Occlusions are Fleeting - Texture is Forever: Moving Past Brightness Constancy

Christopher Ham[1], Surya Singh[1], and Simon Lucey[2]

[1]Robotics Design Lab
The University of Queensland, Australia
{c.ham,spns}@uq.edu.au

[2]Robotics Institute
Carnegie Mellon University, USA
slucey@cs.cmu.edu

Abstract

Recent work in dense monocular 3D reconstruction relies on dense pixel correspondences and assumes brightness constancy and saliency, and thus are fundamentally unable to reconstruct low-textured or non-lambertian objects such as glass or metal. Occlusion boundaries differ from texture in that each unique view generates a unique set of occlusions. By detecting and solving for the depths of occlusion boundaries, we show how dense reconstructions of challenging objects can be integrated with existing monocular reconstruction algorithms by compensating with an increasing number of unique views. In a sequence containing the Stanford Bunny (8.5 cm) the points of our reconstructed edge cloud have an RMS error of 2.3 mm.

1. Introduction

Due to underlying assumptions about brightness constancy or feature saliency, intensity and feature based 3D reconstruction methods are fundamentally unable to densely reconstruct objects with low texture or with materials such as glass and metal. But, is it still possible without dense pixel correspondences?

Only five points are required to completely recover camera extrinsics [8]. Recently, however, the community has become intoxicated with obtaining dense 3D reconstructions using monocular cameras which rely on dense pixel correspondences [6, 13].

Occlusion boundaries, unlike texture, are view dependent. They appear as edges in an image when the surface of an object is tangential with the camera view, and so provide increasingly dense and detailed cues about the limits and shape of an object with an increasing number of unique views. These cues have been well understood for some time [18, 1], although these methods are limited to simple objects and highly controlled environments.

We hypothesize that with a reliable method for detecting occlusion boundaries, any monocular reconstruction

Figure 1: Given camera poses from handheld video sequences, our method can reconstruct edge clouds of traditionally challenging objects. By observing how visual edges evolve as the camera moves, edges that are persistent (*red*) and non-persistent (*blue*) can be detected and reconstructed separately. Non-persistent edges are caused by occlusion boundaries, and by making correspondences to nearby views they provide strong cues about the shape of the surface. With no assumptions on brightness constancy, we can recover the structure of glass or metalic objects.

method can be extended to densely reconstruct an object in the absence of dense texture correspondences by compensating with an increased number of unique views.

The occurrence of visual edges is not exclusive to occlusion boundaries, they also occur due to high texture gradients (Fig. 2). However, the view dependent nature of occlusions allow them to be separated by observing how the visual edges of a scene evolve across views.

Previous work using edges [18, 1, 11, 12, 3] has relied on having full control on the environment while observing an object. Under real world conditions with background clutter, complex geometry, and highly textured objects it becomes challenging to make reliable corresponding pairs between visual edges across views using traditional epipolar geometry.

As shown by [18], a minimum of three views are required to determine the local depth, and curvature of the surface. Following that these findings are based on differential geometry and that views are the discretization of surface observations, we note that increasing the view density (higher spatio-angular resolution) will immediately increase the precision of the surface estimation. This is in contrast to traditional multi-view geometry where a wider baseline and denser spacial observations have a more significant impact on precision than increasing the number of views.

In this paper we show that by allowing the surface around an occluding edge to be locally approximated by a cylinder, edges from multiple nearby views that best fit this model can be used to robustly estimate its depth. This differs from existing work by performing the correspondence and fitting tasks simultaneously. As a result, this method can be efficiently applied to large, challenging sequences with a high spatio-angular resolution, and is capable of densely reconstructing the location *and* curvature of points on the surface of low-textured or non-lambertain objects.

2. Prior Art

The theory for extracting local surface information from occlusion boundaries has been well understood for some time. Vaillant and Faugeras [18] proved fundamentally that at least three nearby views were required to recover up to a second order measurement (depth, tangent, curvature) of the surface at an occlusion boundary caused by a smooth surface (extremal boundary). Under the assumption of approximately linear camera motions, the authors proposed a circle fitting method for estimating the local surface in the epipolar plane defined by an observation and the motions. They further generalized the curvature estimation using a Fourier basis.

Boyer and Berger [1] also used triplets of views, but relaxed assumptions on camera motions by not requiring a plane in which to perform fitting. Instead they use a

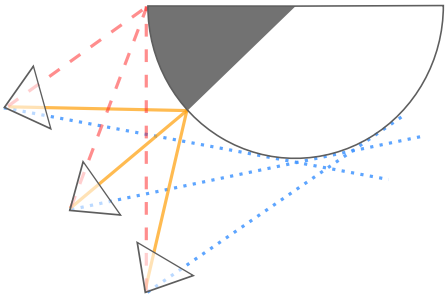


Figure 2: Textural edges (*orange*) and geometric edges (*red dashed*) on a semi-circle are persistent across many views. Purely occluding edges (*blue dotted*) are unique to the viewing angle, but nearby observations (non-persistent correspondences) provide rich information about the local curvature and limits of the surface.

3D parabolic patch parameterized according to the original viewing angle.

Both [18] and [1] show compelling results for highly controlled datasets where the background is mostly one color, the objects are simple with little texture, and baselines between views are large. As the complexity of objects (and thus the density of edges) increases, epipolar edge correspondences become more ambiguous. Even though these fitting methods become sensitive to noise as the baseline is reduced, this can be potentially countered by considering correspondences from views beyond the immediate neighbors.

In this paper, we consider correspondences and fitting at the same time by carefully modelling the problem in a way that can be solved using Locally Optimized RANSAC (LO-RANSAC). This allows us to efficiently consider many views and thus take advantage of the density that small baseline views has to offer while rejecting, for example, background edges that make no sense.

Eventually, the community moved away from using raw edges and towards silhouettes. Silhouettes guarantee closed curves and closed regions which help simplify many aspects of surface reconstruction. However, as a result they are fundamentally unable to consider useful cues such as self occlusions.

In the shape from silhouette literature Liang and Wong [11] explored the recovery of an object's surface from the dual tangent space. Their contribution was in parameterizing and estimating the dual surface according to epipolar correspondences between sequential views.

Mikhnevich and Laurendeau [12] explicitly show the power in using non-texture based methods for reconstructing typically challenging objects such as light-bulbs and wine glasses. They present a segmentation method for defining the image region of an object of almost any sur-

face type in a multi-view stereo rig and use this to perform visual hull reconstructions.

The use of occlusion boundaries to explictly assist rather than reduce hindrance to 3D reconstructions using a large number of views has been explored in [3] and [10] - both methods rely on having highly structured camera motions. Crispell *et al*. [3] use multi-flash photography to isolate occlusion boundaries in a multi-view stereo rig. They show that these internal silhouettes provide strong cues about the limits and curvature of an object's surface. Kim *et al*. [10] rely on occlusion boundaries to recover high fidelity depth maps from light-field arrays. The appearance and disappearance of points helps to anchor discontinuities in the reconstructed depths in a fine-to-coarse approach.

In addition Shan *et al*. [15] in the multi-view stereo community realize that occlusion boundaries are likely to occur at edges and allow for discontinuities in the reconstructed depth maps by lowering the spacial smoothing parameter at these locations. Even so, the primary goal is to accommodate for occlusions rather than to use them explictly for reconstructions.

To detect occlusion boundaries in video sequences Stein and Herbert [16] use a bottom-up patch based approach that does not rely on known camera poses. Occlusions are detected where there are disparities in the motion of neighboring patches that are aligned with and straddle visual edges. This method, however, makes the assumption that foreground and background patches on either side of an occlusion boundary are well textured.

Our work harnesses the view-dependent nature of occlusion boundaries to offer an alternate approach to reconstructing scenes from video. The density of the reconstruction is dependent on the spatio-angular resolution of the sequence.

Dense video reconstruction was pioneered by the seminal work by Newcombe *et al*. [13]. DTAM differs from previous SLAM algorithms by tracking camera motion on raw pixel intensities - although it still requires feature tracking for initialization. It does this by fitting current intensity observations with a dense up-to-date model of the scene. The model is updated by incoming keyframes, whose depths are densely estimated by aligning multiple neighboring images. The algorithm increases its robustness to surfaces that violate brightness constancy by down-weighting those pixels and allowing the optimization to fill in the gaps where necessary. As a result, these surfaces cannot be properly reconstructed.

Building on DTAM, Engel *et al*. [6] introduced LSD-SLAM, which focuses on robust large-scale camera odometry rather than dense reconstructions. Like DTAM, it tracks on pixel intensities but pivots on the realization that not many *good* correspondences are required for accurate pose estimations. It lends its robustness to rigorously culling any pixels that are low-texture, occluded, or do not strictly adhere to assumptions of brightness constancy.

Our algorithm differs in that it attempts to approximate surfaces by observing the evolution of edges over time. In doing so we retain the ability to capture and reconstruct the self-occlusions (or internal silhouettes) of an object.

Fabbri and Kimia [7] presented a method that reconstructs a "3D Curve Sketch" from edge observations. In the final reconstruction only view-stationary (persistent) curves are modelled, with view-nonstationary (non-persistent) curves such as from occlusion boundaries being rejected as they do not adhere to assumptions made by the method.

Building on the work in [7], Usumezbas et al. [17] proposed an algorithm which generates a topologically accurate reconstruction of persistent edge observations by identifying redundancies in the 3D Curve Sketch reconstruction and estimating a set of piecewise connected 3D edges. Similarly to [7] this method does not consider non-persistent edges.

3. Labelling Edges

Persistent edges are edge observations from multiple views that belong to the same curve in 3D space. Non-persistent edges are caused by occlussion boundaries on smooth surfaces (see Fig. 2).

The difference in nature of persistent and non-persistent edges requires that they are first separated before a final reconstruction. Given a set of images with known, calibrated cameras the first stage of processing labels detected edges with estimated depths, radii, and correspondences by geometric reasoning. At its core, the method pivots on the realisation that the apparent radius of curvature of the surface approximated by corresponding edges in other views is zero for persistent edges.

First, visual edges are detected and simplified as polyline segments. A large dataset is generated from the back projected segments; in the same way that points in image space become rays in 3D space, image line segments become triangles in 3D space.

Then, for each triangle, we wish to estimate the depth and curvature of the local surface. This is achieved by mapping nearby triangles to lines in dual space. Triangles that approximate a local cylindrical patch in 3D space map to lines that intersect at a common point in the dual space. The location of the point describes the cylinder's depth and radius (curvature). The segment is classified as persistent if the curvature is above some threshold.

3.1. Edge Contour Extraction

Edges and their normals are extract from images in the form of polyline contours. Vectorizing raw pixel edges "smooths out" the contours within some digitization error.

This provides two immediate advantages: *(i)* less data is required to capture information about the shape and orientation of the edges; *(ii)* the edges' normal estimations are much more stable.

The choice of edge detector is not critical for this work. However, when detecting edges in thousands of images, a faster detector is preferred. The experiments in this paper use Dollár and Zitnick's fast structured forest edge detector [4] as it is competitive with much slower state of the art algorithms.

The edge contours are initially vectorized by finding the ordered connectivity of the edge pixels. Each contour is extracted as an ordered set of points with a start and finish. At points where an edge diverges into multiple directions, the current contour is finished, and new ones are started.

This initial vectorization captures all the discretization errors from the underlying edge detector - the differentiation of which is noisy. The Ramer-Douglas-Peucker algorithm [14, 5] is used to simplify the contours within digitization error.

The final dataset of edges is a set of line segments, each with a normal (perpendicular to the segment). However, in much the same way that a point in image space can be represented by a ray in 3D space (with an origin at the camera center, $\mathbf{t}$, and direction defined by the homogeneous image point, $\mathbf{q}$ and global camera orientation, $\mathbf{R}$ - Eq. (1)), a segment can be represented by a triangle with infinite depth (one vertex at the camera center, and 2 rays, $(\mathbf{q}_1, \mathbf{q}_2)$, defining its connected edges - Eq. (2)). In the following sections we will refer to these interchangeably as edges and edge triangles.

$$\mathbf{p}_{ray}(u) = \mathbf{t} + u(\mathbf{R}\mathbf{q}_l) \tag{1}$$

$$\mathbf{p}_{tri}(u, v) = \mathbf{t} + u(\mathbf{R}\mathbf{q}_1) + v(\mathbf{R}\mathbf{q}_2) \tag{2}$$

where $u >= 0$ and $v >= 0$.

3.2. Correspondence and Fitting

Without any measure of saliency (such as intensity, color, or descriptors), our method relies on geometric reasoning from a large number of views to find the mapping of corresponding edges for each edge. Underpinning the algorithm is the assumption that the surface can be locally approximated by a cylinder whose axis is normal to a "slicing plane", and finding the set of edges from other views that best fit this model.

Definitions Each edge triangle has an origin, $\mathbf{t}_i$, and a unit length *mid-ray*, $\hat{\mathbf{m}}_i$, which is the 3D ray defined by the point at the center of the visual edge. The *slicing plane* is a 2D space with the origin at the selected edge's camera center $\mathbf{t}_s$, X-axis defined by the mid-ray $\hat{\mathbf{m}}_s$, and Y-axis defined by edge's normal vector, $\hat{\mathbf{n}}_s$. The *osculating circle* is a circle in the slicing plane that approximates the radius of curvature

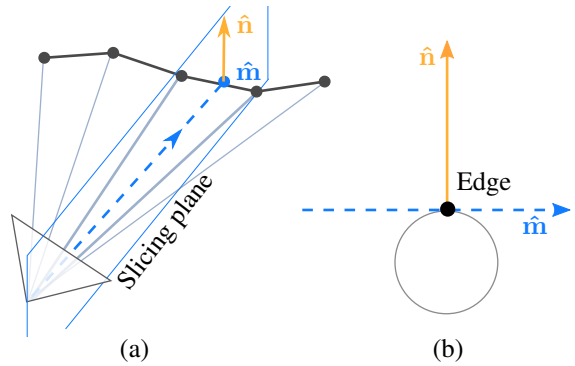


Figure 3: (a) A mid-ray (*blue dashed*) is a ray from the camera center to an edge's middle. An edge's slicing plane (b) is coincident with the camera center and normal to the mid-ray and edge's normal (*orange*). The tangent circle (*grey*) locally approximates the surface's curvature at the point at which the mid-ray is tangent.

of the surface in the direction of the mid-ray at the point to which it is tangent. Figure 3 illustrates these definitions.

Mid-rays of nearby edge triangles are mapped to the slicing plane by and orthogonal projection $\mathbf{K}_s$. The projected 2D rays are defined by their origin $\mathbf{c}_i$, and direction $\mathbf{w}_i$

$$\mathbf{K}_s = [\hat{\mathbf{m}}_s \;\; \hat{\mathbf{n}}_s]^\intercal \tag{3}$$

$$\mathbf{c}_i = \mathbf{K}_s(\mathbf{t}_i - \mathbf{t}_s) \tag{4}$$

$$\mathbf{w}_i = \mathbf{K}_s\hat{\mathbf{m}}_i \tag{5}$$

Based on the cylindrical assumption, likely correspondences are established by finding the set of projected rays that best describe the envelope of a osculating circle. To do this robustly, the rays are mapped to a dual space in which lines that describe such a circle become lines that intersect at a common point (Eq. (6) and Fig. 4), then this point is fitted using locally optimized RANSAC [2].

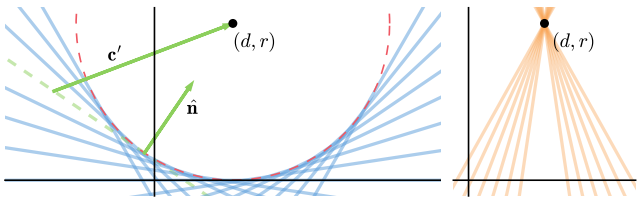
The dual space has been designed to provide two important mappings: lines in the slicing plane to lines in the dual space; and points in the dual space to osculating circles in the slicing plane. See Section 4.5 for a derivation of the mappings.

Given a ray defined by an origin $\mathbf{c}_i$, and normalized direction $\mathbf{w}_i$, in the slicing plane, we map it to a homogeneous line in the dual space by

$$\mathbf{l}'_i = [-w_i^y,\; w_i^x - 1,\; \mathbf{c}_i \cdot [-w_i^y, w_i^x]^\intercal]^\intercal \,. \tag{6}$$

Additionally, a point in the dual space with the position (d, r) is mapped back to an osculating circle in the slicing plane with the equation $(x - d)^2 + (y - r)^2 = r^2$

In reality pose and discretization errors cause the radius of persistent edges to be non-zero, and so persistent edges



(a) Osculating Circle in Slicing Plane (b) Dual Rays

Figure 4: Rays lying tangent to a circle that is tangent to the X-axis (a) are mapped to lines that intersect at a common point in the dual space (b). The X component of the point is the X position of the center of the circle (depth of the current edge), and the Y component is the radius of the circle.

are classified as edges whose estimated radius of curvature is below some threshold scaled by its depth to remain invariant to scale. The estimated depth is also recorded to assist in the 3D reconstruction of edges (Section 4).

Nearby Triangles With at least 0.5 million triangles in many of the sequences we're interested in, fitting a model that may contain only 0.1% of those becomes infeasible. Thus before fitting, we filter for potential correspondences using two tests. A candidate edge is kept if:

1. Its edge triangle intersects with the selected edge triangle, and at a depth in front of the camera.
2. The angle between its mid-ray and the slicing plane is below some threshold since mid-rays that are nearer to the plane more closely trace out the surface as sliced by the plane.

This pre-culling can be further accelerated by caching the occupency of edge triangles in a voxel grid.

4. Reconstructing Edges

With each image edge now given an estimated curvature, depth, and corresponding edges, we wish to reject those with low confidence and classify the remaining as persistent or not. Since persistent edges correspond to the same 3D edge they require an additional stage of processing to "condense" them into one 3D polyline.

4.1. Removing Low Confidence Edges

Each edge has its own set of corresponding edges. The confidence of an edge is approximated by the number of edges who reciprocate the correspondence. And so, low confidence edges are culled by setting a minimum threshold on the number of reciprocated correspondences.

4.2. Reconstructing Non-Persistent Edges

With depths and high confidence, the last remaining ambiguity for a non-persistent edge is its orientation in the edge triangle's plane. This is because the estimated depth has been calculated for the edge's midpoint and so the edge's end points are still unconstrained.

The orientation is determined by first obtaining the intersecting lines of the selected edge plane with the reciprocated corresponding edge planes. The median orientation of these lines in the selected edge plane is then used to calculate the final depths of the edge's end points.

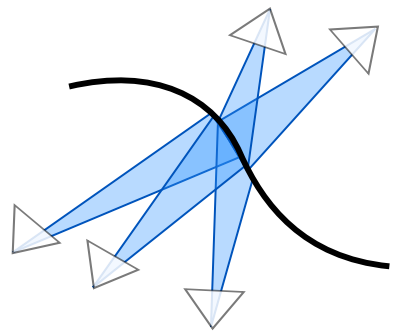


Figure 5: Marching along persistent 3D edges. Edge triangles that share portions of the same 3D edge intersect along the edge. The edge is traced out by minimizing the distance to triangle planes at each step.

4.3. Marching Persistent Edge Intersections

The second stage of processing takes a set of persistent edge observations, represented in global space as triangles (see Section 3), as input and reconstructs them as polylines in 3D space by marching along their intersections. We borrow concepts from the marching method for estimating the polyline intersection of two general 3D surfaces [9] and extend it to estimate a polyline that best fits multiple surfaces (triangles).

Observed 2D edge segments represent an approximately straight portion of the true 3D edge projected into image space. Due to an ambiguity in depth, these observations can be expressed in 3D space as triangles (Eq. (2)). Our generalized marching algorithm aims to find a 3D polyline whose vertices each minimize the distance to likely corresponding triangles (see Fig. 5).

As an overview, a persistent edge is reconstructed by marching along edge triangle intersections by the following process, with further details in sequential paragraphs:

1. Given an initial point, $\mathbf{p}_0$, and tangent vector, $\mathbf{u}$, find likely corresponding triangles.
2. Calculate $\nabla\mathbf{p}$ that minimizes the distance to selected triangles, and is perpendicular to $\mathbf{u}$. Add $\mathbf{p} = \nabla\mathbf{p} + \mathbf{p}_0$ to the current polyline.
3. Label inlying triangles as belonging to this polyline.
4. Estimate new $\mathbf{p}_0$ by taking a step in the direction of $\mathbf{u}$

by a defined step size.

5. Estimate new $\mathbf{u}$ from the last n points in the polyline.

Initialization An initial point, $\mathbf{p}_0$ and direction, $\mathbf{u}$ are defined by selecting a triangle not yet belonging to a polyline. From the triangle's camera center, estimated depth, mid-ray and normal ($\mathbf{c}$, d, $\mathbf{q}$, $\mathbf{n}$) we calculate

$$\mathbf{p}_0 = \mathbf{c} + d\mathbf{q}; \;\; \mathbf{u} = \frac{\mathbf{q} \times \mathbf{n}}{\|\mathbf{q} \times \mathbf{n}\|} \tag{7}$$

Corresponding Triangles Given a point $\mathbf{p}_0$, *likely* corresponding triangles are defined as those where the nearest point on the triangle to $\mathbf{p}_0$ has positive barycentric coordinates, and its distance to $\mathbf{p}_0$ is below some threshold δ_{tri}.

Point Fitting A point that best fits a set of triangles is estimated by minimizing the distance to the triangles' planes, while constraining the point to lie on a plane that is coincident with $\mathbf{p}_0$ and normal to $\mathbf{u}$. The selected triangles are likely but not guaranteed to be observations of this point, and so the linear system is solved using locally optimized RANSAC [2].

Termination The algorithm needs to know when it has reached the end of an edge. This can happen for two reasons: (i) the polyline loops back around to the beginning, or (ii) the polyline "marches off" the end of the edge. A loop is tested for by simply checking if the end is marching towards the beginning and is near. The end of an open segment is tested for by checking the number of inliers in the point fitting step.

Note that an initial point is unlikely to be at either end of the true edge. If the end of the edge is detected, the algorithm begins marching from the initial point in the opposite direction until it reaches the other end.

The algorithm continues to reconstruct 3D edges and label edge triangles until every triangle is labelled.

4.4. Fast Triangle Lookups

Many of the sequences we're interested in have upwards of 5000 views and over 100 edge segments per view. In order to make the fitting problem tractible we wish to efficiently cull edges that cannot possibly be correspondences. We do this by generating an occupancy voxel grid of the edge triangles. Each triangle maps to the voxels with which it intersects, and each voxel maps to triangles that intersect with it.

With this data, it is trivial to obtain a shortlist of triangles near a given triangle by looking up its voxels, then finding the union of the sets of triangles belonging to those voxels. This shortlist can then be handed to the correspondence and fitting task (Section 3.2).

4.5. Derivation of Dual Space

Section 3.2 describes how to obtain a set of rays that potentially describe the local surface at an observed edge. In this work, we assume that the surface can be locally described by an osculating cylinder whose axis is aligned with the observed edge. By orthographically projecting the candidate rays into the slicing plane defined by the edge, we can limit the problem to searching for rays that describe the envelope of an osculating circle to the X-axis.

The problem can be further simplified by use of a dual space in which rays that are tangent to a common osculating circle in the original space map to lines that intersect at a common point in the dual space. The remainder of this section describes how this space is derived.

Each projected candidate ray can be described by its origin $\mathbf{c}$ (camera center) and direction $\hat{\mathbf{w}}$. We assume that the neighboring observations come from nearby views, and thus the X component of the projected rays' directions, w_x, are positive.

As shown in Figure 4, let's define the normal of a ray to be the left hand orthogonal of its direction, $\hat{\mathbf{n}} = [-w_y, w_x]^\intercal$, and the osculating circle's center with respect to the ray's origin, $\mathbf{c}' = [d - c_x, r - c_y]^\intercal$. Under the hypothesis that this ray is tangent to the osculating circle then

$$\mathbf{c}' \cdot \hat{\mathbf{n}} = r \tag{8}$$

$$-(d - c_x)w_y + (r - c_y)w_x - r = 0 \tag{9}$$

$$-w_y d + (w_x - 1)r = \mathbf{c} \cdot [-w_y, w_x]^\intercal. \tag{10}$$

Which describes a line with respect to d and r. The homogeneous form being

$$\mathbf{l}' = [-w_y,\ w_x - 1,\ \mathbf{c} \cdot [-w_y, w_x]^\intercal]^\intercal. \tag{11}$$

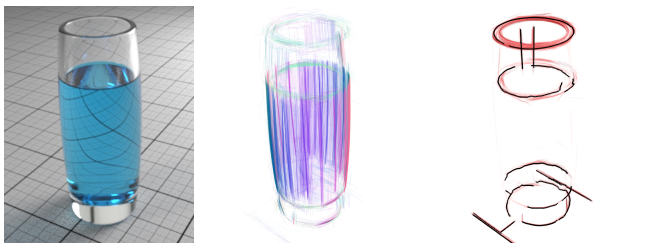
Since the dual space is parameterized by d and r, and point in the dual space maps directly to an osculating circle described by $(x - d)^2 + (y - r)^2 = r^2$.

5. Experiments

To assess our reconstruction, we provide a mixture of quantitative and qualitative results. The first sequence is a photorealistic render of a glass containing liquid and offers a ground truth against which we can compare our reconstruction. The remaining requences are collected using a smartphone camera and tracked using a checkerboard pattern. One of the chosen real world models is a 3D print of the Stanford Bunny which, while not perfect, provides a reference for the quality of the real world reconstructions.

Ideally, one would prefer to track the scene without additional structure, but in these experiments we wish to isolate and assess the efficacy of our reconstruction algorithm.

Parameters The normalized radius threshold for classifying the persistence of an edge is set to 0.03. The curvature of an edge is scaled by its depth before comparing with this value in order to remain invariant to scale. The minimum number of reciprocated neighbors (the confidence) for both



(a) Rendered Glass (b) Non-persistent (c) Persistent + Poly

Figure 6: Synthetic sequence rendered using Blender. RMS eror of non-persistent edge vertices is 2.36mm given a glass height of 15cm and distance of camera to object between 35-55cm. (b) non-persistent edges are colored according to their estimated surface normal directions. (c) red lines indicate persistent edges, black lines are reconstructed poly lines from the persistent edges.

types of edge is set to 180. The threshold distance δ_{tri} for potential corresponding triangles during edge marching is set to 0.025 scaled according to each edge triangle's estimated depth.

5.1. Synthetic Glass

Blender was used to photorealistically render a synthetic sequence of a glass containing liquid[1]. Additionally, the camera motions for the sequence were taken from real world high framerate tracking data and contain 5300 frames. These poses and the rendered camera frames were used as input into our algorithm. The blue liquid provides a high contrast edge that can be tracked as the camera moves through space. Note also that edges in the background are refracting through the glass and strongly interacting with the foreground.

Our algorithm is capable of reconstructing the liquid line, the rim of the glass, and an edge cloud of the surface of the glass (Fig. 6). Thanks to the RANSAC fitting method, edges from the background are correctly rejected and do not cause any significant errors in the result. At the scale of the glass being a height of 15cm and camera to object distance ranging between 35 and 55cm, the RMS error of the non-persistent reconstructions to the ground truth model is 2.36mm. The processing time for this sequence on a machine with a 2.5GHz Intel Core i5 and 8Gb of memory is under 5 minutes.

5.2. Real World

Real world sequences were collected using the high framerate mode on an iPhone 6. A checkerboard was placed

[1] http://www.blendswap.com/blends/view/70630

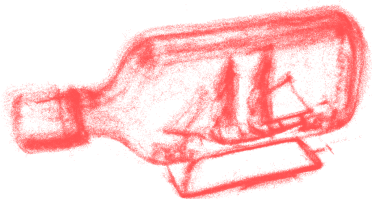


Figure 7: LSD-SLAM's [6] point cloud reconstruction of the ship in a bottle. Due to brightness constancy assumptions it is unable to fully reconstruct the glass bottle.

next to the object of interest and used to track the camera's pose. The video frames and pose estimations are passed as input to our method. We reconstructed four objects (Fig. 8), each chosen to address different types of reconstruction. Additionally, the Stanford Bunny enables a transferable quantitative analysis of the accuracy of its reconstruction.

The Stanford Bunny and orchid sequences sit at opposite ends of the spectrum in terms of edge types. The bunny mostly only generates non-persistent edges with little texture or few well defined edges in its geometry. The orchid mostly on generates persistent edges due to the flat leaves and flowers, and box-shaped pot.

For the Stanford Bunny the RMS error of the reconstructed non-persistent edges is 2.27mm, with camera to object distances ranging from 30cm to 100cm.

The last two sequences reconstruct objects made partly or entirely from glass. This enables us to demonstrate the power of using edge correspondences to recover the structure of objects. The wine glass's stem is not entirely reconstructed since the camera does not sweep around at a low enough height to capture those edges. For the ship in the bottle, our method simultaneously recovers the structure of a lambertian object and the glass bottle around it. This highlights the fact that textural and occlusion observations can be figuratively and often literally orthogonal.

We observe that the separation of persistent vs. non-persistent edges is not perfect. These classification errors mainly come from variation in the estimated curvatures.

As a point of qualitative comparison, LSD-SLAM's [6] point cloud reconstruction of the ship in a bottle is also included (Fig. 7). While it does well to reconstruct the surfaces in the scene that adhere to brightness constancy assumptions, it is unable to densely recover the structure of the glass bottle, especially the flatter front surface. Comparing to our persistent reconstructions of the bottle, the use of edges instead of points allows for crisper reconstructions.

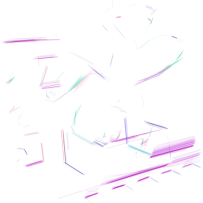
(a) Stanford Bunny (6454 frames)

(b) Non-persistent (55343)

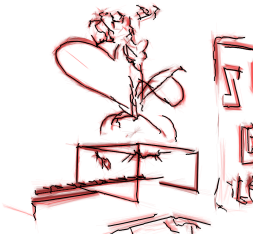
(c) Non-persistent (view 2)

(d) Persistent (8012)

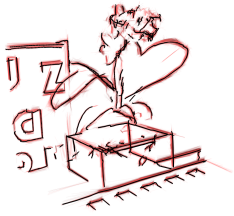
(e) Orchid (5284 frames)



(f) Non-persistent (2877)



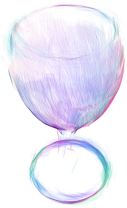
(g) Persistent (view 2)



(h) Persistent (123953) + 3D Poly

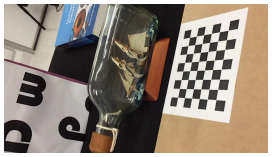
(i) Wine Glass (4824 frames)

(j) Non-persistent (60104)

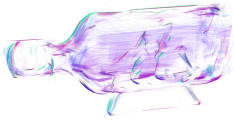


(k) Non-persistent (view 2)

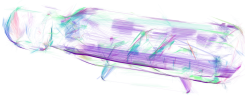
(l) Persistent (8294)



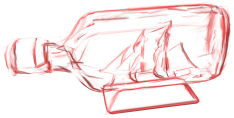
(m) Ship in a Bottle (5137 frames)



(n) Non-persistent (42028)



(o) Non-persistent (view 2)



(p) Persistent (85172)

Figure 8: Persistent vs. non-persistent edge cloud reconstructions for real world objects, annotated with the number of frames and edges. The Stanford Bunny (a) – (d) and the orchid (e) – (h) find themselves at opposite ends of the spectrum in terms of non-persistent occlusions vs. persistent edges. Since our method of edge correspondence does not rely on brightness constancy assumptions, typically challenging materials like glass can be reconstructed (i) – (p). Poly edges are shown for sequences with dense persistent edge clouds.

6. Conclusion and Future Work

In this paper we presented a method that can reconstruct the depths of occlusion boundaries by observing how visual edges evolve with camera motion. We showed that this method was capable of recovering the structure of typically challenging materials, such as glass. By remaining agnostic to the method of pose estimation our algorithm is able to compliment existing 3D reconstruction algorithms - pushing past requirements for reconstructed objects to adhere to assumptions of brightness constancy.

In future work we wish to integrate edge detection with image patch analysis so that visual cues may be mixed with geometric cues for better classifying the persistence of edges. Better edge classification opens the possiblity to camera pose refinements before the final reconstruction is performed.

References

[1] E. Boyer and M.-O. Berger. 3d surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22(3):219–233, 1997.

[2] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[3] D. Crispell, D. Lanman, P. G. Sibley, Y. Zhao, and G. Taubin. Beyond silhouettes: Surface reconstruction using multi-flash photography. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 405–412. IEEE, 2006.

[4] P. Dollár and C. Zitnick. Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848, 2013.

[5] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[6] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014.

[7] R. Fabbri and B. Kimia. 3d curve sketch: Flexible curve-based stereo reconstruction and calibration. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1538–1545. IEEE, 2010.

[8] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[9] E. Hartmann. Geometry and algorithms for computer aided design. *Darmstadt University of Technology*, 2003.

[10] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. H. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Trans. Graph.*, 32(4):73–1, 2013.

[11] C. Liang and K.-Y. K. Wong. Robust recovery of shapes with unknown topology from the dual space. *IEEE transactions on pattern analysis and machine intelligence*, 29(12):2205–2216, 2007.

[12] M. Mikhnevich and D. Laurendeau. Shape from silhouette in space, time and light domains. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 3, pages 368–377. IEEE, 2014.

[13] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[14] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972.

[15] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. M. Seitz. Occluding contours for multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 4002–4009. IEEE, 2014.

[16] A. N. Stein and M. Hebert. Local detection of occlusion boundaries in video. *Image and Vision Computing*, 27(5):514–522, 2009.

[17] A. Usumezbas, R. Fabbri, and B. B. Kimia. From multiview image curves to 3d drawings. In *European Conference on Computer Vision*, pages 70–87. Springer, 2016.

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