Shape from Video: Dense Shape, Texture, Motion and Lighting from Monocular Image Streams

by

Azeem Lakdawalla

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Shape from Video: Dense Shape, Texture, Motion and Lighting from Monocular Image Streams

Azeem Lakdawalla

Master of Science

Graduate Department of Computer Science

University of Toronto

2005

We present a probabilistic framework for robust recovery of dense 3D shape, motion, texture and lighting from monocular image streams. We assume that the object is rigid, smooth, Lambertian, illuminated by one distant light source and subject to transformations that are smoothly time-varying. The problem is formulated as a large optimization where we learn all model and pdf (probability distribution function) parameters simultaneously, using a quasi-Newtonian optimization technique.

Contents

1 Introduction **1**

2 Background **3**

2.1 Shape and motion recovery assuming brightness constancy 3

2.1.1 Feature-Based methods 4

2.1.2 Direct Methods 7

2.2 Shape recovery from shading information 10

2.2.1 Shape-from-shading 12

2.2.2 Photometric Stereo 15

2.3 Hybrid methods 16

3 A Generative Model for Shape from Video **19**

3.1 What are we generating? 19

3.2 Surface and Normal Parameterization 20

3.2.1 Approximating Derivatives 20

3.3 Affine Transformations and Projection 21

3.4 Intensity of projected point $\mathbf{p}_{i,t}$ 22

3.5 Probabilistic formulation for image generation 22

3.5.1 Gaussian distributions 23

3.5.2 Robust imaging model 24

3.5.3 Priors 27

3.6 Summary 29

4 The Problem Statement for Shape from Video **31**

4.1 Maximum A Posteriori (MAP) Estimation 31

5 Optimization **33**

5.1 Overview 33

5.1.1 I and ΔI 34

5.1.2 Constraints 35

5.2 Optimization schedule 36

5.2.1 Initialization 36

5.3 Pre-conditioning 39

5.4 Other attempted optimization methods 39

5.4.1 Stochastic gradient descent method 40

5.4.2 Solution by coordinate ascent 42

6 Results **44**

6.1 Computer generated sequence 46

6.2 "Lady" figurine 51

6.2.1 Comparison with other work 54

6.3 Hatake Kakashi figurine 57

6.4 Uzumaki Naruto figurine 61

6.5 Occluded Uzumaki Naruto figurine 66

6.5.1 With mixture components 67

6.5.2 Without mixture components 71

7 Discussion and Future Directions **75**

7.1 Improvements 75

7.2 Extensions 76

7.3 Validation 76

Bibliography **77**

Chapter 1

Introduction

The visual systems in humans and animals have the uncanny ability to extract detailed scene information based on "images" captured by the eyes. In numerous cases, the way light is reflected off a moving object that has been imaged in 2D provides sufficient queues to enable our brains to deduce shape and motion in 3D. This is done effortlessly by our visual system yet the problem is far from trivial for machines. An image, after all, is simply an matrix of intensity values. How can such data be examined in order to extract relevant shape and motion information? This question has motivated computer vision researchers to develop techniques and algorithms to recover an object's shape and motion from either stereo or monocular image streams. It has become one of the "classic" problems in computer vision (section 2). As well as being an interesting problem in its own right, there are numerous applications for shape and motion reconstruction from video: 3D special effects animation, robotic navigation and interaction, medical imaging, architecture and urban design, etc...

This work presents a method to reconstruct dense 3D rigid geometry, motion, texture and lighting from monocular image streams captured by a freely-moving camera. The imaged objects are assumed to be projected using scaled orthographic projection, illuminated by one distant, static, light source. We require initial, sparse, point tracks

of the object in order to bootstrap the system. In our implementation, this is done via user interaction. To improve convergence and the quality of results, the user is strongly encouraged to manually segment the objects in question from the background by creating a boolean mask. We assume no *a-priori* shape model (section 3.2), only that the object is smooth and moves smoothly across frames (section 3.5.3).

Our method combines feature-based and photometric techniques to obtain dense shape recoveries. While work has been done in this area with impressive results [43, 22], they are not robust to outliers. Since captured video sequences come from a freely-moving camera, robustness to outliers is important since the images may contain occlusions, shadows or reflections. Our system is robust to such outliers (sections 3.5.2 and 6.5).

We formulate the problem in a probabilistic framework (section 3.6), and use non-linear quasi-Newtonian optimization (section 5.1) to obtain all model and distribution-function parameters. This procedure is very computationally demanding, and requires several days of continuous optimization in order to produce visually-acceptable results. We show results of our technique on various textured figurines (section 6), and provide a comparison with other work in the same domain (section 6.2).

Chapter 2

Background

3D object and motion reconstruction from images is one of the most active areas in computer vision. Among the wide body of research in this area, we present here the major techniques that are directly related to this thesis.

We identify three major themes in this domain. The first concerns the recovery of shape and motion from image streams by exploiting the assumption of *brightness constancy* across frames. The second theme concerns shape recovery based on *changes* in intensity of the imaged object's surface points. Finally, the third theme concerns "hybrids" that combines the first two themes into a framework that takes advantage of both techniques.

2.1 Shape and motion recovery assuming brightness constancy

The assumption of brightness constancy states that an imaged object's surface points will not significantly change intensity across frames. The feature-based and direct methods we describe here both use this assumption to "track" points across frames in order to determine the shape and motion of the underlying imaged object.

2.1.1 Feature-Based methods

Feature-Based methods can be thought of as a two step process. First, feature tracks are determined across the image stream. Then, these tracks are fed into a structure-from-motion (SFM) algorithm that examines the tracks and computes the shape and motion of the object. We begin by discussing features, then methods used to obtain point tracks, and finally move on to SFM algorithms.

2.1.1.1 Features

Features are pixel "windows" that have a certain degree of "uniqueness" in that we want to be able to identify the same feature in **different** images. This is done by examining pixel windows in the image for their content, and comparing them with other windows in other images. Lowe [23], for instance, has developed a Scale Invariant Feature Transform (SIFT) to extract salient features from an image by examining difference of Gaussians in scale-space. SIFT points are descriptors that can be used to locate similar features in other images. SIFT has been used successfully in object recognition, using sparse images taken from different viewpoints [23], by comparing SIFT features present in each image with a database of "trained" images.

2.1.1.2 Feature Tracking

Although SIFT can find features in each image separately, determining their correspondences across frames is very tricky. Often similar keypoints will not map directly to their temporal counterparts. Since the images are extracted from a video stream, a temporal model is more appropriate. Lucas and Kanade [24] have shown how to register pixel neighborhoods using an alignment algorithm (2.7) based on brightness constancy between adjacent frames.

Consider the image point (x_i, y_i) at frame t. From one frame to the next, this point

will move some distance (u_{t_i}, v_{t_i}), or:

$$I_{t+1}(x_i, y_i) = I_t(x_i - u_{t_i}, y_i - v_{t_i}) \tag{2.1}$$

We assume that u_{t_i} and v_{t_i} are very small, since we are considering movement from one frame to the next. We can therefore linearize as follows:

$$\begin{aligned} I_{t+1}(x_i, y_i) &= I_t(x_i - u_{t_i}, y_i - v_{t_i}) && (2.2) \\ I_{t+1}(x_i, y_i) &\approx I_t(x_i, y_i) - \begin{bmatrix} \frac{dI_t}{dx_i} & \frac{dI_t}{dy_i} \end{bmatrix} \begin{bmatrix} u_{t_i} \\ v_{t_i} \end{bmatrix} && (2.3) \end{aligned}$$

Rearranging the terms, we obtain a line constraint in uv space:

$$\frac{dI_t}{dx_i} u_{t_i} + \frac{dI_t}{dy_i} v_{t_i} + \frac{dI_t(x_i, y_i)}{dt} = 0 \tag{2.4}$$

where $\frac{dI_t(x_i,y_i)}{dt} = I_{t+1}(x_i, y_i) - I_t(x_i, y_i)$. This is an underconstrained problem since we have one equation with two unknowns. We can therefore consider a *window* of pixels, w, that moves under (u_{t_i}, v_{t_i}). Determining the displacement (u_{t_i}, v_{t_i}) is equivalent to minimizing:

$$E(u_{t_i}, v_{t_i}) = \sum_w \left(\frac{dI_t}{dx_{i_w}} u_{t_i} + \frac{dI_t}{dy_{i_w}} v_{t_i} + \frac{dI_t(x_i, y_i)}{dt} \right)^2 \tag{2.5}$$

Taking the derivative of (2.5) with respect to u_{t_i} and v_{t_i} and setting the two equations to zero gives the following system of equations:

$$\begin{aligned} \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right] &= \begin{bmatrix} \sum_w \left(\frac{\partial I}{\partial x}\right)^2 & \sum_w \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \\ \sum_w \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} & \sum_w \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} && (2.6) \\ [u_{t_i} v_{t_i}] \cdot \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right] &= \left[\sum_w \frac{\partial I}{\partial x} \frac{\partial I}{\partial t} \ \sum_w \frac{\partial I}{\partial y} \frac{\partial I}{\partial t} \right] && (2.7) \end{aligned}$$

Recovering the displacement $[u_{t_i} v_{t_i}]$ requires the matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right]$ to be non-singular. This occurs when pixel windows w contain high spatial frequency content, or when both eigenvalues are similar in magnitude [30]. The entire sequence can be tracked by propagating registered pixel neighborhoods from one frame to the next. It provides an easy way to obtain point tracks and has been used extensively for over 20 years [1].

The Lucas-Kanade tracking method does suffer from sensibility to noise, however. Since the algorithm attempts to find displacements by matching pixel windows, noise can cause the tracks to get "lost" and latch on to other areas of the image that do not correspond to the original feature. The method can also suffers from the aperture problem, or situations where image derivatives are much stronger in one direction than the other (one eigenvalue is much larger than the other). This will cause the tracked windows to "slide" along such edges, since the many windows along it will match the original feature.

2.1.1.3 Structure-from-Motion

Tomasi et al.'s factorization method [34] can be used to obtain rigid shape and motion from orthographic 2D point tracks. They show that stacking the mean-subtracted tracks into a registered measurement matrix $\mathbf{W}$ yields a matrix with at most rank 3. This can be exploited by factoring the matrix using singular-value decomposition, but keeping only the largest 3 singular-values and their corresponding eigenvectors.

$$\mathbf{W}_{2F\times P} = \left[\frac{\mathbf{X}_{F\times P}}{\mathbf{Y}_{F\times P}}\right] \tag{2.8}$$

$$= \mathbf{U}_{2F\times P}\,\mathbf{D}_{P\times P}\,\mathbf{V}^T_{P\times P} \tag{2.9}$$

$$\tilde{\mathbf{W}}_{2F\times P} = \tilde{\mathbf{U}}_{2F\times 3}\,\tilde{\mathbf{D}}_{3\times 3}\,\tilde{\mathbf{V}}^T_{3\times P} \tag{2.10}$$

This new matrix can be factored into two matrices representing rotation $\mathbf{R}$ and shape $\mathbf{S}$.

$$\tilde{\mathbf{W}}_{2F\times P} = \left(\tilde{\mathbf{U}}\tilde{\mathbf{D}}^{\frac{1}{2}}\right)\left(\tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{V}}^T\right) \tag{2.11}$$

$$= \tilde{\mathbf{R}}\tilde{\mathbf{S}} \tag{2.12}$$

$$= (\mathbf{R}\mathbf{Q})\left(\mathbf{Q}^{-1}\mathbf{S}\right) \tag{2.13}$$

where $\mathbf{Q}$ are metric constraints.

The recovered shapes are generally very sparse, since most imaged objects do not contain an abundance of trackable features. Furthermore, if the tracks are unreliable or

noisy, this process will result in poor reconstructions. In general, the factorization method is used to provide a rough estimate of an object's shape and motion for initialization to *other* reconstruction algorithms, as they provide good "initial guesses" [43].

Szeliski and Kang [33] pose the SFM problem as a least squares optimization. This has many benefits, notably the ability to generalize the approach to perspective projection and support partial or uncertain 2D tracking. This provides for a more robust method for shape and motion recovery, since factorization can break down in the presence of noise and cannot implicitly handle occluded point tracks.

Pollefeys et al. [27] describe a complete system for recovering dense structure and camera motion from uncalibrated video. Sparse features are first used to determine the epipolar geometry and camera parameters using the robust RANSAC algorithm [10] to determine inliers. The camera projection matrices are then automatically calibrated, and all images are rectified by warping them so that their epipolar lines coincide. Stereo techniques are then used to compute pixel correspondences to obtain dense reconstruction. The system is very impressive, not only for the resulting reconstructions (figure 2.1) but also because there are no constraining assumptions on the camera (such as the requirement for orthographic projection) and occlusions are supported. Furthermore the system is completely automatic.

2.1.2 Direct Methods

Feature-based methods rely on the tracking of points to derive shape and motion. They try to minimize an error metric based on tracked points, whereas direct methods [17] minimize an error metric based *directly* on raw image data.

Most of the shape recovery algorithms we describe in this section still use features to recover shapes, however the recovery of features is implicitly embedded into the whole mechanism. The two-step process of feature-based methods has been fused into one *direct* process.

Figure 2.1: Dense surface reconstruction. Original frame (left), textured reconstruction (middle) and close-up textured reconstruction (right). Taken from Pollefeys et al. [27]

Irani [16] exploited subspace constraints to develop a multi-point, multi-frame optical flow algorithm. Instead of tracking points **individually from frame to frame** (using 2.7), the whole sequence, with **all** points, is considered at once. We can write a multi-point, multi-frame version of (2.7) by assembling all displacements $[u_{p,t}v_{p,t}]$ into a large $T \times 2P$ matrix which gives:

$$[\mathbf{U}|\mathbf{V}]_{T\times 2P}\begin{bmatrix} A|B \\ \overline{C|D} \end{bmatrix}_{2P\times 2P} = [\mathbf{G}|\mathbf{H}]_{T\times 2P} \tag{2.14}$$

Irani shows that the rank of $[\mathbf{U}|\mathbf{V}]$ will be equal or less than 9. We can therefore exploit these constraints on the correspondences in order to reduce noise and resolve ambiguity in regions exhibiting the aperture problem. The observation here is if $[\mathbf{U}|\mathbf{V}]$ has rank $\leq r$, then $[\mathbf{G}|\mathbf{H}]$ has rank $\leq r$. We can therefore project $[\mathbf{G}|\mathbf{H}]$ to the lower rank $[\bar{\mathbf{G}}|\bar{\mathbf{H}}]$ and use it to calculate $[\bar{\mathbf{U}}_0|\bar{\mathbf{V}}_0]$. Iterating through these steps will yield refined versions of $[\bar{\mathbf{U}}|\bar{\mathbf{V}}]$. The key insight is that the constraints are being applied to raw image data and not the estimated, noisy point tracks. The method is therefore *directly* using the original, raw data.

Torresani et al. [36] and Brand [7] use such rank constraints to correct less reliable data (ie. textureless regions, occlusion, noise) and recover non-rigid geometry and motion by using a factorization method similar to (2.12). Since these methods directly couple

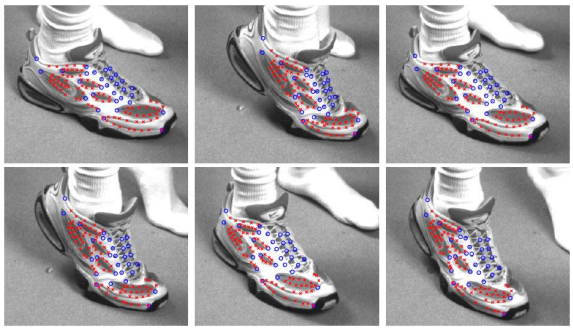

Figure 2.2: Improved tracking using rank constraints. In red are features that have been recovered using rank constraints that would otherwise suffer from the aperture problem. Taken from [36]

Figure 2.3: Robust tracking, shape and motion recovery using a probabilistic model. Points in green are valid features, and points in red are outliers. Taken from Torresani and Hertzmann [35].

tracking with shape recovery, they benefit from both improved tracking (figure 2.2) and improved shape recovery.

Torresani and Hertzmann [35] cast the same problem as an maximum likelihood estimation problem. This is done by specifying a generative model for non-rigid shape and motion based on features, where the features in turn are Lucas-Kanade pixel windows. This is all specified together in one model. Their system then automatically solves for **all** unknown parameters of the model and the probability distribution function. It is also robust, as it handles outliers (figure 2.3). This is the attractive aspect of probabilistic methods. They can be constructed using a *generative* framework (see chapter 3), making it relatively simple to include complicated notions such as occlusions.

Blanz et al. [38] use *a-priori* models to fit morphable face models to images and video.

Figure 2.4: Tracked and recovered shape (bottom) of an input sequence (top). Taken from Blanz et al. [38]

This method is not based on features at all, and instead directly minimizes the distance between image data and a projected morphable head model by summing over all pixels. They can use their model to track and solve for geometry at the same time (figure 2.4). The results are impressive, however the system requires a substantial amount of training data (scanned and aligned head models) to deliver convincing reconstructions. This limits the system since it cannot reconstruct arbitrary geometry. It does, however, demonstrate the effectiveness of constraining the space of possible solutions to yield convincing shape recoveries.

2.2 Shape recovery from shading information

In this section we describe two methods, shape-from-shading and photometric stereo, that attempt to determine a height field's normals from one or more images by using the object's reflectance function and information about lighting.

Shape information can be extracted from one or more images by observing the shading variations across the imaged object's surface. The shading is assumed to be described by Lambert's Law (2.15) [11], which relates the intensity at a point with the angle between the surface's unit normal, $\bar{\mathbf{n}}$, and the light direction, $\bar{\mathbf{l}}$ (figure 2.5).

$$I = \max(\bar{\mathbf{n}} \cdot \bar{\mathbf{l}}, 0) \tag{2.15}$$

The *reflectance function* of an imaged surface point for a Lambertian height field

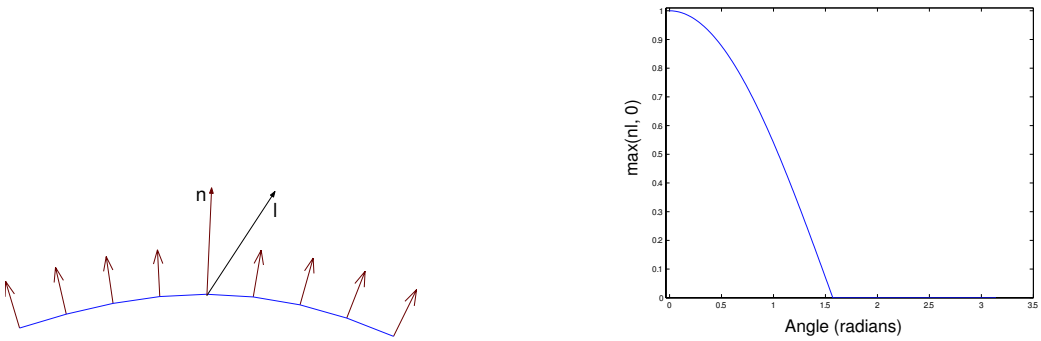

Figure 2.5: Lambertian lighting

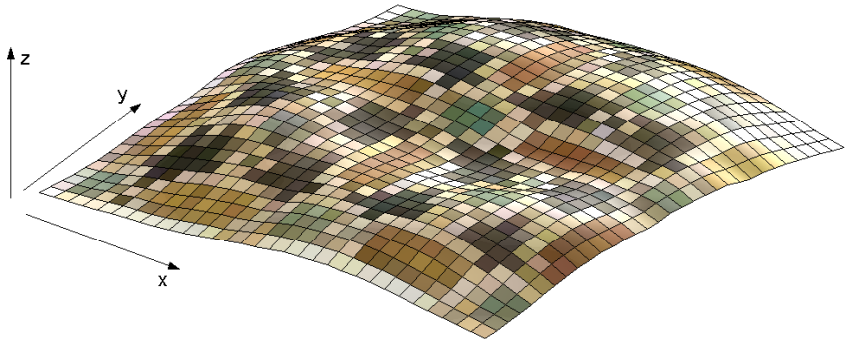

Figure 2.6: A height field $z = f(x, y)$

$z = f(x, y)$ is:

$$R(x, y) = \alpha \, \bar{\mathbf{n}}(x, y) \cdot \bar{\mathbf{l}} \tag{2.16}$$

where α is albedo, $\bar{\mathbf{n}}$ and $\bar{\mathbf{l}}$ are normalized vectors of the surface normal and directional light respectively.

Consider surface points $\mathbf{s}_i = [x_i, y_i, z\,(x_i, y_i)]$ on this height field, where i is an index over locations on the height field's grid. Two orthogonal surface tangent vectors of z at $\mathbf{s}_i$ are:

$$p_i = \frac{\partial z(x_i, y_i)}{\partial x_i} \tag{2.17}$$

$$q_i = \frac{\partial z(x_i, y_i)}{\partial y_i} \tag{2.18}$$

$$\frac{\partial \mathbf{s}_i}{\partial x_i} = [1, 0, p_i] \tag{2.19}$$

$$\frac{\partial \mathbf{s}_i}{\partial y_i} = [0, 1, q_i] \tag{2.20}$$

A normal vector at $\mathbf{s}_i$ can be expressed as the cross product of both tangent vectors:

$$\mathbf{n}(x_i, y_i) = \frac{\partial \mathbf{s}_i}{\partial x_i} \times \frac{\partial \mathbf{s}_i}{\partial y_i} = [-p_i, -q_i, 1] \tag{2.21}$$

The unit normal is:

$$\bar{\mathbf{n}}(x_i, y_i) = \frac{\mathbf{n}(x_i, y_i)}{\|\mathbf{n}(x_i, y_i)\|} = \frac{[-p_i, -q_i, 1]}{\sqrt{p_i^2 + q_i^2 + 1}} \tag{2.22}$$

From (2.21), each normal can be expressed in terms of the surface's tangent vectors. If we have determined a surface's normals, the heights can be calculated expressing the normal's vector components p_i and q_i with finite difference approximations (section 3.2.1), yielding two linear constraints:

$$p_i = z(x_i + 1, y_i) - z(x_i, y_i) \tag{2.23}$$

$$q_i = z(x_i, y_i + 1) - z(x_i, y_i) \tag{2.24}$$

By assembling all constraints for all normals we can construct a large linear system and solve for height values $z(x_i, y_i)$, up to a translational ambiguity in z [2].

2.2.1 Shape-from-shading

Given **one** image, $I(x, y)$, of an object with constant and known albedo, α, illuminated by a known directional light source, shape-from-shading (SFS) [8] uses the reflectance function to determine an imaged object's normals. This is the same as minimizing (2.25).

$$E_{SFS} = \sum_{x,y} \left(I(x, y) - R(x, y)\right)^2 \tag{2.25}$$

Determining the normals, however, is an underconstrained problem since each pixel corresponds to one equation with two unknowns (p and q). Constraints, or penalty terms, are needed to find a unique solution.

The most common constraint used in SFS are the integrability constraint [12] and the smoothness constraint [15, 14]. The integrability constraint (2.26) is used to enforce

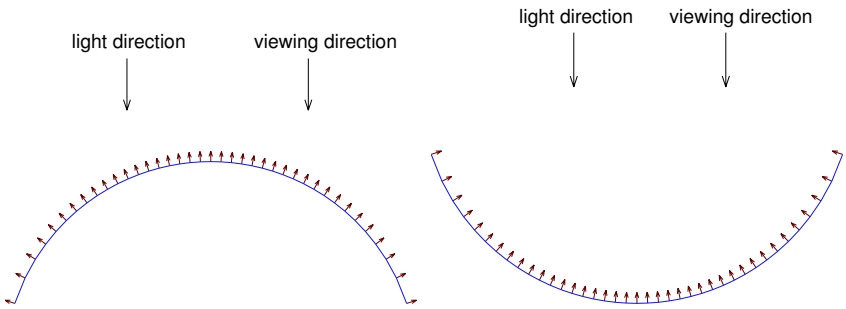

Figure 2.7: Concave-Convex ambiguity. Both imaged surfaces will look the same from the viewing direction.

a valid surface, since not all configurations of normals yield possible surfaces.

$$E_I = \sum_{x,y} \left(\frac{\partial p}{\partial y} - \frac{\partial q}{\partial x} \right)^2 \tag{2.26}$$

This constraint ensures that if we integrate along any path on the surface from (x_{start}, y_{start}) to (x_{end}, x_{end}), we always obtain the same value.

If, however, we express the reflectance function directly with finite differences for the components of $\bar{\mathbf{n}}$ [37] (instead of solving for normals and then integrating), we can eliminate the need for the integrability constraint altogether.

The smoothness constraint ensures that normals across the surface should change gradually (2.27). Most SFS algorithms make use of this "regularization" constraint.

$$E_S = \sum_{x,y} \left(\frac{\partial p}{\partial x} \right)^2 + \left(\frac{\partial q}{\partial y} \right)^2 \tag{2.27}$$

Applying these constraints and obtaining a smooth integrable normal field does not ensure that we obtain the **correct** 3D surface, however. There is still an ambiguity [4] with respect to the concave-convex nature of the shaded region (figure 2.7). Since the intensity is based on the dot product, or angle, between the normal and the light vector, there are two sets of normals that will give the same image appearance.

In general, SFS algorithms do not yield convincing results [44]. Figure 2.8 shows results using the algorithm described in Lee and Kuo [21]. This experiment was conducted

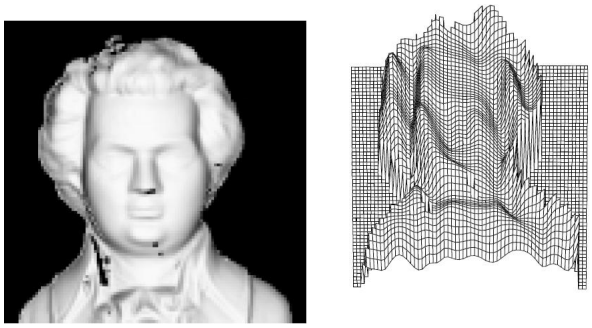

Figure 2.8: Example of Lee and Kuo's [21] SFS algorithm. Taken from Zhang et al. [44]

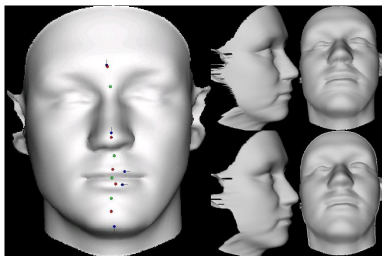

Figure 2.9: User-guided SFS. Taken from Zheng et al. [42]

by Zhang et al. [44] in a comparison of methods. They determined that this algorithm was among the best.

The problem with SFS is due to its inherent ill-posedness, and attempts to add constraints are usually very "hit and miss" since the parameters of these terms come down to guesswork. Also, the omnipresent concave-convex ambiguity usually results in surfaces that are incorrect. Finally, SFS is a difficult technique to use with "real world" scenes as it is very difficult to deduce the light source's direction.

Nonetheless, Zheng et al. [42] have shown that SFS can work if the user guides the process and resolves ambiguities. By allowing the user to place constraints on the image, SFS can deliver very detailed and impressive results (figure 2.9).

2.2.2 Photometric Stereo

Photometric stereo [41] can be thought of as an extension to shape-from-shading. By holding the viewing direction constant and varying the direction of a known light source between successive images, this technique recovers a surface's normals via the reflectance function. If sufficient images are provided, it can recover the albedo of the imaged object as well. Usually photometric stereo is done with the intention of recovering both albedo and normals.

From (2.16), for several images N at a particular point (x, y), we have:

$$\mathbf{i}_{1\times N} = \alpha \bar{\mathbf{n}}^T_{1\times 3} \mathbf{L}_{3\times N} \tag{2.28}$$

From (2.28), if we want to determine the normal vector only, we need a minimum of two images. If we provide a minimum of three images, we can also recover the albedo of the object at that location.

Belhumeur et al. [4] have shown that with three images of an object imaged under **unknown** lighting conditions, the surface can be reconstructed up to a Generalized Bas-Relief (GBR) transformation. The GBR is a linear transformation, $\bar{f}(x, y) = \lambda f(x, y) + \mu x + \nu y$, which essentially amounts to scaling the original function and then adding a plane to it. This observation stems from the fact that there are many combinations of light source directions and normals that will yield the same shaded imaged object (figure 2.10).

It has been observed that the approximately 98% of the reflected light field from a Lambertian object can be represented by the first two modes of its spherical harmonic representation [3, 29]. This means that the set of images produced by a Lambertian object, *regardless of how complex the illumination is*, can be approximated by a 9 dimensional linear subspace. Basri and Jacobs [2] have developed a photometric stereo technique for unknown **general** lighting conditions based on these findings. They use a factorization technique similar to Tomasi and Kanade's [34] to obtain normals and

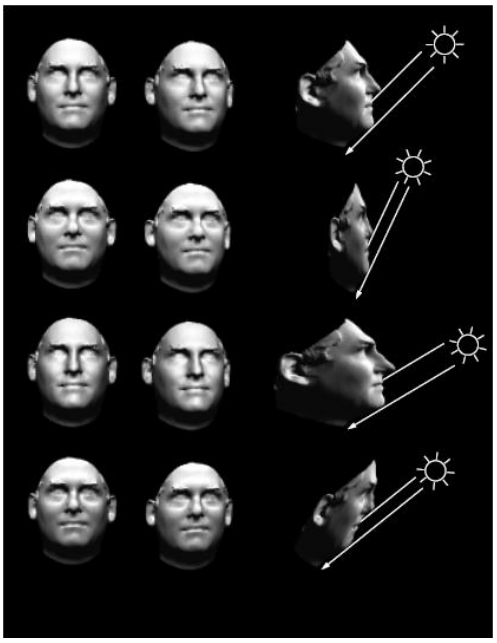

Figure 2.10: The Bas-Relief Ambiguity. Taken from Belhumeur et al. [4]

albedo. This technique is very advantageous as it permits lighting conditions found in "real world" images (figure 2.11).

2.3 Hybrid methods

There has been research that does not strictly fall into each of the above categories. Work has also been done to combine tracking and structure-from-motion with photometric

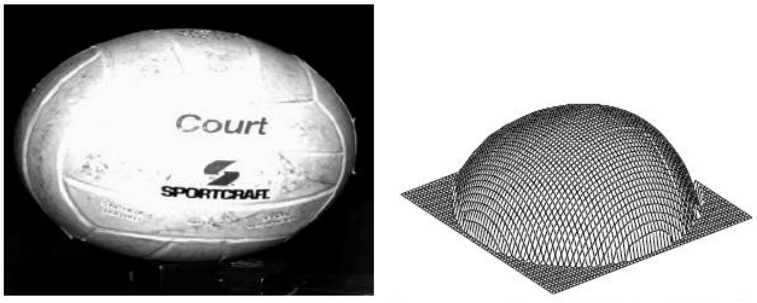

Figure 2.11: Photometric stereo with general, unknown lighting. Taken from Basri and Jacobs [4]

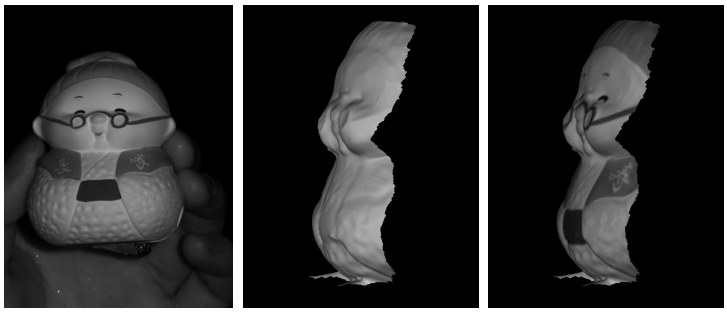

Figure 2.12: The reconstructed "Lady" figurine from Zhang et al. [43]

stereo, thereby creating "hybrid" methods.

Tracking under varying illumination is described by Jin et al. [18]. They successfully deal with intensity variation by introducing a scaling variable $\lambda(p,t) = \frac{I_t(x_{p_t})}{I_0(x_{p_0})}$ so that $I_t(x_{p_t}) = \lambda(p,t)I_0(x_{p,0})$. The $\lambda(p,t)$ are solved for together with the flow. Negahdaripour [25] extends 2.1 even further by not only adding a scaling variable but also an offset variable:

$$I_{t+1}(x_i + u_{t_i}, y_i + v_{t_i}) = M(x_i, y_i, t_i)I_t(x_i, y_i) + C(x_i, y_i, t_i) \tag{2.29}$$

where $M(x_i, y_i, t_i)$ and $C(x_i, y_i, t_i)$ are the multiplier (scaling) and offset variables respectively. The two variables are solved for with the flow, giving a total of 4 unknowns per constraint.

Jin et al's [18] method was used by Zhang et al. [43] to successfully track rank-constrained features on a moving Lambertian object. From these features, structure-from-motion is used to recover an initial sparse shape. Then the shape is refined (using geometric constraints) and normals and lighting are determined (using photometric constraints). This combination of techniques is very effective, since the textureless regions that cannot be tracked can be recovered by the photometric technique (figure 2.12). Unfortunately, their method is not robust to outliers and occlusions.

Lim et al. [22] have developed an even simpler method based on Zhang et al. [43]. They create an intensity matrix based on the recovered SFM parameters. This matrix is factorized into normals and lighting (2.28). A surface is then integrated and the whole

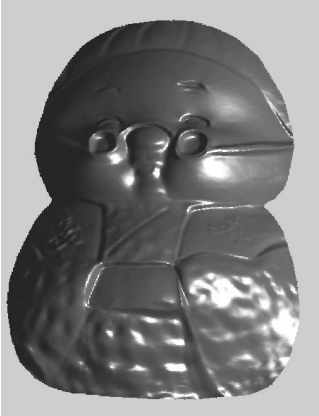

Figure 2.13: The reconstructed "Lady" figurine from Lim et al . [22]

process is reiterated until convergence. They operate directly on high resolution images instead of implementing the typical coarse-to-fine procedures used for such problems. Figure 2.13 shows their results on the same "Lady" sequence.

Chapter 3

A Generative Model for Shape from Video

This chapter introduces a generative model, a probabilistic model which explicitly states how to generate image sequences.

We begin by stating what exactly needs to be generated, and what assumptions we are making to attain this goal. We then move on to describing the geometrical, reflective and temporal properties involved in creating the image sequence.

3.1 What are we generating?

Our model will create an image sequence of a smooth, textured, rigid, Lambertian object undergoing rotation and translation. The object is imaged using scaled-orthographic projection, illuminated by one static directional light source. There is temporal coherence between adjacent frames in the sequence, meaning that the object does not undergo drastically different transformations from one frame to the next.

Figure 3.1 shows example frames of the types of images we would like our generative model to create.

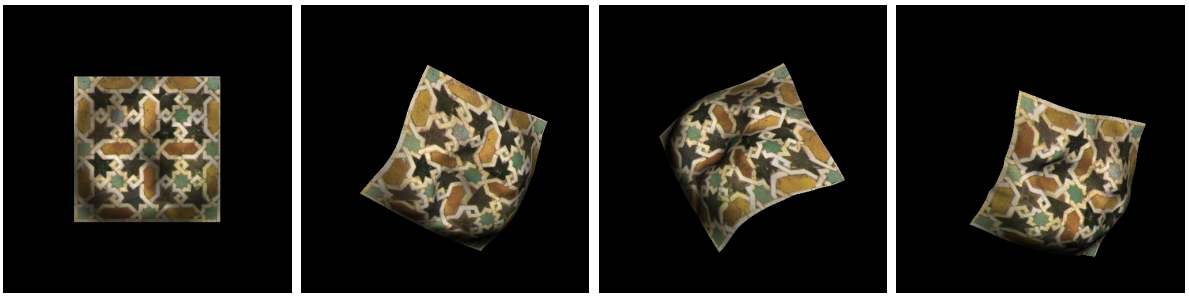

Figure 3.1: 4 frames from a generated image sequence

3.2 Surface and Normal Parameterization

The geometric primitive used in our system is a height field $z = f(x, y)$. The normals, as described in section 2.2, are:

$$\bar{\mathbf{n}}_i = \bar{\mathbf{n}}(x_i, y_i) = \frac{\mathbf{n}(x_i, y_i)}{\|\mathbf{n}(x_i, y_i)\|} = \frac{\left[-\frac{\partial z(x_i,y_i)}{\partial x_i}, -\frac{\partial z(x_i,y_i)}{\partial y_i}, 1\right]}{\sqrt{(\frac{\partial z(x_i,y_i)}{\partial x_i})^2 + (\frac{\partial z(x_i,y_i)}{\partial y_i})^2 + 1}} \tag{3.1}$$

3.2.1 Approximating Derivatives

In the discrete setting, $\frac{\partial z(x_i,y_i)}{\partial x_i}$ and $\frac{\partial z(x_i,y_i)}{\partial y_i}$ can be computed in two ways, by using either finite or central difference approximation [26].

Consider a function f. Expanding the functions $f(x + h)$ and $f(x - h)$ using the Taylor series gives:

$$f(x + h) = f(x) + h\frac{\partial f}{\partial x} + O(h) \tag{3.2}$$

$$f(x - h) = f(x) - h\frac{\partial f}{\partial x} + O(h) \tag{3.3}$$

By using either of the above expansions, we can derive two expressions for $\frac{\partial f}{\partial x}$:

$$\frac{\partial f}{\partial x} = \frac{f(x + h) - f(x)}{h} + O(h) \tag{3.4}$$

$$\frac{\partial f}{\partial x} = \frac{f(x) - f(x - h)}{h} + O(h) \tag{3.5}$$

These are both called the finite difference approximation. The first is the forward difference approximation and the second is the backward difference approximation. Both

have errors on the order of h. $\frac{\partial f}{\partial x}$ can also be calculated by subtracting (3.2) from (3.3):

$$\frac{\partial f}{\partial x} = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \tag{3.6}$$

This gives us another expression called central difference approximation. Notice that the error is on the order of h^2. This method is more accurate.

In our system, we have chosen to use finite differences instead of central differences whenever derivatives are needed. This choice was made because we found, through experimentation, that direct coupling between neighboring data-points is necessary otherwise the system sees our problem as separate sub-problems involving all normals at either odd or even spacing intervals on the height field.

The unit normal using finite difference approximation is:

$$\bar{\mathbf{n}}_i = \frac{[-(z(x_i-1, y_i) - z(x_i, y_i)), -(z(x_i, y_i-1) - z(x_i, y_i)), 1]}{\sqrt{(z(x_i-1, y_i) - z(x_i, y_i))^2 + (z(x_i, y_i-1) - z(x_i, y_i))^2 + 1}} \tag{3.7}$$

3.3 Affine Transformations and Projection

We define $\mathbf{p}_{i,t}$ as being the orthographically-projected, scaled, rotated, and translated surface point $\mathbf{s}_i$ at frame t:

$$\mathbf{s}_i = [x_i, y_i, z(x_i, y_i)]^T \tag{3.8}$$

$$\mathbf{p}_{i,t} = \rho_t \, \mathbf{P} \, \mathbf{R}_t \, \mathbf{s}_i + \mathbf{d}_t \tag{3.9}$$

where

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.10}$$

$$\mathbf{R}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_{x_t}) & -sin(\theta_{x_t}) \\ 0 & sin(\theta_{x_t}) & cos(\theta_{x_t}) \end{bmatrix} \begin{bmatrix} cos(\theta_{y_t}) & 0 & sin(\theta_{y_t}) \\ 0 & 1 & 0 \\ -sin(\theta_{y_t}) & 0 & cos(\theta_{y_t}) \end{bmatrix} \begin{bmatrix} cos(\theta_{z_t}) & -sin(\theta_{z_t}) & 0 \\ sin(\theta_{z_t}) & cos(\theta_{z_t}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

$$\mathbf{d}_t = \begin{bmatrix} d_{x_t} \\ d_{y_t} \end{bmatrix} \tag{3.12}$$

3.4 Intensity of projected point $\mathbf{p}_{i,t}$

Now that we have a way to transform and project the height field onto the screen, we must now determine the color and intensity of each projected point $\mathbf{p}_{i,t}$. There are two factors that will influence the observed $\mathbf{p}_{i,t}$: its RGB texture value and the shading value at that point. We associate an RGB vector $[\alpha_{r_i}, \alpha_{g_i}, \alpha_{b_i}]$ to each surface point s_i of our height field. The color at the projected point $\mathbf{p}_{i,t}$ will therefore be represented by this three-channel texture vector.

If the intensity of $\mathbf{p}_{i,t}$ were given simply by its texture, the resulting object would look very synthetic. The eye is very sensitive to shading cues, and in order to create a proper generative model for imaged objects, it is imperative to include shading.

We can incorporate shading information into our model by using the Lambertian lighting equation (2.15) (see section 2.2). This equation determines the intensity of a surface point by using the angle between the light source and the normal at that point.

Although this model alone is a very simple approximation to real world scenes, it can still provide the necessary cues to convince the eye of depth. We also add an ambient term, l_{a_t}, to simulate the arrival of light arriving from other objects in the scene.

The intensity value at $\mathbf{p}_{i,t}$, for each color channel c, is:

$$\bar{\mathbf{l}} = \frac{[l_x, l_y, 1]}{\sqrt{l_x^2 + l_y^2 + 1}} \tag{3.13}$$

$$\tilde{I}_{t,c}(\mathbf{p}_{i,t}) = \alpha_{c_i} \left(l_{a_t} + (\mathbf{R}_t \bar{\mathbf{n}}_i)^T \bar{\mathbf{l}} \right) \tag{3.14}$$

3.5 Probabilistic formulation for image generation

This section begins with a brief discussion on Gaussian distributions, followed by the specification for a robust generative model of image formation. A final section will outline the smoothness terms that are an important aspect of realism.

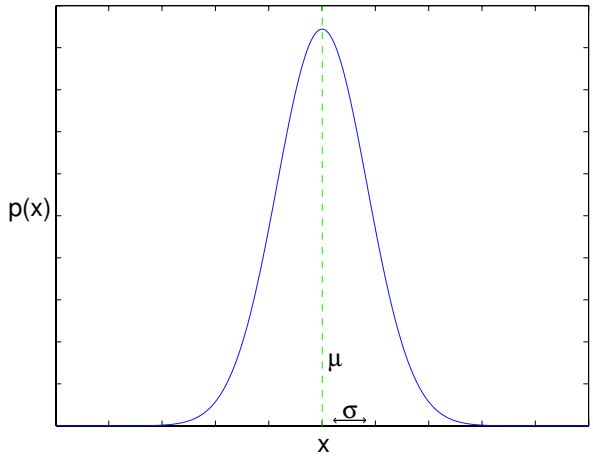

Figure 3.2: General form of Gaussian. The green line indicates the mean.

3.5.1 Gaussian distributions

Before we derive our probabilistic framework, it is imperative to first describe the functions we will be using to create our generative model.

A continuous random variable can be described using a number of distribution functions. By far the most popular function is the Gaussian, or normal, distribution given as:

$$\mathcal{N}\left(x|\mu;\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.15}$$

The Gaussian is a parameterizable distribution with mean and variance. Eq. (3.15) is plotted in figure 3.2.

This distribution has many appealing characteristics. Its parameters are intuitive since the distribution can be expressed in human-understandable terms. It is also believed that most naturally occurring phenomena are normally distributed due to the Central Limit Theorem [5] [20].

The multi-variate version of the Gaussian with covariance matrix Σ and dimensionality d given as follows:

$$\mathcal{N}\left(\mathbf{x}|\mu;\Sigma\right) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{\frac{-(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)}{2}} \tag{3.16}$$

For an isotropic Gaussian with diagonal covariance matrix $\sigma^2 I$, Eq. (3.16) reduces to:

$$\mathcal{N}\left(\mathbf{x}|\mu;\sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{\frac{-\|\mathbf{x}-\mu\|^2}{2\sigma^2}} \tag{3.17}$$

3.5.2 Robust imaging model

We would like to create a probability distribution such that random samples from it will yield a plausible sequence of images produced by the transformation, illumination and projection of the underlying parameterized patch.

The generative model can be derived from (3.14):

$$I_{t,c}(\mathbf{p}_{i,t}) = \tilde{I}_{t,c}(\mathbf{p}_{i,t}) + n \tag{3.18}$$

$$n \sim \mathcal{N}\left(0;\sigma^2_{image}\right) \tag{3.19}$$

This generative model will always create points that are visible. That is, there will always be a value for all $I_{t,c}(\mathbf{p}_{i,t})$ that corresponds to a projected colored surface point. What if we wish to generate an imaged object that is partially occluded? With the current model this is impossible. In order to add more flexibility into our model, we should also allow for the generation of outliers, or pixels that have not been generated from the object.

A simpler example can be used to clarify outlier generation. Imagine a generative model that creates points on the 2d plane about a line, with Gaussian noise. Imagine this line represents some real-world statistical correlation, and we want to design a model that generates points that such a line could pass through. The equation could be as follows:

$$\bar{y}_i = mx_i + b \tag{3.20}$$

$$y_i = \bar{y}_i + n \tag{3.21}$$

$$n \sim \mathcal{N}\left(0;\sigma^2\right) \tag{3.22}$$

This model will always generate points around the line (figure 3.3).

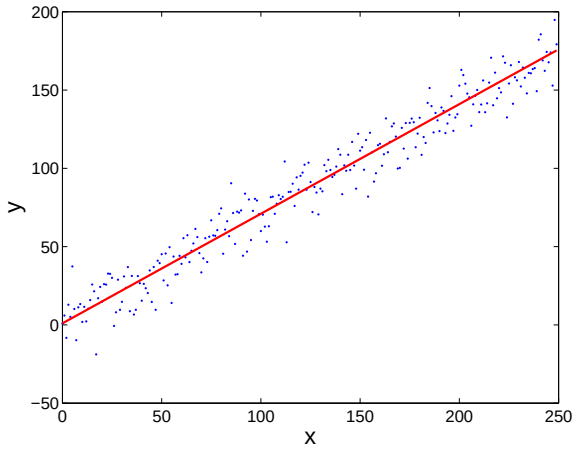

Figure 3.3: Graphical depiction of 3.21 for $m = 0.7$, $b = 1$ and $\sigma^2 = 144$. The red line represents the underlying line that generated the data ($y = 0.7x + 1$)

If this model were generating the results of an experiment, the measurements would never give points that all lie so close to the line. There will certainly be outliers, or points that should be ignored because they violate the general trend found in the data.

We could imagine creating such outlier points by first deciding whether a point should be an outlier, and then sampling from a secondary distribution to determine what value it will be. We define a hidden variable, W_i, where 0 indicates an outlier and 1 indicates a valid point. If the point is not an outlier, we sample from our original line-generating model. If it is an outlier, we generate a random value. This is equivalent to sampling from a uniform distribution c.

$$p(W_i = 1) = \tau \tag{3.23}$$

$$p(y_i|\bar{y}_i, W_i = 1, \sigma^2) = \mathcal{N}\left(y_i|\bar{y}_i; \sigma^2\right) \tag{3.24}$$

$$p(y_i|\bar{y}_i, W_i = 0, \sigma^2) = c \tag{3.25}$$

Sampled points from this model are shown in figure 3.4.

We basically follow the same scheme as above. We create a mixture of two distributions: one being our image distance function derived above, and the other a distribution that generates outlier points. It must be noted that the uniform outlier distribution, c,

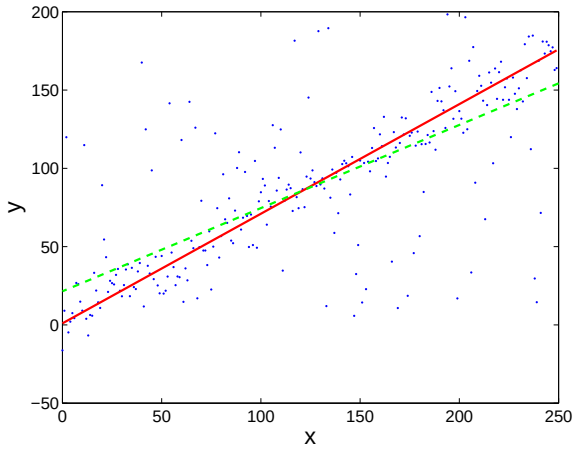

Figure 3.4: Graphical depiction of 3.24 for $m = 0.7$, $b = 1$, $\sigma^2 = 144$ and $\tau = 0.3$. The red line is the line that generated the valid points (3.24). The green line is the least-squares linear fit to all points, valid and outlier.

is a simplified interpretation of what outliers should be. For instance, we know more about occlusions than simply that they come from a uniform distribution. If a hand is occluding the object, there is some "structure" there, notably the area occupied by the hand, it's color etc... This is **not** being modeled here.

Below shows the more robust model used in our system.

RGB color-values at $\mathbf{p}_{i,t}$:

$$\eta_{i,t} = \{I_{t,r}(\mathbf{p}_{i,t}), I_{t,g}(\mathbf{p}_{i,t}), I_{t,b}(\mathbf{p}_{i,t})\} \tag{3.26}$$

$$\tilde{\eta}_{i,t} = \{\tilde{I}_{t,r}(\mathbf{p}_{i,t}), \tilde{I}_{t,g}(\mathbf{p}_{i,t}), \tilde{I}_{t,b}(\mathbf{p}_{i,t})\} \tag{3.27}$$

Generative model:

$$p(W_{i,t} = 1) = \tau \tag{3.28}$$

$$p(\eta_{i,t}|\tilde{\eta}_{i,t}, W_{i,t} = 1, \sigma^2_{image}) = \prod_c \mathcal{N}\left(I_{t,c}(\mathbf{p}_{i,t})|\tilde{I}_{t,c}(\mathbf{p}_{i,t}); \sigma^2_{image}\right) \tag{3.29}$$

$$p(\eta_{i,t}|\tilde{\eta}_{i,t}, W_{i,t} = 0, \sigma^2_{image}) = c \tag{3.30}$$

where $\tilde{I}_{t,c}(\mathbf{p}_{i,t})$ is given in (3.14).

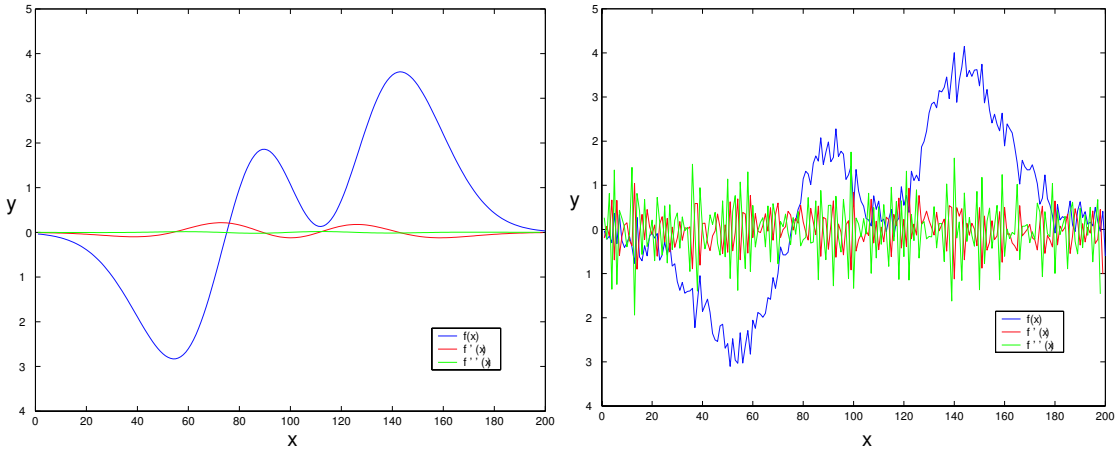

Figure 3.5: Smooth and non-smooth curves with their first and second derivatives

3.5.3 Priors

We have thus far explained the portion of the generative model that generates the resulting images. It is also possible, however, to inject additional information on what we assume about the parameters themselves. These are called priors.

For instance, we assume that objects typically move smoothly from frame to frame. Objects that translate across large distances extremely quickly are very rare, and such motions should therefore be associated with small probabilities.

In our implementation, priors are used to say: "We assume smooth parameters, since most of the objects we see are smooth, either temporally or spatially".

Smoothness can be measured by examining either the first or second derivatives of a curve (figure 3.5). When looking at first derivatives, we are measuring the proximity of neighboring points on the curve. Second derivatives measure the difference between neighboring first derivatives, or how fast the curve is changing direction. Our system uses second derivatives for smoothness enforcement, since it penalizes "noisy" curves.

Using finite differences, the general expression for this difference of derivatives is:

$$\frac{\partial^2 f}{\partial x_i^2} \approx \frac{\partial f}{\partial x_i} - \frac{\partial f}{\partial x_{i-1}} \tag{3.31}$$

$$\approx (f(x_{i+1}) - f(x_i)) - (f(x_i) - f(x_{i-1})) \tag{3.32}$$

$$\approx (f(x_{i+1}) - 2f(x_i) + f(x_{i-1})) \tag{3.33}$$

This is also a way of saying that the value of a point on the curve should be midway between the neighboring points. We can create a Gaussian distribution that expresses this:

$$p(f(x_1), f(x_2), \ldots, f(x_n), \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{1}{2\sigma^2}\sum_i (f(x_{i+1}) - 2f(x_i) + f(x_{i-1}))^2} \tag{3.34}$$

By changing the variance of this Gaussian, we are essentially changing the level of acceptable smoothness. This will be the general form of our prior probabilities.

The objects that we will be reconstructing are generally smooth. Neighboring height field values will generally not be very different from each other. This can be incorporated into our generative model stating that the second derivative of all points on the surface should be low:

$$v_{i_1} = z(x_i + 1, y_i) - 2z(x_i, y_i) + z(x_i - 1, y_i) \tag{3.35}$$

$$v_{i_2} = z(x_i, y_i + 1) - 2z(x_i, y_i) + z(x_i, y_i - 1) \tag{3.36}$$

$$p(z(x_1, y_1), z(x_2, y_2), \ldots, z(x_n, y_n), \sigma^2_{shape}) = \frac{1}{(2\pi\sigma^2_{shape})^N} e^{-\frac{1}{2\sigma^2_{shape}}\sum_i \left(v_{i_1}^2 + v_{i_2}^2\right)} \tag{3.37}$$

Smoothness priors can be applied to transformations as well. By this we refer to rotations, translations and scaling. Our assumptions state that the smoother these entities are, the more realistic the movement. These priors are all implemented in the same form as previously discussed. For clarity, we show the smoothness term for scaling:

$$p(\rho_1, \rho_2, \ldots, \rho_t, \sigma^2_{scale}) = \frac{1}{(2\pi\sigma^2_{scale})^{\frac{T}{2}}} e^{-\frac{1}{2\sigma^2_{scale}}\sum_t (\rho_{t+1} - 2\rho_t + \rho_{t-1})^2} \tag{3.38}$$

3.6 Summary

The complete probability distribution of all variables in our generative model is given below.

Camera parameters:

$$\zeta = \{\mathbf{P}, \rho_1, \theta_{x,1}, \theta_{y,1}, \theta_{z,1}, t_{x_1}, t_{y_1}, \ldots, \rho_t, \theta_{x,t}, \theta_{y,t}, \theta_{z,t}, t_{x_t}, t_{y_t}\} \tag{3.39}$$

Shape and texture parameters:

$$\beta = \{z(x_1, y_1), \alpha_{r_1}, \alpha_{g_1}, \alpha_{b_1}, \ldots, z(x_i, y_i), \alpha_{r_i}, \alpha_{g_i}, \alpha_{b_i}\} \tag{3.40}$$

Model parameters:

$$\gamma = \{\tau, l_{a_1}, \ldots, l_{a_t}, l_x, l_y, \sigma^2_{image}, \sigma^2_{scale}, \sigma^2_{rot}, \sigma^2_{trans}\} \tag{3.41}$$

RGB image intensity values:

$$\kappa = \{I_{1,r}(\mathbf{p}_{1,1}), I_{1,g}(\mathbf{p}_{1,1}), I_{1,b}(\mathbf{p}_{1,1}), \ldots, I_{t,r}(\mathbf{p}_{i,t}), I_{t,g}(\mathbf{p}_{i,t}), I_{t,b}(\mathbf{p}_{i,t})\} \tag{3.42}$$

Complete model:

$$p(\zeta, \beta, \gamma, \kappa) = p(\kappa|\zeta, \beta, \gamma)p(\zeta, \beta, \gamma) \tag{3.43}$$

$$= \underbrace{\prod_{i,t}\left(\sum_{W \epsilon 0,1} p(\{I_{t,c}(\mathbf{p}_{i,t})\}_{c=r,g,b}|\{\tilde{I}_{t,c}(\mathbf{p}_{i,t})\}_{c=r,g,b}, W_{i,t}, \sigma^2_{image})p(W_{i,t})\right)}_{Image-generating\,term}$$

$$\underbrace{p(z(x_1, y_1), \ldots, z(x_n, y_n), \sigma^2_{shape})}_{Shape\,prior}\underbrace{p(\rho_1, \ldots, \rho_t, \sigma^2_{scale})}_{Scale\,prior}$$

$$\underbrace{p(t_{x_1}, \ldots, t_{x_t}, \sigma^2_{trans})p(t_{y_1}, \ldots, t_{y_t}, \sigma^2_{trans})}_{Translation\,prior}$$

$$\underbrace{p(\theta_{x_1}, \ldots, \theta_{x_t}, \sigma^2_{rot})p(\theta_{y_1}, \ldots, \theta_{y_t}, \sigma^2_{rot})p(\theta_{z_1}, \ldots, \theta_{z_t}, \sigma^2_{rot})}_{Rotation\,prior} \tag{3.44}$$

$$
\begin{aligned}
= \quad & \prod_{i,t}\left(\left(\prod_{c}\mathcal{N}\left(I_{t,c}(\mathbf{p}_{i,t})|\tilde{I}_{t,c}(\mathbf{p}_{i,t});\sigma_{image^2}\right)\right)\tau+(1-\tau)c\right) \\
& \frac{1}{(2\pi\sigma^2_{shape})^N}e^{-\frac{1}{2\sigma^2_{shape}}\sum_i(z(x_i+1,y_i)-2z(x_i,y_i)+z(x_i-1,y_i))^2+(z(x_i,y_i+1)-2z(x_i,y_i)+z(x_i,y_i-1))^2} \\
& \frac{1}{(2\pi\sigma^2_{scale})^{\frac{T}{2}}}e^{-\frac{1}{2\sigma^2_{scale}}\sum_t(\rho_{t+1}-2\rho_t+\rho_{t-1})^2} \\
& \frac{1}{(2\pi\sigma^2_{trans})^{T}}e^{-\frac{1}{2\sigma^2_{trans}}\sum_t\left(t_{x_{t+1}}-2t_{x_t}+t_{x_{t-1}}\right)^2+\left(t_{y_{t+1}}-2t_{y_t}+t_{y_{t-1}}\right)^2} \\
& \frac{1}{(2\pi\sigma^2_{rot})^{\frac{3T}{2}}}e^{-\frac{1}{2\sigma^2_{rot}}\sum_t\left(\theta_{x_{t+1}}-2\theta_{x_t}+\theta_{x_{t-1}}\right)^2+\left(\theta_{y_{t+1}}-2\theta_{y_t}+\theta_{y_{t-1}}\right)^2+\left(\theta_{z_{t+1}}-2\theta_{z_t}+\theta_{z_{t-1}}\right)^2} \qquad (3.45) \\
= \quad & (2\pi)^{-(N+3T)}\left(\frac{1}{\sigma^2_{shape}}\right)^N\left(\frac{1}{\sigma^2_{scale}}\right)^{\frac{T}{2}}\left(\frac{1}{\sigma^2_{trans}}\right)^T\left(\frac{1}{\sigma^2_{rot}}\right)^{\frac{3T}{2}} \\
& \prod_{i,t}\left(\left(\left(\frac{1}{2\pi\sigma^2_{image}}\right)^{\frac{3}{2}}e^{-\frac{1}{2\sigma^2_{image}}\sum_c\left(I_{t,c}(\rho_{\mathbf{t}}\,\mathbf{P}\,\mathbf{R_t}\,\mathbf{s_i}+\mathbf{d_t})-\alpha_{c_i}\left(l_{a_t}+(\mathbf{R}_t\bar{\mathbf{n}}_i)^T\bar{\mathbf{l}}\right)\right)^2}\right)\tau+(1-\tau)c\right) \\
& e^{-\frac{1}{2\sigma^2_{shape}}\sum_i(z(x_i+1,y_i)-2z(x_i,y_i)+z(x_i-1,y_i))^2+(z(x_i,y_i+1)-2z(x_i,y_i)+z(x_i,y_i-1))^2} \\
& e^{-\frac{1}{2\sigma^2_{scale}}\sum_t(\rho_{t+1}-2\rho_t+\rho_{t-1})^2} \\
& e^{-\frac{1}{2\sigma^2_{trans}}\sum_t\left(t_{x_{t+1}}-2t_{x_t}+t_{x_{t-1}}\right)^2+\left(t_{y_{t+1}}-2t_{y_t}+t_{y_{t-1}}\right)^2} \\
& e^{-\frac{1}{2\sigma^2_{rot}}\sum_t\left(\theta_{x_{t+1}}-2\theta_{x_t}+\theta_{x_{t-1}}\right)^2+\left(\theta_{y_{t+1}}-2\theta_{y_t}+\theta_{y_{t-1}}\right)^2+\left(\theta_{z_{t+1}}-2\theta_{z_t}+\theta_{z_{t-1}}\right)^2} \qquad (3.46)
\end{aligned}
$$

Chapter 4

The Problem Statement for Shape from Video

Given a video sequence I of a rigid object, we would like to learn the 3D rigid shape, motion and texture of the underlying object assuming scaled orthographic projection and Lambertian illumination.

4.1 Maximum A Posteriori (MAP) Estimation

We wish to obtain the parameters ζ, β, γ that produce the highest probability $p(\zeta, \beta, \gamma | I)$. Using Bayes' rule:

$$p(\zeta, \beta, \gamma | I) = \frac{p(I|\zeta, \beta, \gamma) p(\zeta, \beta, \gamma)}{p(I)} \tag{4.1}$$

The denominator, or evidence, can be ignored, as it is completely independent of the unknowns ζ, β and γ. We are therefore left with the likelihood term and priors (3.43). Maximizing these together with respect to the parameters is our goal.

We need to convert our probabilistic formulation into an optimization problem. Optimization is concerned with the minimization of an objective function that depends on real variables.

We could simply take our probabilistic formulation and try and minimize the negative probability by finding the suitable parameters. It is common practice, however, to minimize the negative log-probability of the expression. Taking the log allows us to deal with larger-scale numbers, and we can avoid common rounding errors that occur when multiplying many small floating point numbers together, as is the case with the multiplication of many probability terms.

Our optimization objective (or error) function therefore becomes:

$$
\begin{aligned}
E &= -\ln\left(p(I|\zeta,\beta,\gamma)p(\zeta,\beta,\gamma)\right) \\
&= \underbrace{-\sum_{i,t}\ln\left(\left(\left(\frac{1}{2\pi\sigma^2_{image}}\right)^{\frac{3}{2}} e^{-\frac{1}{2\sigma^2_{image}}\sum_c\left(I_{t,c}(\rho_{\mathbf{t}}\,\mathbf{P}\,\mathbf{R_t}\,\mathbf{s_i}+\mathbf{d_t})-\alpha_{c_i}\left(l_{a_t}+(\mathbf{R}_t\bar{\mathbf{n}}_i)^T\bar{\mathbf{l}}\right)\right)^2}\right)\tau+(1-\tau)c\right)}_{E_{image}} \\
&\quad \underbrace{+\frac{1}{2\sigma^2_{shape}}\sum_i\Big(\left(z(x_i+1,y_i)-2z(x_i,y_i)+z(x_i-1,y_i)\right)^2 + \left(z(x_i,y_i+1)-2z(x_i,y_i)+z(x_i,y_i-1)\right)^2\Big)}_{E_{shape}} \\
&\quad \underbrace{+\frac{1}{2\sigma^2_{scale}}\sum_t\left(\rho_{t+1}-2\rho_t+\rho_{t-1}\right)^2}_{E_{scale}} \\
&\quad \underbrace{+\frac{1}{2\sigma^2_{trans}}\sum_t\left(t_{x_{t+1}}-2t_{x_t}+t_{x_{t-1}}\right)^2+\left(t_{y_{t+1}}-2t_{y_t}+t_{y_{t-1}}\right)^2}_{E_{trans}} \\
&\quad \underbrace{+\frac{1}{2\sigma^2_{rot}}\sum_t\left(\theta_{x_{t+1}}-2\theta_{x_t}+\theta_{x_{t-1}}\right)^2+\left(\theta_{y_{t+1}}-2\theta_{y_t}+\theta_{y_{t-1}}\right)^2+\left(\theta_{z_{t+1}}-2\theta_{z_t}+\theta_{z_{t-1}}\right)^2}_{E_{rot}} \\
&\quad \underbrace{+(N+3T)\ln(2\pi)+N\ln(\sigma^2_{shape})+\frac{T}{2}\ln(\sigma^2_{scale})+T\ln(\sigma^2_{trans})+\frac{3T}{2}\ln(\sigma^2_{rot})}_{E_{norm}} \qquad (4.2)
\end{aligned}
$$

Chapter 5

Optimization

In the previous chapter we derived the objective function that we want to minimize. This chapter discusses a way in which to attain this goal: using numerical optimization.

We discuss the optimization algorithm, constraints, and specific initializations and optimization schedules used in our system. We close the chapter with an analysis and comparison of other optimization techniques.

5.1 Overview

One of the best performing general optimization algorithms is the quasi-Newton method BFGS [26], named after its inventors: Broyden, Fletcher, Goldfarb and Shanno. BFGS uses both the objective function evaluation and the gradient at the current location to make decisions about future iterates. Our system uses the large scale L-BFGS implementation by Zhu et al. [45].

While calculating the gradient for each step may seem heavy, the overhead can be alleviated by properly optimizing the code. In most optimization problems (and particularly least squares-like problems), values that were calculated in the objective function evaluation step are always reused for gradients. Caching is essential for speedups.

For simplicity, we show the gradient of the image-matching term (E_{image}) taken with

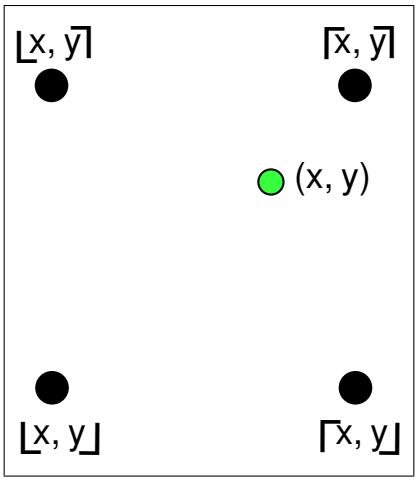

Figure 5.1: A point with it's four closest neighbors

respect to scaling.

$$\frac{\partial E_i}{\partial \rho_t} = -\sum_i \left(\frac{\frac{\tau}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2\sigma_i^2}\sum_c (I-\tilde{I})^2}}{\frac{\tau}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2\sigma_i^2}\sum_c (I-\tilde{I})^2} + (1-\tau)} \right) - \frac{1}{\sigma_i^2} \sum_c \left[(I-\tilde{I})((\nabla I)^T \mathbf{P}\mathbf{R}_t \mathbf{s}_i) \right] \quad (5.1)$$

where σ^2_{image} and E_{image} have been replaced by σ_i^2 and E_i respectively.

5.1.1 I and ΔI

The image consists of a discrete set of pixels over three channels. We are required, however, to provide values for both the image and its gradient (5.1). We represent each channel using linear basis functions. This means that values anywhere on the image are bilinearly interpolated using the four closest neighbors (figure 5.1). This is done for each channel.

The value of $I(x, y)$ is:

$$p = x - \lfloor x \quad (5.2)$$

$$q = y - y\rfloor \quad (5.3)$$

$$\bar{x} = \lceil x - \lfloor x \quad (5.4)$$

$$\bar{y} = y\rceil - y\rfloor \quad (5.5)$$

$$I(x,y) = (1-p)qI_{\lfloor x,y\rceil} + pqI_{\lceil x,y\rceil} + p(1-q)I_{\lceil x,y\rfloor} + (1-p)(1-q)I_{\lfloor x,y\rfloor} \tag{5.6}$$

The gradient $\nabla I(x,y)$ can be calculated by using (5.6) and replacing p and q as follows:

$$p = 1 - \lfloor x \tag{5.7}$$

$$q = 1 - y\rfloor \tag{5.8}$$

5.1.2 Constraints

The optimization, in its current form, does not take into account constraints we may wish to place on some variables. For instance, a variable such as the variance of a Gaussian (σ^2) must always be positive.

BFGS supports constrained optimization by using the projected gradient method [26]. In this technique, variables are monitored for constraint violation. When they are violated, they become "active", and the current downhill direction is projected onto another manifold of active constraints, effectively bending the direction so as to respect the bounds.

It is possible, however, to convert a constrained optimization problem into one which is unconstrained. This has the advantage of simplifying the problem and allowing experimentation by using other optimization algorithms that may not inherently support constraint management.

For σ^2, one way to ensure that it never goes below zero during optimization is with a change of variables: $\sigma^2 = \exp(a)$, and we now optimize a instead of σ^2.

Constraints are also needed for τ and the RGB values. In these cases, the variables must be between 0 and 1. This can be achieved with another change of variables: $\tau = \frac{1}{1+\exp(-b)}$, and we again solve for b instead of τ.

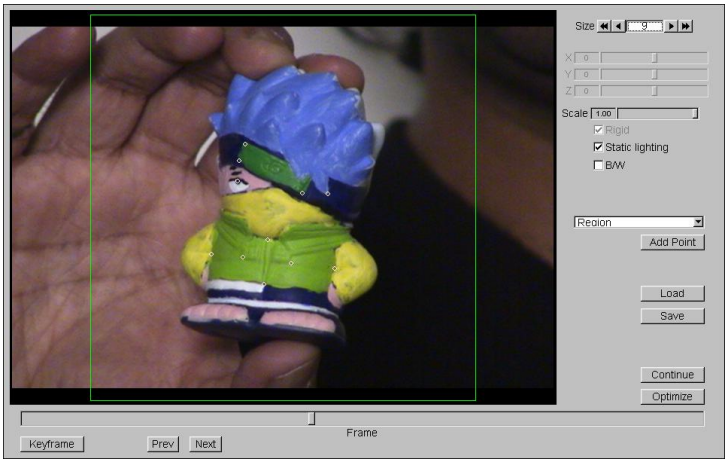

Figure 5.2: Our system's GUI

5.2 Optimization schedule

This section will cover the specifics of initialization followed by details of the full optimization procedure.

5.2.1 Initialization

Simply starting from a random location on the objective function and solving for all variables with BFGS will, in most cases, not work. The objective function is a high dimensional manifold that contains many local minima, most of which are visually unacceptable solutions. Starting points can be crucial for proper convergence.

Before the full optimization can take place, we must find a good starting location. As long as we can initialize the algorithm by placing the starting location in the general vicinity of plausible solutions, we dramatically increase the chances of descending into an acceptable local minimum.

When we refer to general vicinity, we are referring to "good guesses" for all of the parameters. This is done by solving two preliminary optimizations and using these results to initialize the main optimization step. These two optimizations also make use of BFGS.

The first step involves solving for the scale, rotation, translation and shape based on

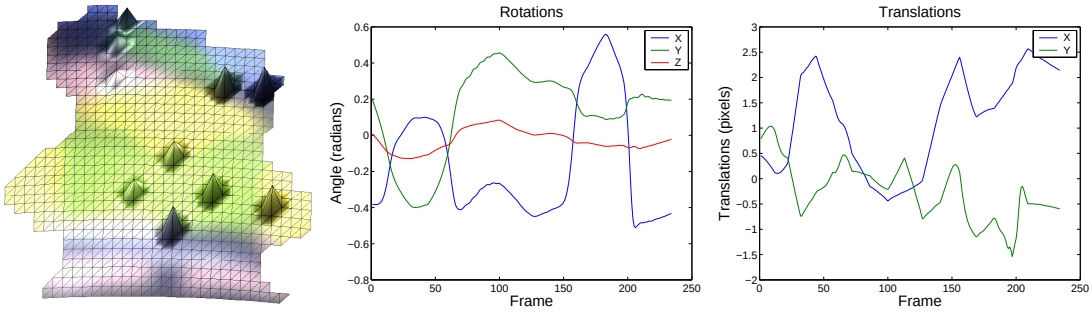

Figure 5.3: Recovered shape, rotations and translations from the tracking stage

selected user-tracked points across the video sequence (figure 5.2). More specifically, we are trying to find the values of these parameters that will project these selected points onto the image plane at the same locations that the user has specified ($\mathbf{p_{user}}_{j,t}$). This corresponds to a least squares optimization:

$$\mathbf{rot}_t = \begin{bmatrix} \theta_{x,t+1} - 2\theta_{x,t} + \theta_{x,t-1} \\ \theta_{y,t+1} - 2\theta_{y,t} + \theta_{y,t-1} \\ \theta_{z,t+1} - 2\theta_{z,t} + \theta_{z,t-1} \end{bmatrix} \tag{5.9}$$

$$\begin{aligned} E_{track} = & \frac{\lambda_{track}}{2} \sum_{j,t} \|\mathbf{p}_{j,t} - \mathbf{p_{user}}_{j,t}\|^2 + \frac{\lambda_{height}}{2} \sum_{j} (z(x_j, y_j))^2 \\ & + \frac{\lambda_{rot}}{2} \sum_{t} \|\mathbf{rot}_t\|^2 \end{aligned} \tag{5.10}$$

The second term, which can be thought of as a prior on the shape, is there to ensure that the height field values are not too large, and will not yield a surface that is far from the XY plane. The third term is a prior on rotations. We found this necessary to reduce sporadic jumps from θ to $\theta + 2\pi$. Figure 5.3 shows recovered shape and transformations from this tracking step.

Once these sparse height values have been found, the rest of the surface is solved for by simply applying the surface smoothness constraint, but holding the points $\mathbf{p_{user}}_{j,t}$ constant so that the surface passes through the user specified points (figure 5.4).

We now have initial values for scale, rotation, translation and shape. We also require

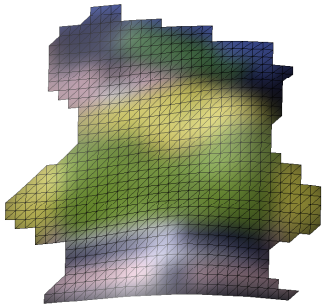

Figure 5.4: Smooth surface by applying the smoothness constraint to the tracked surface

Figure 5.5: Coarse-to-fine progression

initializations for lighting, texture, τ and all σ^2 values. We now run a second optimization step to determine these values.

This can be done by solving for these variables using the full objective function and holding the others variables (the ones we solved for in the previous step) constant. This will leave us with all variables as coherent "good guesses".

5.2.1.1 Full optimization step

Once all initialization is complete, all variables are solved for. The algorithm proceeds in a coarse-to-fine progression in order to resolve temporal aliasing issues (figure 5.5).

At the beginning of each step in the coarse-to-fine refinement, τ and all σ^2 values are solved for alone, as they are the only parameters that are not updated when the resolution is doubled. They must be calculated before continuing so that they match the rest of the data.

5.2.1.2 Summary

1. User selection of feature points across frames

2. Initial optimization to recover shape and motion of feature points

3. Create a smooth surface that passes through the feature points

4. Optimize full objective function using to coarse-to-fine procedure

5.3 Pre-conditioning

Szeliski [32] and Gortler et al. [13] have shown that convergence can be improved if we represent geometry using hierarchical basis functions instead a linear finite element basis that provides only local support. Instead of representing our height field as a set of individual nodes that can move up or down, we use a wavelet basis [31] where coefficients can influence many grid points. This way the optimization can easily make broad changes when needed.

In our system, heights fields are not the only place where such a representation can be beneficial. Since the image-matching term is dependent on both geometry and texture, it is logical to represent all three texture channels in the wavelet form as well.

5.4 Other attempted optimization methods

BFGS is not the only way to optimize. In fact, gradient-based optimization is not the only way to minimize the objective function. This section deals with other methods that were attempted but did not prove satisfactory.

5.4.1 Stochastic gradient descent method

Calculating the full gradient at each step can be quite time consuming. Blanz et al. [6], Jones et al. [19] and Viola and Wells [39] have shown that it is possible to obtain convergence at greater speeds by using the stochastic gradient descent (SGD) method.

In this method, an estimate for the gradient is obtained by calculating it with a random subset of the data. A step is then taken in this direction as it would be in regular gradient descent. These estimates must be unbiased for this technique to be effective [39]. It is also believed that local minima can be avoided due to the noise introduced by the sampling. Furthermore, since less is being calculated, convergence will be attained at faster speeds.

Traditional SGD simply uses a learning rate that decreases over time [5]. Our first implementation of this method performed poorly due to the fact that the step lengths were fixed and did not incorporate local energy-surface information. We decided to develop a custom line search algorithm, where steps are taken by searching along the gradient direction for the best distance to move. This was done because we suspected that larger steps were being overlooked, whereas BFGS relies on such strategies to take giant steps if necessary.

Line search algorithms typically utilize the gradient and objective function evaluations to model a quadratic function (or cubic, if possible) that approximates the true slice of the objective function in the chosen direction. The minimum of this polynomial is found and then tested for acceptability. Should it not be acceptable, another polynomial is modeled that incorporates this new piece of information, and again tested. This process continues until an adequate step length is found [26]. BFGS requires both error function and gradient calculations per step, therefore we assume that it is modeling a cubic internally (figure 5.6).

Our implementation is similar; however since we are not calculating the true gradient we cannot use it to model our polynomial. The gradient and objective function error

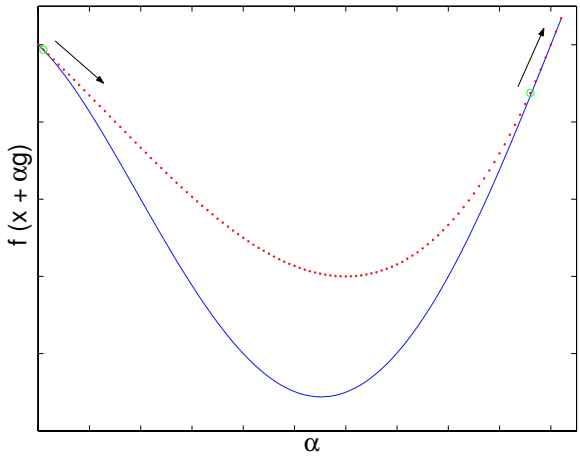

Figure 5.6: Line search by cubic approximation (red) of the true slice $f(\mathbf{x} + \alpha\mathbf{g})$. Four pieces of information are needed to model the cubic. The two function evaluations are represented by green circles, along with the gradients at those points.

at a particular location must coincide with each other or else the polynomial will be completely invalid. We can, however, simply fit a polynomial through data-points and ignore gradient information altogether. If we evaluate the error function at three separate locations, we can do quadratic approximation (figure 5.7).

Unfortunately this increases computation time per step as we now must evaluate the error function three times and the gradient once before modeling the polynomial, whereas before we only evaluate the error function twice and compute gradients along the way. It is also important to note that when BFGS moves to a new location, it starts the next iteration from that location with the gradient that it calculated previously. SGD does not. Since we must evaluate noisy gradients without biasing, we must recompute the starting gradient at the beginning of each iteration thereby adding more computation time. We also lose a little bit of precision, since we are modeling a quadratic and not a cubic. Nonetheless, we decided to implement it anyways for if it found a local minimum in less iterations than BFGS, it would ultimately be worth it.

The custom line search technique performed significantly better than the learning rate implementation however it simply does not come close to the performance of BFGS

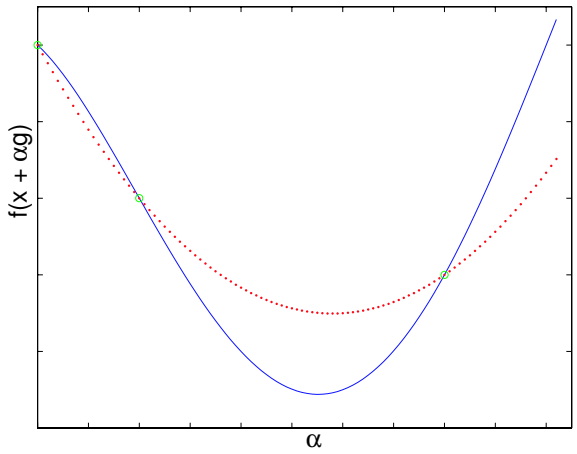

Figure 5.7: Line search by quadratic approximation (red) of the true slice $f(\mathbf{x} + \alpha\mathbf{g})$. Three pieces of information are needed to model the quadratic. The three function evaluations are represented by green circles.

(figure 5.8). This is certainly due to the absence of the second order term approximated by BFGS. We also suspect that our error function is characterized by long valleys, scenarios where gradient descent is notorious for poor convergence. In addition to this, the sheer number of parameters in our optimization is excessively more than those of Blanz et al. [6] and Jones et al. [19], which means our problem contains much more local minima. Finally, the fact that we have not gained tremendously in speed due to our line search technique was enough to cause us to abandon this route. All problems are different, and choosing the right optimization scenario will differ from problem to problem.

5.4.2 Solution by coordinate ascent

BFGS finds a step direction by minimizing the local quadratic approximation of the objective function at the current location. As an alternative optimization method, we can also linearize the image function I around the current point $\left(\hat{I}_{c,t}(\mathbf{p}_{i,t}) = I_{c,t}(\mathbf{p}_c) + \nabla I^T(\mathbf{p}_{i,t} - \mathbf{p}_c)\right)$ and insert it into our objective function.

We can then take the derivatives of the objective function with respect to all variables

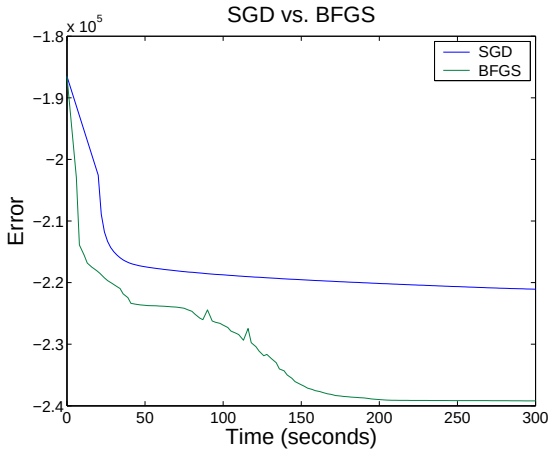

Figure 5.8: BFGS outperforms SGD. Our line-search implementation of SGD decreases the error function slower and ultimately less than BFGS.

and set each equation equal to zero and solve for the new value of the variable in question. This amounts to solving a series of univariate optimization problems for each variable update.

Although this is possible with the Newton's method or the secant method [28] [26] [20], the computation time increases significantly when compared to gradient-based optimization. This is principally due to the fact that once a variable has been updated, all calculations that involve this variable (therefore almost all calculations performed) must be recalculated before solving for the next one. With gradient-based methods these variable updates do not occur and we can store previously calculated information and reuse it when necessary (which is very often). This results in a significant speedup.

Another negative aspect of this method occurs when variables are tightly coupled, and attempting to move them individually simply does not work. An example would be wavelet coefficients. They are all tightly coupled to each other and therefore must change together, and not individually. There are multidimensional versions of the secant method [20] that will work, however computation time is excruciatingly slow. Essentially the full Hessian is calculated, which is something we have been trying to avoid since the beginning.

Chapter 6

Results

Our system has been tested on five video sequences. In each instance, a stationary camera is capturing a rigid object undergoing rotations and translations.

Due to large optimization times, we were not able to operate on all images for each coarse-to-fine level. The recovered transformations, therefore, are "coarse" since we linearly interpolate between recovered values. We also prematurely stop each coarse-to-fine iteration after 3000 iterations, a value found by trial and error. If it were possible to optimize to completion and use all frames, we expect the resulting reconstructions to have much more surface detail. For completeness, the optimization schedule is given for each experiments.

Not all experiments calculated scaling. This was "turned off" if deemed unnecessary and in situations where it was causing the surface to "shrink" significantly, thereby recovering only a portion of the intended surface. Ambient light was not always calculated either. When it was, it was either on a per-frame basis, or as a global ambient term for the whole sequence. Again, this was done on a trial-and-error basis.

Our system operates on blurred versions of the originals in order to reduce noise. The screenshots depicted in this section are the original un-blurred versions, therefore the recovered texture maps will look slightly blurred (figure 6.1)

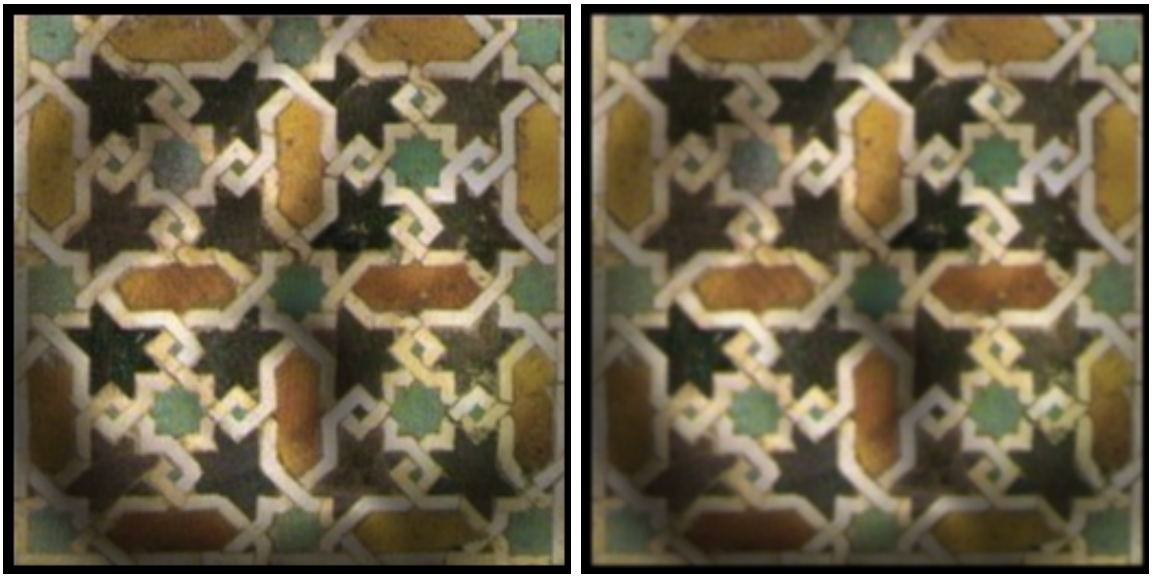

Figure 6.1: Cropped original frame (left) and blurred version used by our system (right)

6.1 Computer generated sequence

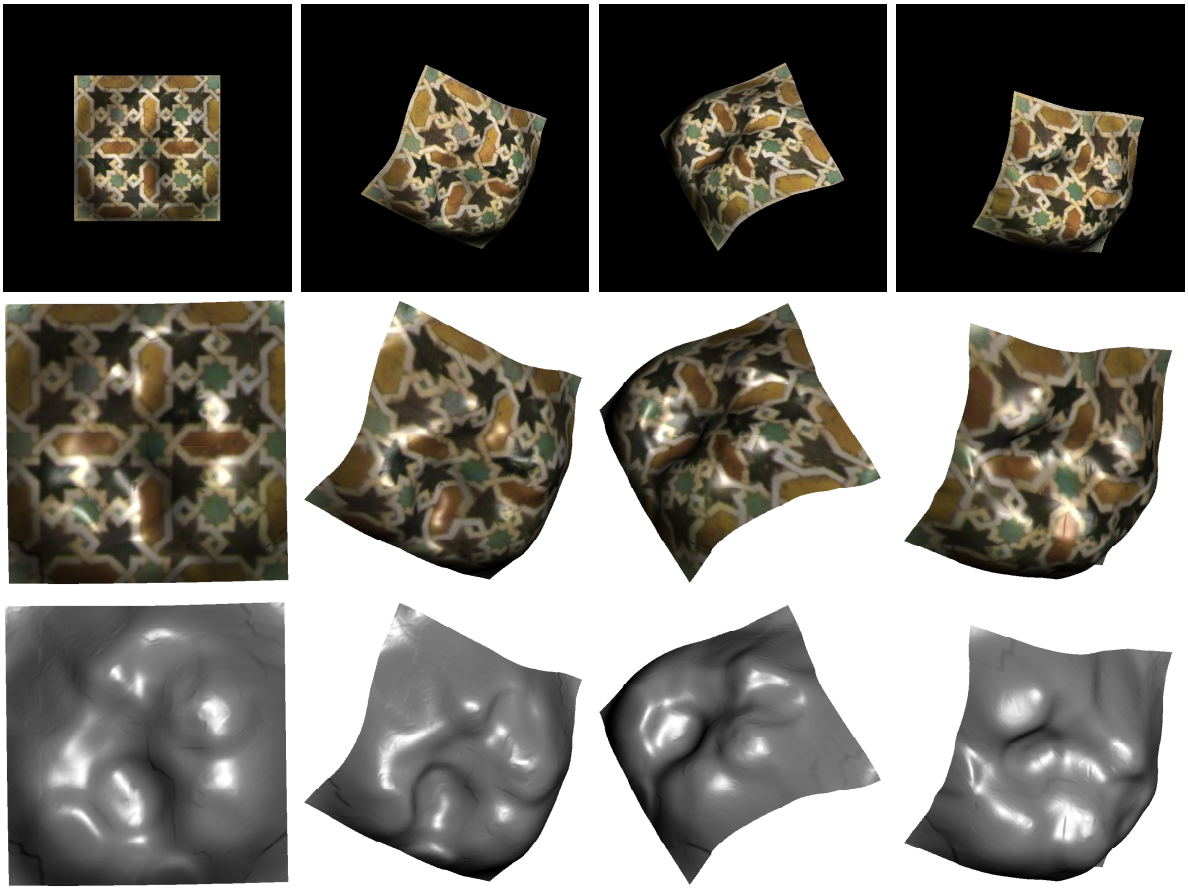

Figure 6.2: 4 frames from a generated 200 frame image sequence (top), and the textured (middle) and untextured (bottom) reconstructions

In order to demonstrate our system's ability to reconstruct shape, motion and lighting, we present our results on synthetic data. This will enable us to compare the reconstruction with "ground truth" data.

Figure 6.2 (top) shows four frames from a sequence generated with OpenGL. It consists of a textured height field undergoing rotations and translations. As the object is rotated, its edges are occluded. For simplicity, there is no ambient light or scaling. The middle and bottom rows of figure 6.2 are the reconstructions of those frames using the recovered shape and transformations.

Resolution	Number of Frames
32 × 32	200
64 × 64	100
128 × 128	50
256 × 256	25

Figure 6.3: Optimization Schedule for computer generated sequence

Table 6.3 shows the number of frames processed per level in the coarse-to-fine progression. The entire optimization took 6 days on a 3GHz/2G RAM desktop.

σ^2_{image}	2.3715×10^{-5}
σ^2_{smooth}	0.0031
τ	0.9434
light	[0.195, -0.197, -1.000]

Figure 6.4: Recovered parameters for the computer generated sequence

Table 6.4 shows the parameters that were estimated. The very low variance σ^2_{image} indicates that the imaged model very closely matches the input sequence. The recovered light source direction closely matches the input sequence's light source direction of $[0.2, -0.2, -1.0]$.

In order to compare the recovered shapes, they must be aligned. This is done by transforming the height field using the recovered rotation and translations (figure 6.7) for the first frame. We can then subtract the recovered height field from our generated one to obtain a *difference surface*. Table 6.5 lists the mean and variance of this surface's height and texture values, which we use to determine the reconstruction error.

If the two surface are identical, we expect the *difference surface* to be flat. As seen by its variance, the surface is indeed quite flat. The mean, however, indicates that the reconstructed version has been translated on the z-axis (its origin has moved). This is

	Mean Error	Variance
Heights	-113.0754	0.4965
Red	-0.0269	0.0026
Green	-0.0237	0.0024
Blue	-0.0155	0.0023

Figure 6.5: Mean and variance of the *difference surface*

in concordance with the wave-like behavior of the recovered translations in figure 6.7. Once the height field is rotated, it must be translated back to compensate for the large movement. If the reconstructed height field did not have this translational bias on the z-axis, we would expect the recovered translations to be similar to the generated ones.

It is important to note that this discrepancy is not problematic because an ambiguity will always exist when reconstructing 3D information from 2D. By moving the shape's origin, there are an infinite number of combinations of rotations and translations that will yield a correct imaged model.

Figure 6.6 provides visual proof that our method works. The recovered shape is nearly identical to the generated one, and the texture map is a faithful duplication of the original. There are a few minor artifacts, but the overall results are very convincing.

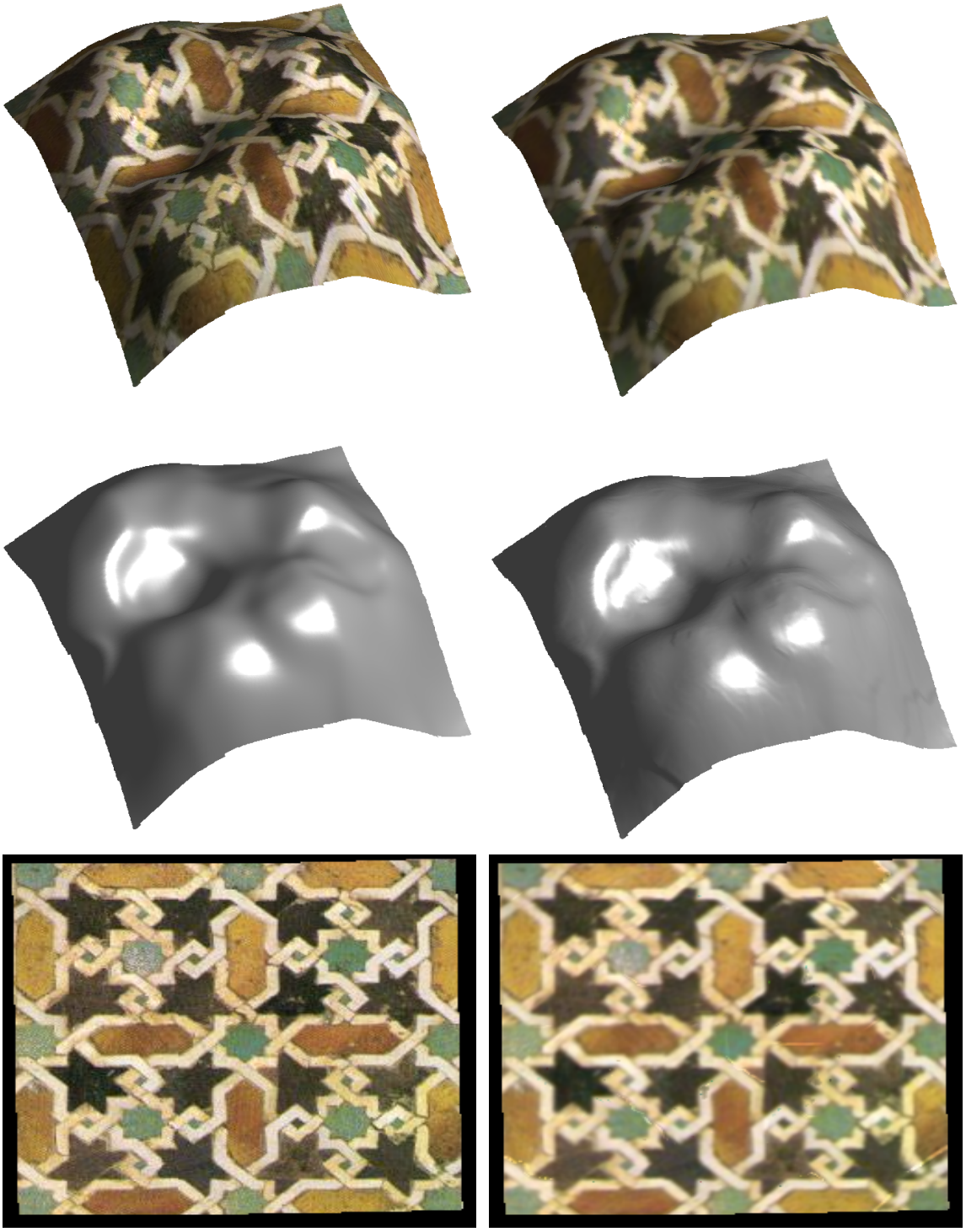

Figure 6.6: Comparison of generated height field and texture map (left) with the recovered geometry and texture map (right). A specular component was added to the untextured surfaces (middle) to help the visualization.

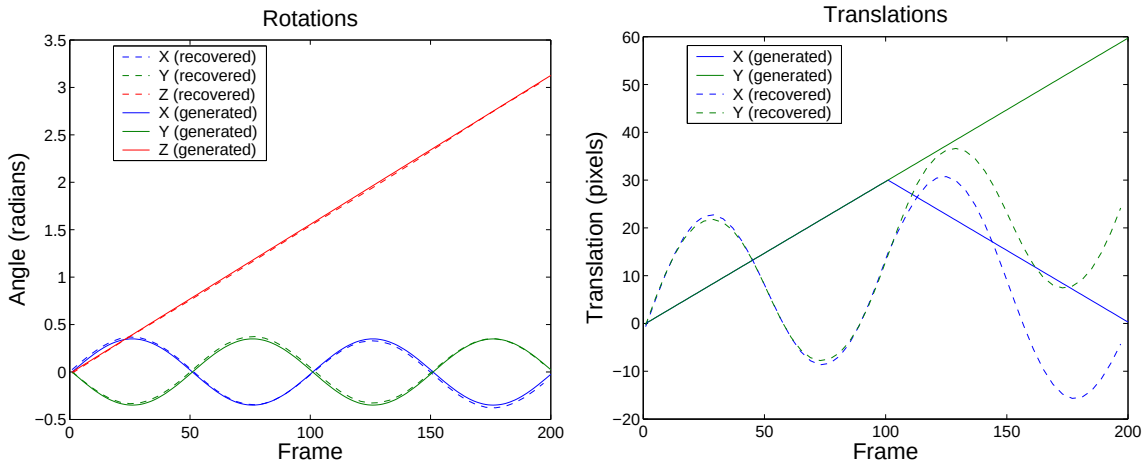

Figure 6.7: Comparison of the generated height field's rotations (left) and translations (right)

6.2 "Lady" figurine

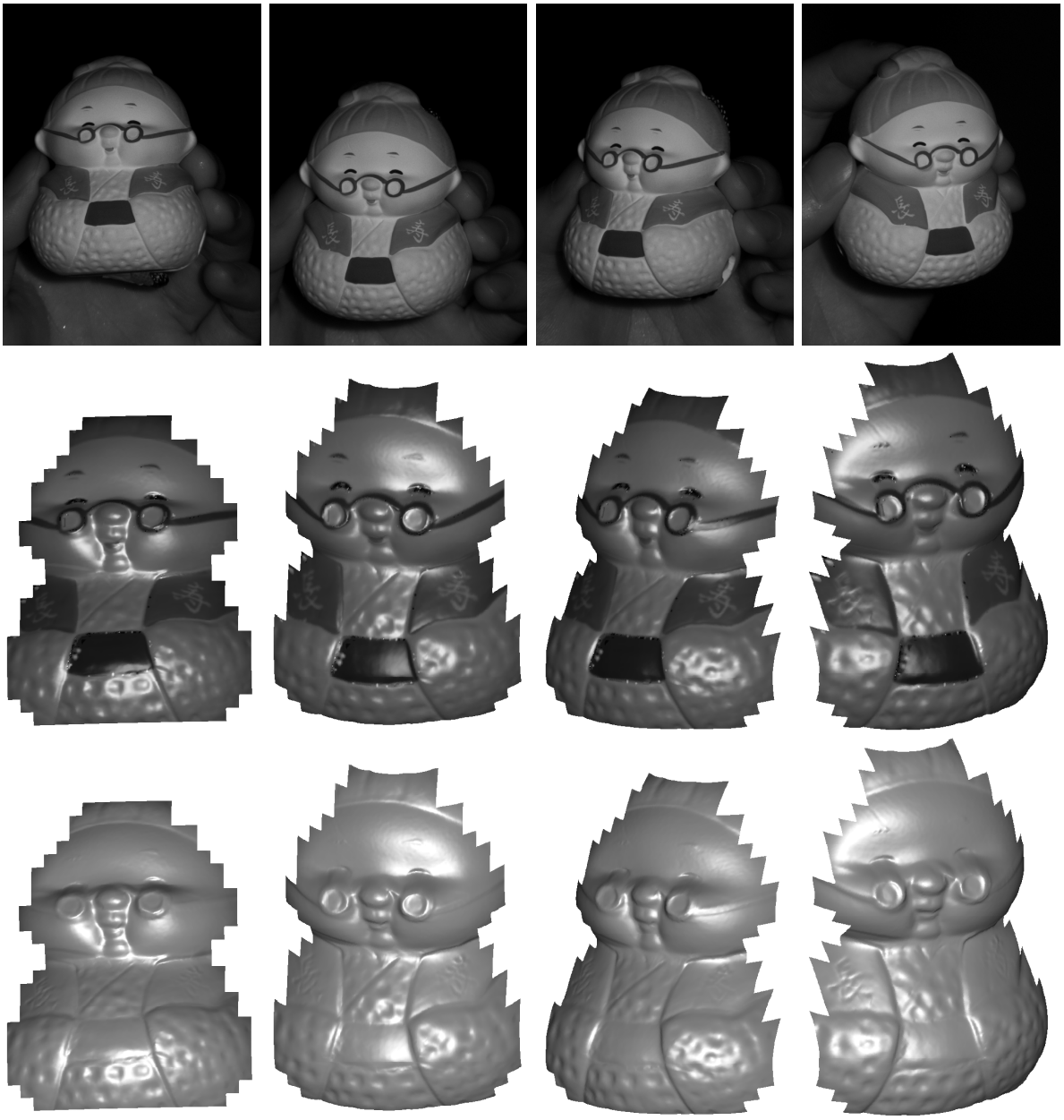

Figure 6.8: 4 frames from the 400 frame "Lady" sequence (top), and the textured (middle) and untextured (bottom) reconstructions. Original video footage courtesy of Zhang et al. [43].

For comparison with recent techniques, we ran our system on the same image sequence as used in Zhang et al. [43]. Figure 6.8 (top) shows 4 frames from this sequence. The

middle and bottom rows of figure 6.8 are the reconstructions of those frames using the recovered shape and transformations.

Resolution	Number of Frames
32×32	400
64×64	50
128×128	25
256×256	25
512×512	13

Figure 6.9: Optimization Schedule for "Lady"

The entire optimization took 6 days on a 1.6GHz/1G RAM laptop. Table 6.10 shows the recovered parameters for this input sequence.

σ^2_{image}	1.9786×10^{-5}
σ^2_{smooth}	0.0030
τ	0.9617
light	[0.06 0.01 -1.00]
	ambient shown in figure 6.13

Figure 6.10: Recovered parameters for "Lady" sequence

From table 6.10 we can see that the image-matching variance σ^2_{image} is very low, indicating a close match between input sequence and imaged model.

Novel views of the reconstructed shape are shown in figure 6.11. Fine detail (spherically-inward bumps) on the figurine's belly has clearly been reproduced. For completeness, we show a side-view comparison of the reconstructed figurine alongside a photograph taken from approximately the same angle (figure 6.12). The recovered rotations, translations, scaling and ambient lighting are also shown, in figure 6.13.

Figure 6.11: The reconstructed “Lady” figurine

Figure 6.12: Side-view comparison of the real figurine (left) and our recovery (right)

6.2.1 Comparison with other work

Figure 6.14 shows side-view reconstructions with those of Zhang et al. [43]. Our results are more detailed, especially in the regions around the belly. The spherically-inward bumps have more definition in our reconstruction.

The Phong-shaded front-view reconstruction, alongside the results from Zhang et al. [43] and concurrent work to ours of Lim et al. [22], can be seen in figure 6.15. Compared to Zhang et al. [43], our reconstruction contains much more detail, especially around the nose and mouth areas.

In concurrent work to ours, Lim et al. [22] have reproduced the figuring with impressive results. Their reconstruction does contain a few artifacts, however, notably on the belly where the left frontal side is indented the wrong way. This is not present in our reconstruction. Furthermore our reconstruction has better definition of the spherically-inwards bumps on the belly.

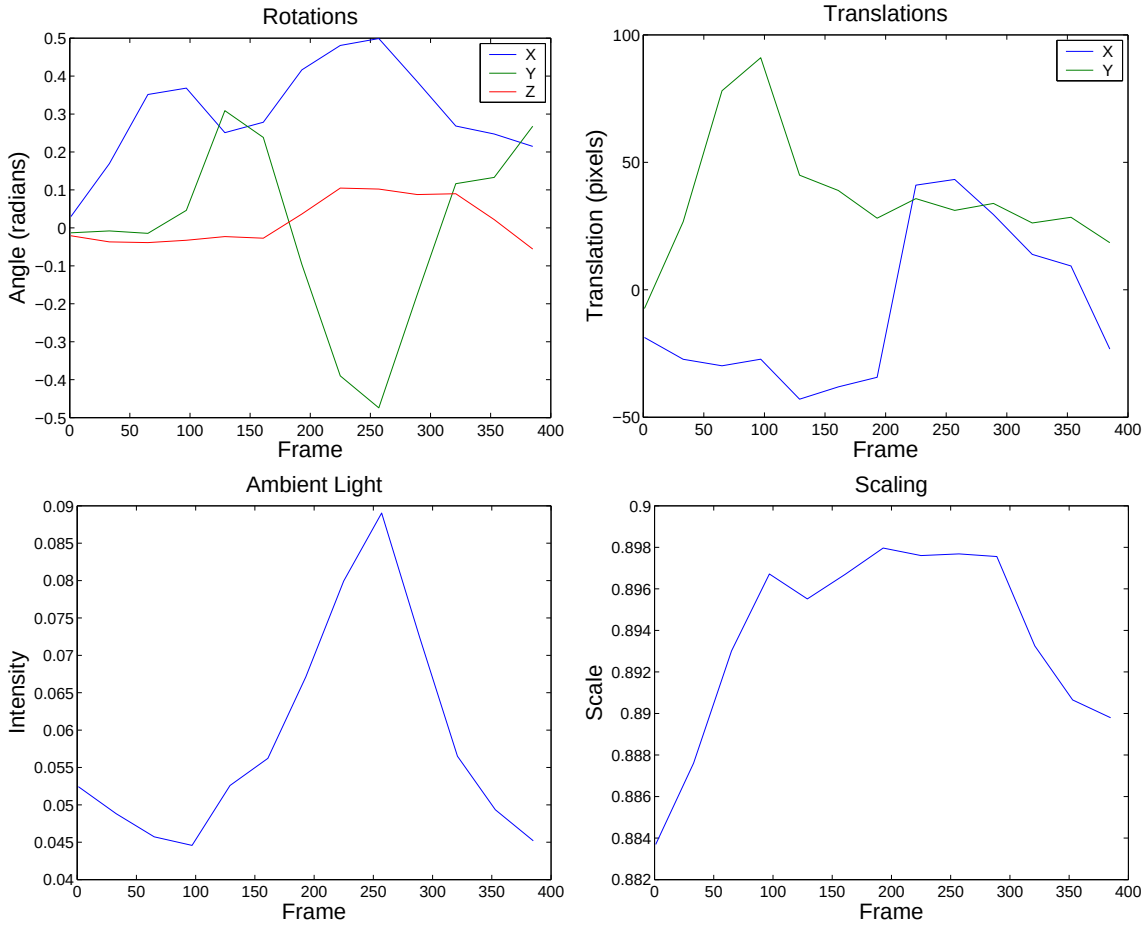

Figure 6.13: "Lady" rotations, translations, ambient light and scaling

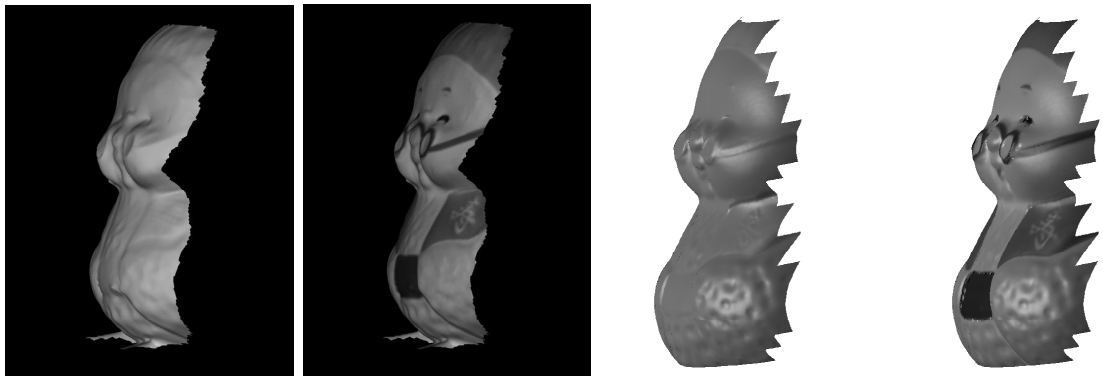

Figure 6.14: Comparison between our reconstruction (bottom) and the reconstructed "Lady" in Zhang et al. [43] (top)

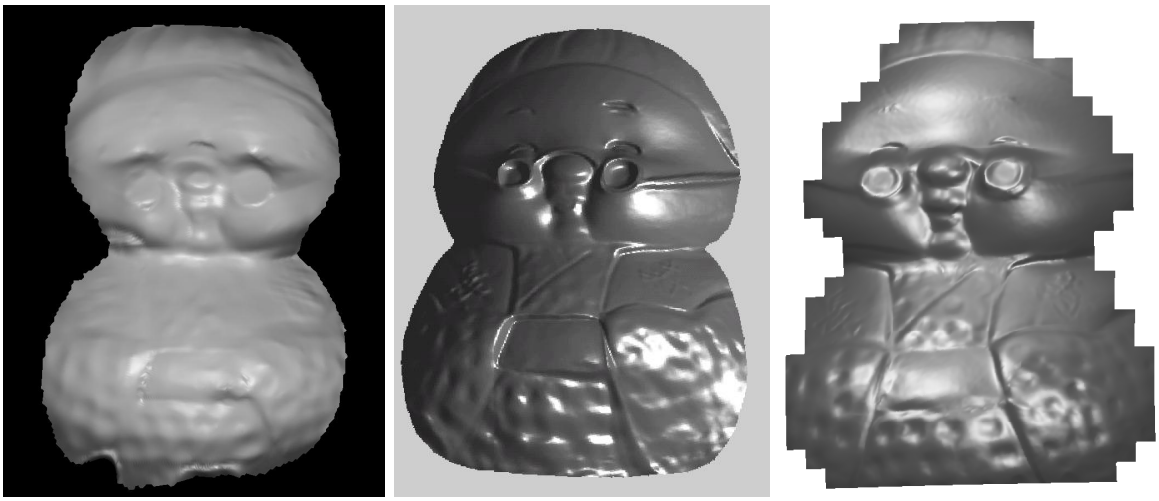

Figure 6.15: Comparison of reconstructions in Zhang et al. [43] (left), Lim et al. [22] (middle) and ours (right)

6.3 Hatake Kakashi figurine

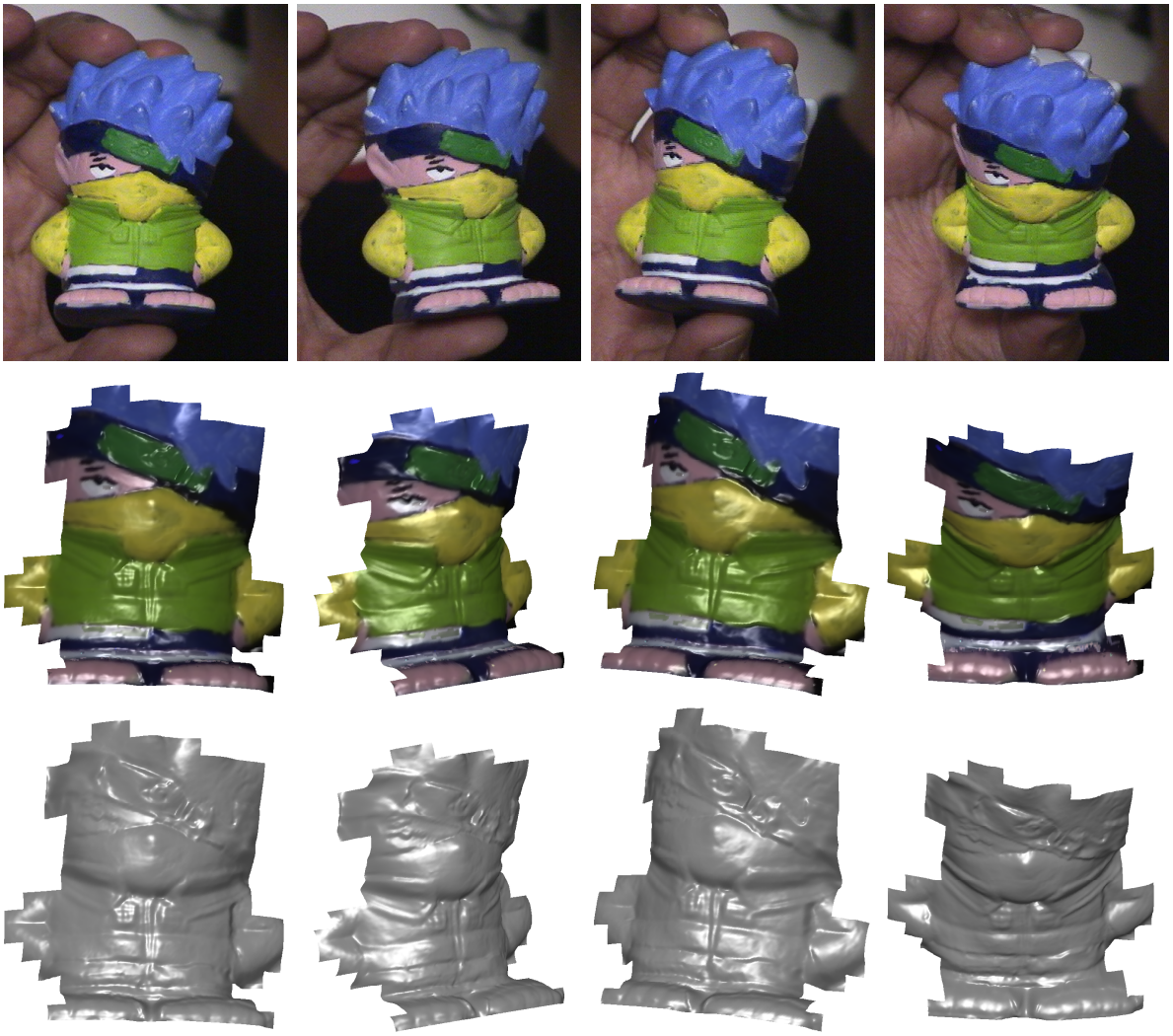

Figure 6.16: 4 frames from a 234 frame image sequence of Hatake Kakashi figurine (top), and the textured (middle) and untextured (bottom) reconstructions.

We show results on a matte-painted figurine exhibiting very fine detail on the torso (figure 6.16, top). The middle and bottom rows of figure 6.16 are the reconstructions of those frames using the recovered shape and transformations. Below is the optimization schedule breakdown (table 6.17).

The entire optimizations took 8 days on 1.6Ghz/1G laptop. Recovered parameters are shown in table 6.18.

Resolution	Number of Frames
32×32	234
64×64	117
128×128	59
256×256	59
512×512	30

Figure 6.17: Optimization schedule for Hatake Kakashi figurine

σ^2_{image}	6.4316×10^{-4}
σ^2_{smooth}	0.0228
τ	0.9565
light	[-0.05 -0.32 -1.00]
	ambient light 0.4140

Figure 6.18: Recovered parameters for Hatake Kakashi figurine

The image-matching variance was slightly lower than the previous examples. This is due to the fact that input image sequence is quite noisy. Severe interlacing effects can be seen in many frames.

Figure 6.19 shows novel views of the geometric reconstruction of the object. As can be seen, the torso contains high detail matching the input sequence, notably the lapelles and the squarish buttons just below the collar.

The headband has been incorrectly reproduced, however. This is due to the fact that it was quite specular. Although painted with a green matte paint, the headband was initially silver in color and the paint was not thick enough to completely cover it. The recovered transformations are also shown in figure 6.20.

Figure 6.19: Reconstruction of the Hatake Kakashi figurine

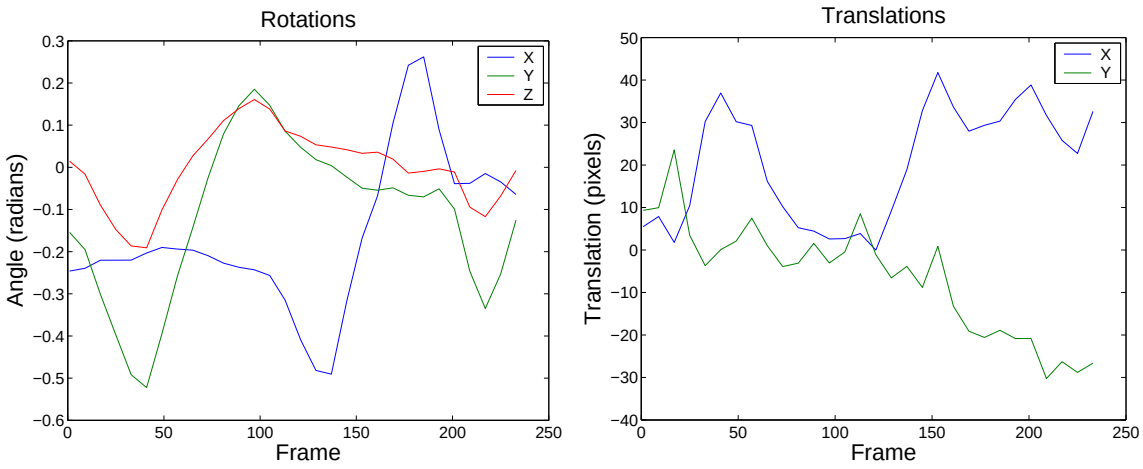

Figure 6.20: Recovered rotations and translations of Hatake Kakashi figurine

6.4 Uzumaki Naruto figurine

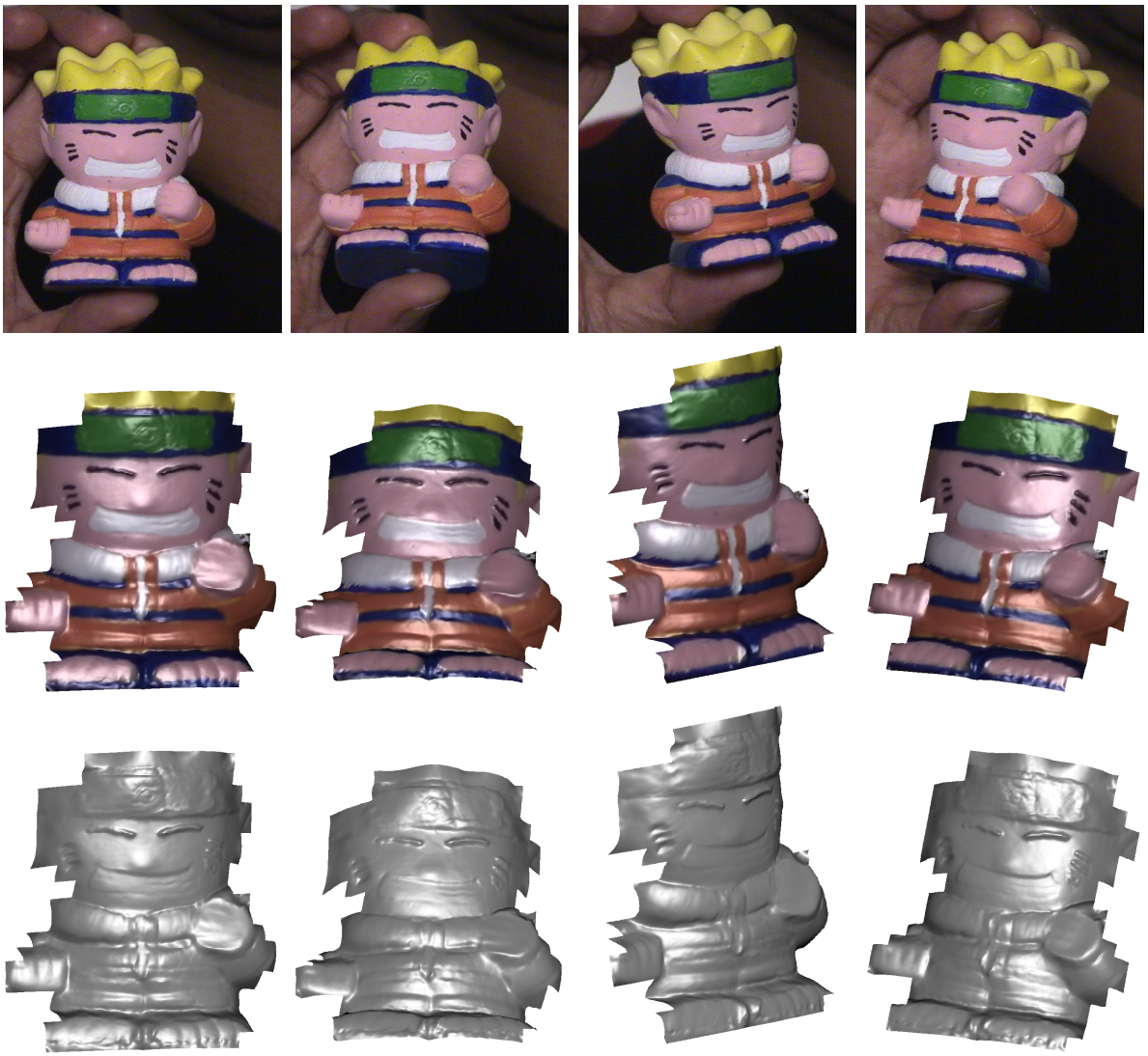

Figure 6.21: 4 frames of a 237 frame image sequence of the Uzumaki Naruto figurine (top), and the textured (middle) and untextured (bottom) reconstructions.

We now show results for a matte-painted figurine of Uzumaki Naruto (figure 6.21, top). The middle and bottom rows of figure 6.21 are the reconstructions of those frames using the recovered shape and transformations. The optimization schedule is shown in table 6.22.

The entire optimization took 4 days on 3.0Ghz/2G machine. The recovered parame-

Resolution	Number of Frames
32×32	237
64×64	119
128×128	60
256×256	60
512×512	30

Figure 6.22: Optimization Schedule for Uzumaki Naruto figurine

ters are shown in table 6.23.

σ^2_{image}	7.0943×10^{-4}
σ^2_{smooth}	0.0094
τ	0.9378
light	[0.04 -0.13 -1.00]
	ambient 1.0205×10^{-4}

Figure 6.23: Recovered parameters for Uzumaki Naruto figurine

As in the previous example, the image variance σ^2_{image} is lower than both the computer generated sequence and the "Lady" sequence. This is normal, as this sequence was also captured with the same camera and digitized using the same hardware. The noise level is about the same as in the previous sequence.

In figure 6.24 we show novel views of the the recovered shape of the figurine. Once again the detail has been successfully captured by our model. The toes and fingers are evident, as are the indentations of the mouth and nose. The recovered rotations and translations are shown in figure 6.25.

There are also a few artifacts. Since the texture on the face is black, there is an inherent ambiguity between texture and shape. Blackness could be either the result of texture or shape variation, and the algorithm has no way of distinguishing between the

two. Extra priors or user guidance are needed to disambiguate.

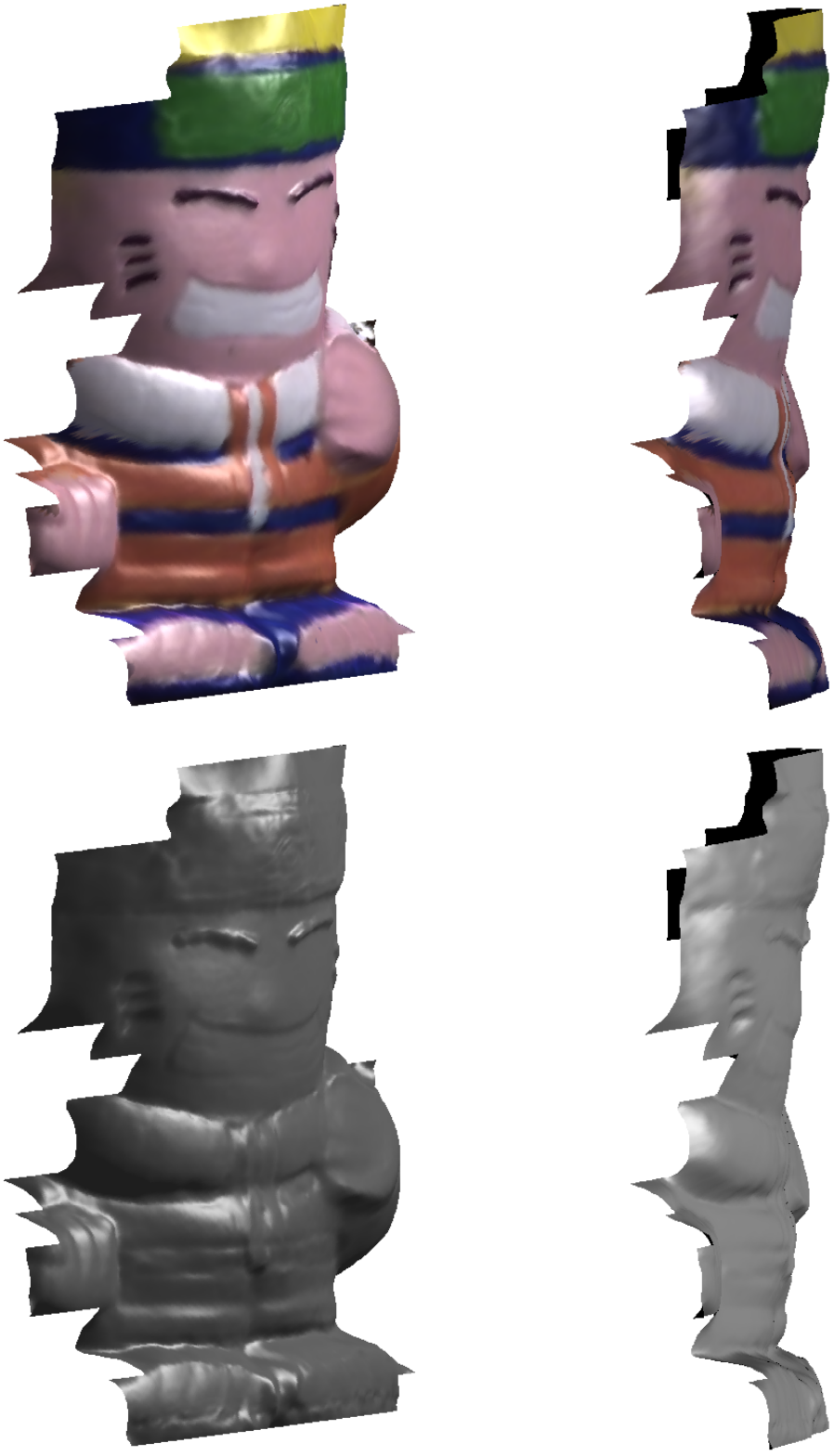

Figure 6.24: Recovered Uzumaki Naruto figurine

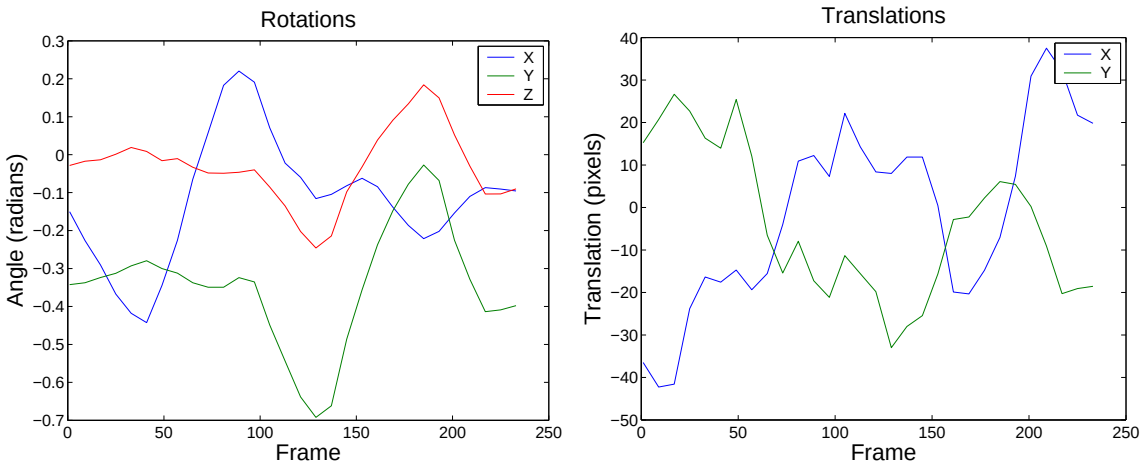

Figure 6.25: Recovered rotations and translations for Uzumaki Naruto figurine

6.5 Occluded Uzumaki Naruto figurine

In this sequence, we occlude 30% of the frames and test our robust mixture model against a model without a mixture component for outliers.

The first section shows results for our robust model and the second section shows results for the model without outlier support. The optimization schedule is identical for both tests, and is given below.

Resolution	Number of Frames
32 × 32	203
64 × 64	102
128 × 128	51
256 × 256	51
512 × 512	26

Figure 6.26: Optimizations Schedule for Occluded Uzumaki Naruto figurine

6.5.1 With mixture components

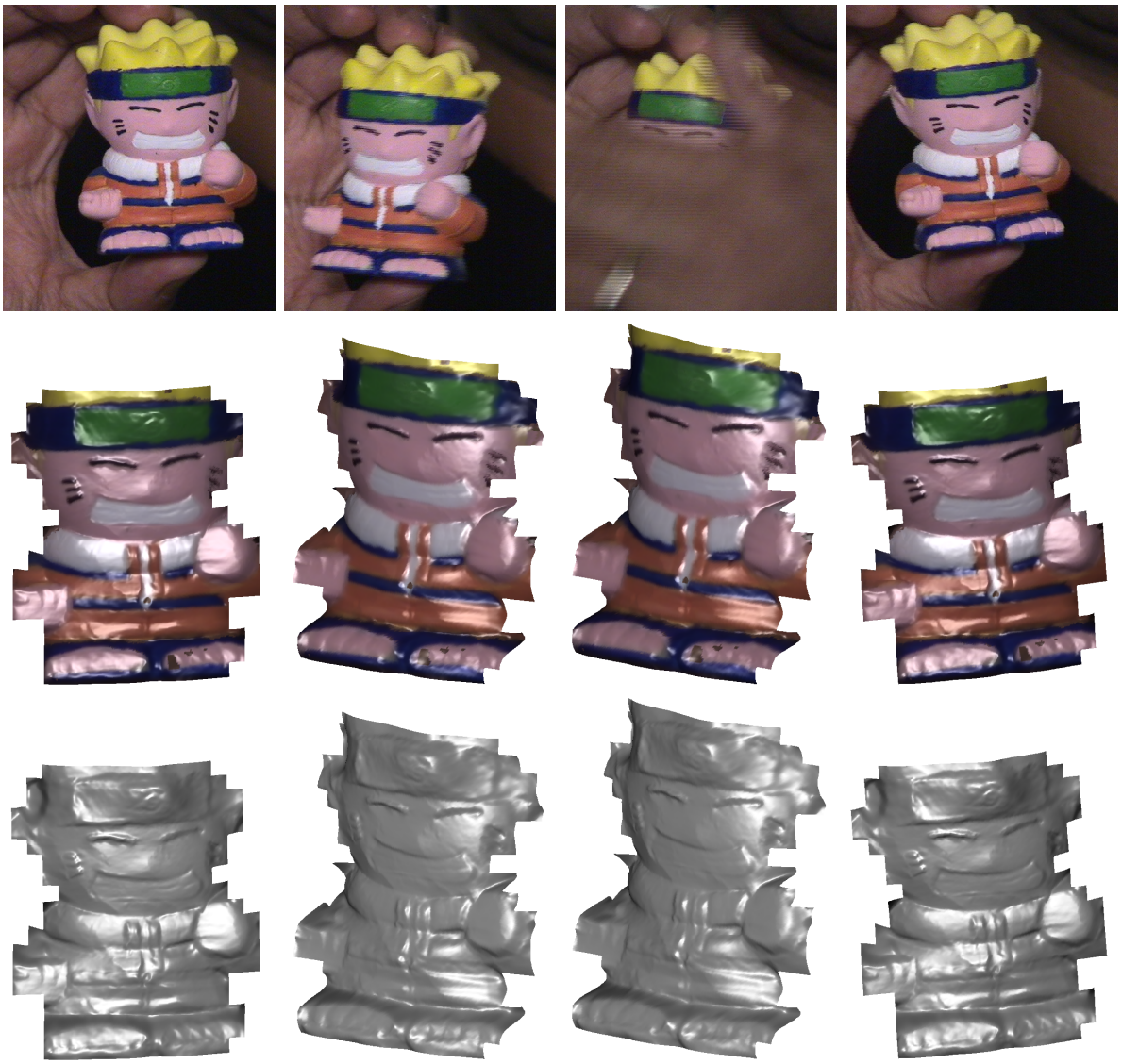

Figure 6.27: 4 frames of the 203 frame image sequence of the occluded Uzumaki Naruto figurine(top), and the textured (middle) and untextured (bottom) reconstructions **using the mixture model**.

The entire optimization took 4 days on 3.0Ghz/2G machine. Below are the recovered parameters (table 6.28).

The variance σ^2_{image} is in the same order as the variance for the previous unoccluded Naruto sequence. Notice the value of τ is significantly smaller than all previous unoc-

σ^2_{image}	8.6109×10^{-4}
σ^2_{smooth}	0.0074
τ	0.7499
light	[-0.22 -0.13 -1.00]
	ambient 1.0198×10^{-4}

Figure 6.28: Recovered parameters for Occluded Uzumaki Naruto figurine using the mixture model

cluded tests. This value tells us that that the model has determined that 25% of the pixels are outliers.

Novel views of the recovered are shown in figure 6.29. The recovered rotations and translations are shown in figure 6.30.

Although a little noisier than the unoccluded Naruto results shown previously, we can see that the shape has nonetheless been faithfully reproduced.

It is important to note that if the optimization schedule was modified to consider more frames per coarse-to-fine level, perhaps the reconstruction would be more exact. It is also worth mentioning again that we prematurely stop each level after 3000 iterations, and it is entirely possible that more iterations may be needed in occluded circumstances.

Figure 6.29: Recovered shape for Occluded Uzumaki Naruto figurine using the mixture model

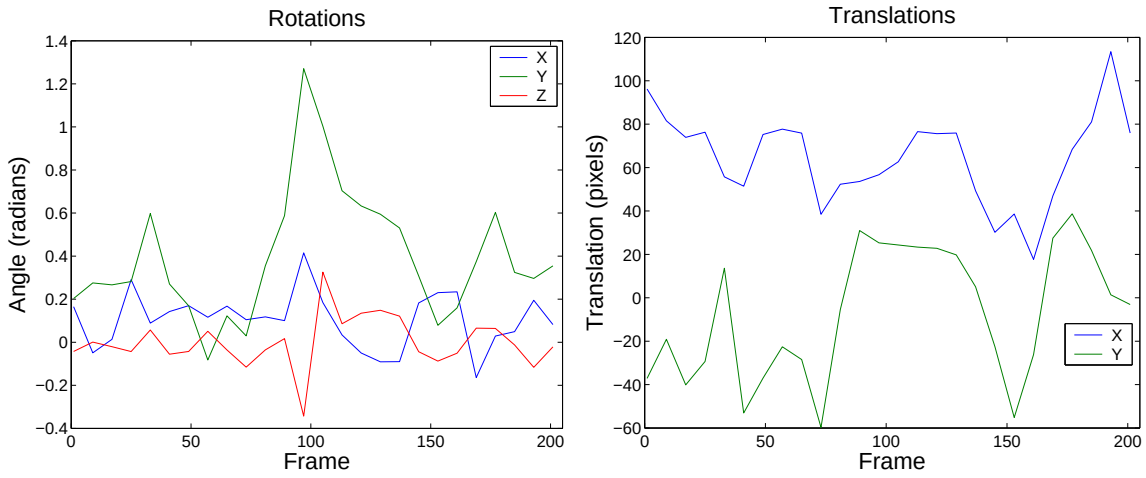

Figure 6.30: Recovered rotations and translation for Occluded Uzumaki Naruto figurine using the mixture model

6.5.2 Without mixture components

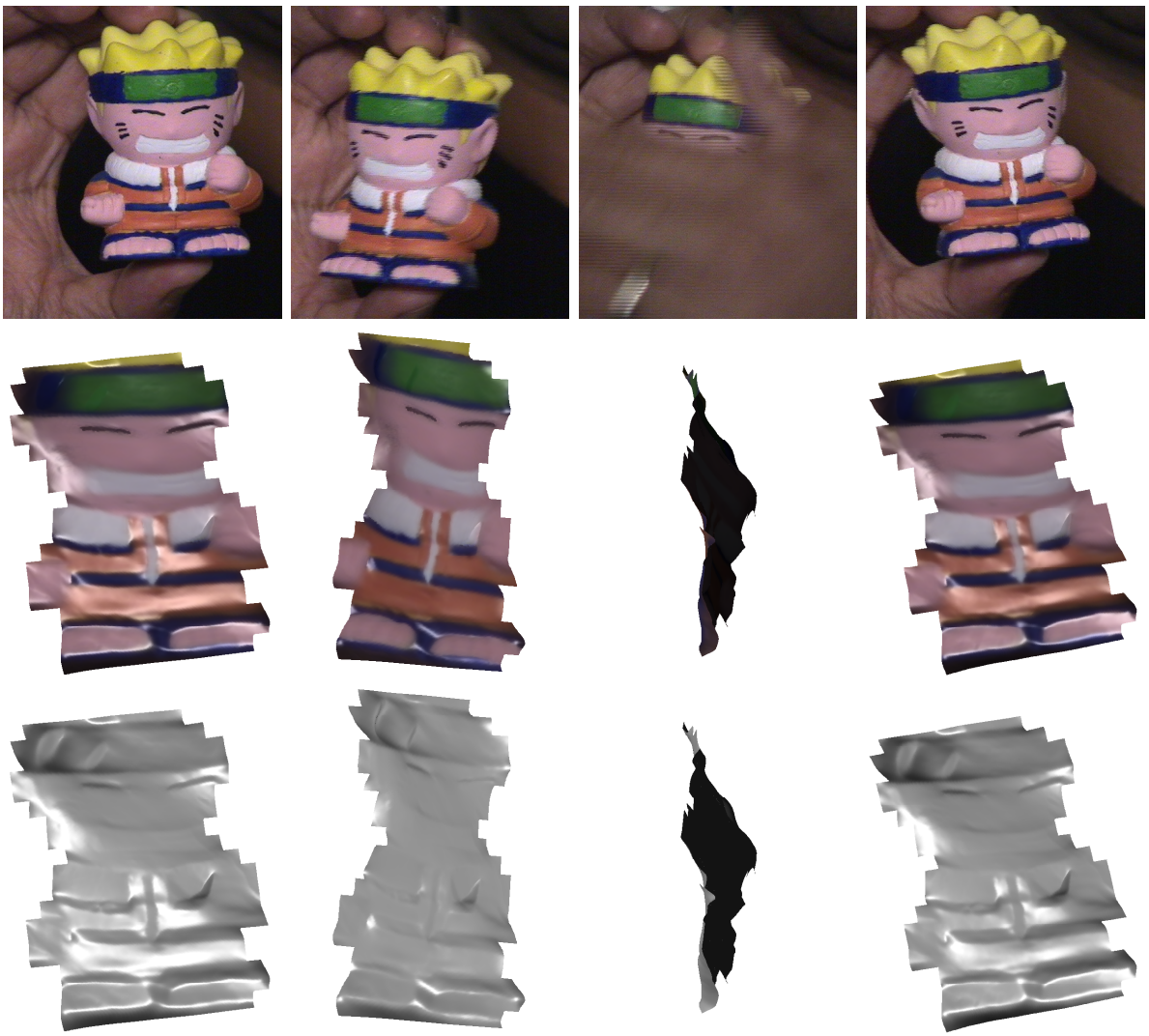

Figure 6.31: The same 4 frames of the 203 frame image sequence of the occluded Uzumaki Naruto figurine(top), and the textured (middle) and untextured (bottom) reconstructions **without using the mixture model**.

We now examine the recovered shape and motion using a simple Gaussian instead of our mixture model. These experiments were conducted by forcing τ to 1.0 during the entire optimization. The recovered parameters are shown below (table 6.32). The entire optimization took 2 days on 3.0Ghz/1G machine.

The value of σ^2_{image} is much larger than it was with the mixture model. This makes

σ^2_{image}	0.0038
σ^2_{smooth}	0.0022
τ	1.0 (forced)
light	[-0.22 -0.13 -1.00]
	ambient 1.0198×10^{-4}

Figure 6.32: Recovered parameters of Occluded Uzumaki Naruto figurine **without** mixture model

perfect sense since the Gaussian must now explain **all** pixels, outliers or not. In order to do so, it must have a large variance.

This large variance means that the generative model is producing images that do properly not match the input sequence. We end up with a mediocre shape reconstruction as a result of this (figure 6.33). The recovered rotations and translations are shown in figure 6.34.

Although the colors are generally correct, the shape is flat and contains no surface detail. Upon examining figure 6.35, the reason for this becomes evident. When the object becomes occluded, the model "matches" the occluded frame by rotating the object out of sight in order to shade it as much as possible to simulate the hand's dark texture. Notice how the mixture model remains robust and relies on the headband to preserve orientation.

Figure 6.33: Recovered shape for Occluded Uzumaki Naruto figurine **without** mixture model

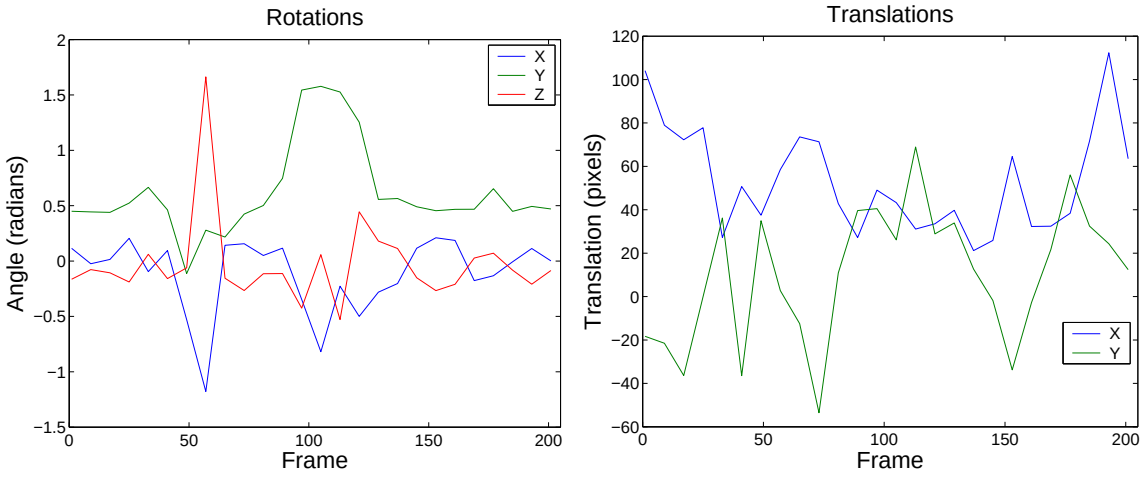

Figure 6.34: Recovered rotations and translations for Occluded Uzumaki Naruto figurine without mixture model

Figure 6.35: Comparison of reconstruction for frame 113 (left) where the mixture model is robust to outliers (middle), and the absence of mixture model leads to reconstruction problems (right)

Chapter 7

Discussion and Future Directions

We have presented a direct, robust method for recovering dense 3D shape, texture, motion and lighting from monocular image streams. We have demonstrated that our method works, even in the presence of occlusion.

7.1 Improvements

The computation time needed for recovery is quite lengthy and this poses a real bottleneck in the system. All examples took around a week of continuous calculations, and even then the optimization never reaches convergence. If we wish to reconstruct long, high resolution video sequences, it is imperative to find ways to speedup the process. The most logical decision would be to clusterize the algorithm. This is entirely possible, since the system calculates gradients by operating on one frame at a time. We could envisage distributing this task so that machines operate on different frames concurrently, and then sum all gradient calculations together before passing them to BFGS.

At present, the user manually tracks sparse features across the input sequence in order to initialize the system. If we wish to have a truly automatic system, it will be necessary to automate the feature tracking. This leads to two options: feature tracking in varying illumination [18, 25], or a custom method that tracks mini-surfels modeled as

flat patches with one normal. This technique could use illumination information to also derive normal orientations. This way we would have tracked features as well as a sparse set of normals which would greatly improve initialization.

7.2 Extensions

A logical future step would be to incorporate deformable models into the framework. This can be achieved by representing shape as a linear combination of n basis shapes and placing temporal priors on shape deformations.

The constraint on Lambertian objects could also be relaxed. By considering other reflectance models [9, 40], we could reconstruct more "real world" objects, since most objects are never purely Lambertian. We could simply replace the Lambertian model with another model.

7.3 Validation

There are important comparisons that need to be made with current feature-based visual modeling systems in order to determine whether shading plays a crucial role in detailed shape recovery. It is unclear, for instance, whether the system proposed by Pollefeys et al. [27] can obtain "close-up" detail the way photometric methods can, since most of their results are from sequences that are taken from vantage points with little illumination changes. A comparison of our and their techniques is needed on objects with fine detail in order to fully justify the importance of shading.

Bibliography

[1] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[2] R. Basri and D. Jacobs. Photometric stereo with general, unknown lighting. 2:374–381, 2001.

[3] Ronen Basri and David W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):218–233, 2003.

[4] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. The bas-relief ambiguity. *Int. J. Comput. Vision*, 35(1):33–44, 1999.

[5] Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, UK, 1996.

[6] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. pages 187–194, 1999.

[7] Matthew Brand. Morphable 3d models from video. In *CVPR (2)*, pages 456–463, 2001.

[8] Michael J. Brooks. *Shape from shading*. MIT Press, Cambridge, MA, USA, 1989.

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

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

[11] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics (2nd ed. in C): principles and practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[12] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):439–451, 1988.

[13] Steven J. Gortler and Michael F. Cohen. Hierarchical and variational geometric modeling with wavelets. In *Symposium on Interactive 3D Graphics*, pages 35–42, 205, 1995.

[14] Berthold K. P. Horn. Height and gradient from shading. *Int. J. Comput. Vision*, 5(1):37–75, 1990.

[15] Katsushi Ikeuchi and Berthold K. P. Horn. Numerical shape from shading and occluding boundaries. *Artif. Intell.*, 17(1-3):141–184, 1981.

[16] Michal Irani. Multi-frame optical flow estimation using subspace constraints. In *ICCV*, pages 626–633, 1999.

[17] Michal Irani and P. Anandan. About direct methods. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 267–277, London, UK, 2000. Springer-Verlag.

[18] Hailin Jin, Paolo Favaro, and Stefano Soatto. Real-time feature tracking and outlier rejection with changes in illumination. In *ICCV*, pages 684–689, 2001.

[19] Michael J. Jones and Tomaso Poggio. Multidimensional morphable models. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 683, Washington, DC, USA, 1998. IEEE Computer Society.

[20] Erwin Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[21] K. M. Lee and C. C. J. Kuo. Shape from shading with a linear triangular element surface model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(8):815–822, 1993.

[22] Jongwoo Lim, Jeffrey Ho, Ming-Hsuan Yang, and David Kriegman. Passive photometric stereo from motion. In *ICCV '05: Proceedings of the Ninth IEEE International Conference on Computer Vision*.

[23] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[24] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[25] Shahriar Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(9):961–979, 1998.

[26] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[27] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3):207–232, 2004.

[28] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 2nd edition, 1992.

[29] R. Ramamoorthi and P. Hanrahan. On the relationship between radiance and irradiance: determining the illumination from images of a convex lambertian object. *J. Optical Soc. of Am. A*, 18(10):2448–2458, 2001.

[30] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[31] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, 15(3):76–84, 1995.

[32] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(6):513–528, 1990.

[33] R. Szeliski and S. B. Kang. Recovering 3d shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.

[34] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Comput. Vision*, 9(2):137–154, 1992.

[35] Lorenzo Torresani and Aaron Hertzmann. Automatic non-rigid 3d modeling from video. In *ECCV (2)*, pages 299–312, 2004.

[36] Lorenzo Torresani, Danny B. Yang, Eugene J. Alexander, and Christoph Bregler. Tracking and modeling non-rigid objects with rank constraints. In *CVPR (1)*, pages 493–500, 2001.

[37] PingSing Tsai and Mubarak Shah. Shape from shading using linear approximation. *Image and Vision Computing Journal*, 12(8):487–498, 1994.

[38] T.Poggio V. Blanz, C. Basso and T. Vetter. Reanimating faces in images and video. pages 641–650, 2003.

[39] P. Viola and III W. M. Wells. Alignment by maximization of mutual information. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 16, Washington, DC, USA, 1995. IEEE Computer Society.

[40] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, New York, NY, USA, 1992. ACM Press.

[41] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. pages 513–531, 1989.

[42] Gang Zeng, Yasuyuki Matsushita, Long Quan, and Heung-Yeung Shum. Interactive shape from shading. In *CVPR (1)*, pages 343–350, 2005.

[43] Li Zhang, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multi-view stereo. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 618, Washington, DC, USA, 2003. IEEE Computer Society.

[44] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[45] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, 1997.