

Stochastic Mesh-Based Multiview Reconstruction

John Isidoro
Computer Science Department,
Boston University, Boston, MA, 02215

Stan Sclaroff
Computer Science Department,
Boston University, Boston, MA, 02215

E-mail: (jisidoro, sclaroff)@cs.bu.edu

Abstract

A method for reconstruction of 3D polygonal models from multiple views is presented. The method uses sampling techniques to construct a texture-mapped semi-regular polygonal mesh of the object in question. Given a set of views and segmentation of the object in each view, constructive solid geometry is used to build a visual hull from silhouette prisms. The resulting polygonal mesh is simplified and subdivided to produce a semi-regular mesh. Regions of model fit inaccuracy are found by projecting the reference images onto the mesh from different views. The resulting error images for each view are used to compute a probability density function, and several points are sampled from it. Along the epipolar lines corresponding to these sampled points, photometric consistency is evaluated. The mesh surface is then pulled towards the regions of higher photometric consistency using free-form deformations. This sampling-based approach produces a photometrically consistent solution in much less time than possible with previous multi-view algorithms given arbitrary camera placement.

1 Introduction

Algorithms for automatic reconstruction of 3D models from multiple views have many practical applications, such as virtual reality, movie special effects, computer aided design, image compression, *etc.* The ability to create a 3D representation of real world objects is also useful for creating models for computer games. In addition to this, recovered models could be used as inputs to vision algorithms for object recognition, object tracking, or even human motion capture. Such 3D models are also useful in robotic tasks such as navigation and planning.

Texture-mapped polygonal meshes are the *de facto* standard for object representation when using computer graphics hardware. Meshes can be rendered extremely quickly in comparison to other representations; most modern consumer graphics boards can render millions of triangles per second. Moreover, meshes are generally more compact than other general 3D object representations (*e.g.*, volume data), and there are many “level of detail” schemes that can be used for progressive mesh transmission. Obtaining a polygonal mesh as the direct output of a 3D reconstruction algorithm therefore offers a number of important benefits.

In this paper, a novel method for reconstruction of 3D models from multiple views is presented. Stochastic techniques are used to construct a texture-mapped semi-regular polygonal mesh of the object in question. Given a set of views and segmentation of the desired object in each view, constructive solid geometry (CSG) is used to build a visual hull from silhouette prisms. In contrast with previous techniques, this formulation uses sampling methods in conjunction with free-form deformations (FFD) to produce a photometrically consistent polygonal mesh. Because of this the system is able to produce a photometrically consistent solution in much less time than possible with previous algorithms.

2 Related Work

One of the classic problems of machine vision is the reconstruction of a 3D model from a collection of images of an object. There have been a number of different classes of techniques proposed for 3D model reconstruction, each with its own set of constraints on the input images.

Structure from motion techniques usually rely on images being captured close together in time or space. Because the images (hopefully) do not change large amounts from frame to frame, corresponding features can be tracked from frame to frame. A 3D model is then reconstructed given rigidity constraints. These techniques work well given a moving object, moving camera, or many cameras spaced close together. They generally break down under non-rigid object motion.

Stereo techniques rely on cameras placed close together in order to find correspondences. These techniques work quite well, but they are only capable of producing a 2.5D model (images with per-pixel depth). A stereo-based model reconstruction technique that is closely-related to the one in this paper is due to Fua, *et al.* [8]. A stereo algorithm is used to build a polygonal mesh which is further refined by moving vertices. However, their solution yields only a 2.5D model.

Another class of techniques relies on additional equipment to build 3D models. Laser range finders, structured light projection, and other specialized equipment can be used to find a set of 3D points which lie on the surface of the object. After this, a polygonal mesh is fitted to the points. Usually this mesh has much more detail than is required and a variety of mesh post-processing steps are used to simplify the mesh. Although the source of the data is fundamentally different, many of the mesh post-processing steps are directly-related to the technique presented in this paper.

An alternative approach to multiview reconstruction is to try to fit a deformable shape model to the data. Many techniques have been proposed [3, 16, 18, 25]. Most of these techniques involve a physically-motivated framework where a penalty term is associated with deviation from an initial mesh. These penalty terms usually penalize sharp edges in the mesh. The problem with having this type of prior on shape deformation is that is that the solutions they push the solution toward are not necessarily desirable. According to the theory of space carving [12], the actual surface lies strictly inside of the visual hull, and what is desired is a holonomic constraint that keeps the reconstructed surface inside the visual hull at all times.

The technique presented in this paper falls into the class of multi-view reconstruction techniques that allow for cameras to be placed far apart, with the caveat that the cameras' intrinsic and extrinsic calibration parameters are known beforehand. Some of the most prominent recent work in this class reconstructs a model of the object using voxel based techniques [4, 7, 13, 21, 22, 23]. These techniques work by starting from an initial lattice of voxels, and carving away those which do not satisfy photometric consistency. The problem of occlusion requires these techniques to make multiple passes over the voxel grid in order to carve away voxels. As a result, these techniques take a large amount of time to run (most papers cite close to an hour for a $256 \times 256 \times 256$ grid). An exception to this is the original voxel coloring paper by Seitz and Dyer [20] and a subsequent paper by Prock and Dyer [19]. These papers simplified the visibility problem by limiting the camera setup to require the object to lie outside of the convex hull of the cameras' centers of projection. This *ordinal visibility constraint*, allows the voxel grid to be processed in a single pass instead of multiple passes for arbitrary camera setups allowing for solution times in minutes (and in some cases seconds) instead of hours. Note that the technique presented this paper does not require this restriction on camera placement. The only requirement it has is that every point on the surface of the object to be reconstructed should be visible in at least two of the camera views.

Another advantage of this technique over voxel based techniques has to do with computational efficiency. A sampling-based polygonal hull technique is employed to avoid the expense of checking photometric consistency at every discretized point in space multiple times. Photometric consistency is evaluated on a series of stochastically generated 1D strands throughout space, thus reducing the amount of computation.

Another problem common to voxel techniques is that a single mis-carved voxel due to image noise or specular highlights can produce a chain reaction of mis-carved voxels that cuts a spurious hole in the model. This problem is amplified by the fact that photo-consistency methods compare pixels from multiple images, each one having independent noise. To help alleviate this problem, a probabilistic framework can be used to assign an existence probability to each voxel [2, 5]. Another possible solution to the mis-carving chain reaction is to search a neighborhood of pixels around a re-projected voxel's position for a color-consistent one [11]. These approaches offer promising results; however, they are generally much more computationally expensive than basic binary voxel carving. The solution presented in this paper to the mis-carving problem is very different.

Instead of carving away voxels to estimate the correct object shape, a polygonal visual hull is deformed. The deformations applied maximize color consistency along a set of sampled epipolar lines aka *strands*. Strands which do not have an adequate color consistency solution are rejected and are not used in the deformation stage. Because of this, over-deformation usually does not occur. In the cases when it does occur, the next iteration of the algorithm can deform the region back outward. The technique presented does not have the disadvantage of irreversibly carving away portions of the model, that most voxel carving techniques have.

Another related approach due to Zhang and Seitz[26] starts with a polygonal mesh for the object's bounding ellipsoid,

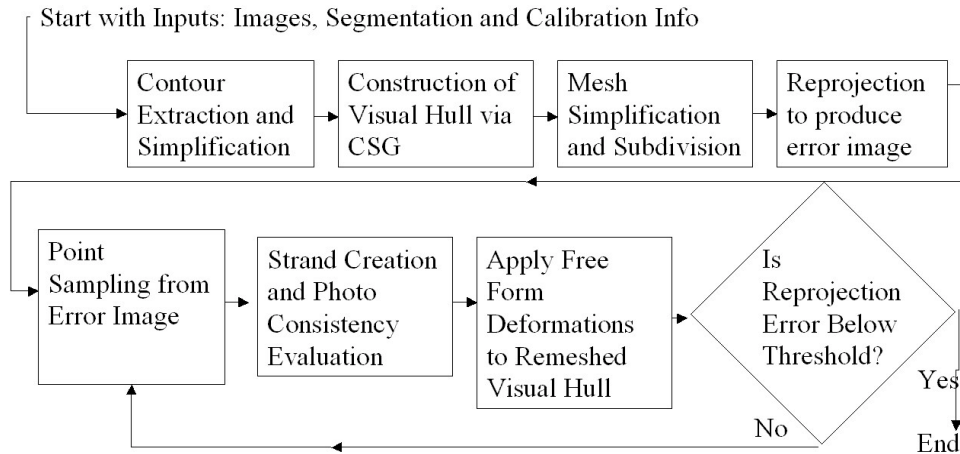


Figure 1. System Overview Block Diagram

and then performs a series of vertex insertion and edge-collapses in order to more closely match the appearance of the object in multiple views. In each iteration, the reference images are projected onto the object, and the image differences in the regions of overlap are minimized by moving model vertices, followed by re-meshing. A limitation of this approach is the computational cost of repeated mesh subdivision and re-meshing. The approach presented in this paper differs in two main ways.

The first is that a sampling approach is used to minimize the reprojection error. This approach makes it so photometric consistency only has to be evaluated near the regions of the model which contribute to the reprojection error. The free-form deformation technique only deforms the regions of the mesh which are not photometrically consistent. This also results in a performance advantage.

The second difference is that the technique presented uses a polygonal visual hull as a starting point. A visual hull is almost always a closer approximation to the true shape of the object than a bounding ellipsoid. This is especially true in the case of objects which are not genus 0 (objects with holes in them). If a hole is visible from any point of view, the hole will be present in the final reconstruction, and thus non-genus 0 objects can be reconstructed. Another result of starting with the visual hull as a starting point is that object does not require as much re-meshing during the deformation process. In practice, model only needs to be remeshed a single time as an initialization step. Also, convergence of the model to a photometrically consistent solution, almost always takes fewer iterations starting from a visual hull then from a bounding ellipsoid.

There are a few techniques capable of building visual hulls at interactive frame rates; *e.g.*, Szeliski's octtree approach [24], Matusik's image based approach [15], and Matusik's silhouette prism polygonal CSG approach [14]. The technique presented in this paper not only efficiently extracts a visual hull, but takes it the process one step further, by re-meshing and deforming the visual hull in order to produce a photometrically consistent solution. However, Matusik's polygonal technique could be used as an input to the system presented after a post-processing step to produce a watertight mesh is used.

3 Approach

Fig. 1 gives an overview of the basic approach. Given a set of views and segmentation of the object in each view, constructive solid geometry is used to build a visual hull from silhouette prisms. The resulting polygonal mesh is then simplified and subdivided to produce a semi-regular mesh. Regions of model fit inaccuracy are found by projecting the reference images onto the mesh from different views. The resulting error images for each view are used to compute probability density functions for each image (probability of model error), and several points are sampled from it. Along the epipolar lines corresponding to these sampled points, the probability of the surface intersecting each position on the line is evaluated using a photometric consistency score. The mesh surface is then pulled towards regions of lower error using a free-form deformation

technique. The technique is iterative and is repeated from the error image construction step until convergence. In the rest of this section, the details of each step in the algorithm will be given.

3.1 Extracting the Polygonal Visual Hull

The preliminary visual hull mesh O_{pvh} is built using the intersection of the silhouette prisms for each reference image. First, simple background subtraction, thresholding, and binary image morphology operations are used to extract a region of interest (ROI) from each reference image. Second, line segment contours are extracted from the ROI functions implemented in the Intel OpenCV library [17]. The OpenCV allows the ROI to have multiple levels of internal contours; using this, the holes in the object can be represented in the visual hull. Next, the contours are simplified using the OpenCV implementation of an algorithm due to Douglas and Peucker [6]. The contours are then extruded in the direction of the view vector to build a silhouette prism for each reference image. The geometric intersection of the polygonal silhouette prisms is found using the TWIN solid modeling package [1].

Something interesting to note about contour simplification is that in this application, it can be viewed as a preliminary form of mesh simplification. The simpler the contour, the fewer polygons each silhouette prism will have, and consequently the fewer polygons in the visual hull. In practice we have found that contour simplification can make visual hull generation more than twice as fast with negligible loss in the quality of the simplified visual hull.

However, the resulting mesh after CSG will still have many sliver polygons, and the vertices will be located only at the intersection points of the silhouette prisms. To perform deformations on this mesh, a roughly uniform distribution of vertices over the mesh surface is needed. The reason for this is that the vertex positions define the shape of the object and act as sample points for the surface. In order to define the shape of the object in the non-silhouette regions the creation of additional vertices is required. However, blindly subdividing the mesh produces more vertices than are needed in the silhouette regions. This is where both mesh simplification and subdivision are useful.

3.2 Mesh Simplification and Subdivision

Once the initial visual hull has been produced as described above, then a second mesh that has a more uniform distribution of vertices over the surface is created using re-meshing techniques.

The first stage is to simplify the object. A variant of Garland and Heckbert's quadratic error metric [9] is used to simplify the mesh. This technique relies on *edge collapses* to simplify the object, where each edge collapse removes one vertex and two triangles from the mesh. However, the criteria for which edge to remove is based on edge length, rather than on the quadratic cost function. This criteria was chosen due to the fact that many of the visual hull meshes contained a large range of edge lengths due to the CSG intersection operation. The variance of edge lengths can be considered as a simple metric for uniformity of vertex distribution. Removing the shortest edge at every iteration of the mesh simplification makes the vertex distribution more uniform over the surface of the mesh. For the mesh simplification routine, the quadratic error metric is used only to find the new position of the vertex after an edge collapse. Just using a naive average of the two edge vertices has a tendency to cause the object to shrink during simplification.

One consideration that has to be made in simplifying the mesh is to not change the topological genus of the mesh, or cause the mesh to become non-watertight. This could happen inadvertently by collapsing a triangular hole into an edge or by collapsing a triangular cross section of the model into an edge. Because the geometry is a closed manifold, there is a simple heuristic that can be used to prevent this change in genus from happening. If the two vertices of the edge to be collapsed have more than two valence vertices in common, then the edge should not be collapsed. In practice, the mesh is simplified down to approximately 200 vertices before proceeding to the next step: subdivision.

After mesh simplification is used to obtain a roughly uniform distribution of vertices over the surface of the object, the object is subdivided by adding a vertex on each edge, and dividing each triangle into four via quaternary triangulation. This subdivision step is to get a denser sampling of vertices on the surface of the mesh to allow the free form deformations to produce a more detailed surface estimate.

The remeshed polygonal visual hull O_c can be used as the starting point for the iterative portion of the technique. In the final stage of each iteration, free form deformations are used to warp the current polygonal mesh towards a photometrically consistent solution. In order to get to this stage, image reprojection is required to find the non-photo consistent regions of the mesh.

3.3 Occlusion Compatible Image Reprojection

Sampling methods will be used to refine the areas where the model is not photometrically consistent. Photometric consistency can only be evaluated on the regions of the model which are visible in least two of the reference images.

In order to find these regions, the reference images must be reprojected back onto the portions of the model visible from each reference image’s center of projection (COP). A hardware accelerated volumetric stencil shadow algorithm known as Carmack’s Reverse [10] is used to reproject the reference images onto the COP-visible portions of the model. In addition to this, binary visibility images v_{i_o} are generated for each viewpoint i_o and each reprojection i . During each iteration of the algorithm, this step can be seen as successively approximating the visibility of each point on the object’s surface from each of the views.

For each reference viewpoint, each of the other reference images are reprojected separately onto the model, and images are captured. For each region of overlap, the photometric consistency metric is evaluated. In this paper we assume a Lambertian reflectance model, with additive Gaussian white noise; therefore, a sum of squared differences of pixel values is used in the error metric. However, when an image is reprojected into another view, there are pixel sampling rate problems that must be properly accounted for in the formulation of the error metric.

3.4 Reprojection Error Metric

The density of pixel samples reprojected from one viewpoint into another viewpoint can vary. The certainty of the photometric consistency model depends on how many pixels in the reprojected image map to a particular region of the model. This relationship can be expressed in terms of the variance of a weighted sum of pixel values within the reprojected region. Each pixel k has a value p_k in the reference images that can be thought of as a color value c_k plus additive Gaussian noise (sensor noise) $u_k = \mathbf{N}(0, \sigma^2)$,

$$p_k = c_k + u_k \quad (1)$$

In order to find the variance of a weighted sum of sampled pixels within a region on the model P , the following formula is used:

$$Var\left(\sum_{k \in P} \omega_k p_k\right) = \mathbf{E}\left(\left(\sum_{k \in P} \omega_k p_k\right)^2\right) - \mathbf{E}\left(\sum_{k \in P} \omega_k p_k\right)^2 \quad (2)$$

After algebraic manipulation the variance of a weighted sum of pixel values can be found to be:

$$Var\left(\sum_{k \in P} \omega_k p_k\right) = \sigma^2 \sum_{k \in P} \omega_k^2 \quad (3)$$

Finding the exact weights for all the pixels which project to a particular region on the model is computationally prohibitive; therefore, we approximate the weights of the pixels using the area \mathbf{A}_k of the projection of the pixel in a view onto the model, the area of the region on the model \mathbf{A}_P , and the number of pixels projecting to the region n . Making the simplification that all pixels in the projected area have equal weights ($\omega = \frac{\mathbf{A}_\mu}{\mathbf{A}_P}$) where ($\mathbf{A}_\mu = \frac{1}{n} \sum_{k \in P} \mathbf{A}_k$), the variance of the weighted sum becomes proportional to the average pixel projection area \mathbf{A}_μ :

$$Var\left(\sum_k \omega_k p_k\right) = \sigma^2 \sum_k \left(\frac{\mathbf{A}_\mu}{\mathbf{A}_P}\right)^2 = \sigma^2 \frac{\mathbf{A}_\mu}{\mathbf{A}_P} \quad (4)$$

The basic intuition that can be taken from this is that the regions on the object where more texels project to can be reconstructed more accurately given sufficient image detail. However, finding the exact projection area of each pixel’s projection is computationally expensive. It requires clipping the current mesh to each pixel’s frustum and computing the areas of the resulting polygons. The following approximation is used instead:

$$\mathbf{A}_{k,i} \simeq \frac{d_{k,i}^2}{(N_{k,i} \cdot V_{k,i})} \quad (5)$$

For each pixel k in the image i , the projection area on the model is proportional to the squared distance to the object $d_{k,i}^2$, and approximately inversely proportional to the cosine of the angle of incidence the view vector V_k makes with the surface normal $N_{k,i}$. To put it simply, the projection area depends on how close the camera is to the object, and how edge-on the view vector for each pixel is to surface normal of the object. The beauty of this approximation is that it is very efficient to compute using the per-pixel capabilities of modern graphics hardware.

Now that we have an efficient way to approximate pixel projection areas, the error term can be written as a weighted sum of squared differences between image i_o and the other images i projected into the view i_o .

$$Err_{i_o} = \sum_{i, i \neq i_o} v_{ii_o} \frac{1}{A_i A_{i_o}} G[(I_{i_o} - P_{ii_o}(O_c, I_i))^2]. \quad (6)$$

This error term gives the photometric consistency model used in the formulation. P_{ii_o} projects an image from view i into view i_o using O_c which is the current estimate mesh of the object. G is a function that sums the rgb color channels together.

The whole idea behind this technique is to minimize Err_{i_o} by deforming the mesh. The goal is to find a set of deformations which deform the visual hull in a way that minimizes the reprojection mismatch. In order to perform this minimization, this technique chooses regions of high error within the image stochastically and then uses the epipolar lines corresponding to these points as inputs to an epipolar stochastic search technique.

3.5 Stochastic Epipolar Search

Once the reprojection error images are created, they can be used to compute a series of PDFs from which points can be sampled. The main idea of the sampling technique is to generate more samples from regions in the image that have higher error. Given the PDF for each viewpoint, the Cumulative Distribution Function (CDF) is calculated. The CDF is computed using a row major running sum of pixel values over the error image, which is then divided by the final sum. To sample from the PDF, first a uniformly distributed random value $u = [0, 1]$ is chosen. To sample a point from the error image, a bisection search in the CDF image for the value u is performed. The resulting point is the sample point. One optimization used to expedite this calculation is to multiply the value of u by the PDF image sum rather than divide every pixel in the CDF by the sum. This is mathematically equivalent. Probabilistically, these sample points will project to the regions of the model with high error.

For each sample point chosen in each image, an epipolar line is generated. These strands will help to form the basis for the error minimizing free-form deformations.

Each strand is generated from a stochastically chosen point in each image i_o . Along each ray starting from the first point of intersection of the strand with the visual hull, photometric consistency is evaluated at evenly-spaced sampling intervals along the strand until the next intersection point with the visual hull. Note that the visual hull is used to bound the sample space rather than the current estimate so that backtracking is possible if some sections of the estimate are deformed too far inward. For the results in this paper, the sampling interval is determined using the length of the visual hull's bounding box's diagonal.

At each point along the strand, the 3D point location is reprojected into each of the other images i that are visible from the first point the strand intersects with the model. This strand visibility can be approximated quickly by using the value of visibility image v_{ii_o} at the sampled point. This gives the visibility of a point on the surface before deformations. Photometric consistency is only evaluated using the images visible from the strand intersection point. Of course, the actual visibility may be different than the approximate visibility for points along the strand. To help account for this, we use an error metric which allows for rejecting viewpoints for which the point is not visible.

At each point, when photometric consistency is evaluated, any views which have a consistency score above a threshold are given a score corresponding to the worst possible photometric consistency score. The threshold is determined by the sensor noise, and can be considered as a way to reject views which are not visible from a certain point on the strand. This way, solution points having different visibility than the strand's intersection with the current estimate model can be found. In addition to this, the thresholding acts as a form of outlier rejection, and gives the system some robustness to specular highlights, which are view dependent and are generally present in a particular position on the model in a single view.

As a result of the thresholding, the minimum value on the strand is the one which is photometrically consistent in the largest number of views. If there are many points which are consistent in the same number of views, the point with the lowest

average error is used.

However there is still another issue to be considered when evaluating photometric consistency on the strands. Finding the exact region of pixels in image i which project to the pixel k in view i_o would require us to deform the model and reproject the image onto the model, for each point on the strand. Obviously, this is computationally prohibitive. To sample this projected area adequately, a jittering technique is used. Instead of sampling from a single epipolar ray per strand, samples are taken from a pencil of rays generated by generating uniformly distributed random points within the chosen pixel, and extruding a ray for each one through the COP. The results of sampling are averaged across the samples taken from each strand.

After the photometric consistency is evaluated over the strand, the point with minimum error value is used to determine the free form deformation (FFD) vector. The FFD vector is calculated by using the first intersection of the strand with the current object as the initial position of the deformation. The minimum error point on the strand is used as the target position, to be reached via the FFD. After all the strand minimums are found, and the free form deformation vectors are calculated, the next step is to deform the visual hull using these vectors.

3.6 Free Form Deformation using Stochastic Strands

Given a set of scattered FFD vectors in space, in order to use these vectors as a basis for free form deformations, the vectors must be interpolated through space. The method used to accomplish this is a kernel-based, scattered data interpolation method. The basic idea is that every vertex in the model will be deformed using a weighted average of deformation vectors. For scattered data interpolation, Franke and Neilson's distance weighted method is used:

$$\omega_j(\mathbf{p}) = \begin{cases} \left(\frac{R_j - d_j(\mathbf{p})}{R_j(d_j(\mathbf{p}) + \epsilon)} \right)^\mu & d_j(\mathbf{p}) < R_j \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

This formulation allows each deformation vector j to have a region of influence defined by radius R_j and a Euclidean distance from center function $d_j(x) = \sqrt{(x - c)^T(x - c)}$ where c is the initial position of the free form deformation. Any points within this region are affected by the corresponding FFD vector. The value ϵ is used to prevent numeric instability from division by values close to zero.

Given a point \mathbf{p} , the weights for the FFD can be calculated using the above formula, and then normalizing to make the weights have a unit sum:

$$\hat{\omega}_j(\mathbf{p}) = \frac{\omega_j(\mathbf{p})}{\sum_{m=0}^n \omega_m(\mathbf{p})}. \quad (8)$$

After finding the weights $\hat{\omega}_i(\mathbf{p})$, the interpolated FFD vector for a point can be found as the weighted average of the FFD vectors \mathbf{f}_j :

$$\mathbf{f}(\mathbf{p}) = \sum_{j=0}^n \hat{\omega}_j(\mathbf{p}) \mathbf{f}_j. \quad (9)$$

In practice $\mu = 2$ and a value of one tenth the length of the diagonal of the visual hull's bounding box for R_j worked well.

To deform the object, the FFD vectors are calculated for each vertex of the visual hull, and the hull is deformed. After the deformation, the deformed object is used as the input to the reprojection stage and the process becomes an iterative one. Each iteration creates a new set of strands to correct for the high error regions until either the error images become less than the sensor noise (e.g. photometrically consistent), or the improvement in the error images from iteration to iteration diminishes.

4 Results

The technique was tested on synthetic and real data on an 1200Mhz AMD Athlon with 2.0GB RAM.

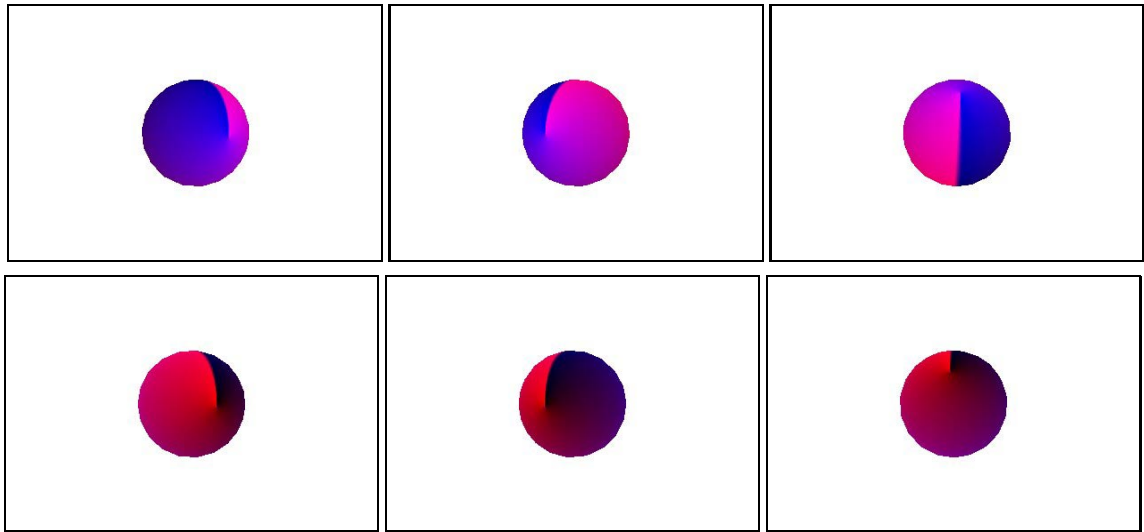


Figure 2. Synthetic Data Test 1: Six views of a texture mapped sphere to reconstruct a model from.

4.1 Synthetic Data

To clearly demonstrate how the technique works, and as a proof of concept, a reconstruction of a simple synthetic object (a multicolored sphere) is shown. Six views of the object were generated for this test, three nearby views of the top of the sphere and 3 nearby views of the bottom. Figure 2 shows the different views used. Image collection, contour extraction, contour simplification, and silhouette prism creation took less than a second. Computing the visual hull, and the subsequent remeshing each also took less than a second. The post-CSG visual hull contained 92 vertices, and the remeshed hull contained 770 vertices.

Error image creation using reprojection took 15 seconds for the six 400x300 views. More than 75 percent of the time was taken by calls to `glReadPixels`, meaning that the copying from video memory to main memory (and any data conversion internal to the function that took place) was the bottleneck for performance. For each of the six error images 50 strands were created. Each strand was sampled along 20 jittered epipolar rays at up to 100 points within each the hull. The time taken to process the total of 300 strands was 7 seconds. Performing the FFD using the strands took less than 1 second. A photometrically consistent solution was found in a single iteration.

Figures 3 and 4 show the various stages of reconstruction. This example illustrates how the visual hull can differ from the actual shape of the object, and how our technique deforms the hull towards a photometrically consistent solution. Notice how the football shaped visual hull gets deformed into the correct shape, a sphere.

Another test using synthetic data was run using images of a bowl shaped object. Ten separate 400x240 views of the object were used as input to the system. The computed visual hull contained 270 vertices, and was remeshed to 3170 vertices. The remeshing took 4 seconds. Error image creation took 28 seconds for 10 views at 400x270. Strand creation took 32 seconds (100 strands per view, 20 jittered rays per strand, up to 200 sample points per ray). Free form deformations took 9 seconds total. As you can see, our stochastic strand technique is able to recover the concavity in the bowl quite well. Note that the silhouettes contain no information whatsoever about the concavity. The example shows quite clearly the additional information that can be extracted using color consistency.

The system was also tested using real world data. The first real world data example used 5 views of a rose. Contour extraction and visual hull creation took two seconds. The visual hull was remeshed from 238 vertices to 3170 vertices in 3 seconds. Error image creation took 8 seconds (5 images at 400x270) and strand creation took 15 seconds. (again 100 strands per view, 20 jittered rays per strand, up to 200 sample points per ray). The free form deformations took 4 seconds. Notice how our photometric technique extracts more detail out of the top and the left petal of the rose.

The next example used 11 views of a toy dinosaur as input to the system. This example was more computationally intense than the others, contour extraction took two seconds, and visual hull creation took another two seconds. The resulting visual

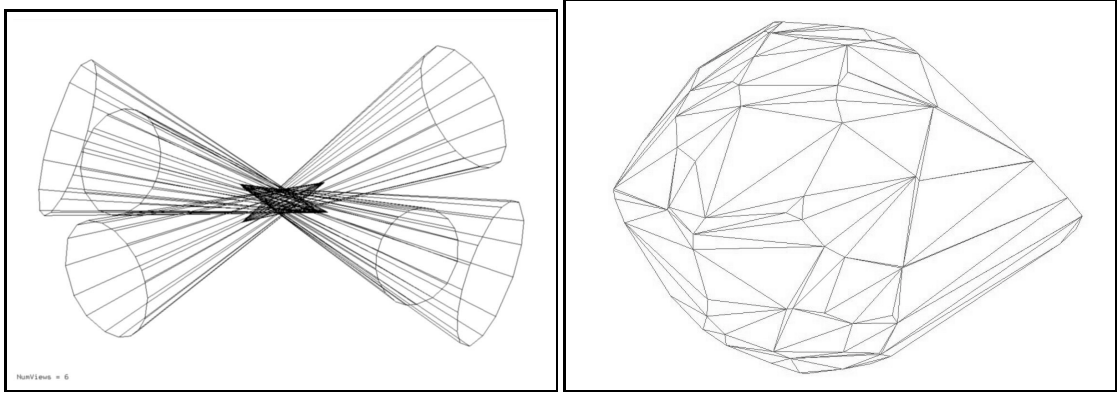


Figure 3. Synthetic Data Test 1: The image on the left shows the six silhouette prisms generated from the images. The image on the right shows the resulting visual hull created using polygonal CSG.

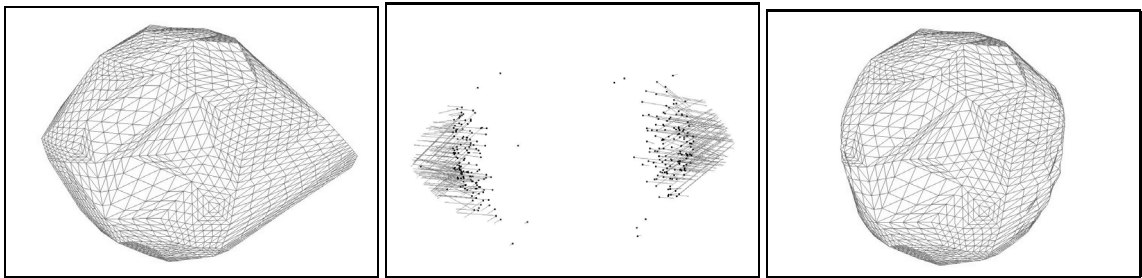


Figure 4. Synthetic Data Test: The leftmost image shows the visual hull after simplification and subdivision. The center image shows the 300 FFD vectors created using the stochastic strands. The rightmost image shows the object after free form deformations. This object is photometrically correct, and very close in shape to the original mesh used to generate the test images.

hull contained 1049 vertices, and was remeshed to 16634 vertices in three seconds. Error image creation (11 images at a resolution of 486x640), took 34 seconds. Strand creation (550 strands total, 50 strands per view, and 20 jittered rays per strand) took 28 seconds. The free-form deformations using the strands took 12 seconds.

For this example, the resulting meshes are displayed as flat shaded for the reason that a wireframe or textured image does not show the shape of the surface well in the figures. In the reconstructed view of the dinosaur, it can be seen that how our technique extracts more geometric detail than present in the visual hull. You can see the ridge of the spine as well as the ridge where the leg meets the body. Also, notice how the bottom of the arm has become more defined.

5 Conclusion and Future Work

This paper demonstrated a system for computing a photometrically consistent polygonal mesh from a calibrated set of cameras. The cameras do not have to be oriented or positioned any special way. The only requirement is that regions of the model to be reconstructed must be visible by at least two cameras. In addition to this, due to our stochastic sampling technique, and use of graphics hardware, the system runs much quicker than other systems which generate photometrically correct solutions given arbitrary camera placement.

One of the key strengths of this approach is the fact that it is mesh based. Because of this, not only is the shape of the model estimated, but surface normals can be extracted relatively easily as well. These surface normals are used to account for differences in the numbers of pixels projected onto a particular area of the model. Accounting for these differences in

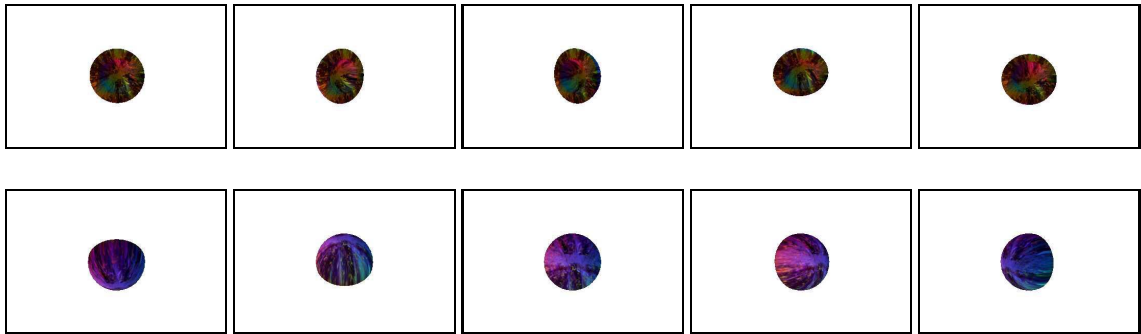


Figure 5. Synthetic Data Test 2: Ten views of a texture mapped bowl to reconstruct a model from.

pixel sampling rate is another key advantage of our approach. One area of future work we would like to look into is to use the surface normals in order to incorporate models of surface lighting other than simple Lambertian ones.

Also, we have been looking into other error metrics based on view-to-view pixel projection areas rather than view-to-model pixel projection areas. By doing this, the variance of the image differences can be computed, and the error can take the form of a Mahalanobis distance. However, this error metric tends to heavily weight the pixels projecting to regions with large angles of incidence from view i_o . Because of this, the system becomes very sensitive to geometric camera calibration error. To use this error metric, geometric and color camera calibration error also need to be accounted for in the formulation as well. This is a topic we are actively pursuing.

One topic we would like to investigate further is a way to automatically choose parameters for our contour simplification, mesh simplification, and mesh subdivision based on the scene complexity. Obviously, the more the contours are simplified, the less the visual hull has to be simplified to remove sliver polygons. The tradeoff is that the computed visual hull may be further from the true shape of the object, and the technique may require more iterations to produce an acceptable result.

Acknowledgments

Special thanks to Professor Steven Seitz for providing the dinosaur and rose sequences.

References

- [1] D. Anderson. TWIN multiplatform faceted BREP modeller. Technical Report , Computer Aided Design and Graphics Laboratory (CADLAB), School of Mechanical Engineering, Purdue University, West Lafayette, IN, 1995. <http://www.cadlab.ecn.purdue.edu/twin/>.
- [2] A. Broadhurst, T. W. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *Proc. ICCV*, pages I:388–393, 2001.
- [3] Y. Chen and G. Medioni. Fitting a surface to 3D points using an inflating balloon model. In *Proc. Second CAD-Based Vision Workshop*, pages 266–273, 1994.
- [4] W. B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *ICCV Workshop on Vision Algorithms and Theory and Practice*, pages 100–115, 1999.
- [5] J. S. de Bonet and P. Viola. Roxels: Responsibility weighted 3D volume reconstruction. In *Proc. ICCV*, pages 418–425, 1999.
- [6] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.
- [7] A. W. Fitzgibbon, G. Cross, and A. Zisserman. Model construction for turn-table sequences. In *Proc. SMILE'98*, pages 155–170. Lecture Notes in Computer Science 1506, 1998.
- [8] P. Fua and Y. G. LeClerc. Object centered surface representations: Combining multiple-image stereo and shading. *International J. of Computer Vision*, 16(1):35–56, 1995.
- [9] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Proc. SIGGRAPH*, pages 209–216, 1997.
- [10] M. Kilgard. More advanced hardware rendering techniques. Technical report, NVIDIA, Santa Clara, CA, 2001. <http://www.nvidia.com/Marketing/Developer/DevRel.nsf/TechnicalPresentationsFrame?OpenPage2>.
- [11] K. N. Kutulakos. Approximate N-view stereo. In *Proc. ECCV*, pages 67–83, 2000.

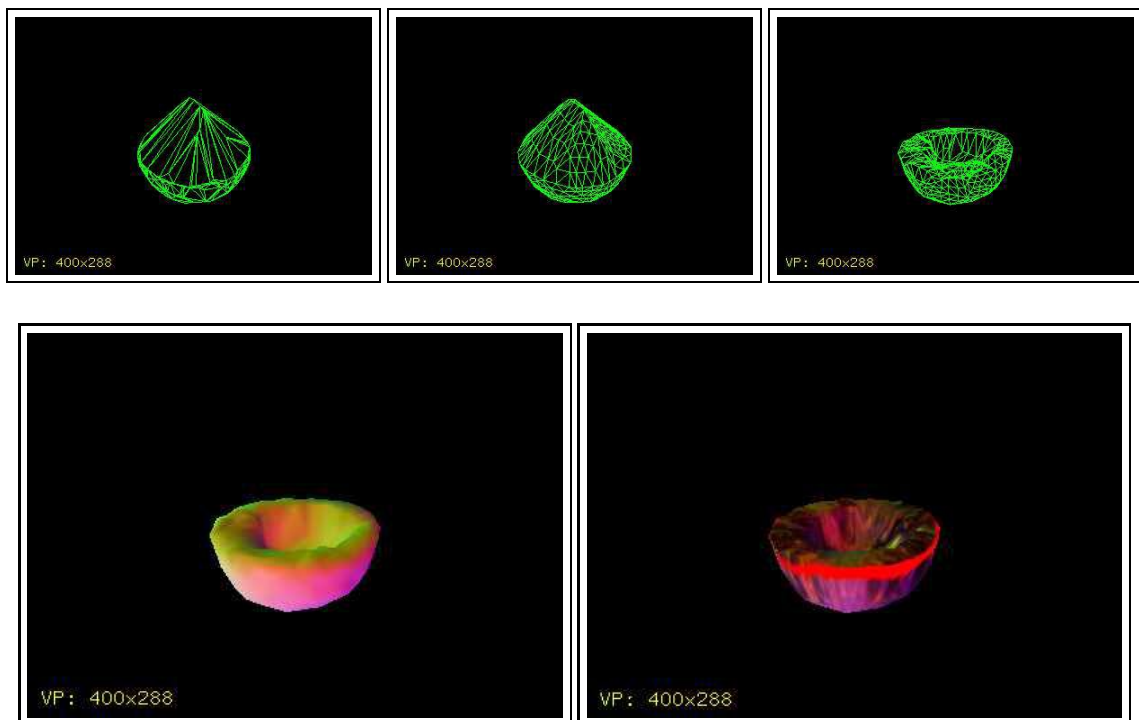


Figure 6. Synthetic Data Test 2: Top Row: Visual Hull before simplification, after simplification, and after free form deformations. Bottom Row: Object after free form deformations shown with smooth shading and with projective texture mapping using the original images.

- [12] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *Proc. ICCV*, pages 307–314, 1999.
- [13] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International J. of Computer Vision*, 38(3):197–216, 2000.
- [14] W. Matusik, C. Buehler, and L. McMillan.
- [15] W. Matusik, C. Buehler, R. Raskar, L. McMillan, and S. J. Gortler. Image-based visual hulls. In *Proc. SIGGRAPH*, pages 369–374, 2000.
- [16] T. McInerney and D. Terzopoulos. A finite element model for 3-D shape reconstruction and nonrigid motion tracking. In *Proc. ICCV*, pages 518–523, 1993.
- [17] OpenCV. Intel open source computer vision library. <http://developer.intel.com/software/products/opensource/libraries/cvfl.html>.
- [18] A. Pentland and S. Sclaroff. Closed-form solutions for physically-based shape modeling and recognition. *IEEE T-PAMI*, 13(7):715–729, July 1991.
- [19] A. Prock and C. Dyer. Towards real-time voxel coloring. In *Proc. DARPA Image Understanding Workshop*, 1998.
- [20] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. CVPR*, pages 1067–1073, 1997.
- [21] G. Slabaugh, W. B. Culbertson, T. Malzbender, and R. Schafer. Improved voxel coloring via volumetric optimization. Technical report, Center For Image and Signal Processing, Georgia Inst. of Technology, Atlanta, GA, 2000. <http://www.ece.gatech.edu/slabaugh>.
- [22] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proc. CVPR*, pages 1:345–352, 2000.
- [23] E. Steinbach and B. Girod. 3D reconstruction of real-world objects using extended voxels. In *Proc. ICIP*, volume 3, pages 823–826, 2000.
- [24] R. Szeliski. Real-time octree generation from rotating objects. Technical Report 90/12, Digital Equipment Corporation, Cambridge Research Lab, 1990.
- [25] D. Terzopoulos and D. Metaxas. Dynamic 3-D models with local and global deformations: Deformable superquadrics. *IEEE T-PAMI*, 13(7):703–714, 1991.
- [26] L. Zhang and S. Seitz. Image-based multiresolution modeling by surface deformation. Technical Report CMU-RI-TR-00-07, Carnegie Mellon University, Pittsburgh, PA, January 2000.



Figure 7. Real World Data Test 1: 5 views of a rose to reconstruct a model from.

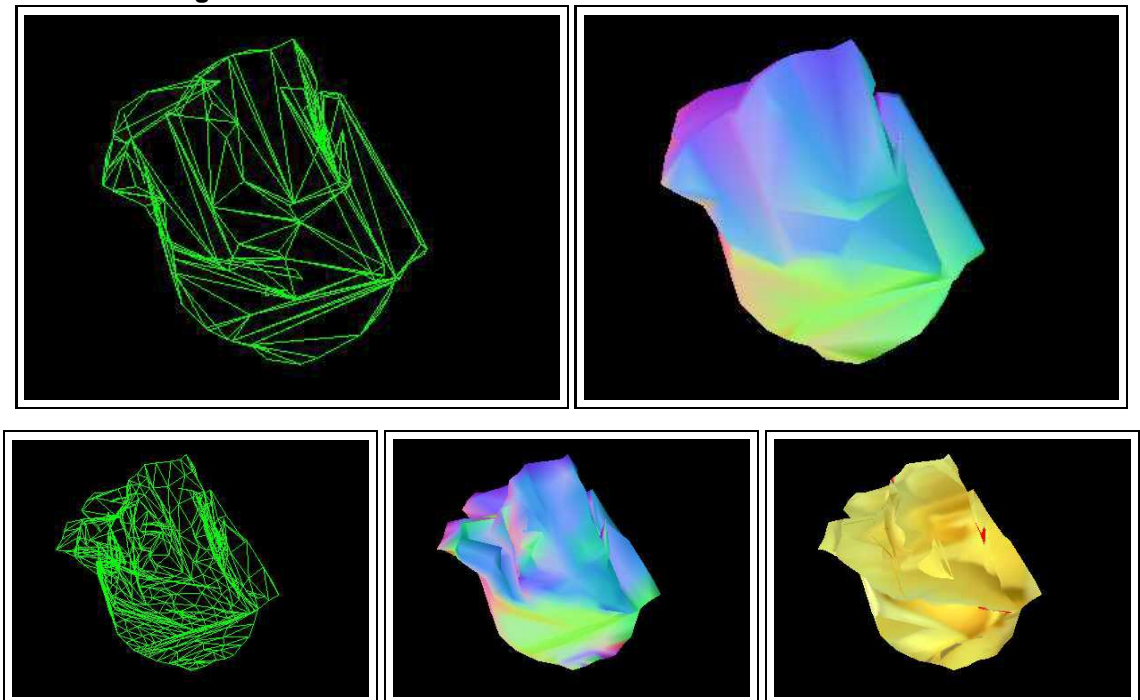


Figure 8. Real World Data Test 1: The top row shows wireframe and smooth shaded views of the visual hull, and the bottom row shows wireframe, smooth shaded, and texture mapped views of the object after deformation.

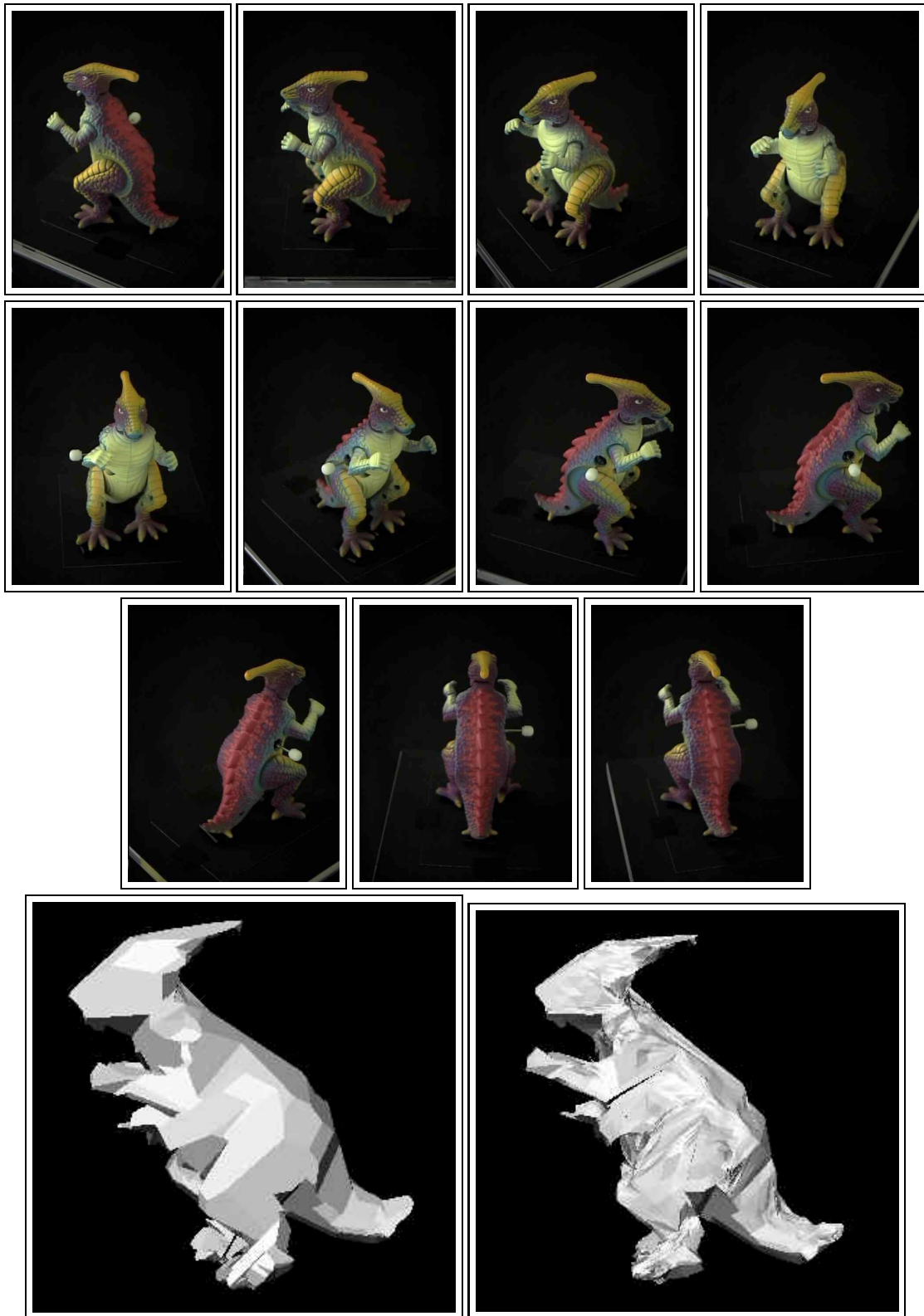


Figure 9. Real World Data Test: The top three rows contains the eleven views used to reconstruct the dinosaur. The bottom row shows the visual hull, and the object after deformation.