Appears in: Proc. CVPR'97, pp.53-59, 1997.

Shape from the Light Field Boundary

Kiriakos N. Kutulakos
kyros@cs.rochester.edu
Department of Computer Sciences
University of Rochester
Rochester, NY 14627-0226 USA

Abstract

Ray-based representations of shape have received little attention in computer vision. In this paper we show that the problem of recovering shape from silhouettes becomes considerably simplified if it is formulated as a reconstruction problem in the space of oriented rays that intersect the object. The method can be used with both calibrated and uncalibrated cameras, does not rely on point correspondences to compute shape, and does not impose restrictions on object topology or smoothness.

1. Introduction

There has been considerable interest recently in representing 3D objects in terms of the rays of light leaving their surface (e.g., the light field [9]). One common feature of these ray-based representations is that they contain sufficient information to synthesize arbitrary views of an object, yet they can be built from multiple images without computing point correspondences or 3D shape. Little is known, however, about what implicit 3D shape information is captured by these representations, and about whether it is possible to convert this information into an explicit 3D model.

This paper addresses these questions in the context of recovering shape from a dense sequence of silhouette images of an unknown curved object. We show that the problem of recovering shape from silhouettes becomes considerably simplified if it is formulated as a reconstruction problem in the space of oriented rays that intersect the object. In particular, we show that by rotating an object about a single axis, we can compute the shape of planar slices of the object by (1) computing convex hulls on the oriented projective ray space T^2, and (2) computing planar arrangements of lines. The only requirements are that (1) at least three points rigidly attached to the object can be tracked across frames, (2) the object's projection can be separated from the background, and (3) the rotation of the object has a specific relationship with the camera geometry. The method can be used with both calibrated and uncalibrated cameras, does not rely on point correspondences to compute shape, and does not impose restrictions on object topology or smoothness.

The support of the National Science Foundation under Grant No. CDA-9503996, of the University of Maryland under Subcontract No. Z840902, and of Honeywell under Research Contract No. 304931455 is gratefully acknowledged.

Very little attention has been paid in the computer vision literature to ray-based representations of shape. These representations have been studied exclusively in the context of the Hough transform [6, 15] and have been traditionally used for detecting shapes in images. Our purpose is to show that these representations become a powerful tool for recovering 3D shape because they describe objects in terms of quantities (optical rays) that can be extracted directly even from a single image. Our work combines elements from previous work on non-metric scene reconstruction [10], silhouette-based shape recovery [4] and epipolar plane image analysis [1], and motivates the use of oriented projective geometry [8, 12] and convex duality theory [3, 12] for studying shape recovery in ray space.

Our approach is based on the observation that a planar slice of a 3D object can be represented implicitly by the light field boundary, which is the manifold of rays that "graze" the slice. This implicit representation can be converted into an explicit shape description by computing the envelope of the light field boundary, and samples of the light field boundary can be readily obtained from the optical rays through the silhouette.

The crucial issue that one must address is how to reconstruct the light field boundary from a collection of samples. The key contribution of this paper is to show that by representing rays on the oriented projective sphere T^2 and by exploiting results from convex duality theory, we can reconstruct the light field boundary from a large set of images and samples by manipulating *entire sets of rays* rather than individual (and possibly noisy) ray samples. We provide a detailed geometrical analysis of the problem and present experimental results that involve recovering shape from a few thousand silhouette images.

2. Shape from Silhouettes

Our approach attempts to overcome a number of difficulties encountered in previous silhouette-based shape recovery methods. These methods fall in roughly two categories, surface-based and volumetric. Surface-based methods compute local surface shape (e.g., curvature) by establishing correspondences between curves on the silhouette across a small number of frames and by computing the envelope of optical rays through corresponding points [4]. While shape can be recovered accurately under certain conditions [2, 16], surface-based methods pose several difficulties:

- **Tracking curves across frames:** It is impossible to guarantee the validity of inter-frame curve correspondences even for simple shapes (Figure 1(a)). This inevitably leads to wrong reconstructions.
- **Detecting & handling degenerate cases:** Shape computations are degenerate when the surface is flat or has creases. Even though these cases are difficult to detect in practice, accurate reconstruction relies on this ability [14, 17].
- **Handling dense image sequences:** Local shape cannot be computed when optical rays through corresponding curves are almost coincident. Reconstruction accuracy can therefore degrade as the density of the image sequence increases.
- **Using uncalibrated cameras:** Surface-based methods have so far been formulated within a Euclidean framework and cannot be used when the camera is affinely- or projectively-calibrated [10].
- **Enforcing global shape constraints:** Surface-based methods rely on local shape computations that cannot impose global constraints such as convexity.
- **Building global surface models:** A post-processing step is needed to merge or reconcile local shape estimates [16].

Volumetric methods, on the other hand, incrementally refine a volumetric object representation by intersecting the volumes bounded by optical rays through each silhouette image [13]. Even though volumetric reconstruction does not involve curve tracking and produces a global object description, current methods require calibrated cameras and raise two additional issues:

- **Intersecting volumes accurately & efficiently:** The difficulty of computing volume intersections has forced analytic methods to rely on few input images [11] and has motivated the use of voxel-based representations of space that limit reconstruction accuracy to the size of individual voxels [13].
- **Recovering unexposed surface regions:** Volumetric methods cannot recover the shape of regions that project to occluding contour curves [2] that are not part of the silhouette.

As in surface-based methods, we recover shape by computing the envelope of families of rays (the light field boundary). Rather than relying on successive images to determine

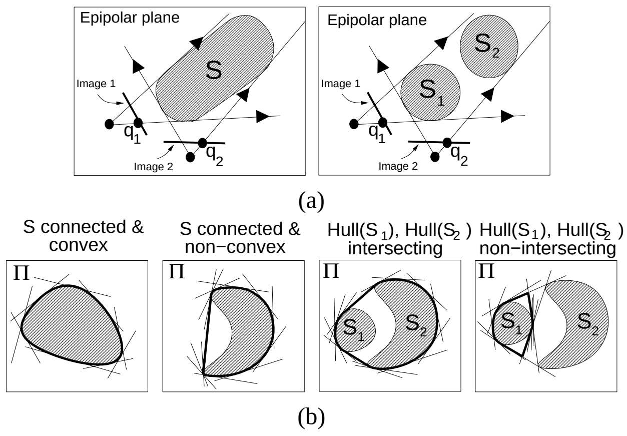

Figure 1. (a) It is impossible to decide locally whether or not the rays through silhouette points q_1 and q_2 touch the same connected region. If q_1, q_2 are always matched, the reconstructed shape may erroneously merge two or more regions (right figure) [16]. (b) Geometry of the visual hull. The visual hull's boundary is indicated by a thick solid line. Also shown are rays grazing the slice S. The visual hull in general consists of segments of S and segments of rays that are tangent to multiple points on S (rightmost figure).

these families, however, we map the set of rays defining the entire silhouette sequence to a set of points in ray space, and approximate the points in that set with curves whose envelopes are non-degenerate and consistent with all the image data. Here we show that this approach (1) leads to a weaker tracking problem that can always be resolved, (2) eliminates degeneracies that are hard to detect, (3) extracts accurate shape information from dense sequences that contain several hundred images, (4) can be used with both calibrated and uncalibrated cameras, (5) does not limit accuracy by requiring an *a priori* voxelization of 3D space, and (6) produces a globally-consistent 3D shape. Furthermore, even though this paper focuses on the problem of recovering shape from silhouettes, the approach can be generalized to recover unexposed surface regions from occluding contour curves that do not belong to the silhouette.

3. Viewing Geometry

Suppose that viewpoint is constrained to move on a plane Π whose intersection with the object is a non-empty slice S. We restrict our attention to the problem of computing the shape of S from its one-dimensional image on Π. This image consists of a collection of connected segments of object pixels and background pixels.

For every viewpoint, the optical rays defining the silhouette bound an infinite region on Π that contains the slice S. The *2D visual hull* of S is the intersection of these infinite regions for all viewpoints outside S's convex hull [7]. In general, the visual hull is a collection of convex supersets of the connected regions in S and represents the best approximation to S that can be obtained from silhouette images (Figure 1(b)). The visual hull of a slice does not change if the slice's connected regions are replaced by their convex hulls. We assume for simplicity that S contains a collection of convex connected regions, $S_1, \ldots, S_N$ which give rise to N distinct components in the visual hull.

Every component of the visual hull is the envelope [4] of the family of rays that graze region S_i and do not intersect the interior of S. Our goal is to compute this family for every connected component of the visual hull. We therefore need a way to map pixels in the image to rays on Π.

The mapping from pixels to rays depends on the camera model and on whether or not the camera is calibrated or uncalibrated. The specific choices do not affect our method. Here we assume that the camera is orthographic and uncalibrated [10], and determine the pixel-to-ray mapping by tracking the projection of three points on Π. Under these assumptions, Proposition 1 defines the mapping from pixels to rays in terms of image measurements:

Proposition 1 *Let p_1, p_2, p_3 be three non-collinear points defining an affine reference frame on Π, and let q_1, q_2, q_3, be their projections. The ray through pixel q satisfies the equation*

$$\left[\begin{array}{ccc} q_1 - q_0 & q_2 - q_0 & q_0 - q \end{array}\right] \cdot \left[\begin{array}{ccc} x & y & 1 \end{array}\right]^T = 0, \quad (1)$$

where $[x \;\; y \;\; 1]^T$ is a point along the ray.

Definition 1 (Pixel-to-Ray Mapping) *Pixel q is mapped to the vector of coefficients of Eq. (1).*

Shape is recovered in our approach by computing the envelope of a discrete set of rays. The envelope is an operator that maps an ordered sequence of rays to the ordered sequence of their pairwise intersection points. Given two linearly-independent and consecutive rays ρ_1, ρ_2 in the sequence, the homogeneous coordinates of their intersection are given by the vector product $\rho_1 \wedge \rho_2$. If ρ_1, ρ_2 are vectors defined by the Pixel-to-Ray Mapping, $\rho_1 \wedge \rho_2$ gives the affine coordinates of p with respect to the affine frame of p_1, p_2, p_3 [10]. We assume in the following that the projections of the affine basis points p_1, p_2, p_3 are known.

3.1. Oriented Projective Ray Representation

A key step in our shape recovery method is to map the optical rays that define the silhouette at a given viewpoint to points on the oriented projective sphere T^2 [12]. The resulting *oriented projective ray representation* plays a crucial role in our approach because, unlike other line representations commonly used in computer vision (e.g., the (r, θ) representation [15]), it guarantees that the light field boundary is always a collection of finite, non-degenerate, and convex curves in ray space. The representation is derived directly

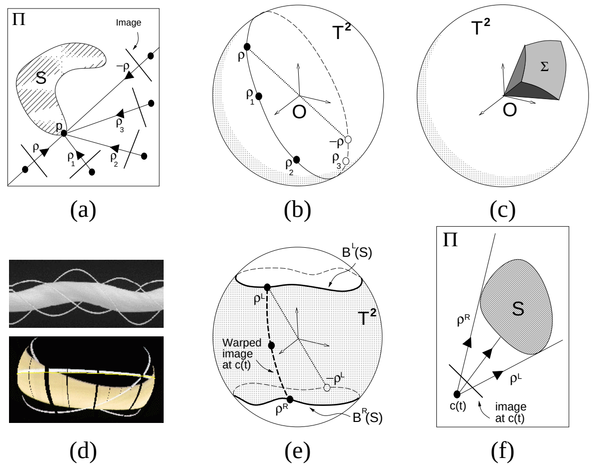

Figure 2. The oriented projective ray representation. (a) Rays on Π. Ray ρ separates Π into a left half-plane (shaded) and a right half-plane. (b) Rays on T^2. (c) Convexity on T^2. (d) *Top:* Spatiotemporal image of a slice of the object in Figure 4(a). *Bottom:* The warped spatiotemporal image. Since the image is warped onto a sphere, only part of it is visible. Note that the warped spatiotemporal trajectories of individual points are always great circles. The black vertical bands across the warped image correspond to viewpoints from which no images were taken. (e) The light field boundary of a connected region. The warped spatiotemporal image carves two empty areas on T^2. The upper area is the convex dual of S (Section 5). (f) Top view of plane Π.

from the Pixel-to-Ray Mapping and is general enough to represent rays with respect to Euclidean, affine, and projective reference frames [10]. The main elements of the representation are briefly described next.

The Pixel-to-Ray Mapping assigns a vector to every ray ρ through the silhouette. This vector is a *signed homogeneous vector*: vector $k\rho$ is equivalent to ρ when k is a positive constant, and ρ and $-\rho$ are *antipodal rays*, i.e., rays of opposite orientation. The space T^2 of signed homogeneous vectors is an oriented projective space homeomorphic to the unit sphere [12]. Every homogeneous vector can be mapped uniquely onto the unit sphere via the mapping

$$\rho \to \frac{\rho}{||\rho||}. \quad (2)$$

Together with the Pixel-to-Ray Mapping, Eq. (2) allows us to map pixels in the image to points on T^2 (Figure 2).

Since T^2 is an orientable manifold the notions of "orientation" and "convexity" are well-defined. Convexity in T^2 extends the familiar notion of convexity in R^3. In particular, Σ is a convex set on T^2 if and only if the conical volume defined by Σ and the origin of the unit sphere is convex in R^3 [12] (Figure 2(c)). We exploit this definition in Section 6 to compute the convex hull of rays on T^2.

4. The Light Field Boundary

The visual hull of a slice S is completely determined by the set of rays that graze S. We call this set the *light field boundary*, $\mathcal{B}(S)$, of S. Our goal is to compute the visual hull by first recovering a representation of the light field boundary from a collection of images. When mapped to the oriented projective sphere, $\mathcal{B}(S)$ bounds the set of all rays that intersect S. We exploit this property of $\mathcal{B}(S)$ to express it in terms of the spatiotemporal image of a rotating object.

Suppose that the viewpoint's motion is a rotation of at least 2π radians about a point within the convex hull of S, and let $I(t)$ be the spatiotemporal image at viewpoint $c(t)$. The Pixel-to-Ray Mapping "warps" the spatiotemporal image onto the oriented projective sphere by mapping every pixel of $I(t)$ to a ray on T^2 (Figure 2(d)). Since the viewpoint's motion guarantees that every ray that intersects S contains $c(t)$ for some t, the *warped spatiotemporal image* is a representation for the set of all rays that intersect S. Its boundary is equal to $\mathcal{B}(S)$ and is traced by the spatiotemporal trajectory of the silhouette.

The topology of the light field boundary depends on the connectivity of S. If S is connected, its warped spatiotemporal image is a strip and the light field boundary consists of two curves (Figure 2(e),(f)). In this case, every image of S contributes exactly two samples to its light field boundary, each belonging to a distinct curve of the boundary. The two curves of the light field boundary are distinguished by the way their rays bound S :

Definition 2 *The* left light field boundary, $\mathcal{B}^L(S)$, *is the set of all rays in* $\mathcal{B}(S)$ *that graze a region lying in their left half-plane. The* right light field boundary, $\mathcal{B}^R(S)$, *is the antipodal set of* $\mathcal{B}^L(S)$.

When S contains multiple connected regions $S_i, i = 1, \ldots, N$, its warped spatiotemporal image is a superposition of the warped spatiotemporal images of the individual regions (Figure 2(d)). As a result, the rays in $\mathcal{B}^L(S)$ that graze a region S_i and define the i-th component of the visual hull are disconnected segments of $\mathcal{B}^L(S_i)$.

Our goal is to recover the shape of every component of the visual hull by reconstructing the curve that defines its left light field boundary, $\mathcal{B}^L(S_i), i = 1, \ldots, N$. We achieve this by rotating the object by at least 2π radians on Π, mapping the pixels on the silhouette to rays on T^2, and performing three operations on the resulting set of rays:

- **Ray grouping:** Compute the number of regions in the slice and group together all rays of $\mathcal{B}^L(S_i)$ (Section 7).
- **Light field boundary reconstruction:** Approximate $\mathcal{B}^L(S_i)$ by a closed polygonal curve (Sections 5 and 6).
- **Visual hull reconstruction:** Recover the shape of the i-th component of the visual hull by computing the envelope of the polygonal curves that approximate $\mathcal{B}^L(S_i)$.

Light field boundary reconstruction is the basic step of our approach. Ray grouping reduces the problem of computing the visual hull of a slice that contains N regions to the case where S is a single connected region. Visual hull reconstruction amounts to intersecting adjacent rays in the polygonal curve that approximates $\mathcal{B}^L(S_i)$.

5. Shape from the Convex Dual

The basic step of our approach is to compute a polygonal curve that approximates the left light field boundary of a connected slice S. We use results from the theory of *convex duals* to perform this step using a simple convex hull operation on T^2. See [12] for an introduction to convex duality and [3] for applications in computational geometry.

The theory of convex duals studies convex sets in R^n and T^n. The basic tenet of the theory is to provide an alternate representation for a convex set, called the *convex dual*, in which the set is described in terms of the hyperplanes that bound it. The theory becomes an important tool for studying the left light field boundary because the left light field boundary is also the boundary of the convex dual. The results of the theory that serve as our starting point are summarized in the following two theorems. Theorem 1 shows that the convex duality mapping from sets of points to sets of rays preserves convexity, and Theorem 2 shows that the intersection of a set of half-planes can be expressed as a convex hull operation on T^2 [12]:

Definition 3 (Convex dual) *Let S be a convex region on the plane. The set of all rays whose left half-planes contain* S *is a set on* T^2 *and is called the* convex dual, $(S)^*$, *of* S.

Theorem 1 $(S)^*$ *is a convex set on* T^2.

Theorem 2 *Let* Σ *be a finite or infinite subset of the convex dual of a convex region* S, *and let* S' *be the intersection of all left half-planes of rays in* Σ. *The envelope of* Σ*'s convex hull is the boundary of* S'. *Furthermore,* $S \subseteq S'$.

The immediate consequence of Theorem 1 is that the left light field boundary of a connected slice is a convex curve on T^2. This provides a formal explanation for the light field's structure in Figure 2(e). The convexity of the light field boundary makes it particularly easy to approximate the boundary from a collection of ray samples—we simply need to compute the convex hull of those samples on the oriented projective sphere. Moreover, Theorem 2 tells us that (1) such an approximation is equivalent to approximating the visual hull of a slice by intersecting the left half-planes of all the available ray samples, and (2) as the number of available samples increases, the approximation converges monotonically to the slice's visual hull.

Theorems 1 and 2 emphasize the special significance of the Oriented Projective Ray Representation for representing rays on the plane, since this representation preserves convexity.[1] We exploit these theorems in the next section to reconstruct the left light field boundary of a connected slice.

6. Light Field Boundary Reconstruction

To apply Theorems 1 and 2 to the problem of reconstructing the left light field boundary of a connected slice we must answer two questions: (1) how to incorporate all the available information about the shape of the slice into the computation, and (2) how to compute the convex hull of a set of rays on T^2?

Every image of S contributes two ray samples ρ^L, ρ^R to its light field boundary, belonging to $\mathcal{B}^L(S)$ and $\mathcal{B}^R(S)$, respectively. Since $\mathcal{B}^L(S)$ and $\mathcal{B}^R(S)$ are antipodal images of each other, ρ^L and $-\rho^R$ are both samples of $\mathcal{B}^L(S)$. We can therefore guarantee that all samples contribute to S's shape by separating the samples of the left and right light field boundary into two sets Σ^L, Σ^R, respectively, and computing the convex hull of the set $\Sigma = \Sigma^L \cup \left(-\Sigma^R\right)$.

To compute the convex hull of Σ we exploit the relationship between Σ's convex hull and the 3D convex hull of $\Sigma^0 = \Sigma \cup \{(0,0,0)\}$. In particular, the convex hull of Σ^0 defines a conical polyhedron in R^3 whose apex is at the origin and whose intersection with T^2 is the convex hull of Σ. Hence, the rays in Σ that define its convex hull are the vertices of the conical polyhedron that share an edge with the origin. These considerations lead to the following algorithm for reconstructing the left light field boundary from a collection of ray samples:

Light Field Boundary Reconstruction Algorithm

Step 1 Let Σ^L, Σ^R be the ray samples of the left and right light field boundary, respectively.

Step 2 Compute the conical polyhedron that defines the 3D convex hull of $\Sigma^L \cup \left(-\Sigma^R\right) \cup \{(0,0,0)\}$.

Step 3 Let v_0 be the polyhedron vertex that corresponds to the origin, and let $v_1, \ldots, v_n$ be all vertices that are adjacent to v_0 in the polyhedron's adjacency graph.

Step 4 Return the vertices $v_1, \ldots, v_n$ linked into a closed chain according to the polyhedron's adjacency graph.

The above algorithm fails when the convex hull polyhedron is not conical, i.e., it does not have a vertex at the origin. This happens only if (1) the rays in Σ span a region greater than a hemisphere on T^2, or (2) all rays in Σ lie along a great circle. A fundamental property of convex sets on T^2 is that they cannot span a region greater than a hemisphere. Since Σ is a convex set, the algorithm will never fail due to this condition. The second condition corresponds to the degenerate case of a slice consisting of a single point on the plane. We detect this condition by checking whether the matrix that holds the coordinates of all rays in Σ is rank-deficient (Section 8). If the matrix is rank-deficient, the slice is defined by the singleton $\{(x,y,z)\}$, where $\{(x,y,z)\}$ is the direction of the normal of Σ's great circle. To compute the 3D convex hull in all other cases we use *qhull*, a package developed at the University of Minnesota Geometry Center.

[1]Wright *et al* [15] recently suggested a convex hull algorithm based on the Hough transform. Unfortunately, the non-linearity of the (r, θ) line representation used in their approach obscures the global structure of the Hough space. This results in sub-optimal convex hull algorithms and methods that rely on explicit discretization of the space.

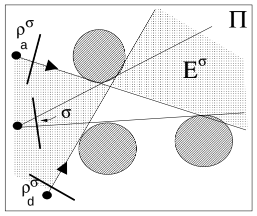

Figure 3. Rays $\rho_a^\sigma, \rho_d^\sigma$ are separating bitangents. The segment σ of background pixels shrinks to a point and disappears for viewpoints along the two separating bitangents. Rays ρ_a^σ and ρ_d^σ define an unbounded area E^σ on the plane that contains no object regions (shaded area).

7. Ray Grouping

The Light Field Boundary Reconstruction Algorithm assumes that all samples of $\mathcal{B}^L(S)$ belong to the left light field boundary of a single connected region. This assumption is violated when S contains multiple regions $S_1, \ldots, S_N$. It is therefore necessary to organize the ray samples of $\mathcal{B}^L(S)$ into N convex groups, each defining the left light field boundary of a single region.

To organize ray samples into groups we need to determine the number of regions in S. Below we show that we can compute N and perform the actual grouping operation by partitioning Π into cells defined by the slice's *separating bitangents*. The separating bitangents are the rays that belong to $\mathcal{B}^L(S)$ and contact the object at two distinct regions, with the regions of contact lying on opposite half-planes (Figure 3). These rays mark the appearance and disappearance of segments of background pixels in the image sequence.

7.1. Region counting

The separating bitangents partition plane Π into a finite collection of potentially-unbounded cells. To determine the number of regions in the slice we observe that this partitioning is maximal—no cell may contain more than one region and no region may span more than one cell. Hence, we need to find the cells that contain a region of the slice. We achieve this by pairing the separating bitangents in a way that determines the occupancy of all cells in the partitioning.

Suppose that a segment σ of background pixels first appears and then disappears during the object's rotation. The separating bitangents $\rho_a^\sigma, \rho_d^\sigma$ that mark the appearance and disappearance of σ, respectively, separate the plane into two parts, one of which, E^σ, is always empty (Figure 3). Theorem 3 shows that E^σ contains all the information we need to determine the occupancy of a cell C in the partitioning:

Theorem 3 *Suppose that the object is rotated by an angle of at least 2π radians and let $\sigma_1, \ldots, \sigma_n$ be the segments of background pixels that appear and disappear during the object's rotation. Cell C is empty if and only if it is contained in the union $\bigcup_{k \leq n} E^{\sigma_k}$.*

The theorem suggests that we can determine the number of object regions by simply rotating the object and tracking segments of background pixels through the image sequence. A rotation of at least 2π radians is required to detect all separating bitangents. These considerations lead to the following algorithm for counting the regions in a slice:

Region Counting Algorithm

```
Step 1  (Tracking) For every image $i$,
    a. find all segments of background pixels,
    b. for every background segment $\sigma$, find its corresponding
       segment in images $i-1$ and $i+1$; if no matching
       segment is found in image $i-1$, $\sigma$ appeared in image
       $i$; if no matching segment is found in image $i+1$, $\sigma$
       disappeared in image $i+1$
    c. if $\sigma$ appeared in $i$, define $\rho_a$ to be the ray through the
       midpoint of $\sigma$
    d. if $\sigma$ disappeared in $i+1$, define $\rho_d$ to be the ray through
       the midpoint of $\sigma$; compute the empty area $E$ defined
       by pair $(\rho_a, \rho_d)$ and add it to the list of empty areas
Step 2  (Partitioning) Compute the partitioning of $\Pi$ induced by
    the separating bitangents.
Step 3  (Counting) Set $N = 0$. For every cell $C$ in the partitioning,
    a. compute a point $p$ in the interior of $C$,
    b. if $p \in E$ for an empty area $E$ in the list, $C$ is empty;
       otherwise, set $N = N + 1$ and mark the cell occupied.
```

We establish segment correspondences in Step 1 by searching for overlapping background segments in consecutive images and by using sufficiently dense sequences. Step 2 is performed by computing the *arrangement* of N lines on the plane [3] using an implementation due to Goldwasser [5]. The complexity of this computation is $O(N^2)$.

7.2. Group assignment

Every object region S_i separates the samples of $\mathcal{B}^L(S)$ into two categories, those that belong to the convex hull of $\mathcal{B}^L(S_i)$ and those that do not. Since the Light Field Boundary Reconstruction Algorithm is based on a convex hull computation, we must ensure that the algorithm is applied only to the samples of $\mathcal{B}^L(S)$ that belong to $\mathcal{B}^L(S_i)$'s convex hull. Theorem 4 shows that this detection problem is easy to solve given the plane partitioning computed by the Region Counting Algorithm:

Theorem 4 *Suppose S_i is contained in cell C of the partitioning. Ray $\rho \in \mathcal{B}^L(S)$ intersects C if and only if ρ belongs to the convex hull of $\mathcal{B}^L(S_i)$.*

Theorem 4 tells us that we can use the occupied cells in the partitioning to "filter-out" all rays of $\mathcal{B}^L(S)$ that adversely affect the computation of the light field boundary of individual regions. Specifically, given a region S_i contained in cell C_i in the partitioning, it suffices to assign to S_i's group all rays that intersect C_i.

8. Experimental Results

To demonstrate the effectiveness of our shape recovery approach we have performed preliminary experiments with real data. The experimental setup is shown in Figure 4(a). In a typical run, we acquire over 2000 images during a single object rotation (approximately two minutes of video). Due to the amount of data involved, we process only a small number of rows (slices) of the entire image. For each frame, we track three affine basis points on a plane parallel to the plane of the slice, and separate the object from the background using a simple blue screen technique. The output of this stage consists of trajectories of the basis points and a spatiotemporal bitmap image indicating the object regions in each frame. The only tunable parameters in the entire system are those controlling background subtraction.

Results from an example run of the system are shown in Figures 4(b)-(g). Processing time was approximately 20 seconds on an SGI R4400 workstation. The example was chosen to illustrate several points. First, the object slice contains four connected regions and therefore ray grouping is necessary. Second, the three thin regions in the slice correspond to the case where the Light Field Boundary Reconstruction Algorithm is degenerate, i.e., when regions degenerate to a single point on the plane. Third, we made no effort to ensure regularity in the spacing between the 2040 images acquired, and in many instances, variations in the silhouette position due to noise were greater than the motion of the silhouette across frames.

Figure 4(b) shows the ray samples plotted on the oriented projective sphere and Figures 4(d),(e) show results of the final reconstruction. Reconstruction accuracy can be evaluated in terms of reprojection errors, i.e., the distance between the silhouette predicted by the reconstructed region and the silhouette in the spatiotemporal image (Figure 4(f)). The average reprojection error across the entire sequence is 2.5 pixels. The large spikes correspond to viewpoints where the silhouette of the duck region merges with one of the silhouettes of the thin regions. When all differences between actual and predicted silhouettes above 5 pixels are removed,

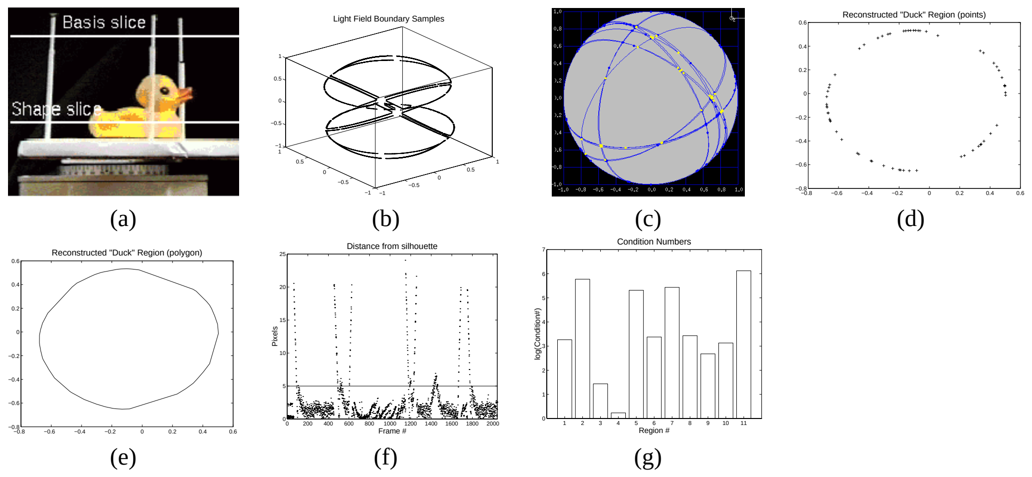

Figure 4. (a) Experimental setup. The object's rotation plane is aligned with the image rows. The spatiotemporal image of the rotating slice was shown in Figure 2(d). (b) The samples of $\mathcal{B}^L(S)$. A total of 11395 sample rays were acquired. (c) The line arrangement computed by the Region Counting Algorithm, projected onto a hemisphere. Sixteen bitangents were detected, partitioning the plane into 133 cells. Eleven of these cells were classified as object regions. (d),(e) Reconstruction of the "duck" region. The polygon defining the region has 64 vertices. (f) Reprojection errors. (g) Condition numbers of all ray groups. Group 4 corresponds to the "duck" region. The remaining groups can be identified as degenerate from their condition number.

the mean pixel error drops to 1.46 pixels and the variance to 1.05 pixels. These errors are comparable to the noise in the background subtraction process. Note that no smoothing was performed on the input or the reconstructed data.

One observation of particular significance is that reprojection errors are low even though a very small fraction of the available ray samples contributed to the region's shape (64 out of 3470). We do not have explicit control over the number of points that will be reconstructed on a region's boundary—this is effectively determined by the data. This suggests that a great deal of information about a region's shape is ignored by the convex hull computation, and that *even better* accuracies should be attainable from the shape information in the remaining ray samples. This topic is currently under investigation.

9. Concluding Remarks

The main limitations of our approach are that (1) simultaneous reconstruction of multiple 2D slices can be performed only under orthographic projection, and (2) the optical axis of the camera must lie in the motion plane. We are currently investigating how the approach can be extended to unrestricted camera geometries by considering the reconstruction of closed curves on the object's surface that are not necessarily planar.

References

[1] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1:7–55, 1987.

[2] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *IJCV*, 9(2):83–112, 1992.

[3] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[4] P. Giblin and R. Weiss. Reconstruction of surfaces from profiles. In *Proc. 1st ICCV*, pages 136–144, 1987.

[5] M. Goldwasser. An implementation for maintaining arrangements of polygons. In *Proc. 11th Ann. Symp. on Computational Geometry*, pages C32–C33, 1995.

[6] J. Illingworth and J. Kittler. A survey of the Hough transform. *CVGIP*, 44:87–116, 1988.

[7] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE T-PAMI*, 16(2):150–162, 1994.

[8] S. Laveau and O. Faugeras. Oriented projective geometry for computer vision. In *Proc. 4th ECCV*, pages 147–156. Springer-Verlag, 1996.

[9] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. SIGGRAPH '96*, pages 31–42, 1996.

[10] J. L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[11] P. Srinivasan, P. Liang, and S. Hackwood. Computational geometric methods in volumetric intersection from 3D reconstruction. *Pattern Recognition*, 23(8):843–857, 1990.

[12] J. Stolfi. *Oriented Projective Geometry*. Academic Press, 1991.

[13] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: IU*, 58(1):23–32, 1993.

[14] R. Vaillant and O. D. Faugeras. Using extremal boundaries for 3-d object modeling. *IEEE T-PAMI*, 14(2):157–173, 1992.

[15] M. Wright, A. Fitzgibbon, P. J. Giblin, and R. B. Fisher. Convex hulls, occluding contours, aspect graphs and the Hough transform. *Image Vision Computing*, 14:627–634, 1996.

[16] C. Zhao and R. Mohr. Global three-dimensional surface reconstruction from occluding contours. *CVIU*, 64(1):62–96, 1996.

[17] J. Y. Zheng. Acquiring 3-D models from sequences of contours. *IEEE T-PAMI*, 16(2):163–178, 1994.