Combining Cues: Shape from Shading and Texture

Ryan White
David Forsyth

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2005-14
http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-14.html

November 17, 2005

Combining Cues: Shape from Shading and Texture

Ryan White, D.A. Forsyth
{ryanw,daf}@cs.berkeley.edu

November 17, 2005

Abstract

We demonstrate a method for reconstructing the shape of a deformed surface from a single view. After decomposing an image into irradiance and albedo components, we combine normal cues from shading and texture to produce a field of unambiguous normals. Using these normals, we reconstruct the 3D geometry. Our method works in two regimes: either requiring the frontal appearance of the texture or building it automatically from a series of images of the deforming texture. We can recover geometry with errors below four percent of object size on arbitrary textures, and estimate specific geometric parameters using a custom texture even more accurately.

Keywords: Shape-from-texture, shape-from-shading, 3D reconstruction, surface tracking, deformable models

1 Overview

We demonstrate reconstructions from a single view using a combination of shape and texture cues. We show our reconstructions are geometrically accurate by comparison with reconstructions from multiple views. Traditionally, reconstruction techniques are limited by ambiguities in the local cues — and reconstructions are performed by breaking these ambiguities using global consistency. Instead, we break ambiguities locally and only introduce long scale consistency in the finals steps of the geometric reconstruction.

We start with an outline of the reconstruction process. First, we obtain an estimate of the frontal appearance of the texture. This can be supplied manually, or reconstructed automatically from a sequence of images (section 5, figure 6). Second, we decompose the image into an irradiance map and a texture map (section 3.1, figure 5). Third, we obtain ambiguous estimates of the surface normals using the frontal texture, the texture map and the assumption of local orthography [2, 3]. The normals are disambiguated using a shading model applied to the irradiance map (section 3.2). Finally, the surface is reconstructed using perspective effects to break a final concave convex ambiguity (section 4).

The resulting reconstructions are highly accurate – estimating geometry within four percent. Only a small number of papers numerically quantify the accuracy of a geometric reconstruction from a single image — and none of them in similar situations using real data. Forsyth reconstructs the shape of a synthetic image of a textured sphere [3] and Savarese et al. reconstruct the shape of a cylindrical mirror using the reflection of a calibration pattern [11].

2 Background

Shape-from-Shading Starting in [5], most work in shape-from-shading makes simplifying assumptions: the scene is illuminated with a point light source, the surface of the object is lambertian and inter-reflections are minimal. Methods typically use either local intensity or local gradient information. Constraints such as surface smoothness, integrability and boundary conditions break ambiguities. However, scenes exhibit effects from mutual illumination [4], surfaces are not entirely lambertian and often there are multiple light sources. Even in the cases where the assumptions hold, current reconstruction methods are unreliable [16].

Shape-from-Texture Texture normals can be computed as a transformation between an observed patch and a frontal view by assuming that the patch is locally flat. Local estimation of surface normals is limited by two problems: one must know the frontal appearance of the textured patch, and each estimated normal is actually an ambiguous pair: the estimated normal and its mirror across the viewing direction. Forsyth [2, 3] and Lobay and Forsyth [7] focus on estimating the frontal texture from a single view of a repeating pattern and use surface smoothness to resolve the texture normal ambiguity. Loh and Hartley also use perspective effects to break ambiguities [8].

We take a different stance on both topics. First, we estimate the frontal appearance of a non-repeating texture pattern using multiple images of the deforming object. Second, we break ambiguities in texture normals using shading cues. There is no reason to limit combinations of the two techniques. One could estimate the frontal pattern of a repeating texture and use shading cues to break ambiguities. Though less reliable, one could also estimate the frontal view of an un-shaded deforming surface in multiple views and use smoothness to break texture normal ambiguities.

Combined Shading and Texture Choe et al [1] reconstruct the 3D shape of shaded images of physically generated random textures based on surface irregularities. They focus on models of natural textures (reconstructing the shape of a tree trunk) and do not consider using shading to break texture ambiguities.

Deformable Surface Tracking Several methods have been proposed to track nonrigid motion [12, 9]. Pilet et al [9] describe a method to detect the surface deformation using wide-baseline matches between a frontal view and the image to estimate a transformation smoothed using surface energy. This method has difficulty stabilizing texture for 3 reasons: there are few reliable keypoints in textures we consider; because internal keypoints are used, boundaries are not tracked well; and the surface energy term makes it difficult to track highly deformable objects. Also, they cannot obtain an irradiance estimate using the surface track because errors in correspondence would make it impossible.

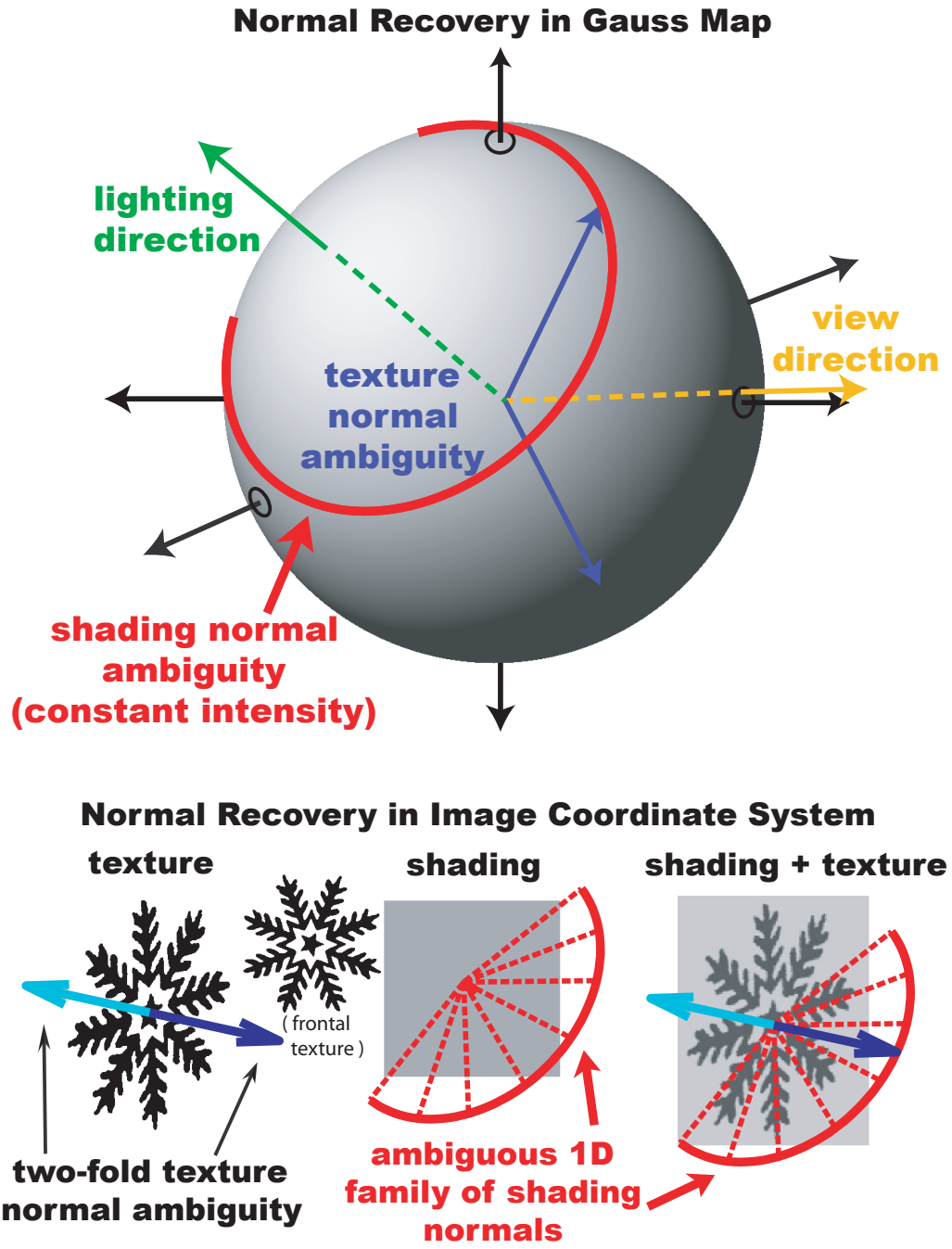

Figure 1: *While local cues for estimating normals are ambiguous when considering shading or texture alone, the ambiguities can be broken in most cases by combining cues. On the* ***top****, we view a gauss map of a sphere shaded with a point light source. The red arc denotes an isophote (constant pixel intensity) — a single measurement of image intensity constrains the normal to this curve. Observations of texture produce a two-fold ambiguity symmetric across the viewing direction. In most cases, the ambiguities do not line up (for exceptions, see figures 2)and the normal can be determined. On the* ***bottom****, we show the same phenomenon from the viewpoint of the camera. An observation of a textured patch can have two possible normals, while a single measurement of intensity has a 1 parameter family of normals. By combing cues, the normal can be determined.*

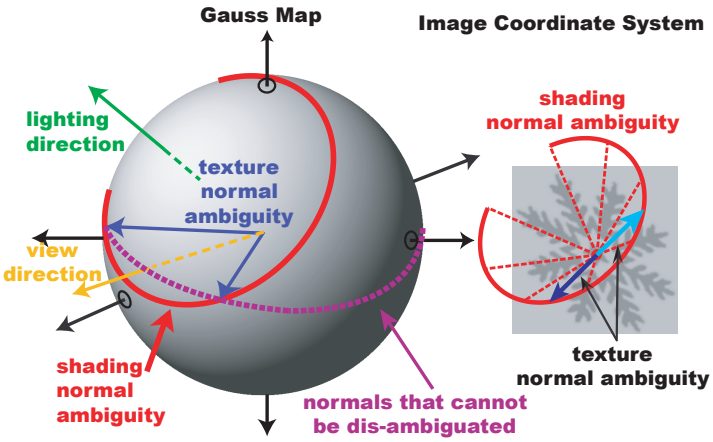

Figure 2: *Shading usually breaks texture normal ambiguities; however, there are cases when the two ambiguities line up and the ambiguity can not be broken. In this image, the two ambiguous texture normals lie on the same arc of constant irradiance, meaning that the reconstruction is still ambiguous. The purple line shows the set of all normals where shading will not break the texture ambiguity: we call this* ***combined cue ambiguity****.*

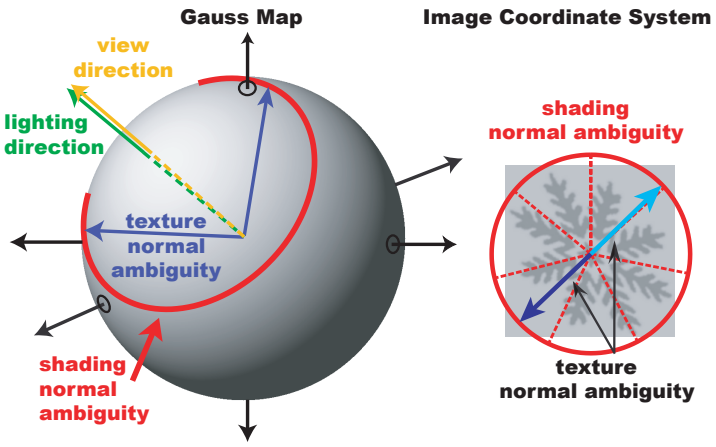

Figure 3: *When the lighting direction and the viewing direction are the same, none of the normals can be dis-ambiguated.*

3 Estimating Normals

Our method requires the extraction of accurate normals to estimate geometry. To compute surface normals, we estimate ambiguous normals using texture cues, then break the ambiguity using a lighting model and shading cues. As part of the process, we decompose images into two components: albedo and irradiance — which correspond to the two aspects of normal estimation: texture and shading.

Because there are several distinct ambiguities, we adopt the following terminology: **Texture normal ambiguity** (or **texture ambiguity**) refers the two-fold ambiguity in estimating a single normal from an observed texture. **Combined cue ambiguity** refers to the uncommon texture ambiguity that remain ambiguous after using shading cues to break ambiguities. Finally, **scene ambiguity** refers to the concave convex ambiguity that results from confusion about the source of the lighting.

In general, we break **texture normal ambiguity** using shading information. We use belief propagation and surface smoothness to break the infrequent **combined cue ambiguity**. Finally, we use perspective effects on the scale of the scene to eliminate **scene ambiguity**.

3.1 The Lighting Model

In this section, we describe the lighting model and use it to decompose the image into two distinct parts: a texture map and an irradiance map. We detail the process of breaking the texture ambiguity in the next section and subsequently discuss the determination of the lighting parameters in section 3.3.

Following previous work in shape-from-shading, we assume that our scene is lit by a combination of a point light source and a non-directional constant background illumination. Using $\mathbf{L}$ as the direction to the light source, $\mathbf{N}$ as the surface normal, ρ_d as the surface albedo, L_p as the point source intensity and L_a as the background illumination intensity, the intensity $\mathcal{I}_i$ of a pixel i in the image is:

$$\mathcal{I}_i = \mathbf{L} \cdot \mathbf{N} \cdot \rho_d \cdot L_p + \rho_d \cdot L_a$$

If we know the surface albedo ρ_d and the lighting model parameters ($\mathbf{L}$, L_p, L_a), then we can compute the angle between the light and the normal (although the normal direction itself is limited to a 1 parameter family) and subsequently determine an irradiance image — one where the surface albedo is constant for the entire image.

Estimating Irradiance: Following the work in [14], we focus on screen print items (ones that have a finite set of colors) and use a classifier to determine which dye color generated each pixel. The output of the classifier is the texture map for the image. To build the irradiance map, we estimate an image with constant albedo ($\rho_d = 1$). This image can be generated by dividing the observed pixel intensity $\mathcal{I}_i$ by the albedo for the dye color (ρ_d). While a texture map may provide a rough guide to the surface albedo, the color classifier is much better because it is not effected by small alignment errors.

3.2 Breaking Texture Ambiguity with Shading

Figure 1 provides a geometric perspective on the process of breaking texture ambiguity. Each observation of irradiance ($\mathcal{I}_i$) confines the surface normal to a 1 parameter family of possible normals. Each textured patch limits the normal to a set of size two. In the ideal case, the intersection of these sets produces one and only one normal. However, in practice the sets do not intersect. Because we believe normal estimates from texture to be substantially better than normal estimates from lighting, we use the shading cue to break the texture ambiguity.

We formulate the problem of breaking the texture ambiguity as an error between the irradiance computed using texture normals under the lighting model and the irradiance image. Writing N_{i1} and N_{i2} for the two possible texture normals and δ_i as an indicator variable to switch between the two choices, we write the cost as:

$$\sum_i (\mathbf{L} \cdot N_{i1} \cdot L_p + L_a - \mathcal{I}_i)^2 \delta_i + (\mathbf{L} \cdot N_{i2} \cdot L_p + L_a - \mathcal{I}_i)^2 (1 - \delta_i)$$

If we assume that L_a, L_p and $\mathbf{L}$ are known, then the process of breaking the texture ambiguity (or, correspondingly the value of δ_i) is simple: for each ambiguous normal choose the normal with the lower cost.

However, when the lighting (cost) for the two possible texture normals is the same, shading will not break the texture ambiguity, resulting in a **combined cue ambiguity**. While this formal

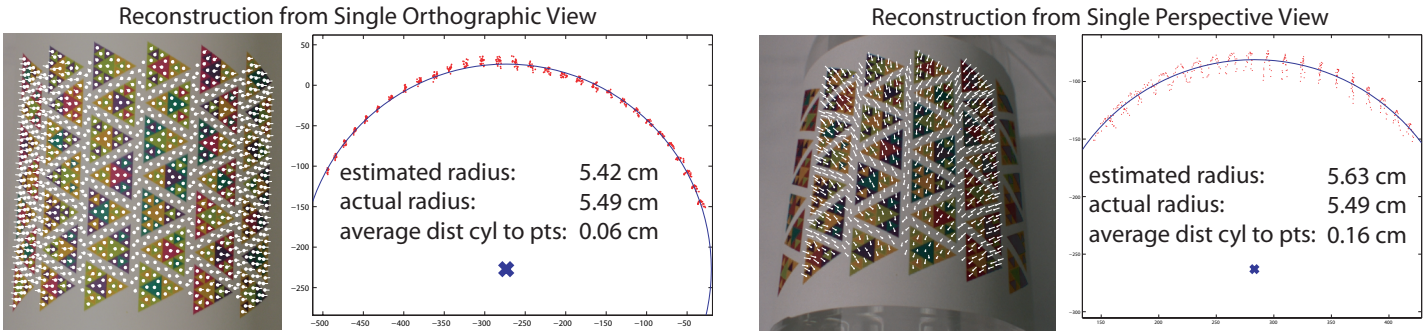

Figure 4: *Using both texture and lighting cues, accurate models of shape can be made **from a single view**. Using a custom pattern taped to a cylinder (a 2L bottle), we reconstruct the shape twice: in an image with minimal perspective effects (**left**) and in an image with more prominent perspective effects (**right**). In both cases, the reconstructions are performed using the same code, assuming orthography on the scale of a triangle, and perspective effects on the scale of the scene. The 3D reconstructions are viewed down the axis of the best fit cylinder — where the blue x denotes the estimated axis and the arc denotes the best fit surface. The red point are the vertices of the points. The quality of the fit is the average distance between the cylinder and the point cloud — in the orthographic case 2.98 pixels (0.637 mm) and 4.97 pixels (1.57 mm) in the perspective case. Errors in estimating the radius: 2.82 pixels (0.602 mm) in the orthographic image and 4.46 pixels (1.41 mm) in the perspective image.*

ambiguity is limited to a single arc on the gauss map, because of unmodeled effects the region of remaining ambiguity is a thick line region around the formal ambiguity. Even worse, when the lighting direction and the viewing direction are the same, shading cues will not break any texture normal ambiguities. (figure 3)

3.3 Determining the Lighting Direction

Lighting parameter recovery is based on two observations: First, texture cues provide information about the accuracy of lighting parameters. Second, because the lighting model has a small number of degrees of freedom (four: two for the lighting direction $\mathbf{L}$, one each for the point light intensity L_p and the background intensity L_a) it is not necessary to search over the 2^D values of δ_i to find an optimal set of parameters. (D is the number of normals)

In fact, for a given lighting model there are not 2^D values of δ_i because some choices necessarily have higher costs than others. There are in fact $O(D^4)$ by the following argument. The value of δ_i changes at points where either of the two linear constraints is true:

$$(\mathbf{L} \cdot N_{i1} \cdot L_p + L_a - I_i) = (\mathbf{L} \cdot N_{i2} \cdot L_p + L_a - I_i)$$
$$(\mathbf{L} \cdot N_{i1} \cdot L_p + L_a - I_i) = -(\mathbf{L} \cdot N_{i2} \cdot L_p + L_a - I_i)$$

This means that the δ_i are constant within the cells of the arrangement of hyperplanes given by these constraints, and there are $O(D^4)$ such cells. Given a particular δ_i, the error is a convex function within a convex cell and can be minimized by standard methods. Instead of building the arrangement, we grid the variables (also $O(D^4)$) and start an optimization problem at the minimal grid point.

The location of the lighting source, however, is still ambiguous up to a **scene ambiguity**: a mirror light source across the viewing direction will produce the same error. The two light sources translate into an ambiguity in the sign of the depth for the entire image (or a concave convex ambiguity). Even in scenes with little depth, this ambiguity can be broken with perspective effects. (section 4).

3.4 Breaking the Remaining Ambiguities

As noted in previous sections, there are some normals that are intrinsically ambiguous using shading and texture cues. We break the ambiguity in these normals by assuming that the surface is smooth and that the surface normals change slowly – therefore neighboring normals should be similar. While previous approaches used this assumption to break all ambiguities, our methed is more reliable because only a small set of normals have combined cue ambiguity.

We detect ambiguous normals by comparing the lighting errors between the two texture normals. Small differences indicate ambiguity. We threshold, asserting that roughly 90% of the gauss map is not ambiguous. The confidence for remaining 10% of normals is weighted linearly by the difference in errors between the two texture normals. We set up a new graph, where the nodes in the graph are the triangles in the surface mesh and the edges in the graph are edges in the mesh. Using a probability model that favors neighboring normals that agree, a few steps of loopy belief propagation on this graph produces un-ambiguous results.

4 Recovering Geometry

To recover a full 3D model from dis-ambiguated normals, we iterate over two stages: (1) compute scene depth assuming orthography using normals; (2) adjust the point locations to account for perspective effects. Because of **scene ambiguity**, the sign of the depth is ambiguous and we start the process twice: once for each sign of the depth. By picking the solution with minimal error, we get a metric reconstruction of the scene from a single view.

We assume that perspective effects are not observable on the scale of an individual triangle but are only observable (albeit minimal) on the scale of the scene. If perspective effects were viewable on the scale of the element, then either (a) the element is very large and not very descriptive; or (b) the element is at a highly oblique angle and detection is unreliable. In practice, the validity of these assumptions is demonstrated by two empirical facts: (1) texture normals computed assuming local orthography are very accurate; (2) in practice, scene ambiguity can be resolved in images with incredibly weak perspective effects (figure 4).

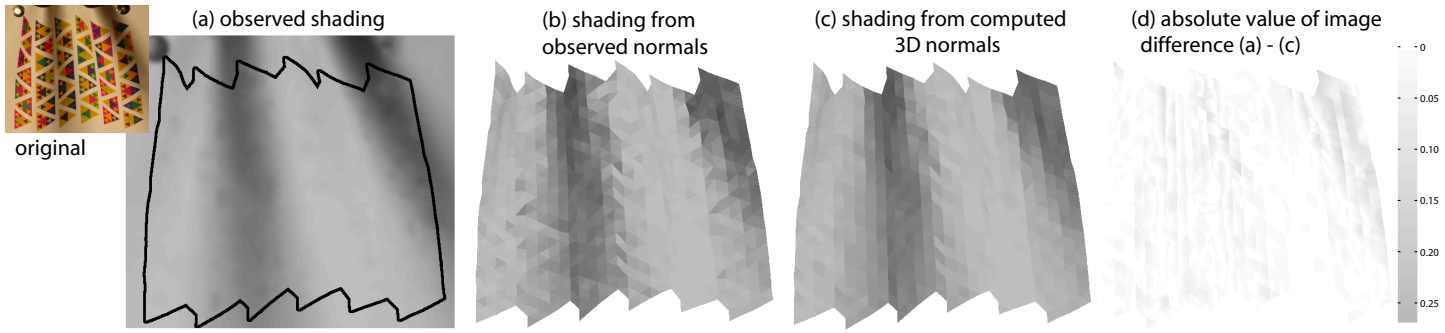

Figure 5: *By comparing the original image shading with two different shading reconstructions, we show (1) that agreement between shading and texture cues is fairly robust and (2) that estimating 3D geometry effectively smoothes over noisy normals. In (a), we show an image of the shading (derived from the frontal image in the inset, also shown in figure 8). This shading was estimated using the method in [14]. In (b), we render the lighting assuming our simplified lighting model, using the dis-ambiguated normal to generate appropriate shading. Because the normals are calculated up to two-fold ambiguity from texture cues, we can interpret the agreement between this image and the extracted shading image as the agreement between texture and shading cues. Finally, in (c), we render the normals from the 3D geometry in the same way. While no additional information about shading was incorporated between the middle and right images, by estimating 3D geometry, we effectively smooth the normal field — producing more realistic results. The difference image (d) shows that the resulting errors are small.*

4.1 Orthographic Depth from Normals

We estimate the orthographic depth from normals using a mesh constructed out of triangles in the image plane. Using gradient descent, we constrain the x and y locations of the vertices while allowing the z value to vary. Our objective function is a combination of two costs: agreement between the estimated and 3D normal and strain in the 3D mesh. Instead of computing the alignment between the 3D normal (n_i) and the image normal, we use tangents to the image normal (t_i^1, t_i^2) to make our cost function quadratic:

$$c_{\text{normals}} = \sum_{i \in \text{normals}} (t_i^1 \cdot n_i)^2 + (t_i^2 \cdot n_i)^2$$

However, this problem has local minima: normals that flip across a tangent are unlikely to return — typically causing a single vertex to appear far from the actual surface.

We include the strain term to regularize the solution. Using L as the rest length of a triangle edge and ΔL as the observed change in length, strain is $\frac{\Delta L}{L}$. By penalizing strain, the mesh acts like a set of springs. Strain penalties have been useful in graphics [10] and cloth surface reconstruction [15]. We include strain in the optimization by adding a cost the square of the strain to our objective function.

This method of estimating the depth smoothes the normal field because it computes the minimum error solution to an over constrained problem. Roughly speaking, our cost has two constraints per normal (one normal per triangle) with one degree of freedom per vertex. Although mesh connectivity varies, we typically have more triangles than points — often by nearly a factor of two. As a result, there are roughly four times as many constraints as free variables. Using the lighting model, the estimated 3D normals produce smaller errors in estimated irradiance than the 2D normals computed from the surface texture (figure 5).

4.2 Adjusting for Perspective Effects

We incorporate perspective effects by computing the image locations for points viewed using a hypothetical orthographic camera. Using these modified points, the depth estimation procedure in the previous section can be used on images with perspective effects. To compute the hypothetical orthographic image, we estimate the depth between the scene and the camera and use this depth to adjust the observed (x,y) image locations to their orthographic equivalent. The distance between each point and the camera center is multiplied by a depth adjustment term based on the relative distance between the point and the camera. We assume that the camera center is the middle of the image and define z_i as the estimated depth of point i and $\bar{z}$ as the average depth of the scene:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{d + z_i}{d + \bar{z}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Using this adjustment term, nearby points move closer to the center of the image, while farther points move farther to eliminate perspective effects. We optimize by running gradient descent over the depth value d, using the same cost function as in the previous section.

5 Estimating a Frontal Appearance

In general, a frontal view of a texture is required to compute the surface normals in an arbitrary image. Forsyth [3] considered the case of a single view of a repeating pattern, and showed that 3 views of the repeating pattern (assuming no degeneracies) are enough to determine the frontal view. This method assumes that texture elements are small and therefore approximately planar. We extend this notion to larger textures with repetition in time instead of repetition in space. By observing a video of a moving *non-rigid* texture, we estimate the frontal view by assuming that small regions of the texture are roughly flat. Stitching together the estimated frontal appearance of each small region, we get the frontal appearance of the entire texture.

As a result, we can reconstruct the shape of a deforming surface from a single video. We only require that the user select the texture to track by clicking four points in a single frame of the sequence. (At present, we also require the user to individually select the colors of the screen print pattern in order to build a classifier [14])

Figure 6: *We automatically compute the frontal appearance of a deforming surface — allowing us to compute normals and thus estimate shape (figure 7). The estimated frontal texture is an average of the eight images, each pushed through the appropriate warping. We require the user to manually select the four corners of the texture in a single view (in this case, the upper right image). Then we automatically deform this model to find matches in each of the other frames (a total of eight in this case). The blue mesh in each of the images on the left indicates the match. Following the method outlined in the text, we compute estimated frontal edge lengths, then, treating these lengths as rest-lengths for springs, we minimize energy. Without assuming any form of surface rigidity, this method requires three views of the deforming surface. However, because of degeneracies that arise in practice, we use eight. Degenerate triangles occur when there is a single axis of rotation between views — detecting these degeneracies is described in section 5.1.*

Figure 7: *Using the frontal appearance estimated in figure 6, we use shading and texture cues to estimate the surface normals and consequently the 3D geometry. The original image is shown on the* ***left****. and a re-rendering of the extracted 3D geometry from a new viewpoint is shown on the* ***right****.*

Because we operate under the same assumptions as Forsyth, this procedure requires only 3 views of the deforming surface. However, in practice we typically need more — a degeneracy in any triangle can cause problems in the resulting mesh. In figure 6 we estimate the frontal appearance from 8 views of 128 triangles because no set of three views lacked degeneracies. However, in dynamic video sequences a short segment should be sufficient.

5.1 Implementing Frontal Estimation

Assuming that we have a mesh of static connectivity that tracks the surface over a sequence of frames, we treat each triangle separately. A 3D reconstruction can be computed (with some sign ambiguities) assuming orthography from 3 planar points [6, 13]. We extract sets of three triangles, compute a 3D reconstruction, check the reconstruction for degeneracies by computing the angles between cameras and add the lengths of each edge to separate running edge length lists. After iterating over sets of three views of each triangle and iterating over triangles in the mesh, we have a list of estimated edge lengths for each edge in the mesh. We pick the median value, then minimize a spring energy term to confine the points to a plane.

6 Feature Representation

For arbitrary textures, we use an improved version of the tracking system proposed in [14]. This tracker is a deformable model *without* surface energy. The model is fit hierarchically: first with a rough search for the location of the pattern, then gradient descent to fit a 2 triangle model. To refine, we alternate between subdividing the mesh and using gradient descent to fit. The mesh structure implicitly gives us ambiguous texture normals — for convergence reasons, we use unit quaternions to parameterize the normal.

This representation has several advantages: first, even coarse textures without many distinctive points can be localized. Second, we can refine the transformation as appropriate. At some point, continued refinement is no longer appropriate: the surface texture may not have localizeable texture below a certain scale. (Figure 10)

7 Experiments

We empirically establish the validity of our method by a sequence of experiments of increasing difficulty.

First, we focus on images of custom textures with known frontal appearance. These experiments are done in 3 forms: one with an orthographic image of an object of known geometry (figure 4), one with a perspective image of an object known geometry (figure 4) and finally a perspective image of a surface of unknown geometry (figure 8). In the first two cases, we reconstruct without knowledge of the geometry, then compare the results to our model. In the third case, we use multi-view geometry to reconstruct the shape and compare the results between the methods. In all cases, we generate highly accurate 3D models – errors in 3D geometry are less than three percent.

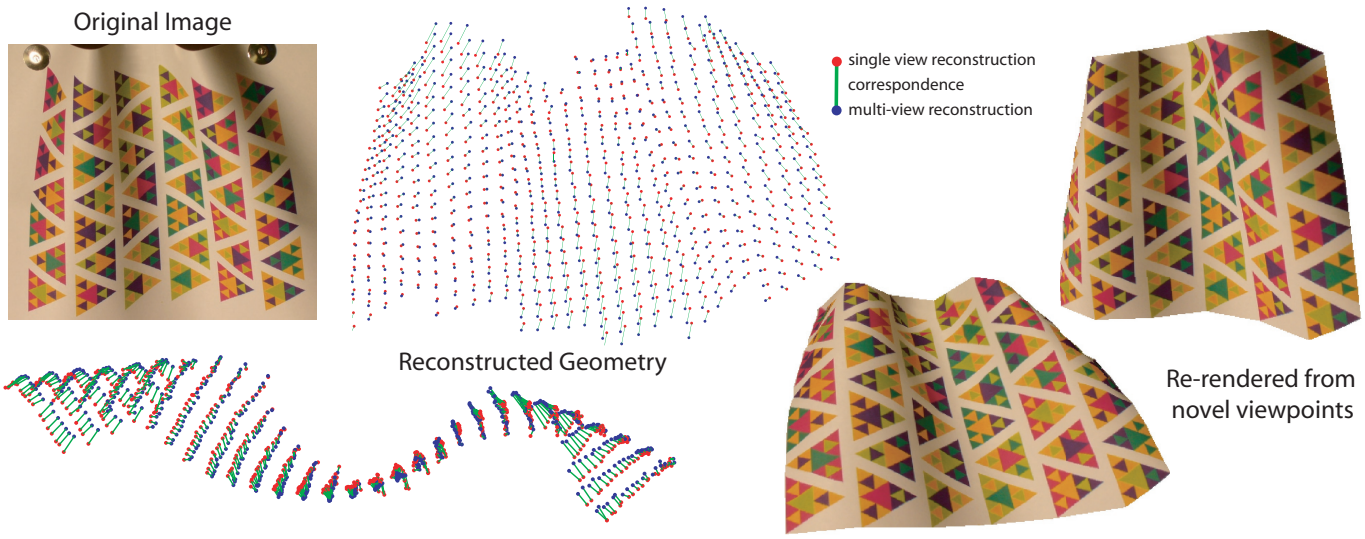

Figure 8: *We demonstrate the quality of the shape reconstruction by recovering the geometry of an unknown object, and compare to a more accurate reconstruction obtained from multi-view geometry. In the reconstructed geometry, the red points are the reconstruction obtained from single view texture and shading while the blue points are obtained using multi-view reconstruction from 4 images. The green lines show the correspondence between the two methods — longer green lines indicate larger discrepancies. Because of the accuracy of multi-view reconstruction (reprojection error of 0.37 pixels and 0.0082 cm or 82 μm), we can reasonably consider this discrepancy to be the error in the single view reconstruction. The average discrepancy is 10.6 pixels or 2.335 mm on an object with a side-length of 7.8 cm. Note that this test is more challenging than the tests in figures 4 and 9 because it also penalizes texture slip — yet results are still accurate to roughly 1 in 40. We use the multi-view calibration method described in [15].*

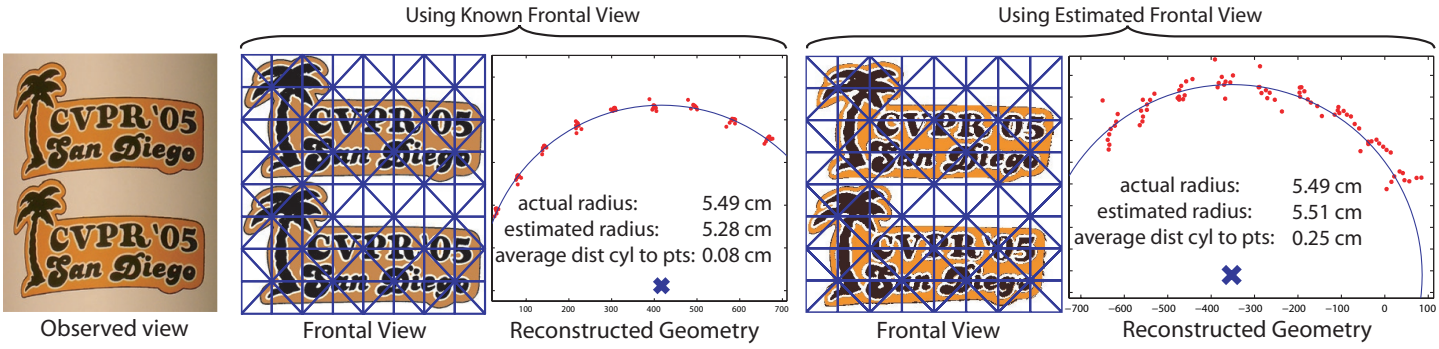

Figure 9: *Again, we compute the shape **from a single view** using texture and shading cues. Instead of requiring a custom pattern to compute the shape (figure 4), we use a supplied frontal view of the texture on the **right** and our estimated frontal view from figure 6 on the **left**. Using the supplied frontal texture the 3D point cloud is more consistent with the cylindrical model than the points generated using the estimated texture — however both textures produce accurate estimates of the radius of the cylinder. We use a deformable model to fit the mesh to a novel image of the pattern taped to a 2L bottle (section 6). This transformation allows us to compute ambiguous normals. A subsequent stage determines the location of the light source (section 3.3), breaks the ambiguity (section 3.2) and uses the normals to produce a 3D mesh (section 4). In images of the reconstructed geometry, we view the 3D point cloud down the axis of the cylinder.*

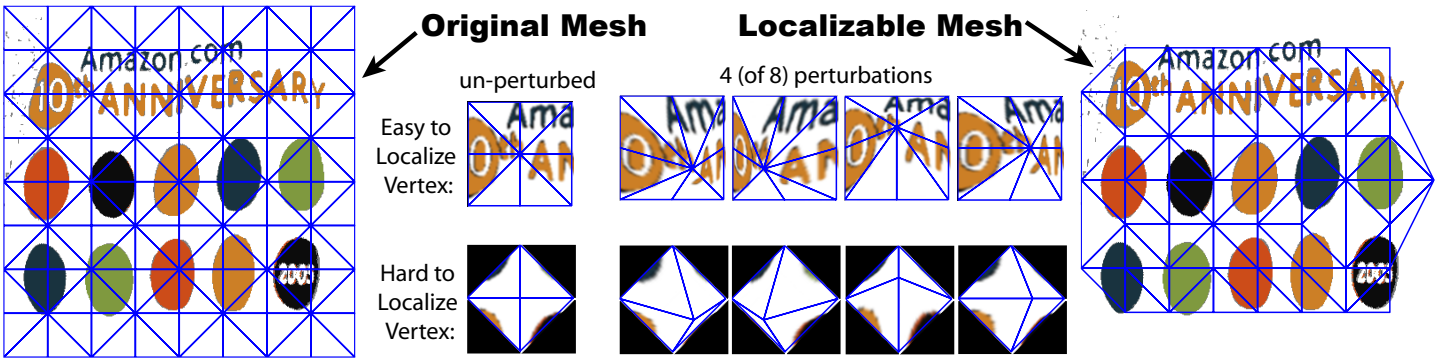

Figure 10: *We generate fine scale meshes by subdividing a coarse mesh, then measure the localizability of the mesh vertices — pruning points that cannot be easily localized. A point is localizable if a perturbation of the location forces a significant change in the transformed image. After throwing away points that are not localizable, we compute a delaunay triangulation of the resulting points and throw away triangles that are ill-conditioned to get a final mesh.*

Second, we consider arbitrary textures. We reconstrut the shape of a known geometry under two circumstances: a user-provided frontal texture, and an estimated frontal texture (figure 9) generated from eight views of the different deformed surfaces (figure 6). Despite the more challenging arbitrary texture, we are still able to estimate geometric parameters within four percent. The estimation of the frontal texture causes only minor increases in fitting error compared to a user supplied frontal texture.

Finally, we reconstruct the frontal appearance of an arbitrary texture from several deformed views (figure 6) and use this frontal texture to estimate the geometry in one of the observed views (figure 7). We test this method visually on a video sequence of a deforming t-shirt by reconstructing the geometry in one view and rendering the result from a alternate view and compare with an actual recording (figure 11).

8 Discussion

We have demonstrated high quality reconstructions from single views. Our method requires a texture estimate, which could be obtained as prior knowledge, by observing a moving object, or possibly from repeated texture. We think that this method of obtaining shape has several applications: including object recognition and scene understanding, graphics, and physical measurement. The quality of our reconstruction provides a middle ground between the flexibility of single cue single view reconstruction and the accuracy of multi-view techniques.

We have considered the estimation of normals to be a process that simply breaks texture ambiguities. However, figure 5 suggests that we could take the process a step further — and produce a normal estimate that minimizes the error between the texture estimate and the shading estimate.

We have put considerable effort into reconstructing shape *without* boundary conditions. However, boundaries provide significant shape cues. A method that identifies boundaries and provides additional constraints would add another information source and probably provide more accurate reconstructions.

References

[1] Yoonsik Choe and R. L. Kashyap. 3-d shape from a shaded and textural surface image. *PAMI*, 1991. 1

[2] D.A. Forsyth. Shape from texture and integrability. In *Int. Conf. on Computer Vision*, 2001. 1

[3] D.A. Forsyth. Shape from texture without boundaries. In *Proc. ECCV*, volume 3, 2002. 1, 5

[4] D.A. Forsyth and A.P. Zisserman. Reflections on shading. *PAMI*, 13(7):671–679, 1991. 1

[5] B.K.P. Horn. Shape from shading : a method for obtaining the shape of a smooth opaque object from one view. Ai tr-232, MIT, 1970. 1

[6] T.S. Huang and C.H. Lee. Motion and structure from orthographic projections. *PAMI*, 1989. 6

[7] A. Lobay and D.A. Forsyth. Recovering shape and irradiance maps from rich dense texton fields. In *CVPR*, 2004. 1

[8] A. Loh and R. Hartley. Shape from non-homogeneous, non-stationary, anisotropic, perspective texture. In *BMVC*, 2005. 1

[9] Julien Pilet, Vincent Lepetit, and Pascal Fua. Real-time non-rigid surface detection. In *CVPR*, 2005. 1

[10] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, 1995. 5

[11] S. Savarese, M. Chen, and P. Perona. Recovering local shape of a mirror surface from reflection of a regular grid. In *ECCV*, 2004. 1

[12] L. Tsap, D. Goldgof, and S. Sarkar. Nonrigid motion analysis based on dynamic refinement of finite element models. *PAMI*, 2000. 1

[13] S. Ullman. *The Interpretation of Visual Motion*. MIT press, 1979. 6

[14] Ryan M White and David Forsyth. Deforming objects provide better camera calibration. Technical Report UCB/EECS-2005-3, EECS Department, University of California, Berkeley, October 03 2005. 3, 5, 6

[15] Ryan M White and David Forsyth. Retexturing single views using texture and shading. Technical Report UCB/EECS-2005-4, EECS Department, University of California, Berkeley, October 03 2005. 5, 7

[16] R. Zhang, P. Tsai, J. Cryer, and M. Shah. Shape from shading: A survey. *PAMI*, 1999. 1

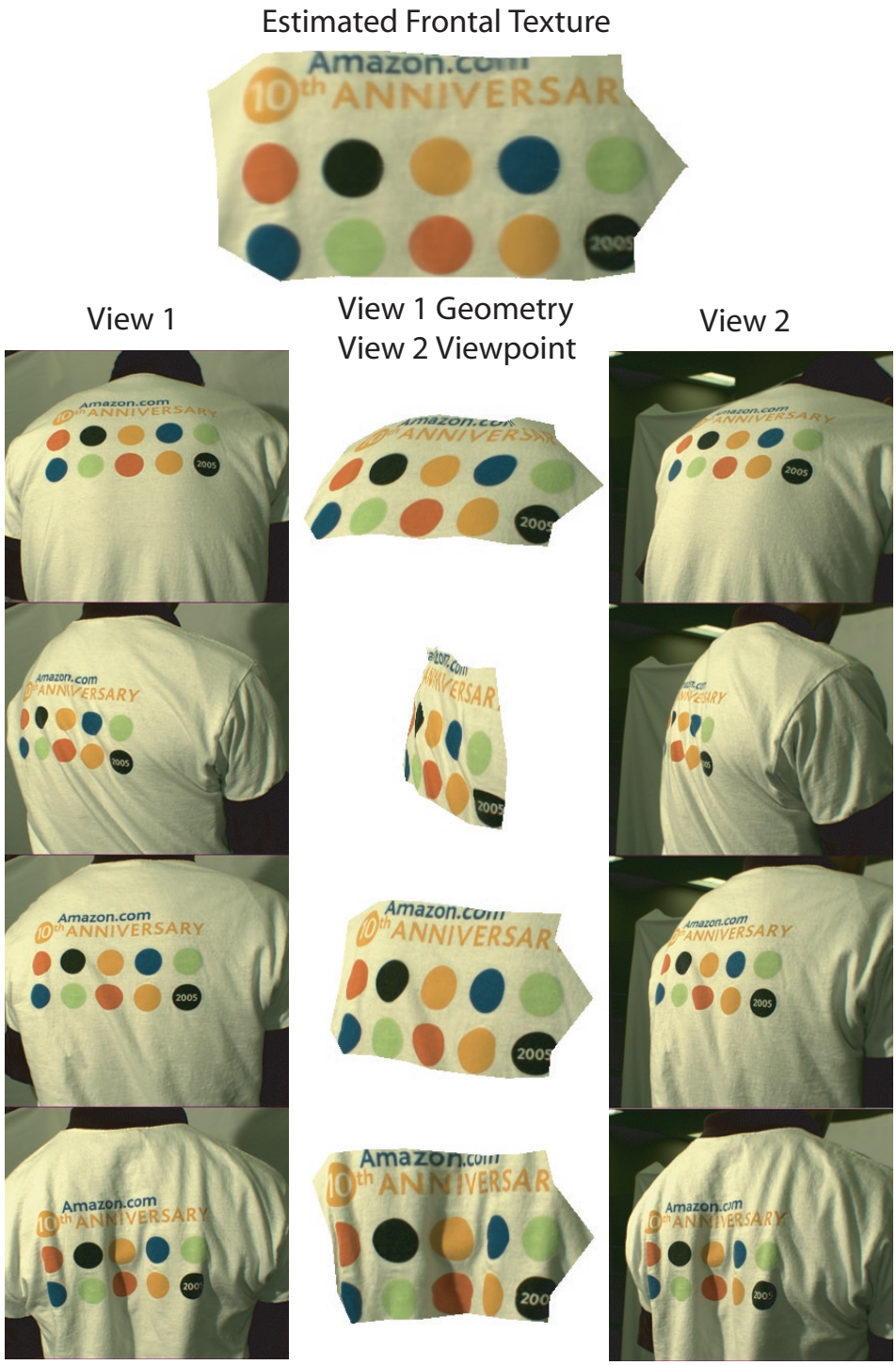

Figure 11: *We use a dynamic sequence of the back of a t-shirt to recover the frontal view (**top**) and subsequently recover surface geometry (**bottom**). In this case, we film the scene from two angles: one that we use to compute the geometry and the frontal view and another to use for comparison. By rendering geometry from the first view from the perspective of the second view, we can visually verify the quality of the result. The shape is only recovered for the localizable textured region of the cloth as described in figure 10. The frontal texture was computed from a total of 24 frames.*