Retexturing Single Views Using Texture and Shading

Ryan White
David Forsyth

Electrical Engineering and Computer Sciences
University of California at Berkeley

Retexturing Single Views Using Texture and Shading

Ryan White, D.A. Forsyth

`{ryanw,daf}@cs.berkeley.edu`

October 3, 2005

Abstract

We present a method for retexturing non-rigid objects from a single viewpoint. Without reconstructing 3D geometry, we create realistic video with shape cues at two scales. At a coarse scale, a track of the deforming surface in 2D allows us to erase the old texture and overwrite it with a new texture. At a fine scale, estimates of the local irradiance provide strong cues of fine scale structure in the actual lighting environment. Computing irradiance from explicit correspondence is difficult and unreliable, so we limit our reconstructions to screen printing — a common printing techniques with a finite number of colors. Our irradiance estimates are computed in a local manner: pixels are classified according to color, then irradiance is computed given the color. We demonstrate results in two situations: on a special shirt designed for easy retexturing and on natural clothing with screen prints. Because of the quality of the results, we believe that this technique has wide applications in special effects and advertising.

1 Overview

We describe a novel image-based rendering technique to retexture fast-moving, deforming objects in video while preserving original lighting. Our method uses simple correspondence reasoning to recover texture coordinates, and colour reasoning to recover a detailed, dense irradiance estimate. Our retextured images have textures that appear to be stable on the surface, at spatial scales that cannot, in fact, be recovered. We believe that our excellent irradiance estimate is a significant component of the sense of shape that is produced.

Retexturing starts with [3], who demonstrate a method based on shape from texture. The method is not notably successful, and does not use irradiance estimates. Fang and Hart show that, in static images, a local shape from shading estimate of normals is sufficient to retexture an image patch satisfactorily [2]: they do not need to estimate irradiance for synthesis because they combine images multiplicatively. The shape estimate is very weak (shape from shading is

notoriously inaccurate [4, 12]), but sufficient for good results. Several methods have been proposed to track nonrigid motion [10, 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 cannot stabilize texture for 3 reasons. First, there are few reliable keypoints in the texture we consider, especially in oblique views. Second, by using keypoints, the method does not track boundaries — and oscillations in the boundary conditions are noticeable in their videos. Third, the rigid smoothing using surface energy makes their method stiff, and limited in scope. In addition, they cannot obtain an irradiance — small errors in their correspondence would make it impossible.

Irradiance estimation is now common in the image-based rendering community [1, 7, 11], usually relying on objects of known geometry and albedo. More recently, Lobay and Forsyth showed that a good irradiance estimate is available from a repeating texture [7].

Applications: Retexturing is a useful and pleasing image level utility. Retexturing clothing has a variety of applications if it can be done cleanly. First, one could sell the advertising space on the back of a sports-player's shirt multiple times — different adverts could be retextured for different television markets. Second, one could change the clothing of figures in legacy images or footage to meet modern tastes.

Conceptual advances: A growing theme of modern computer vision is the number of useful applications that are possible with little or no shape information. We show that high quality images can be rendered in realistic lighting conditions without 3D geometry (figure 2, 5, 9, 10). This can be achieved without high accuracy in localization. Furthermore, we adduce evidence that suggests that good irradiance estimates may be very useful indeed in sustaining a perception of 3D shape.

Procedure: Our method builds a non-parametric regression estimate of irradiance (section 2). We then use quite simple correspondence reasoning to obtain texture coordinates (section 3), from either a frontal view of the texture or a known, engineered pattern. This information is then composited using a new texture map to produce output frames. There is no elastic model of the material and no dynamical model of correspondence — the video is handled frame by frame.

2 Lighting Replacement: Modeling Irradiance

Careful estimates of irradiance are very useful, and appear to create a powerful impression of shape. Their significance for renderings of clothing is probably due to *vignetting*, an effect which occurs when a surface sees less light than it could because other surfaces obstruct the view. The most significant form for our purposes occurs locally, at the bottom of gutters where most incoming light is blocked by the sides of the gutters. This effect appears commonly on clothing and is quite distinctive [5, 6]. It is due to small folds in the cloth forming gutters

Figure 1: *Many common textured articles are made using screen printing — where each color is printed in a separate pass. Often, this method is cheaper than using a full color gamut. Screen printing is widespread: many T-shirts, advertisements, and corporate logos are composed of a small number of solid colors. Recovering irradiance is easier in this setting: correspondence to a frontal view of the pattern is not required. Instead, each color can be detected independently in order to recover irradiance. Because screen print items are composed of large regions of uniform color, they are robust to motion blur.*

and shadows, and could not be represented with a parametric irradiance model unless one had a highly detailed normal map.

However, we do not have and cannot get a detailed normal map or depth map. Furthermore, as we shall see in section 3, the estimates of material coordinates are of limited accuracy. What this means is that it is not wise to estimate irradiance by assuming that the estimate of material coordinates gives albedo (by reference to the original texture map) and from this irradiance can be obtained. A single pixel error in position on the texture map can, for example, mean that an image location produced by a dark patch on the shirt is ascribed to a light patch on the shirt — this would result in a catastrophically inaccurate irradiance estimate. This probably explains why correspondence tracking methods ([9]) don't estimate irradiance or use it to retexture — the correspondence is not pixel accurate, meaning that irradiance estimation would probably fail.

What we do have is an assumption that the clothing pattern is screen-printed, using a small set of highly coloured dyes in regions of constant color. Furthermore, we assume that the pattern is sufficiently complex that colours will fall next to a reference colour (which we call white, though any colour could be used).

In this case, we do not need a formal estimate of irradiance. Instead, at any point in the image, we need an estimate of what the reference colour would look like, if it appeared at this point. By taking this view, we avoid difficulties with scaling between pixel values and radiance, for example. We can obtain this estimate in three steps. First, we build a table that indicates, for each of the dyes in the screen print, what the reference colour looks like in illumination that produces a given set of image R, G and B values from the dye. Second, at

Figure 2: *Lighting cues provide a strong sense of shape — with or without a new texture.* **Left**, *an image from a video sequence taken directly from our video camera.* **Middle**, *we remove the texture items by estimating the irradiance and smoothing.* **Right**, *a retextured image. Notice that irradiance estimates capture shape at two scales: the large folds in the cloth that go through the middle (starting at the red arrow, follow the fold up and to the right) and the finer creases.*

each image pixel, we determine what (if any) dye is present, and use the look-up table to estimate the appearance of the reference colour at that point. Third, we smooth the resulting field.

2.1   Regressing the effects of irradiance

We do not require irradiance in fact. It is sufficient to know what a white patch would look like, when a given dye patch has a given appearance. This information can be obtained by regressing from observations. We build one table for each dye, using the following approach. We use our colour classifier (below) to identify pixels from that dye that lie next to pixels produced by white patches. It is reasonable to assume that, if the pixels are sufficiently close, they undergo the same irradiance. We now have a series of examples, linking image RGB of the dye to image RGB of white. The number of examples is enormous; one might have 10^5 or even 10^6 pixel pairs in a given video. However, some examples may be inconsistent, and some image RGB values may have no entry.

We obtain a consistent entry for each image RGB value that occurs by identifying the mode of the examples. We now have a table with some missing entries (where there were no examples). We use a version of parzen windows to obtain an interpolating smoothing of this table.
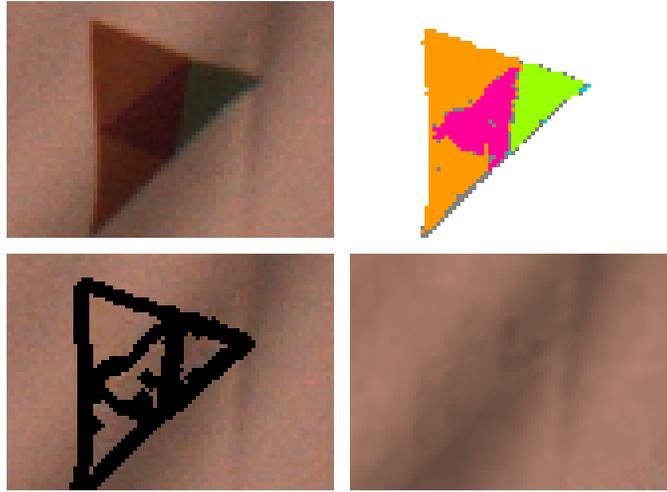
Figure 3: *We estimate lighting for each dye independently, throwing away confusing pixels and boundary regions.* **Upper left**, *an un-altered image of a triangle in our pattern contains strong lighting cues. However, the boundary regions yield conflicting cues: boundary colors change in somewhat unpredictable ways, hiding the strong lighting cues.* **Upper right**, *the output of our color classifier, run on a per pixel basis. As noted, colors at edges can be confusing (indicated in gray) or misclassified (notice the blue pixels at the right tip).* **Lower left**, *after eroding each colored region, we extract lighting cues for each pixel independently. At this stage, the are two problems in our lighting model: gaps in the irradiance estimates and slight chromatic aberrations. In the* **lower right**, *we interpolate regions using a gaussian of varying widths. To smooth out chromatic aberrations, we convert to HSV and heavily smooth both hue and saturation.*

2.2   What dye is present?

We determine what dye is present with a classifier that quantizes the color of each pixel to a pre-determined finite set (determined by the user) based on the pixels component colors. The classifier is a set of one-vs-all logistic regressions on first and second order powers of RGB and HSV. To classify pixels, the maximal response from the array of classifiers is selected, except when all classifiers respond weakly, in which case the pixel is labeled as ambiguous. We do not attempt to classify pixels close to colour boundaries, because blur effects in the camera can lead to classifier errors. At present, we require the user to click on each color to train the classifier, but believe that clustering could remove this step of user intervention.

Figure 4: *A custom color-printed pattern provides more accurate deformation models. Our custom pattern is not screen printed, but follows the same model: a discrete number of colors composed in uniform colored blobs. This pattern creates a grid like structure over the image and allows us to track many more points than the deforming models we use on natural textures. Our transformations are computed locally — triangles are individually detected, colors classified, and correspondences computed. Missing correspondences can cause artifacts in video. On the **left** a single triangle is not detected. We interpolate the location of the triangle using a linear estimate based on the locations of the other triangles (shown as white Xs). In the next frame (**right**), the corresponding triangle is detected with a substantially different position — causing a pop in the video sequence.*

2.3   Interpolating, Smoothing, and Blending

We now take the pool of relevant pixels, determine what dye is present, then present the RGB values to the table indicated by that dye. The result is a representation of what the image would look like at that pixel if the dye had been white. However, this is not available at every pixel — the classifier might refuse to classify, the pixel might be close to a colour boundary and so dangerous to classify. Missing pixels are interpolated using a gaussian weight to sum up nearby pixels, with the variance corresponding to the distance to the nearest pixel. Our irradiance estimates often have slight errors in color. Observing that color variations in lighting tend to be low frequency, we heavily smooth the hue and saturation of the recovered irradiance. (figure 3). Finally, using the domain of the texture map (derived below), we combine our lighting estimate with the original pixels to get a 'blank' surface. We replace pixels in the textured region, blend nearby pixels, then use the original image pixels for the rest of the image (figure 2).
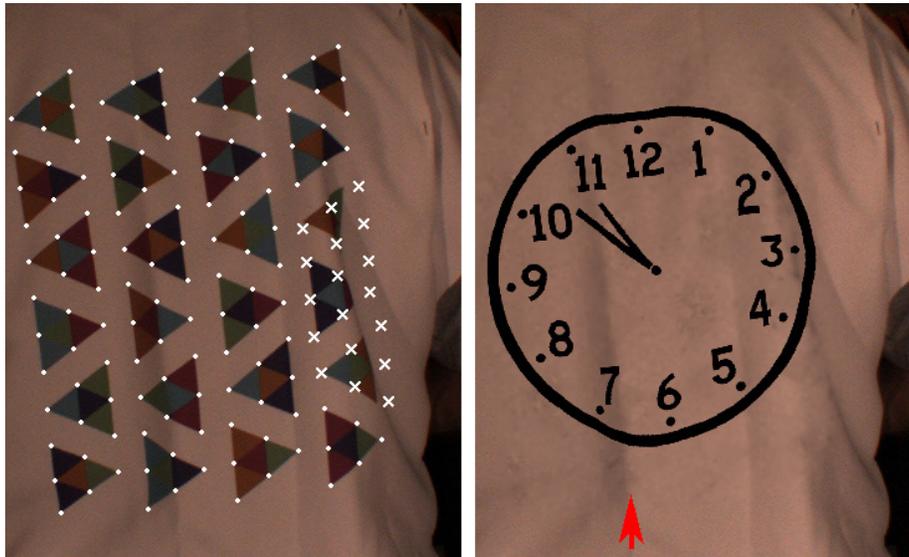
Figure 5: *In some cases, our estimate of the transformation is poor. On the **left**, vertex locations for missed triangles were approximated inaccurately. Because our method **does not** rely on explicit correspondence to compute an irradiance estimate, the reconstructed image on the **right** does not contain obvious artifacts. While the image appears plausible, the re-textured surface is not physically plausible — the texture and lighting cues disagree. Again, we point out that irradiance captures shape at multiple scales: fine creases (follow the red arrow) and larger folds.*

3   Texture Replacement

We need to compute a set of texture coordinates (or, equivalently, material coordinates) from our observed image. There are two possible strategies. First, we could engineer a pattern that worked like a map; it would be easy to determine where a particular colour lies in the map, and so we could use the colour classifier outputs to determine which point on the map corresponds to a particular pixel in the image. Second, we could — at the expense of less precise estimates of texture coordinates — compare the image to a frontal view of the texture. Both methods are successful.

3.1   Using a map

We use a custom printed pattern composed of colored triangles to create high quality texture mappings. The pattern is designed to ensure that neighborhoods are unique and easily discriminable. As a result, our method for computing the planar transformation is essentially local: for each region of image, we compute a map to the source image, then stitch together these neighborhoods to compute

Figure 6: *Specialized patterns make it possible to track very large numbers of points. However, such large numbers are not necessary and can even be slightly detrimental: irradiance estimation becomes difficult because more pixels are classified as edge pixels.*

a complete mapping using nearby transformations to fill in gaps.

Our neighborhoods maps are encoded using uniquely colored triangles: we detect triangles independently, use the color to determine the triangle's identity, then use a deformable model to localize the vertices — creating an accurate transformation between domains. When triangles are detected and correspondences computed properly, this process is very accurate: experiments indicate that typical errors are less than a third of a pixel. Figures 4 and 6 show two sample frames with point locations. We now obtain detailed texture coordinates by a bilinear interpolate within each triangle.

3.2   Using Frontal Appearance

While the pattern detection approach in Section 3.1 is compelling, it is somewhat specific to the custom printed pattern. For arbitrary screen print textures, localization becomes a problem. Instead, we adopt a top-down method to fit the texture: first, search for the rough shape (figure 7) then refine the mapping (figure 8). We use a triangle mesh to represent the mapping, splitting triangles as the mapping is refined.

This method has several advantages. First, that no region of the pattern needs to be particularly discriminative only the pattern as a whole has to be unique. Second, highly oblique views still exhibit the overall shape and can be detected. Third, edges are powerful cues in this model: they provide a constraint that isn't easily recorded using point feature correspondences. Fourth, in contrast to surface energy methods, this method does not have many of the common stiffness properties. However, the disadvantages of a top-down approach are not insignificant: it is not robust to occlusions and subject to local minima. Practically, this means that partial views of the surface may not be

Figure 7: *Our method requires a frontal view of the texture (**top row**) and a target image (**bottom row**). We quantize the colors using a color classifier, then compute a 4×4 color histogram, with separate bins for each color channel. In this case, with three colors (black, orange and yellow), our descriptor is 4×4×3. We visualize this descriptor by reconstructing the colors and normalizing appropriately (**right**). A search for the descriptor in a subsampled version of the target image reveals the closest match (**bottom right**).*

retextured properly.

Estimating an Initial Correspondence: Our method of fitting proceeds in two steps: first, estimate the rough location and scale of the logo, then refine the estimate. Because the quantized image has fewer lighting effects, both stages are performed on the output of our color classifier, not the original image. To detect the rough location of the object we use a color histogram with 16 spatial bins (arranged in a 4×4 grid) and the same number of color bins as colors in the texture, resulting in a histogram of size $4 \times 4 \times \mathcal{C}$. Following other work with histograms [8], we normalize the values, suppress values above 0.2, then renormalize. Using the descriptor from the frontal image as a query, we perform a combinatorial search over scale, location and aspect ratio in a downsampled version of the target image.

Refining the Transformation: Once the rough transformation has been computed, we refine over scales. (figure 8) At each stage in the refinement, we implement the same algorithm: blur the color quantized image (giving each quantized color its own channel), then run gradient descent over the locations of the vertices using the sum of squared distances between the transformed frontal texture and the blurred target image. Our model of the transformation
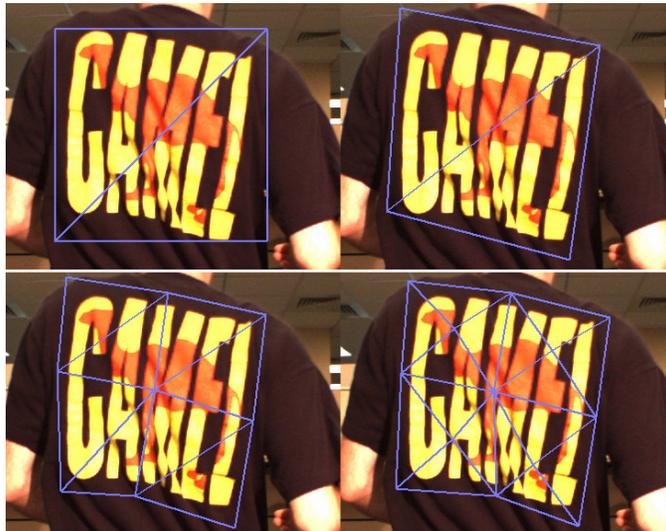
Figure 8: *Our method refines estimates of texture location over scale. Starting with the output of initialization step (figure 7), we have an axis aligned box that corresponds roughly to the location. We use gradient descent on blurred versions of the color quantized image to improve the transformation. Iteratively, we refine the number of vertices (and correspondingly the number of triangles) while reducing the blur to get a better match. Our final model contains only 16 triangles.*

is coarse: we start with a 4 vertex 2 triangle model, then refine to 9 vertices and 8 triangles, and finally 13 points and 16 triangles.

4   Results and Discussion

We have demonstrated the power of retexturing using irradiance on several videos of deforming non-rigid surfaces, including t-shirts and plastic bags. In general, results using a map are better: large numbers of correspondences provide a better replacement texture (figures 2, 9). However, our irradiance estimation is robust — meaning that irradiance estimates are correct even when the texture map is coarse (figure 10). This is important because irradiance estimates are a powerful cue to surface shape. As a result, denser maps do not provide better estimates of irradiance (figure 6). Different background colors do not present a problem: we show results on a shirt with a dark albedo as well (figure 11).

Our results suggest several areas for future work. First, the local method does not interpolate missing triangles well — implying that a hybrid approach may be more effective. Second, our method of interpolating irradiance can be improved: we believe that using texture synthesis could provide more realistic

Figure 9: *Retexturing isn't limited to static images — here we retexture with a ticking clock. (there are 4 frames between each image) On the **left**, the red arrow points to a strong folds that pierces the middle of the clock — giving a strong cure about surface shape.*

results.

We interpret our results to indicate that surface energy terms may be unnecessary for retexturing. Furthermore, a model that reflects the underlying mechanics poorly can result in significant correspondence errors. A 2D elastic model has difficulty managing the very large appearant strains created by folds and occlusions.

References

[1] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 189–198, July 1998.

[2] Hui Fang and John C. Hart. Textureshop: texture synthesis as a photograph editing tool. *ACM Trans. Graph.*, 23(3):354–359, 2004.

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

[4] D.A. Forsyth and A.P. Zisserman. Reflections on shading. *IEEE T. Pattern Analysis and Machine Intelligence*, 13(7):671–679, 1991.

[5] J. Haddon and D.A. Forsyth. Shading primitives. In *Int. Conf. on Computer Vision*, 1997.

[6] J. Haddon and D.A. Forsyth. Shape descriptions from shading primitives. In *European Conference on Computer Vision*, 1998.

Figure 10: *Retexturing is not limited to clothing. This plastic bag exhibits a significantly different type of motion from textiles. Since our model does not include a surface deformation term, our detection method continues to work well. In addition, our lighting model can even account for the highlights due to specular reflection.*

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

[9] Julien Pilet, Vincent Lepetit, and Pascal Fua. Real-time non-rigid surface detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[10] Leonid V. Tsap, Dmitry B. Goldgof, and Sudeep Sarkar. Nonrigid motion analysis based on dynamic refinement of finite element models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(5):526–543, 2000.

[11] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs

Figure 11: *Dark albedos present a challenge for our irradiance estimation. The range of intensities is smaller and there is more noise. Our irradiance estimates are smooth and have a distinctly different appearance. (look closely at the lower left of the logo) However, the rendered surface is also dark, making errors harder to spot.*

from. In Alyn Rockwood, editor, *Siggraph99, Annual Conference Series*, pages 215–224, Los Angeles, 1999. Addison Wesley Longman.

[12] 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.