Capturing, Processing, and Synthesizing Surfaces with Details

Sema Berkiten

A Dissertation

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

by the Department of

Computer Science

Adviser: Szymon Rusinkiewicz

September 2016

Abstract

Geometry acquisition and processing have become increasingly popular in computer graphics and vision, with demand for high-quality models driven by advances in 3D printing, realistic real-time renderings of 3D avatars in video games, digital libraries for historical objects etc. In this thesis, we focus on techniques to produce and process detailed geometry including acquisition of real world objects, processing and fusing the captured data, and synthesizing new surfaces from existing ones.

First, we summarize a 2D acquisition technique called photometric stereo to capture high resolution surface details. As a validation step, we solve a misalignment problem for photometric datasets. After the dataset is validated, the surface normals can be computed using one of the photometric stereo algorithms. In order to decide which algorithm to use, we present a synthetic photometric benchmark to evaluate various algorithms for different scenarios.

To produce a detailed surface in 3D, we propose an approach to combine a rough 3D geometry with detailed normal maps obtained from different views. We begin with unaligned 2D normal maps and a rough 3D geometry, and automatically align each normal map to the 3D geometry. Next, we map the normals onto the surface, correct and seamlessly blend them together. We then optimize the geometry to produce a high-quality 3D model.

Next, we introduce a semi-automated system to convert photometric datasets into geometry-aware non-photorealistic illustrations of surface details that obey the common conventions of epigraphy (black-and-white archaeological drawings of inscriptions). This system is composed of rectification of the surface normals to correct camera perspective, segmentation of the inscriptions from the background, classification of the inscription based on carving technique, and stylization of the inscriptions in various styles.

Finally, we present an algorithm for realistically transferring surface details (specifically, displacement maps) from existing high-quality 3D models to simple shapes that may be created with easy-to-learn modeling tools. Our key insight is to use metric learning to find a

combination of geometric features that successfully predicts detail-map similarities on the source mesh, and use the learned feature combination to drive the detail transfer.

Acknowledgements

I would like to thank my advisor Szymon Rusinkiewicz for all his guidance and support during my time at Princeton. I also would like to thank my thesis committee members – Szymon Rusinkiewicz, Thomas Funkhouser, Adam Finkelstein, Hao Li, and David Dobkin.

Thanks all the current and previous members of Tiggraph and Princeton Computer Graphics Group, especially Xinyi Fan, Maciej Halber, Ohad Fried, Thiago Pereira for their support and suggestions.

I thank Carla Schroer and Mark Mudge from Cultural Heritage Imaging for introducing me Reflectance Transformation Imaging (RTI) and giving me motivation in my earlier projects.

I would like to thank our archaeologist collaborators Kate Liszka and Bryan Kraemer, for their valuable ideas and providing manual digital epigraphy, and Xinyi Fan who made the project in Chapter 4 possible.

I also would like to thank all the collaborators – Maciej Halber, Justin Solomon, Chongyang Ma, and Hao Li – for their help to accomplish the work in Chapter 5.

My graduate studies at Princeton University and this thesis were supported by Princeton Fellowship, the NSF grants CCF-1027962, IIS-1012147, and IIS-1421435, as well as Intel ISTC-VC.

Finally, I would like to thank my family for their endless support.

To my family...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The demand for high-quality 3D models has been increasing in the last few decades. For instance, thanks to the advances in 3D printing technology it is possible to duplicate objects with under millimeter precision – see Figure 1.1a. Now the bottleneck is to capture or create the level of detail with the same precision. Video gaming industry is also getting better everyday. They can render highly complicated scenes in real-time, however again the bottleneck is the content creation. As shown in Figure 1.1b, 3D characters with sophisticated surface details are popular in video games and filming industry and they are all created manually by 3D artists. Depending on the complexity of the model, nowadays the cost of a single 3D model varies from a few hundred dollars to a thousand dollars. Another field which benefits from the advances in digital capturing techniques is cultural heritage. Many museums have been digitizing their artefacts by using 3D scanners, gigapixel computational imaging, or various other photometric techniques. Figure 1.1c demonstrates an example of reflective transformation imaging (RTI) which is a photometric technique to capture surface colors as well as some geometric information such as surface normals. RTI data later can be used for various applications such as re-lighting the object, enhancing the details, changing the reflectance etc. Additionally, thanks to the advances in digital data storage technologies

(a) 3D Printing (b) 3D Avatar (c) Cultural Heritage

Figure 1.1: **(a)** Microfabrica additive manufacturing technology can achieve under millimeter precision. **(b)** A 3D avatar with fine details. **(c)** RTI representation used by cultural heritage practitioners for preservation and visualization.

in the last few decades, the cost and space to store terabytes of data decreased dramatically. In turn, we can store captured surfaces in high resolution at low costs.

We can summarize some of the challenges encountered by the industries mentioned above in three categories: (i) *capturing* high-resolution data in 2D and/or 3D, (ii) *processing* the high-resolution data, and (iii) creating (*synthesizing*) new high-resolution 3D models.

Capturing. Both 2D and 3D scanning technologies are faced with a trade-off between object size and resolution. For example, among many 3D scanning technologies as summarized in Chapter 3, laser scanners such as NextEngine and Matter & Form have higher resolutions around 0.1-0.5 mm however they have limited maximum object sizes around 20x20x20 cm. In addition, to reconstruct a complete 3D model one need to combine all the scans from different views and this will result in the loss of some details depending on the accuracy of the alignments.

Among photometric techniques, Light Stage [113] uses a variant of 2D active scanning technique, photometric stereo, to estimate both diffuse and specular surface normals. This setup can capture faces with a resolution high enough to capture skin pores. However, it

is a fixed-sized setup which makes it harder to adapt to capture objects with varying sizes. Moreover, It is expensive to build such a light dome and it occupies a lot of space.

Nehab et al. [130] propose a unified acquisition system that captures both rough 3D geometry via a temporal stereo triangulation scanner and normal maps via photometric stereo. Since they share the same camera for both captures, 3D positions and the normal map are perfectly aligned. However, again it is a fixed-sized setup, limits the size of the object to be captured and it is not flexible to adapt for different object sizes.

In this thesis, Chapters 2 and 3 propose solutions to problems in capturing high-quality surfaces by *optimizing* the existing technology and *combining* various technologies to achieve higher quality than either one alone.

Processing. Extracting meaningful information from either 2D or 3D data has been studied extensively. For example, line drawings from RGB images or 3D surfaces are one of the most popular problems [86, 180, 77, 46, 80, 96]. However, previous work suffers from low signal-to-noise ratios. For rough surfaces with shallow depths or noisy colors, those methods usually fail to differentiate between the surface bumps and the actual information. In Chapter 4, we *convert* the wealth of information from the captured high-resolution data into interpretable concise representations.

Synthesizing. The number of 3D models available online is tremendous however the number of high-quality 3D models freely available is severely limited. That is why, designers are usually forced to generate models by hand for many applications such as realistic renderings for furniture catalogs, 3D environments for CGI, video games etc. 3D artists usually resort to highly sophisticated modeling tools such as ZBrush [136] and physical simulators to generate realistic looking 3D contents. However, those tools require highly trained artists and even so it takes a lot of manual work to create a single model. There are various works in the literature focusing on content generation using procedural modeling [71] however they are usually tailored for specific tasks such as generating trees, terrains etc. and

cannot be applied to different types of objects. On the other hand, Mertens et al. [120], Lu et al. [109], and Chen et al. [38] try to transfer surface colors from one model to another with some geometric supervision to generate 3D models with a similar appearance to the existing ones, which can be adapted to apply to surface details. However their methods are not sophisticated enough to fully capture the relationship between the surface details and the underlying geometry. In Chapter 5, we propose *creating* new contents with realistic appearance by transferring surface details from existing 3D models with high-quality details to low resolution 3D models.

1.1 Contributions

We summarize the main contributions of this thesis on surfaces with details in four categories:

- **Capturing and Validation:** One of the most common method of capturing high resolution surface details is active shape from shading (photometric stereo). To obtain the highest quality normal maps from those datasets, however, we need to validate the data and align them if necessary. We leverage having multiple images by constructing a spanning tree and aligning images with similar illumination to each other as opposed to aligning each image to a single target image. We also analyze the behavior of various photometric stereo algorithms under varying different factor and propose an iterative least squares approach for general purpose photometric stereo problem. (Chapter 2),

- **Data Fusion:** 3D scans are usually sufficient to capture overall coarse-scale shape of a surface, however difficult to obtain at extremely high resolutions. On the other hand, normal maps are easily obtained at high resolution using photometric stereo even though they do not provide reliable information about the coarse shape. In order to obtain high resolution 3D models, we propose a pipeline to combine a coarse geometry from the 3D scans and fine surface details from the normal maps. The advantage of

this pipeline is that it does not require any initial alignment or resolution/precision compatibility between the 3D scan and the normal maps (Chapter 3),

- **Illustration:** Normal maps contain a wealth of information about fine surface details which is why we also use them to analyze nearly flat surfaces with fine details and convert them into geometry-aware non-photorealistic (NPR) illustrations. Specifically, we focus on rock faces with ancient carved or scraped off inscriptions which are located in a desert, constantly under strong sunlight, making capture challenging. Another challenge is to attenuate the noise caused by surface roughness and weathering which causes undesirable results for previous NPR techniques. We propose segmenting the foreground to discard the noise by using a graph-cut algorithm. We find that using both the surface geometry and the color information gives better results than using only one of them, (Chapter 4),

- **Realism:** Even though there are many 3D models available online for free they usually lack of fine details which give realistic appearance to them. We propose using few available data with fine details to increase the number of such models by transferring its details to a wide variety of low-resolution 3D models. Our key insight is to use metric learning to find a relationship between the coarse-scale geometry and the fine details, and use this relationship to drive the detail transfer from the high-resolution to low-resolution 3D models, (Chapter 5).

Chapter 2

High-fidelity Surface Detail Acquisition

This chapter focuses on *capturing* and recovering surface orientations from the variations of the observed intensity of an object under different illuminations – also known as *photometric stereo*. Since this approach provides per pixel orientations, by using a digital-SLR camera we can obtain scans with higher resolution than other 3D scanning technologies such as LIDAR, Kinect, and laser scanners. In this chapter, we first visit the photometric stereo algorithm, then propose a method to validate and correct any misalignments in the photometric datasets, and finally analyze the effects of several factors such as ambient illumination, inter-reflections, material etc. on various photometric stereo algorithms including a new general purpose photometric stereo algorithm using iterative least squares.

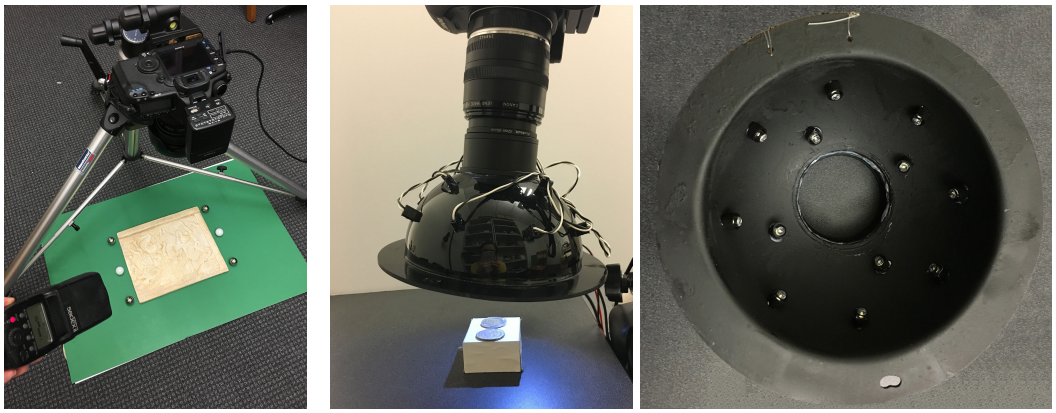
2.1 Photometric Stereo

Photometric stereo is an active range acquisition technique which requires multiple images of the same object from a fixed camera position under varying light directions. The simplest formulation of the surface orientation recovery from images assumes that the surface is Lambertian, meaning that the outgoing radiance is the same from all directions. Given a single light source in each image and a fixed camera position, we can formulate this problem as:

$$I_{ij} = \rho_i N_i \cdot L_j \tag{2.1}$$

where I_{ij} is the observed intensity at pixel i under light direction L_j, and N_i and ρ_i are the surface normals and albedo at pixel i, respectively. Given at least three images, I, and their associated light directions, L, we can solve for N and ρ with the linear least squares approach. However, in most cases more images are necessary for better estimation because of outliers such as self-shadowing, specularity, and image noise. If the light directions, L, are unknown, one can estimate them by solving a Generalized-Bas-Relief (GBR) ambiguity problem with the constraint of surface integrability [190].

Capture Setup. All the photometric datasets presented in this thesis are captured with a setup similar to the method proposed by Toler-Franklin et al. [161]. Specifically, a series of digital-SLR images of the object are captured with a fixed camera position, and a hand-held flash is moved around the object to obtain different light directions as shown in Figure 2.1a. A black cloth can be used to shade the object and the camera for outdoor captures. Several mirror (chrome) and white diffuse spheres, which will be used to estimate the light directions and intensities on each image later, are placed next to the object. Also, one image with no flash is taken at the beginning of each capture session, which will be subtracted from all other images with flash to remove ambient lighting as much as possible. We also built a small light dome 15 centimeters in diameter with 12 LED lights evenly spaced. As shown in



(a) Hand-held flash (b) Light Dome

Figure 2.1: Photometric capture setups with **(a)** a hand-held flash and **(b)** a mini light dome with fixed light positions (**Left:** the light dome is attached to the camera lens, **Right:** light positions inside the dome).

Figure 2.1b, the dome can be attached to the camera lens and very portable. The advantage of a light dome is that since the light positions are fixed we can calibrate for the lights once and use them for all datasets, however we can use this mini light dome for small objects (a few centimeters wide) only. After the photometric datasets are captured, we use a variant of the photometric stereo algorithm [182] to estimate surface normals and the true albedo from those images – see the iterative least squares approach in Section 2.3.5 for details.

A Taxonomy of Photometric Stereo. Photometric stereo (PS) algorithms can be divided into two categories: calibrated and uncalibrated, based on the availability of information about the *illumination conditions*. In the same fashion, one can call algorithms (un)calibrated based on other factors such as BRDF, shadow maps, and possibly geometric priors; however we use lighting conditions as the main factor to categorize the algorithms, because it has the biggest effect on the accuracy of the recovered surface orientations, as we discuss in Section 2.3. We can list the sub-factors that shape the PS algorithms can also be broadly categorized: illumination, material, outliers, and geometry. We sub-categorize

the (un)calibrated PS algorithms based on those factors bellow. A more comprehensive survey can be found in [5].

2.1.1 Calibrated Photometric Stereo (CPS)

For calibrated photometric stereo, acquisition setup can be designed for different purposes with following options:

- Fixed light positions with respect to the camera. For example, light stage [177] or light domes [117] with built in lights and camera(s) on it are commonly used techniques to capture photometric data. They are robust in terms of acquisition of the light directions/positions and repeatability. Drawbacks of these setups are: they are expensive, hard to maintain, take a lot of space, and are not easily scalable.

- Hand-held light sources, with calibration objects such as chrome spheres used to estimate light position and diffuse white spheres to measure the intensity and the color of the light source [162]. This setup supports mobile light sources as opposed to the fixed lights in light domes.

- The light emitters can be modeled as point sources or as area lights. While LED lights usually act as a point light sources, a flash of a camera behaves as an area light source that emits almost parallel light rays.

Illumination. Instead of a single light source per frame, Brostow et al. [31] used 3 colored light- red, green, blue- to perform fast dense-normal recovery on video sequences. Abrams et al. [2] used the outdoor day-light images by using the sun as their light source which requires calibrations based on the time of the day and the day of the year in a similar way to Ackermann et al. [4]. Other than a point or a directional light source, Ma et al. [114] proposed using a polarized spherical gradient illumination so that they only need 4 images to recover the surface normals.

Material. Hertzmann and Seitz [73] used a known geometry (sphere) in the scene with the same material of the object to create a normal lookup table. However, in real-life it is hard to find or create these calibration objects. By assuming that most objects are composed of a few fundamental materials, Goldman et al. [60] proposed solving the normals for a few BRDFs with varying weights per pixel. Tan et al. [157] assumed that the surface is composed of micro-facets instead of assuming Lambertian surface per pixel to recover higher resolution normals than the original images.

Outliers. Drew et al. [49] handled shadows and specularity by using radial basis function interpolation. Chandraker et al. [36] proposed an algorithm to segment different shadows regions. On the other hand, Wu et al. [184] assumed that the outliers are sparse so they optimize for a low-rank observation matrix plus a sparse outlier matrix.

Geometry. Basri and Jacobs [12] suggested to use a 9D linear subspace using spherical harmonics to map the per-pixel surface properties of convex Lambertian objects.

2.1.2 Uncalibrated Photometric Stereo (UPS)

Uncalibrated photometric stereo does not require any special setup other than a fixed camera position and varying illumination.

Illumination. Zhou and Tan [195] solved uncalibrated PS by adding some constraints on the light configuration even though the light positions are not known.

Material. Alldrin et al. [8] assumed that the number of distinct albedos is low, and they solve the generalized bas-relief (GBR) ambiguity by minimizing the entropy of the albedo distribution. Under the Lambertian assumption, Favaro and Papadhimitri [54] exploited maximal Lambertian reflections to solve the GBR ambiguity. Shi et al. [146] clustered

the pixels to leverage the strong correlation between the surface normals and the intensity profiles of pixels.

Outliers. Tan et al. [158] leveraged the non-Lambertian reflectance components to solve the Generalized-Bas-Relief (GBR) ambiguity. Similarly, Chandraker et al. [37] used the inter-reflections on the non-convex surfaces to solve the GBR ambiguity. Mukaigawa et al. [128] proposed using the photometric linearization by introducing a classification-based criterion for specular and diffuse reflections, and cast and attached shadows.

Geometry. Basri et al. [13] solved for the GBR ambiguity by using only a few images under the assumption of observed objects' being convex. Queau et al. [138] used total variation regularization to solve for a piece-wise smooth surface. Yuille and Snow [190] showed that the surface integrability constraint reduces the GBR ambiguity to a problem with 3 unknowns.

2.2 Alignment of Images with Varying Light Directions

In this section, we focus on alignment of photometric datasets (high-resolution images taken with a fixed camera under different light directions). Although the camera position is largely fixed, there might be some misalignment due to perturbations to the camera or to the object, or the effect of optical image stabilization, especially in long photo shoots. Based on our experiments, we observe that feature-based techniques outperform intensity-based ones for this problem. We found that SIFT [107] and SURF [14] provide very reliable features for most cases. For feature-based approaches, one of the main problems is the elimination of outliers, and we solve this problem using the RANSAC framework. Furthermore, we propose a method to automatically detect the transformation model between images. The datasets that we focus on have around 10-100 images, of the same scene, and in order to take advantage of having many images, we explore a graph-based approach to find

the strongest connectivities between images. Finally, we demonstrate that our alignment algorithm improves the results of photometric stereo by showing normal maps before and after alignment.

2.2.1 Introduction

Image alignment is one of the first steps for most computer vision and image processing algorithms. Image fusion, image mosaicing, creation of panoramas, object recognition/detection, photometric stereo and enhanced rendering are some of the examples in which image alignment is a crucial step. We can define the problem as the process of overlaying images of the same scene under different conditions, such as from different viewpoints, with different illumination, using different sensors, or at different times. In this section, we focus on the alignment of multiple images of the same scene under varying illumination.

In the literature, there are many works on image alignment, and most of them aim to solve a specific problem. For example, algorithms for medical image alignment mostly use intensity values of the image, while object recognition algorithms usually try to find some descriptive salient features. In short, the structure of the algorithm is shaped by the specifications of the problem itself.

We focus on aligning images of the same object under different light directions, keeping other conditions almost the same. This section presents the following contributions:

- Evaluation of which approaches to image alignment are most stable under different illumination,
- Progressive solution for more and more complex image transformations, to use the least general (and hence most robust) model necessary for good alignment,
- Simultaneous alignment of multiple images using spanning trees of image graphs,
- Demonstration of the effect of the image alignment on photometric stereo.

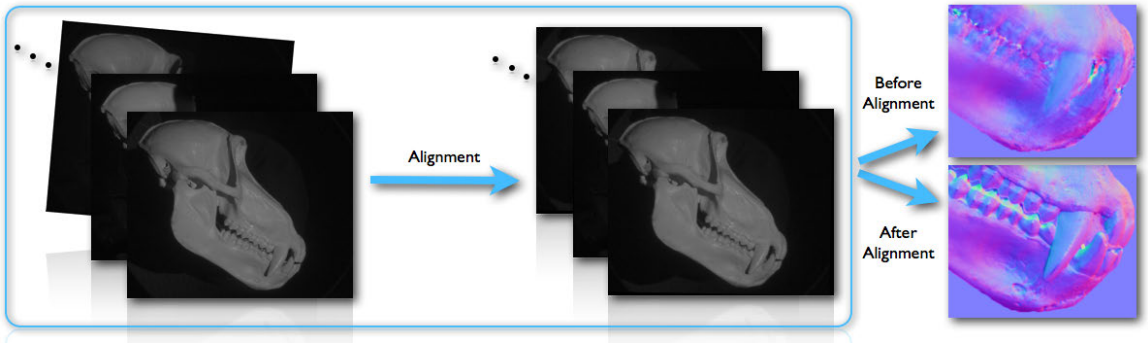


Figure 2.2: Misaligned images are aligned with our graph-based alignment method so that they can be used in various applications such as photometric stereo. **Left to Right:** images before and after alignment, and normal maps calculated from them.

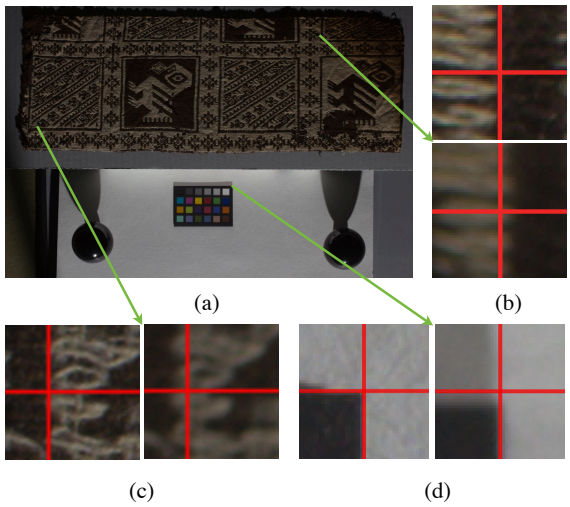


Figure 2.3: Effect of image stabilization (IS). **(a)** A sample image from the textile dataset. **(b), (c), (d)** Closeup images showing the effect of image stabilization, red lines are placed on the same locations to show the misalignments between two frames caused by IS.

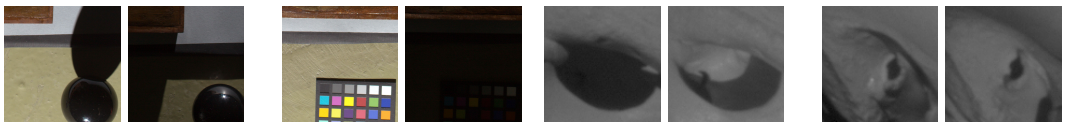


Figure 2.4: Closeup images from the panel dataset, **(a) and (b)**, and the skull dataset, **(c) and (d)**. Different light directions might cause different cast shadows (a, b, and c), moving specular highlights (a, and d), and different visible edges with different sharpness (all).

The image datasets that we focus on are usually used as inputs to other applications, such as polynomial texture mapping (PTM) [116], shape and detail enhancement [53], photometric stereo and enhanced rendering [115] etc. For example, PTM takes several images taken from the same view point but under different light directions, and constructs the coefficients of a bi-quadratic polynomial per texel. These coefficients are used later to reconstruct the surface color under varying light directions. Although the camera and the object in the capture setup should be kept still, sometimes there might be some misalignment between images because of perturbations to the camera/object in the scene or the effect of optical image stabilization. For example, Figure 2.3 shows some misalignments caused by image stabilization. Therefore, image registration is necessary to make sure that input images are well aligned.

However, many techniques in the literature are either only partially invariant to illumination change or not invariant at all [196]. For images taken under different light directions, the main problem is the non-linear illumination change, which is not trivial to deal with because different light directions may lead to cast shadows, moving specular highlights, local changes in brightness on the image, or even the loss of some information (e.g, if the light direction is perpendicular to a geometric edge on the object, it might not be recognizable on the image; or a texture edge might not be visible if it is in shadow), Figure 2.4 illustrates these problems. In early experiments, we observed that pixel-based methods such as normalized cross correlation, L2 difference of intensity values, pixel dot product etc. fail to find correct alignments for our datasets most of the time. It is because pixel-based methods rely on intensity values only, and these are very unreliable under non-linear illumination changes. That is why, in this section, we focus on feature-based methods.

In this section, we first summarize some image registration algorithms in the literature. Then we explain our approach to solve the matching problem for images taken under different illumination. Finally, we show results demonstrating the necessity of the image alignment on photometric stereo.

2.2.2 Related Work

Image registration (alignment) is transforming a source image to the coordinate system of the reference image. Images may be taken either at different times, from different views, under different lighting, or with different sensors. Researchers have been working on the image registration problem for decades to solve alignment problems in various fields such as medical imaging, remote sensing, image fusion, panorama, change detection, recognition etc.

Brown [34] published a very broad survey on image registration techniques in 1992. More recently, Zitova et al. published another survey paper in 2003 including more recent works on image alignment [196]. In 2006, a tutorial for image alignment and stitching was published by Szeliski [153].

The many algorithms for image alignment can be broadly categorized as pixel-based or feature-based. Pixel-based methods use intensity values directly: for example they may rely on normalized-cross-correlation and its variants, similarity measures using pixel dot products or L2 distance. Matching, fitting and validation steps are calculated simultaneously for a preselected transformation model [196].

For template matching, Kaneko et al. proposed the selective correlation coefficient, which is very similar to cross-correlation (CC), but it extracts a correlation mask-image differently than CC. They used increment sign correlation to extract the mask, and the mask is enhanced with four-pixel majority rule [84], [85]. Silveira et al. proposed a method for real-time visual tracking. They modeled the illumination change and image motion by solving a second-order optimization problem minimizing the intensity difference based on illumination and image motion models [147]. By using only the strongest image gradients with a pyramidal refinement strategy, Eisemann and Durand align flash and no flash images [51].

One of the well known pixel-based algorithms in computer vision and graphics is optical flow, proposed by Lucas and Kanade [110]. It assumes constant flow in a local neighborhood

and solves optical flow equations by least squares. Optical flow and its variants have been used to solve image alignment problems in various works [87], [94], [11], [18]. To find only translational misalignment, Ward proposed median threshold bitmaps in image pyramids for hand-held photographs with varying exposures [173].

Feature-based algorithms consist of three main blocks: salient feature extraction, feature matching, and estimation of the transformation. Various types of feature detection algorithms have been proposed throughout the years such as line, contour, and region detectors. However, salient feature points are easier to deal with than lines, contours or surfaces. The Harris corner detector [68] has been used for many years to detect corner-like points. Recently, feature descriptors became more popular because of their distinctive and invariant natures. The Scale-Invariant Feature Transform (SIFT) was proposed by Lowe in 2004; since then, it has been used widely because of its shift and scale-invariance and its distinctive descriptors [107]. Furthermore, Lowe showed that SIFT shows high performance on object recognition. Later on, Brown and Lowe used SIFT on unordered panorama images for stitching, together with the RANSAC framework for outlier elimination and a probabilistic model for verification [35]. Tang et al. similarly used a variant of SIFT with RANSAC to align medical microscopic sequence images [159]. It is shown in [171] that the same approach (SIFT and RANSAC) works well for multi-modal image registration-aligning infrared to visible images. Bay et al. proposed a similar but faster approach to SIFT called Speeded Up Robust Features (SURF) [14]. Winder et al. introduced another configuration to compute salient feature descriptors and presented comparisons to SIFT [178], [179].

For feature matching, the least efficient method is brute force comparison of L2 distance between feature descriptors. If the number of features is large, such as for object recognition, k-d trees or similar data structures can be used to speed up the search [35]. Even if a highly distinctive descriptor is used, there might be some false matches called outliers, and a randomized framework to eliminate outliers called RANSAC is often preferred because it works with up to fifty percent outliers [57]. Mikolajczyk et al. published comparisons

of steerable filters, PCA-SIFT, differential invariants, spin images, SIFT, complex filters, moment invariants, and cross-correlation for different types of interest regions [122]. Also an intensive survey on local invariant feature descriptors can be found in [165].

As a hybrid of feature- and pixel-based methods, normalized cross-correlation is used with a Harris-Laplacian detector in [192] to make NCC rotation and scale invariant.

2.2.3 Single-Target Image Alignment

Image alignment is a correspondence problem of mapping one image to another. In this subsection, we explain the single-target alignment algorithm in three parts: feature extraction, feature matching, and transformation models.

Feature Extraction.

Although invariance to geometric deformation, and shift and scale invariance to illumination are explored well in previous work, there is no feature extraction algorithm which is invariant to non-linear illumination changes such as varying light direction. We therefore explore a number of feature detection algorithms on our datasets, with the aim of discovering which ones perform best in the presence of large-scale lighting changes.

Normalized Harris Corner Detector. Intuitively, we expect that many corner-like features in an object's texture can be precisely localized regardless of illumination. We therefore begin by exploring the performance of the Harris corner detector. Unfortunately, while this detector is invariant to shifts in brightness (since it is based on the gradient), it is not invariant to multiplicative changes in brightness. As a result, this detector is highly sensitive to illumination, and extracts more corner points in bright regions. We therefore explore a *normalized* Harris corner detector, based on a structure tensor that is normalized by the local

image intensity:

$$C = \frac{\begin{bmatrix} \sum_W g(i,j) I_x(\mathbf{w})^2 & \sum_W g(i,j) I_x(\mathbf{w}) I_y(\mathbf{w}) \\ \sum_W g(i,j) I_x(\mathbf{w}) I_y(\mathbf{w}) & \sum_W g(i,j) I_y(\mathbf{w})^2 \end{bmatrix}}{\sum_W g(i,j) I(\mathbf{w})^2} \tag{2.2}$$

where W is the interest window around each pixel, (i, j) are pixel offsets within the window, g is a 2D Gaussian filter, **w** is the global pixel location, and $I(\mathbf{w})$ is the image intensity at that pixel.

SIFT. The Scale-Invariant Feature Transform (SIFT) is a scale- and rotation-invariant local feature detector proposed by Lowe [107], and used for many computer vision problems since then. There are several works showing its robustness in the literature [122], [134], [91]. SIFT consists of four main steps: scale-space extrema detection, accurate key-point localization, orientation assignment, and calculation of a key point descriptor.

In the first step, candidate interest points are detected by finding extrema over scale and image space. The second step refines the interest points' locations by fitting a 3D quadratic function to the scale-space function, which is approximated by the Taylor expansion. In the third step, each feature point is assigned a dominant orientation, which is detected by finding the maximum of the local orientation histogram around the feature point. In the final step, a descriptor vector is calculated for each feature point. This descriptor is built by concatenating local orientation histograms of 4x4 sub-windows of a 16x16 window around the feature point. Further details of SIFT can be found in [107].

SURF. The Speeded-up robust feature (SURF) detector proposed by Bay et al. [14], is also a scale and rotation invariant local feature detector which is partially inspired by SIFT. Its main purposes are to be computationally less expensive and to be as distinctive as SIFT. In order to speed-up the computations, SURF uses integral images and box-filter approximations to the second derivative of Gaussian.

Method selected by progressive algorithm	**Ground-truth error for:** Translation	Tr+Rot	Tr+Rot+Sc	Affine	Projective
Translation	**0.13**	0.25	0.33	0.38	15.7
Translation+Rotation	6.26	**0.23**	0.23	0.44	4.7
Translation+Rotation+Scale	17.7	2.64	**0.43**	0.56	4.7
Affine	14.5	2.94	1.13	**0.47**	2.1
Projective	18.4	2.5	1.01	0.55	**0.23**

Table 2.1: Average ground-truth alignment errors in pixels for image collections in Figure 2.9. The progressive algorithm selects the method shown at left in each row and it automatically picks the transformation type giving the lowest error.

Comparison of Feature Extraction Methods. In Figure 2.5, the repeatability of features on a typical dataset is explored for the three different types of feature detectors above. The repeatability ratio between the target (t) and the source (s) images is formulated as follows:

$$R = \frac{\text{Number of } fp_i\text{'s}}{\text{Total number of features}} \tag{2.3}$$

where $fp_i = \{\text{feature pair i} \mid \sqrt{2} \geq \| \ (x_s(i), y_s(i)) - (x_t(i), y_t(i)) \ \|\}$.

It is obvious that, despite the improvements of normalization, the Harris corner detector is outperformed by SIFT and SURF on this dataset, and indeed we find that this behavior generalizes across many different types of images. As a result, we use SIFT as the feature extractor throughout the rest of the section.

Feature Matching

The naive way to match feature points is to calculate the Euclidean distance between feature descriptors and compare them. Another approach is to calculate the ratio of Euclidean distance to the closest neighbor and to the second closest neighbor and eliminate the ones

which have high ratio, because a low ratio implies that it is a correct correspondence with high probability. For tasks which require searching a large feature database, such as object recognition, special data-structures such as k-d tree or search strategies are used. For example, [107] uses the Best-Bin-First (BBF) search algorithm, which returns the closest neighbor with high probability. The BBF is a variant of k-d tree, using a priority queue based on closeness, and it terminates after a specific number of neighbors are explored.

In this subsection, we use exhaustive search on Euclidean distances, but we apply a constraint on the distance between the locations of feature points on the image plane, because we have assumed the deformation between images will be small, so corresponding points cannot be very far away from each other (we use 128 pixels as threshold).

Transformation Models

To calculate a transformation matrix for given feature mappings, the transformation type has to be selected first. We consider several classes of transformations, of increasing numbers of degrees of freedom (DOF): translation only (2 DOF), translation and rotation (3 DOF), similarity (4 DOF), affine (6 DOF), and projective or homography (8 DOF).

The projective transformation model is the most generic among all, which is why it is commonly used, especially if the type of deformation is not given a priori. However, this generality comes at a price, since incorrect correspondences and even small errors in feature point localization can result in significant errors in the transformation.

Progressive Transformation Model. Selection of the transformation model determines the number of degrees of freedom, and thereby the constraints on the transformation matrix. When the deformation type on input images is not known in advance, the projective transformation model is commonly chosen. However, when there is only a translational difference between two images, for example, the estimated transformation will be more erroneous

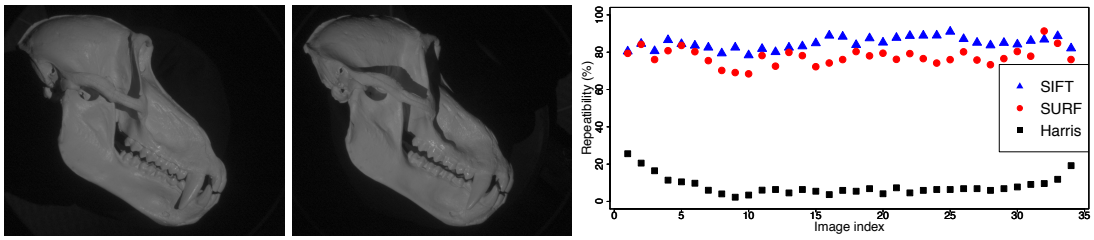
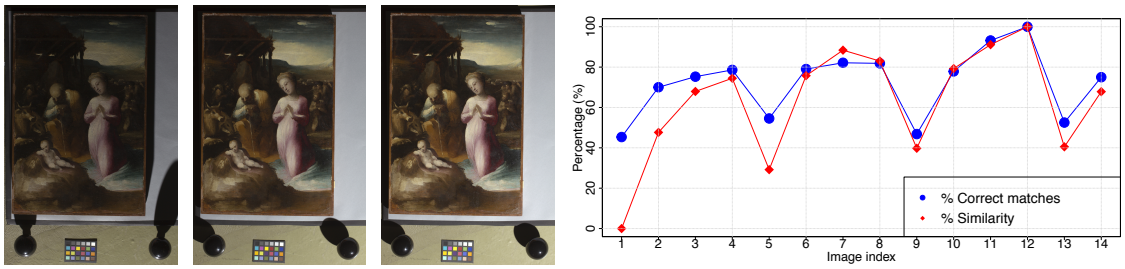
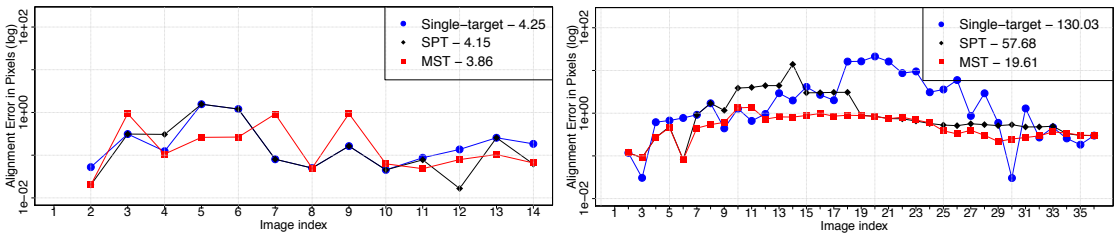


Figure 2.5: **Left:** Two images from the skull dataset. **Right:** Repeatability ratios of different feature detectors on this dataset. (First image is selected as the target image.)



(a) Image 1 (b) Image 11 (c) Image 12 (d)

Figure 2.6: (a-c) Images 1, 11, and 12 in the dataset. (d) Percentage of the correct feature matches, as well as image similarities (inverse of normalized and scaled L2 differences of low-resolution images) for the panel dataset. Image 12 is the target image.



(a) Panel (b) Skull

Figure 2.7: Alignment errors in pixels (in log scale) for each image. Single-target alignment, SPT, and MST for panel (a) and skull (b) datasets. The sum of alignment errors are indicated in the legend.

than it would be if a translational model were selected, because of localization error on the feature extraction step.

In order to obtain maximally accurate and robust estimates of the transformation, we propose a *progressive* model to select a deformation type automatically, similar to the model selection algorithm proposed by [163]. In particular, we use the following algorithm inspired by RANSAC:

Algorithm for progressive transformation

---

repeat

- For each transformation model, from translational to projective:
  - Estimate the transformation matrix by fitting the least squares on feature matches in a RANSAC framework, store the number of correspondences which agree with the estimated matrix (inlier).

until the maximum number of inlier matches is smaller than the predefined threshold, τ:

Select the final transformation: the one with the maximum number of inlier correspondences.

The reason why this approach works is the presence of localization error on the feature point locations. Otherwise, if all feature points were localized perfectly, we would expect to get the same results for different transformation models. In Table 2.1, average single-target alignment errors for different types of transformation models are shown. We observe that, in a majority of cases, the progressive algorithm picks the transformation model giving the lowest ground-truth pixel error.

2.2.4 Graph-Based Image Alignment

So far we have explored single-target image alignment, but aligning all images in the dataset to one target image does not give good results for all cases. In particular, it is inevitable to get badly aligned results when images in the dataset have a lot of geometrical variations and

less texture, such as images in the skull dataset, because different light directions will result in different shadows, varying local gradients, and thereby different local image features. For example, Figure 2.6 shows the number of correct matches between one fixed target image (Image 12) and the remaining images in the dataset. We observe that the more similar the illumination condition, the higher the number of correct matches.

Based on experiments such as this, we conclude that it is possible to leverage the availability of multiple images by not attempting alignment to a single target. Instead, we formulate multi-image alignment as finding a *spanning tree* in a graph in which each vertex V represents an image and each edge E represents similarity. For efficiency, it is desirable to have the edge weights easily computable. Fortunately, as shown in Figure 2.6, simple L2 image difference (on down-sampled images) is highly correlated with the number of correct feature matches. We may therefore use low-resolution L2 similarity as a proxy for feature similarity when constructing our graph. Also, we observed that this proxy gives higher weights for image pairs with close light positions.

One way of visualizing the similarities between images quantitatively is Laplacian Eigenmaps [15]. This is a spectral clustering technique used to solve the dimension reduction problem. It takes an affinity matrix whose elements are the Euclidean distance between corresponding images. Its optimal solution is the eigenvector corresponding to the second smallest eigenvalue. Figure 2.8 shows the eigenvectors corresponding to the second and third smallest eigenvalues for the skull dataset of 36 images.

Minimum Spanning Tree vs Shortest Path Tree

Once we have constructed an image similarity graph G, we are left with the task of extracting a subset of edges on which to perform full image alignment. (Of course, including more than the minimum subset of edges and combining the results with least squares could reduce error, but also increases sensitivity to bad correspondences, and is not explored in this thesis.) We compare two algorithms for extracting a spanning tree:

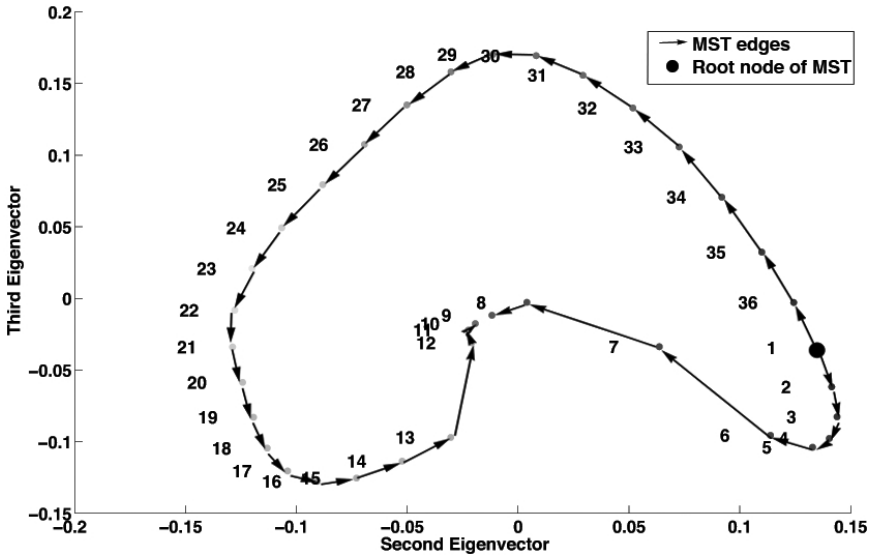


Figure 2.8: Spectral analysis on image similarities for the skull dataset. Each dot represents an image and located based on the second and third eigenvectors of the Laplacian Eigenmaps for the skull dataset (36 images). MST-edges are shown with black arrows, the first node is the root node of the MST.

- A minimum spanning tree (MST) is the one which has the smallest total weight among all possible spanning trees of G. We use Prim's algorithm [137] to construct the MST.

- A shortest path tree (SPT) with root vertex v is the spanning tree of G containing all shortest paths from v to other vertices. Dijkstra's algorithm [48] can be used to construct an SPT from a connected graph.

Figure 2.7 shows alignment errors (on a logarithmic scale) for single-target alignment, SPT, and MST. We observe that graph-based methods typically outperform single-target alignment. Total alignment errors are indicated in the legends, and for the skull dataset we observe that the total single-target alignment error (130 pixels, 3.7 pixels per image) is far from acceptable. On the other hand, MST gives roughly 0.5 pixel error per image for the same dataset. For the panel dataset, alignment results for each approach are very close to each other (about 0.3 pixel error per image) because this dataset is feature-rich, texture-rich, and the object has a flat surface. On the other hand, the skull dataset is more challenging

and MST outperforms both single-target and SPT in this dataset. Although it is not very clear that MST always performs better than SPT, we use MST for the tests in the Results section, because MST outperforms both single-target and SPT in the most challenging case that we have (skull dataset).

2.2.5 Results

In Figure 2.9, the test results on different datasets for three different algorithms (single-target alignment with homographic transformation type, single-target alignment with progressive transformation, graph-based alignment with progressive transformation type) are demonstrated. Three test cases are formed: i) datasets with ground truths: images on the original datasets are perfectly aligned and images are manipulated randomly (transformation types, from translation only to homographic, and manipulation amounts are set randomly); ii) datasets with gold standard: the original datasets have misalignments and they are aligned by manually selecting correspondences to calculate the gold standard; iii) experimenting on the number of key-points on each image in the dataset. In the table, the third column is the average number of key-points on images in a dataset, the fourth column is the pixel resolutions of images in the dataset, and the subsequent columns show average and maximum alignment errors for the three algorithms. Alignment error for a given estimated matrix (E) and the ground truth matrix (G) is calculated as follows:

$$e = \frac{1}{4} \sum_{i=1,2,3,4} ||G^{-1}Ep_i - p_i|| \tag{2.4}$$

where the p_i are the four corner points of the image. The reason for calculating the alignment error on the corner points is that the maximum error will appear on the corners for the transformation types that we consider.

We observe that single-target alignment fails when the projective transformation model is assumed for large collections. On the other hand, the progressive model mostly gives

				Alignment Errors (in pixels)						Running-Time (in secs)		
				Projective		Progressive		MST-Progressive		Key calc	Single-	Graph-
Dataset	#Images	#Points	Resolution	Avg.	Max	Avg.	Max	Avg.	Max	/image	target	based
Ground Truth: Perfectly aligned images are randomly manipulated (ranging from only translation to homographic manipulations) to create test sets.												
	36	430.7	1190x980	**45.47**	113	**2.197**	13.4	**0.5957**	1.87	1.7	3.9	2.2
	19	1748	2184x1456	**43.19**	245.3	**6.676**	32.87	**1.351**	3.081	4.1	9.6	8.2
	49	1073	1728x2592	**71.04**	284.4	**0.5062**	1.507	**0.6031**	1.571	1.8	7.4	10.0
	64	572.6	2184x1456	**27.73**	186.4	**0.5868**	4.11	**0.9675**	2.304	1.4	8.3	8.1
	36	1153	1312x864	**29.25**	88.99	**0.6456**	2.612	**0.8497**	2.281	1.8	9.5	7.1
	42	488.4	1024x1024	**29.66**	102.5	**2.309**	15.47	**1.655**	6.451	5.6	6.2	5.2
	34	986.7	2184x1456	**47.74**	286.5	**3.46**	38.84	**1.698**	3.732	14.1	19.0	14.7
Gold Standard: it is acquired by manually aligning images												
	47	873.3	1728x2592	**105.6**	406.4	**0.8966**	2.096	**0.9756**	1.73	1.6	7.5	7.9
	4	1206	2592x1728	**13.61**	46.68	**1.148**	2.143	**1.277**	2.658	27.53	1.15975	1.10477
Experiment on the number of key-features												
	68	4569	2799x1868	**87.63**	499.3	**0.5423**	1.333	**0.6249**	1.56	3.2	51.6	66.0
	68	489.3	2799x1868	**104.6**	484.3	**1.052**	5.943	**0.9628**	3.335	5.2	11.6	11.1

Figure 2.9: Test results for different datasets.

successful results. The graph-based method using MST and progressive transformation leads to the alignment error of 0.5-1.5 pixels on average, while the single-target alignment algorithm results in alignment error of 0.5-7 pixels on average. And the graph-based algorithm works robustly on very challenging datasets such as the first and second collections in Table 2.9. While the maximum error is unacceptable on most of the datasets when the single-target alignment is used, the graph-based method gives acceptable results.

We also show the effect of the alignment on photometric stereo in Figure 2.2 by demonstrating the normal map before and after alignment. On the first three examples, it is obvious that the details cannot be recovered with photometric stereo if they are not well-aligned. And in the last row, we observed that misalignment causes embossing effect on moderately

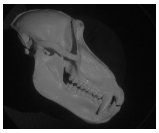
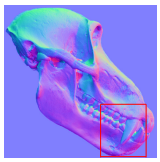
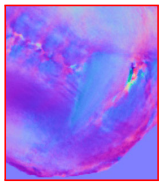
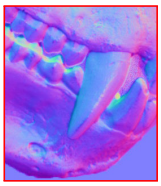
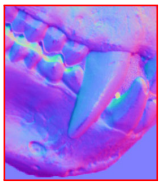
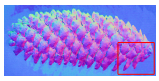
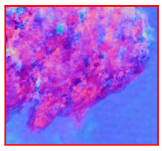
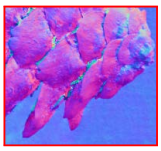
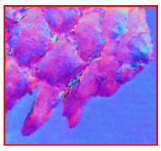
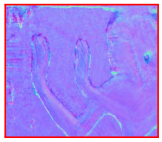
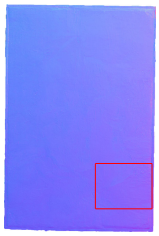
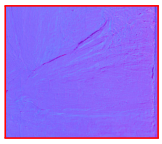
RGB Image	Ground Truth	**Misaligned**	**Ground Truth**	**Aligned**
				
		MSE: 0.004		MSE: 0.0003
				
		MSE: 0.026		MSE: 0.01
				
		MSE: 0.005		MSE: 0.0009
				
		MSE: 0.001		MSE: 0.0002

Table 2.2: Effect of alignment on photometric stereo. The first column shows example images from each dataset, the normal maps calculated on either ground truth or gold standard dataset are shown on the second column. The third, fourth and the fifth columns demonstrate close-up images of the normal maps calculated on misaligned, ground truth and aligned datasets respectively. Mean square errors (MSE) between normal maps for aligned and ground truth datasets and between normal maps for misaligned and ground truth datasets are indicated under the close-up images.

flat surfaces. Also, mean-square errors (MSE) between ground truth and each normal map are indicated under each closeup image. We observed that image alignment improves the MSE at least ten times.

2.3 An RGBN Benchmark

In the previous section, we presented an algorithm to verify the captured photometric dataset and correct it if misalignments are present. Now we will focus on algorithms that use photometric datasets and how to evaluate them.

In addition to the photometric stereo, a variety of algorithms use photometric datasets (an object or scene captured with fixed camera but varying illumination). Evaluating these algorithms is frequently challenging because of the lack of ground truth on the one hand, and insufficiently realistic and varied synthetic datasets on the other. In this section, we present a synthetic benchmark for applications and justify it by comparing to real-life objects and their rendered models. Additionally, we introduce a system that allows the user to create scenes by combining arbitrary 3D models, materials, and light configurations. The system outputs physically-based renderings as well as dense ground-truth maps of quantities such as normals, height map, BRDF specifications, and albedo. We present a number of synthetic datasets, and we provide a few photometric datasets of real-life objects. This section demonstrates that real objects can be simulated well enough so that the conclusions about accuracy drawn from our synthetic datasets match those based on real objects. It also demonstrates a use case for this RGBN benchmark: the evaluation of photometric stereo algorithms. We investigate the causes of errors in several photometric stereo algorithms, and propose a variant that iteratively estimates shadowing.

2.3.1 Introduction

Photometric datasets have been used by various applications in computer vision and graphics including BRDF acquisition [60], photometric stereo [181], (non)photo-realistic rendering, illumination representations such as polynomial texture mapping [117], linear image subspaces [12], and digitization of cultural heritage. Despite the fact that photometric datasets are used in a variety of fields, the literature lacks a realistic benchmark with ground truth.

This may be due to the fact that it is tedious to create a benchmark that (1) is based on real-world data with ground truth, and (2) contains several versions of the same scene, with one parameter (e.g., noise level, lighting configuration, material, etc.) varied while everything else is kept constant.

We propose a synthetic but realistic (physically-based) and validated (against ground-truth data) benchmark for evaluation of photometric algorithms. Each benchmark dataset includes a collection of images of a scene under different lighting, together with ground-truth surface normals, height map, BRDF attributes, and albedo. The datasets are organized into "experiments," in which one parameter of the scene is varied while others are kept constant.

There are two major benefits to using synthetic data for our benchmark. First, the ground truth available with each dataset is *real* ground truth. An alternative would have been to use real images, but obtain "ground truth" using, say, a 3D scanner. This has a major problem: the precision and the calibration of the scanning system, alignment of partial scans to form the whole model, and alignment of the 3D model to the photometric data are all sources of error in the "ground truth" geometry or normal maps. Even with high-precision scanners, the user of the benchmark might not know whether the biases introduced in the "ground truth" might favor some algorithms over others.

The second major benefit of using synthetic data is that it is possible to isolate the effects of noise, lighting, materials, etc. By conducting "experiments" in which individual factors vary in isolation, users of the benchmark are able to make more precise conclusions about the relative strengths and weaknesses in the algorithms they are evaluating.

There are also two potential drawbacks to using synthetic data, and we have attempted to mitigate them. First, the renderings might not be realistic. To accurately simulate as many light-transport phenomena as possible, we generate the benchmark images using Mitsuba Renderer [79]. Mitsuba Renderer is a physically-based rendering tool that supports various surface materials, subsurface-scattering, glittering effects etc. It also takes all global illumination into account during rendering.

Even with a physically-based renderer, there may be a concern that conclusions about algorithms based on synthetic images might not match those based on real-world images. Therefore, to justify the use of synthetic data, we conduct an experiment comparing the error patterns of a photometric stereo algorithm on several real objects and their 3D scanned models in Section 2.3.4.

In addition to the benchmark itself, we provide a flexible but simple user interface that allows users to create their own datasets while controlling variables such as light configuration, light intensity, noise level, amount or characteristics of ambient lighting. Finally, we explore one of the applications of the benchmark: evaluation of various photometric stereo algorithms. Photometric stereo is the general name of the algorithms that try to estimate surface properties, most notably normal maps and surface color, from multiple images under varying illumination.

In summary, our main contributions in this section are:

- A new calibrated photometric stereo algorithm which estimates the surface normals and the albedo with an iterated least squares method by updating "pixel confidence" values (Section 2.3.5);
- An example usage of the proposed benchmark: evaluation of various photometric stereo algorithms (Section 2.3.5).

2.3.2 Related Work

In the literature, there are several publicly available photometric datasets under different illuminations. For example, [7], [73], [183], [4], [185], and [162] provide photometric datasets that they used in their work. However, there is no precise ground truth for these datasets to enable quantitative comparisons between algorithms.

The most relevant benchmark there exist is proposed by [1] . In their work, they present a robotic system to capture single-view images of an object under varying illuminations

as well as a 3D scan via a structured-light scanner. They use the 3D scans to generate the ground truth, however the 3D point cloud is not dense in order to form a high quality and accuracy depth map or normal map. Another disadvantage of this benchmark is the limited number of lights per view (20 LED lights in a planar arrangement) and it is not possible to expand the benchmark by other users.

Other related benchmarks are for intrinsic image algorithms, published by Grosse et al.[65] and Bell et al.[16]. The datasets in these benchmarks are a subset of all the photometric datasets we focus on this section. For intrinsic image estimations, they only need one image as the input data and to create ground truth Grosse et al.[65] captured 10 more images, while Bell et al.[16] used crowd-sourcing to obtain the ground truth. In summary, these benchmarks provide one image as input and *relative* shading and reflectance per scene, as opposed to the dense ground truth data for normal map, material, and height map provided by our benchmark.

For other applications in computer graphics/vision, there are several reputable benchmarks. For example, Scharstein and Szeliski published a comprehensive benchmark on two-frame stereo [144], and subsequently a number of works used their system and datasets to validate their work. A benchmark for 3D object segmentation was presented by [39], and [118] published a database for 2D image segmentation. One common property of all these benchmarks is that they are manually created. In the literature, there are not only real-world datasets which are manually labeled by users, but also some synthetic benchmarks. The most recent benchmark which proposes using synthetic datasets is for 3D surface reconstruction [19].

The importance of a well-prepared benchmark is undeniable. When a field lacks of a reference to compare different approaches, not only the users but also the researchers in the field have difficulties to determine the best approach/algorithm for their problems. As a result, some problems such as photometric stereo do not have consensus best-of-breed solutions, even though they are well-studied. We believe that a benchmark with high

precision will help us to understand the problem better and explore the reasons why some algorithms actually fail.

2.3.3 Benchmark Overview

Our system is designed to create realistic photometric data along with their corresponding ground truth from arbitrary 3D scenes. We use the Mitsuba Renderer [79] to produce photo-realistic renderings. We also capture a few real-world objects to use in justification of the synthetic datasets.

Synthetic Data

In this subsection, we introduce the process of synthetic dataset creation from a single viewpoint under different illuminations. Although our system is designed for single-view photometric data, it can be used for multi-view photometric data applications, such as [72], without any modification.

The production of the synthetic benchmark is composed of two parts: i) creation of an arbitrary scene, ii) producing the photometric data and ground truth.

User Interface. We provide a simple user interface to create scenes for our benchmark. Although general modeling packages, such as Blender, could be used for this, our interface is customized to produce photometric data: it provides fewer options than a package such as Blender, and it can create variations of the same scene automatically.

The user interface is customized to create arbitrary scenes from 3D models.The user can set each object in the scene, and assign either preset materials or her own material specifications (limited to the materials supported by the Mitsuba Renderer). These materials can be as simple as diffuse single-color albedo or as complex as human skin with both texture and subsurface scattering.

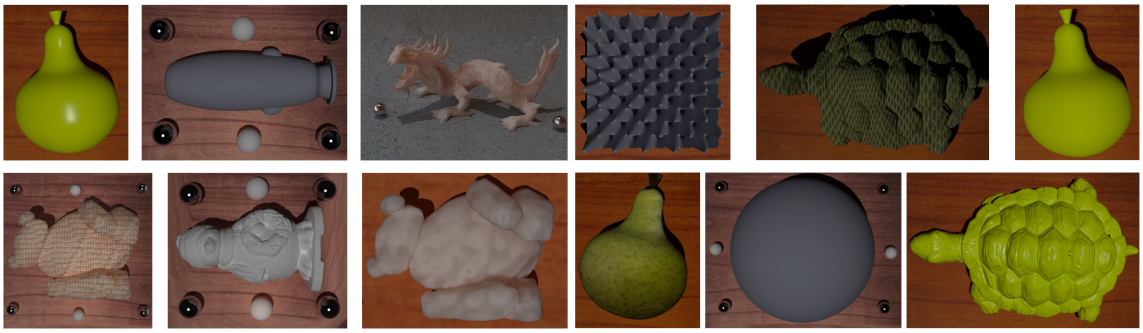
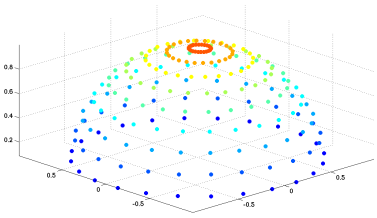


Figure 2.10: Some rendered scenes from the benchmark.

Images and Ground Truth Data. We produce two types of output: photometric data and ground truth data. Photometric data consists of physically-based renderings of the scene under various light directions are produced by Mitsuba Renderer [79].



A typical light configuration is shown on the left; however the user can turn on/off different light-rings and right/left hemispheres, as well as change the number of lights and the radius of the light dome.

While the resulting renderings include a great deal of global illumination complexity, see Figure 2.10., it is our goal to understand the extent to which that complexity affects the estimation of normals or other photometric quantities. For this reason, we output a set of "clean" ground truth images consisting of:

- **Normal Map:** Per pixel-surface normals are produced as RGB images, using an embedding that maps $[-1..1]$ to $[0..255]$ in each channel.
- **Depth Map:** Euclidean distance in 3D from the camera center to the surface for rays passing through each pixel is saved as floating-point EXR images.
- **BRDF:** Diffuse RGB albedo per pixel is rendered using the Mitsuba Renderer's utility functions. For non-Lambertian materials, the full BRDF specifications can be obtained from the scene files.

- **Light direction/position per image:** For a directional light model, light directions are saved in camera coordinates. For a point light source model, light positions are saved in camera coordinates.

Real-world Data

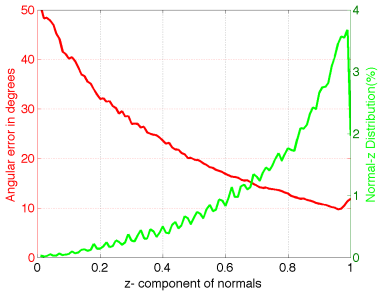
One potential concern about a synthetic benchmark is that it might not capture the full complexity of real-world images. Therefore, despite the limitations of scanned 3D data, we perform a real-world evaluation to indicate the plausibility of our synthetic benchmark. We captured several objects using the same setup as [162]: fixed camera on a tripod and a hand-held flash. We took around 36 images per object and one image without flash to compensate for the ambient lighting. We also scanned all the objects with a NextEngine laser scanner to obtain a 3D model, and use this to extract ground truth data.

2.3.4 Justification of the Synthetic Data

To verify that we can simulate a real-life object, we conducted an experiment that compares photometric stereo results of a real dataset to one created synthetically. First, we pick an object with a single color and relatively diffuse surface but fairly complicated geometry, as shown in Figure 2.11.a. We obtain photographs of the object, and obtain "ground truth" from a scanned 3D model. The latter is created by scanning the object with a NextEngine laser scanner, aligning partial scans with ICP, and reconstructing a final mesh with the Poisson surface reconstruction algorithm [89]. The resolution of the voxel size for surface reconstruction is 0.25 mm, and the image resolution is set accordingly to have a similar resolution. We would like to point out that because of the imperfections and smoothing effects in the reconstruction algorithm, the final 3D model lacks many of the high-frequency details of the real object.

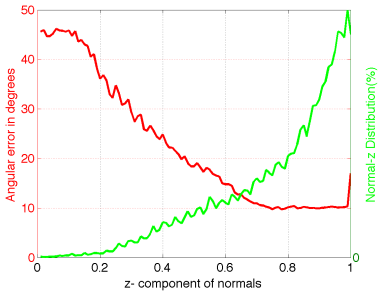
For material acquisition, a flat region of the object, which is the bottom of the object in this case, is photographed with an X-rite color checker as shown in Figure 2.12. A

(a) Real object (b) Blur image (c) Synthetic



(d) Real Object: 14.4

(e) Real (f) Smoothed (g) 3D model



(h) Synthetic Object: 13.1

Figure 2.11: **Verification of the synthetic dataset: (a)** Photograph of the real object, **(b)** Gaussian filtered image of (a), **(c)** Rendering of the 3D model of the object, which is scanned with a NextEngine laser scanner. **(e)**, **(f)**, and **(g)** are the normal map estimated from real photos, smoothed real normal map, and the ground truth respectively. **(d)** and **(h)** show the angular error distribution of the normal maps recovered from the real photographs and the synthetic renderings respectively. Normal maps are estimated by linear least squares with shadow/specularity thresholding, and average angular errors are indicated under each plot.

Figure 2.12: **Albedo acquisition of the Penguin:** An X-rite color-checkerboard is used for color calibration and albedo estimation.

color profile is extracted from the calibration board and the image is corrected accordingly. Assuming that the object has a single color, we average the color values on the flat region, which is manually selected, of the object. This way, a single color value is obtained for the Penguin. To determine the specularity and roughness of the object, we experiment with varying materials and parameters on the Mitsuba Renderer to get the most realistic image. We use the *roughplastic* material with the diffuse reflectance of *RGB(225,226,228)*, specularity of *0.4*, and *0.1* roughness.

A ground truth normal map of the object is created as follows: the normal map of the object is estimated from the real photographs, the 3D model is aligned to the normal map estimated from the real photographs by using the alignment algorithm in [20], and the normals of the 3D model are rendered in the aligned camera coordinates as shown in Figure 2.11.d, and f.

Since the 3D model does not have very high-frequency details because of the smoothing effect of the reconstruction, the estimated normal map of the real object is smoothed, as shown in Figure 2.11.e. Since we do not know how much the reconstruction introduces smoothing to the 3D scan, we determine the amount by comparing the rendered object in Figure 2.11.c to the real photographs smoothed with different amounts. In Figure 2.11.b the closest blurred photograph to the rendered image is shown. The plots in Figure 2.11.g and h show the angular errors in normal maps recovered from the real photographs (2.11.e) and the synthetic renderings (2.11.f). The average angular errors are comparable: 14.4 and 13.1 in degrees, respectively. The error distributions over the z- component of the ground truth normals also demonstrate a very similar pattern. Therefore, we conclude that we can create synthetic benchmarks which are comparable to real-life scenarios.

For the rest of the section, we will use only synthetic data for evaluations unless otherwise stated.

2.3.5 Evaluation of Photometric Stereo

In this subsection, we conduct various experiments on several photometric stereo algorithms. These experiments are designed to reveal the effects of various factors such as image noise, shadows, inter-reflections, uncompensated ambient illumination, material type, and number of images. We evaluate the following photometric stereo algorithms:

Calibrated Photometric Stereo (CPS):

1. (Lsq) We implemented a least-squares minimization with outlier rejection based on hard thresholds for shadows (40 in the range of $[0, 255]$) and specularity (254 in the range of $[0, 255]$) [181],

2. (RPCS) Low-Rank Matrix Completion and Recovery [184]: solves the problem of recovering a low-rank matrix with both missing and corrupted entries, which models all types of non-Lambertian effects such as shadows and specularities,

3. (Iter Lsq) Motivated by the sensitivity of the simple least-squares algorithm to shadow estimation, we propose an iterative weighted least-squares minization that repeatedly assigns per-pixel confidence values to each image (initially based the intensities in all observed images), computes normals and albedos using those confidences, then updates the confidences based on whether the pixel values agree with (Lambertian) re-renderings of the estimated normals and albedos.

4. (HSH) Hemispherical harmonics [49]: proposes a robust version of PTM [117] by identifying both specular and shadow pixels and then modeling their contribution, separately or together, using a radial basis function interpolation.

Uncalibrated Photometric Stereo (UPS):

1. (SCPS) Self Calibrating Photometric Stereo [146]: automatically determines a radiometric response function and resolves the generalized bas-relief ambiguity by

analyzing color/intensity profiles in the RGB and irradiance-time domains – we have tested on only a few cases,

2. (LDR) Lambertian diffuse reflectance [54]: solves the generalized bas relief ambiguity by exploiting points where the Lambertian reflection is maximal,

3. (Entropy) Entropy minimization [8]: proposes a new prior on the albedo distribution that the entropy of the distribution should be low,

4. (TV) Total Variation [138]: solves the GBR ambiguity by performing a total variation regularization on both the estimated normal field and albedo.

We conduct experiments on the following isolated factors:

- Uncompensated ambient illumination;
- Inter-reflections: tested on cones with the same radius but varying heights;
- Material: tested on materials of different combinations of diffuse, specular, subsurface scattering, and textured materials;
- Number of images: tested on different number of images ranging from 3 to 36;
- Image noise: tested by adding artificial Poisson image noise to renderings;
- Light-ring position: tested by changing the height of the light ring over the dome;
- Amount of shadows: tested on a wave shape with different depths;
- Surface roughness and specularity.

Median values and standard deviations of angular errors per experiment are shown in Figure 2.3.5 for both CPS and UPS algorithms. We mark the best results with blue. We draw the following conclusion from these statistics:

		CPS				UPS		
		Lsq	RPCS	Iter Lsq	HSH	LDR	Entropy	TV
Ambient	med	18.73	20.74	**6.95**	26.78	24.64	65.92	**17.19**
	std	16.12	7.27	**6.62**	**5.07**	21.27	24.62	**2.79**
Inter-reflection	med	44.67	44.57	**44.51**	55.40	87.65	**83.16**	93.33
	std	13.76	12.14	14.30	**8.37**	**10.69**	41.85	24.27
Material	med	3.33	8.17	**3.09**	11.79	**13.75**	56.32	18.42
	std	4.58	0.55	**0.51**	1.09	6.31	31.94	**1.68**
Number of Images	med	**0.87**	4.76	**0.83**	13.43	**5.87**	8.52	18.51
	std	**0.88**	13.25	**0.94**	23.30	20.25	35.41	**8.10**
Noise	med	0.96	6.74	**0.80**	91.00	**10.20**	60.95	17.76
	std	0.45	0.39	**0.23**	4.04	0.95	22.28	**0.26**
Ring Position	med	**2.97**	10.84	**3.39**	68.00	**29.80**	60.94	45.79
	std	**13.67**	15.03	**12.40**	16.35	34.76	34.70	**25.67**
Roughness	med	5.74	7.21	**3.77**	14.95	**13.53**	46.37	27.65
	std	1.73	0.46	**0.11**	10.39	**2.87**	26.53	7.39
Shadow	med	**13.70**	51.94	43.24	58.16	99.58	**53.65**	67.39
	std	**5.94**	20.14	27.95	23.22	14.50	20.84	**7.54**
Specularity	med	2.67	6.78	**1.16**	10.74	**7.50**	70.08	13.33
	std	1.76	**0.07**	**0.13**	0.63	**0.79**	3.20	0.86

Figure 2.13: **Evaluations:** Angular error statistics (median values and standard deviations) of estimated normal maps for various experiments. Errors are measured in degrees.

(a) Global Illumination (b) Direct Illumination (c) $\|(a)-(b)\|$ (d) CPS low-freq error

Figure 2.14: **(a) and (b)** show the rendered images with and without global illumination. **(c)** is the difference image between (a) and (b). **(d)** is the low frequency component of the angular error of the normal map calculated with iterative least squares method. (Difference images are scaled for better visualization.)

- In general, CPS algorithms outperform UPS algorithms in all experiments as expected, which implies that the illumination conditions have the greatest effect on the PS algorithms;
- Among the CPS algorithms, our iterative method either outperforms or performs as well as the others;
- Among the UPS algorithms, LDR produces less error than other methods on average, while TV is more robust to variations in the isolated factors, producing less standard deviation in errors;
- Inter-reflections and shadows are the factors that have the strongest influence on the errors. These two factors are hard to separate in the experiments even though we tried to have a minimum amount of self-shadowing in inter-reflections experiment by adjusting the light positions accordingly; however it is unavoidable to get rid of the self-shadowing completely and vice versa;
- On the other hand, image noise and surface specularity have the least effect on errors in the PS algorithms;
- Increasing the number of images is only important up to a point for each algorithm, which varies between 5 and 10 images;
- The material type or the surface roughness, although tested on a limited subset, does not affect the normal estimation as much as shadows or inter-reflections.

Furthermore, we do error analysis over normal-z distribution and in frequency domains as follows:

Error Distribution over Normal-z Direction. In Figure 2.15, we show average errors in both normals and albedo on 3 datasets for various CPS and UPS algorithms mentioned above. The x-axis of the plot is the z-component of normal vectors, while the y-axis is the

average angular error for portions of the surface having a normal with that z-component. As the z-component approaches 1, the surface approaches facing to the camera. The histogram of z-components is shown with the blue curve (legend on the right y-axis). The first row shows an image from the datasets, while the 2nd and 3rd rows are angular and albedo error curves for calibrated photometric stereo algorithms, respectively. The last two rows are the error plots for uncalibrated photometric stereo in the same fashion. Total average errors and the valid pixel ratios (some algorithms fail to estimate normals/albedo at some parts of the objects) are indicated in the legends per algorithm from left to right. The conclusion of this test is that there is a high correlation between the error and the angle between the camera and the true surface normal.

Frequency Domain Analysis. We also analyze the frequency distribution of errors, and separate the low-frequency and high-frequency errors. Low-frequency and high-frequency error images for normal maps are calculated as follows:

$$D = abs(N_{gt} - N_e) \tag{2.5}$$

$$L = \text{cross\_bilateral\_filter}(D, N_{gt}) \tag{2.6}$$

$$H = abs(D - L) \tag{2.7}$$

where N_{gt} is the ground truth normal map, N_e is the estimated normal map, and L and H are the low and high frequency components of the error respectively.

In Figure 2.14, the penguin is rendered with and without global illumination, shown on the top row. We observe that the difference of these two images resembles to the low-frequency component of the angular errors as shown in the bottom row. We do not show the high-frequency component of the angular error here because it is not as interesting, high-frequency errors concentrated around the silhouette of the objects. Given that the object has a diffuse material, the difference between global and direct illuminations is mostly caused by inter-reflections evidenced by the concavities being the areas with the biggest

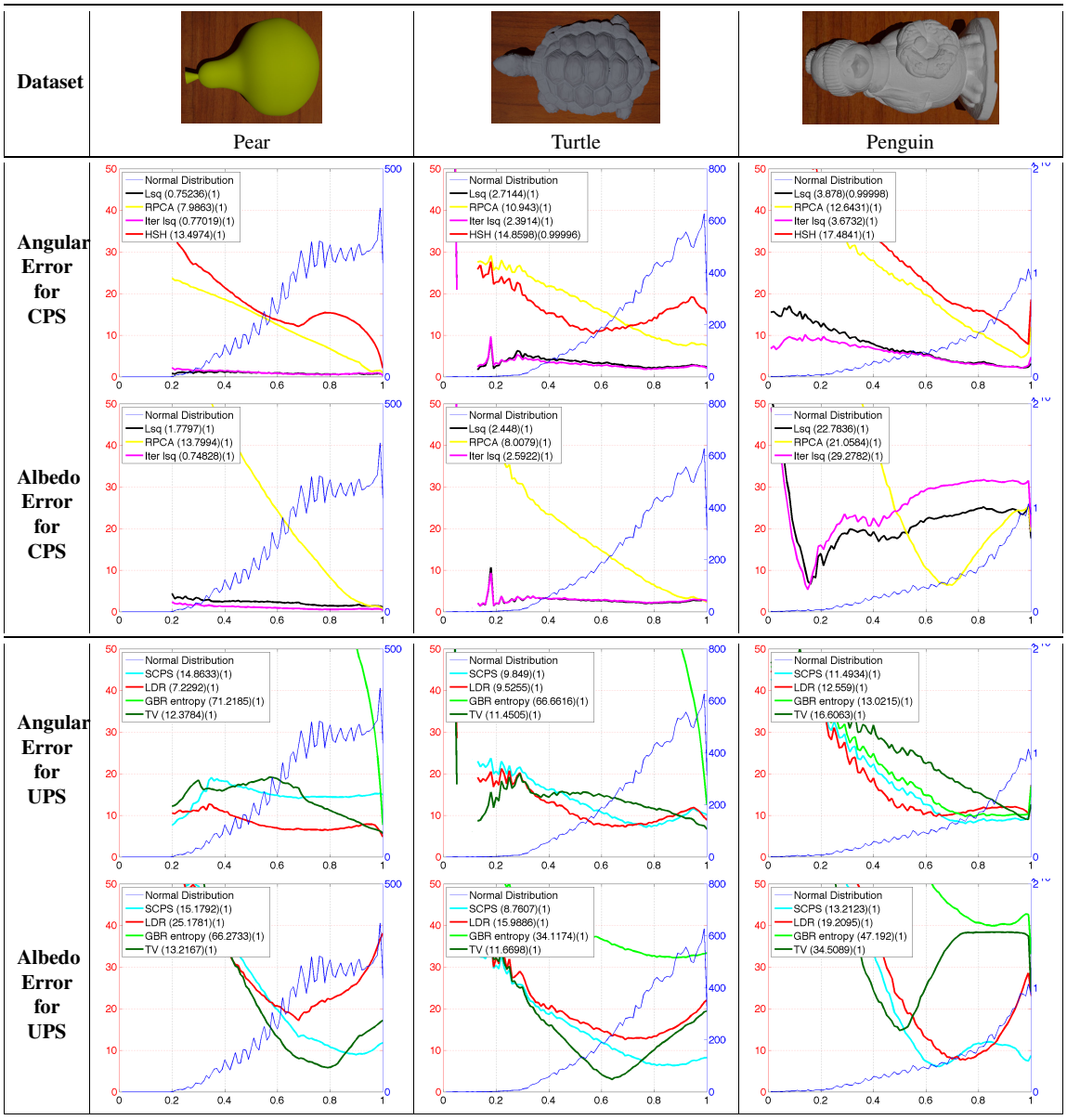


Figure 2.15: **Angular and albedo errors for selected objects (pear, turtle, penguin):** x-axis is the magnitude of the z component of normal vectors [0,1]; y-axis is the angular or albedo errors. The thin blue curve shows the histogram of the z- component of normal vectors in the ground truth normal map, and the second y-axis on the right side of each plot is the number of pixels for each histogram bin. In the legends, total average errors and ratio of the valid pixels are indicated from left to right respectively for each algorithm.

difference. Thus, we can conclude that the low-frequency errors are dominated by global illumination which is not accounted for by the algorithms.

2.4 Conclusion and Future Work

In this chapter, we first visited the photometric stereo algorithm and provided a taxonomy. Secondly, we proposed a feature-based framework to align images of the same object exposed to light from different directions. We showed that total alignment error for a dataset can be reduced by a graph-based approach rather than single-target alignment. Also, we demonstrated the importance of the image alignment by showing how much our algorithm improves the results of photometric stereo. The obvious limitation of this method is caused by feature detectors because they are not invariant to non-linear illumination changes. Even though they can handle small illumination changes, there is no guarantee that feature locations and descriptors will be consistent for large changes. In particular, lack of texture and geometry variations results in less reliable salient features. In future work, in order to cope with unreliable image features, surface geometry (normal-maps) can be included to feature descriptors iteratively. For badly warped images, we can allow the user to add some hard constraints, such as selecting a few control points on target and source images.

Finally, we presented an RGBN benchmark including a number of synthetic datasets along with an interface which enables the user to create arbitrary scenes and a few photometric datasets of real-life objects. We will make all the code and the data available online with the hope that researchers will make use of this benchmark in the future. We think that a synthetic benchmark would be very useful to understand which algorithms, using photometric data, work better under which conditions. Although we discussed that the synthetic datasets are comparable to the real ones, they are limited by the physically-based rendering system. Even though the today's rendering machines are very powerful and capable, they have limitations. For example, Mitsuba Renderer cannot simulate the wave

properties of light, does not account for polarization, and the accuracy of any rendering system relies on the underlying floating point computations [79]. We also did not intend to create a benchmark with all the varieties but we would like to let it grow by accepting submissions from other users of our interface. We hope that our evaluations inspire improved photometric stereo algorithms that are robust to many types of effects. While our iterative least-squares photometric stereo algorithm provides a first step by exhibiting increased resistance to shadowing, both the calibrated and uncalibrated cases would benefit from targeted improvements based on the test cases we have developed as part of this benchmark.

Chapter 3

3D Surfaces with High-quality Details

In the previous chapter, we visited some techniques to acquire high-resolution surface details and propose methods on how to achieve the best quality from those datasets. In this chapter, we focus on *capturing* surfaces with details in 3D. We first review some 3D acquisition and surface reconstruction techniques, then present a complete system to enhance rough 3D geometry with multiple high resolution normal maps.

3.1 3D Surface Acquisition & Reconstruction

Geometry acquisition has become increasingly popular in computer graphics and vision, with demand for high-quality models driven by advances in 3D printing, realistic real-time renderings of 3D self-avatars in video games, digital libraries for historical objects etc. In this section, we will visit several surface acquisition and reconstruction techniques that are also used to capture 3D models shown later in this chapter.

3.1.1 Acquisition Systems

In the literature there are various types of technologies for range acquisition such as contact-based scanners [74], volume scanning systems (MRI, CT, ultrasound) [67], sonar [170],

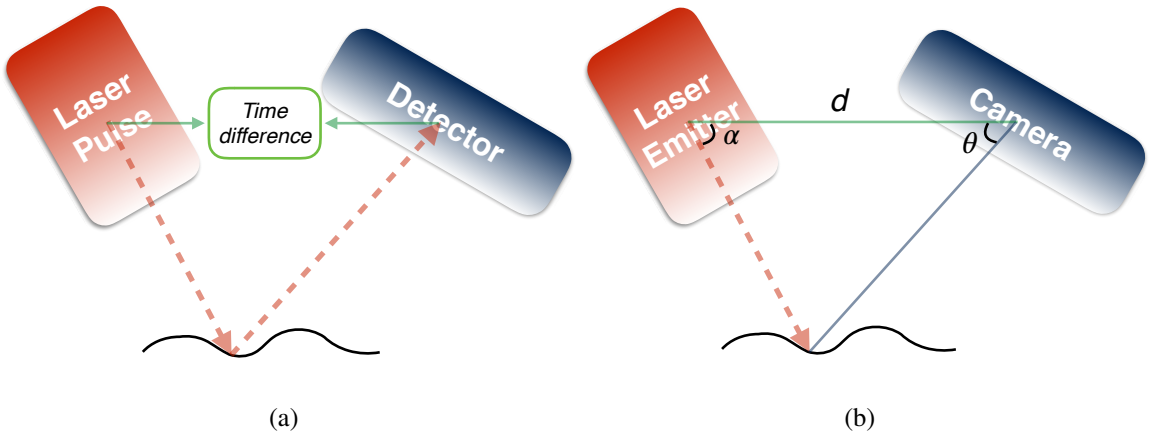
radar [139], optical systems [21], seismic survey [25] etc. In this thesis, we focus on optical shape acquisition systems which can be broadly categorized into two: passive and active systems.

Passive range acquisition systems do not interfere with the scene while scanning the object as opposed to the active systems. Passive methods are also known as *Shape from X* where *X* can be motion, stereo, shading, texture, focus – see [154] for further details.

Active scanning systems interfere with the scene by sending a signal to the object and measuring its response. These systems can be sub-categorized into three:

1. **Time-of-Flight** systems, such as Light Detection And Ranging (LIDAR) [55] and the second version of the Microsoft Kinect sensor, determine the distance between the device and an object by sending a laser pulse to the object and measuring the amount of time which takes the laser pulse to reach to the target and come back to the sensor, as shown in Figure 3.1a. These systems have the accuracy of a few centimeters because of the fact that it is hard to measure the travel time of light in small distances. Which is why they are usually preferred to measure larger distances such as room-sized objects and buildings.

2. **Triangulation** method is the most common way of measuring the distance of an object. Triangulation-based laser scanners are composed of two main parts: a laser emitter to shine the object and a camera to localize the laser dot where the laser beam hits the object as shown in Figure 3.1b. This configuration then forms a triangle among the source of the laser beam, the point where the laser beam hits the object, and the camera. The distance between the camera and the laser emitter, d, is fixed and known, as well as the angle, α, between the laser beam and the line between the laser emitter and the camera. The angle on the camera corner, θ, can also be calculated based on the pixel location. Those three known values then form a unique triangle



(a) (b)

Figure 3.1: Basic setups for **(a)** time-of-flight and **(b)** triangulation-based laser scanners.

and allow us to compute the exact location of the laser dot where the laser beam hits the object.

Another form of triangulation-based systems is structured-light scanners which project multiple stripes at the same time to speed up the scanning process. Using multiple stripes at a time raises a correspondence problem which is usually solved by using a time-coded light pattern such as gray-coded patterns [143].

3. **Active Shape from X** methods are the variants of the passive acquisition systems which are usually modified by projecting a known texture or pattern, such as checker-board, onto the object [154]. Another form of active shape from X is the photometric stereo which requires fixed camera and object positions while a single light source moves around the object – see Chapter 2 for details.

3.1.2 Surface Reconstruction

The need for a separate surface reconstruction step stems from the fact that an entire 3D model typically cannot be acquired in one sweep. To acquire a complete 3D model we need to capture the object from multiple views for example with one of the techniques mentioned in the previous section. Even though the camera poses of each scan can be

estimated with some calibration technique, in practice, alignments between different scans are not usually perfect. The most common algorithm used for aligning range scans to each other is iterative-closest-point algorithm and its variants [142, 32]. Once all the scan from different views are aligned, one of the surface integration technique can be used to fuse all the scans to obtain a single model. One of the most common surface integration algorithm uses a volumetric representation, accumulates weighted signed distance function, and extracts an isosurface from the volumetric grid [44]. Another technique formulates the surface reconstruction from oriented points as a Poisson problem [89, 90]. A basic pipeline for 3D model acquisition including texture acquisition and parametrization can be found in the work proposed by Bernardini and Rushmeier [22].

3.2 Combining Multiple Normal Maps with 3D Surfaces

In this section, we introduce an approach to enhance rough 3D geometry with fine details obtained from multiple normal maps. We begin with unaligned 2D normal maps and rough geometry, and automatically optimize the alignments through a 2-step iterative registration algorithm. We then map the normals onto the surface, correcting and seamlessly blending them together. Finally, we optimize the geometry to produce high-quality 3D models that incorporate the high-frequency details from the normal maps. We demonstrate that our algorithm improves upon the results produced by some well-known algorithms: Poisson surface reconstruction [89] and the algorithm proposed by Nehab et al. [130].

3.2.1 Introduction

3D surface acquisition is a critical problem in various fields as we discussed in the previous section. However, producing detailed geometry requires not only precise acquisition devices, but also a pipeline of processing steps including registration of multiple views (since an entire 3D model typically cannot be acquired in one sweep) and merging of different scans.

This section presents a high-quality reconstruction pipeline that combines data from two different sources: a rough 3D model obtained with an active (laser-stripe or structured-light) scanner and a few 2D normal maps calculated by a photometric stereo algorithm.

The motivation for combining these two kinds of data is strong. Normal maps are easily obtained at high resolution using active shape from shading (photometric stereo). They can contain a wealth of information about the fine details of a surface, but suffer from low-frequency bias. 3D scans, in contrast, are difficult to obtain at extremely high resolutions, but they contain all of the information about the topology and overall coarse-scale shape of the surface.

Although direct reconstruction of a 3D surface from normal maps is possible, in practice the results are far from satisfactory because of bias present in low frequencies, noise, and insufficient constraints among disconnected patches. This problem was addressed by the work of Nehab et al. [130], which proposed combining the low-frequency components of the geometry with the high-frequency details of the normal maps. However, that work assumed perfect alignment between the 3D geometry and a single 2D normal map. Also, their optimization for full 3D models (as opposed to height fields) is an approximation that forces the surface to be locally planar, resulting in over-smoothing of high-frequency details.

In this section, we present a complete system to enhance rough geometry with multiple detailed normal maps. The pipeline consists of four main steps:

1. ***Acquisition*** of 3D rough geometry, as well as extraction of 2D normal maps from images under varying light directions with respect to the object;

2. ***Alignment*** of the normal maps to the rough geometry without any initial alignment;

3. ***Blending*** of multiple normal maps to produce a seamless normal field over the surface;

4. ***Optimization*** of the 3D surface to incorporate high-frequency details from the normal maps.

This pipeline takes two types of inputs (multiple 2D normal maps and coarse 3D geometry), and produces an enhanced 3D model. The advantage of this system is that it requires neither initial alignment nor resolution/precision compatibility between different data types.

Main contributions of this pipeline are four-fold:

- A new feature detector tailored for normal maps;
- An algorithm to align 2D normal maps to a 3D surface by iteratively minimizing dissimilarity of normals;
- Seamless mapping of the aligned 2D normal maps to the 3D geometry;
- A method for combining the original 3D positions with the mapped normals.

3.2.2 Previous Work

Registration. Pairwise alignment has been a common problem in many research areas such as computer vision/graphics, medical imaging, robotic vision, etc. The types of input signals can be varied, and can include 2D images (visible/infrared image, x-ray, magnetic resonance imaging, computed tomography , positron emission tomography etc.) and/or 3D geometry (point clouds, range maps, or surface meshes). Registration of different combinations of these input types has been studied in the literature, but there is little previous work addressing the problem of aligning normal maps to rough 3D surface geometry.

Surface normals are frequently used for $\mathbb{R}^3 \Leftrightarrow \mathbb{R}^3$ registration in various works [100, 129, 124]. Several papers explored invariant features and motion estimation for small nonrigid deformations between 3D models [82, 83, 102, 101, 104]. Wang et al.[172] proposed a method to register 3D face template to normal maps by selecting 8 points manually and deforming it to the generic model. All of these works showed that normal-guided registration is more robust than others.

The closest related problem to ours is that of aligning color images to geometry. Viola and Wells [169] and Corsini et al. [43] proposed such methods based on mutual information. While these techniques could be adapted for normal maps, both of them assume that the details in the 3D model are similar to those in the images. In our case, however, we accept rough 3D geometry as input and develop a method of reliable alignment to (possibly warped) normal maps.

Blending. Pérez et al. [135] proposed using the Poisson equation to blend two images seamlessly, and Chuang et al. [42] extended it to blending multiple texture images on a 3D object using the Laplace-Beltrami operator. Li et al. [103] also used Poisson blending, after recovering reflectance and global illumination. We extend these kinds of methods to blending normals, observing that the low-frequency bias in normal maps should be corrected before blending.

Combining 3D positions and normals. Nehab et al. [130] proposed a linear formulation to combine the high-frequency details of normals and low-frequency shape from 3D positions. However, their acquisition setup used the same camera to capture both range scans and normal maps (using photometric stereo [182]). That is why they assumed that their different data types were perfectly aligned, and they did not take missing parts or holes in the 3D positions into account. However, the condition of perfect alignment between the two different types of data cannot always be satisfied, as when 3D surface and normal maps are acquired using completely independent devices. Building upon the work of Nehab et al. [130], our system does not require or assume any alignment or similar resolution between the 3D positions and the normal maps.

Yu et al. [189] proposed an approach to improve the quality of normal maps and consequently the geometry acquired with MS Kinect. They try to enhance the quality of the normals which is calculated from 3D data (depth); while we do not focus on that problem,

rather we assume that we have normal maps with higher quality than the geometry and use that extra information to improve the geometry.

A more general method for combining positions and normals is provided by Poisson surface reconstruction [89, 90]. This is a generic surface reconstruction algorithm that takes a point cloud with corresponding normals as input. However, if the accuracies and resolutions of the positions and normals are different, Poisson surface reconstruction often generates a smoothed result, as shown later in the chapter.

3.2.3 Data Acquisition

Both synthetic and acquired data are used for the experiments in this section. Creating the synthetic dataset begins with a ground-truth 3D model of the armadillo, and the followings are generated:

- **2D normal maps:** We render normal maps from multiple views and save the viewing directions for evaluation;
- **Rough 3D model:** We smooth the original 3D model with a Gaussian filter of $\sigma = 4\times$(average edge length).

Real data acquisition uses two separate capture setups:

- **2D normal maps:** We use the acquisition technique described in Chapter 2 to obtain each normal map;
- **3D geometry acquisition:** Of the many sensors available on the market for 3D acquisition, see Chapter 3.1, we experiment with two very different ones. First, we use a relatively high-resolution NextEngine scanner, which is based on laser stripe triangulation. We follow the steps in [33] to reconstruct the surface. Secondly, to explore even lower-resolution geometric input, we have also experiment with Kinect [121] – Skanect [132] software is used to reconstruct the surface from the captured data.

3.2.4 Alignment

Unlike other common alignment work-flows such as ICP [23, 41] which start with a user-provided rough initial alignment , we design a 2-step alignment algorithm to have a fully automatic system: i) estimation of an initial alignment via a feature-based algorithm , ii) iterative refinement by minimizing the dissimilarity between 3D surface normals and the normal map.

Initial Alignment. Inspired by SIFT [106], we design an adapted feature (nSIFT), which detects scale-invariant feature points and computes feature descriptors on normal maps. Denoting the normal map by $N(x, y)$ and the x, y components of it by $N_x(x, y)$ and $N_y(x, y)$ (which are embedded in R and G channels), we first construct a Gaussian scale space for $N_x(x, y)$ and $N_y(x, y)$. In order to detect stable keypoint locations in the scale space, we approximate the scale-normalized Laplacian of Gaussian as follows:

$$\sigma G(x, y, \sigma) * \frac{\partial N_x(x, y)}{\partial x} + \sigma G(x, y, \sigma) * \frac{\partial N_y(x, y)}{\partial y} \tag{3.1}$$

where $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$. Similar to [106], keypoints are detected by looking for local extrema in the 3×3 neighborhood at the scale-space. The orientation of each keypoint is assigned by the x and y components of the normal at that point and the descriptor is obtained by computing a 3D spatial histogram of the normals in the neighborhood of each keypoint, rotated by its orientation (Figure 3.2). Since the real normal map has more details than the rendered normal map we first smooth the real normal map by a Gaussian kernel of $\sigma = 3$; and during feature matching, we only use features detected in higher scales (with larger σ) of the real normal map to reduce false positive matches. After computing the matches among features using the nearest neighbor matcher, we use the RANSAC framework [56] to eliminate outliers. We also incorporate the location of feature points

relative to the center of the model's bounding box to further eliminate outliers. The algorithm summarizes our feature-based alignment.

Algorithm Feature-based Rough Alignment

Input: 3D model M, real normal map N, view sphere V, sample rate r.

repeat

Sample V at rate r to obtain set of viewpoints V_r

$\forall$ viewpoint $v \in V_r$

N_v = rendered surface normals of M from v

Find N's nearest neighbor N_v^* by nSIFT

Feature matching in RANSAC framework

$V \leftarrow N_v^*$'s neighborhood, $\quad r \leftarrow 2r$

until $\| N$ - $N_v^* \| < \tau$

Refinement. Inspired by ICP [23, 41], we propose an iterative alignment algorithm that begins with a rough alignment between a 2D normal map and 3D model and improves the registration by minimizing an energy function. To evaluate the energy function, we transform the 3D model given some candidate alignment, then render it into image space using the projection matrix of the normal map. The mesh is colored based on its normal map (transformed by the rotational portion of the alignment transform), so that the rendered image allows us to determine the mesh normal $\mathbf{N}_m$ that is mapped to each pixel (i, j). We then evaluate the dissimilarity between those mesh normals and the corresponding normals $\mathbf{N}_{img}$ from the normal map:

$$\mathcal{E} = \sum_{i=1}^{\#rows} \sum_{j=1}^{\#cols} f\big(\mathbf{N}_m(i,j), \mathbf{N}_{img}(i,j)\big). \tag{3.2}$$

The dissimilarity function f is defined as:

$$f(\mathbf{a}, \mathbf{b}) = \begin{cases} 0 & \text{if both } \mathbf{a} \text{ and } \mathbf{b} \text{ are masked,} \\ \pi & \text{if either } \mathbf{a} \text{ xor } \mathbf{b} \text{ is masked,} \\ |w \times cos^{-1}(\mathbf{a} \cdot \mathbf{b})| & \text{otherwise.} \end{cases} \tag{3.3}$$

The masks hold the locations of background pixels, so that this function imposes a large penalty if a pixel is part of the background of either the model or normal map, but not both. The weight w is calculated as $w = 1 - (\mathbf{b} \cdot \mathbf{C})$, where $\mathbf{C}$ is the camera direction. So, if the normal is facing towards the camera it gets a higher weight, and vice versa.

To optimize this energy function we need to transform the model and render the normal map, making the problem non-linear. We therefore use the Nelder-Mead algorithm [131], which initializes a simplex in the 6-dimensional rotation-translation space and heuristically grows, moves, and shrinks the simplex. For added robustness we perform randomized restart, computing a new random simplex (with maximum extent half the size as at the previous restart) once the simplex becomes sufficiently small, or if the algorithm performs more than 8 "shrink" operations in a row. We also re-initialize the simplex if 100 energy function calculations have been performed since the last restart. We terminate the algorithm after 8 restarts. Figure 3.2 shows the value of the energy function over time, for two different models. Restarts are marked with vertical dashed lines. At the beginning and end, and at each restart, we visualize the remaining misalignment: the normal map is rendered in red, while the mesh is in blue. We observe that the energy drops quickly during the first several iterations, eventually converging after 2–3 restarts.

To measure the accuracy of the alignment, we use two synthetic normal maps (front and side) obtained from the armadillo model. For each of 100 trials, we randomly perturb the ground-truth alignment by up to 6% of the radius of the model's bounding sphere to obtain an initial registration, and then run our iterative optimization to align the smoothed

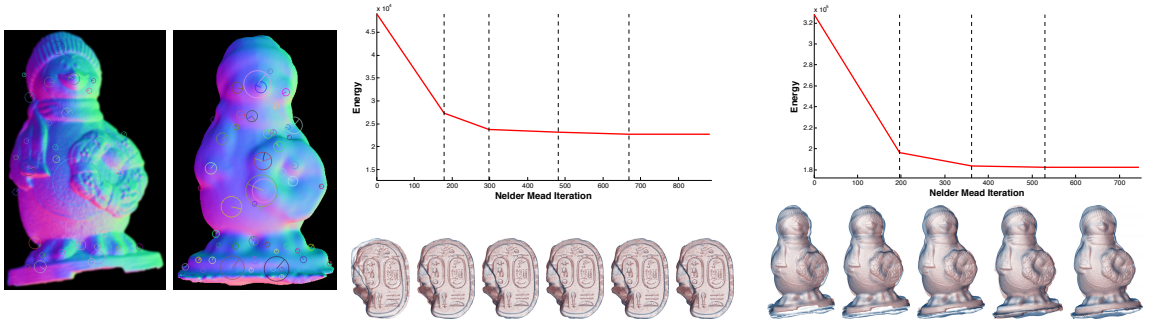


Figure 3.2: nSIFT feature points detected on pre-smoothed real normal map (first column) and nSIFT feature points detected on rendered normal map (second column). Optimized energy as a function of Nelder-Mead iterations for a scarab model (scanned with a laser scanner, third column) and a penguin model (acquired with MS Kinect, last column). False-colored visualizations of the normal maps (in red) blended with the 3D models (in blue) are shown below each graph for the initial alignment, after each restart, and the final alignment from left to right (restart positions are marked with vertical dashed lines).

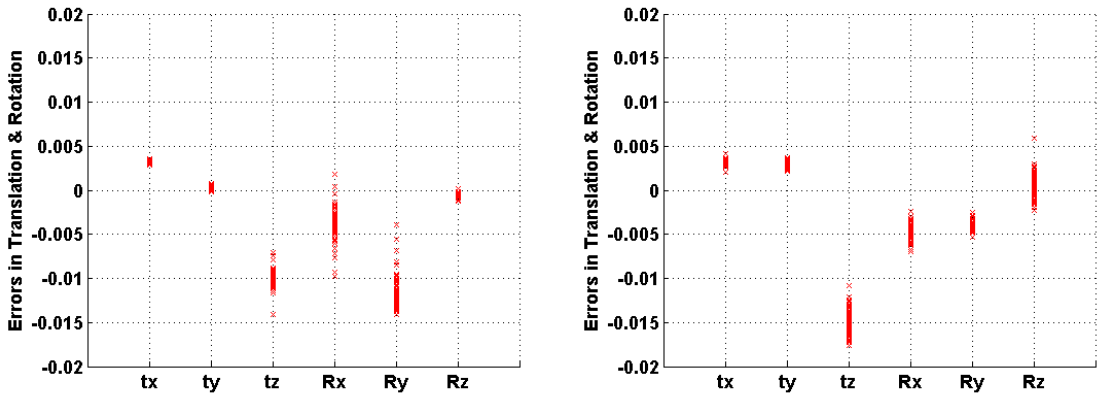


Figure 3.3: Alignment error for the armadillo dataset from the front (left) and the side (right). Error is measured for six parameters of rigid transformation: errors in translation (t_x, t_y, t_z) are measured as a fraction of the diameter of the bounding sphere of the model, while errors in rotation (R_x, R_y, R_z) are measured in radians.

3D mesh to the normal map. Scatter plots of the errors in translation (t_x, t_y, t_z) and rotation (R_x, R_y, R_z) are shown in Figure 3.3, where translation is measured as a fraction of the diameter of the model's bounding sphere and rotation is measured in radians.

Note that different components of the alignment are recovered with different accuracies, reflecting the screen-space perturbation resulting from changing each parameter. For example, for the front view (shown in Figure 3.3, left), it is more difficult to recover the z component of translation, corresponding to moving along the camera's direction of view, than it is to find the x and y components.

3.2.5 Blending Normal Maps

After the normal maps are aligned to the coarse 3D geometry, we would like to assign a single normal to each vertex on the 3D surface (after isometrically re-meshing the surface to have the desired output resolution). If the normal maps were perfect, and perfectly aligned, we could assign to each surface point a normal from an arbitrary normal map that observed it (e.g., the one that observed it most head-on). In practice, however, both of these hypotheses are false. Normal maps are *not* perfect due to limitations with the photometric stereo algorithm, which imperfectly accounts for shadows, non-Lambertian materials, non-opaque surfaces, and inter-reflection. The alignment is *not* perfect, since we are aligning to coarse 3D meshes that sometimes do not contain enough detail to reliably constrain the alignment. So, if we only assigned a single normal to each surface point, the resulting normal field would be discontinuous. Averaging all the normals projecting to a given 3D surface point would not solve the problem, since we would still see seams where we switched between including, and not including, the contribution of each normal map.

A partial solution to this problem was proposed by Nehab et al.[130]. They observed that the dominant error in normal maps is low-frequency. In contrast, while scanned 3D models might have high-frequency noise or lack high-frequency detail, their low frequencies are reliable. It is therefore possible to *correct* the normal maps by replacing their low-frequency

components by the low-frequency component of normals obtained from the 3D model. We adopt this strategy, including it in all of the results shown in this chapter. Nevertheless, we find that this correction is not always sufficient for avoiding seams between data from different maps.

Inspired by Poisson image editing [135], we propose blending between the (already-corrected) normal maps in a way that preserves their gradients. To do this we optimize an energy function with two terms, a vertex term that pushes the result towards a consensus (weighted average) among samples, and an edge term that preserves differences between neighboring normals:

$$\mathcal{E} = \lambda \mathcal{E}_{vert} + (1 - \lambda)\, \mathcal{E}_{edge}, \tag{3.4}$$

where λ controls the relative significance of the terms. The vertex energy seeks to minimize the weighted difference between the mesh normal $\boldsymbol{n_v}$ and the projected normal from each normal map $\boldsymbol{n_v^s}$:

$$\mathcal{E}_{vert} = \sum_{v=1}^{\#vert} \sum_{s=1}^{\#maps} w_v^s \, \|\boldsymbol{n_v} - \boldsymbol{n_v^s}\|^2, \tag{3.5}$$

where higher weight is given to normal maps that see the surface more directly: $w_v^s = (\boldsymbol{d_{viewer}} \cdot \boldsymbol{n_v^s})^p$.

The edge energy ensures that local gradients among the samples and over the surface will be consistent:

$$\mathcal{E}_{edge} = \sum_{e=1}^{\#edge} \sum_{s=1}^{\#maps} [(\boldsymbol{n_{ev1}} - \boldsymbol{n_{ev2}}) - (\boldsymbol{n_{ev1}^s} - \boldsymbol{n_{ev2}^s})]^2, \tag{3.6}$$

$\boldsymbol{n_{ev1}}$, $\boldsymbol{n_{ev2}}$ are the normals at the endpoints of edge e.

This energy function can be minimized with linear least squares, and it is sparse because there is only one coefficient for each vertex constraint and two coefficients for each edge constraint. We set the parameter $\lambda = 0.4$ to blend multiple normal maps for all results demonstrated in this chapter.

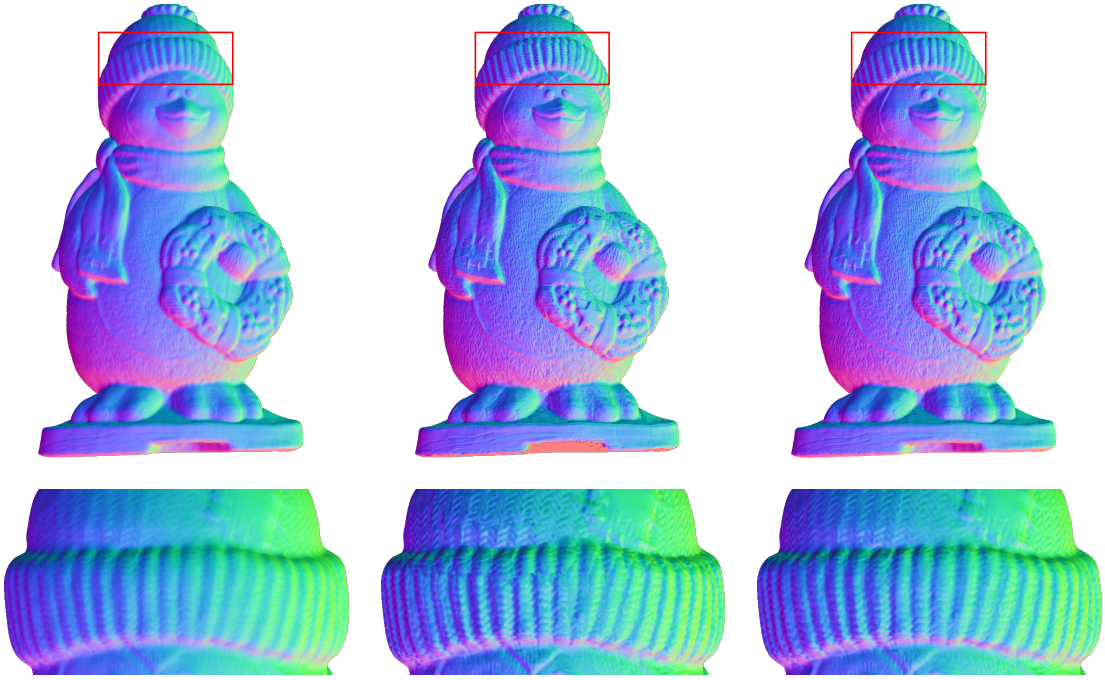


Figure 3.4: Comparison of different blending methods. **Left:** averaging all sample normals. **Center:** choosing the normal which is captured from the most direct view point. **Right:** our proposed blending method. Normals are false-colored by embedding in the RGB color space.

We compare our method to two alternative approaches: averaging samples, or just taking the normal captured from the most direct view. As shown in Figure 3.4, both of these methods create dramatic transitions, especially close to silhouettes. Our method successfully blends samples coming from different normal maps, and handles silhouettes well.

3.2.6 Surface Enhancement

Given high-quality normals and rough vertex positions, the true surface can be approximated by Poisson surface reconstruction[89], or using the method of Nehab et al.[130]. However, one of the most common problems with these methods is loss of some high-frequency information. Even though the work of Nehab et al. was designed to compensate for this problem, their full-model optimization (as opposed to their method for height fields) sacrifices some

high-frequency details to make the problem linear, by assuming that the surface is locally planar.

To overcome this drawback, we perform a non-linear optimization, minimizing an energy function consisting of two terms (similar to the previous work [130]) as follows:

$$\mathcal{E} = \lambda \mathcal{E}_p + (1 - \lambda)\mathcal{E}_n, \tag{3.7}$$

where $\mathcal{E}_p$ is the position term and $\mathcal{E}_n$ is the normal term (see [130] for notations). These are formulated as follows:

$$\mathcal{E}_p = \sum_{v=1}^{\#vert} \|\boldsymbol{p_v} - \boldsymbol{p_v^{mesh}}\|^2 \tag{3.8}$$

$$\mathcal{E}_n = \sum_{v=1}^{\#vert} -\|\boldsymbol{n_v} \cdot \boldsymbol{n_v^{measured}}\|^2, \tag{3.9}$$

where the optimized positions $\boldsymbol{p_v}$ are intended to match those coming from the mesh, and the optimized normals $\boldsymbol{n_v}$ are intended to match those coming from the normal maps (after alignment, low-frequency correction, and blending). We have also experimented with calculating $\mathcal{E}_n$ by summing over the faces rather than over the vertices, but found that this increased computational complexity with little benefit visible in the final result.

We minimize this energy function with the conjugate gradient method, using analytically-computed Jacobians and a line search at each iteration. We use the parameter $\lambda = 0.4$ for the enhancement in both our method and the algorithm of Nehab et al. [130] for comparison.

Figure 3.5 shows a quantitative comparison of our method to Poisson surface reconstruction [89] and the method of Nehab et al. [130]. We found that the latter two methods exhibit less smoothing if the surface positions are close to the correct ones, and so we actually run these methods in an *iterative* manner. We start with a smoothed armadillo model and normal maps obtained from the original mesh. We then repeatedly assign the original normals

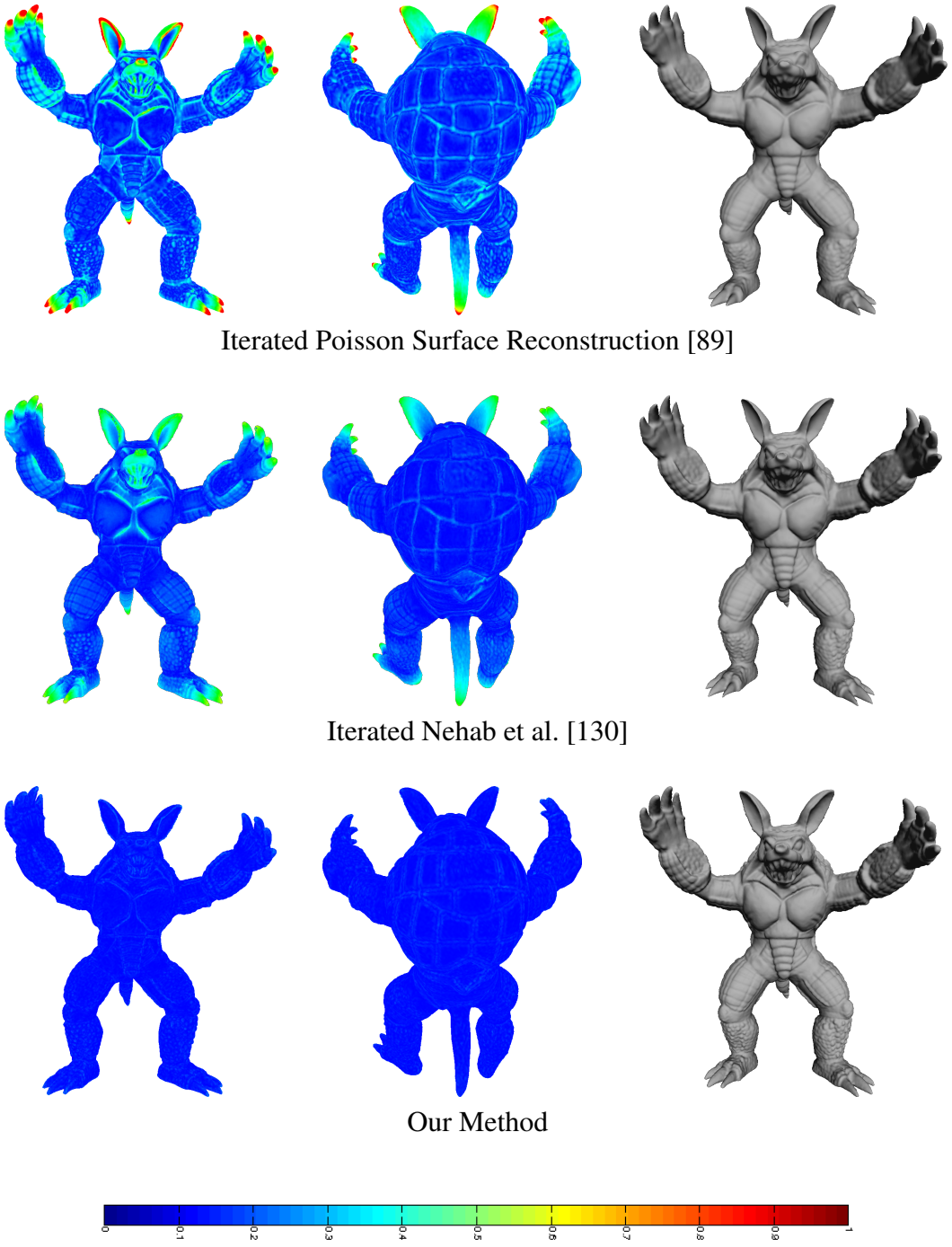
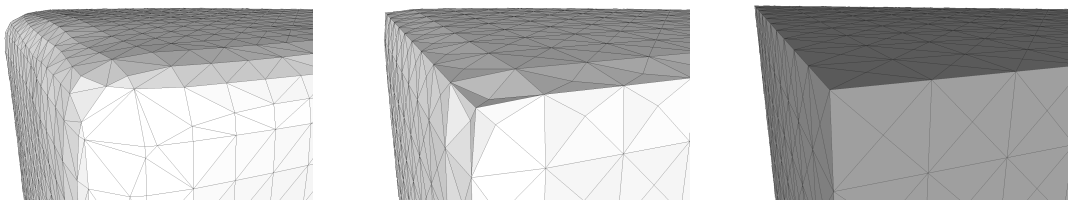


Figure 3.5: Comparisons of our method to Poisson surface reconstruction and Nehab et al. Blue represents zero distance between the reconstructed surface and ground truth, while red is greater than or equal to 0.001 of the radius of the bounding sphere.



Iterated Poisson Iterated Nehab et al. Our Method

Figure 3.6: Comparison of results of Poisson surface reconstruction [89], Nehab et al. [130], and our method on a cube. The ground-truth vertex positions and normals of the cube were used as input.

onto the enhanced model from the previous iteration (using ground-truth alignment), and reconstruct/enhance the surface, until each algorithm converges.

The visualizations in the figure are color-coded by perpendicular distance from the original model to the enhanced surface, with the color mapping shown at the bottom of Figure 3.5. It is obvious that Poisson reconstruction smooths out high frequency details, and flattens parts with high curvature such as the tips of the ears, hands, noise, and feet. Note, however, that our scenario was not really the one for which this algorithm was designed. Note also that we do not use the more recent Screened Poisson Surface Reconstruction algorithm [90] because it gives more weight to the vertex positions, which would result in even worse results.

The algorithm by Nehab et al., which *was* designed for this specific problem, produces better results. However, our method, which performs the correct non-linear optimization, loses fewer high-frequency details and produces results very close to the true surface.

In Figure 3.6, we used a cube to test an extreme case. We used the original positions and normals of the cube as input, in order to observe the behavior of the three algorithms in a controlled setting. The previous algorithms are unable to preserve the sharp corners of the cube, introducing significant smoothing. In contrast, our method neither introduces any artifacts nor smooths over the corners.

3.2.7 Results and Discussion

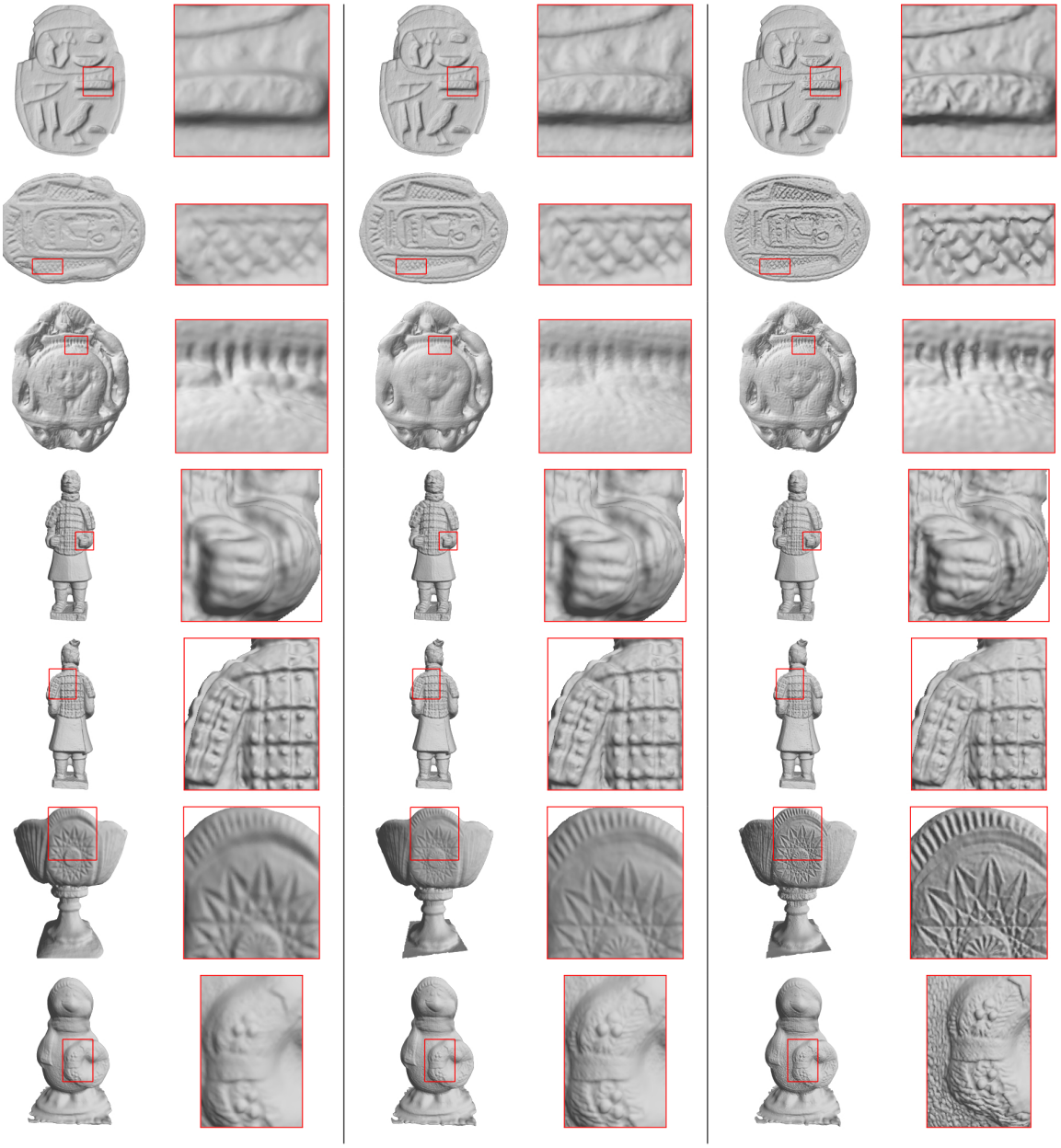
Figure 3.7 demonstrates results for the three algorithms described above, using source data from two different types of 3D acquisition techniques: moderate-resolution models acquired with the NextEngine laser scanner and coarse models obtained with MS Kinect. The leftmost column in Figure 3.7 shows the results and close-up images for the iterative Poisson reconstruction; the middle column shows the results of the algorithm proposed by Nehab et al. [130]; and finally the rightmost column shows our results. All of these used our proposed alignment and blending algorithms.

Our method preserves more of the high-frequency information from the normal maps, as compared to the other two methods. This is particularly visible in the close-ups in the last two rows of Figure 3.7, which used the comparatively coarse Kinect reconstructions as input. Also note the additional detail in the fingers of the soldier (fourth row) preserved by our method. Of course, in some cases this may also mean that our method preserves more noise, since it does not smooth the normal maps — this effect is especially visible in the third row of Figure 3.7.

3.2.8 Conclusion and Future Work

In this chapter, a complete pipeline for combining coarse 3D models with multiple unaligned 2D normal maps was presented, including acquisition, fully automatic alignment, blending the normal maps, and enhancing the model to incorporate high-frequency normals. We demonstrated that our algorithm outperforms two well-known methods: Poisson surface reconstruction from point clouds with normals [89] and the work proposed by Nehab et al. [130].

As future work, we can use more information from the normal maps, such as extracting the object silhouettes from them and constraining the 3D object to match those silhouettes. We can also explore large-scale applications, such as combining normal maps of a building with 3D point clouds from Google Street View or enhancing only a part of a big object.



Iterated Poisson | Iterated Nehab et al. | Our Method

Figure 3.7: Comparison of results of Poisson surface reconstruction [89], Nehab et al. [130], and our method. The 3D models in the first five rows are acquired with the NextEngine laser scanner, while the last two are obtained with MS Kinect.

Chapter 4

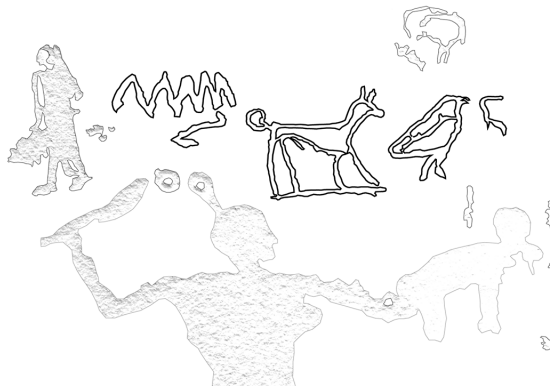
Abstraction from Surface Details:

Semi-Automatic Digital Epigraphy from Images with Normals

In this chapter, we propose a system to *process* acquired surface details in the form of normal and albedo maps. Specifically, we present a semi-automated system for converting photometric datasets (RGB images with normals) into geometry-aware non-photorealistic illustrations that obey the common conventions of epigraphy (black-and-white archaeological drawings of inscriptions). We focus on rock inscriptions formed by carving into or pecking out the rock surface: these are characteristically rough with shallow relief, making the problem very challenging for previous line drawing methods. Our system allows the user to easily outline the inscriptions on the rock surface, then segment out the inscriptions and create line drawings and shaded renderings in a variety of styles. We explore both constant-width and tilt-indicating lines, as well as locally shape-revealing shading. Our system produces more understandable illustrations than previous NPR techniques, successfully converting epigraphy from a manual and painstaking process into a user-guided semi-automatic process.

(a) Color Image

(b) Surface Normals

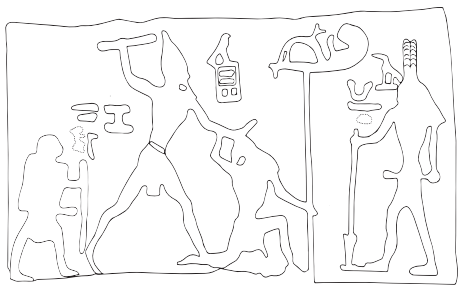


(c) Manual Digital Epigraphy

(d) Our Result

Figure 4.1: **(a)** Albedo and **(b)** normal maps are estimated with a photometric stereo algorithm. **(c)** Digital epigraphy done manually on Adobe Photoshop and Illustrator by archaeologists which took 7-8 hours in total. **(d)** Result from our user-guided system which took around 6 minutes.

(a) Photography



(b) Epigraphy

Figure 4.2: Photograph of a rock inscription and digital epigraphy done manually.

4.1 Introduction

Scientific illustrations are used in various fields to represent objects in a more effective and succinct way than is possible with photography. In archaeology, it is paramount for publications of record to use illustrations in agreed-upon styles, to document findings and communicate them to others for further study. For example, the conventions for epigraphic documentation, used for writing, drawings, and other inscriptions in stone, are based on traditional pen-and-ink drawings. While it offers great clarity, making epigraphic illustrations by hand can be time-consuming and imprecise. The quality and accuracy of the drawing usually depends on the archaeologist's talent, the size of the drawing paper and the object, and the time spent on the drawing. Recently, archaeologists have begun to adapt this painstaking process into the digital realm, by using tools such as Adobe Illustrator to trace over photographs. While still time-consuming (typically taking many hours per illustration), digital epigraphy removes the need to be on-site to execute the drawing, although it still may be necessary to go back to the site to correct perspective and add details that are hard to see in the photograph. Figure 4.2 shows an example of digital epigraphy of rock inscriptions from the Sinai done manually by archaeologists [156].

In traditional epigraphy, experts only use line drawings, sometimes with varying thickness and color. However, it is tedious and difficult to illustrate small surface details without creating distractions with line drawings. Recently, experts started looking for new ways to illustrate details. Museums and archaeologists have begun using Reflectance Transformation Imaging (RTI) for both visualization and digital preservation purposes [126]. RTI capture consists of photographing the object under study from a fixed viewpoint, but under varying lighting. Using this collection of photometric data, techniques such as albedo/normal estimation, re-lighting, and material modification may be used to produce enhanced visualizations. On the other hand, RTI is not as effective at conveying large-scale structures at a glance, and hence has not been adopted in publications of record as a replacement for epigraphy.

In this chapter, we present a user-guided system to create digital epigraphy with both line drawings and detail illustrations from photometric datasets. We focus on ancient rock inscriptions that are created by either carving into the rock or roughening ("pecking out") the surface. We make use of a photometric dataset of 4000-year-old inscriptions located in a desert, constantly under strong sunlight, making capture challenging. Furthermore, the rock surface is usually very rough and eroded because of weathering, which makes any type of captured data very noisy. The main challenges are to capture these inscriptions in high precision, find the relevant information, and attenuate the effects of noise caused by surface roughness and weathering. We propose a semi-automatic pipeline to solve this problem as follows:

- Capturing photometric datasets in challenging terrain, and estimating surface albedo and normals;
- Rectification of the surface normals to correct perspective;
- Segmentation of the inscriptions from the rock surface and classification based on carving technique (either slightly deeper grooves, or shallow pecked-out regions); and
- Stylization of the inscriptions in various styles.

For grooves, we produce illustrations in a traditional epigraphic line-drawing style, with line thickness optionally modulated by the grooves' depth to give a sense of relief. For pecked-out regions, we explore shaded and stippled styles, with control over whether the shading depicts global shape or only local detail. Our pipeline offers the user the control needed to produce clear illustrations, and deals with poor signal-to-noise more effectively than existing NPR methods.

4.2 Related Work

Segmentation. Image segmentation has been one of the oldest and most widely studied problems in the field [155]. Popular approaches include clustering techniques which aggregate pixels into regions with similar contents [78], graph cuts and energy based methods [28] [95]. However, the problem can become trickier in some situations, including segmenting the rock inscriptions which we focus on. Less work has been addressed in the context of segmenting on nearly flat surfaces. Zatzarinni et al. [191] have designed a method to extract reliefs and details from relief surfaces based on depth, but it is hard to apply their method to our dataset where some pecked-out inscriptions do not even have noticeable depth change. For robust segmentation on these datasets, we adopt the idea of combining color and normals for segmentation [161].

2D Line Drawings. Line extraction and drawing on 2D images has been extensively studied. For instance, Kang et al. [86] proposed using anisotropic Difference of Gaussians (DoG) guided by edge tangent flow to detect coherent lines while suppressing noise. Son et al. [149] extracted lines by imitating human line drawing process using estimation of a likelihood function with consideration of feature scales and line blurriness. Shape and shading on the image were represented by using tone and strokes by Lu et al. [108]. Winnemoller [180] reviewed the variations of the DoG operator, which was shown to give the most aesthetic results. Xu and Kaplan [186] depicted images using only black and white colors by formulating the problem of thresholding as an optimization over a graph of segmentation connectivity. All of those methods solely depend on the color variation in the image, which can be a drawback on rock inscriptions, where the color difference between the inscription and the background is usually small, and the surface roughness results in noise with traditional edge detection algorithms. On the other hand, we propose using both color and surface normal statistics at the same time to detect inscriptions.

3D Line Drawings. Several works have been published on how to represent 3D models with curves. DeCarlo et al. [45] proposed a new type of view-dependent curves — suggestive contours — which are the loci of points at which occluding contours appear with minimal change in viewpoint. Later, they proposed a new family of lines — suggestive highlights — which are complementary to the suggestive contours [46]. Judd et al. [80] introduced apparent ridges, which were defined as the loci of points that maximize a view-dependent curvature. Kolomenkin et al. [97] proposed another type of view independent curve: demarcating curves, which are the zeros of the normal curvature in the curvature gradient direction. Later, they defined relief edges as the zero crossings of the normal curvature in the direction perpendicular to the relief edge [96], and extended multi-scale edge detection on images to 3D surfaces by determining the optimal scale for each point on the surface [98]. These methods only use the geometry to find the extrema points, which can fail if the inscriptions are shallow and the surface is rough.

Shape Illustration. There are various non-photorealistic techniques in the literature to depict a shape from 2/2.5/3D images. For instance, Kim et al. [92] generated a stippling texture on an image from input samples by measuring a texture similarity metric. Toler-Franklin et al. [161] investigated various non-photorealistic illustrations on 2.5D images (images with normals). Brazil et al. [30] used Hermite Radial Basis Function Implicits to generate a robust distribution of points on a given 3D surface to position the drawing primitives, which are then rendered to depict shape and tone using silhouettes with hidden-line attenuation, drawing directions, and stippling. Vergne et al. [168] proposed a view-dependent shape descriptor called apparent relief, which combines the convexity of the 3D object and the normal variations in image-space. Several works focused on using varying line/stroke thickness to depict a 3D shape. Sousa et al. [151] extracted and rendered feature edges by varying the thickness based on perceived curvature of the 3D model and they [150] also used pen strokes to depict the 3D shape by making them thicker at certain curvatures,

junctions and creases. House and Singh [75] mimicked the human drawing process by following the connectivity of the feature edges and rendering them using a ribbon metaphor, where the thickness is determined by the twist of the ribbon. Goodwin et al. [62] computed the stroke thickness of contours and suggestive contours from depth, radial curvature, and light direction.

The previous works have been tailored to work with only either color or geometry information. However, we believe that the archaeological illustrations such as epigraphy represent not only the geometry of the artifact, but also variations in texture. We can use the photometric data for both geometry and texture information (normal maps and albedo extracted from the photometric data, respectively). Examples of why we need both sources of information are shown in Figure 4.3. Bartesaghi et al. [10] and TolerFranklin et al. [161] showed how to use photometric data for stylized non-photo realistic renderings. The former produces geometry and tone-aware hatching-like renderings; while the latter explores how to reformulate some previous work, such as suggestive contours and ridges and valley lines, for photometric data. Unfortunately, neither of these produces good results for archaeological illustrations. They are either incomplete or noisy, as we will show in Section 4.7.

4.3 Overview

Our main challenge is to produce meaningful (shape-indicating), relevant (including the most representative features) drawings of archaeological inscriptions. Unfortunately, the signal-to-noise ratio is typically low: the inscriptions are shallow, while there is considerable noise caused by surface roughness. We briefly review how our datasets are acquired and rectified (Section 4.4), then divide the problem into two major components: finding and classifying the inscriptions with user-guided segmentation (Section 4.5), and then rendering them in different geometry-aware non-photorealistic styles (Section 4.6). We present results

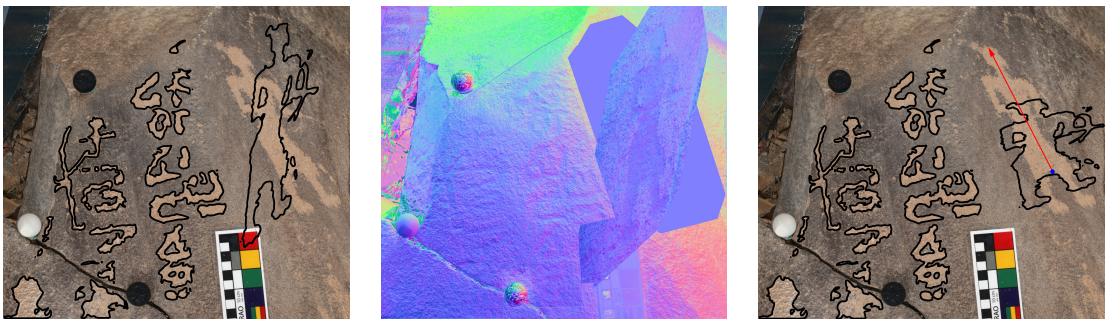
on a number of datasets, and argue that this type of stylization is difficult to achieve with previous methods (Section 4.7).

4.4 Data Acquisition and Preprocessing

We begin with datasets captured at an ancient amethyst mining site in an area of Upper Egypt called Wadi el Hudi, which was documented in 1952 by [52]. The terrain consists of several hills covered with granite rocks. The inscriptions are mostly located on the ridges and their position and surroundings make them hard to capture. The ridges are steep and the rocks on the ridges are usually loose, which makes it challenging to set up an acquisition system on the ground. Because of the environmental conditions, the recording must be done during the day under strong sunlight. Also, carrying heavy equipment or power sources to the site is impractical, since it takes a short hike to reach to some of the inscriptions. However, the resolution of the acquisition system should still be high, because the inscriptions are mostly on flat surfaces with shallow depth.

For these reasons, the most practical setup is one based on a camera and flash, in preference to 3D acquisition systems such as structured-light scanners, multi-view stereo, or a laser scanner. Specifically, the datasets are captured using a photometric acquisition setup, shown at left, similar to the one used by Toler-Franklin et al. [161], as described in Chapter 2. Estimated surface normals and albedo from those photometric datasets are shown in Figure 4.3. In this chapter, we show results for several representative datasets.

Figure 4.3: **Left column:** Color images of some rock inscriptions; **Right column:** Normal maps. **Top row:** While the inscriptions are visible on the color image, some of them are not recognizable on the normal map. **Bottom row:** The normal map reveals more information than the color image for some other cases.



(a) (b) (c)

Figure 4.4: **(a), (b)** Rectified inscription and normal map where the planar region to be rectified is defined by user. **(c)** User defined rotation axis (marked with red arrow) and flattened inscription.

4.4.1 Rectification

One of the important conventions of documenting rock drawings is that they need to be recorded from a perpendicular view to the surface. However, it is not always possible to place the camera perpendicularly to the surface because of the position and height of the object or the steepness of the terrain around it. Another problem is that the inscriptions sometimes start from one face of the rock and continue on the other. Recording them from an angle will make them look skewed, and this is not desirable for documentation. Archaeologists usually warp the image to straighten the parts of the image by either eyeballing to flatten these regions, or if they recorded some 3D points via some terrain mapping systems such as Total Station, they manually select the control points on the image corresponding to those 3D points and rectify the image. However, it is time-consuming to both capture some 3D points and rectify the image based on those points.

In our system, we allow the user to rectify the image without any extra data. The user can select a nearly planar region R, by selecting a few points around the region to create a closed polygon. Given this selection, we automatically rectify the region as follows:

$$\vec{n}_{mean} = \texttt{normalize}\Big(\sum_{(x,y)\in R} N(x,y)\Big) \tag{4.1}$$

$$\vec{r} = \vec{n}_{mean} \times \langle 0,0,1\rangle \tag{4.2}$$

$$\alpha = \texttt{acos}(\vec{n}_{mean} \cdot \langle 0,0,1\rangle) \tag{4.3}$$

where N is the normal map, $\vec{r}$ is the rotation axis, and α is the rotation angle. Once the rotation axis and angle are computed, the region is rotated around $\vec{r}$ and all the normal values are rotated accordingly as shown in Figure 4.4a-b. This approach assumes that the region is nearly planar which is the case for most of the datasets. We also allow the user to draw a rotation axis manually by drawing a line on the image and use this axis instead to rotate the region around, which requires two rotation operations: first we rotate the region so that the

average normal along the user-defined rotation axis would be equal to the z-direction and then we rotate the region around this fixed rotation axis to make the region flat as shown in Figure 4.4.c. The user also can control scaling of the preselected region after flattening, as done in Figure 4.4.

4.5 Segmentation

We focus on rock inscriptions, which can be classified into two types: carved grooves, and "pecked out" areas on the rock surface.[1] Their characteristic roughness and incompleteness caused by erosion make the creation of epigraphy very difficult. Existing NPR techniques are insufficient to produce nice and understandable illustrations for such rock inscriptions. We approach the problem by first performing a robust segmentation to create masks for the two types of inscriptions, and then stylize them separately. An implementation pipeline for this segmentation problem is described in the following subsection in detail.

4.5.1 Implementation Pipeline

Preprocessing includes downsampling, histogram equalization, and bilateral filtering to reduce noise and increase contrast between the foreground inscriptions and the background rock surface. Figures 4.5a-b show a color image before and after the preprocessing, respectively.

Over-segmenting the color image can help to enforce local consistency, thus better maintaining the boundaries of the foreground inscriptions. We apply a clustering-based algorithm [3] to aggregate pixels into superpixels and create an over-segmentation of the image: Figure 4.5c shows such an example.

[1] Pecking-out is a term used to describe areas where the rock surface has been roughened to form a design or figure.

Feature descriptors are extracted from both colors and normals. Color-based features are based on statistics over each superpixel. These are augmented with the information from normal maps that helps improve descriptiveness, especially when some inscriptions are difficult to see in the color image. The normals also help with distinguishing between carved grooves and pecked-out areas. Details are explained in Section 4.5.3.

Labeling is provided by the user who draws sparse strokes to indicate pixels belonging to 3 different classes. As shown in Figure 4.5d, red strokes refer to foreground grooves, green strokes refer to foreground pecked-out areas, and blue strokes mark the background.

A graph-cut approach [29, 95, 28] is used to group the pixels into foreground and background. Energy functions are designed based on feature descriptors and user labeling as explained in Section 4.5.3.

Post-refinement is applied to the graph-cut segmentation output to enhance foreground connectivity and remove noise on the background. Figure 4.5e-f give the segmentation results before and after the post-refinement processing.

Foreground classification is done after foreground/background segmentation to produce masks for the two types of rock inscriptions. A Naive Bayes classifier is trained based on the user input labels. Figure 4.5g shows an example classification result. We also allow the user to edit the label of each connected component, so that they can fix mislabeled components or remove irrelevant parts from the foreground, as illustrated in Figure 4.5h.

4.5.2 Feature Design

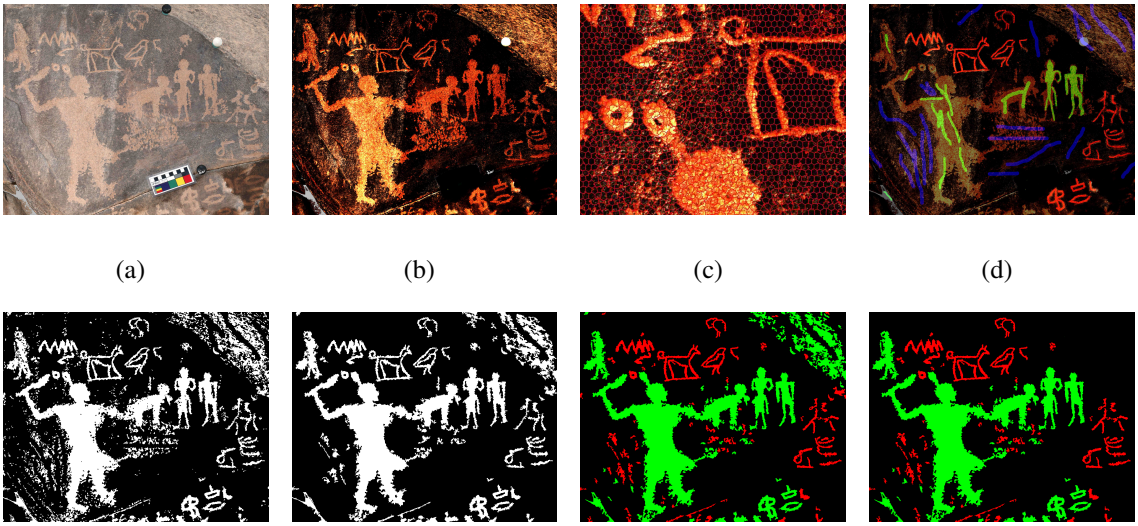
We compute a feature descriptor for each pixel using cues from both color and normal images.

Color-based features. Color images of rock carvings are noisy, and just using the color of each pixel as a feature yields results that are not robust. Instead, we first over-segment the color image into near-equal-sized superpixels. As shown in Figure 4.5c, these superpixels preserve the boundaries in the original image while capturing redundancy in the data, and they also help to reduce computational cost by allowing features to be computed only once per superpixel (instead of per pixel). Means and standard deviations are computed for each RGB channel within the neighborhood defined by each superpixel. Pixels from the same superpixel are assigned the same color feature. Figure 4.6a-b visualize the color features for the image shown in Figure 4.5a.

Normal based features. By only looking at the color image, it is hard to distinguish the carved grooves from the pecked-out regions. However, these two types of inscriptions have recognizably different normal statistics, as illustrated in Figure 4.7. Therefore, it is reasonable to compute features from normals in addition to color. We take the x and y components from the normal image to approximate surface gradient, and then compute the gradient magnitude. We also compute the covariance matrix in a Gaussian weighted square neighborhood for each pixel, and add the bigger eigenvalue as a new entry to the feature descriptor: this helps to distinguish grooves (which will have one big eigenvalue) from roughened regions (which will have two small eigenvalues). As visualized in Figure 4.6, normal based features help to capture edges on the surface.

4.5.3 Inscription Segmentation

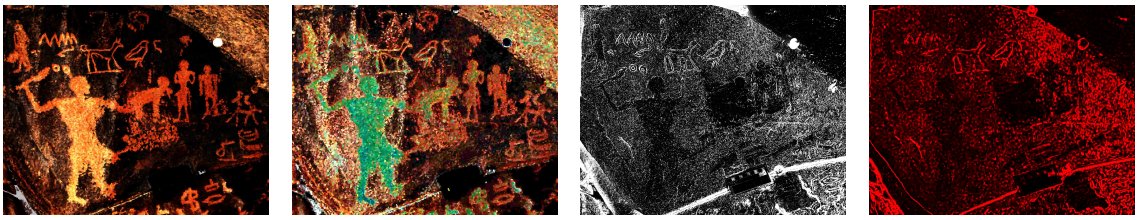
In order to produce labeled segmentation for both grooves and pecked areas, we first run a graph-cut based binary segmentation to obtain foreground and background, and then train a classifier on the foreground segment to distinguish between the two types of inscriptions. Based on our experiment, such a two-fold process is more efficient and more reliable, compared to multi-label graph-cut segmentation.



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

(e) (f) (g) (h)

Figure 4.5: **(a)** Original color image. **(b)** Color image after preprocessing. **(c)** A close up image of the over-segmentation by superpixels. **(d)** User input strokes for labeling. **(e)** Graph-cut foreground segmentation result. **(f)** Foreground segmentation result after post-refinement. **(g)** Foreground classification. **(h)** Foreground mask after user cleanup.



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

Figure 4.6: **(a)** Visualization of the color mean within each superpixel. **(b)** Visualization of the color standard deviations within each superpixel. **(c)** Visualization of the approximated surface gradient magnitude. **(d)** Visualization of the bigger eigenvalues of the structured tensors. (The contrast of the images are increased for better visualization.)

Training. We ask the user to label the two types of foreground inscriptions and the background by drawing strokes in different colors. Let $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}_1$ represent the set of labeled pixels, where $\mathcal{L}_0$ denotes the background labeled pixels and $\mathcal{L}_1 = \mathcal{L}_{10} \cup \mathcal{L}_{11}$ denotes the two types of foreground labeled pixels. For labeled pixels from $\mathcal{L}_0$ and $\mathcal{L}_1$, we run hierarchical clustering [127] to aggregate them into clusters in feature space. Let $f^s_{\alpha j}$ denote the centroid of the j-th cluster with label $\alpha \in \{0, 1\}$. We call f^s_{0j} background seeds and f^s_{1j} foreground seeds.

Energy function. We define a graph on the image in which each pixel is a node and each node has 4 edges connecting its 4-neighbors. A binary graph-cut segmentation algorithm is applied to find the least-cost boundaries that smoothly partition the graph into foreground and background by minimizing an energy function $E(\boldsymbol{\alpha}, \boldsymbol{f})$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)$ is the label assignment, $\boldsymbol{f} = (f_1, f_2, ..., f_n)$, is the vector of feature descriptors, and n denotes the number of pixels. We choose to apply the graph-cut segmentation on pixels instead of superpixels because the normal features are extracted from per-pixel neighborhoods. The energy function consists of two terms:

$$E(\boldsymbol{\alpha}, \boldsymbol{f}) = U(\boldsymbol{\alpha}, \boldsymbol{f}) + V(\boldsymbol{\alpha}, \boldsymbol{f}), \tag{4.4}$$

where $U(\boldsymbol{\alpha}, \boldsymbol{f})$ denotes the data term and $V(\boldsymbol{\alpha}, \boldsymbol{f})$ denotes the smoothness term.

We define the data term as

$$U(\boldsymbol{\alpha}, \boldsymbol{f}) = \sum_{p \in \mathcal{L}} -\log \frac{\min_j \left\| f_p - f^s_{\alpha_p j} \right\|}{\sum_\alpha \min_j \left\| f_p - f^s_{\alpha j} \right\|} + \sum_{p \notin \mathcal{L}} -\log \epsilon, \tag{4.5}$$

where $\| \cdot \|$ denotes Euclidean distance and $\epsilon > 0$ is a user defined small constant.

The smoothness term is defined as

$$V(\boldsymbol{\alpha}, \boldsymbol{f}) = \gamma \sum_{(p,q) \in \mathcal{N}} I(\alpha_p \neq \alpha_q) e^{-\beta (f_p - f_q)^2}, \tag{4.6}$$

where γ and β are user-defined positive weighting parameters. $\mathcal{N}$ is the set of neighboring pixel pairs in the grid graph. $I(\cdot)$ is an indicator function that is 1 if $\alpha_p \neq \alpha_q$, 0 otherwise.

Cleanup. After the graph-cut segmentation, we perform a post-refinement process to enhance foreground connectivity and remove speckles. For each superpixel, we assign the same label to all pixels within it by taking a majority vote on pixel labels from the graph-cut output. We then remove noise in the background and fill small holes in the foreground by removing small connected components.

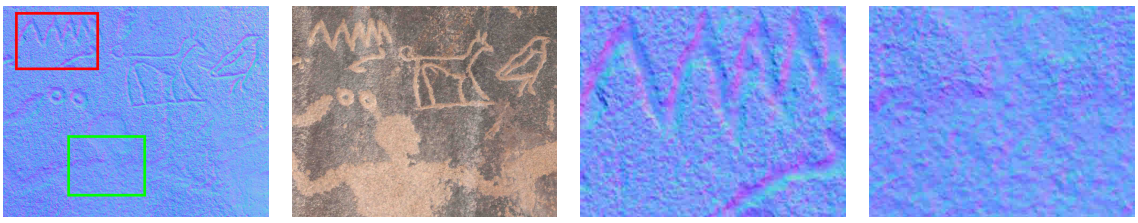
Label classification. Finally, we train a Naive Bayes classifier [59] based on feature descriptors of labeled foreground pixels from $\mathcal{L}_1 = \mathcal{L}_{10} \cup \mathcal{L}_{11}$, and predict the type to be either a carved-in or pecked-out inscription. We then assign pixels within each connected component with the same label by conducting a majority vote, and create a labeled segmentation mask for stylization.

Iterative refinement. In order to give the user fine control over segmentation, we implement an iterative scheme that allows the user to add strokes to refine the segmentation. It takes $3 - 4$ iterations on average for an experienced user to create a decent segmentation result. An example is shown in Figure 4.8.

4.5.4 Evaluation and Comparisons

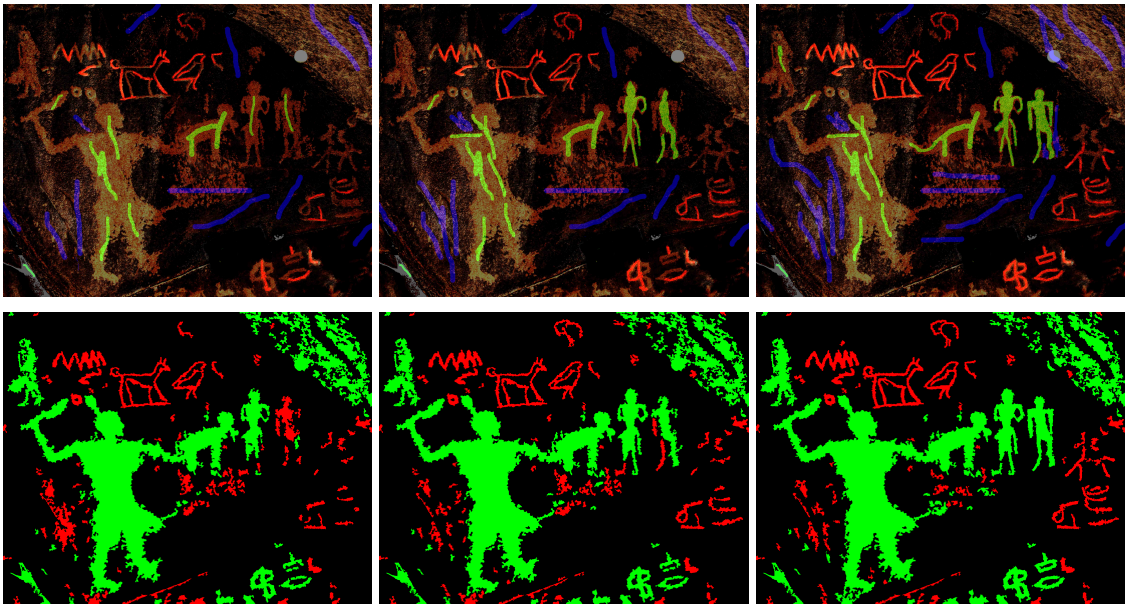
Graph-cut on color. We show the results of applying graph-cut segmentation on only the color image. Figure 4.9 shows that color alone is not enough to capture the real foreground inscriptions, especially the grooves, and therefore it is necessary to have more robust features based on normals as well.

K-means clustering. We compare our foreground/background segmentation result to that of performing k-means clustering on pixels in feature space. Figure 4.10 shows side-by-side



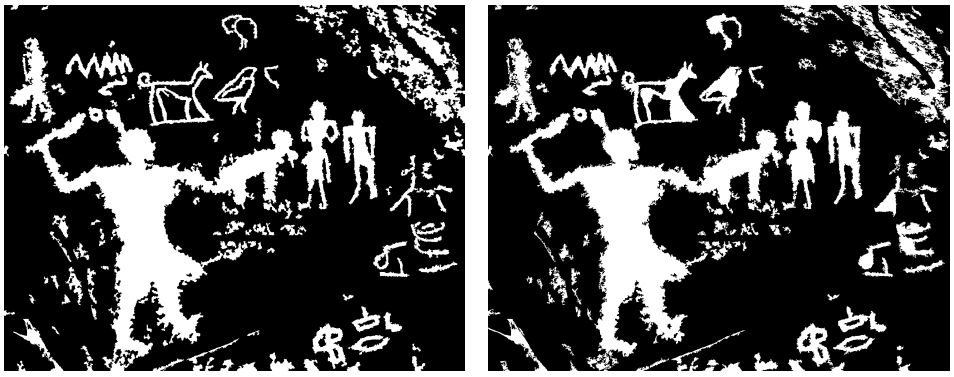
(a) (b) (c) (d)

Figure 4.7: **(a)** Normal and **(b)** color images with 2 types of inscriptions. **(c)** Close-up images of the carved in region (red rectangle) and **(d)** the pecked out region (green rectangle).



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

Figure 4.8: **Top row:** Labels drawn by the user. **Bottom row:** Segmentation results.



(a) (b)

Figure 4.9: **(a)** Our foreground/background classification results. **(b)** Foreground/background classification results of graph-cut on color.

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

Figure 4.10: **(a)** Our segmentation result before cleanup. **(b)** K-means clustering result before cleanup. **(c)** Our segmentation result after cleanup. **(d)** K-means clustering result after cleanup.

comparisons between our results and the k-means results. We notice that simple clustering yields more noise and incomplete foreground.

We also compare our final labeled segmentation result to aggregating the pixels into 3 clusters by k-means, as shown in Figure 4.11. We conclude that a supervised approach yields more reliable results for segmenting rock inscriptions.

Multi-label graph-cuts. We show a result of directly performing multi-label graph-cut segmentation and compare it to foreground/background segmentation followed by foreround classification. The computation time of the multi-label segmentation is roughly 50% higher

than our method. As shown in Figure 4.12, our two-stage process produces more reliable results.

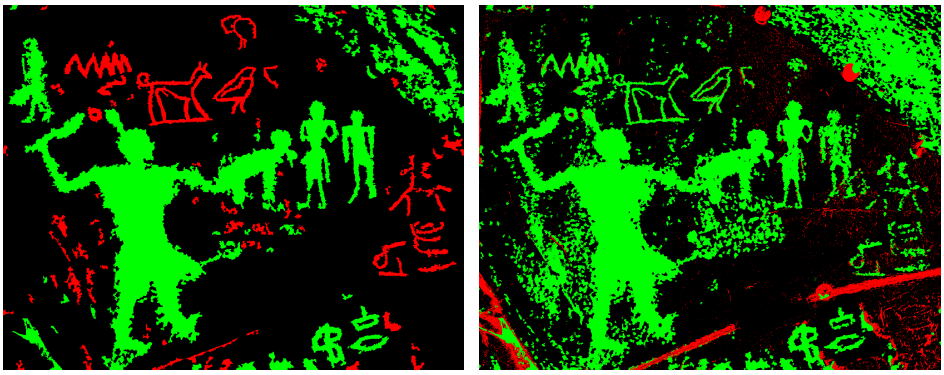
4.6 Stylization

Our system uses the labeled segmentations to drive stylized rendering. We describe two categories of stylization: *shape illustration* that depicts the depth of grooves or the general shape of the whole rock surface, and *detail illustration* to depict surface details in the pecked-out regions.

4.6.1 Shape Illustration

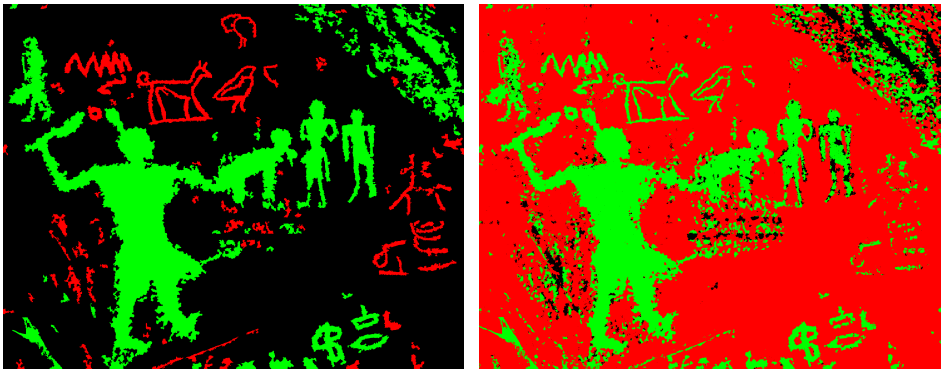
We first extract contours from the segmentation masks and convert each into a discrete curve by sub-pixel sampling the contour. Then, the user can manipulate the contours by controlling parameters for Gaussian smoothing, bilateral filtering, and unsharpening along the contour. Each filter can be applied to change either the contours' shape or their properties such as normals, curvatures, and colors. The user can also add a sense of relief by varying stroke thickness, or provide an impression of overall shape using stippling.

Relief. Some of the inscriptions are carved into the rock (grooves) which makes them either sunk or bas-relief depending on the carving technique. One way of representing reliefs with line drawings is to draw them with different line thicknesses. One of most common conventions used for reliefs in digital epigraphy is to assume a light source at a 45°angle from the top-left side of the image. The goal is to use line thickness to indicate the shading of the surface under this hypothetical lighting. In digital epigraphy, this is done manually by offsetting the mask for the inscriptions. However, this method only gives us general information about whether it is sunk relief or bas relief. For a more precise depiction, we use the depth of the inscription to determine the line thicknesses.



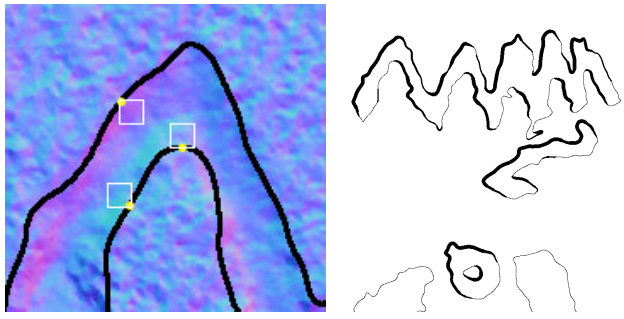
(a) (b)

Figure 4.11: **(a)** Our final labeled segmentation result. **(b)** K-means 3 clustering result.



(a) (b)

Figure 4.12: **(a)** Our final labeled segmentation result. **(b)** 3-label graph-cut result.



(a) (b)

Figure 4.13: **(a)** Thickness calculation. **(b)** Example of a relief illustration.

To determine the depth of the inscription without actually integrating the normal map, we take a small window, W, from each curve point in the opposite direction of the curve normal as shown in Figure 4.13a and take the average of the normals within that window. Later, we could just use the dot product of the light direction with the average normal to determine the line thickness. However, since the surface is not necessarily perpendicular to the camera and we want to get rid of the noise caused by the rough surface, we first apply a band-pass filter (Difference of Gaussians) on the normal map, N:

$$N_{DoG} = N * G(\sigma_1) - N * G(\sigma_2) \tag{4.7}$$

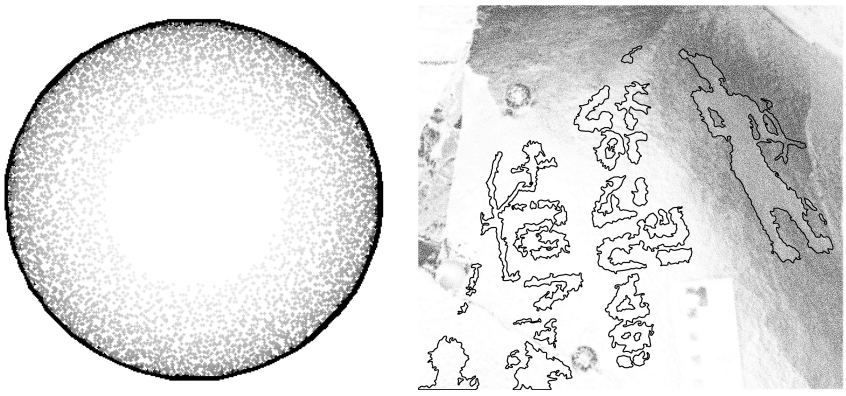
where $G(.)$ are 2D Gaussian functions with σ_1 removing the noise caused by the surface roughness and σ_2 removing the low frequency component of the normal map, which represents the rough shape of the surface. We later use the DoG to compute the average direction for each point on the curve, and the thickness is set proportional to the dot product of the average DoG and the imaginary light direction.

$$t(i) = \langle 1, -1, 0 \rangle \cdot \Big(\frac{1}{|W_i|} \sum_{(x,y) \in W_i} N_{DoG}(x, y) \Big) \tag{4.8}$$

where $t(i)$ is the computed line thickness at point i. Later, the user can further smooth the thickness along the curve to have a smoother transition along the contour as described below:

$$t(i)_{smooth} = \frac{1}{g_{total}} \sum_{j=-r}^{r} g(j, r) t(i + j) \tag{4.9}$$

where $g(j, r)$ is a Gaussian function in which the variance of the function is calculated from the maximum range r. In Figure 4.13b, an example is shown where the relief illustration is applied to the grooves while the pecked out regions are rendered with a constant line thickness.



(a) (b)

Figure 4.14: **(a)** Stippling on a sphere. **(b)** Stippling to depict the rough shape of a curved boulder.

Stippling. To depict the rough shape of the surface, we use stippling derived from the normal map. We compute the stippling density in small grids by taking the dot product of the z direction with the average normal within the grid, W, as follows:

$$d = 1 - \left(\langle 0, 0, 1 \rangle \cdot \left(\frac{1}{|W|} \sum_{(x,y) \in W} N(x, y) \right) \right) \tag{4.10}$$

A number of stippling points proportional to the density are randomly rendered within the small grid with a color inversely proportionally to the density. When the surface is tilted away from the camera, the stippling density will be close to 1 and the color will be darker, and vice versa, as shown for a sphere in Figure 4.14a. This method can be used over all the image to indicate the general shape of the surface when the rock surface is curved as shown in Figure 4.14b.

4.6.2 Detail Illustration

To illustrate the surface details, we compute the dot product of the surface normals with either the z-direction or the local average normal as follows:

$$I(x,y) = \begin{cases} d = N(x,y) \cdot B(x,y) & \text{if } d < \tau \\ 1 & \text{otherwise.} \end{cases} \tag{4.11}$$

where τ is a user defined threshold, which controls the elimination of the flatter features and $B(x,y)$ is either $\langle 0,0,1 \rangle$ or the low frequency component of the normal map, $N_g(x,y)$, computed by Gaussian filtering the normal map. When the z-vector is used, it will depict the general shape of the surface such as curviness; on the other hand, if $N_g(x,y)$ is used, it will reveal the local surface details only. If the surface is flat and perpendicular to the camera, both methods will give similar results. In our results, extracted details are applied to the pecked out regions as a texture.

User Control. The user can control all the sigma and threshold values, and can use either the albedo, gray tones, or black and white for details. The stippling method described in the previous section can also be used for detail illustration by using $N_g(x,y)$ instead of the z-direction. Different methods for detail illustration are shown in Figure 4.15.

4.7 Results and Discussion

In this section, we show results on several inscription and non-inscription datasets, as well as comparisons to some previous work. Figure 4.16 demonstrates several styles of our epigraphy pipeline on one dataset. We demonstrate generating outlines of the inscriptions, adding surface details, giving relief effect to the grooves, stippling for shape depiction, and stippling to illustrate the surface details respectively. For comparison, Figure 4.17 shows a digital epigraphy done manually by archaeologists in two styles. As they reported, it took 5-6 hours to trace over the image and another 1-2 hours to add texture details to the pecked out regions.

(a) $B = \langle 0, 0, 1 \rangle$ (b) $B = N_g(x, y)$ (c) $B = \langle 0, 0, 1 \rangle$ (d) $B = N_g(x, y)$

Figure 4.15: Detail illustrations for pecked out regions. **(a), (b)** details computed using Equation 4.11; **(c), (d)** details are depicted by random stippling.

In Figures 4.18 and 4.19, several previous works are compared to our method. Results from image-based line drawings are shown in Figure 4.18, where we compare our result to Coherent Line Drawings (CLD) by Kang et al. [86] and extended Difference of Gaussians (xDoG) by Winnemoller [180]. Even though xDoG handles the noise caused by surface roughness better than CLD, it fails when the variation in color is small, such as in the right column of the figure. To test 3D line drawing methods, we project the surface normals onto a 3D plane and use those projected normals to compute 3D contours. As seen in Figure 4.19, previous works cannot handle the noise well, and even when the surface is smoother it does not produce contours enclosing the inscriptions as needed for epigraphy.

Results on several other rock inscriptions are shown in Figure 4.20, where the last two datasets are deeply engraved stones different than other rock inscriptions. We also demonstrate our system's usability on non-inscription datasets in Figure 4.21.

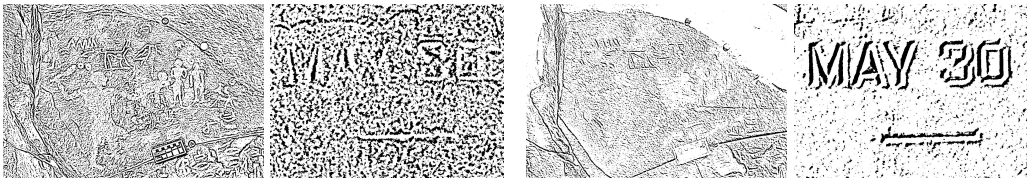
4.8 Conclusion

In this chapter, we presented a user-guided system to produce digital epigraphy from photometric datasets. The pipeline consists of segmenting the inscriptions from the rock surface and labeling them based on their carving techniques and rendering them in various non-photorealistic styles. We stated that our system efficiently and quickly (in less than

(a) Outlines (b) Details (c) Relief (d) Stippling (e) Stippled details

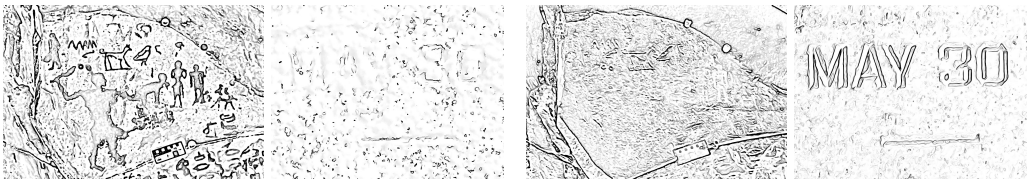
Figure 4.16: **(a)** Outlines of the inscriptions computed by the segmentation algorithm, grooves are drawn with thicker lines than pecked out regions. **(b)** Pecked out regions are textured by adding surface details using Equation 4.11. **(c), (d), and (e)** are closeup images showing relief, random stippling for shape depiction and stippling for detail illustration.

Figure 4.17: Digital epigraphy done manually by archaeologists using Adobe Photoshop and Illustrator.



(a) CLD on Albedo [86]

(b) CLD on Normals [86]



(c) xDoG on Albedo [180]

(d) xDoG on Normals [180]

(e) Our Results

Figure 4.18: Comparisons to previous image-based line drawing techniques.

(a) Ridges & Valleys [77]

(b) Suggestive Contours & Highlights [46]

(c) Apparent Ridges [80]

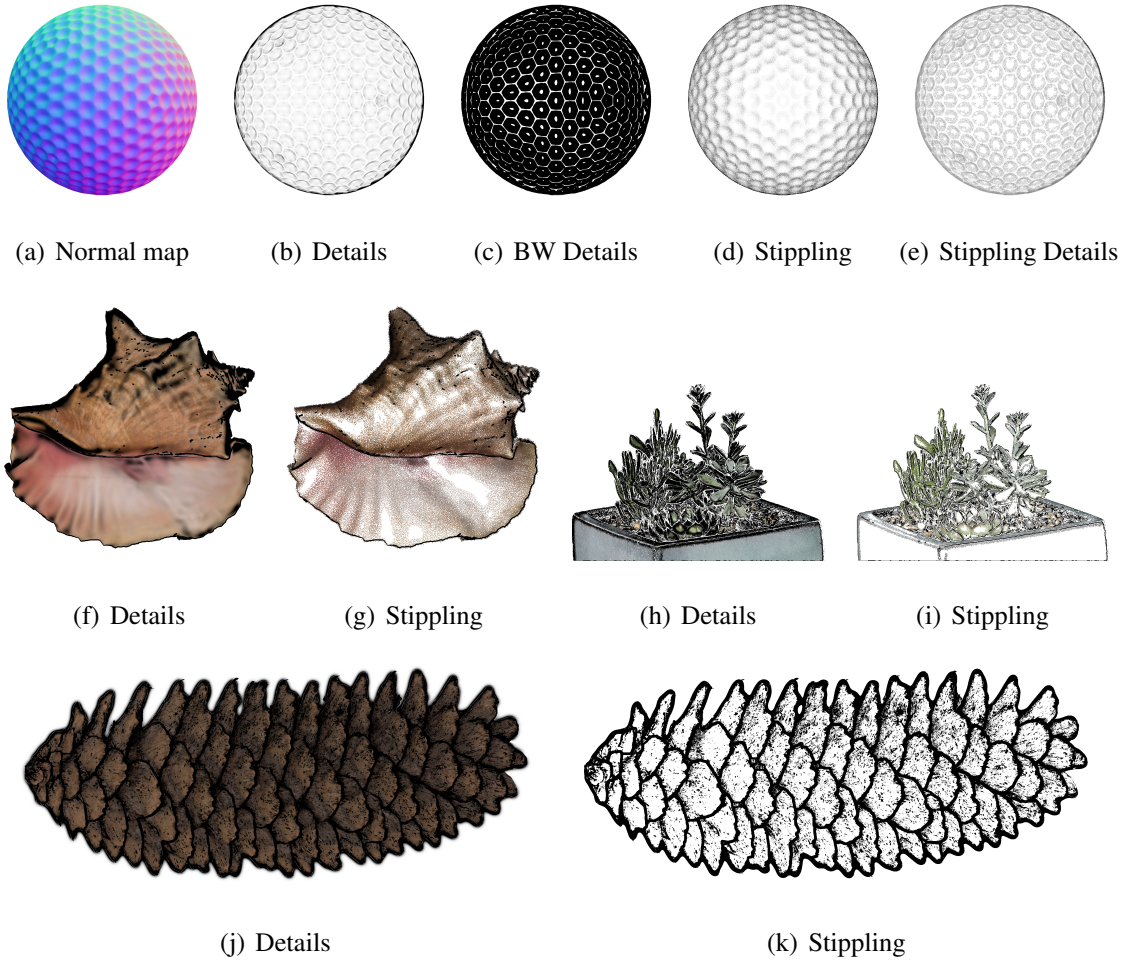
(d) Relief Edges [96]

(e) Isophotes

Figure 4.19: Comparisons to previous 3D model-based line drawing techniques.

Figure 4.20: Results on several other datasets.

10 minutes) produces results comparable to the epigraphic drawings done manually by archaeologists in hours. We compared our results to several previous works on both 2D and 3D line drawing algorithms. Finally, we showed results for several rock inscriptions, rock engravings, and non-inscription objects.



(a) Normal map (b) Details (c) BW Details (d) Stippling (e) Stippling Details

(f) Details (g) Stippling (h) Details (i) Stippling

(j) Details (k) Stippling

Figure 4.21: Several non-inscription objects are used with our system where we use the depth discontinuity to find the contours.

Chapter 5

Transferring Surface Details:

Learning Detail Transfer based on Geometric Features

Finally, we propose a method for realistically *transferring details* (specifically, displacement maps) from existing high-quality 3D models to simple shapes that may be created with easy-to-learn modeling tools. Our key insight is to use metric learning to find a combination of geometric features that successfully predicts detail-map similarities on the source mesh, and use the learned feature combination to drive the detail transfer. The latter uses a variant of multi-resolution non-parametric texture synthesis, augmented by a high-frequency detail transfer step in texture space. We demonstrate that our technique can successfully transfer details among a variety of shapes including furniture and clothing.

5.1 Introduction

Increasingly customizable characters in video games, massive multi-user immersive virtual environments, and advances in 3D printing have all driven the demand for high-quality

Figure 5.1: Our algorithm learns combinations of geometric features that predict the spatial arrangement of details on surfaces. **Left:** Input (target) mesh without details. **Center/Right:** Details from each source mesh (blue) are synthesized on the target mesh (pink).

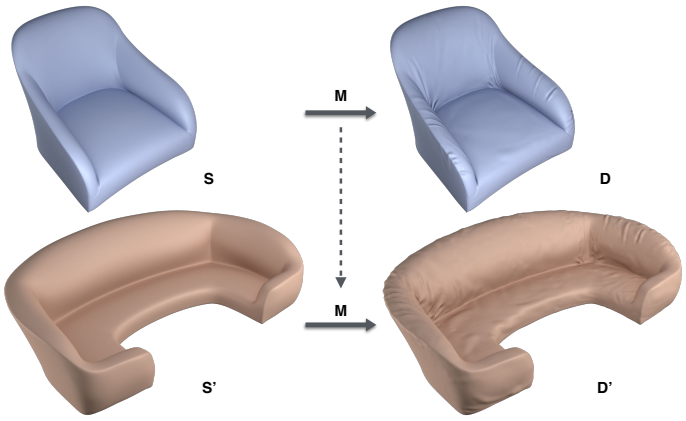


Figure 5.2: Our algorithm learns a mapping M between a coarse source shape S and its detail map D, and uses the same mapping to synthesize a detail map D' for a target mesh S'. **Left:** the source and target models without details. **Right:** source and target meshes are rendered with their corresponding detail maps.

3D models. It is time-consuming and expensive, however, to hand-model or scan 3D surfaces with realistic fine details. Furthermore, the number of high-quality 3D models freely available online or elsewhere is severely limited, forcing designers to generate models by hand or obtain them from repositories of low-resolution shapes. Even if a detailed shape is available in a desired *class*, edits that preserve fine-scale detail are tedious at best. For this reason, most high-quality models continue to be generated by highly-trained 3D artists who sculpt all the details manually using modeling tools such as ZBrush [136].

In this chapter we propose that the effort to create a single high-quality model may be amortized by transferring its details to a wide variety of easily-produced and widely-available low-resolution meshes. Specifically, we consider details that may be represented as displacement maps (so that their geometric extent is limited) and whose statistics are distinctive over different parts of the surface (so that they may be treated within the framework of texture analysis and synthesis). This encompasses surface features such as wrinkles, fabric patterns, wood grain, scratches, and cloth seams, which are critical to realism yet most typically missing in the low-resolution meshes created by amateurs. However, while there is a rich literature on texture analysis, the chief difficulty lies in the fact that realistic surfaces contain many *different* kinds of details. We must therefore learn which parts of the source model to draw from when synthesizing details on every point of a target model.

The key novelty of our work is to use *metric learning* to find a combination of geometric features that best predicts which regions of the source mesh have similar details. The learned metric is then used to evaluate source-target similarity, which guides the texture synthesis. This builds upon the insight (also exploited in the work of Mertens et al. [120] and Lu et al. [109]) that the patterns of details on a surface are usually correlated with large-scale geometry.

It is important to note that our method does not attempt to find a one-to-one correspondence, or even a fuzzy correspondence [93], between the source and target meshes. Instead, we perform an analysis *entirely within the source mesh* to learn which geometric features

are most important for predicting the similarity of local detail-texture statistics. One way of thinking about our system is to imagine that it solves an analogy problem (Figure 5.2): we learn predictive relationships M between a low-resolution source mesh S and its detail map D, and apply them to the target mesh S' to synthesize a detail map D'.

This approach lets us transfer details between models of significantly different size, complexity, and topology. For example, the center column in Figure 5.1 demonstrates that our algorithm can transfer details between similar shapes such as armchairs, while the right column illustrates that we can also successfully transfer the diamond pattern from a pillow to a chair. Moreover, notice that the diamond pattern occurs only on the front side of the pillow, and accordingly is transferred to only the front side of the chair. This is because our method learns that the presence of the diamond pattern is correlated with surface orientation. We demonstrate that the learned relationship between coarse geometry and texture detail is highly dependent on shape class and context, highlighting our method's contributions beyond existing surface correspondence and texture synthesis algorithms.

Contributions. The major new components of our work described in this chapter are

- learning the relationships between a wide range of geometric features and the statistics of a detail map expressed as a texture, via metric learning;
- transferring surface details between different 3D shapes in a non-parametric texture synthesis framework;
- combining coarse- and mid-scale synthesis over the surface with fine-scale synthesis in texture space, yielding an algorithm with the stability of the former and the quality and efficiency of the latter; and
- making our database of high-quality 3D models generated by artists available to the public, augmenting the currently-limited number of high-quality models freely available online.

5.2 Related Work

Texture Synthesis. Texture synthesis is a long-standing topic in the computer graphics literature; Wei et al. [174] present a comprehensive survey. The original study of texture synthesis was in the 2D image domain [69, 50, 175] and was later extended to synthesis of color patterns over surfaces [164, 176]. More recent work in texture synthesis considers volumetric surface details [24], mesh quilting [194], using terrain as a height field [193], feature-aligned texturing [187], material interpolation for appearance synthesis [47], and self-tuning optimization [88].

While most of the existing techniques are based on a Markov random field (MRF) assumption, i.e., that the texture pattern is both *local* and *stationary*, several methods have been proposed to handle non-stationary textures. Lu et al. [109] correlate the distributions of texture appearance to some simple context parameters, such as ambient occlusion and principal directions. Rosenberger et al. [141] automatically generate control maps for layered textures. More recently, Chen et al. [38] adapt the PatchMatch algorithm to 3D and transfers textures from one geometry to another with the help of some geometric features.

The previous work most relevant to our approach is that of Mertens et al. [120]. It uses canonical correlation analysis (CCA) to find correlations between RGB colors and geometric features, attempting to explain texture variations caused by processes such as dust accumulation and weathering. In contrast, our work employs a wider set of recently-developed geometric features along with the framework of metric learning, allowing meaningful correlations to be learned for coarser models and more complex geometry/texture relationships — see Figure 5.15 for a comparison. In addition, our method better handles structured textures such as wrinkles and sewing stitches, as compared to the highly-stochastic textures for which the work of Mertens et al. [120] is best suited.

Shape Correspondence. A key goal of our system is to establish which points on the source shape should be considered when synthesizing detail at each point on the target.

The simplest version of this problem would be to compute a one-to-one correspondence between the source and target meshes, and a long line of research has been devoted to this problem [166]. Kim et al. [93] extend this line of work to consider *fuzzy* correspondences among a set of 3D models, with the goal of facilitating the exploration of shape collections. Solomon et al. [148] compute a fuzzy map from the source surface into the target by minimizing the distortion between their distance fields. However, most of existing techniques for shape correspondence require the shapes to have the same or similar global structure, which severely restricts their applicability to detail synthesis. Instead, our learned metric can be used to effectively compute a similarity score between *every* pair of points on the source and target shapes. This retains some notion of correspondence if the shapes are similar, while smoothly degrading if the shapes are dissimilar in overall structure.

Geometry Synthesis. One application we consider is the synthesis of wrinkle and crease patterns on cloth, furniture, and other shapes. Golovinskiy et al. [61] synthesize facial wrinkles assuming both input and training 3D face models are perfectly aligned. They segment these aligned faces into regions and calculate statistics on each of them to synthesize new wrinkles for each region separately. Wang et al. [171] synthesize wrinkles for cloth which closely fits the body, based on analysis of simulated examples. Rohmer et al. [140] generate dynamic wrinkles by using the stretch tensor of the coarse cloth animation as the guidance.

The broader problem of shape synthesis by example has received considerable attention in other forms. One line of research focuses on part-based synthesis of new models from shape repositories [81, 188, 6]. Other methods leverage symmetry information to generate complex shapes from small examples [133, 26, 76]. Recently Ma et al. [112] use an analogy-driven framework to transfer style from an exemplar to a target model. In their framework, the source and the exemplar models need to have the same structure to build dense point-to-point correspondences. Furthermore, the input source and target shapes in their cases

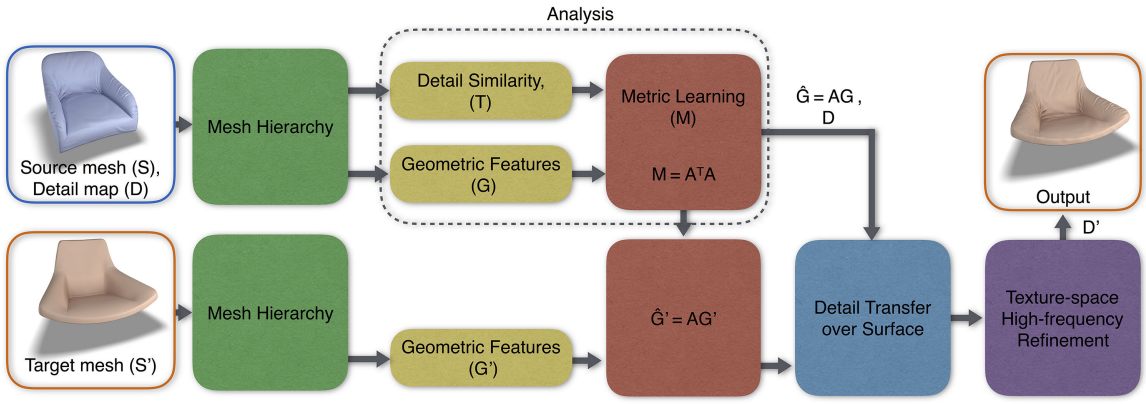
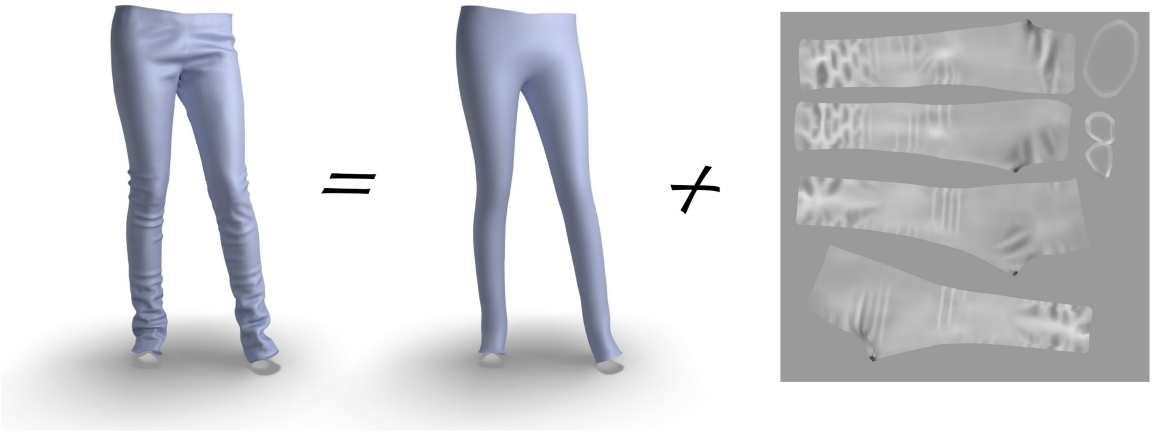


Figure 5.3: Algorithm overview.

have to be of the same style to allow the computation of source-to-target analogy assembly. In our framework, instead of computing full correspondences, we transfer details from the source shape to the target shape via learning from geometric features, which works for more general input shapes.



Source **Low-poly Mesh** **Displacement map**

Figure 5.4: A source mesh represented by a low-polygonal mesh and a displacement map.

5.3 Algorithm

In order to guide our texture synthesis algorithm, we need to know which areas on the source mesh should be considered for each point on the target. As hinted above, we dare not compute a full correspondence between source and target: that would leave our method fragile in the face of changes to gross shape, size, and topology. Instead, we argue that points on the target should draw from *geometrically similar* areas of the source. We can evaluate this similarity on the basis of a variety of geometric features, but this prompts another question: which features are most important? Our insight is that feature importance is not universal — in some situations the details might vary with curvature (such as the wrinkles on the armrest of the armchair at center in Figure 5.1), in others with normals (such as the diamond pattern on one side of the pillow at right in Figure 5.1), and in still other situations the variation might be well-predicted by one of the more recent feature descriptors such as the wave kernel signature (WKS) [9]. So, we must learn feature importance for each *particular* source mesh and every *particular* detail texture.

This motivation gives rise to the pipeline in Figure 5.3. We begin with a source triangle mesh S and its detail map $D\colon S \to \mathbb{R}$, where values in D represent displacements from the base mesh S along surface normals — see Figure 5.4. For both S and the target mesh S', we use repeated remeshing to create a hierarchy of lower-resolution versions.

Our key new step is to analyze which regions of D are self-similar and learn a metric M that predicts these similarities on the basis of geometric features G. The learned metric M is applied to target features G', allowing us to evaluate the similarity of points on the source and target. This, in turn, is used to guide a multi-resolution texture synthesis algorithm, which yields the target detail map D'. In a refinement step, we transfer the highest-frequency details from D directly to D', operating in texture space rather than on the mesh.

5.3.1 Input Preprocessing

Canonical Scale and Orientation. Our first preprocessing step is to scale each mesh consistently with the others, and orient it such that its semantically-meaningful "up" direction points along the positive y axis. While we perform this manually, we believe that automatic algorithms could perform nearly as well [58].

Mesh Hierarchy. We create a multi-scale mesh hierarchy for both the source and target meshes. We use an isometric remeshing algorithm proposed by Botsch and Kobbelt [27] to create multi-resolution mesh hierarchies for both the source and the target meshes. Specifically, we use the implementation provided by Möbius and Kobbelt [123] and create four-scale hierarchies in which the average edge length of each level is equal to half the average edge length in the previous level.

Tangent Frame. Our texture synthesis algorithm operates one patch at a time, and so we need to define the orientations of patches that will be used for search and synthesis (see Figure 5.5a-b). The first step is to define the orientation of a tangent frame at each vertex on the surface. A variety of strategies could be used, including using the gradient of an artist-provided uv map of texture coordinates, principal directions, or a user-specified vector field [164]. However, all of these strategies in practice lead to inconsistent orientations between similar patches on the source and target meshes. Exploring the problem of defining consistent tangent frames between meshes is an interesting avenue for future work, but for the purposes of this project we adopt a simple strategy that appears to work in most cases: we project the constant vector $(1, 0, 0)$ into the plane perpendicular to the surface normal n, using that as the first tangent vector u. The other vector v is defined as $n \times u$. While this results in some discontinuities, we have not found this to be a problem in practice.

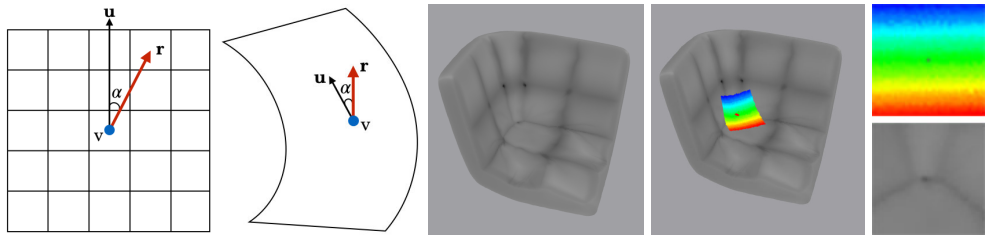
5.3.2 Analysis

Our first step is to analyze how features in the detail map D correlate with the geometry of the source surface S. This proceeds in three steps: (1) determining which regions of D have similar textures; (2) computing a vector of geometric features G at each point on the surface; and (3) applying a metric learning algorithm to determine a linear transform of G that best matches the texture similarities. In the detail transfer stage (Section 5.3.3), this metric will be used to select patches on S whose details can be transferred to different points on S'.

Detail Features

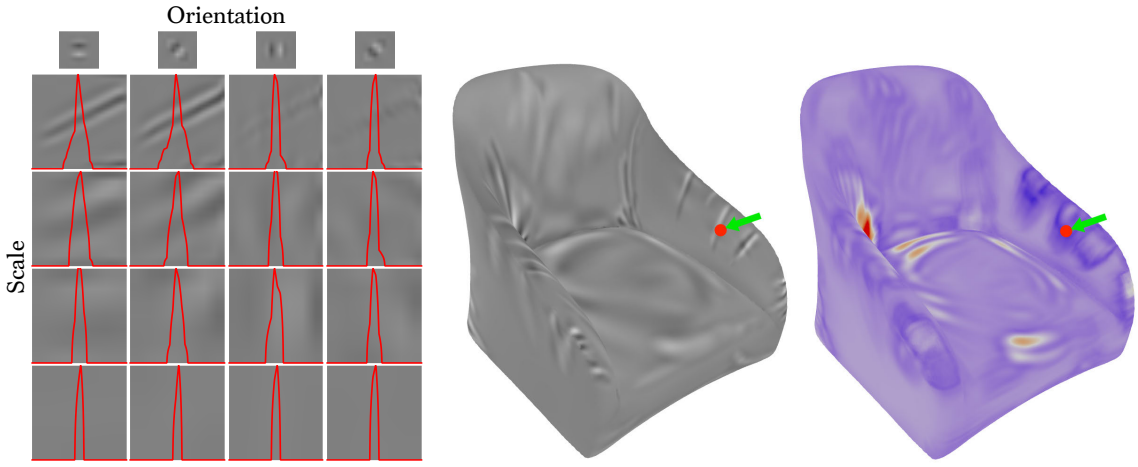
In order to determine which regions of the surface have similar detail texture, we compute a "detail feature" for each vertex on the surface, characterizing the statistics of the texture in its neighborhood. We begin by mapping a 2D square patch on the tangent space around each vertex $\mathbf{v}$ onto the surface S via an approximate exponential map, as shown in Figure 5.5. We use a technique similar to that proposed by Melvaer and Reimers [119]. We choose this parameterization at each $\mathbf{v} \in S$ because in the smooth case it maps the tangent plane at $\mathbf{v}$ into S with minimal distortion about the center point [70].

We then compute the texture statistics of each unfolded 2D patch. We use the low-dimensional feature vector proposed by Golovinskiy et al. [61], which consists of standard deviations of the histograms of steerable-pyramid filter bank outputs [69] at four orientations and four scales, along with a high-pass filter. Hence, the detail map around each vertex $\mathbf{v}_i$ is identified with a 17-dimensional feature t_i. Figure 5.6 shows the steerable pyramid outputs for the patch around the selected vertex marked in Figure 5.6c, along with the detail similarity between the selected point and all other points on the mesh. Histograms of the steerable pyramid are overlaid to demonstrate the varying standard deviation over different scales and orientations.



(a) (b) **S** (c) (d) (e)

Figure 5.5: Exponential mapping used to sample around vertex **v**. **(a)** Tangent space of a surface **S** around vertex **v** with the tangent vector u and the relative location vector r of a pixel; **(b)** corresponding u and r vectors on the 3D surface **S**; **(c)** example 3D model with detail map visualized as vertex colors; **(d)** exponential mapping around the corner of the seat; **(e)** corresponding 2D patches: a synthetic color mapping and actual sampled patch around the selected vertex.



(a) (b)

Figure 5.6: Computation of detail features. **(a)** The steerable pyramid of a patch around the vertex highlighted in **b** for 4 scales (rows) and 4 orientations (columns), overlaid with their histograms; **(b)** a detail map with one point highlighted, together with its similarity (blue = similar, red = dissimilar) to all other vertices on the shape.

Geometric Features

We compute the following geometric features at each vertex of the source mesh:

- **Curvature (C):** The two principal curvatures, along with the mean curvature, Gaussian curvature, and L^2 norm of the two principal curvatures.
- **Height (H):** The y-coordinate of each vertex.
- **Normals (N):** The surface normal at each vertex.
- **Shape diameter function (SDF):** A measurement of local object diameter, useful for separating thin and thick parts of the object [145].
- **Distance to segmentation boundary (Seg):** The geodesic distance from each vertex to the closest segmentation boundary obtained using SDF [145], normalized by the maximum distance within the each segment as described by Chen et al. [40].
- **Statistics on geodesic distances (Geo):** The mean, median, maximum, standard deviation, tenth percentile, and ninetieth percentile of geodesic distances from each vertex to all others [40].
- **Relative *xyz* coordinates (Box):** Each vertex's coordinates with respect to the center of the bounding box, normalized by the size of the corresponding dimension of the bounding box.
- **Heat kernel signature (HKS):** Measurements of the dissipation of heat from a point onto the rest of the shape over time [152], sampled at 20 points in time.
- **Wave kernel signature (WKS):** The average probability to find a particle of a given energy at a given point [9], sampled at 20 points on the frequency range.

Concatenating all of these features, we obtain a 60-dimensional geometric feature vector g_i at each vertex $\mathbf{v}_i$. Figure 5.7 visualizes each geometric feature for an armchair. A

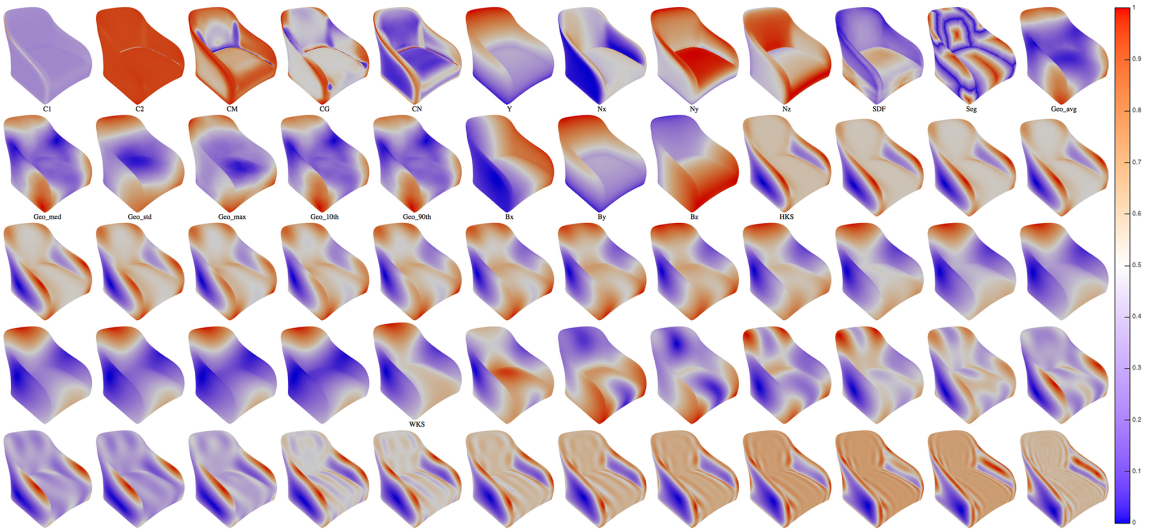


Figure 5.7: Visualizations of geometric features for an armchair. Each image visualizes one of the 60 geometric features, with minimum and maximum values of each feature mapped to blue and red, respectively.

diverging color map [125] is used, with blue mapped to the minimum value and red mapped to the maximum.

Metric Learning

Measuring distances or similarities between data is ubiquitous in machine learning, pattern recognition, and data mining, but it is generally difficult to tailor metrics for specific problems. This has led to the emergence of metric learning, which automatically adjusts a distance or similarity function using training data [99, 17].

In this chapter, we use a supervised global distance metric learning method to find a transform to be applied to our geometric features, such that distances between transformed features are predictive of distances between statistics of surface details. Although the expressive power of a global metric is limited relative to localized alternatives, the resulting convex objective is easier to optimize and suffices for our application. Furthermore, computation

of a single global metric is more resistant to over-fitting relative to nonlinear and local methods [17].

We define G to be a geometric feature matrix, with entry G_{ij} corresponding to the i-th feature of the j-th vertex on the source mesh. Additionally, we define the detail similarity matrix T to have entries $T_{ij} = \langle t_i, t_j \rangle$, where t_i is the texture feature vector of the i-th vertex. Intuitively, assuming we subtract the mean texture features ahead of this step, we can think of T_{ij} as measuring the similarity of vertices i and j with respect to the texture features.

We wish to learn a metric on geometric features that mimics the metric on the t_i. Formally, we can think of this problem as learning a distance function $d(\cdot, \cdot)$ between columns of G that imitates the distances encapsulated in T. For simplicity, we restrict ourselves to *Mahalanobis metrics* of the form

$$d(g_i, g_j) = (g_i - g_j)^\top M \, (g_i - g_j), \tag{5.1}$$

for some matrix $M \succeq 0$; the positive-definite constraint is needed for our metric to satisfy the triangle inequality.

We propose learning M via the following optimization problem:

$$\begin{aligned} \min_M \quad & \|T - G^\top M G\|_{\text{Fro}}^2 + \lambda \|M\|_1 \\ \text{s.t.} \quad & M \succeq 0. \end{aligned} \tag{5.2}$$

We solve this convex problem using CVX [64, 63]. The first objective term encourages the inner products of geometric features with respect to M to look similar to dot products of the texture features. The second objective term, modulated by a regularization parameter $\lambda \geq 0$, promotes *sparsity* of M [17], in this case reflecting the intuition that most features are uncorrelated and hence should have entry $M_{ij} = 0$. The examples in this paper use $\lambda \equiv 0.1$, which worked effectively for all of our experiments.

We experimented with a variety of metric learning alternatives, most notably a formulation minimizing differences of L^2 distances instead of inner products. While still convex, the resulting optimization problem is considerably more difficult to solve because the quadratic terms become dense in the optimization variables. Our simpler formulation yielded similar results in far less computation time.

Comparison to Context-Aware Textures. Lu et al. [109] propose a method for capturing time-varying textures and context parameters (a few geometric features) to control texture transfer. They first compute a diffusion map to represent the time-varying texture with a set of orthogonal functions on vertices. They then compute a correlation by taking the inner product of the diffusion map and context parameters (defined by ambient occlusion, signed mean curvature, principle curvature directions, normals in optimal and non-optimal surface directions). They only use these correlations to select the most important context parameter, since the numerical values of these correlations are not meaningful. To transfer time-varying textures, they provide a user interface in which the user can control the transfer by selecting a context parameter. We found that this approach is limited to mostly stochastic textures resulting from phenomena such as rusting, weathering, or chemical agents, for which the color variation can be simply explained with a single geometric feature such as ambient occlusion. We provide an empirical comparison later in the chapter (Figure 5.15b).

Comparison to CCA. Mertens et al. [120] propose guiding geometric texture synthesis using canonical correlation analysis (CCA). Their work projects geometric features onto three directions that are predictive of RGB triplets, computed using a greedy sequence of eigenvector computations [66]. We found that this low-dimensional projection and use of color rather than texture descriptors is not as good as metric learning at taking advantage of the rich relationships between shape and texture. Furthermore, our formulation (5.2) is globally optimal, allows for sparsity-based regularization, and is specifically tuned to the

specific task of computing *distances* between features. We provide an empirical comparison to CCA later in the chapter (Figure 5.15c).

5.3.3 Detail Transfer

Transferring details from one surface to another can be thought of as a texture synthesis problem. In the literature, there are various works on texture synthesis in both 2D and 3D. Although 2D domains have regular grid structures that enable fast processing, projecting back into 3D introduces distortion. On the other hand, achieving high resolution in 3D requires higher computational complexity than in 2D. To leverage the advantages of both methods, we propose a hybrid approach: first transferring details between 3D surfaces, then post-processing the synthesized detail maps, in texture space, to enhance fine details.

First, all the geometric features described in the previous section are computed for the full multi-resolution hierarchy of the target mesh S'. At this point, we have a new *Mahalanobis* distance function to compute geometric similarity between points on the source and target meshes, as follows:

$$d(g_i, g'_j) = (g_i - g'_j)^\top M (g_i - g'_j) \tag{5.3}$$

$$= \|Ag_i - Ag'_j\|_2^2, \tag{5.4}$$

where the positive semidefinite matrix M is factored as $M = A^\top A$ using Cholesky decomposition. We apply the transformation matrix A to all geometric feature vectors of both the source and the target meshes to simplify subsequent processing.

Detail Transfer over Surfaces

We adapt a variant of non-parametric texture synthesis (which has been explored for both images [50] and 3D surfaces [164]) to transfer details from one surface to another. We use a hierarchical method in which the synthesis progresses from the coarsest to finest level

Algorithm Detail transfer from source to target.

function TRANSFER-TEXTURE(H_{src}, H_{tgt}, n)

H_{src}: mesh hierarchy for the source shape

H_{tgt}: mesh hierarchy for the target shape

n: patch size

compute sweep order on H_{tgt}

for each level in H_{tgt}, coarsest to finest:

upsample vertex colors

SYNTHESIZE-LEVEL(H_{src}, H_{tgt}, n)

reverse sweep order

SYNTHESIZE-LEVEL(H_{src}, H_{tgt}, n)

$n \leftarrow 2 * n + 1$

end for

end function

---

function SYNTHESIZE-LEVEL(H_{src}, H_{tgt}, n)

sample patches around evenly-distributed source vertices

for each target vertex, in sweep order:

if displacement weight $w_{disp} > 0.5$ **then**

skip the vertex

end if

sample a patch around the vertex

match features to find candidate source patches

match patches to find best candidate

blend best patch into target mesh

end for

end function

in a mesh hierarchy. We also propose using a sweep order based on the number of good candidates per vertex. In other words, vertices with fewer good candidate source patches should be synthesized first, because they have fewer degrees of freedom. The synthesis is guided by a two-step matching phase where both geometric features and already-synthesized details are considered. We also choose to synthesize a patch at a time, in preference to per-vertex synthesis, to both accelerate the process and preserve local coherency.

Our detail transfer algorithm is presented in the Algorithm . The main blocks of the algorithm are as follows:

- **Sweep order:** The number of good correspondences per target vertex is computed by thresholding the (transformed) distances $d(g_i, g_j)$ from that vertex to all vertices on the source mesh. A few vertices with the fewest good correspondences are selected as initial *seed vertices*. Those seed vertices are then used to determine the sweep order, based on the geodesic distance from the seed vertices to others. However, to reduce order dependence, we reverse the sweep order at each iteration.

- **Sampling:** At each level in the hierarchy, the source mesh is sampled evenly and exponential surface patches around those samples are stored along with their embedded geometric features, as explained in §5.3.2.

- **Matching:** A two-step matching algorithm is used to find the best-fitting patch from the source mesh to the target mesh. First, a set of candidate patches are found from the source mesh based on geometric feature similarity. Second, among those candidates the one which is the most similar to the already-synthesized part of the detail map on the target mesh is selected. In both steps, a translational refinement is applied when searching for the most similar feature or displacement patch.

- **Blending:** The best candidate patch is copied onto the target mesh and blended seamlessly by using a weighted average controlled by two weights, w_{loc} and w_{val}, inversely proportional to: (i) the distance from the center of the patch, x_c, to the pixel location, x_i,

and (ii) the difference between the new, d_{new}, and already synthesized, d_{curr}, displacement values, respectively. In particular, the per-vertex displacement weights w_{disp} are updated as follows:

$$w_{disp} \leftarrow w_{disp} + w_{loc}\, w_{val} \tag{5.5}$$

$$w_{loc} = f\big(\|x_c - x_i\| \,/\, (n/\sqrt{2})\big) \tag{5.6}$$

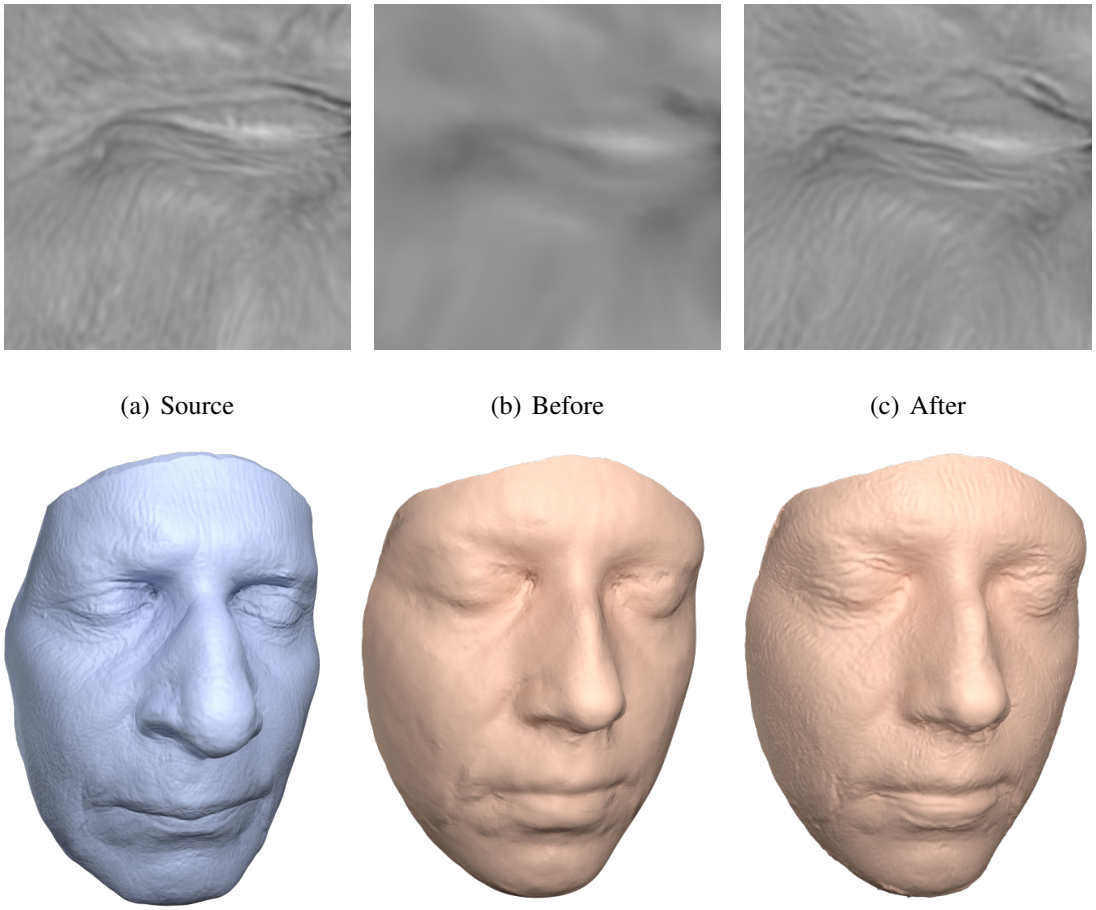
$$w_{val} = f\big(\|d_{curr} - d_{new}\| \,/\, d_{max}\big), \tag{5.7}$$

where n is the patch size, $f(x) = 2x^3 - 3x^2 + 1$ as in [164], and $d_{max} = 1$, assuming the displacement values are in $[0, 1]$.

Texture-space High-frequency Detail Transfer

Detail transfer over surfaces is limited by the finest mesh resolution in the mesh hierarchy. In order to transfer details with higher frequency than the mesh sampling, we propose a hybrid synthesis approach: detail transfer over surfaces (Section 5.3.3) followed by a refinement step in texture space (below).

The texture-space refinement begins by rasterizing the per-vertex displacements synthesized on the target mesh into a target texture map. We then consider small patches p_i from the source texture, centered at each vertex, along with their corresponding locations in the target texture. We warp each p_i to match the uv texture parameterization of the target, and further refine its translation to match the target texture as well as possible. Finally, we transfer only the high frequencies from p_i to the target, where the cutoff frequency is set according to the median edge length in uv texture space. The results before and after this refinement step are shown in Figure 5.8.



(a) Source (b) Before (c) After

(d) Source (e) Before (f) After

Figure 5.8: High-frequency refinement of transferred texture. **(a)** Close-up image from a source detail map. **(b)** Before the refinement step, the transferred texture is blurrier than the original. **(c)** After texture-space refinement, high-frequency details are restored. **(d)** Source model with details, **(e), (f)** target model with details before and after the refinement step.

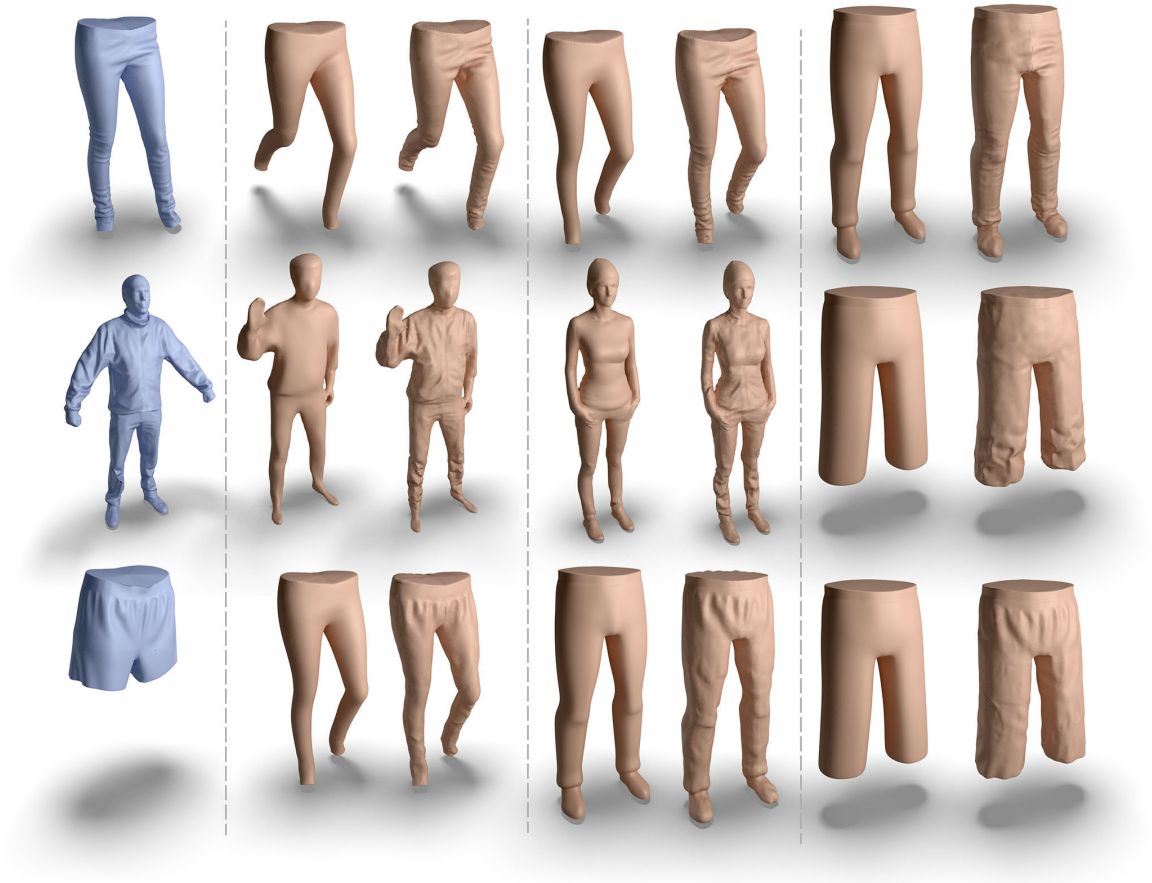


Figure 5.9: Detail transfer for clothing models. For each source mesh (left column), we transfer details to three different target meshes.

5.4 Results

We show the results of our detail-transfer pipeline on several classes of objects.

Wrinkles on Clothing. In Figure 5.9, we demonstrate that our framework can be used to add a realistic appearance of fabric to clothing models. We not only transfer details between shapes in the same category, such as pants, but also transfer details from a full-body scan to a single piece of clothing. In this case, the automatically-learned metric is able to restrict the textures that are selected to ones appropriate to the target. We believe that this framework has a wide variety of applications, perhaps extending to synthesizing details in areas with poor visibility on a 3D scanned model.

In all, we generated a significant number of high-quality 3D clothing models from 6 source models and 11 target models.

Style Transfer on Furniture. Figure 5.1, Figure 5.10, and 5.15e show some of the results of transferring details including wrinkles, seams, and fabric patterns among models of upholstered furniture. In each case, the source mesh is shown in blue, with subsequent pink models being the targets.

The results show that we are able to capture the semantically-meaningful distribution of details on the surface. For example, similar kinds of wrinkles and seams tend to appear in corresponding places on the surface, even in the presence of large differences in overall shape. Note also that our non-parametric texture synthesis algorithm is able to handle patterns ranging from structured to completely stochastic.

We also demonstrate in Figure 5.11 that our algorithm can be used on low-polygonal shape collections to convert them into high-quality models from little data (i.e. from a single armchair to four sofas with different sizes as shown in the figure) and produce similar-looking but not identical details which is important for a realistic appearance of details like wrinkles.

In the furniture category, 17 artist-designed source models were used to create a number of high-quality target models.

Other Transfer Results. We also demonstrate a few results on shapes from other classes. In Figure 5.12, we use the armadillo as a source mesh and transfer the details to a bunny and a pair of pants. Our algorithm transfers the square patern from the armadillo's shell to the bunny's back, and the bumpy pattern on the armadillo's legs to the pants. Additionally, we apply our algorithm on few faces from the Merl face database [61], as shown in Figure 5.13.

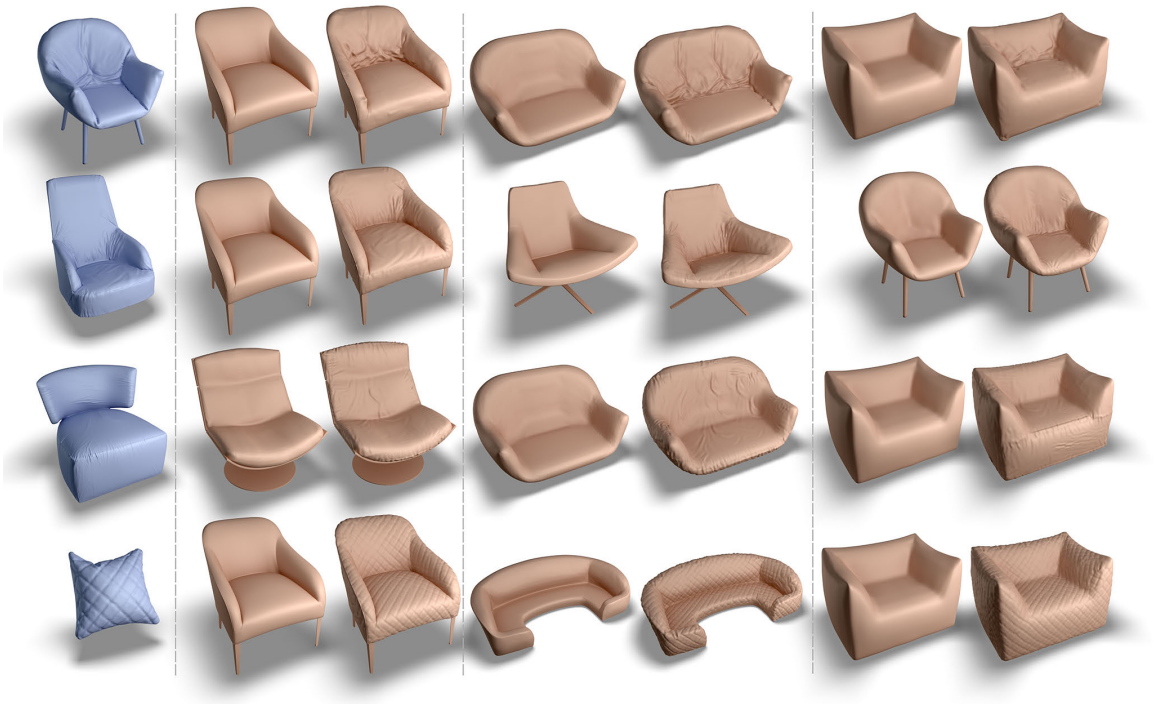


Figure 5.10: Detail transfer for furniture models. For each source mesh (left column), we transfer details to three different target meshes.

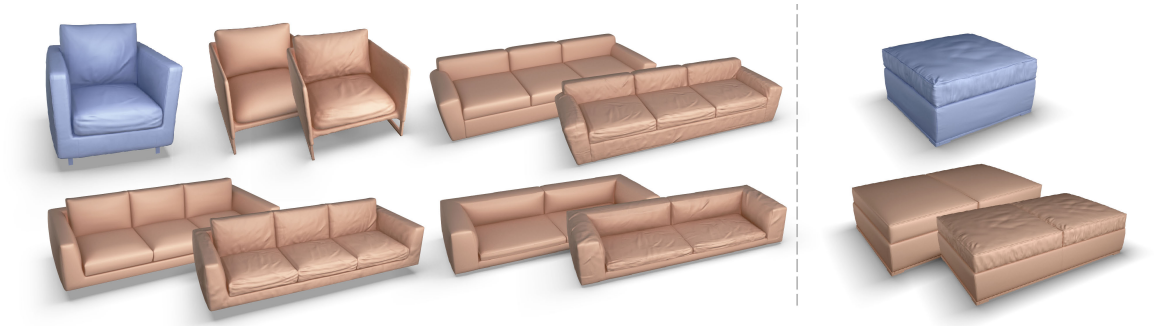


Figure 5.11: Details from the source meshes shown in blue are transferred to multiple target models to demonstrate that our algorithm produces similar but not identical details.

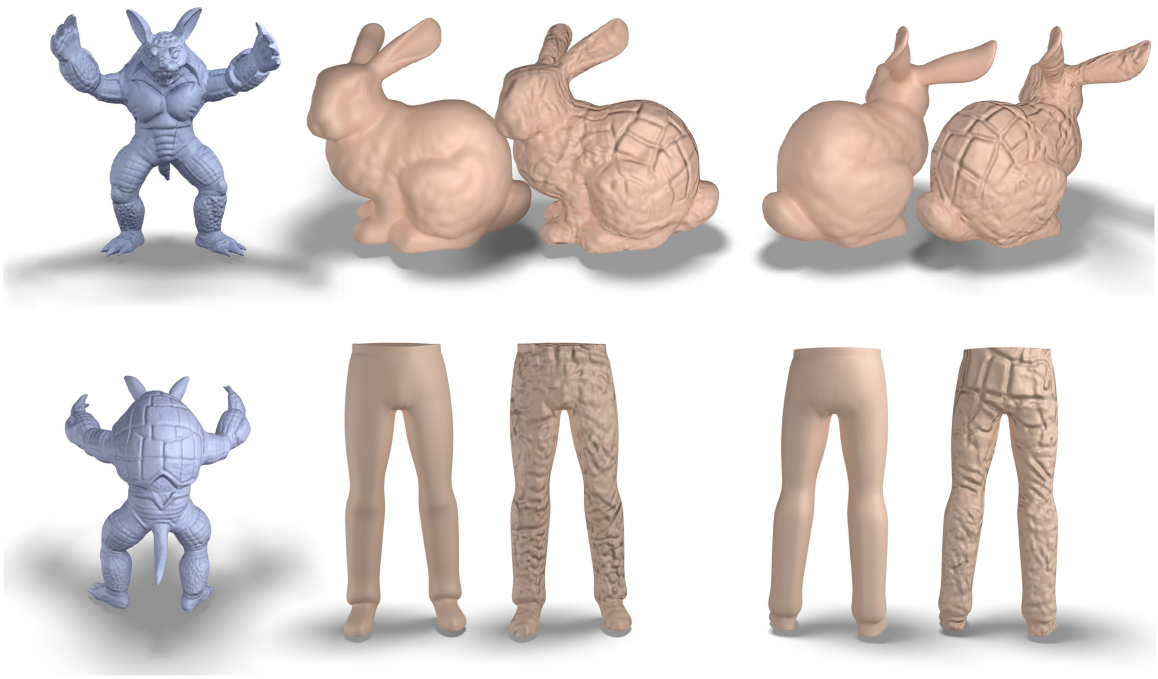


Figure 5.12: Detail transfer from Armadillo to Bunny and pants.

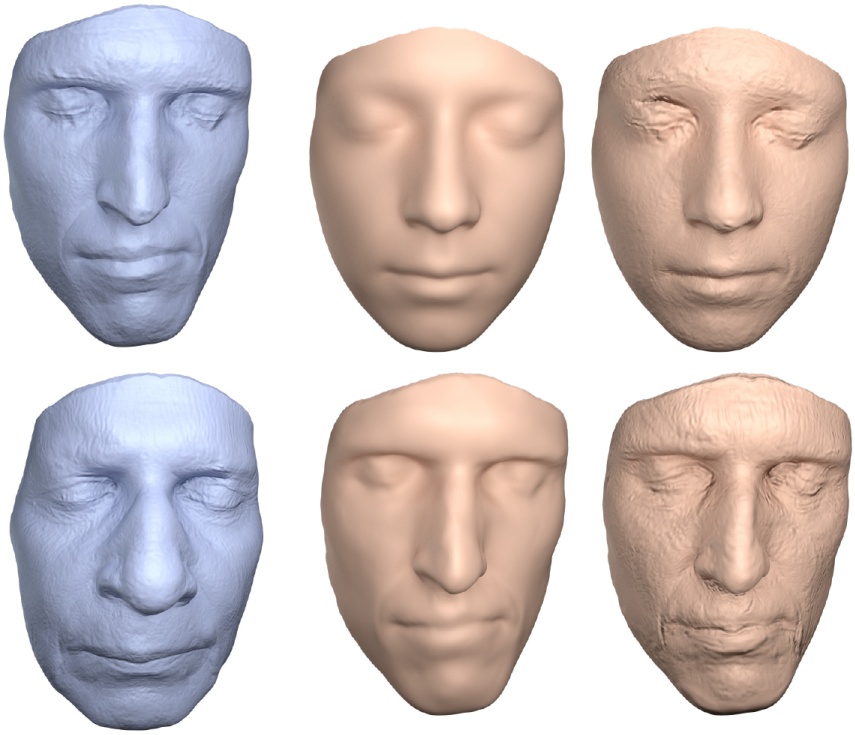
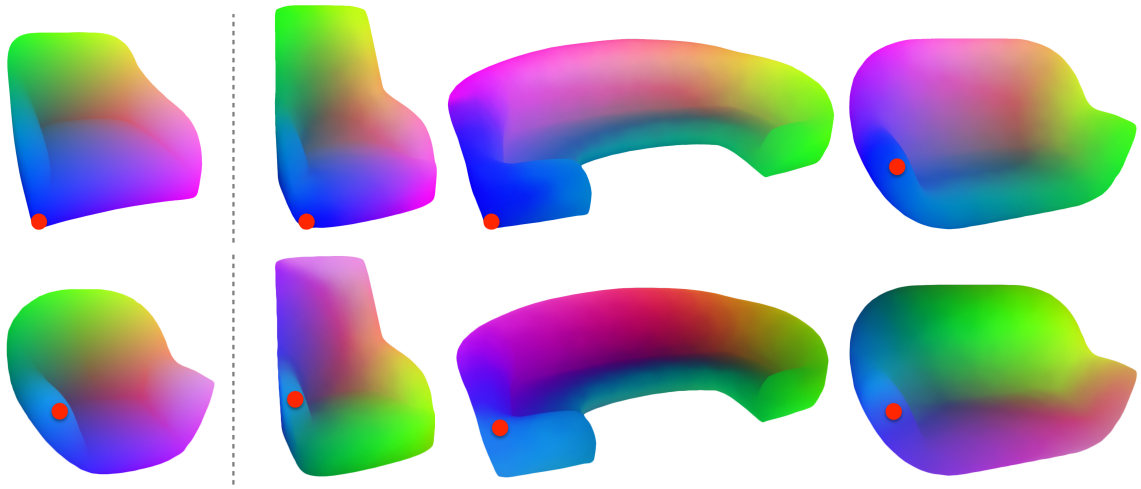


Figure 5.13: Detail transfer on faces from Merl Database from the source models in blue to the target models in pink.



Source **Targets**

Figure 5.14: Full correspondences produced by using Solomon et al. [148]. Red points are set manually as constraints. False coloring is used to visualize correspondences between the source and target meshes. The same correspondences are used in Figure 5.15d for comparison.

5.5 Experiments and Comparisons

Comparison to Other Algorithms. We compare our algorithm to several alternatives:

1. **No Metric Learning:** We disable the metric learning part of our algorithm, giving equal weight to each geometric feature. Specifically, we subtract the mean of each feature, and then divide by its standard deviation across the shape. This method can also be understood as a version of MeshMatch, proposed by Chen et al. [38], where the geometric features are equally weighted. As shown in Figure 5.15a, when the algorithm relies on all the geometric features equally and the source and target meshes have dissimilar local geometry, it fails to transfer meaningful details. For example, as shown in the middle column, it transfers wrinkles from the corner of the source armchair to the middle of the couch.

2. **Lu et al. [109]:** Although the method proposed by Lu et al. [109] focuses on time-varying textures, we adapt their idea of context-aware textures to compare to our method. Specifically, we compute the eigenvalue decomposition of the learned metric M and use the geometric feature (their "context parameter") corresponding to the highest eigenvalue. In other words, we transfer details using guidance from a single geometric feature to explain the correlation between geometry and details of the source mesh. In Figure 5.15b, the rightmost column shows that this technique works if the details are highly correlated with a single geometric feature. In this example, the highest eigenvalue corresponds to mean curvature, which explains the distribution of the wrinkles on the surface very well. On the other hand, the first and second columns show that this method is not sophisticated enough to discover details whose relationship with the underlying geometry is not captured by a single descriptor.

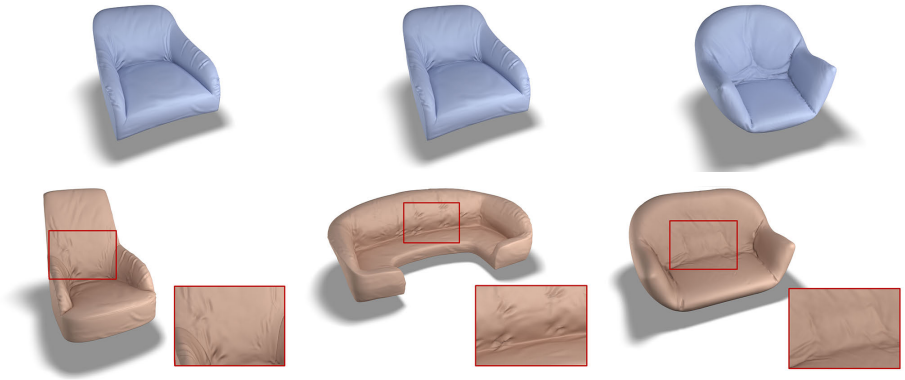
3. **Canonical Correlation Analysis (CCA):** Previous work by Mertens et al. [120] uses CCA to find correlations between RGB and geometric features (a smaller set than is used in this paper). To provide fair comparison to metric learning, we use CCA to find the relationship between the detail-texture descriptors described in Section 5.3.2 and the geometric features listed in Section 5.3.2. We then use this correlation, instead of metric learning, to guide detail transfer. As shown in Figure 5.15c, CCA fails to capture a meaningful relationship between the details and the geometric features, and results in poor guidance.

4. **Mesh correspondence:** One could also imagine using a mesh correspondence algorithm as guidance for texture transfer. Figure 5.15d illustrates the advantages of metric learning over this more restrictive pipeline. In this experiment, we use the method of Solomon et al. [148] to compute a probabilistic map from the source surface into the target while minimizing geodesic distortion (regularizer $\alpha = 0.005$); we mark a single point as a constraint. We then use the 10 highest-ranking matches for each target point to guide

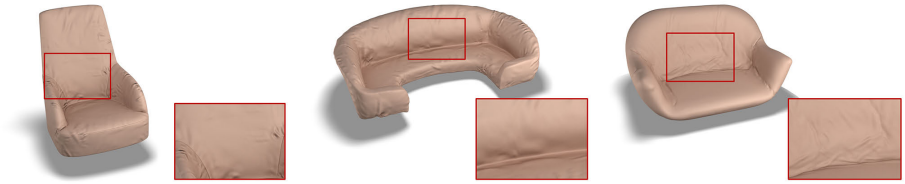
texture synthesis. Algorithms like [148] minimize distortion while seeking bijective maps. As illustrated in the results of this test, these properties can be problematic for texture synthesis. Textures may be better aligned with features captured by our metric, even if this induces stretch of the source onto the target. Furthermore, the search for a bijective mapping is subject to local minima. Figure 5.14 shows some instances in which our one-point constraint led to smooth but counterintuitive maps; these maps fail to capture critical semantic relationships thanks to the geometric variability of the models.

We also experimented with turning off the geometric feature matching altogether. However, with no constraint on which regions of the source mesh should be used at different locations on the target, the texture synthesis algorithm quickly "got stuck" by gravitating towards a single texture, most typically the one with lowest variation. This is because the matching part of the algorithm tries to minimize the difference between adjacent detail patches, and completely flat patches minimize this cost function most effectively.

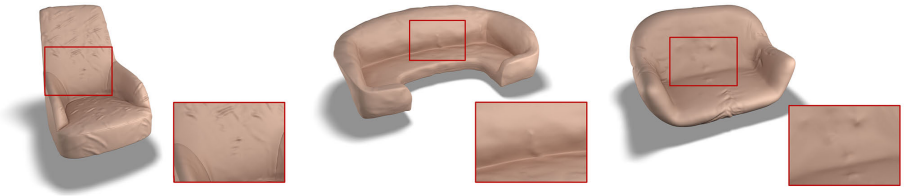
Significance of Geometric Features. To demonstrate the significance of different geometric features used for the detail transfer, we learn the relationships between the surface details and the geometric features, as explained in Section 5.3.2, for models in two different categories: (i) clothing, with six models, and (ii) furniture, with 17 models. We plot the average of the first eigenvectors (i.e., corresponding to the largest eigenvalues) of matrices M for both categories in Figure 5.16. To make the graphs more compact, related geometric features are grouped into one; for instance, the "curvature" data point (C) represents the total significance of all the features derived from surface curvatures, as listed in Section 5.3.2. The only exceptions are the Heat Kernel Signatures (HKS) and Wave Kernel Signatures (WKS), which are partitioned into low-frequency (HKS1 and WKS1) and high-frequency (HKS2 and WKS2) components. The error bars on each data point represent variance of the corresponding class of features across models within a category.



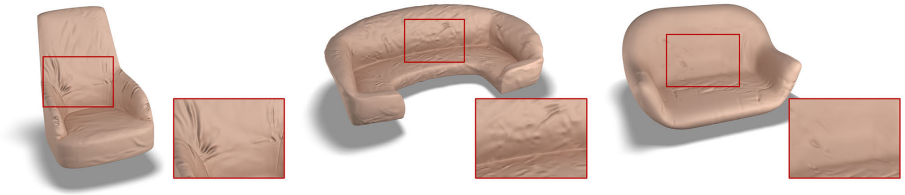
(a) No Metric Learning



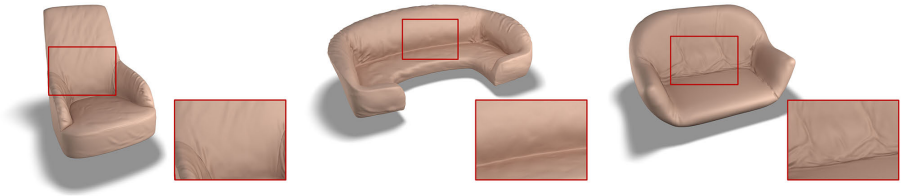
(b) Lu et al. [109]



(c) CCA, Mertens et al. [120]



(d) Surface correspondence, Solomon et al. [148]



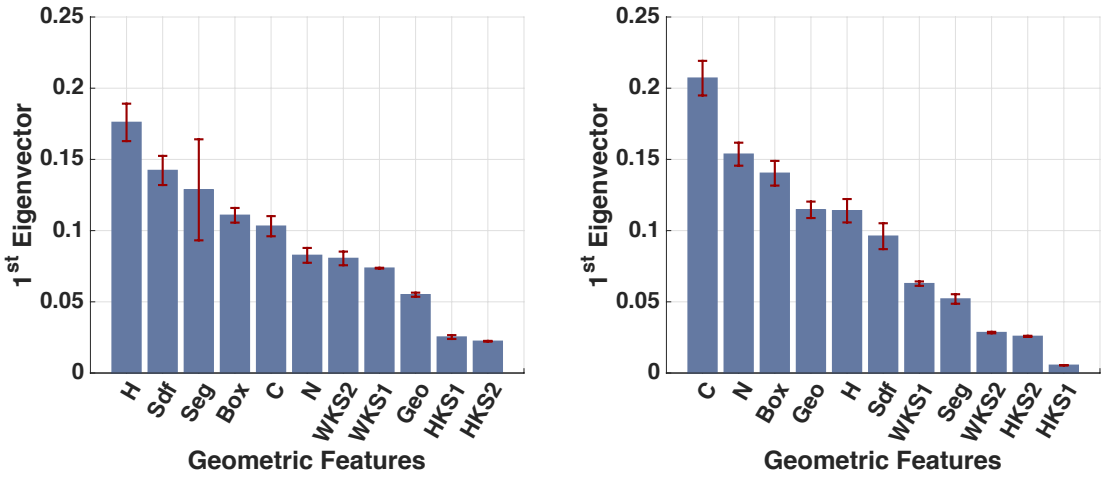
(e) Our results

Figure 5.15: Comparisons to several algorithms. Source meshes are shown at the top row in blue and the results are shown in pink.

We observe that the variation in the surface details for clothing is highly dependent on the height and SDF, while curvatures and surface normals are more important for furniture. On the other hand, the metric learning finds HKS features to be the least important for both categories. This can be explained by the presence of WKS features, since WKS is known to be more robust and distinctive compared to HKS.

Because meshes can vary in which geometric features are most strongly expressed, we also experimented with including the target mesh geometry into the metric learning — essentially, ensuring that the learned metric on the source mesh only includes features that are present and important on the target. To do this, we first find the principal components of the 60-dimensional geometric features of the target mesh, then project the source geometric features onto the first k of those principal components; we experimented with using $k = \{5, 15, 30, 45, 60\}$. We use these projected geometric features, instead of the original ones, in the metric learning and the detail transfer. While we believe that there may be situations in which this yields a benefit, we did not observe significant improvements by adding this step. A more thorough exploration of using target feature distribution to guide the metric learning would be an interesting avenue for future work.

Stability of Detail Transfer. Figure 5.17 shows the result of an experiment intended to evaluate how well our learning step can match the distribution of textures across a model. Specifically, we first transfer details from the model in **a** to the one in **b**. Then, we use the synthesized model in **b** as the source mesh and transfer details from **b** back to the model in **a** (ignoring its original detail map), yielding the result in **c**. We observe that our framework is stable enough to produce a plausible result (although, as with any texture-synthesis result, we would not expect perfect agreement).



(a) Clothing (b) Furniture

Figure 5.16: Significance of different geometric features, for **(a)** clothing and **(b)** furniture models. The y axis shows the average values of the first eigenvectors of learned matrices M, with the error bars in red representing variation of feature significance among the models in a shape category. Categories of geometric features are grouped along the x axis.

(a) (b) (c)

Figure 5.17: Experiment evaluating the stability of detail transfer. **(a)** Source mesh; **(b)** target mesh, with details synthesized from the source mesh; **(c)** source mesh, with details transferred **back** from the mesh in (b).

5.6 Conclusions

This chapter presented an algorithm to transfer surface details from one shape to another in a non-parametric texture synthesis framework, guided by learned relationships between the surface details and the geometric features of the source shape. We showed that there is no one global set of geometric features to guide the surface detail transfer across different shape categories: this is why a metric learning algorithm is needed to compute the appropriate weightings of different geometric features for each source shape and input detail map.

Discussion. The success of our method depends on finding a meaningful correlation between geometric features and the distribution of details across a surfce. This is most easily accomplished when the source and targets are in the same class, and indeed this was the case for which our algorithm was designed. When the source and target differ more drastically, we make the following observations:

- If the source has no regions closely matching the target, our algorithm will still find the *closest* match. As with all data-driven methods, a lack of training data can lead to suboptimal results.
- Because we use a rich feature set, including global features, even points with identical local geometry, such as the two armrests of an armchair, will have distinct features. This allows our algorithm to learn whether these regions should have similar or different textures.
- If two points have similar texture but different geometry, our algorithm will learn that similar geometric features are *not* predictive of texture (though, as mentioned above, "global" features may differ).
- If two objects come from completely different classes, our method may still produce intuitive results, such as in Figure 5.1, right, or Figure 5.12, bottom. In other situations, such as armadillo-to-armchair detail transfer, however, the result will not be meaningful.

Limitations and Future Work. One of the limitations of the detail transfer is the dependence on the tangent frame. Since (to our knowledge) there is no prior work that creates consistent tangent fields between two different 3D shapes, we rely on their initial alignment and the projection of a single global direction. Although we could search over different orientations during synthesis, this would result in higher computational complexity and inconsistently (and unnaturally) oriented details such as wrinkles. As future work, we would like to explore how to generate consistent tangent frames between shapes for texture synthesis over surfaces.

As with all MRF-based texture synthesis techniques, our algorithm performs less than ideally for highly-structured patterns. Specifically, neither the self-similarity of the detail map nor the geometric features that are used to transfer detail are precise enough to represent highly-structured repetitive patterns and exactly where they should appear on the surface. Exploring more sophisticated texture descriptors and geometric features tailored to structured patterns would be an interesting future direction.

While in this chapter we focused on small surface details, which are conveniently represented by displacement maps, recent work in 3D shape analysis has begun to focus on the understanding and transfer of larger-scale mesh "style" [112, 111, 105]. We believe that our work may form part of a toolbox for analyzing the expression of style across different frequencies in 3D models, and it would be an interesting direction to explore unified solutions for transferring geometric elements of all scales between meshes.

Chapter 6

Conclusion

The visual richness of computer graphics applications is frequently limited by the difficulty of obtaining high-quality, detailed 3D models. In this thesis, we address several research problems on capturing, processing, and synthesizing surfaces with high-quality details and propose solutions which are beneficial to many applications such as 3D printing, 3D content creation, video gaming, visual effects, and cultural heritage.

First, we focused on capturing high-resolution surface details via photometric stereo. We validated the captured data and corrected any misalignments if necessary to achieve the highest possible resolution from the dataset. We then analyzed the effects of several internal and external factors on various photometric stereo algorithms and proposed an iterative least squares approach for general purpose photometric stereo problem.

Second, we proposed a pipeline to combine two different types of captured data to achieve higher quality than either one of them alone. We used normals maps estimated by the photometric stereo algorithm mentioned in Section 2.3 and 3D scans captured with either one of the acquisition techniques mentioned in Section 3.1. Photometric stereo is good at capturing high-frequency details while they suffer from low-frequency bias. On the other hand, 3D scans are hard to obtain in very high resolution but they provide good coarse-scale shape. We offered a fully automatic solution to align, blend, and optimize these

two types of data without any prior information to obtain a high resolution 3D model with a coarse-scale shape of the 3D scan and high resolution details of the normal maps.

Third, we presented a user-guided system for converting the captured high-resolution photometric data into interpretable geometry-aware non-photorealistic illustrations. Specifically, we captured rough rock surfaces with shallow reliefs formed by carving into or scraping off the rock surface, making the problem very challenging for previous line drawing methods. We proposed using a segmentation algorithm which uses both the color and geometry information of the rock surface to handle the low signal-to-noise ratio of the datasets caused by weathered rough surfaces. We then classified each inscription based on their carving methods and non-photorealistically illustrated them in different styles.

Finally, we proposed a method for transferring surface details from existing high-quality 3D models to coarse 3D shapes to give them realistic appearances. We argued that we can learn a mathematical relationship between the geometry and the surface details of the high-quality 3D model and use this relationship to guide the detail transfer. We first transferred the mid-scale details over 3D surfaces then the finer details on texture space to benefit from the stability of the former and the quality and efficiency of the latter.

6.1 Future Work

We believe that there are various interesting future directions for the problems that we addressed in this thesis and we summarize them in three categories:

Capturing. Researchers have been working on sophisticated capture setups to have more control over the scene and reduce the external noise causing factors. For instance, Light stage proposed by Wenger et al. [177] provides more control over the illumination and repeatable results, however these type of acquisition setups are expensive, hard to maintain, take a lot of space, and are not portable or easily scalable. It is an interesting avenue to explore cheaper and scalable solutions. Moreover, real-time and mobile acquisition systems

have became more and more important as the data hungry algorithms emerge and perform better than their predecessors, such as deep learning. A unified real-time system which captures both shape and the appearance in high-quality would make collecting data practical and enables researchers to approach traditional computer graphics problems from a different angle.

Processing. In the literature, there is extensive amount of work on 3D shape analysis including shape matching, retrieval, segmentation, classification etc. [167, 160]. Those works, however, focus on coarse-scaled 3D models. We proposed a solution on segmenting surfaces with details captured with photometric stereo where the previous computer graphics techniques would fail. We leave solving other shape analysis problems for surfaces with details as future work.

Synthesizing. In this thesis, we focused on transferring details from a single source mesh (to arbitrary targets), because of the limited number of high-quality 3D models available as input. Even though our framework can be used to extend the available collections of high-quality meshes, the resulting styles will still be limited to those present in the original source collection. An interesting avenue would be to explore interactive tools for novice users to create high-quality mesh detail maps quickly, possibly exploiting inputs such as photographs. Furthermore, once there are more high-quality models available, we would like to explore transferring details from collections of shapes, instead of being limited to a single source mesh.

Bibliography

[1] Henrik Aanæs, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. Interesting interest points. *Int. J. Comput. Vision*, 97(1):18–35, March 2012.

[2] Austin Abrams, Christopher Hawley, and Robert Pless. Heliometric stereo: shape from sun position. In *Proceedings of the 12th European conference on Computer Vision-Volume Part II*, pages 357–370. Springer-Verlag, 2012.

[3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. PAMI*, 34(11):2274–2282, Nov 2012.

[4] J Ackermann, F Langguth, S Fuhrmann, and M Goesele. Photometric stereo for outdoor webcams. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 262–269. IEEE, 2012.

[5] Jens Ackermann and Michael Goesele. A survey of photometric stereo techniques. *Found. Trends. Comput. Graph. Vis.*, 9(3-4):149–254, November 2015.

[6] Ibraheem Alhashim, Honghua Li, Kai Xu, Junjie Cao, Rui Ma, and Hao Zhang. Topology-varying 3D shape creation via structural blending. *ACM Trans. Graph.*, 33(4):158:1–158:10, 2014.

[7] N Alldrin, T Zickler, and D Kriegman. Photometric stereo with non-parametric and spatially-varying reflectance. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[8] N G Alldrin, S P Mallick, and D Kriegman. Resolving the Generalized Bas-Relief Ambiguity by Entropy Minimization. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7. IEEE, 2007.

[9] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Proc. ICCV Workshops*, pages 1626–1633, Nov 2011.

[10] Alberto Bartesaghi, Guillermo Sapiro, Tom Malzbender, and Dan Gelb. Three-dimensional shape rendering from multiple images. *Graph. Models*, 67(4):332–346, July 2005.

[11] Adrien Bartoli. Groupwise Geometric and Photometric Direct Image Registration. In *British Machine Vision Conference*, pages 157–166, 2006.

[12] R Basri and D W Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[13] Ronen Basri, David Jacobs, and Ira Kemelmacher. Photometric Stereo with General, Unknown Lighting. *International journal of computer vision*, 72(3):239–257, June 2006.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin / Heidelberg, 2006.

[15] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001.

[16] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. Graph.*, 33(4):159:1–159:12, July 2014.

[17] A. Bellet, A. Habrard, and M. Sebban. *Metric Learning*. Morgan & Claypool, 2015.

[18] James R. Bergen, P. Anandan, Th J. Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. pages 237–252. Springer-Verlag, 1992.

[19] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2):20, 2013.

[20] Sema Berkiten, Xinyi Fan, and Szymon Rusinkiewicz. Merge2-3D: Combining multiple normal maps with 3D surfaces. *3DV 2014, International Conference on 3D Vision*, page 8, December 2014.

[21] Garry Berkovic and Ehud Shafir. Optical methods for distance and displacement measurements. *Advances in Optics and Photonics*, 4(4):441–471, December 2012.

[22] Fausto Bernardini and Holly Rushmeier. The 3d model acquisition pipeline. *Computer Graphics Forum*, 21, 08/2002 2002.

[23] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. PAMI*, 14(2), 1992.

[24] Pravin Bhat, Stephen Ingram, and Greg Turk. Geometric texture synthesis by example. In *Proc. SGP*, pages 41–44, 2004.

[25] Biondo L. Biondi. *3D Seismic Imaging*. Society of Exploration Geophysicists, 2006.

[26] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, 29(4):104, 2010.

[27] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proc. SGP*, pages 185–192, 2004.

[28] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Trans. PAMI*, 26(9):1124–1137, Sept 2004.

[29] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. PAMI*, 23(11):1222–1239, Nov 2001.

[30] Emilio Vital Brazil, Ives Macêdo, Mario Costa Sousa, Luiz Velho, and Luiz Henrique de Figueiredo. A few good samples: Shape & tone depiction for hermite RBF implicits. In *Proc. NPAR*, pages 7–15, 2010.

[31] Gabriel J Brostow, Carlos Hernández, George Vogiatzis, Bjoern Stenger, and Roberto Cipolla. Video Normals from Colored Lights. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2104–2114, October 2011.

[32] Benedict Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-D scans. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), August 2007.

[33] Benedict J Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, Tim Weyrich, and Tim Weyrich. A system for high-volume acquisition and matching of fresco fragments: reassembling Theran wall paintings. *ACM Trans. Graph.*, 27(3):84, 2008.

[34] LG Brown. A survey of image registration techniques. *ACM computing surveys (CSUR)*, 1992.

[35] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision*, 74(1):59–73, August 2007.

[36] M Chandraker, S Agarwal, and D Kriegman. ShadowCuts: Photometric Stereo with Shadows. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[37] Manmohan Krishna Chandraker, Fredrik Kahl, and David J Kriegman. Reflections on the generalized bas-relief ambiguity. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 788–795. IEEE, 2005.

[38] Xiaobai Chen, Tom Funkhouser, Dan B Goldman, , and Eli Shechtman. Non-parametric texture transfer using MeshMatch. *Adobe Technical Report 2012-2*, 2012.

[39] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3d mesh segmentation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 73. ACM, 2009.

[40] Xiaobai Chen, Abulhair Saparov, Bill Pang, and Thomas Funkhouser. Schelling points on 3D surface meshes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, August 2012.

[41] Y Chen and G Medioni. Object modeling by registration of multiple range images. In *Proc. Robotics and Automation*, pages 2724–2729, 1991.

[42] Ming Chuang, Linjie Luo, Benedict J Brown, Szymon Rusinkiewicz, and Michael Kazhdan. Estimating the Laplace-Beltrami operator by restricting 3D functions. In *Proceedings of the Symposium on Geometry Processing*, volume 28, pages 1475–1484, 2009.

[43] Massimiliano Corsini, Matteo Dellepiane, Federico Ponchio, and Roberto Scopigno. Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties. *Computer Graphics Forum*, 28(7):1755–1764, 2009.

[44] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM.

[45] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. In *Proc. ACM SIGGRAPH*, pages 848–855, 2003.

[46] Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. In *Proc. NPAR*, August 2007.

[47] Olga Diamanti, Connelly Barnes, Sylvain Paris, Eli Shechtman, and Olga Sorkine-Hornung. Synthesis of complex image appearance from limited exemplars. *ACM Trans. Graph.*, 34(2):22:1–22:14, 2015.

[48] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[49] Mark S Drew, Yacov Hel-Or, Tom Malzbender, and Nasim Hajari. Robust estimation of surface properties and interpolation of shadow/specularity components. *Image and Vision Computing*, 30(4-5):317–331, May 2012.

[50] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, pages 1033–1038, Corfu, Greece, 1999.

[51] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph.*, 23(3):673–678, August 2004.

[52] A. Fakhry. *The inscriptions of the amethyst quarries at Wadi el Hudi*. Service des antiquités de l'Égypte. Government Press, 1952.

[53] Raanan Fattal, Maneesh Agrawala, and Szymon Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), August 2007.

[54] P Favaro and T Papadhimitri. A closed-form solution to uncalibrated photometric stereo via diffuse maxima. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 821–828. IEEE, 2012.

[55] Juan Carlos Fernandez Diaz, William E. Carter, Ramesh L. Shrestha, and Craig L. Glennie. *Handbook of Satellite Applications*, chapter Lidar Remote Sensing, pages 757–808. Springer New York, New York, NY, 2013.

[56] M A Fischler and R C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[57] MA Fischler. Random sample consensus. *Communications of the ACM*, 1981.

[58] Hongbo Fu, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer. Upright orientation of man-made objects. *ACM Trans. Graph.*, 27(3):42:1–42:7, 2008.

[59] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2nd Ed.)*. Academic Press, 1990.

[60] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. Shape and Spatially-Varying BRDFs from Photometric Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, June 2010.

[61] Aleksey Golovinskiy, Wojciech Matusik, Hanspeter Pfister, Szymon Rusinkiewicz, and Thomas Funkhouser. A statistical model for synthesis of detailed facial geometry. *ACM Trans. Graph.*, 25(3):1025–1034, 2006.

[62] Todd Goodwin, Ian Vollick, and Aaron Hertzmann. Isophote distance: A shading approach to artistic stroke thickness. In *Proc. NPAR*, pages 53–62, 2007.

[63] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, pages 95–110. 2008.

[64] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, 2014.

[65] Roger Grosse, Micah K Johnson, Edward H Adelson, and William T Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2335–2342. IEEE, 2009.

[66] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[67] M. I. Hariz and L. Zrinzo. *Textbook of Stereotactic and Functional Neurosurgery*, chapter CT/MRI Technology: Basic Principles, pages 269–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[68] C Harris and M Stephens. A combined corner and edge detector. *Alvey vision conference*, 1988.

[69] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proc. ACM SIGGRAPH*, pages 229–238, 1995.

[70] Sigurdur Helgason. *Differential geometry, Lie groups, and symmetric spaces*. Academic Press, 1978.

[71] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 9(1):1:1–1:22, February 2013.

[72] C. Hernandez, G. Vogiatzis, and R. Cipolla. Multiview photometric stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(3):548–554, March 2008.

[73] A Hertzmann and S M Seitz. Example-based photometric stereo: Shape reconstruction with general, varying BRDFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1254–1264, August 2005.

[74] Robert .J. Hocken and Paulo H. Pereira. *Coordinate Measuring Machines and Systems, Second Edition*. Manufacturing Engineering and Materials Processing. CRC Press, 2016.

[75] Donald H. House and Mayank Singh. Line drawing as a dynamic process. In *Proc. Pacific Graphics*, pages 351–360, 2007.

[76] Qixing Huang, Leonidas J Guibas, and Niloy J Mitra. Near-regular structure discovery using linear programming. *ACM Trans. Graph.*, 33(3):23, 2014.

[77] V. Interrante, H. Fuchs, and S. Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *Proc. IEEE Visualization*, pages 52–59, 438, Oct 1995.

[78] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

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

[80] Tilke Judd, Frédo Durand, and Edward Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3), July 2007.

[81] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4):55, 2012.

[82] C Kambhamettu, D B Goldgof, and M He. On a study of invariant features in nonrigid transformations. In *Proc. Qualitative Vision*, pages 118–127, 1993.

[83] C. Kambhamettu, D.B. Goldgof, and M. He. Determination of motion parameters and estimation of point correspondences in small nonrigid deformations. In *Computer Vision and Pattern Recognition*, pages 943–946, Jun 1994.

[84] Shun'ichi Kaneko, Ichiro Murase, and Satoru Igarashi. Robust image registration by increment sign correlation. *Pattern Recognition*, 35(10):2223 – 2234, 2002.

[85] Shun'ichi Kaneko, Yutaka Satoh, and Satoru Igarashi. Using selective correlation coefficient for robust image registration. *Pattern Recognition*, 36(5):1165 – 1173, 2003.

[86] Henry Kang, Seungyong Lee, and Charles K. Chui. Coherent line drawing. In *Proc. NPAR*, pages 43–50, 2007.

[87] Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High dynamic range video. *ACM Trans. Graph.*, 22(3):319–325, July 2003.

[88] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. *Computer Graphics Forum*, 34(2):349–359, 2015.

[89] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70. Eurographics Association, 2006.

[90] Michael Kazhdan and Hugues Hoppe. Screened Poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):1–13, 2013.

[91] N.Y. Khan, B. McCane, and G. Wyvill. Sift and surf performance evaluation against various image deformations on benchmark dataset. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 501 –506, dec. 2011.

[92] Sung Ye Kim, Ross Maciejewski, Tobias Isenberg, William M. Andrews, Wei Chen, Mario Costa Sousa, and David S. Ebert. Stippling by example. In *Proc. NPAR*, pages 41–50, 2009.

[93] Vladimir G. Kim, Wilmot Li, Niloy Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring collections of 3D models using fuzzy correspondences. *ACM Trans. Graph.*, 31(4):54:1–54:11, 2012.

[94] Yeon-Ho Kim, Aleix M. Martnez, and Avi C. Kak. Robust motion estimation under varying illumination. *Image and Vision Computing*, 23(4):365 – 375, 2005.

[95] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Trans. PAMI*, 26(2):147–159, Feb 2004.

[96] M. Kolomenkin, I. Shimshoni, and A. Tal. On edge detection on surfaces. In *Proc. CVPR*, pages 2767–2774, June 2009.

[97] Michael Kolomenkin, Ilan Shimshoni, and Ayellet Tal. Demarcating curves for shape illustration. *ACM Trans. Graph.*, 27(5):157:1–157:9, December 2008.

[98] Michael Kolomenkin, Ilan Shimshoni, and Ayellet Tal. Multi-scale curve detection on surfaces. In *Proc. CVPR*, pages 225–232, 2013.

[99] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

[100] Carlos Lara, Leonardo Romero, and Félix Calderón. A robust iterative closest point algorithm with augmented features. In *MICAI 2008: Advances in Artificial Intelligence, LNCS 5317*, pages 605–614. Springer, 2008.

[101] P. Laskov and C. Kambhamettu. Curvature-based algorithms for nonrigid motion and correspondence estimation. *PAMI*, pages 1349–1354, 2003.

[102] Pavel Laskov and Chandra Kambhamettu. Comparison of 3D algorithms for non-rigid motion and correspondence estimation. In *Proc. British Machine Vision Conference*, 2001.

[103] Hao Li, Etienne Vouga, Anton Gudym, Linjie Luo, Jonathan T Barron, and Gleb Gusev. 3D self-portraits. *ACM Trans. Graph.*, 32(6):1–9, 2013.

[104] Min Li, Chandra Kambhamettu, and Maureen Stone. Nonrigid motion recovery for 3D surfaces. *Image and Vision Computing*, 25(3):250–261, 2007.

[105] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3D furniture models. *ACM Trans. Graph.*, 34(4):85:1–85:9, 2015.

[106] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[107] DG Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[108] Cewu Lu, Li Xu, and Jiaya Jia. Combining sketch and tone for pencil drawing production. In *Proc. NPAR*, pages 65–73, 2012.

[109] Jianye Lu, Athinodoros S. Georghiades, Andreas Glaser, Hongzhi Wu, Li-Yi Wei, Baining Guo, Julie Dorsey, and Holly Rushmeier. Context-aware textures. *ACM Trans. Graph.*, 26(1), 2007.

[110] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. pages 674–679, 1981.

[111] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. Elements of style: Learning perceptual shape style similarity. *ACM Trans. Graph.*, 34(4):84:1–84:14, 2015.

[112] Chongyang Ma, Haibin Huang, Alla Sheffer, Evangelos Kalogerakis, and Rui Wang. Analogy-driven 3D style transfer. *Comput. Graph. Forum*, 33(2):175–184, 2014.

[113] Wan-Chun Ma, Tim Hawkins, Pieter Peers, Charles-Felix Chabert, Malte Weiss, and Paul Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. pages 183–194, 2007.

[114] Wan-Chun Ma, Tim Hawkins, Pieter Peers, Charles-Felix Chabert, Malte Weiss, and Paul Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *EGSR'07: Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association, June 2007.

[115] Thomas Malzbender, Bennett Wilburn, Dan Gelb, and Bill Ambrisco. Surface enhancement using real-time photometric stereo and reflectance transformation. In Tomas Akenine-Mller and Wolfgang Heidrich, editors, *Rendering Techniques*, pages 245–250. Eurographics Association, 2006.

[116] Tom Malzbender, Dan Gelb, and Hans Wolters. Polynomial texture maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 519–528, New York, NY, USA, 2001. ACM.

[117] Tom Malzbender, Tom Malzbender, Dan Gelb, Dan Gelb, Hans Wolters, and Hans Wolters. Polynomial texture maps. In *Computer Graphics, SIGGRAPH 2001 Proceedings*, 2001.

[118] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[119] Eivind Lyche Melvær and Martin Reimers. Geodesic polar coordinates on polygonal meshes. *Computer Graphics Forum*, 31(8):2423–2435, 2012.

[120] Tom Mertens, Jan Kautz, Jiawen Chen, Philippe Bekaert, and Frédo Durand. Texture transfer using geometry correlation. In *Proc. EGSR*, pages 273–284, 2006.

[121] Microsoft, Inc. Kinect. www.xbox.com/en-US/kinect, 2010.

[122] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.

[123] Jan Möbius and Leif Kobbelt. OpenFlipper: an open source geometry processing and rendering framework. In *Proc. Curves and Surfaces*, pages 488–500, 2012.

[124] H. Mohammadzade and D. Hatzinakos. Iterative closest normal point for 3d face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):381–397, Feb 2013.

[125] Kenneth Moreland. Diverging color maps for scientific visualization. In *Proc. ISVC*, pages 92–103, 2009.

[126] Mark Mudge, Jean-Pierre Voutaz, Carla Schroer, and Marlin Lum. Reflection transformation imaging and virtual representations of coins from the Hospice of the Grand St. Bernard. In *Proc. VAST*, pages 29–39, 2005.

[127] Marius Muja and David G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV)*, pages 404–410, 2012.

[128] Yasuhiro Mukaigawa, Yasunori Ishii, and Takeshi Shakunaga. Analysis of photometric factors based on photometric linearization. *Journal of the Optical Society of America a-Optics Image Science and Vision*, 24(10):3326–3334, October 2007.

[129] Daniel Münch, Benoît Combès, and Sylvain Prima. A modified ICP algorithm for normal-guided surface registration. In *Proc. SPIE 7623, Medical Imaging 2010: Image Processing*, 2010.

[130] Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. Efficiently combining positions and normals for precise 3D geometry. *ACM Trans. Graph.*, 24(3):536–543, 2005.

[131] J A Nelder and R Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, Jan 1965.

[132] Occipital, Inc. Skanect. skanect.manctl.com, 2013.

[133] Mark Pauly, Niloy J. Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J. Guibas. Discovering structural regularity in 3D geometry. *ACM Trans. Graph.*, 27(3):43:1–43:11, 2008.

[134] Theo Pavlidis. An evaluation of the scale invariant feature transform (sift). *An Evaluation of SIFT*, 2008.

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

[136] Inc Pixologic. ZBrush. http://pixologic.com/zbrush, 2015.

[137] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, pages 1389–1401, 1957.

[138] Yvain Quéau, François Lauze, and Jean-Denis Durou. Solving uncalibrated photometric stereo using total variation. *Journal of Mathematical Imaging and Vision*, pages 1–21, 2014.

[139] M.A. Richards, J.A. Scheer, J. Scheer, and W.A. Holm. *Principles of modern radar*. Number v. 1 in Principles of Modern Radar. SciTech Publishing, Incorporated, 2010.

[140] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph.*, 29(6):157:1–157:8, 2010.

[141] Amir Rosenberger, Daniel Cohen-Or, and Dani Lischinski. Layered shape synthesis: Automatic generation of control maps for non-stationary textures. *ACM Trans. Graph.*, 28(5):107:1–107:9, 2009.

[142] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152, 2001.

[143] Joaquim Salvi, Jordi Pags, and Joan Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827 – 849, 2004. Agent Based Computer Vision.

[144] D Scharstein and R Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[145] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, 24(4):249–259, 2008.

[146] Boxin Shi, Y Matsushita, Yichen Wei, Chao Xu, and Ping Tan. Self-calibrating photometric stereo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1118–1125. IEEE, 2010.

[147] G. Silveira and E. Malis. Real-time visual tracking under arbitrary illumination changes. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –6, june 2007.

[148] Justin Solomon, Gabriel Peyré, Vladimir Kim, and Suvrit Sra. Entropic metric alignment for correspondence problems. *ACM Transactions on Graphics (TOG)*, 35(4), 2016.

[149] Minjung Son, Henry Kang, Yunjin Lee, and Seungyong Lee. Abstract line drawings from 2D images. In *Proc. Pacific Graphics*, pages 333–342, 2007.

[150] Mario Costa Sousa, Kevin Foster, Brian Wyvill, and Faramarz Samavati. Precise ink drawing of 3D models. *Computer Graphics Forum*, 22(3):369–379, 2003.

[151] Mario Costa Sousa, Przemyslaw Prusinkiewicz, Mario Costa, and Sousa Przemyslaw Prusinkiewicz. A few good lines: Suggestive drawing of 3D models. *Computer Graphics Forum (Proc. Eurographics*, 22:2003, 2003.

[152] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proc. SGP*, pages 1383–1392, 2009.

[153] Richard Szeliski. Image Alignment and Stitching: A Tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[154] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[155] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[156] Pierre Tallet. *La zone minière pharaonique du Sud-Sinaï*. Institut français d'archèologie orientale du Caire - IFAO, 2012.

[157] Ping Tan, Stephen Lin, and Long Quan. Resolution-enhanced photometric stereo. In *Proceedings of the 9th European conference on Computer Vision-Volume Part III*, pages 58–71. Springer-Verlag, 2006.

[158] Ping Tan, S P Mallick, Long Quan, D Kriegman, and T Zickler. Isotropy, Reciprocity and the Generalized Bas-Relief Ambiguity. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[159] Chunming Tang, Yan Dong, and Xiaohong Su. Automatic registration based on improved sift for medical microscopic sequence images. In *Proceedings of the 2008 Second International Symposium on Intelligent Information Technology Application - Volume 01*, IITA '08, pages 580–583, Washington, DC, USA, 2008. IEEE Computer Society.

[160] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *Shape Modeling Applications, 2004. Proceedings*, pages 145–156, June 2004.

[161] Corey Toler-Franklin, Adam Finkelstein, and Szymon Rusinkiewicz. Illustration of complex real-world objects using images with normals. In *Proc. NPAR*, pages 111–119, 2007.

[162] Corey Toler-Franklin, Adam Finkelstein, and Szymon Rusinkiewicz. Illustration of complex real-world objects using images with normals. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 111–119, New York, New York, USA, August 2007. ACM Request Permissions.

[163] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London A*, 356:1321–1340, 1998.

[164] Greg Turk. Texture synthesis on surfaces. In *Proc. ACM SIGGRAPH*, pages 347–354, 2001.

[165] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors - a survey. *Foundations and Trends in Computer Graphics and Vision*, 2008.

[166] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[167] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[168] Romain Vergne, Pascal Barla, Xavier Granier, and Christophe Schlick. Apparent relief: A shape descriptor for stylized shading. In *Proc. NPAR*, pages 23–29, 2008.

[169] P. Viola and W.M.III Wells. Alignment by maximization of mutual information. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 16–23, Jun 1995.

[170] A.D. Waite. *Sonar for Practising Engineers*. Wiley, 2002.

[171] Bingjian Wang, Dongliang Wu, Wenzheng Xu, Quan Lu, Fan Li, Shangqian Liu, Guowang Gao, and Rui Lai. A new image registration method for infrared images and visible images. In *Image and Signal Processing (CISP), 2010 3rd International Congress on*, volume 4, pages 1745 –1749, oct. 2010.

[172] Zhongjie Wang, M Grochulla, T Thormahlen, and H P Seidel. 3D face template registration using normal maps. In *Proc. International Conference on 3D Vision*, 2013.

[173] Greg Ward. Fast, robust image registration for compositing high dynamic range photographs from handheld exposures. *JOURNAL OF GRAPHICS TOOLS*, 8:17–30, 2003.

[174] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics '09 State of the Art Reports (STARs)*. Eurographics, 2009.

[175] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. ACM SIGGRAPH*, pages 479–488, 2000.

[176] Li-Yi Wei and Marc Levoy. Texture synthesis over arbitrary manifold surfaces. In *Proc. ACM SIGGRAPH*, pages 355–360, 2001.

[177] Andreas Wenger, Andrew Gardner, Chris Tchou, Jonas Unger, Tim Hawkins, and Paul Debevec. Performance relighting and reflectance transformation with time-multiplexed illumination. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 24(3):756–764, July 2005.

[178] Simon A. J. Winder and Matthew Brown. Learning local image descriptors. In *In CVPR*, pages 1–8, 2007.

[179] Simon A. J. Winder, Gang Hua, and Matthew Brown. Picking the best daisy. In *CVPR*, pages 178–185, 2009.

[180] Holger Winnemöller. Xdog: Advanced image stylization with extended difference-of-gaussians. In *Proc. NPAR*, pages 147–156, 2011.

[181] Robert J Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. In *22nd Annual Technical Symposium*, pages 136–143. International Society for Optics and Photonics, 1979.

[182] Robert J Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[183] Chenglei Wu, Yebin Liu, Qionghai Dai, and Bennett Wilburn. Fusing multiview and photometric stereo for 3d reconstruction under uncalibrated illumination. *Visualization and Computer Graphics, IEEE Transactions on*, 17(8):1082–1095, 2011.

[184] Lun Wu, Arvind Ganesh, Boxin Shi, Yasuyuki Matsushita, Yongtian Wang, and Yi Ma. Robust photometric stereo via low-rank matrix completion and recovery. In *Computer Vision–ACCV 2010*, pages 703–717. Springer, 2011.

[185] Ying Xiong, Ayan Chakrabarti, Ronen Basri, Steven J Gortler, David W Jacobs, and Todd Zickler. From Shading to Local Shape. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2014.

[186] Jie Xu and Craig S. Kaplan. Artistic thresholding. In *Proc. NPAR*, pages 39–47, 2008.

[187] Kai Xu, Daniel Cohen-Or, Tao Ju, Ligang Liu, Hao Zhang, Shizhe Zhou, and Yueshan Xiong. Feature-aligned shape texturing. *ACM Trans. Graph.*, 28(5):108:1–108:7, 2009.

[188] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: Set evolution for inspiring 3D shape galleries. *ACM Trans. Graph.*, 31(4):57:1–57:10, 2012.

[189] Lap-Fai Yu, Sai-Kit Yeung, Yu-Wing Tai, and S. Lin. Shading-based shape refinement of rgb-d images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1415–1422, June 2013.

[190] Alan Yuille and Daniel Snow. Shape and albedo from multiple images using integrability. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 158–164. IEEE, 1997.

[191] Rony Zatzarinni, Ayellet Tal, and Ariel Shamir. Relief analysis and extraction. In *ACM SIGGRAPH Asia*, pages 136:1–136:9, 2009.

[192] Feng Zhao, Qingming Huang, and Wen Gao. Image matching by normalized cross-correlation. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 2, page II, may 2006.

[193] Howard Zhou, Jie Sun, Greg Turk, and James M. Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, 2007.

[194] Kun Zhou, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. Mesh quilting for geometric texture synthesis. *ACM Trans. Graph.*, 25(3):690–697, 2006.

[195] Zhenglong Zhou and Ping Tan. Ring-light photometric stereo. In *ECCV'10: Proceedings of the 11th European conference on Computer vision: Part II*. Springer-Verlag, September 2010.

[196] B Zitova. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, October 2003.