variations only as observed in the data. As simulation frameworks get more powerful, we expect that the practicability of our method for fashion design and modeling will improve.

In this work, we learned shape spaces specialized to different garment types (`shirt`, `skirt`, and `kimono`). However, the trained network are garment-specific

and cannot be reusable for new garment types. A difficult research question is how to unify these different garment-specific shape spaces via a common space. This is challenging given the complex discrete and continuous changes necessary to transition from one garment type to another. One possibility is to perform interactive exploration where the user annotates additional cut/fold lines as recently demonstrated in [90]. Note that the lack of discrete parameters also limited the design complexity of our system and hence we cannot support operations such as insertion of necklines, collars, or cuffs.

Table 6.1: Starting with an input sketch (top) or a set of garment and body shape parameters (bottom), we report the average L2 error in the estimated draped garment (both in terms of PCA basis and vertex positions), body shape (both in terms of SMPL parameters and vertex positions), and the garment parameters. All the parameters are normalized to the range $[0, 1]$.

From **sketch**	Garment mesh				Body shape				Garment parameter	
	Training set		Testing set		Training set		Testing set		Training set	Testing set
	L2 PCA	L2 Vert.	L2 PCA	L2 Vert	L2 SMPL	L2 Vert.	L2 SMPL	L2 Vert	L2	L2
Shirt	1.99%	1.58%	4.21%	3.82%	2.02%	1.77%	5.22%	4.67%	5.26%	6.73%
Skirt	1.76%	1.14%	2.38%	2.11%	1.59%	1.13%	3.29%	2.06%	3.79%	4.99%
Kimono	2.28%	1.70%	5.45%	4.48%	3.84%	2.74%	7.48%	5.35%	6.94%	8.47%

From **parameter**	Garment mesh				Body shape				Garment parameter	
	Training set		Testing set		Training set		Testing set		Training set	Testing set
	L2 PCA	L2 Vert.	L2 PCA	L2 Vert	L2 SMPL	L2 Vert.	L2 SMPL	L2 Vert	L2	L2
Shirt	1.56%	1.16%	3.57%	3.01%	1.47%	1.20%	3.51%	3.29%	3.88%	3.89%
Skirt	1.29%	0.98%	2.03%	1.61%	1.09%	0.86%	2.14%	1.68%	2.47%	3.60%
Kimono	1.80%	1.58%	5.21%	2.59%	2.80%	2.01%	4.81%	3.33%	4.13%	4.33%

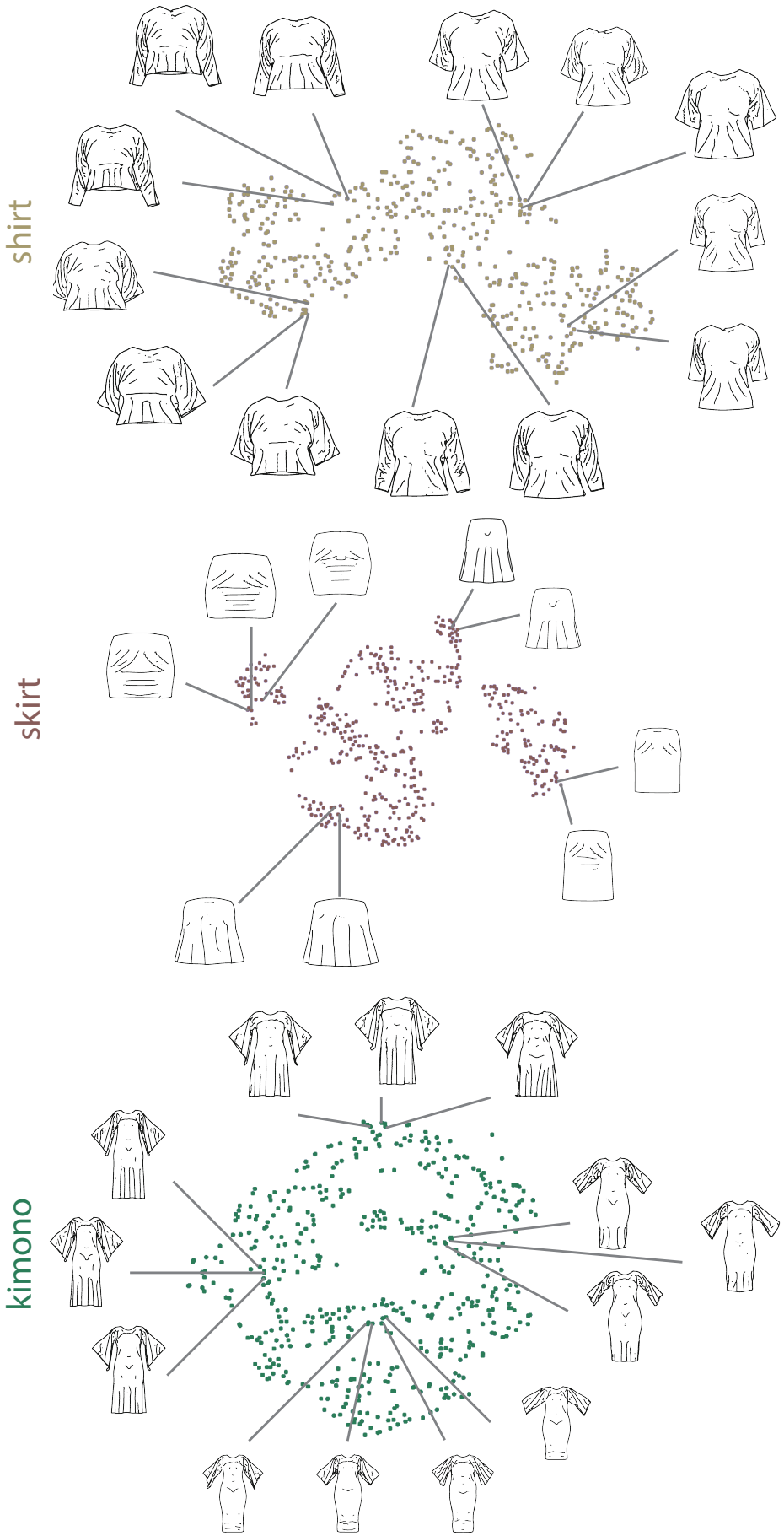


Figure 6.16: The distance embedding we learn in Section 6.2.4 clusters similar style garments together.

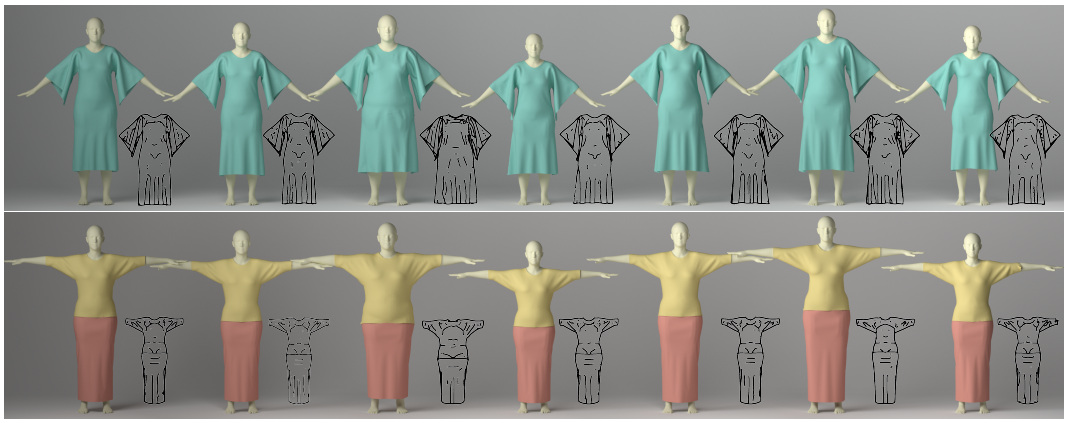


Figure 6.17: Given a reference garment style on the left, we retarget it to various body shapes. For each example, we also provide the NPR renderings of the draped garments.

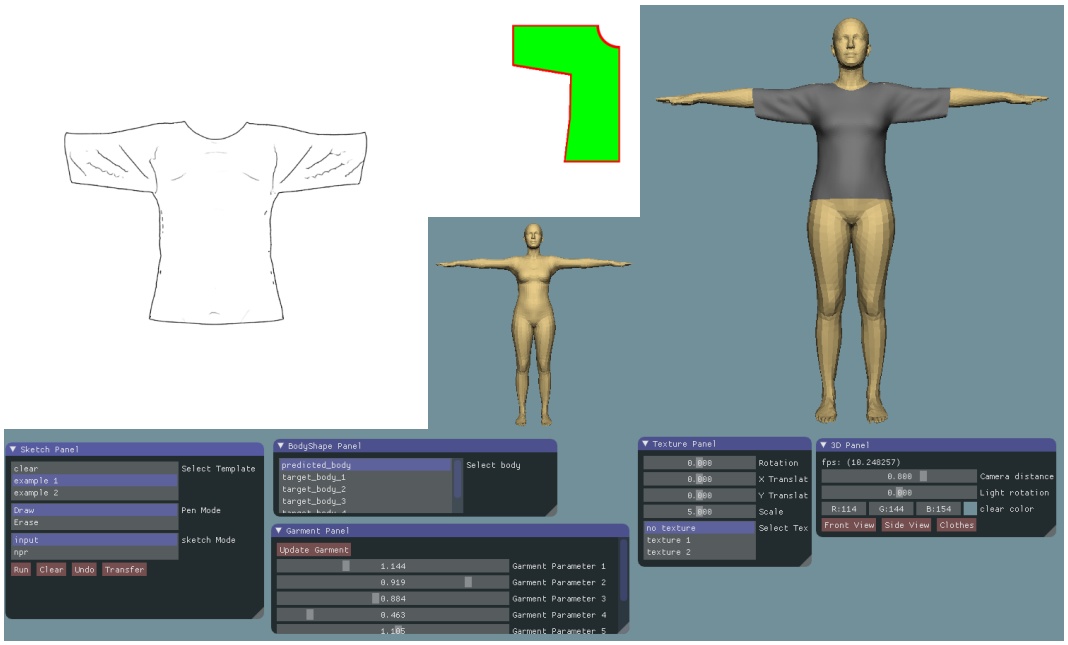


Figure 6.18: Our user interface allows the user to edit the sketch, garment or body shape parameters, or the texture coordinates and visualizes the updated draped garment in 3D at interactive rates enabling multimodal garment design.

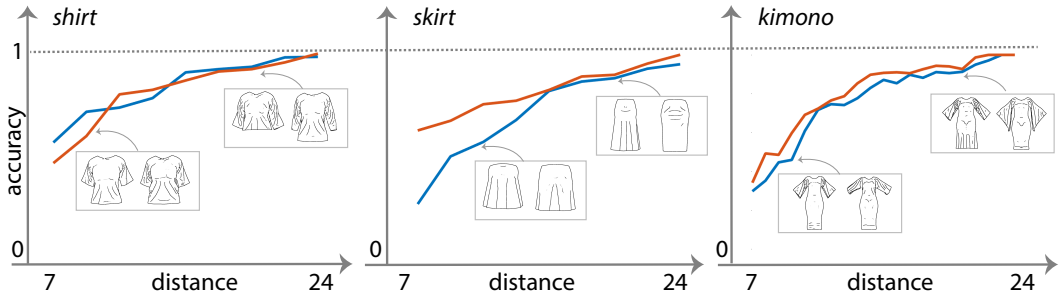


Figure 6.19: We asked users to pair two given sketches with two draped garment shapes both in the form of the network output (in blue) and the simulation result (in pink). We plot the accuracy of the pairings versus the similarity of the given sketches. The users are accurate with pairings unless the input sketches are very similar.

Chapter 7

Conclusion and Future Work

In this thesis, we developed methods for acquiring high-quality and plausible 3D content (i.e. geometry, texture, material, and illumination) from a real 3D scenario under several different settings.

In Chapter 3, we presented an algorithm for dynamic SfM to recover both part structure and their motion for articulated objects starting from only a pair of input images. The main gain is to detect scene changes from sparse uncontrolled measurements. We proposed a solution that simultaneously recovers the 3D structures along with respective part motion, and in the process achieves increased accuracy.

Several future avenues remain unexplored: First, we believe having additional prior will further regularize the problem (e.g., using a partially known set of known objects like chairs, tables, etc. in office environments). Second, it will be worthy to make the algorithm faster to support near real-time performance. For example, the RANSAC sampling and initialization can be performed in parallel, possibly using GPU speedup. We would also like to test the multi-view setting with images coming from multiple mobile phone inputs.

In Chapter 4, we formulated and provided a solution to the problem of unsupervised appearance transfer from in-the-wild 2D images to 3D model collections. The main insight of our approach is that knowledge of the geometry of a 3D model *similar* to the imaged object allows us to separate and undo geometric and photometric texture distortions in the image. Our pipeline is simple and relies on discovering a low complexity texture model in the form of a set of base texture patches and

how they are oriented on the image object, and an illumination correction. We have extensively evaluated our algorithm, have compared against baseline methods and state-of-the-art alternatives, and have presented applications to several computer graphics and vision tasks.

As for the future work, we find the idea of seamless transfer of information between image and model collections to be attractive and deserving further investigation. Marrying the two data forms can combine their complementary strengths, allowing applications not otherwise possible. In the short term, we would like to investigate better handling of narrow regions and edge effects, as well as transfer of base textures directly between shapes for shape-to-shape transfer, avoiding distortions due to unnecessary texture parameterization for each shape. More interestingly, we would like to better understand the importance of texture and its relations to object geometry, both for recognition/classification tasks and for object design. Texture can carry important information about the object style [165] (e.g., fabric patterns can help classify modern versus baroque chairs) or function [166] (e.g., wooden grain can indicate hard surfaces, while leather can indicate soft cushioned surfaces). Last but not least, we believe that the ability to generate realistic textured models and illumination maps can be further exploited for various design tasks and for generating on-demand training data for different learning applications.

In Chapter 5, we presented a novel optimization formulation for joint material and illumination estimation from collections of Internet images when different objects are observed in varying illumination conditions sharing coupled material and/or illumination observations. We demonstrated that such a linked material-illumination observation structure can be effectively exploited in a scalable optimization setup to recover robust estimates of material (both diffuse and specular) and effective environment maps. The estimations can then be used for a variety of compelling and photo-realistic image manipulation and object insertion tasks.

The proposed method opens up several interesting future directions to pursue. In this work, we manually curated the photo sets to test our approach. While collecting datasets with star structure for the observation matrix is not so difficult – one can

search with single keywords, especially for iconic designs – gathering more data linked with tighter connections (*e.g.,* loops, full matrix, etc.) is more challenging. One option would be to harvest user annotations and existing links (like in Houzz or Pinterest websites) to collect such data. Such data sets can open up new material-illumination estimation pipelines – this is particularly exciting as we can update our estimates in an incremental way leading to simultaneous improvement of all the associated estimates. Another interesting direction would be to improve the environment map estimates – one option would be to additionally use data priors to project the estimates environment maps to some data-driven manifold of environment maps measurements. Finally, an interesting future direction is to use the material and illumination estimates to improve geometry and pose estimates by refining correspondences by (partially) factoring our shading and illumination effects.

In Chapter 6, we presented a data-driven learning framework for obtaining a joint shape space linking 2D sketches, garment (2D sewing patterns and material properties) and human body specifications, and 3D draped garment shapes in the context of garment design. The learned shape space enables a novel multimodal design paradigm that allows users to iteratively create complex draped garments without requiring expensive physical simulations at design time. We also utilize the learned space to formulate an optimization that allows designed garments to be retargeted to a range of different human body shapes while preserving the original design intent. We evaluated our method, both quantitatively and qualitatively, in different usage scenarios and showed compelling results.

Currently, our method does not handle pose variation. We assume the garments are designed for a body at a fixed target pose. In the future, we plan to expand the shape space by also regressing across common body pose variations. In Figure 7.1, we show preliminary results in this direction. We augment our dataset by considering an additional degree of freedom for body pose that interpolates between different arm configurations. Then, given the input sketches in the top row, we infer the draped garments and body shape and pose parameters shown in the bottom row. These results indicate that by augmenting the dataset, learning pose variations is possible.

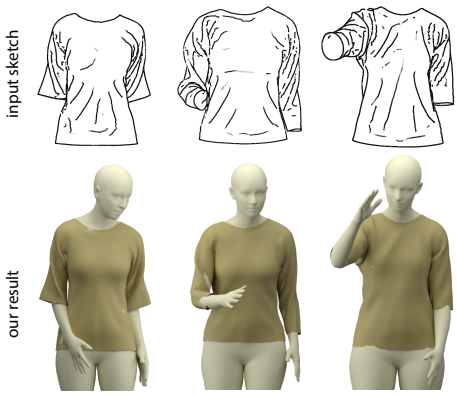


Figure 7.1: By augmenting our training dataset, we show preliminary results our method inferring body pose.

The challenge, however, is how to decide which key poses to include to effectively sample the configuration space. We would also like to explore the effectiveness of the joint neural network learning in other design contexts involving multimodal inputs in the future.

As we are in the middle of a fast-changing era of computer science, new methodologies and new tools continue to emerge rapidly and reshape the landscape of computer graphics research. In this thesis, we develop data-driven optimization methods to reinvestigate classical problems, i.e., reconstruction of a dynamic scene. We establish a fully automated pipeline with carefully designed building blocks to extract texture and illumination from a 2D image. We train neural networks to fit the mapping between different domains, and use neural networks as a part of the objective function in an optimization problem to make the formulation computationally effective. For the pipeline and workflow proposed, rely on exisiting fundamental building blocks such as image feature extraction, data dimension reduction, gradient descent optimization, particle-based physical simulation, etc. The improvement for these building blocks may propagate to our work and further improve our performance, for example, a neural network-based method for better image correspondence, or a more powerful solver for large-scale least square fitting so that the approach in Chapter 5 can be performed on an Internet-scale dataset and more exciting problems may be revealed. On the other hand, the methods proposed here may also serve as tools in many research fields for data extraction or generation, such as a large

object repository with realistic texture and illumination which may, in turn, improve the accuracy of a neural network classifier. This mutual synergy encourages us to continue to explore this direction.

Bibliography

[1] Zhengdong Zhang, Arvind Ganesh, Xiao Liang, and Yi Ma. TILT: transform invariant low-rank textures. *IJCV*, 99(1):1–24, 2012.

[2] Shan Yang, Tanya Ambert, Zherong Pan, Ke Wang, Licheng Yu, Tamara Berg, and Ming C. Lin. Detailed garment recovery from a single-view image. *CoRR*, abs/1608.01250, 2016.

[3] Tuanfeng Y. Wang, Pushmeet Kohli, and Niloy J. Mitra. Dynamic sfm: Detecting scene changes from image pairs. *Symposium on Geometry Processing 2015*, 2015.

[4] Tuanfeng Y. Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas Guibas, and Niloy J. Mitra. Unsupervised texture transfer from images to model collections. *ACM Trans. Graph.*, 35(6):177:1–177:13, 2016.

[5] Tuanfeng Y. Wang, Tobias Ritschel, and Niloy J. Mitra. Joint material and illumination estimation from photo sets in the wild. In *Proceedings of International Conference on 3DVision (3DV)*, 2018. selected for oral presentation.

[6] Tuanfeng Y. Wang, Duygu Ceylan, Jovan Popovic, and Niloy J. Mitra. Learning a shared shape space for multimodal garment design. *ACM Trans. Graph.*, 35(6):177:1–177:13, 2018.

[7] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[8] Joao Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 1071–1076. IEEE, 1995.

[9] Charles William Gear. Multibody grouping from motion images. *International Journal of Computer Vision*, 29(2):133–150, 1998.

[10] Amit Gruber and Yair Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–707. IEEE, 2004.

[11] Amit Gruber and Yair Weiss. Incorporating non-motion cues into 3d motion segmentation. In *Computer Vision–ECCV 2006*, pages 84–97. Springer, 2006.

[12] Konrad Schindler, David Suter, and Hanzi Wang. A model-selection framework for multibody structure-and-motion of image sequences. *International Journal of Computer Vision*, 79(2):159–177, 2008.

[13] Kemal E Ozden, Konrad Schindler, and Luc Van Gool. Multibody structure-from-motion in practice. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(6):1134–1141, 2010.

[14] Hanbyul Joo, Hyun Soo Park, and Yaser Sheikh. Map visibility estimation for large-scale dynamic 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1122–1129, 2014.

[15] René Vidal and Richard Hartley. Motion segmentation with missing data using powerfactorization and gpca. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–310. IEEE, 2004.

[16] Allen Y Yang, Shankar R Rao, and Yi Ma. Robust statistical estimation and segmentation of multiple subspaces. In *Computer Vision and Pattern*

Recognition Workshop, 2006. CVPRW'06. Conference on, pages 99–99. IEEE, 2006.

[17] Shankar Rao, Roberto Tron, Rene Vidal, and Yi Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(10):1832–1845, 2010.

[18] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.

[19] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[20] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012.

[21] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *International journal of computer vision*, 96(1):1–27, 2012.

[22] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] Mariano Jaimez, Christian Kerl, Javier Gonzalez-Jimenez, and Daniel Cremers. Fast odometry and scene flow from rgb-d cameras based on geometric clustering. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3992–3999. IEEE, 2017.

[24] Joao Fayad, Chris Russell, and Lourdes Agapito. Automated articulated structure and 3d shape recovery from point correspondences. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 431–438. IEEE, 2011.

[25] Anastasios Roussos, Chris Russell, Ravi Garg, and Lourdes Agapito. Dense multibody motion estimation and reconstruction from a handheld camera. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 31–40. IEEE, 2012.

[26] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[27] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[28] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015.

[29] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014.

[30] Moos Hueting, Maks Ovsjanikov, and Niloy Mitra. Crosslink: Joint understanding of image and 3d model collections through shape and camera pose variations. *ACM Trans. Graph (Proc. SIGGRAPH Asia)*, 34(6), 2015.

[31] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proc. ICCV*, pages 2686–2694, 2015.

[32] Joseph J. Lim, Aditya Khosla, and Antonio Torralba. FPM: fine pose parts-based model with 3d CAD models. In *ECCV*, pages 478–493, 2014.

[33] Noa Fish*, Melinos Averkiou*, Oliver van Kaick, Olga Sorkine-Hornung, Daniel Cohen-Or, and Niloy J. Mitra. Meta-representation of shape families. *ACM SIGGRAPH*, 2014.

[34] Hao Su, Qixing Huang, Niloy J. Mitra, Yangyan Li, and Leonidas Guibas. Estimating image depth using shape collections. *SIGGRAPH*, 2014.

[35] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *arXiv preprint arXiv:1604.00449*, 2016.

[36] Qixing Huang, Hai Wang, and Vladlen Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Trans. Graph.*, 34(4):87:1–87:10, July 2015.

[37] Yunhai Wang, Minglun Gong, Tianhua Wang, Daniel Cohen-Or, Hao Zhang, and Baoquan Chen. Projective analysis for 3d shape segmentation. *ACM Trans. Graph.*, 32(6):192:1–192:12, November 2013.

[38] Natasha Kholgade, Tomas Simon, Alexei A. Efros, and Yaser Sheikh. 3D object manipulation in a single photograph using stock 3D models. *ACM Trans. Graph.*, 33(4), 2014.

[39] Hao Su, Fan Wang, Li Yi, and Leonidas Guibas. 3d-assisted image feature synthesis for novel views of an object. *arXiv preprint arXiv:1412.0003*, 2014.

[40] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE CVPR*, pages 1538–1546, 2015.

[41] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *CoRR*, abs/1511.06702, 2015.

[42] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *EG-STAR*, 2009.

[43] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. *Comp. Vis. Sys.*, 1978.

[44] Stephen R Marschner and Donald P Greenberg. Inverse lighting for photography. In *Color and Imaging Conf.*, pages 262–5, 1997.

[45] Berthold K. P. Horn and Michael J. Brooks, editors. *Shape from Shading*. MIT Press, 1989.

[46] Konstantinos Rematas, Chuong Nguyen, Tobias Ritschel, Mario Fritz, and Tinne Tuytelaars. Novel views of objects from a single image. *TPAMI*, 2016.

[47] Li Shen, Ping Tan, and Stephen Lin. Intrinsic image decomposition with non-local texture cues. In *CVPR*, 2008.

[48] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. Graph.*, 33(4):159, 2014.

[49] Tom Haber, Christian Fuchs, Philippe Bekaer, Hans-Peter Seidel, Michael Goesele, Hendrik P Lensch, et al. Relighting objects from image collections. In *CVPR*, 2009.

[50] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Two-shot SVBRDF capture for stationary materials. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 34(4):110:1–110:13, 2015.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[52] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep lambertian networks. In *ICML*, 2012.

[53] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. *PAMI*, 2015.

[54] Takuya Narihira, Michael Maire, and Stella X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015.

[55] Jian Shi, Yue Dong, Hao Su, and Stella X Yu. Learning non-lambertian object intrinsics across shapenet categories. *arXiv:1612.08510*, 2016.

[56] Carlo Innamorati, Tobias Ritschel, Tim Weyrich, and Niloy J. Mitra. Decomposing single images for layered photo retouching. *Computer Graphics Forum (Proc. Eurogr. Symp. on Rendering)*, 36(4):15–25, 2017.

[57] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Tinne Tuytelaars, and Luc Van Gool. Natural illumination from multiple materials using deep learning. *arXiv:1611.09325*, 2016.

[58] Simon Gibson, Toby Howard, and Roger Hubbold. Flexible image-based photometric reconstruction using virtual light sources. *Comp. Graph. Forum*, 20(3):203–14, 2001.

[59] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *PAMI*, 25(3):290–300, 2003.

[60] Alexandros Panagopoulos, Dimitris Samaras, and Nikos Paragios. Robust shadow and illumination estimation using a mixture model. In *CVPR*, pages 651–8, 2009.

[61] Jean-François Lalonde, Alexei A Efros, and Srinivasa G Narasimhan. Estimating natural illumination from a single outdoor image. In *CVPR*, pages 183–190, 2009.

[62] G. Oxholm and K. Nishino. Shape and reflectance estimation in the wild. *PAMI*, 2015.

[63] Stephen Lombardi and Nishino. Reflectance and illumination recovery in the wild. *PAMI*, 2016.

[64] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-based rendering*. Springer Science & Business Media, 2008.

[65] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. *SIGGRAPH*, 1998.

[66] Ron O Dror, Thomas K Leung, Edward H Adelson, and Alan S Willsky. Statistics of real-world illumination. In *CVPR*, 2001.

[67] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 2003.

[68] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3d shapes. *ACM Trans. Graph.*, 30(4), 2011.

[69] Chuong H. Nguyen, Oliver Nalbach, Tobias Ritschel, and Hans-Peter Seidel. Guiding image manipulations using shape-appearance subspaces from co-alignment of image collections. *Computer Graphics Forum (Proc. Eurographics 2015)*, 34(2), 2015.

[70] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan C Russell, Josef Sivic, et al. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3762–3769. IEEE, 2014.

[71] Tuanfeng Y. Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas Guibas, and Niloy J. Mitra. Unsupervised texture transfer from images to model collections. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 35(6), 2016.

[72] Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley Osher. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86, 2011.

[73] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Material memex: automatic material suggestions for 3d objects. *ACM Transactions on Graphics (TOG)*, 31(6):143, 2012.

[74] Kang Chen, Kun Xu, Yizhou Yu, Tian-Yi Wang, and Shi-Min Hu. Magic decorator: Automatic material suggestion for indoor digital scenes. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 34(6), 2015.

[75] James T Kajiya. The rendering equation. In *ACM SIGGRAPH*, 1986.

[76] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH*, 2001.

[77] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *Proc. SIGGRAPH*, 2001.

[78] Erum Arif Khan, Erik Reinhard, Roland W Fleming, and Heinrich H Bülthoff. Image-based material editing. *ACM Trans. Graph.*, 2006.

[79] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3sweep: Extracting editable objects from a single photo. *ACM Trans. Graph.*, 32(6), 2013.

[80] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3D object manipulation in a single photograph using stock 3d models. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33(4), 2014.

[81] K Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth. Automatic scene inference for 3d object compositing. *ACM Trans. Graph.*, 33(3), June 2014.

[82] Optitext fashion design software. `https://optitex.com/`, 2018. Accessed: 2018-03-30.

[83] Marvelous designer. `https://www.marvelousdesigner.com`, 2018. Accessed: 2018-03-30.

[84] Floraine Berthouzoz, Akash Garg, Danny M. Kaufman, Eitan Grinspun, and Maneesh Agrawala. Parsing sewing patterns into 3d garments. *ACM SIGGRAPH*, 32(4):85:1–85:12, July 2013.

[85] Emmanuel Turquin, Marie-Paule Cani, and John F. Hughes. Sketching garments for virtual characters . In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 175–182, August 2004.

[86] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual Garments: A Fully Geometric Approach for Clothing Design. *CGF*, 2006.

[87] C. Robson, R. Maharik, A. Sheffer, and N. Carr. Context-aware garment modeling from sketches. *Computers and Graphics (Proc. SMI 2011)*, pages 604–613, 2011.

[88] Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Begault, Laurence Boissieux, and Marie-Paule Cani. Sketching folds: Developable surfaces from non-planar silhouettes. *ACM TOG*, 34(5):155:1–155:12, November 2015.

[89] Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. Bendsketch: Modeling freeform surfaces through 2d sketching. *ACM TOG*, 36(4):125:1–125:14, July 2017.

[90] Minchen Li, Alla Sheffer, Eitan Grinspun, and Nicholas Vining. Foldsketch: Enriching garments with physically reproducible folds. *ACM TOG*, 37(4), 2018.

[91] Xiaoguang Han, Chang Gao, and Yizhou Yu. Deepsketch2face: a deep learning based sketching system for 3d face and caricature modeling. *ACM Transactions on Graphics (TOG)*, 36(4):126, 2017.

[92] Nobuyuki Umetani. Exploring generative 3d shapes using autoencoder networks. In *SIGGRAPH Asia 2017 Technical Briefs*, page 24. ACM, 2017.

[93] Huamin Wang. Rule-free sewing pattern adjustment with precision and efficiency. *ACM Transactions on Graphics (TOG)*, 37(4):53, 2018.

[94] Bin Zhou, Xiaowu Chen, Qiang Fu, Kan Guo, and Ping Tan. Garment modeling from a single image. *CGF*, 32(7), 2013.

[95] Moon-Hwan Jeong, Dong-Hoon Han, and Hyeong-Seok Ko. Garment capture from a photograph. *Comput. Animat. Virtual Worlds*, 26(3-4):291–300, May 2015.

[96] Xiaowu Chen, Bin Zhou, Feixiang Lu, Lin Wang, Lang Bi, and Ping Tan. Garment modeling with a depth camera. *ACM SIGGRAPH Asia*, 34(6):203:1–203:12, October 2015.

[97] R. Daněřek, E. Dibra, C. Öztireli, R. Ziegler, and M. Gross. Deepgarment: 3d garment shape estimation from a single image. *CGF Eurographics*, 36(2):269–280, May 2017.

[98] Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. Sensitive couture for interactive garment modeling and editing. *ACM TOG*, 30(4):90:1–90:12, July 2011.

[99] Yuwei Meng, Charlie CL Wang, and Xiaogang Jin. Flexible shape control for automatic resizing of apparel products. *Computer-Aided Design*, 44(1):68–76, 2012.

[100] Tsz-Ho Kwok, Yan-Qiu Zhang, Charlie CL Wang, Yong-Jin Liu, and Kai Tang. Styling evolution for tight-fitting garments. *IEEE transactions on visualization and computer graphics*, 22(5):1580–1591, 2016.

[101] Aric Bartle, Alla Sheffer, Vladimir G. Kim, Danny M. Kaufman, Nicholas Vining, and Floraine Berthouzoz. Physics-driven pattern adjustment for direct 3d garment editing. *ACM SIGGRAPH*, 35(4):50:1–50:11, July 2016.

[102] Remi Brouet, Alla Sheffer, Laurence Boissieux, and Marie-Paule Cani. Design preserving garment transfer. *ACM TOG*, 31(4):36:1–36:11, July 2012.

[103] P. Guan, L. Reiss, D. Hirshberg, A. Weiss, and M. J. Black. DRAPE: DRessing Any PErson. *ACM SIGGRAPH*, 31(4):35:1–35:10, July 2012.

[104] Weiwei Xu, Nobuyuki Umentani, Qianwen Chao, Jie Mao, Xiaogang Jin, and Xin Tong. Sensitivity-optimized rigging for example-based real-time clothing synthesis. *ACM TOG*, 33(4):107:1–107:11, July 2014.

[105] D. Pritchard and W. Heidrich. Cloth motion capture. *CGF Eurographics*, 22(3):263–271, 2003.

[106] Volker Scholz, Timo Stich, Michael Keckeisen, Markus Wacker, and Marcus Magnor. Garment motion capture using color-coded patterns. *CGF Eurographics*, 24(3):439–447, 2005.

[107] Ryan White, Keenan Crane, and D. A. Forsyth. Capturing and animating occluded cloth. *ACM SIGGRAPH*, 26(3), July 2007.

[108] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM SIGGRAPH*, 27(3):99:1–99:9, August 2008.

[109] Tiberiu Popa, Qingnan Zhou, Derek Bradley, Vladislav Kraevoy, Hongbo Fu, Alla Sheffer, and Wolfgang Heidrich. Wrinkling captured garments using space-time data-driven deformation. *CGF Eurographics*, 28(2):427–435, 2009.

[110] Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F. O'Brien. Example-based wrinkle synthesis for clothing animation. *ACM Trans. Graph.*, 29(4):107:1–107:8, July 2010.

[111] Damien Rohmer, Marie-Paule Cani, Stefanie Hahmann, and Boris Thibert. Folded Paper Geometry from 2D Pattern and 3D Contour. In Sylvain Lefebvre Nick Avis, editor, *Eurographics 2011 (short paper)*, pages 21–24, April 2011.

[112] A. Neophytou and A. Hilton. A layered model of human body and garment deformation. In *Int. Conf. on 3D Vision*, volume 1, pages 171–178, Dec 2014.

[113] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael Black. ClothCap: Seamless 4D clothing capture and retargeting. *ACM SIGGRAPH*, 36(4), 2017.

[114] Marco Paladini, Alessio Del Bue, Marko Stosic, Marija Dodig, Joao Xavier, and Lourdes Agapito. Factorization for non-rigid and articulated structure using metric projections. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2898–2905. IEEE, 2009.

[115] Changchang Wu. Visualsfm: A visual structure from motion system. *URL: http://homes. cs. washington. edu/˜ ccwu/vsfm*, 9, 2011.

[116] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[117] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[118] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[119] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1362–1376, 2010.

[120] Michal Jancosek and Tomás Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3121–3128. IEEE, 2011.

[121] Fabian Langguth Simon Fuhrmann and Michael Goesele. MVE - A Multi-View Reconstruction Environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, 2014.

[122] Mathias Rothermel, Konrad Wenzel, Dieter Fritsch, and Norbert Haala. Sure: Photogrammetric surface reconstruction from imagery. In *Proceedings LC3D Workshop, Berlin*, 2012.

[123] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. Registration of point cloud data from a geometric optimization perspective. In *Symposium on Geometry Processing*, pages 23–31, 2004.

[124] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[125] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[126] Daniel Kuettel, Matthieu Guillaumin, and Vittorio Ferrari. Segmentation propagation in imagenet. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *ECCV*, pages 459–473, 2012.

[127] Matthieu Guillaumin, Daniel Küttel, and Vittorio Ferrari. Imagenet auto-annotation with segmentation propagation. *IJCV*, 110(3):328–348, 2014.

[128] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *PAMI*, 33(5), 2011.

[129] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH)*, 33(4), 2014.

[130] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image Melding: Combining inconsistent images using patch-based synthesis. *(Proc. SIGGRAPH)*, 31(4):82:1–82:10, 2012.

[131] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proc. ACM CGI*, pages 341–346. ACM, 2001.

[132] Bruno Vallet and Bruno Lévy. What you seam is what you get. Technical report, INRIA - ALICE Project Team, 2009.

[133] Melinos Averkiou, Vladimir G. Kim, and Niloy J. Mitra. Autocorrelation descriptor for efficient co-alignment of 3d shape collections. *Computer Graphics Forum*, 35(1):261–271, 2016.

[134] David R Hunter. Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, pages 384–406, 2004.

[135] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J. Mitra. Interactive images: Cuboid proxies for smart image manipulation. *ACM Transactions on Graphics*, 31(4):99:1–99:11, 2012.

[136] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proc. ICCV*, pages 1520–1528, 2015.

[137] Paul Debevec. Uffizi environment map, 2004. http://www.pauldebevec.com/Probes/.

[138] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *PAMI*, 2015.

[139] Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. Anisotropic spherical gaussians. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 32(6), 2013.

[140] Paul Green, Jan Kautz, Wojciech Matusik, and Frédo Durand. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proc. i3D*, 2006.

[141] Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 28(5):133, 2009.

[142] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. A sparse parametric mixture model for BTF compression, editing and rendering. *Comp. Graph. Forum*, 30(2):465–73, 2011.

[143] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33(4):101, 2014.

[144] Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for CNN: Viewpoint estimation in images using cnns trained with rendered 3D model views. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[145] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[146] X. Wang, David F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015.

[147] Fred E Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied optics*, 1965.

[148] Yu-Ting Tsai and Zen-Chung Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph.*, 25(3):967–76, 2006.

[149] Yusuke Tokuyoshi. Virtual spherical gaussian lights for real-time glossy indirect illumination. *Comp. Graph. Forum*, 34(7):89–98, 2015.

[150] Marios Papas, Wojciech Jarosz, Wenzel Jakob, Szymon Rusinkiewicz, Wojciech Matusik, and Tim Weyrich. Goal-based caustics. *Comp. Graph Forum (Proc. Eurographics)*, 30(2):503–511, 2011.

[151] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proc. EGSR*, pages 195–206, 2007.

[152] James Arvo, Kenneth Torrance, and Brian Smits. A framework for the analysis of error in global illumination algorithms. In *Proc. SIGGRAPH*, pages 75–84, 1994.

[153] C. CZhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550—-60, 1997.

[154] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[155] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004.

[156] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM SIGGRAPH Asia*, 34(6):248:1–248:16, October 2015.

[157] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter V. Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[158] NVIDIA flex. `https://developer.nvidia.com/flex`, 2018. Accessed: 2018-03-30.

[159] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):848–855, July 2003.

[160] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[161] torchvision. `https://pytorch.org/docs/master/torchvision/`, 2018. Accessed: 2018-03-30.

[162] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[163] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[164] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2008.

[165] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3D furniture models. *ACM SIGGRAPH*, 34(4), 2015.

[166] Ruizhen Hu, Oliver van Kaick, Bojian Wu, Hui Huang, Ariel Shamir, and Hao Zhang. Learning how objects function via co-analysis of interactions. *ACM SIGGRAPH*, 35(4), 2016.