Shape from Pictures

Scott McCloskey

COMP 766 - Shape in Computer Vision

December 20, 2003

1 Introduction

They say that a picture is worth a thousand words, but nobody has ever explained what those thousand words describe. While we, as humans, have little trouble describing scenes based on photographs, the same problem is very difficult in the computational sense. When, as vision scientists, we want to feel better about our inability to solve a problem, it's common to attribute human ability to some vaguely-defined innate ability or prior experience. Of course this is a valid point in certain cases, but the burden of proof is rarely shouldered to validate such an assertion. Given a picture of some unfamiliar object, humans can usually describe it in several ways: color, size, and ... wait for it ... shape.

With respect to shape, there are fundamental limitations due to the fact that the picture is a 2D projection of a 3D scene. Even with these limitations, there are several cues that we undoubtedly use to eliminate shapes that are inconsistent with the picture. It is the notion of determining shape by photographic consistency that will be discussed in this paper.

Unfortunately, it seems, most attempts to reconstruct shape in this manner choose a one or a small number of cues with which to work. The advantage of these limited notions of consistency is that they lead to elegant methods. The disadvantage is that the algorithm is pigeonholed, or that it requires a large amount of data. This paper will review two methods, both of which use simple notions of consistency to recover shape from a series of photographs.

2 Outline

The paper will proceed as follows: in section 3 we describe the method of shape by space carving as outlined in [2] by Kutulakos ans Seitz. Section 4 outlines, without much editorializing, Seitz and Hertzmann's method of shape by reference using lighting cues. The following section reviews the method and suggests some interesting questions to be investigated. An implementation of this method is described in the next section, followed by some experimental results. The paper closes with some remarks about the algorithm, and potential future work.

3 Shape by Space Carving

3.1 Overview

In this paper, Kutulakos and Seitz outline a method by which a scene's shape can be determined from a sequence of pictures taken from different viewpoints. All of the images are assumed to be perspective projections, as in normal photography.

Each image of the scene is viewed as a set of constraints to be placed on the reconstructed scene. By applying the constraints given in a series of photographs, then, one can carve voxels out of an initial object until the volume is consistent with them. If the initial volume was larger then the true scene, the algorithm results in a consistent volume representing the original scene. Projections of the resulting volume give (possibly new) views of the scene.

It is interesting to note that the algorithm works for any number of images. If the number of images, and thus the number of constraints, is small, the algorithm produces a volume that is likely to be larger then the true scene. The authors prove that the behavior is more meaningful then that. The space carving algorithm is shown to produce the *photo hull*, the union of all consistent volumes that could produce the given images. The photo hull itself is shown to be a consistent volume, and their algorithm is shown to produce it.

3.2 Constraints of Photo-Consistency

The constraints are deceptively simple, hiding the gruesome details of an actual implementation. First of all, images are assumed to have been segmented into foreground and background regions. Given this separation, the three notions of consistency that allow for space carving are as follows:

Point Photo-Consistency - A point on a potential shape is said to be photo-consistent with the photograph from camera position c if it does not project to a background pixel and the color at the projection to c is equal to the point's radiance.

Shape-Radiance Photo-Consistency - A shape and its associated radiance map are said to be photo-consistent with the photograph from c if all of its visible points are photo-consistent with c and the projections of the volume's points span all non-background pixels of the image.

Shape Photo-Consistency - A shape is said to be photo-consistent with a series of photographs from camera positions $c_1, c_2, \ldots c_n$ if there is an assignment of radiance values to the surface so that it is photo-consistent with all positions c_i.

3.3 The Space Carving Algorithm

With these definitions in mind, the space carving algorithm is given as a three step process.

Step 1 : Initialize the volume to anything that contains the true scene.

Step 2 : For each voxel on the surface of the current volume, find the color of the pixel of its projection and the rays connecting the voxel to each camera position c_i. If they are not consistent, remove the voxel from the volume under consideration.

Step 3 : When no more inconsistent voxels are found, the current volume is the photo hull.

The authors go on to given an upper bound for the number of consistency checks that are required by the algorithm. That quantity is NM, where N is the number of images and M is the number of voxels in the original volume.

3.4 An Implementation Nightmare

The simplicity of this three-step algorithm hides quite a bit of complexity that needs to be addressed. As the authors admit, tracking a voxel's visibility is highly non-trivial. The removal of an inconsistent voxel requires that visibility be recalculated for every voxel in every projection.

In response to this, the authors outline an implementation of this algorithm that minimizes the amount of backtracking required by picking a specific order in which voxels are inspected.

Even with this implementation in mind, image acquisition is no small task. In order to perform the projections, the camera location and orientation are required. Moreover, the camera and lens must be well characterized in order to get accurate results. Given the time constraints, it would be difficult to perform any novel testing of this algorithm. In a later paper, [3], Kutulakos outlines a method by which an approximate shape can be recovered based on images with inaccurate information about camera position. While this lowers the bar a bit, both papers mention the use of hardware acceleration for some of their operations. This, of course, adds to the time required to generate an implementation, thus limiting the amount of experimentation that could be performed.

4 Shape by Example Using Orientation Consistency

4.1 The Model

The method outlined in this paper applies to settings where the intensity of a point in the image can be expressed as a function of camera position, light position, and local properties of the surface. That is, the intensity at point p can be expressed as

$$I_p = \rho f(N_p, v, L)$$

where N_p is the normal to the surface at point p, v is the viewing direction, and L represents the incident illumination. The illumination is assumed to be distant, so that incoming rays are parallel to one another. This allows the surface to have any reflectance ρ with respect to, for example, Lambertian or specular reflectance. This model does not, however, deal with non-local effects such as shadowing, reflection or transparency.

4.2 Simplified Orientation Consistency

Given a reflectance function that depends purely on local conditions - and absent any noise - surface points with the same normal will produce image intensities that are equal. With respect to our model and a single image, note that all three of the parameters to the reflectance function are equal for such points.

It is not generally true, however, that surface points with equal image intensities have the same normals. Consider the case where the lighting direction is parallel to the camera direction. If we photographed a sphere under these conditions we would find circular contours of equal intensity. No two points on the same contour, though, would have the same normal - the only thing that they share is the same angle with respect to the incident light rays.

In the simpler version of the algorithm surface normals of a test object are recovered by comparing image intensities with a known reference object under multiple lighting angles. The underlying assumption is that the uncertainty just described can be eliminated by using a sufficiently large number of lighting angles.

The method takes as input a series of n images, each taken with a different lighting angle. Each image is assumed to be the result of an orthographic projection, and must contain both the test and reference object in known locations.

Given an orthographic projection, the use of a sphere as the test object is ideal because it has normals in every direction. If the test object is known a priori to have only normals in the horizontal (vertical) direction then an upright (sideways) cylinder will suffice. Because spheres were used in most test cases, the word 'sphere' will be used interchangeably with 'the reference object'.

In order to find points on the reference object that have the same normal as points on the test object, we measure the distance between points in n-space. Each point p on a surface corresponds to a point in n-space by way of an *observation vector* (OV)

$$V_p = \begin{bmatrix} I_{1,p} \\ I_{2,p} \\ \vdots \\ I_{n,p} \end{bmatrix}$$

where $I_{i,p}$ represents the intensity of the projection of point p in the i^{th} image.

For each point p on the test surface, we compute the value $\|V_p - V_q\|$ for all points q on the reference surface. The normal of the test object at point p is taken to be that of the reference object at the point q that minimizes this distance.

The reader will likely notice that nothing has been said yet about the color of the image. As described, this method deals with monochrome images. The extension suggested by the authors to deal with color is to generate an observation vector that contains three points for each image: the red, green, and blue intensities. Because the distance metric is indifferent to the ordering of dimensions, it doesn't matter now the color data is interleaved.

This method, unfortunately, is of limited use because of the mountain of assumptions upon which it sits. Those assumptions are:

1. The photographs are the result of an orthographic projection. This can be approximated by a camera with a long lens.
2. The lighting must be distant enough that the incoming rays are effectively parallel. The images used by the authors were taken with lighting 10-15 feet away.
3. The lighting directions must not cause shadowing, and must be bright enough to overwhelm inter-reflections.
4. A reference object of the same material and color must be photographed alongside the object of interest.

In order to eliminate some of these requirements, the authors revise the initial algorithm in later sections of the paper.

4.3 Modeling Color Variation

One of the most restrictive of the assumptions needed for the basic method to work is that the reference object be the same color and have the same reflectance function as the test object. In order to handle test objects that have a different color then the reference object, the authors present an extension to the basic method. This extension also handles test objects that aren't solid-colored.

Referring back to our model, we modify the notion of intensity to consider surfaces with changing reflectance

$$I_p = \rho_p f(N_p, v, L)$$

If two points p (on the test object) and q (on the reference sphere) have the same normal then we know that the values of f are equal. Ergo,

$$I_p = \rho_p f(N_q, v, L) = \frac{\rho_p}{\rho_q} I_q$$

If the two points have the same normal, then the intensities in each image differ by a scalar. In order to remove the scale difference, we can normalize the observation vectors. Instead of computing $\|V_p - V_q\|$ for each p and q, we compute $\|\frac{V_p}{\|V_p\|} - \frac{V_q}{\|V_q\|}\|$.

4.4 Modeling Material Variation

While the method in the previous section handles test objects whose color differs from that of the reference object, it still requires the material to be the same type. That is, if the test object is a pure Lambertian reflector then the test object must be, as well. If the test object has both specular and Lambertian components, then so must the reference object. This is unfortunate, especially if we endeavor to characterize a wide range of test objects with a fixed set of reference objects. Moreover, if the test object doesn't have the same reflectance type everywhere, then the simple method would fail. In order to characterize a wide range of test objects - including those that have more then one reflectance type - with a small set of reference objects, we must further revise the original method.

The fundamental assumption underlying this extension is that different reflectance types can be modeled as linear combinations of a small set. The authors cite "strong empirical evidence that a wide variety of reflectance maps may be represented as a linear combination of a small number of basis functions" ([1, page 5]).

Using k basis functions, we revise the model

$$I_p = \sum_{i=1}^{k} \rho_{i,p} f_i(N_p, v, L)$$

The assumption of linearity is a bit of a problem, due to the nonlinear tonescale of digital cameras. This hasn't previously been a consideration because both the test and reference object points are subject to the same non-linearities. Given a reference object that represents a pure specular reflector, though, most of the points will be either very bright (in parts where the surface normal is parallel with the incident light rays) or very dark (elsewhere). High and low intensities are found in the shoulder and toe regions of the tonescale function, respectively. Both the toe and shoulder have significant non-linearities, so it may be necessary to either correct for or prevent (by the use of a camera's raw mode) these effects.

Now we photograph the test object under a variety of lighting directions with a set of k reference objects. Though it is not necessary, we assume that each reference object is homogeneous. The authors, in their examples, use two reflectance objects.

In order to make this method work, it is necessary to register the images of the reference objects. We want them registered so that the point q has the same normal in each of the k images. This is certainly possible for reference objects that have different shape, though it is much simpler if they are the same, and even simpler if they are all spheres.

For each point p on the test image, we can say that the observation vector

$$V_p = \sum_{i=1}^{k} m_{i,p} V_q^i$$

for some $m_{1,p}, m_{2,p}, \ldots m_{k,p}$ where the V_q^i are the observation vectors at corresponding points in the k reference objects. This can be written as a matrix product

$$V_p = W_q m_p$$

where

$$m_p = \begin{bmatrix} m_{1,p} \\ m_{2,p} \\ \vdots \\ m_{k,p} \end{bmatrix}$$

which is called the *material index* and

$$W_q = \left[V_q^1, V_q^2 \ldots V_q^k\right]$$

is the concatenation of the observation vectors from each reference object.

Of course we don't know the material index of each point on the object, so it's necessary to find it in terms of what we already have - observation vectors of the test and reference objects. Using this information, we can derive the material index by the relation

$$m_p = W_q^+ V_p$$

Where the authors use $^+$ to represent the pseudo-inverse of a non-square matrix.

In order to find the surface normals of the test object using this generalized form of orientation consistency, it is necessary to compute the projection of the test object's observation vectors with respect to the basis vectors $V_q^1, V_q^2, \ldots V_q^k$ for all q. Instead of searching for a nearest neighbor, we search for the position of q such that the error of the projection

$$\|W_q m_p - V_p\|^2$$

is minimized. Note that the squaring of the vector length is used because it gives rise to a faster implementation. Of course, minimizing the square of a non-negative number is the same as minimizing that number.

So, for each point p on the test surface and q on the reference surfaces, we compute the error of the projection of p's observation vector relative to the basis formed by the observation vectors of the different reference objects. The reference surface at the point q that minimizes this error is said to have the same normal as the test surface at point p.

This method is extended to color by computing different projections for each color. In essence, for each point p, we find three observation vectors - one each for red, green, and blue. We compute the errors of the projections relative to the basis functions of the corresponding color channel from the reference images. The authors suggest that a single point q be chosen from the reference surfaces that minimizes the sum of the projection errors from each of the three color channels.

4.5 Material Grouping

In order to account for material variation in the way just outlined, we compute a material index in order to find the projection of each observation vector. The material index, though, can also be used to group points on the surface of the test object.

Each material index is a k-vector representing the reflectance of the object with respect to different models of illumination. It's not too much of a stretch, then, to consider points with similar material indexes as being composed of the same material. Because the material indexes are derived from a variety of incident light directions, we have decoupled image intensity due to surface orientation from that due to intrinsic properties of the surface material. Note that the word material, as it is used in this paper, includes color as well as the surface reflectance function.

The authors give an iterative method to find group labels for each point. It starts by running the algorithm described in the last section. Interestingly, the material indexes produced in the execution of this algorithm are discarded. Instead, only the corresponding points from the previous section are used to generate the labellings.

Before presenting the algorithm, it is necessary to introduce the notation. For each point p on the test image, we define $s(p)$ to be the point on the reference surfaces that have the same orientation. Each point p has K (note that K and k are not related, though likely $k < K$) indicators $\lambda_{c,p}$ that discretely represent group membership. That is, the values of the $\lambda_{c,p}$ are 1 or 0, and

$$\sum_{c=1}^{K} \lambda_{c,p} = 1$$

The authors state the problem as finding labels λ, materials m, and correspondences $s(p)$, that minimizes

$$E(\lambda, m, s) = \sum_{p} \sum_{c=1}^{K} \lambda_{c,p} \|V_p - W_{s(p)} m_c\|^2$$

The iterative algorithm, which won't be presented in full detail here, starts by generating material indexes based on the labellings (initially random) and correspondences (from the previous section). Those material indexes are used to update *both* the labellings and the surface correspondences. The material indexes are updated based on these new labellings and surface correspondences, and the cycle continues in this manner until convergence is achieved. The authors state that, since the objective function ($E()$, presumably) is non-increasing, convergence is guaranteed.

Interestingly, the material indexes can be used to construct what the authors refer to as *virtual reference objects*, which are essentially what a reference sphere would look like if it had the properties of a particular material index.

5 Analysis and Avenues of Experimentation

This paper was chosen for this project because, in part, it seems like a simple setup. Camera calibration isn't necessary, and neither is the exact position of the light. The requirements placed on the images aren't particularly

strong, so it should be easy to generate some new tests to confirm that the algorithm works. The authors have made their test data available, but we already know that it works. It can be used to verify an implementation, but isn't particularly interesting beyond that.

5.1 Simplified Orientation Consistency

As the later revisions of the algorithm indicate, there's much more that can be done when all of the assumptions hold. These assumptions seem like a bit of overkill in the simple case where the object has one color and material. So it is logical to ask if we recover shape information with anything less then the full set of assumptions.

In particular, what can we say about the object without a reference object? It is certainly possible, by comparing observation vectors, to find regions of the object's surface that have the same (still unknown) normal. If we assume that the object's surface is smooth then we know that the normal must, at some point, be in the direction of the Z axis (perpendicular to the image plane). Moreover, we know that the normal on the object's rim is equal to the normal of the occluding contour. Can the arrangement of the contours be used to recover estimates of the surface normal? This question will be explored later.

Also, the authors never address the minimum number of images necessary for this algorithm to work. That number is certainly more then one, but it's unclear why more then two (appropriately chosen) lighting directions are required. The test data given by the authors consists of 8 or more images per object. How how much do the surface normals change when the third, fourth, etc. images are considered? While this doesn't prove anything, it might give empirical evidence to suggest the solution.

5.2 Modeling Color Variation

Recall that, in the simple case, the fundamental uncertainty has to do with the fact that points in a single image have equal intensities if their normals form a common angle with the incident light rays. Strictly speaking, the normalization proposed to handle objects of different and varying color introduces another uncertainty. While we intentionally discard scale information in order to eliminate the effect of a points ρ value, we also discard scale information that arises from surface orientation. It is unclear whether this uncertainty is fundamentally different then what was present before, but it certainly doesn't lessen the required number of images.

Whether or not this is a problem, the authors suggest against using this method. Readers are encouraged to handle more complicated test objects using the method described in the following section. They cite poor performance for objects with dark albedos and in the presence of noise. Moreover, since the authors don't give any examples of this method, one must question if it works at all.

Because material variation encompasses color variation, this method isn't really necessary. The authors suggest using the material variation algorithm, so I do not intend to implement and test it.

5.3 Modeling Material Variation

The examples from this section are rather impressive, and are generated using two reference objects. The first, a black snooker ball, is intended to be a pure specular reflector while the second, a pool ball sprayed with gray primer, represents a pure Lambertian reflector. As we take the reference objects to represent the basis reflectance functions, it seems that the authors are interested in an orthogonal basis set, though it doesn't seem to be necessary. It might be the case that the authors want all of the material indexes to have non-negative entries, but this is never stated.

The treatment of color throughout the paper seems tacked on, and makes one wonder if the approaches are chosen because they're right, or because they are easy to implement. In this method, a single point q is chosen from the reference objects to minimize the sum of the errors over the three channels. Given that the digital camera samples the scene with a color filter array, it's not clear that this is the best choice.

While exact specifications vary from one manufacturer to another, most digital camera sensors have pixels that sense either red, green, or blue light. In order to have product specifications that impress their customers, each physical pixel on the sensor corresponds to a (three color) pixel in the output image. Thus, for any pixel in the image, two of the three color values are interpolated from neighboring pixels. While this might be fine for smooth objects, smoothness is not one of the assumptions made by the authors.

With this in mind, then, we might match each point on the test object based on the one color that isn't interpolated. This is certainly inconvenient, if nothing else, because the arrangement of the color filter array isn't passed along with the image. It's even more inconvenient when one considers that the resolution of the reference objects is effectively reduced by a factor of three.

Given that the image is noisy to some extent, the case can be made to find potentially different positions on the reference object for each of the three channels. The final output of the algorithm could be the average of the normals at these three positions, in hopes of averaging noise. The differences between the three could also provide a sanity check of sorts - if the three channels match points with very different normals then it's probably the case that more test images are necessary.

Unfortunately this is difficult to test, as there is no established ground truth. The authors present renderings of their surfaces next to renderings that come from a laser scanner, but it seems that the scanner's performance is rather limited. Without established ground truth, it is impossible to quantitatively assess the algorithm's performance.

5.4 Material Grouping

While the motivation for this method is appealing, the final result is somewhat unsatisfying. In particular, the authors suggest that using random labellings as a starting point produces better results then using material indexes generated by the point matching algorithm. What's more alarming is that nothing is taken to be fixed; the correspondences, labels, and material indexes are all updated in each iteration of the labeling. The authors do not address how much the correspondences change in the course of this algorithm. Are the correspondences at the time of convergence any better then the ones that we started with?

The authors refer to the objective function without actually defining it, though it seems that they are referring to the function $E(\lambda, m, s)$. We're assured that the function is non-increasing with respect to the iterative method that they outline, but this is unsatisfactory. When do we know that we've reached the point of convergence, and that we have an optimal labeling? Unless the objective function is also shown to be nicely behaved - i.e. it doesn't have long stretches between improvements - we can't know when to stop iterating.

The algorithm is presented as a way to form K groups of points, each representing surface points with similar qualities. It does not, however, say how the value of K is found. The groupings might appear artificial if the value of K were less then the actual number of different-looking points, so it seems that this value must be chosen by hand.

The example of the cat shows that points with very different appearances might be assigned to an outlier group if there are a small number of them. This brings up the point that the groups might not be allocated in the same way that a person might group them. Because of the mathematical model, two groups of very dissimilar appearance might be lumped together if they have limited representation, while two groups might be formed of very similar points when there are many of them.

6 My Implementation

6.1 Simplified Orientation Consistency

While this algorithm is very simple to explain - it boils down to finding a nearest neighbor in n-space - the implementation has to be somewhat involved in order for it to execute in a reasonable amount of time.

As much as I like Matlab, it is particularly unsuited to handle this problem. In order to write Matlab code that is reasonably fast it is necessary to remove for loops. Unfortunately this often comes at the expense of redundant storage in memory, and this problem is no exception. Because of the large number of comparisons that must be performed, the memory required to find corresponding points is unreasonably large. For the test data provided by the authors, the loop-free Matlab implementation would require around 2 exabytes (2 million gigabytes) of memory. This might be practical in a few decades, but our semester isn't quite that long.

While it would be possible to minimize looping by computing correspondences in chunks whose size matches the amount of physical memory in the system, it's not worth the effort. The basic step of the algorithm - computing distance between points in n-space - is simple enough to perform in C. Matlab is still useful to read the image file formats, generate the observation vectors, and calculate the normals of the reference object.

Before considering the images of the test object, Matlab code reads the (registered) images of the reference object. Because they're spheres, it is simple to find surface normals given (as we are) a mask that indicates which image points represent the sphere. We know that the outward normal at any point on the sphere is just a continuation of the vector connecting the center of the sphere to the point in question. Since we have an orthographic projection, we can find the height of the reference sphere at any (x, y) point by the relation $z = \sqrt{r^2 - x^2 - y^2}$, where the radius r is half the width of the mask at its widest point. We can assume that the center of the circle is the origin, and thus a unit outward normal is equal to

$$N(x, y) = \frac{(x, y, z)}{\sqrt{x^2 + y^2 + z^2}} = \frac{(x, y, z)}{r}$$

After the normals of the reference object have been characterized, the observation vectors of the sphere are generated and stored in a 2D array. In order to speed up the nearest neighbor search, the vectors are sorted based on their first entries. Once generated and sorted, the observation vectors are written to a binary file for use by the C code that performs the nearest neighbor match.

Still in Matlab, the observation vectors of the test image are generated and written to another binary file.

These two binary files are read in to the C program, which performs the nearest neighbor match. In order to shorten the execution time, an observation vector from the test object is only compared to a subset of the reference object's observation vectors. Since the OVs of the reference object have been sorted, memory locality (and thus speed) are high for this match. The size of the subset is determined empirically.

Once the matching is performed, the C code outputs a binary file containing the index of the reference observation vector that best matches each test OV. The Matlab code reads this in and generates a map of the test object's surface normals, returning it to the caller.

6.2 Generalized Orientation Consistency

The code for this version of the algorithm is significantly more complicated then the base case. The basic step involves two matrix-vector multiplications in addition to finding distance in k-space. This only increases the amount of memory that would be required for the non-looping Matlab solution, so we stick with a hybrid approach.

The Matlab pre-processing gathers the observation vectors from the test image and finds the W and W^+ matrices for each point on the reference objects. These values are written to binary files that the C program uses to compute the best matches. Because the test object doesn't have the same color as the reference objects, it is no longer possible to restrict our search in the simplistic manner described above.

We might be tempted to restrict our search to points on the reference objects in the neighborhood containing the corresponding point of a neighbor of the test point. This is ill-advised, though, as the authors have made no assumptions about the object's smoothness. As such, the search inspects all of the points on the reference objects.

As before, the C program outputs an index for each point indicating which point on the reference objects is the best match. Matlab reads this information and generates the normals at each location, which are returned to the caller.

7 Examples and Experiments

My attempt to generate new image inputs seemed doomed from the start. The first problem was finding suitable reference objects. In the spirit of the paper, I tried to reproduce the reference objects that the authors use in the general algorithm: a black, specular sphere and a gray, Lambertian one. I purchased two street hockey balls and cans of glossy black and matte gray spray paint. In what would be an omen for the rest of this attempt, it started to rain on the balls while the paint dried outside. This resulted in some significant imperfections on both spheres, and it only got worse. The black paint seemed never to dry, so transporting it was troublesome; the ball would roll around and stick to everything. When the time came to take the pictures, the paint was marred by fingerprints, cardboard, and some gray paint from the other sphere.

A larger, and more fundamental, problem has to do with the assumption that no inter-reflections are present. This is particularly difficult with a shiny black ball - marred though it may have been. When placed near the test object there was an obvious reflection of it in the black paint. This was temporarily solved by using a black divider to block the reflections, but this further limited the range of light directions that could be used.

Illumination proved to be a problem, as well. The algorithm calls for a single, distant light source so that incoming rays are essentially parallel. The light that I brought proved to be too dim for these purposes - 30 seconds would not have been long enough to give a proper exposure. Based on availability, I used an overhead projector to illuminate the scene. While this was certainly bright enough, it created other reflection problems, particularly with the floor. Moreover, the projector was too large to move up and down, limiting the range of light directions to one direction.

Most of these problems seem to have resulted from a lack of foresight. Given a large room and one studio-quality light, this shouldn't be too difficult. Unfortunately, there is not enough time to give it another try.

7.1 Simplified Orientation Consistency

The implementation was tested with the *greenbottle* test data given by the authors. It results in a reasonable looking vector field that is in line with expectations. Without an established ground truth it is difficult to estimate the real accuracy in a scientific way.

Generally, though, there were two problem areas. The green paint on the bottle's cap had started to fall off, so significant errors are found in this area. Also, some of the normals around the rim of the object don't look right. This may have resulted from my (potential) misinterpretation of the mask image. Pixels in the mask images were valued in the range [0,255], and I interpreted any location with a non-zero value to be on the object.

Qualitatively, of course, a vector plot of the surface normals isn't particularly helpful. We're used to seeing intensity images of bottles, not vector fields, so it's helpful to render the surface under different lighting conditions. Refer to the demo section to judge for yourself.

In order to get an impression of impact of the number of test images, the surface normals were found based on subsets of the 8 test images. Figure 1 shows a plot of the root mean squared error of the surface normals as a function of the number of images in the test set. The actual surface normals are unknown, so the surface normals for the full set of images is taken to be correct. As a result, the mean squared error for the full test set is 0.

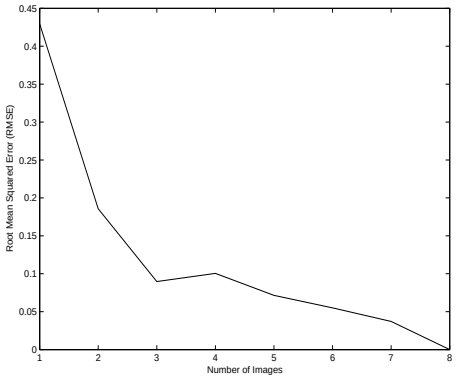

Figure 1: Error as a function of the number of images.

While the metric is somewhat suspect, the rate of change is informative. Specifically, while there is a significant drop in error when the second image is added to the test set, the normals are significantly changed with the consideration of additional images. In the continuous case two properly-chosen lighting angles should suffice, so I wonder if the discretization impacts the number of images that are required.

7.2 Generalized Orientation Consistency

My implementation of this method was tested with both the *fish* and *cat* image sets given by the authors. In addition to taking a long time to find - about 10 hours for the cat - the results aren't terribly impressive. The plot of the vector field indicates discontinuities where I don't expect them - the eyes of the cat, for instance. Of course, since there is no ground truth available, it's hard to say anything more then this. You can judge for yourself by looking at the demonstrations. The renderings of the cat and fish surfaces aren't particularly impressive, and look worse then those in the paper. I wonder if their rendering method smooths the surface at all, or if my implementation is really that much worse.

All of the avenues of experimentation are blocked because we don't know enough about the test surface or the camera that took the pictures.

7.3 Staring at Contours of Equal Normals

It's important to note that, when I say 'contour', it's a bit misleading. While the surface would certainly have contours, it's unlikely that any two observation vectors will be exactly equal after discretization and noise are considered. So we use the word contour to mean segments of the object's surface that have similar normals. The degree of similarity - measured by the norm of the difference in observation vectors - that's allowed within a single contour is taken to be a parameter.

Before even attempting to generate the contours, it was necessary to consider how they might be displayed. Matlab's `contour` function is tempting, but it doesn't behave nicely when the number of contours is large, or when the contours are wide. Because of this, a false-color visualization was chosen. For each point on the test object we would be given a number representing the contour to which the point belongs. The challenge is to falsely color the image in such a way that adjacent contours look different. Lacking the time (and interest) to construct a color mapping that maximized contrast between adjacent contours, I decided to fall back on randomness. The color map

was generated as a random permutation - one for each color channel - of a 0-1 ramp. While this doesn't always produce a nice color map - adjacent contours might be assigned a similar color - you can just keep trying until, by chance, things look right.

7.3.1 A Shot in the Dark

The first attempt at generating contours took a naive approach. A piece of Matlab code loaded all of the images and generated the observation vectors for each point on the test surface. Those observation vectors were written to a binary file to be used by a C program.

That C program started by assuming that the first point (as represented by its observation vector) was an exemplar of a contour. For each subsequent point, then, the observation vector was compared to the current collection of exemplars. If the norm of the difference vector was smaller then a certain threshold, the point was said to belong to the contour for which the norm was minimized. If the norm exceeded that threshold, then the point was said to be an exemplar of a new contour. In this implementation the user controls the threshold and thus, indirectly, the number of contours.

Not surprisingly, this method has a number of problems. The largest of these is that points don't necessarily belong the the contour whose exemplar's observation vector is closest to its own. Because there is no second pass through the observation vectors, a point is grouped based on its observation vector's similarity to the exemplars that exist *at the time it is analyzed.*

The manifestation of this problem can be seen in figure 2, where the division between the brown and peach-colored contours appears as a vertical line. This appears as a vertical rather then horizontal demarcation because Matlab organizes memory in column-major order. As a result of this the observation vectors are written to disk in column-major order, and the C program visits them in the same order.

The reason that this is such an offensive problem is that points on either side of a contour boundary might have very similar normals (if the surface is locally smooth). In fact, the points at the contour boundary often have observation vectors (and thus normals) that are more like the nearest point in a different contour.

Figure 2: A first stab at iso-normal contours

7.3.2 Getting Warmer...

In a superficial way, this problem can be solved by randomizing the ordering of the observation vectors in the binary file that the C program takes as input. This certainly eliminates the vertical lines in the contour visualization (see figure 3), but doesn't really solve the problem. Each observation vector is still compared to a potentially incomplete set of exemplars, so there is no guarantee that the point is assigned to the contour whose exemplar is closest to its observation vector. This manifests itself in the example as points within one contour which are completely surrounded by members of other contours.

Also, after a bit of experimentation it became annoying that the number of contours was determined indirectly from the distance threshold.

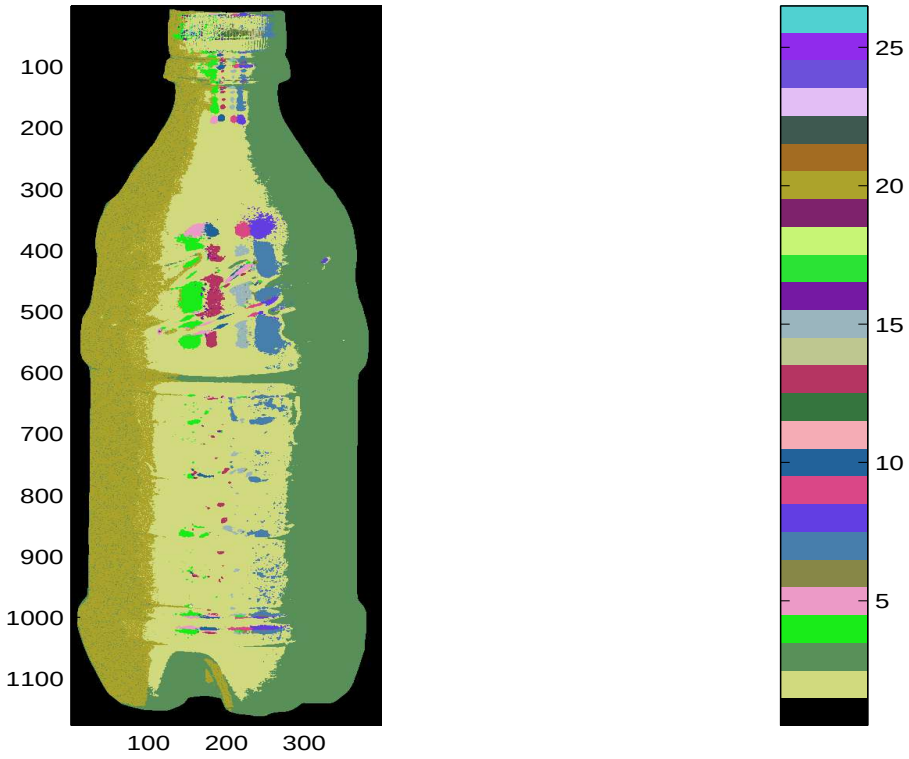

Figure 3: Iso-normal contours using random visitation

7.3.3 A Better Contour Generator

In order to get around both of these problems, I decided to pick exemplars in a more principled manner, before running the C program. Moreover, the number of contours, rather then the maximum allowable distance, was given as a parameter. This presents a bit of a problem, if only because there are so many choices for the exemplars. Ideally, if we knew the lighting directions, we might generate synthetic observation vectors for the specified number of orientations. We're not assuming to know the lighting direction, though, so we're left to choose from our set of test observation vectors.

Remembering the analysis of the material labeling section, this implies a fundamental question: do we choose the exemplars in proportion to representation, or based on the range of variation?

If we really wanted to do either, we'd pick the exemplars to minimize a function in the style of $E()$ from section 4.5. In order to get quick results, though, a more heuristic approach was taken: sort the observation vectors based on their first entries, and choose the specified number of equally-spaced vectors as exemplars. Note that, by "equally-spaced", I am referring to equally spaced indexes, not observation vectors.

The final result looks reasonable enough, and is shown in figure 4.

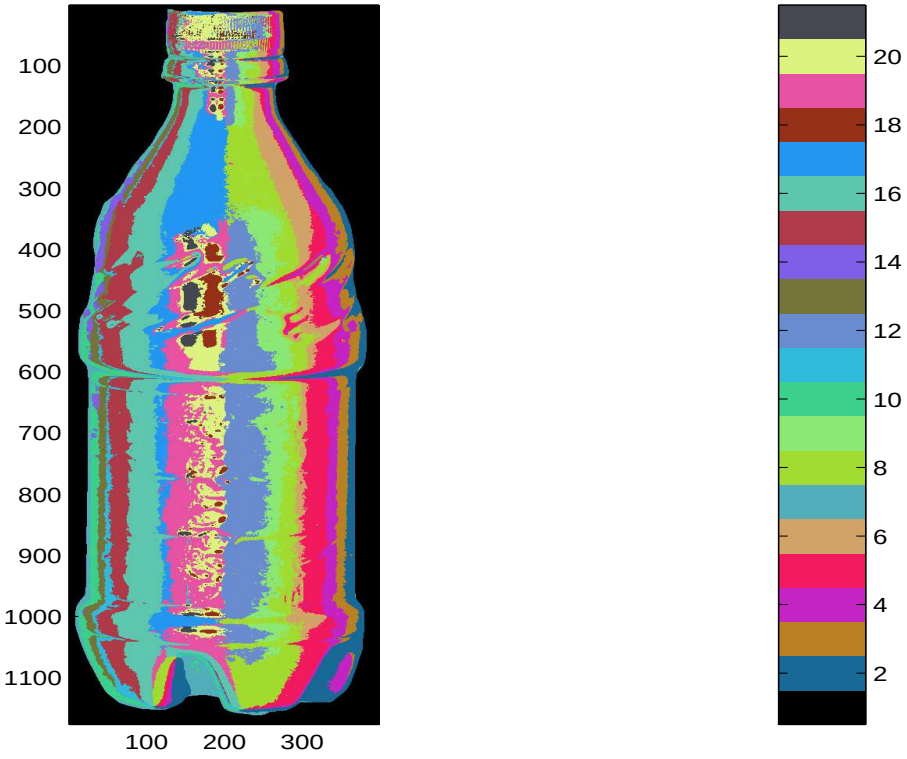

Figure 4: Iso-normal contours using pre-selected exemplars

7.3.4 Analysis of Iso-Normal Contours

After staring at the iso-normal contour visualizations of the bottle for a while, I have been unable to say anything meaningful about shape without presupposing something unpleasant.

In the spirit of thinking out loud (or, in this case, on paper), it seems that we should be able to determine surface normals by iterating on the number of contours. Suppose that at some time we have n contours and an estimate of the average normal of the points in each contour. If we generate a map with $n + 1$ contours, we could say that the average normal in each contour is equal to the average of the members' normals from the previous iteration. The average would almost certainly have to be with respect to spherical coordinates, as the average of Euclidean coordinates $(-1, 0, 0)$ and $(1, 0, 0)$ wouldn't give us what we want.

The - or, at least, one - problem with this idea is that it's hard to come up with a starting point. As previously mentioned, we can determine the surface normals along the rim if we assume that the surface is smooth. Such points can be grouped in contours, but what do we do with interior points?

A mathematical model for the true average normal of a contour might help define this starting point, but is outside the scope of this project. It seem, however, to be potentially fruitful avenue of investigation.

8 Demos

There are two ways that you can view the output of these methods. The vector fields given directly by my implementation are best viewed in Matlab, so that you can zoom in on places of interest. In order to view these, open a shell, go to `~scott/comp766/project/demo` and run Matlab. Type "show bottle", "show cat" or "show fish" (without the quotes) to view the results for the three test cases given by the authors. Maximize the figure and zoom around to some of the interesting locations. Note that the vectors are only the X and Y components of the normals. Since each is a unit normal, the component in the Z direction is left up to your imagination - normals in the direction of the camera are shown as points.

A more natural way to view the results is in the form of a rendering. Renderings of the three objects are available on the web at `www.cim.mcgill.ca/~scott/766/project`. Renderings are shown for light vectors in 5 lighting directions: one from the direction of the camera and the other four from directions 45 degrees left, right, above and below the camera direction. Note that these renderings are simplistic, as they do not account for shadowing.

9 Conclusions

At first, because the method is easily explained, it seems to have wide applicability. As I discovered when attempting to generate my own test images, however, the requirement that intensity be locally determined is quite restrictive. The general consistency model seems to have questionable results, though a ground truth is necessary to cast serious dispersions on its performance.

As the authors suggest in their closing remarks, it might be more fruitful to render simulated reference objects instead of requiring their presence in the original images. This only requires that we know the direction of incident illumination, and lessens the need for a controlled environment.

The iso-normal contours seem to merit further analysis, though it is outside the scope of this project.

On the whole, there is something unsatisfying about the paper and this report. In particular, the fundamental questions - how many images are necessary? what lighting directions reduce ambiguity the most? - are still unanswered. The appropriate use of color information is still unclear, as well.

References

[1] A. Hertzmann and S. M. Seitz. *Shape and Materials by Example: A Photometric Stereo Approach.* Proc. IEEE CVPR 2003. Madison, Wisconsin. June 18-20, 2003.

[2] K. N. Kutulakos and S. M. Seitz *A Theory of Shape by Space Carving.* Proc. 7^th IEEE International Conference on Computer Vision. Corfu, Greece. pp 307-314. 1999.

[3] K. N. Kutulakos *Approximate N-View Stereo.* Proc. 6^th European Conference on Computer Vision. Dublin, Ireland. pp 67-83. 2000.