Capturing Cloth

by Ryan McKenzie White

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

_____

Professor David Forsyth
Research Advisor

_____

(Date)

* * * * * * *

_____

Professor James O'Brien
Second Reader

_____

(Date)

Abstract

Capturing Cloth

by

Ryan McKenzie White

Master of Science in Computer Science

University of California at Berkeley

Professor David Forsyth, Research Advisor

We present a method for capturing the geometry and parameterization of fast-moving cloth using multiple video cameras, without requiring camera calibration. Our cloth is printed with a multi-scale pattern that allows capture at both high speed and high spatial resolution even though self-occlusion might block any individual camera from seeing the majority of the cloth. We show how to incorporate knowledge of this pattern into conventional structure from motion approaches, and use a novel scheme for camera calibration using the pattern, derived from the shape from texture literature. By combining strain minimization with the point reconstruction we produce visually appealing cloth sequences. We demonstrate our algorithm by capturing, retexturing and displaying several sequences of fast moving cloth.

Contents

Chapter 1

Introduction

The premise of this work is that cloth simulation does not cover all cloth modelling needs. Instead of using physically based techniques to model cloth, we aim to model cloth by capturing it in the real world. Pure simulation of cloth is a difficult problem — solutions are subject to a number of stiffness problems. Complicated environmental interactions, including wind and collision are time consuming to integrate. Simulation stability is often problematic. A reasonable summary of the state of the art is contained in [5].

We pose the problem of capturing cloth in the same context as human motion capture: using multiple video cameras, we aim to capture changing geometry as a function of time. However, our problem differs with human motion capture — most notably in the number of degrees of freedom. Human motion is fundamentally tied to the skeleton — a collection of a small number of rigid bodies. Each rigid body has a small number of degrees of freedom, meaning that a relatively small number of motion capture markers can resolve all degrees of freedom. By contrast, cloth exhibits a much larger number of degrees of freedom — effectively limited only by the display resolution.

We capture the motion of cloth by printing a **custom pattern** composed of triangles at multiple scales. By using information at multiple scales, the points on the cloth can be localized with high accuracy while allowing fast processing. Accurate localization is important because the point

Figure 1: *In a typical cloth simulation environment, combining a fluid simulator with a cloth modeler presents many challenges. However, complicated aerodynamic effects are easy to produce by cloth capture.*

locations are the input for both the calibration and the subsequent reconstruction. Camera calibration is automatic and based on a shape from texture method that uses both points and normals. We use **bundle adjustment** to refine the camera parameters and 3D locations. We take the reconstructed points and use **strain reduction** to produce a higher quality reconstructed mesh.

Chapter 2

Previous Work

2.1 Cloth Motion Capture

Our work is not the first in this field — in our review of previous work, we identify what we consider to be the key concepts in cloth motion capture.

The idea of using a custom pattern on a cloth to drive reconstruction dates to Guskov et al [3]. The authors print a black and white checkerboard pattern on the cloth, using the vertices as reconstruction points. This method has the advantage that the markers are easy to localize but difficult to differentiate. Locally, each marker is indistinguishable — meaning that correspondence relies on global reasoning — and subsequently, that occlusion can not be dealt with properly. In an extension to the same work [2], the pattern is modified to use different colors to make each checkerboard marker distinctive. This work has the advantage of allowing real-time capture with occlusions. Requiring the system to run in real-time limits its applicability — time consuming steps such as bundle adjustment and strain minimization are not possible. The resulting mesh appears heavily smoothed.

Working in a different direction, Pritchard and Heidrich [9, 8] use a non-repeating line pattern and a stereo vision camera to obtain both shape and parameterization. As far as we know, this work is the first to focus on the importance of a parameterization. While their pattern is made up

of distinctive elements, it is not robust to motion blur and the elements are easily confused. While Guskov et al recover roughly 100 points, Pritchard et al recover several thousand. However, some of these points are erroneous, and significant work is done to remove confusion.

Scholz et al [11] used optical flow of a highly texture cloth to obtain a mesh over time. The mesh is dense and their tracker is fairly consistent over time. When the tracker failed, they use a silhouette matching procedure. Results are demonstrated on a synthetic image sequence — without fast movement or significant self occlusion.

Our work combines the salient elements of each procedure. Following Guskov, we use different colored markers to distinguish different parts of the cloth. Following Pritchard, we record a parameterization of the cloth surface. Wanting both large numbers of features (Pritchard) and unambiguous reconstruction (Guskov), we use elements that contain information at multiple scales.

2.2 Structure from Motion

Our approach to cloth motion capture relies fundamentally on a structure from motion argument. By viewing the cloth at each point in time from multiple cameras, we can reconstruct the geometry. The structure from motion community has a long history, and a good book is available on the subject [4].

Typical structure from motion relies on estimating camera parameters and 3D locations from multiple views of a rigid object. Typically, the object is either still, and multiple images are taken from different viewpoints, or the object is rigid and moving and images from different points in time are used. Because our cloth is not rigid, we must use multiple cameras to reconstruct the cloth at any given point in time. Previous research tends to focus on two extremes — reconstruction from only 2 views or reconstruction from many more (often in the tens or hundreds). We cannot afford to have tens or hundreds of cameras running simultaneously, yet require the difference in viewpoint and increase in accuracy derived from more than 2 cameras.

The standard method in the area uses corresponding **interest points** in each view of the

rigid object. Assuming a set of projective cameras, a **projective reconstruction** is obtained by using **epipolar** and **multi-view** constraints. At this point, the cameras and geometry are known up to a single projective transform. Further constraints, in the form of known geometry, are imposed to upgrade the reconstruction to Euclidean. Typically, the known geometry relies on parallel and perpendicular lines at a scale near the scale of the image. Lacking such information about arbitrary scenes, a **calibration object** can be used.

A simpler form of this is available if the cameras are assumed to be orthographic: in two views, point clouds reveal a Euclidean reconstruction up to a single ambiguity. [13, 6] This ambiguity can be resolved with a known angle or length. Our work in calibration extends upon this by using a repeated geometry (often called a texture) to allow the Euclidean upgrade.

In **bundle adjustment**, the camera parameters and 3D locations are simultaneously updated in a single large optimization. Although this process is slow, relying on non-linear optimization techniques, it has been carefully studied and contains significant structure. [12] The benefit to bundle adjustment is the careful refinement of the solution that is difficult to achieve with coordinate descent style techniques. The cost function for bundle adjustment is typically **reprojection error**. Each point in the 3D reconstruction can be projected back onto the image and compared with the feature location. Reprojection error is the average of the distance between the reprojected points and the observed data.

In reconstructing the shape of the cloth, we use contributions from several aspects of the structure from motion community. First, we use shape-from-texture arguments and surface normals to provide a Euclidean upgrade. Second, we use a combined optimization, similar to bundle adjustment, to solve for the 3D locations with a joint cost function over reprojection error and strain.

Chapter 3

Cloth Pattern

We print a pattern on our cloth which is carefully chosen to allow robust observation. Our pattern is a set of large equilateral color coded triangles where the coloring of each triangle identifies the location and orientation on the cloth. Each large triangle consists of a number of small triangles — where the vertices of the small triangles form a fine grid like structure over the cloth (figure 2). First, elements are highly distinctive and there is little repetition over a large area of cloth. If the cloth is moving quickly, some cameras may see only a small fraction of the entire cloth, so that global correspondence reasoning is impractical. A distinctive element allows reconstruction even in this difficult case. For small sheets, we use entirely unique elements, but on larger sheets we distribute unique triangles over the cloth. Second, our pattern offers a high degree of spatial accuracy, while allowing robust observations during dynamic sequences. These two requirements are in tension because motion blur tends to obscure the high frequency information need for accurate localization. Our large triangles are relatively easy to identify despite motion blur; a deformable template approach then yields the interior structure (15 vertices of the smaller triangles), in a form of coarse-to-fine search.

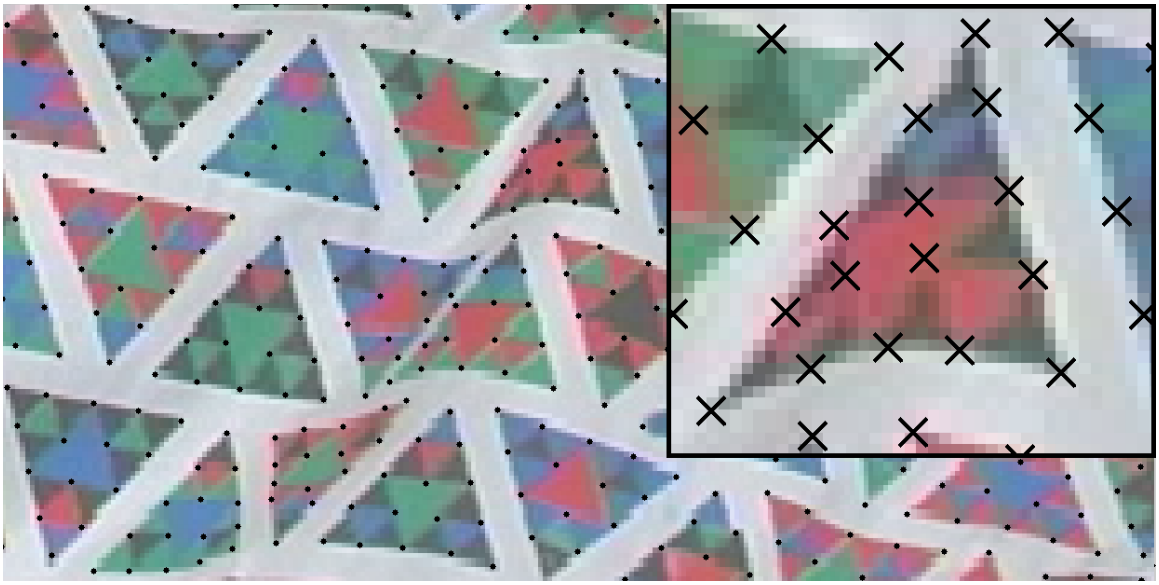

Figure 2: *Our pattern consists of repeating triangles at multiple scales. The large scale triangle has a unique pattern that can easily accommodate 216 unique elements. The vertices of the small scale triangles allow for fine sampling of point locations. The results of our localization code are shown as black dots. Notice how the patterns within the large triangles identify them, and so, through the small triangles, the vertices, leading to a mesh structure — and so a measurement of the cloth parameterization. Accurate localization drives the reconstruction process — important for both bundle adjustment and strain reduction (section 5.2).*

3.1 Course step— finding large triangles and their normals

We assume that, over the scale of an individual triangle, perspective effects are negligible; that over the scale of the whole frame, the effects of perspective are small; and, for the purposes of normal estimation, that the surface curvature at the scale of a triangle is small. Our search runs through several steps: threshold the image at different values, perform rough fit of locations and normals, combine triangles at different thresholds, and finally perform a nonlinear optimization over triangle parameters. In each case, the current step either narrows the search space, or provides a better initialization for the next step.

Given variation in lighting and shadows, different regions have greatly varying intensities. A simple threshold for intensity on grayscale images is not enough to find all of the triangles in these different regions, so we threshold at multiple values. For each thresholded image, we use morphological operations to find all blobs of the appropriate size. Because perspective is negligible at the scale of an element, each triangle is imaged through an affine transformation that is a function of camera scale, the slant and tilt of the plane on which the triangle lies, and the in-plane rotation of the triangle. We obtain a rough estimate of camera scale by assuming some triangles in the sequence will be viewed frontally by this camera, so that the largest triangles offer an estimate of camera scale. We now use the scale estimate to precompute views of each triangle at a set of different slants, tilts and in-plane rotations. The blobs are compared against this precomputed set — using the number of mismatched pixels as a cost for the quality of the fit.

To combine the triangles at different thresholds, we keep the blob in each area of the image that has the lowest cost. Using the precomputed triangle as a start point, we run a continuous optimization over scale, slant, tilt and in-plane rotation (we attempted to optimize over location as well, but found that typical variation was less than a 1/4 pixel and increased convergence time). At the end of this optimization, we have a model of the triangle location and normal without taking into account local curvature.

3.2 Fine step — localizing triangle vertices

For each large triangle, at the fine scale we extract two quantities: sub-triangle colors and sub-triangle vertex locations. Again, we work through several steps of processing, moving from high level information to lower level information, using higher level information to drive the lower level search. Continuing the refinement, we start with a course deforming model for each triangle, then assign colors and finally run a fine deforming mesh. Figure 3 contains an overview of this process.

The course scale deforming model is made up of 4 triangles with 6 unique vertices. Initializing with the planar triangle defined in the previous section, we allow each of the 6 vertices to move freely, penalizing errors with the thresholded image of the triangle taken from the corresponding threshold choice.

Dividing this course triangle mesh into 16 triangles, we have indices for determining the sub-triangle colors. Image color is a surprisingly poor guide to object color, as it is affected by shadows, variations in printing, lighting and camera sensitivities. Because we know the location of the large triangles, we can rectify the color with a simple strategy. We know that the color pixels are distributed uniformly amongst red, green, blue and black. We then allocate pixels to colors using a greedy strategy, rather like round-robin: assign the red most pixel the label red, the green most green, et cetera, then repeat until no pixels are left. Now, to determine colors for each of the sub-triangles, we group sub-triangles by known relationships (for instance, the four sub-triangles in the middle are always the same color), choose a color and work outward. The sub-triangles that are the farthest from the center of the large triangle are the most problematic. Triangles that deviate from known patterns are thrown out as poor matches.

We now localize each point using a deformable model. Starting with the course scale deforming model, we allow all 15 of the vertices of the smaller triangles to deform. In many cases, the colors assigned to each pixel in the previous stage are erroneous because large variations in lighting across the image create large deviations in color. Instead of making a hard assignment, for

the final matching, we warp the color space in a heuristic way to estimate likelihood. On a triangle by triangle basis, we take the raw image values from the original image and warp the color-space to force the black most pixels to be black, red most pixels to be red, et cetera. In the optimization over the deformable model, we charge for deviations from expected colors. We take the final positions of these vertices to be the vertices used for reconstruction.

3.3 Implementation note

In several parts of this section, nonlinear optimization is used to match a model of the triangle to an observation of the triangle. To achieve reasonable results, it is important that the objective function be continuous. These objective functions are defined as a sum over pixel differences. To achieve continuity, the pixel values from the model must change continuously with the locations of the vertices. For pixels at the edge of the triangle, we approximate the area of the pixel covered by the triangle as linear in the distance from the side of the triangle.

Results are shown in figure 2.

3.4 Manual cleanup

In the interest of time, in some of the shorter sequences, we manually deleted a number of erroneous matches between large triangles (less than thirty per sequence). In the sequence of the skirt, this process was automated.

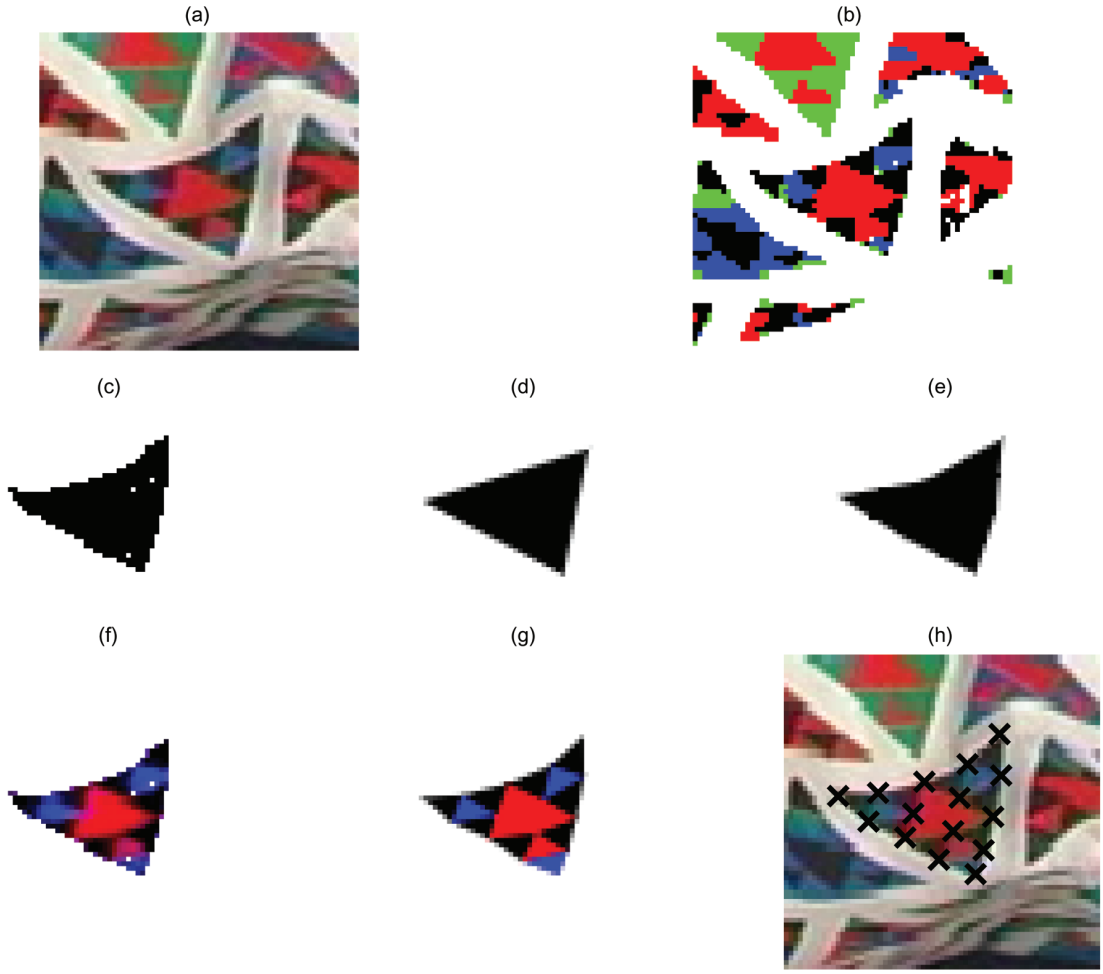

Figure 3: *The triangle matching procedure slowly works from a course model to a fine model, starting with the original image data. Assuming that the appropriate threshold for the image has been picked, we start by cutting out the original image* ***(a)****. Using information from the entire frame, we can assign a color to each pixel using a round robin scheme* ***(b)****. Looking at this figure, one should note that the color assignment problem is the biggest bottleneck — erroneous assignments are common. The process of matching the shape of the triangle to our internal model involves* ***(c)*** *segmenting and thresholding the image,* ***(d)*** *fitting a planar triangle model to the image and* ***(e)*** *using a course deformable model to account for some amount of curvature. Using this deformable model* ***(e)*** *with the color assignment* ***(b)****, we can record the colors of each sub-triangle, and use these colors to warp the original image colorspace* ***(f)****. Finally, we use a fully deformable template to find the vertex positions* ***(g)*** *and mark them on the original image* ***(h)****.*

Chapter 4

Camera Calibration

We calibrate our cameras in two parts: first we assume local orthography and use the normals of the texture elements to estimate pairwise calibration. This work is an extension of the shape-from-texture literature which uses texture normals to estimate shape [1, 7]. We can calibrate more that two cameras by performing pairwise calibration to get an initial estimate, then perform gradient descent on the complete set of cameras. Once we have a consistent orthographic camera calibration, we upgrade to a perspective set of cameras by initializing with the orthographic parameters and reasonable depth estimates, then use bundle adjustment to optimize.

4.1 Orthographic Calibration

For each patch in each image, there is a an (x,y) location, an (x,y,z) normal and an unknown sign bit that signifies the two-fold ambiguity of the normal. We will adopt the following notation for patch i in image k: the location is $(p^i_{k,x}, p^i_{k,y})$, the (unit) normal is $(n^i_{k,x}, n^i_{k,y}, n^i_{k,z})$ and the sign bit is $s^i_k \in \{-1, 1\}$. We will denote the transformation between the two cameras as the matrix Q and the relative scale of the two cameras as K. Since we assume that our cameras are orthographic, we know that Q should be orthonormal.

Using this notation, we can state our problem as find a Q that satisfies the following two

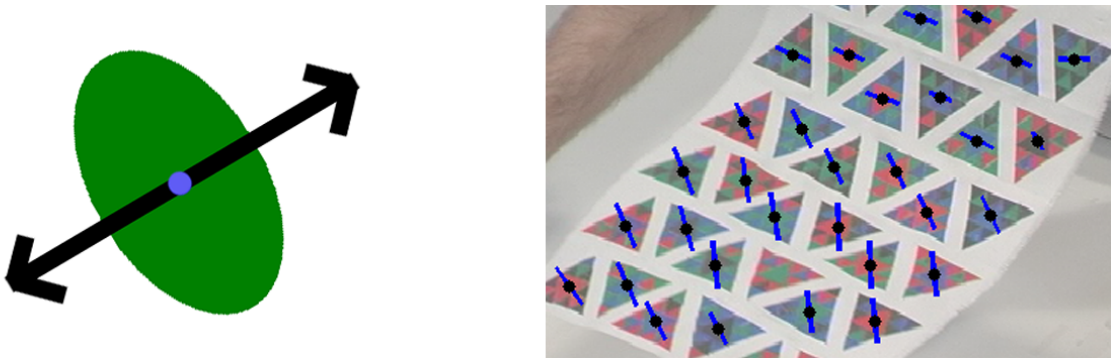

Figure 4: On the left, an oblique view of a circular texture element. Since we assume that we know that the frontal image of the texture element is circular, we can estimate the normal up to a two-fold ambiguity. The ambiguous normals are shown as the black errors and cannot easily be distinguished from a single view. However, with the solution provided in this paper will allows us to determine which normal is correct. On the left is this ambiguity show on real data.

equations:

$$\begin{bmatrix} p^1_{1,x} & \dots & p^n_{1,x} \\ p^1_{1,x} & \dots & p^n_{1,x} \\ \times & \dots & \times \end{bmatrix} = K \begin{bmatrix} & Q & \end{bmatrix} \begin{bmatrix} p^1_{2,x} & \dots & p^n_{2,x} \\ p^1_{2,x} & \dots & p^n_{2,x} \\ \times & \dots & \times \end{bmatrix}$$

$$\begin{bmatrix} s^1_1 \cdot n^1_{1,x} & \dots & s^n_1 \cdot n^n_{1,x} \\ s^1_1 \cdot n^1_{1,y} & \dots & s^n_1 \cdot n^n_{1,y} \\ n^1_{1,z} & \dots & n^n_{1,z} \end{bmatrix} = \begin{bmatrix} & Q & \end{bmatrix} \begin{bmatrix} s^1_2 \cdot n^1_{2,x} & \dots & s^n_1 \cdot n^n_{2,x} \\ s^1_2 \cdot n^1_{2,y} & \dots & s^n_1 \cdot n^n_{2,y} \\ n^1_{2,z} & \dots & n^n_{2,z} \end{bmatrix}$$

Here, the points p^i_k and the normals n^i_k are known and the signs s^i_k and orthonormal matrix Q are unknown. In some cases we will assume that the signs s^i_k are known and use the following simplified notation:

$$\left[\begin{array}{ccc|c} & P_1 & & \\ \times & \dots & \times & N_1 \end{array}\right] = \begin{bmatrix} & Q & \end{bmatrix} \left[\begin{array}{ccc|c} & K \cdot P_2 & & \\ \times & \dots & \times & N_2 \end{array}\right] \tag{4.1}$$

For each of these problems, ideally, we can solve for the matrix Q, the sign bits s_k^i and the corresponding depths $p_{k,z}^i$ which were represented above as $\times$. We will assume from this point on that the system of equations is over constrained. Exact solutions are, in general, not possible and we will focus on methods that minimize a cost function that charges for deviation from equality. In the following sections, we will explore different cost functions and corresponding solution methods.

4.1.1 Problem Characterization

Our problem, as defined above is actually ambiguous – even with normal information. Suppose that we find a solution to the problem in the form:

$$\begin{aligned} Q &= \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \\ s_k^i &= \hat{s}_k^i \end{aligned}$$

Then, we can compose an equally valid solution using:

$$\begin{aligned} Q^* &= \begin{bmatrix} q_{11} & q_{12} & -q_{13} \\ q_{21} & q_{22} & -q_{23} \\ -q_{31} & -q_{32} & q_{33} \end{bmatrix} \\ s_k^i &= -\hat{s}_k^i \end{aligned}$$

Using the notation from before:

$$\begin{bmatrix} \\ P_1 \\ \\ \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \begin{bmatrix} \\ K \cdot P_2 \\ \\ \end{bmatrix}$$

We will denote the ambiguity in the observation of the normals as N_1 for one consistent assignment of the ambiguity and N_1^* for the opposite assignment. These two assignments correspond to two different scenarios, but are indistinguishable from the image data alone.

$$\begin{bmatrix} N_1 \end{bmatrix} = \begin{bmatrix} Q \end{bmatrix} \begin{bmatrix} N_2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N_1 \end{bmatrix} = \begin{bmatrix} Q \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N_2 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N_1 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & -q_{13} \\ q_{21} & q_{22} & -q_{23} \\ -q_{31} & -q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N_2 \end{bmatrix}$$

$$\begin{bmatrix} -\hat{s}_1^1 \cdot n_{1,x}^1 & \dots & -\hat{s}_1^n \cdot n_{1,x}^n \\ -\hat{s}_1^1 \cdot n_{1,y}^1 & \dots & -\hat{s}_1^n \cdot n_{1,y}^n \\ n_{1,z}^1 & \dots & n_{1,z}^n \end{bmatrix} = \begin{bmatrix} Q^* \end{bmatrix} \begin{bmatrix} -\hat{s}_2^1 \cdot n_{2,x}^1 & \dots & -\hat{s}_1^n \cdot n_{2,x}^n \\ -\hat{s}_2^1 \cdot n_{2,y}^1 & \dots & -\hat{s}_1^n \cdot n_{2,y}^n \\ n_{2,z}^1 & \dots & n_{2,z}^n \end{bmatrix}$$

$$\begin{bmatrix} N_1^* \end{bmatrix} = \begin{bmatrix} Q^* \end{bmatrix} \begin{bmatrix} N_2^* \end{bmatrix}$$

This ambiguity is a failure of the points and normals representation of the information in the scene, and can only be resolved using other vision cues or more than two cameras. For the rest of this section, we will assume $s_1^1 = 1$.

4.1.2 Orthographic solution

We solve this problem in two parts: first, we present the solution if all of the signs (s_k^i) are known. Then, we'll present a heuristic method for searching for the correct signs.

Known Signs We start by solving for the matrix Q assuming that we know the sign bits. This allows us to simplify our problem to a nonlinear optimization problem in three variables. To do this, we define two cost functions: one for the points and one for the normals. Following the work from previous sections, these costs are:

$$\begin{aligned} C_p &= \|\mathcal{C}\mathcal{C}^\dagger\mathcal{D} - \mathcal{D}\| \\ C_n &= -\text{tr}(N_1^T \cdot Q \cdot N_2) \\ C &= C_p + \lambda C_n \end{aligned}$$

The normal cost is the dot product of the normal in the first frame with the transformed normal from the second frame. We will parameterize the rotation as a composition of three Given's rotations:

$$Q = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

One can easily verify that this matrix is orthogonal, and Euler's Rotation Theorem states that it defines the entire class of rotations in $\Re^3$.

Now, with a cost function that can be computed as a function of three variables, we perform gradient descent using the built in optimization toolbox in MATLAB.

Unknown Signs An arbitrary system requires a search over the $(2^N)^2$ possible sign bits. However, an over constrained system that satisfies the equality constraints can be solved in a search over a constant number of sign flips.

If we denote the number of points as N_p, our system has $3 + N_p$ unknowns (3 for the rotation matrix Q and N_p for the depth of each points) and $6 \cdot N_p$ equations (3 for each point and 3 for each normal). Technically, that means that any patch should be enough to solve for the transformation matrix Q. Therefore, if the coefficients in the system of equations had no errors, we could compute the rotation matrix Q from a single match (with a search over a single normal ambiguity in each view) and then compute all subsequent normal ambiguities. We compute the two sign bits using the following combinatorial minimization over 4 possibilities:

$$\min_{(s_1^i, s_2^i) \in \{-1,1\}} \begin{bmatrix} s_1^i \cdot n_{1,x}^i \\ s_1^i \cdot n_{1,y}^i \\ n_{1,z}^i \end{bmatrix}^T \cdot Q \cdot \begin{bmatrix} s_2^i \cdot n_{2,x}^i \\ s_2^i \cdot n_{2,y}^i \\ n_{2,z}^i \end{bmatrix}$$

Since we have errors in our coefficients, we will robustly search for a camera by picking N_m matches at random and combinatorially search over $(2_m^N)^2$ minimization problems as defined in the previous section. Once we have selected the rotation matrix Q, we disambiguate the rest of the matches, then recompute a more accurate Q using all of the matches. Since any pair of matches may be inaccurate, we repeat this process many times and pick the lowest score in the set.

4.2 Perspective Calibration

We optimize over a richer set of parameters corresponding to perspective cameras in a fashion similar to bundle adjustment. While few individual images of the cloth contain significant perspective effects, the range of motion of the cloth does — see figure 5. Our parameterization of the perspective camera is similar to the established model. Using q as a rotation matrix, t as translation and f as the focal length, our model is:

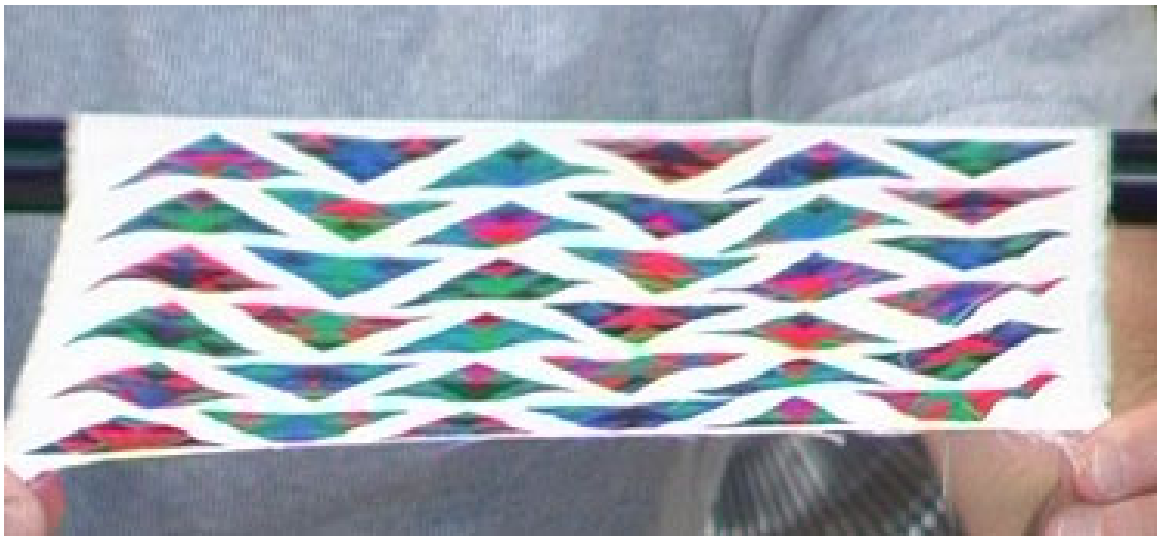

Figure 5: *Cloth can show significant perspective effects in some views, typically when the plane of the cloth is at about* 90° *to the plane of the camera.*

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{ft_z} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The key modification here is that we think of the camera as viewing point near the origin. As the camera moves away from the origin, the characteristics become closer to orthographic. In our parameterization, as $t_z \to \infty$ the model simplifies to an orthographic model:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

We solve for the parameters of the perspective camera by initializing them to be near our orthographic predictions, except for the estimated depth, which we set based on physical estimates

of our recording setup. With these initial estimates we optimize over the camera parameters. Our cost function is the reconstruction error of the points in each image. Using (x_i^c, y_i^c) as the observed points in camera c, $\mathcal{C}$ as the set of cameras that observe the point, p as the point in 3d, (r_1, r_2, r_3) as the rows of the rotation matrix R, and w_c as the relative weighting of camera c, we compute the reconstruction error as:

$$\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} w_c \sqrt{(x_i^c - f\frac{t_z(r_1 p + t_x)}{r_3 p + t_z})^2 + (y_i^c - f\frac{t_z(r_2 p + t_y)}{r_3 p + t_z})^2}$$

The weighting allows us to control which cameras contribute more to the reconstruction and which contribute less. For now, we set the weights w_c to 1, and optimize over the 3d location p. With the optimal locations of all of the points in the mesh, we can compute an average reconstruction cost associated with a set of parameterize – allowing a larger optimization in these parameters. This search is slow and non-linear, but the initialization with an orthographic camera is good enough to yield good results.

Chapter 5

3D Reconstruction

5.1 Matching Triangles

In cloth segments that are large enough for real world applications, our pattern does not include enough distinctive elements to cover the entire cloth. We compensate for this by minimizing repetition and distributing a small number of unique elements over the larger surface. (in the skirt example, there are roughly 35 unique triangles on a cloth with 432 total triangles)

The unique triangles are used for camera calibration, and provide a starting point for matching. Non unique triangles face two problems: correspondence and parameterization. Correspondence can be easily found through epipolar constraints, but parameterization is harder. We phrase the problem as follows: Given a number of triangles in the 2D cloth domain with similar coloring, which triangle in the 3D reconstruction corresponds with which 2D triangle? We solve this with a simple heuristic — local neighborhoods should be similar. While our heuristic is fallible, in practice we have observed no failures.

5.2 Combined Optimization

Our reconstruction method takes on an unusual form of structure from motion. Because cloth changes shape in every frame, the number of views of any one configuration is small. We have

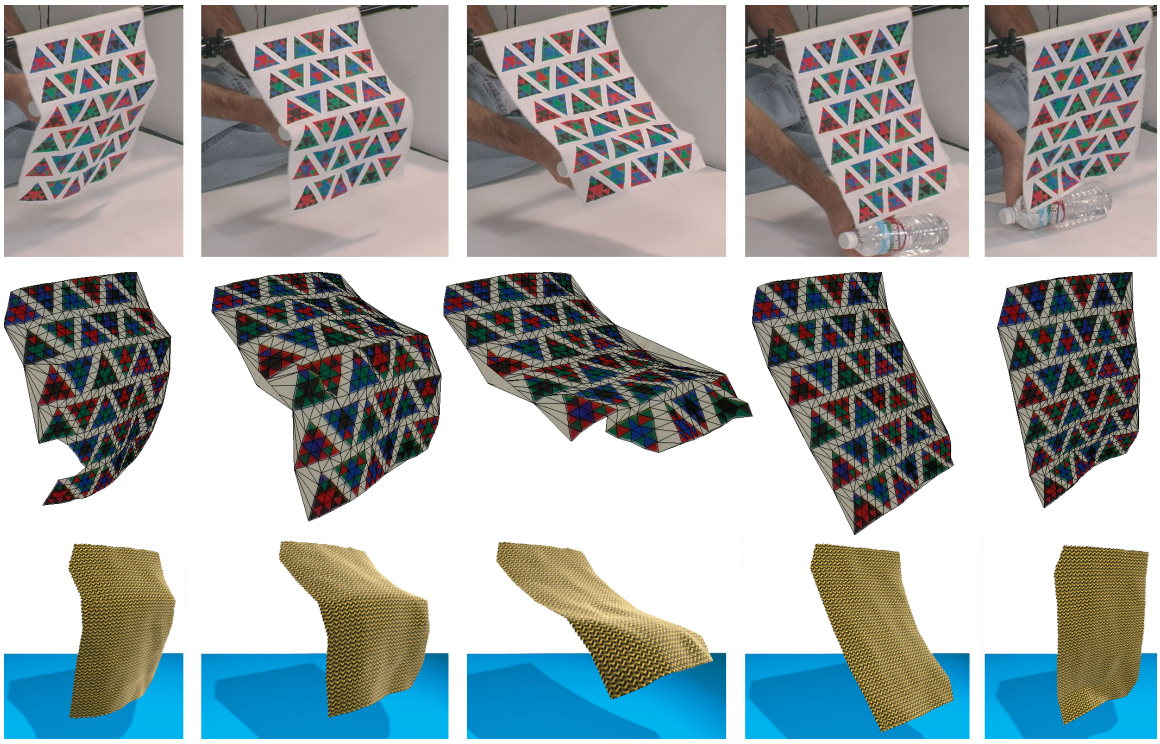

Figure 6: *On the top row, camera frames showing the cloth being pushed by a bottle. The middle row shows reconstruction without interpolation or strain reduction. On the bottom row, the sequence has been strain reduced as a post-processing step and is rendered with a new cloth texture. This suggests that combining the reconstruction and strain reduction will produce a sequence that is both visually convincing and true to the image data.*

only four cameras — significantly fewer than a typical structure from motion setup. As a result, heavily foreshortened triangles are problematic, and can easily be missed in some views. It is not uncommon for these triangles to be viewed exclusively by cameras with a small baseline, causing minor errors in observation to result in large errors in depth estimation. Figures 7 and 9 emphasize these problems.

We build on the standard approach by using cloth specific knowledge to drive reconstruction. The conventional argument prescribes minimizing the reprojection error to reconstruct the 3D locations of points from image correspondence. We improve upon this by penalizing large strains, after an idea due to Provot [10]. Small strains in cloth result in relatively small forces, but larger strains can produce very large forces. Because we have recovered a parameterization, we can

observe strains in the recovered cloth model. We create a global cost function that combines the reconstruction error in each camera with the strain on the mesh (defined by a Delaunay triangulation of the points in the cloth domain). Using $\|\mathrm{e}\|$ as the edge length, $\|\mathrm{e}_r\|$ as the rest length, $E_r(\mathbf{p})$ as the reconstruction error defined in the previous section and k_s as the weight of strain relative to reconstruction error; our cost function is defined as:

$$\begin{aligned} \text{strain}(\mathrm{e}) &= \begin{cases} (\|\mathrm{e}\| - \|\mathrm{e}_r\|)^2 & \text{if } \|\mathrm{e}\| > \|\mathrm{e}_r\| \\ 0 & \text{otherwise} \end{cases} \\ \text{cost} &= k_s \sum_{\mathrm{e} \in \text{edges}} \text{strain}(\mathrm{e}) + \sum_{\mathbf{p} \in \text{points}} E_r(\mathbf{p}) \end{aligned}$$

Because optimizing this objective function involves simultaneously solving for thousands of variables, we adopt a multi-stage approach to reconstructing the 3D points. First, the points are reconstructed without any strain information because each 3D location can be computed independently. Because many observational errors occur at the scale of the large triangles, we minimize a course scale version of the global objective function to produce a start-point for the final optimization problem.

Even with a good starting point, the large optimization problem is intractable without careful attention to detail. First, we reduce computation in numerically computing the gradient by exploiting conditional independence between points on the surface that are significantly far apart. Second, by exploiting the connectivity structure of the surface, we constrain numerical estimates of the Hessian to a sparse matrix form (c.f. [12]).

The combined strain reduction, point reconstruction has little effect on the actual reprojection errors: typically an increase of less than 0.2 pixels. Because the most accurate views of a triangle are typically separated by a small baseline, small errors in localization become large errors in depth estimation. The strain reduction only needs to have small effects on the reprojected location of the point to dramatically increase the quality of the reconstructed mesh, as shown in Figure 7.

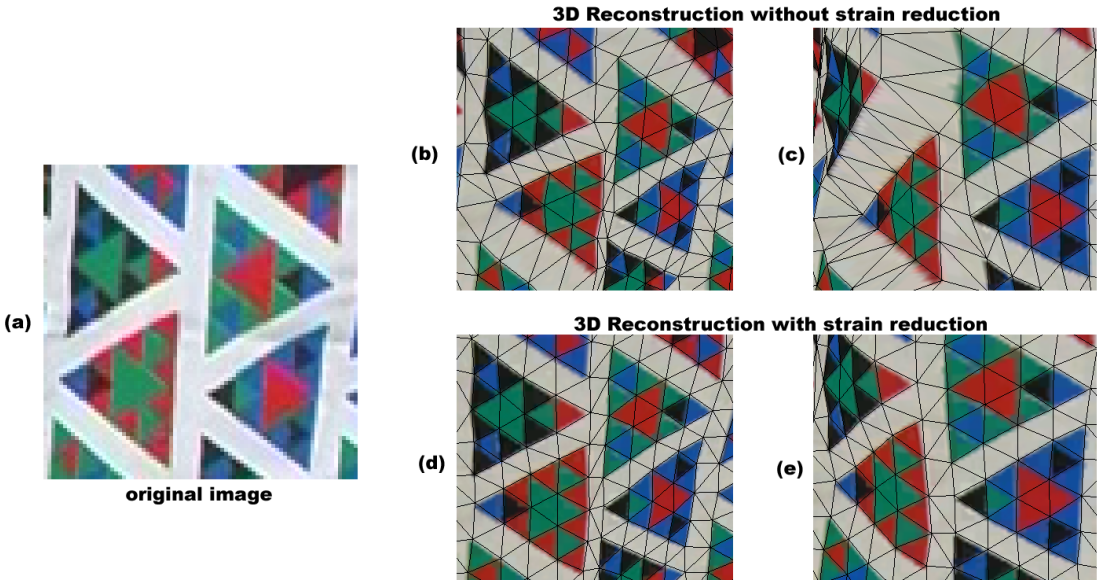

Figure 7: *Because we have a small number of views of each triangle, minimizing the reprojection error alone only produces an accurate mesh when viewed from similar viewpoints. Images* ***(b)*** *and* ***(d)*** *are* ***rendered*** *views of the* ***reconstructed*** *mesh (textured with a frontal view of the flattened cloth) taken from viewpoints similar to the original image* ***(a)****. However, without strain reduction, novel views do not exhibit cloth-like structure. The reconstructed mesh in image* ***(c)****, produced by minimizing reprojection error alone, is rendered from a view significantly different from all the original cameras. Note that this results in significant variance in the mesh — indicated by large variations in edge length. Image* ***(e)*** *shows a similar rendered view of a reconstructed mesh produced by* ***simultaneously*** *minimizing reprojection error and strain (section 5.2). Now, the structure of the reconstructed mesh is significantly more realistic — seen as uniform edge lengths — while still true to the original image data.*

5.3 Post Processing

In general, we wish to post process reconstructions as little as possible, because we have been careful with our measurements. However, some steps produce a worthwhile improvement. We start with a slightly noisy mesh with holes for each time frame.

Interpolation works by identifying a neighborhood of the points — ideally, in space and time; in time, if spatial neighbors are missing; in space, if temporal neighbors are missing — and re-estimating the configuration of the missing point using a multilinear interpolate. This estimation is done *before* the final minimization of strain and reprojection error — so that large strains are removed. Because the interpolated points are not observed, there is no corresponding reprojection

cost.

Smoothing: There is still a high-frequency component of temporal noise in the reconstruction; we polish the 3D points with a Gaussian low-pass filter, with σ of one inter-frame interval.

Chapter 6

Results

6.1 Printing the cloth

Despite a slew of other options, we found that screen printing cloth inks onto rayon provided the best comprise in printing. We used several methods to evaluate different printing methods: ease of printing, vibrance of colors, ability to represent fine detail and preservation of cloth dynamics.

Screen printing allows us to repeatedly print the same pattern with different colors with a high degree of spatial resolution. The colors are vibrant and the impact on the cloth dynamics is somewhat low. (The triangles are less likely to bend than the remaining material, but the results were still better than other methods) However, aligning screens of different colors still proved to be problematic — resulting in small gaps between colors. Because these gaps make triangle identification significantly more difficult, we used markers to fill in the gaps.

Other methods include iron-on transfers, markers, spray paint and printing directly on the cloth using a cloth printer. Iron-on transfers altered the cloth dynamics too significantly. Markers had almost no effect on the dynamics, but were too time consuming to use and difficult to get fine detail (the markers bled at the corners). Spray paint was difficult to control, and we didn't own a cloth printer.

In printing, we used 4 colors: red, green, blue and black. While these colors are a decent set, in retrospect, we would eliminate black, which was easily confused for other colors and caused significant problems.

6.2 Video-taping the cloth

To film the dynamics of the cloth, we used 4 digital video cameras, each sampling at 30 frames per second with a shutter speed of 1/250. Faster shutter speeds adversely affected the color quality of the recording while slower shutter speeds had unacceptable motion blur. To minimize shadows and increase the depth of focus, we heavily lit the scene, using 5 lights, each consuming 500 - 1000 W. The cameras did not record frames simultaneously and were only synchronized to within one frame using visual cues.

The lack of synchronization is a significant problem. We can gauge quality of the results by reconstruction error, and note that in slow moving parts of the sequence, reconstruction error was below 0.5 pixels. However, faster moving parts had reconstruction errors on the order or 3 pixels — which is roughly the scale of the observed motion (typically up to 10 pixels per frame).

6.3 Rendering / Retexturing

To display the effectiveness of our algorithm, we captured several different sequences of cloth movement. We display our mesh retextured with a frontal photo of the original texture for easy comparison in figure 8. A larger portion of the sequence, textured with new image is shown in figure 10.

Figure 8: *Large scale folds and wrinkles can be captured with high precision and detail. On the mid and lower portions of the skirt, the folds are faithfully recovered. However, fine scale folds can be missed when triangles are heavily fore-shortened, occluded or curved. In this figure, some level of detail is lost in the upper left hand corner. However, without viewing the original image, the resulting mesh is still convincingly cloth-like.*

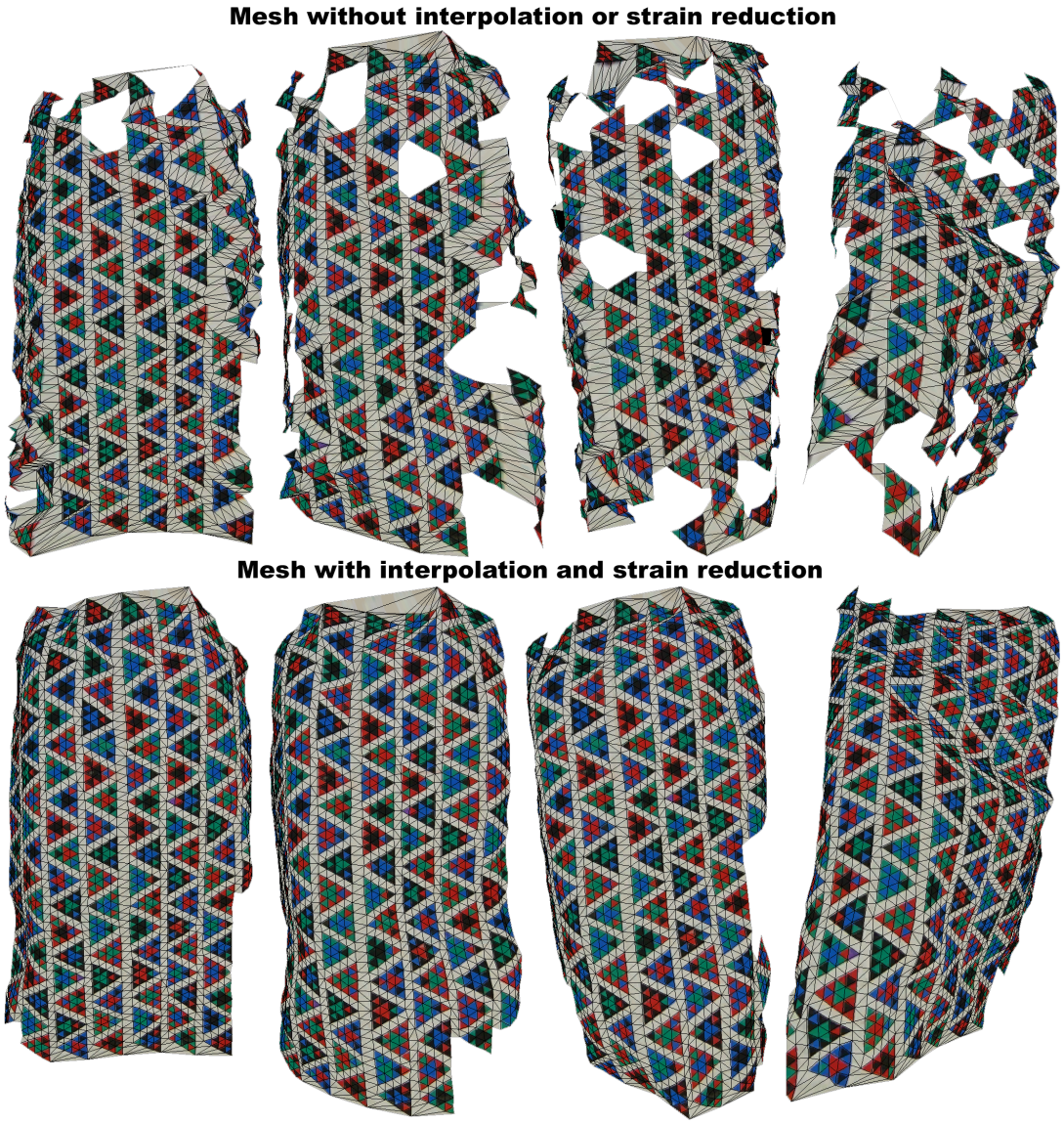

Figure 9: *We demonstrate the strength of our pattern, the accuracy of localization and the importance of combining the minimization of strain and reprojection error. This sequence shows the front of a skirt, viewed from a direction not seen by any camera, retextured with the original pattern for clarity. Each frame contains roughly 2000 vertices comprising almost 4000 triangles. In the* ***top row****, the surface has been reconstructed by minimizing reprojection error alone — gaps appear in the mesh when fewer than two cameras view any large triangle. Furthermore, when relatively few cameras observe a camera, the reconstruction can be inaccurate in some directions — seen as triangles that deviate significantly from the rest of the mesh. This can be corrected by taking strain into account. The* ***bottom row*** *uses interpolation to fill in missing points before running a simultaneous minimization of both reprojection error and strain. An actual image of the skirt taken from one of the cameras can be found in figure 8*

Figure 10: *Re-texturing a sequence with any pattern or texture is easy because the coordinates in the parameter space are kept at every stage of the sequence. The* ***top row*** *shows a sequence of the front of a skirt rendered with a new texture from a new view. The* ***bottom row*** *shows the same sequence of from one of the camera viewpoints.* ***Important:*** *There is roughly a* 45° *change in angle between the camera viewpoint and the cloth to emphasize the folds in the captured images.*

Bibliography

[1] D.A. Forsyth. Shape from texture without boundaries. In *Proc. ECCV*, volume 3, pages 225–239, 2002.

[2] Igor Guskov, Sergey Klibanov, and Benjamin Bryant. Trackable surfaces. In *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 203–208, 2003.

[3] Igor Guskov and Leonid Zhokov. Direct pattern tracking on flexible geometry. In *Winter School of Computer Graphics*, pages 203–208, 2002.

[4] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2000.

[5] D.H. House, D. Breen, and D. Breen, editors. *Cloth Modelling and Animation*. A.K. Peters, 2000.

[6] T.S. Huang and C.H. Lee. Motion and structure from orthographic projections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):536 – 540, 1989.

[7] Anthony Lobay and D.A. Forsyth. Recovering shape and irradiance maps from rich dense texton fields. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2004.

[8] D. Pritchard. Cloth parameters and motion capture. Master's thesis, University of British Columbia, 2003.

[9] D. Pritchard and W. Heidrich. Cloth motion capture. *Computer Graphics Forum (Eurographics 2003)*, 22(3):263–271, September 2003.

[10] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface 95*, pages 147–154, 1995.

[11] Volker Scholz and Marcus A. Magnor. Cloth motion from optical flow. In B. Girod, M. Magnor, and H.-P. Seidel, editors, *Proc. Vision, Modeling and Visualization 2004*, pages 117–124, Stanford, USA, November 2004. Akademische Verlagsgesellschaft Aka GmbH.

[12] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372. Springer-Verlag, 2000.

[13] S. Ullman. *The Interpretation of Visual Motion*. MIT press, 1979.