From images to models (and beyond): a personal retrospective

Richard Szeliski
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
szeliski@microsoft.com

Abstract

This paper surveys a number of techniques for extracting 3D geometry and photometry from multiple images. The paper describes the recovery of volumetric models from silhouettes, the extraction of 3D surface curves from profiles, the extraction and integration of 3D surface points from stereo, and a dynamic physically-based surface model for integrating the outputs of these procedures. It also presents recent work in image-based rendering, i.e., the representation of objects and scenes using collections of images which are warped and blended to create novel views.

1 Introduction

The reconstruction of 3D models from imagery has long been one of the central topics in computer vision [6, 5, 19]. Initially, it was believed that high-level 3D models were a prerequisite for image and scene understanding, as well as for robotics tasks such as navigation and manipulation. Today, it is widely accepted that visual information can be extracted and used at many levels. For instance, recognition tasks can often be solved using a collection of 2D views or features.

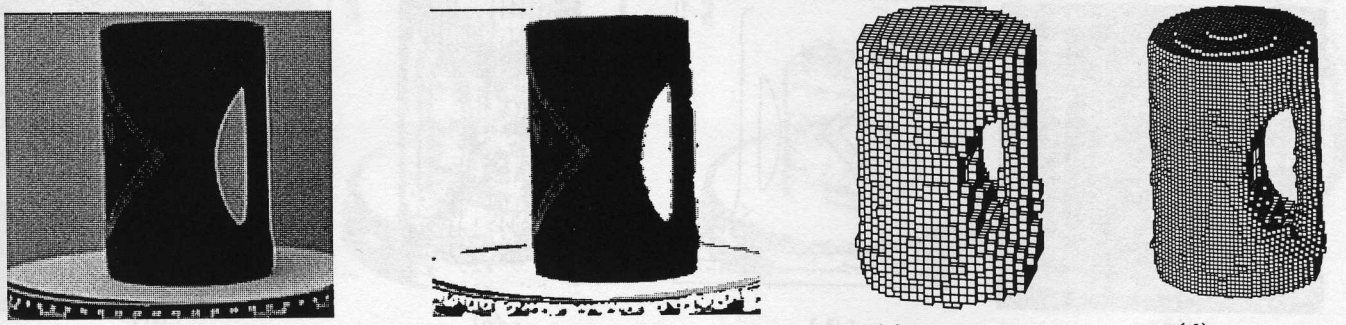
The last few years have seen a renewed interest in the acquisition of photorealistic 3D models for applications such as computer graphics, special effects, and the creation of virtual environments. Active range sensors such as laser scanners provide one way of acquiring such data [10, 43]. Passive image-based reconstruction techniques provide a potentially lower-cost alternative which can be applied to any sized object. These techniques, which reconstruct 3D shape from a number of views of an object or scene, also have the potential to capture the full visual richness of a complex object or scene through the use of view interpolation and view-dependent texture maps [24, 17].

This paper surveys a number of 3D reconstruction techniques which I have developed over the last ten years. My interest in 3D modeling began with an investigation of multiframe stereo algorithms [37]. I then developed a number of full 3D reconstruction algorithms using both surface [54, 65] and volumetric [55] models. I also became interested in the problem of surface reconstruction from sparse data [62, 64], and structure from motion techniques for recovering camera motion [59]. More recently, I have been investigating techniques for building large panoramic scene descriptions [56, 60, 57, 30] using a combination of projective structure from motion and multiframe stereo techniques. Most recently, I have been involved in research into image-based rendering [24].

This paper parallels the chronology of my research into these topics. Section 2 presents a technique for building volumetric models from binary foreground/background silhouettes. Section 3 describes our algorithm for estimating surface shape from tracked contours. Section 4 describes our method for integrating narrow-baseline stereo estimates using a cloud of 3D points with associated uncertainties. Section 5 discusses our technique for extracting and modeling 3D surfaces of arbitrary topology using collections of oriented surface elements. Section 6 discusses some approaches and recent results in extracting 3D surface color distributions (i.e., texture maps) from multiple images. Section 7 presents some recent results on image-based rendering. Section 8 briefly mentions issues related to camera calibration and structure from motion. We close with a discussion of potential applications and topics for future research.

2 Volumes from silhouettes

The first three reconstruction algorithms described in this paper use a controlled motion platform and a stationary video camera connected to a frame grabber. The motion platform is an inexpensive spring-loaded microwave turntable, to which we affixed a paper strip with an 8-bit pattern which encodes the turntable's angular position (Figure 1a) [54]. As the object spins on the turntable, video images are captured, the turntable angle is computed, and selected images (typically at 3° to 10° spacing) are retained for further processing. The relative position of the video camera and the turntable, and also the internal calibration parameters of the camera, must be known in order to recover 3D shape. We perform this calibration prior to model acquisition by placing a simple hexagonal pattern on the top of the



(a) (b) (c) (d)

Figure 1: Shape from silhouettes example: (a) input image, (b) silhouette image (non-white areas are considered part of the object), (c) lower-resolution octree model (5 levels), (d) higher-resolution octree model (6 levels).

turntable.[1] See Section 8 for alternative techniques for calibrating the camera and estimating its pose.

The first algorithm we developed for extracting 3D shape from multiple views constructs a volumetric representation of the object from the binary *silhouettes* of the object (Figure 1b). These silhouettes can easily be extracted from imagery by first acquiring an image of a blank turntable, and then doing simple pixel differencing.[2] This approach requires the colors of the object and background to be different, but this is not difficult to ensure in practice.

Each silhouette, together with the corresponding center of projection for the camera, defines a generalized cone in space within which the object must lie. The intersection of these cones in 3D defines a bounding volume for the object, which under many conditions is a reasonable approximation to the object's shape. Several different techniques have been developed in the past for computing this volume, including techniques which represent the volume as a polyhedron [53] and as an octree [42]. Our algorithm uses the octree representation, combined with the multi-resolution refinement algorithm described below, to minimize the overall amount of computation [55].

Octrees are tree-structured volumetric representations constructed by recursively subdividing cubes into eight equal sub-cubes [45]. Leaf nodes in the tree can be either *white* (empty) or *black* (occupied), while interior nodes are called *gray* and have children of different colors. Figure 2a shows a graphical view of a small octree, and Figure 2b shows its associated colored tree representation. The octree is usually more efficient than a pure voxel representation since the interior of the object is typically described by large cubes. For reasonably smooth objects, the total number of nodes in the tree

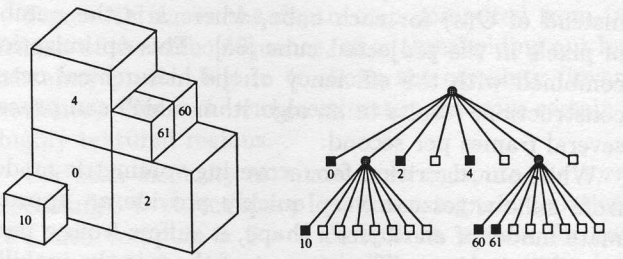


Figure 2: A simple two-level octree and its tree representation

is proportional to the object's surface area instead of its volume.

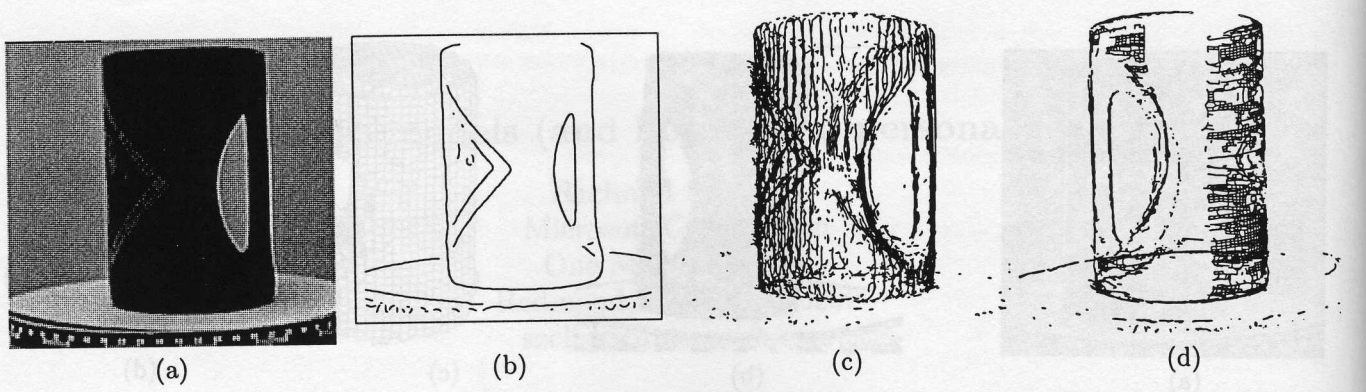
To construct the bounding volume from a sequence of silhouettes, we perform a series of forward projections from the 3D octree model to the 2D image plane using the known camera model (calibrated camera matrix plus turntable angle). Cubes in the octree which fall completely into the background can immediately be removed from the object model. Cubes which are partially inside the silhouette may be partially occupied, and must be subdivided to a finer resolution.

Our algorithm therefore constructs the octree in a hierarchical coarse-to-fine fashion. For a series of increasing octree resolutions, the inner loop of the algorithm refines the 3D volume by testing projected octree cubes against the sequence of silhouettes. After one complete revolution of the object, those cubes whose occupancy is uncertain are subdivided. Because the octree is completely constructed at one resolution before refining the next one, we perform fewer total computations than a non-hierarchical algorithm.

To rapidly compute whether a projected cube is inside the background or silhouette, we first compute a bounding box for the eight projected corners. A distance map, which is computed in time proportional to the image size, provides the minimum distance from any image point to the silhouette. This distance map applied to the bounding box gives a silhouette test that is $O(1)$

[1] The use of a planar calibration pattern requires that the aspect ratio of the camera be known. We have verified empirically that this ratio is close to 1 for our cameras.

[2] To close up small holes in the silhouette, we use a combination of morphological operations and connected components [44].



(a) (b) (c) (d)

Figure 3: Shape from contours example: (a) input image, (b) extracted edges, (c) reconstructed 3D edges (side view), (d) a portion of the surface mesh.

instead of $O(s)$ for each cube, where s is the number of pixels in the projected cube [55]. This optimization, combined with the efficiency of the hierarchical octree construction, results in an algorithm which can process several frames per second.

While our algorithm for recovering volumetric models from silhouettes can very quickly provide an approximate model of an object's shape, it suffers from a number of limitations. The greatest of these is the inability to sense certain kinds of cavities (such as the inside of the coffee cup). More precisely, our technique can recover the *visual hull* of an object, i.e., the complement to the space swept out by all lines that do not penetrate the object [32]. Other limitations include the susceptibility to misclassification errors which may be caused by specularities and accidental matches between object and background colors, as well as the limited precision available with volumetric representations.

3 Surface curves from profiles

The shape from silhouette method provides a bounding volume for a surface, but as described above it has limitations. To overcome some of these limitations, we developed an algorithm to directly compute 3D surface patches and curves from a sequence of image contours [65] (see [15, 71, 74, 12] for related techniques). Image contours come from surface markings, surface tangent discontinuities, and occluding contours. Surface markings and tangent discontinuities are viewpoint invariant, whereas occluding contours are not, and so cannot be used directly in two-frame stereo reconstruction algorithms.

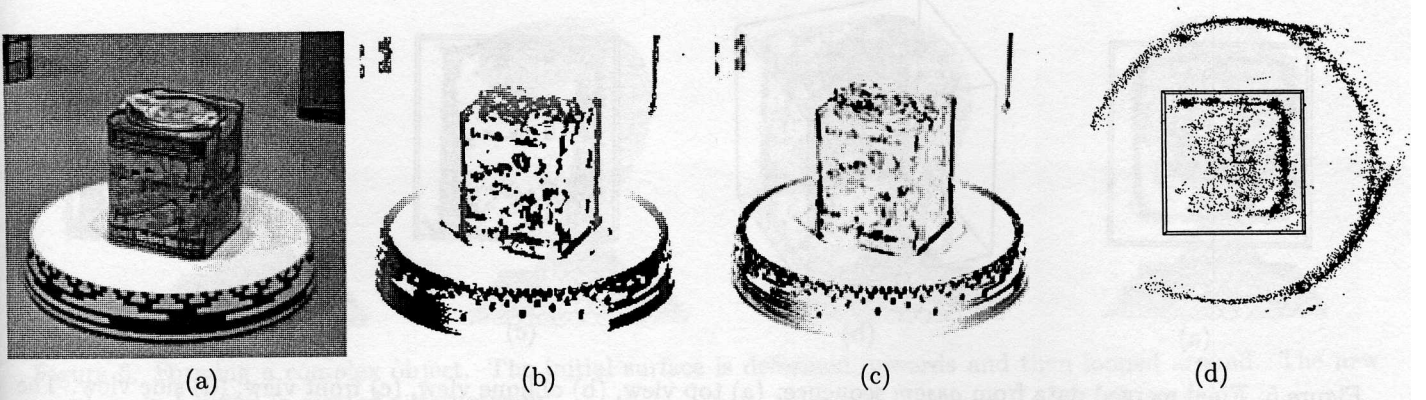
Given a sequence of images of an object, we extract edges and link them into contours (Figure 3b). The contours are tracked over the sequence of images, and 3D contours are estimated (Figure 3c). The 3D contours are linked by the trajectories of the tracks of the edge points in the images (Figure 3d). The known motion of the camera is used in two ways: it constrains the search for corresponding points during tracking, and it is used in the computation of the 3D points.

Edge detection is done using first order *steerable filters* [21], since they provide good angular resolution. Once discrete edgels have been detected, we use local search (based on proximity and continuity of orientation) to link the edgels into contours. Note that in contrast to some of the previous work in reconstruction from occluding contours [15], we do not fit a smooth parametric curve to the contour since we wish to directly use all of the edgels in the shape reconstruction, without losing detail.[3]

The pointwise correspondence between contours is based on the *epipolar constraint*. For two images, i.e., classical *binocular stereo*, this epipolar constraint is a set of straight lines which are the intersection of *epipolar planes* with the image plane [19]. For more than two images, the epipolar plane through a point is determined by the view direction at that point and the instantaneous camera velocity. The projection of the epipolar plane into the next frame, i.e., the epipolar line, is used to find the best matching edgel in the next frame. Our technique compares all candidate edgels within the epipolar line search range (defined by the expected minimum and maximum depths), and selects the one that matches most closely in orientation, contrast, and intensity. Once an initial estimate for the 3D location of an edgel has been computed, the search range can be dramatically reduced.

Given three or more edgels tracked with our technique, we compute the location of the surface and its curvature by fitting a circular arc to the lines defined by the view directions at those edgels. This curve fitting problem is *linear*, which allows us to use recursive least squares techniques [65]. To further improve the quality and reliability of our estimates, we apply *robust statistics* to reduce the effects of *outliers* which are due to grossly erroneous measurements as well as large changes in the surface curvature [28].

[3] However, we do perform a small amount of curvature-dependent smoothing along the curves to reduce noise. This can be viewed as part of the edge extraction stage.



(a) (b) (c) (d)

Figure 4: 3D depth map computed from *assam* image sequence: (a) first image in sequence, (b) depth map from flow (darker is nearer), (c) certainty in depth estimates (darker is higher certainty), (d) top view of 3D point cloud.

Once the circular arc is fitted, a point on the surface is computed which corresponds to the middle view direction. The topology of the surface in the form of a mesh is captured by maintaining the epipolar curves, which are given by the edge point trajectories, and the 3D contours. Figure 3c shows the complete set of 3D contours reconstructed from a 360° view of a cup, while Figure 3d shows a portion of the 3D mesh (contour curves plus epipolar curves).

Occluding contours sweep out surface patches on the *visible* region of the surface, which is the union of all critical sets that are not occluded. In general, these curves and patches will provide information in many places where the line hull differs from the original surface. The limitations of reconstruction from contours is that it may not produce a closed surface and the reconstruction degrades as the contours become parallel to the epipolar constraints. The technique may also fail in highly textured regions where it is difficult to consistently extract edges.

4 3D points from stereo

An alternative to extracting surface shape by matching 2D edges is to use dense (correlation-based) stereo, and then integrating the resulting depth maps into a complete 3D model [54].

Our algorithm first computes dense disparity maps from successive pairs of images in the input image stream. The images are re-sampled (*orthorectified*) so that corresponding epipolar lines are horizontal [19], and each image is interpolated horizontally by a factor of 4. Disparities are estimated using a Sum of Squared Differences (SSD) algorithm [3], since it provides not only sub-pixel disparity estimates, but also uncertainty estimates for each pixel. The sub-pixel disparity estimate at each pixel is computed by fitting a parabola to the minimum SSD value; the analytic minimum is used to compute the disparity, while the variance in this estimate is computed using the second derivative of the parabola and a local noise estimate [37].

Figure 4b shows the depth map computed from two frames of the video sequence, after thresholding out low-certainty pixels. Figure 4c shows the certainty (inverse variance). Notice how the estimates are more certain in highly textured regions.

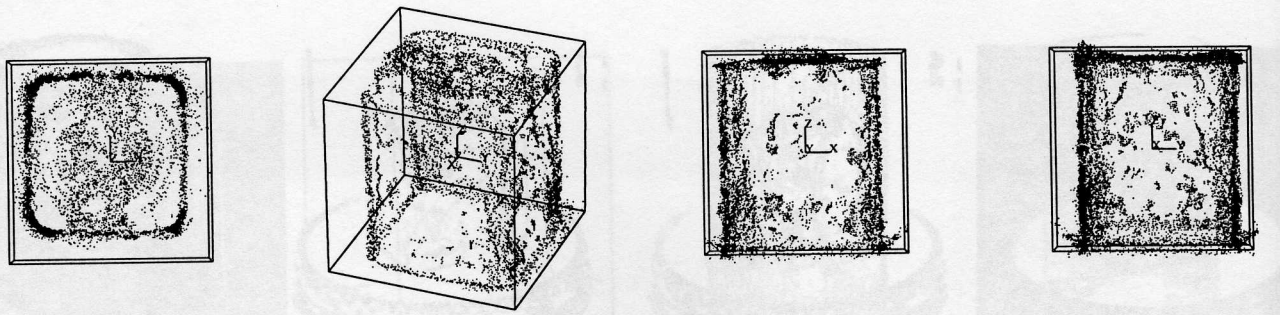
For each disparity estimate d we compute the corresponding 3-D object space location using triangulation.[4] We also compute a 3×3 covariance matrix for each point which characterizes the shape and magnitude of the point's positional uncertainty. The component of this covariance along the viewing ray is computed using the disparity estimate variance and the Jacobian of the inverse projection (triangulation) operator. The other two axes of the covariance ellipsoid can be chosen arbitrarily and their length (standard deviation) set to a suitably chosen constant.

The result of our two-frame stereo matching and backprojection into object space gives us a "cloud" of uncertainty-tagged points lying on the surface of the object (Figure 4d). As the object continues to rotate and more points are acquired, point collections from successive frames are merged in order to reduce the noise in point location estimates.

To merge neighboring 3D points from different frames, we start by computing an uncertainty-weighted distance measure. If this distance is sufficiently small, we merge the two points and replace them with a single measurement with a reduced uncertainty.

The problem with this simple approach is that there may be many candidate matches for a given point, especially if one elongated uncertainty ellipsoid overlaps several other points. To reduce this problem, we limit merges to points whose uncertainty ellipsoid major axes are nearly parallel and which also meet the previous distance criteria. In practice, we make the merging step simpler by re-projecting the 3D locations and their uncertainties into the camera image plane. Two points are

[4] Our *object-centered* coordinate reference frame is fixed to the top of the turntable and rotates with it.



(a) (b) (c) (d)

Figure 5: Final merged data from *assam* sequence: (a) top view, (b) oblique view, (c) front view, (d) side view. The wireframe bounding box is just a 3" reference cube aligned with the turntable top.

merged if their image plane centers lie within a small distance of each other (say, 1/2 pixel) and their depths overlap sufficiently (using a 1-D version of the uncertainty-weighted distance). The thresholds for merging points are set high enough so that neighboring measurements from the same frame are not merged (we want our final model to be at least as accurate as the input image) but low enough so that oversampling (the density of 3D points per image pixel) is not too great.

Figure 5 shows the results of processing a complete 250 image sequence. The data is shown as isolated points from 4 different views. As you can see, the overall shape of the tea tin is recovered well, although the surface is not very smooth. In general, these results are not as accurate as those obtained with our edge-based surface reconstruction technique, due mainly to the narrower baselines involved (the contour-based algorithm typically uses 5–7 frames for each reconstructed contour, whereas two-frame stereo was used for these results). However, the algorithm works better in highly textured areas where reliable edge extraction is difficult.

The algorithm we use for merging 3D points is related to other recent range-data merging algorithms [52, 70, 26, 16]. Unlike these algorithms, it explicitly models the 3D uncertainty associated with range measurements (as does [29]), which produces estimates which are more statistically optimal. Unlike these algorithms, however, it does not produce a single coherent surface. To produce such an estimate, we developed a novel topologically flexible surface interpolation algorithm, which we present next.

5 Oriented particle-based surface modeling

The problem of fitting 3D models to sparse range data has been extensively studied in computer vision. Popular models include include generalized cylinders [1], superquadrics [41], and triangular meshes [11]. Physically-based models, which have internal deformation energies and can be fitted through external forces to 2D images and 3D range data have also been widely studied [67, 18, 66, 38].

Unfortunately, all of these models require the user to specify the topology (and often the rough shape) of the object ahead of time. To overcome this limitation, we have developed a surface modeling and fitting system based on the concept of *oriented particles* (also known as surface elements, or *surfels*) [62, 64, 63].

Our surface modeling and reconstruction method has two components. The first is a dynamic particle system which discovers topological and geometric surface structure implicit in the data. The second component is an efficient triangulation scheme which connects the particles into a continuous global surface model that is consistent with the particle structure. The evolving global model supports the automatic extension of existing surfaces with few restrictions on connectivity, the joining of surfaces to form larger continuous surfaces, and the splitting of surfaces along arbitrary discontinuities as they are detected.

The most novel feature of our approach to surface reconstruction is the use of a *molecular dynamics* simulation in which particles interact through long-range attraction forces and short-range repulsion forces. In our system, each particle has an associated surface normal. To control the behavior of these particles, we designed new interaction potentials which favor locally planar or locally spherical arrangements of particles [62]. Thus, the oriented particles support smoothness constraints similar to those inherent in the deformation energies of physically-based surface models. When reconstructing an object of arbitrary topology, the particles can be made to "flow" over the data, extracting and conforming to meaningful surfaces.

Our oriented particles were used as the basis for an interactive surface modeling system [62]. With this system, users can spray collections of points into space to form elastic sheets, shape them using deformation tools, and then freeze the surfaces into the desired final shape. They can create any desired topology with this technique. For example, they can deform a flat sheet into an object with a protrusion and then change the topology

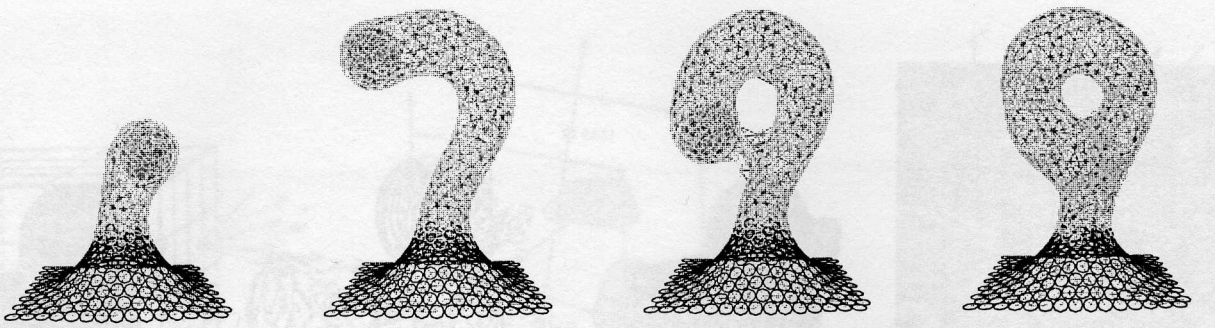


Figure 6: Forming a complex object. The initial surface is deformed upwards and then looped around. The new topology (a handle) is created automatically.

to create a looped handle (Figure 6). Forming such surfaces with traditional spline patches is a difficult problem that requires careful attention to patch continuities [36]. Other examples of modeling operations in this system include "cold welding" two surfaces together by abutting their edges, "cutting" surfaces with a knife-like constraint tool, and "creasing" surfaces by designating certain particles to be *unoriented* [62].

Another important application of our oriented particle systems is the interpolation and extrapolation of sparse 3-D data. This is a particularly difficult problem when the topology of the surface to be fitted is unknown. Oriented particles provide a solution to this problem by extending the surface out from known data points. This technique is particularly useful for interpolating the sparse position measurements obtained using the techniques described in this paper.

The basic components of our particle-based surface extension algorithm are two heuristic rules controlling the addition of new particles. These rules are based on the assumption that the particles on the surface are in a near-equilibrium configuration with respect to the flatness, bending, and inter-particle spacing potentials.

The first (*stretching*) rule checks to see if two neighboring particles have a large enough separation between them to add a new particle. If two particles are separated by an appropriate distance, we create a candidate particle at the midpoint. The second (*growing*) rule allows particles to be added in all directions with respect to a particle's local tangent plane. The rule is generalized to allow a minimum and maximum number of neighbors and to limit growth in regions of few neighboring particles, such as at the edge of a surface.

With these two rules, we can automatically build a surface from collections of 3-D points. We create particles at each sample location and fix their positions and orientations. We then start filling in gaps by growing particles away from isolated points and edges. After completing a rough surface approximation, we can release the original sampled particles to smooth the final surface, thereby eliminating excessive noise. If the set of data points is reasonably distributed, this approach will result in a smooth continuous closed surface.

In conjunction with fitting the surface, we can estimate local differential geometric quantities such as the principal directions and minimum and maximum curvatures. This can be achieved by simply adding an extra potential function that induces a torque around the local z axis and which forces the x and y axes to align themselves in the directions of minimum and maximum curvature [63]. The resulting system of oriented particles resembles the collection of interacting Darboux frames used by Sander and Zucker [46].

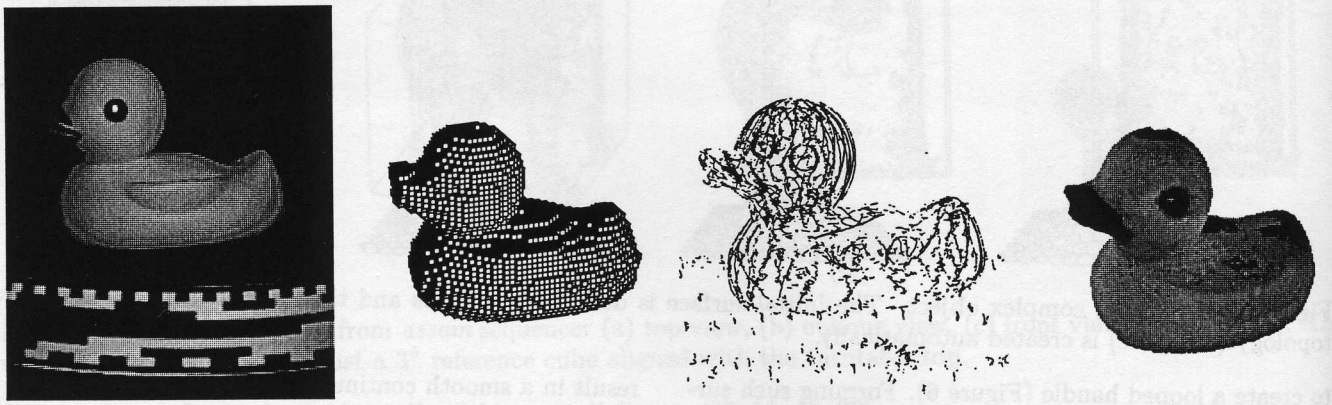
To summarize, oriented particles (surfels) allow us to interpolate and smooth both sparse and dense 3D data, without the need for any prior shape specification or user intervention (see [27, 23] for alternative approaches). Our system is flexible and general enough to model elastic surfaces of arbitrary topology with creases and discontinuities, and has also been extended to produce local measures of intrinsic geometry such as principal curvatures. Limitations of our approach include the large amounts of computation required to simulate the particle dynamics, the difficulty of tuning these dynamics, and occasional undesirable side-effects, such as the tendency of neighboring of surfaces to stick together.

6 Inverse texture mapping

The final step in creating a photorealistic model from imagery is to recover the color distribution over the surface of the object. In computer graphics, this is often called *texture map recovery* or *extraction*. A more general problem is to estimate the bidirectional reflectance distribution function (BRDF) at each pixel [47, 48].

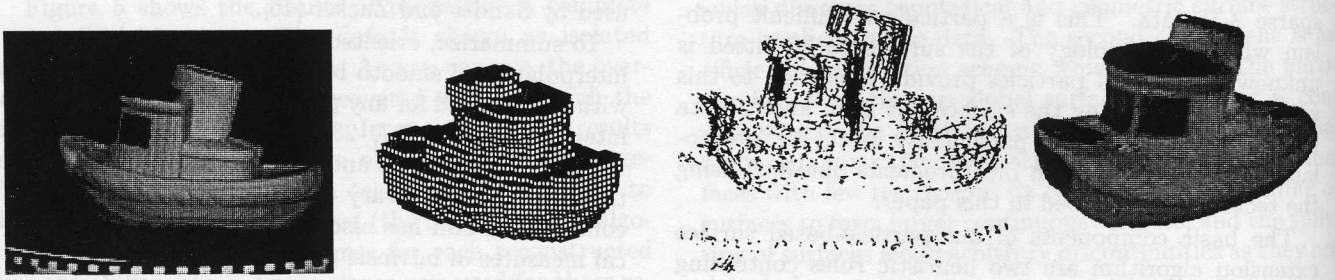
A simple way to obtain such distributions is to first segment the object or scene into piecewise planar regions, and to then project the input images onto these planar regions [13, 4, 17]. Another possibility is to use a triangulated surface model, and to project the images onto the triangles [7], or to just estimate the colors at the vertices (Figures 7–8).

Regardless of the approach used, the visibility of surfaces in each image must first be computed. The most convenient way to do this is to use a *z-buffer* or *item*



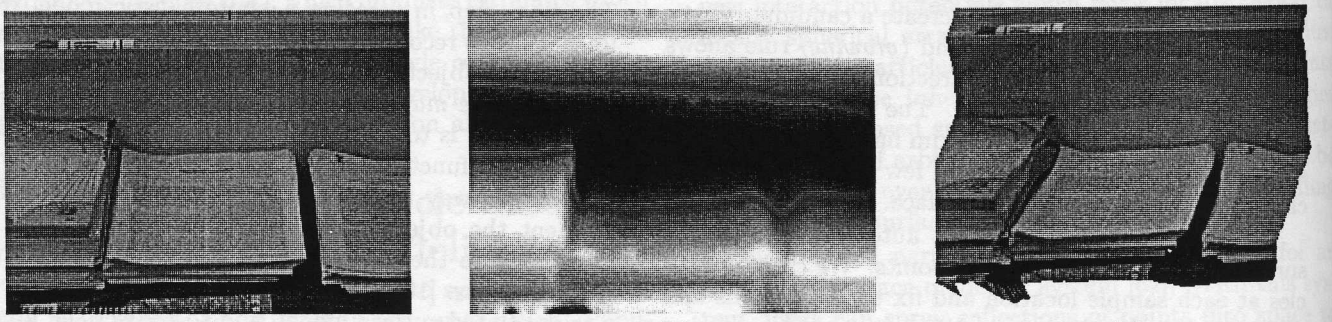
(a) (b) (c) (d)

Figure 7: Photometric recovery example (duck sequence): (a) input image from duck sequence, (b) octree model, (c) 3-D edges, (d) inverse texture-mapped model.



(a) (b) (c) (d)

Figure 8: Photometric recovery example (tug sequence): (a) input image from duck sequence, (b) octree model, (c) 3-D edges, (d) inverse texture-mapped model.



(a) (b) (c)

Figure 9: Projective depth recovery example—table with stacks of papers: (a) input image, (b) intensity-coded depth map (dark is farther back) (c) texture-mapped surface seen from novel viewpoint.

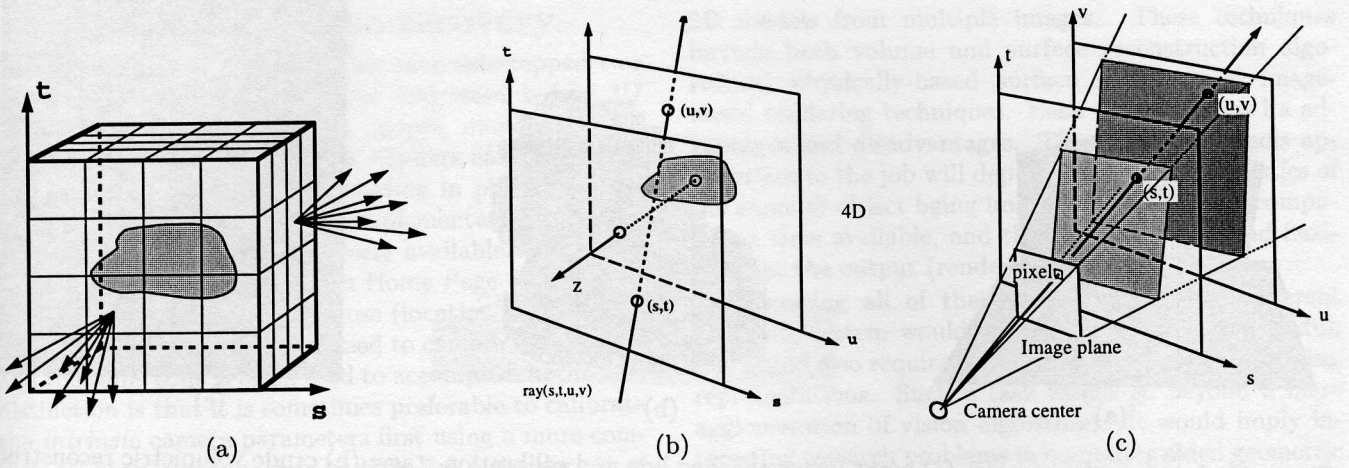


Figure 10: The Lumigraph: (a) the cube surface captures all light rays emanating from the object; (b) the 4D (s,t,u,v) parameterization of the rays; (c) computing the 4D coordinate of a pixel seen from a novel view.

buffer [72]. Furthermore, the contribution of each input pixel to the final color or texture map should be weighted so that pixels viewed at oblique angles contribute less. A convenient way to do this is to use the dot product between the camera viewing direction (or the ray corresponding to a pixel) and the surface normal. Finally, shading effects should be compensated for.

In the examples shown in Figures 7–8, we use a simple Lambertian shading model, with the light source placed directly behind the camera. While the recovered texture maps look plausible, they still suffer from visible artifacts. The first of these is a general darkening of colors near the edges of the objects' visible regions (e.g., the back of the duck, the top of the tugboat). This could be due to a failure of the Lambertian model, non-linearities in the image acquisition process, or simply a lack of information in highly compressed and poorly lit areas.

The second major problem is that the recovered color distributions (textures) are much blurrier than the original images. This is mostly due to residual inaccuracies in the recovered object's shape and possible miscalibrations in the acquisition system. Two approaches could be used to solve this problem. The first is to perturb the shape model in order to better align all of the input images [22]. A second approach is to simply deform each input image to better match the current re-rendered model, which should also compensate for other unmodeled sources of mis-registration errors. We are currently investigating these possibilities.

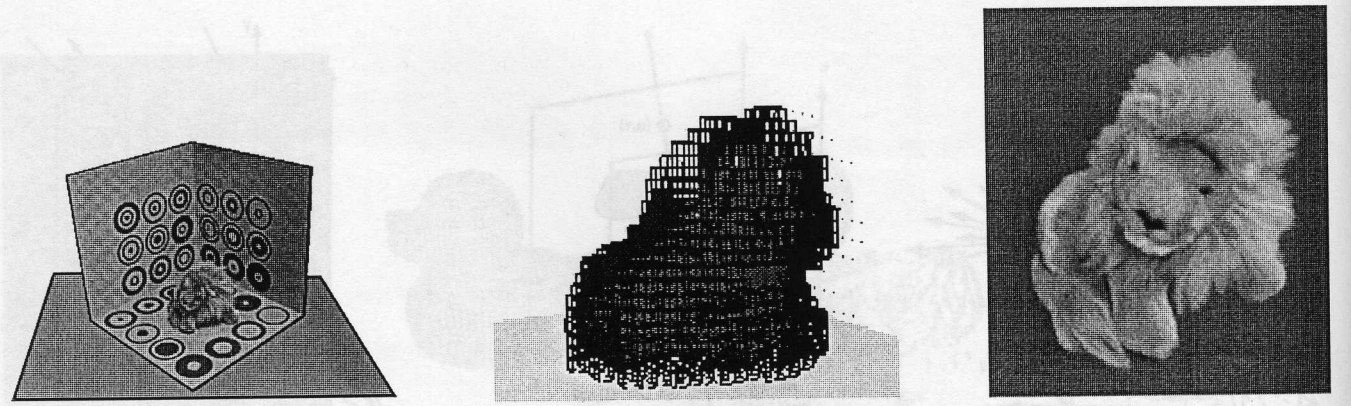
7 Beyond 3D models: image-based rendering

The techniques and algorithms presented thus far all have one goal in common: the reconstruction of detailed, photorealistic 3D models of arbitrary topology. My interest in this area was first stimulated while I was working on the recovery of accurate depth maps from multiframe stereo [37]. While we were able to obtain good depth maps and use these to re-project the scene into novel views, it quickly became apparent that only a very limited range of novel viewpoints could be supported with this representation.

Ironically, there has been a resurgence of interest in both the computer graphics and computer vision communities in such 2-$\frac{1}{2}$D representations. In computer graphics, this idea is called *image-based rendering* or *view interpolation* [14, 39]. The basic idea is to pre-render (or alternatively, to capture with photographic means) a number of views of a complex scene. At rendering time, these images (views) are then combined using *morphing* techniques [9, 50], i.e., by warping and blending images. The advantage of this approach is that complex lighting simulations are eliminated (or pre-computed), thus speeding up the rendering.

In computer vision, these representations are sometimes called *visual scene representations* [2]. The ideas here are similar, but with an emphasis on computing the necessary correspondences between input images to create large *mosaics* of scenes and to extract the required depth maps, often without any explicit knowledge of camera calibration or pose [31, 60]. One way to address this latter problem is to reconstruct a *projective* representation of the scene, i.e., one which is related to the true Euclidean structure through an unknown collineation [20, 25, 40, 59].

When combined with visual scene reconstruction, this often results in estimators based on first computing a planar homography, and then a residual parallax motion [31, 49, 58]. Figure 9 shows the depth map recovered using our plane + parallax technique, and also a novel view rendered by mapping one image onto the depth map. Note that the depth map is only known up to an arbitrary scale and planar offset. To generate the novel



(a) (b) (c)

Figure 11: Lumigraph example: (a) input image showing lion and calibration stage; (b) crude volumetric reconstruction of lion's shape; (c) novel image rendered from the Lumigraph.

view in Figure 9, we simply set the scale by hand to a reasonable value. Techniques for automatically deriving the proper mapping to novel views from additional correspondences have been developed [33].

The most recent work in image-based rendering suggests that correspondence between views may not be necessary if the spatial sampling of views is sufficiently high [34, 24]. The fundamental observation is that a set of images sampled on a surface enclosing any object (or conversely, surrounding some convex region of free-space) is sufficient to re-synthesize the appearance of the object or scene from any novel view in the free-space (Figure 10a). A convenient parametrization for the rays emanating from this volume uses two planes (Figure 10b). To compute the appearance of the object from a novel view, it then suffices to compute the 4D (s, t, u, v) index of each new pixel, and to look up the appropriate color in the 4D data set (Figure 10c).

Unfortunately, while this works perfectly well in theory, in practice the 4D light field must be sampled discretely, resulting in a finite number of images (say a 32×32 array of 256×256 images). When looking up a new ray value, we must interpolate this value from the neighboring discrete samples. If the neighboring samples are too far apart, then the blending of adjacent images will result in *ghosting* or double images, even if multidimensional (quadralinear) interpolation is used. This can be mitigated somewhat by pre-blurring the images (low-pass filtering in 4D space to remove aliasing), but results in blurry output images.

A sensible way to overcome this problem is to reintroduce some notion of depth or disparity into the 4D representation. Fortunately, we do not need to have a reliable depth estimate for each sample. A low-resolution geometric model is sufficient to remove most of the visible artifacts, and efficient rendering algorithms have been developed to exploit such models within a traditional graphics pipeline [24].

Figure 11 shows an input image from our Lumigraph system, a coarse geometric model computed using the volumetric technique described in Section 2, and a final re-synthesized image. Notice how this scene would be very difficult to model using a traditional 3D model because of the extremely fine detail in the lion's hair. The Lumigraph has similar advantages with respect to 3D models when it comes to capturing subtle photometric effects such as specularities and interreflections [24].

Our work on the Lumigraph suggests that there is a continuum of models and representations available for photorealistically representing the 3D world. At one end is the pure light field representation, which is simply a large collection of images taken from known vantage points. The rendering of such representations requires no correspondences, but does require very high sampling rates, and hence large storage costs (but see [34] on how these data sets may be compressed). At the other end are traditional texture-mapped 3D models. If appropriate reflectance models can also be estimated, then a standard graphics pipeline can be used to obtain photorealistic renderings (ignoring issues such as interreflections).

In between are the more interesting hybrid models. The Lumigraph uses a low-resolution representation of shape to perform a warping of the light field images in order to obtain better renderings at lower sampling rates. Visual scene representations and view interpolation techniques associate a depth value with each pixel in order to support re-projection through warping and blending. The rendering of such models on current graphics architectures (or in software) may be slow unless simplifying assumptions are made [14, 39]. Finally, view-dependent texture maps offer the possibility of increased realism when combined with traditional 3D models [17]. An alternative to view-dependent texture maps are full per-pixel BRDF models, but this research is still in its infancy [47, 48].

8 Camera pose recovery

Throughout our presentation, we have sidestepped two important issues which must be addressed before 3D models can be extracted from images, namely camera calibration and pose estimation. Camera calibration is a well-established discipline originating in photogrammetry [51]. Reg Willson's public implementation of Tsai's calibration algorithm [69] is freely available and widely used (see the Computer Vision Home Page at Carnegie Mellon University). Camera pose (location and orientation) estimation is intimately tied to camera calibration, and the same code is often used to accomplish both. The distinction is that it is sometimes preferable to calibrate the *intrinsic* camera parameters first using a more complex calibration setup , and to then use a simpler set of markers for determining the *extrinsic* (pose) parameters [19].

The most interesting design decision with respect to calibration and pose estimation is often the choice of stage and markers. For my research on shape from rotation, I used a simple hexagonal pattern affixed to the top of the turntable shown in Figure 4a [54]. For our Lumigraph project, we used a three-walled stage painted in blue (to support blue-screen matte extraction) with circular calibration targets (Figure 11a) [24]. In outdoor scenarios such as special effects for the film industry, it is common to place golf ball at known locations in the scene which are later manually removed from the live footage.

An alternative to marker-based pose estimation is to use structure from motion, i.e., to track a collection of features through several frames, and to then simultaneously reconstruct the point and camera positions. Many different structure from motion algorithms have been developed [35, 68, 73, 59]. Two of the more recent algorithms, which have been incorporated into full texture-mapped 3D model extraction systems are [4, 7]. The latter work is particularly interesting since it uses current structure and motion estimates to validate tracking hypotheses, and a robust (RANSAC) technique to throw out outliers.

Unfortunately, structure from motion algorithm are notoriously bad at estimating the true pose and structure unless a very wide range of views or baseline is used [61]. In some applications, however, such as the merging of live video and graphics, it may not be important to get a fully Euclidean reconstruction (unlike, say, for 3D model construction). Techniques which use stronger prior models of structure, e.g., building or room models, do not suffer from this problem as much, but can only be applied in restricted situations [8, 17].

9 Discussion

In this paper, we have surveyed a number of techniques for reconstructing and rendering photorealistic 3D models from multiple images. These techniques include both volume and surface reconstruction algorithms, physically-based surface models, and image-based rendering techniques. Each technique has its advantages and disadvantages. The selection of tools appropriate to the job will depend on the characteristics of the scene or object being imaged, the amount of computation time available, and the desired quality and flexibility at the output (rendering) stage.

Integrating all of these techniques into a coherent modeling system would be very interesting and useful, but would also require a careful design of the underlying representations. Such a task would go beyond a mere agglomeration of vision algorithms. It would imply interesting research problems in computer aided geometric design and multi-sensor fusion. Of course, each individual component (e.g., stereo matching) still has a lot of room for improvement, and we expect to see interesting future contributions in all of these areas.

Envisioning useful and exciting application of image-based modeling is another interesting way to predict their eventual deployment and to select research problems. Some possibilities include:

- *CAD/CAM*: the acquisition of shape and color models for design refinement and manufacturing
- *3D fax*: displaying the acquired models at a remote location or reconstructing them using stereolithography or other means
- *architecture*: constructing building and site models from photographs, with applications to design and remodeling, and even home sales
- *biomedical*: acquiring anatomical models (e.g., faces) for surgical planning and visualization
- *special effects (FX) and virtual studios*: merging live video or film with 3D graphics
- *3D world building*: for virtual environments (3D chat systems and games) and virtual travel
- *3D avatar construction*: head and body models to populate virtual environments, or to support a "3D videophone"

As these applications suggest, image-based 3D modeling is likely to be widely used in the future, and is bound to remain a fascinating area for future research.

Acknowledgements

It has been my honor to have worked over the last ten years with an outstanding collection of colleagues and collaborators, including Larry Matthies, Takeo Kanade, Demetri Terzopoulos, David Tonnesen, Richard Weiss, Sing Bing Kang, James Coughlan, Heung-Yeung Shum, Steven Gortler, Radek Grzeszczuk, and Michael Cohen, among others. The research described in this paper is a testament to their creativity, insight, and hard work.

References

[1] G. J. Agin and T. O. Binford. Computer description of curved objects. *IEEE Trans. Computers*, C-25(4):439–449, April 1976.

[2] *IEEE Work. Representations of Visual Scenes*, Cambridge, MA, June 1995.

[3] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Int'l J. Computer Vision*, 2(3):283–310, January 1989.

[4] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. Pattern Analysis Machine Intelligence*, 17(6):562–575, June 1995.

[5] H. H. Baker. Three-dimensional modeling. In *Int'l Joint Conf. Artificial Intelligence*, pages 649–655, 1977.

[6] B. G. Baumgart. Geometric modeling for computer vision. Technical Report AIM-249, Artificial Intelligence Laboratory, Stanford University, October 1974.

[7] P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *European Conf. Computer Vision*, volume 2, pages 683–695, Cambridge, England, April 1996. Springer-Verlag.

[8] S. Becker and V. M. Bove. Semiautomatic 3-D model extraction from uncalibrated 2-d camera views. In *SPIE Vol. 2410, Visual Data Exploration and Analysis II*, pages 447–461, San Jose, CA, February 1995.

[9] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH'92)*, 26(2):35–42, July 1992.

[10] P. J. Besl and R. C.. Jain. Three-dimensional object recognition. *Computing Surveys*, 17(1):75–145, March 1985.

[11] J.-D. Boissonat. Representing 2D and 3D shapes with the Delaunay triangulation. In *Seventh Int'l Conf. Pattern Recognition*, pages 745–748, Montreal, July 1984.

[12] E. Boyer and M. O. Berger. 3D surface reconstruction using occluding countours. *Int'l J. Computer Vision*, 1997.

[13] I. Carlbom et al. Modeling and analysis of empirical data in collaborative environments. *Communications of the ACM*, 35(6):74–84, April 1992.

[14] S. Chen and L. Williams. View interpolation for image synthesis. *Computer Graphics (SIGGRAPH'93)*, pages 279–288, August 1993.

[15] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *Int'l J. Computer Vision*, 9(2):83–112, November 1992.

[16] B. Curless and M. Levoy. A volumetric method for building complex models from range images. Proc. SIGGRAPH'96 (New Orleans), pages 303–312, August 1996.

[17] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. Proc. SIGGRAPH'96 (New Orleans), pages 11–20, August 1996.

[18] H. Delingette, M. Hebert, and K. Ikeuichi. Shape representation and image segmentation using deformable surfaces. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 467–472, Maui, Hawaii, June 1991.

[19] O. Faugeras. *Three-dimensional computer vision: A geometric viewpoint.* MIT Press, Cambridge, MA, 1993.

[20] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *European Conf. Computer Vision*, pages 563–578, Santa Margherita Liguere, Italy, May 1992. Springer-Verlag.

[21] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis Machine Intelligence*, 13(9):891–906, September 1991.

[22] P. Fua and Y. G. Leclerc. Using 3–dimensional meshes to combine image-based and geometry-based constraints. In *European Conf. Computer Vision*, volume 2, pages 281–291, Stockholm, Sweden, May 1994. Springer-Verlag.

[23] P. Fua and P. Sander. Segmenting unstructured 3d points into surfaces. In *European Conf. Computer Vision*, pages 676–680, Santa Margherita Liguere, Italy, May 1992. Springer-Verlag.

[24] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. Proc. SIGGRAPH'96, pages 43–54, August 1996.

[25] R. Hartley, R. Gupta, and T. Chang. Estimation of relative camera positions for uncalibrated cameras. In *European Conf. Computer Vision*, pages 579–587, Santa Margherita Liguere, Italy, May 1992. Springer-Verlag.

[26] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *European Conf. Computer Vision*, volume 1, pages 117–126, Cambridge, England, April 1996. Springer-Verlag.

[27] W. Hoppe et al. Surface reconstruction from unorganized points. *Computer Graphics (SIGGRAPH'92)*, 26(2):71–78, July 1992.

[28] P. J. Huber. *Robust Statistics.* John Wiley & Sons, New York, 1981.

[29] A. E. Johnson and S. B. Kang. Registration and integration of textured 3-d data. In *Int'l Conf. Recent Advances in 3-D Digital Imaging and Modeling*, Ottawa, May 1997.

[30] S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 364–370, San Francisco, June 1996.

[31] R. Kumar, P. Anandan, and K. Hanna. Shape recovery from multiple views: a parallax based approach. In *Image Understanding Work.*, pages 947–955, Monterey, CA, November 1994.

[32] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Analysis Machine Intelligence*, 16(2):150–162, February 1994.

[33] S. Laveau and O. D. Faugeras. 3-d scene representation as a collection of images. In *Int'l Conf. Pattern Recognition*, volume A, pages 689–691, Jerusalem, Israel, October 1994.

[34] M. Levoy and P. Hanrahan. Light field rendering. Proc. SIGGRAPH'96 (New Orleans), pages 31–42, August 1996.

[35] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[36] Charles Loop and Tony DeRose. Generalized B-spline surfaces of arbitrary topology. *Computer Graphics (SIGGRAPH'90)*, 24(4):347–356, August 1990.

[37] L. H. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *Int'l J. Computer Vision*, 3:209–236, 1989.

[38] T. McInerney and D. Terzopoulos. A finite element model for 3D shape reconstruction and nonrigid motion tracking.

In *Int'l Conf. Computer Vision*, pages 518–523, Berlin, May 1993.

[39] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH'95)*, pages 39–46, August 1995.

[40] R. Mohr, L. Veillon, and L. Quan. Relative 3D reconstruction using multiple uncalibrated images. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 543–548, New York, June 1993.

[41] A. P. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28(3):293–331, May 1986.

[42] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40:1–29, 1987.

[43] M. Rioux and T. Bird. White laser, synced scan. *IEEE Computer Graphics and Applications*, 13(3):15–17, May 1993.

[44] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, New York, 1976.

[45] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1989.

[46] P. T. Sander and S. W. Zucker. Inferring surface trace and differential structure from 3-D images. *IEEE Trans. Pattern Analysis Machine Intelligence*, 12(9):833–854, September 1990.

[47] Y. Sato and K. Ikeuchi. Reflectance analysis for 3d computer graphics model generation. *Graphical Models and Image Processing*, 58(5):437–451, September 1996.

[48] Y. Sato and K. Ikeuchi. Object shape and reflectance modeling from observation. Proc. SIGGRAPH'97 (Los Angeles), August 1997.

[49] H. S. Sawhney. Simplifying motion and structure analysis using planar parallax and image warping. In *Int'l Conf. Pattern Recognition*, volume A, pages 403–408, Jerusalem, Israel, October 1994.

[50] S. M. Seitz and C. M. Dyer. View morphing. Proc. SIGGRAPH'96 (New Orleans), pages 21–30, August 1996.

[51] Chester C. Slama, editor. *Manual of Photogrammetry.* American Society of Photogrammetry, Falls Church, Virginia, Fourth Edition, 1980.

[52] M. Soucy and D. Laurendeau. Multi-resolution surface modeling from multiple range views. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 348–353, Champaign, Ill., June 1992.

[53] P. Srivasan, P. Liang, and S. Hackwood. Computational geometric methods in volumetric intersections for 3D reconstruction. *Pattern Recognition*, 23(8):843–857, 1990.

[54] R. Szeliski. Shape from rotation. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 625–630, Maui, Hawaii, June 1991.

[55] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, July 1993.

[56] R. Szeliski. Image mosaicing for tele-reality applications. In *IEEE Work. Applications of Computer Vision*, pages 44–53, Sarasota, Florida, December 1994.

[57] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, pages 22–30, March 1996.

[58] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 194–201, Seattle, Washington, June 1994.

[59] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *J. Visual Communication and Image Representation*, 5(1):10–28, March 1994.

[60] R. Szeliski and S. B. Kang. Direct methods for visual scene reconstruction. In *IEEE Work. Representations of Visual Scenes*, pages 26–33, Cambridge, MA, June 1995.

[61] R. Szeliski and S. B. Kang. Shape ambiguities in structure from motion. In *European Conf. Computer Vision*, volume 1, pages 709–721, Cambridge, England, April 1996. Springer-Verlag.

[62] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics (SIGGRAPH'92)*, 26(2):185–194, July 1992.

[63] R. Szeliski, D. Tonnesen, and D. Terzopoulos. Curvature and continuity control in particle-based surface models. In *SPIE Vol. 2031 Geometric Methods in Computer Vision II*, pages 172–181, San Diego, July 1993.

[64] R. Szeliski, D. Tonnesen, and D. Terzopoulos. Modeling surfaces of arbitrary topology with dynamic particles. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 82–87, New York, New York, June 1993.

[65] R. Szeliski and R. Weiss. Robust shape recovery from occluding contours using a linear smoother. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 666–667, New York, June 1993.

[66] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Trans. Pattern Analysis Machine Intelligence*, 13(7):703–714, July 1991.

[67] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Rcovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, August 1988.

[68] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *Int'l J. Computer Vision*, 9(2):137–154, November 1992.

[69] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J. Robotics and Automation*, RA-3(4):323–344, August 1987.

[70] G. Turk and M. Levoy. Zippered polygonal meshes from range images. *Computer Graphics (SIGGRAPH'94)*, pages 311–318, July 1994.

[71] R. Vaillant and O. D.. Faugeras. Using extremal boundaries for 3-D object modeling. *IEEE Trans. Pattern Analysis Machine Intelligence*, 14(2):157–173, February 1992.

[72] H. Weghorst, G. Hooper, and D. P. Greenberg. Improved computational methods for ray tracing. *ACM Trans. Graphics*, 3(1):52069, January 1984.

[73] J. Weng, T. S. Huang, and N. Ahuja. *Motion and Structure from Image Sequences.* Springer-Verlag, Berlin, 1993.

[74] J. Y. Zheng. Acquiring 3-D models from sequences of contours. *IEEE Trans. Pattern Analysis Machine Intelligence*, 16(2):163–178, February 1994.