Range Sensor and Silhouette Fusion for High-Quality 3D Scanning

Karthik S. Narayan, James Sha, Arjun Singh, and Pieter Abbeel

***Abstract*— We consider the problem of building high-quality 3D object models from commodity RGB and depth sensors. Applications of such a database include instance and object recognition, robot grasping, virtual reality, graphics, and online shopping. Unfortunately, modern reconstruction approaches have difficulties in reconstructing objects with major transparencies (e.g., KinectFusion [22]) and/or concavities (e.g., visual hull). This paper presents a method to fuse visual hull information from off-the-shelf RGB cameras and KinectFusion cues from commodity depth sensors to produce models that are substantially better than either approach on its own. Extensive experiments on the recently published BigBIRD dataset [25] demonstrate that our reconstructions recover more accurate shape and detail than competing approaches, particularly on challenging objects with transparencies and/or concavities. Quantitative evaluations indicate that our approach consistently outperforms competing methods and achieves under 2 mm RMS error. Our code is available for download online at `http://rll.berkeley.edu/icra2015modelquality`.**

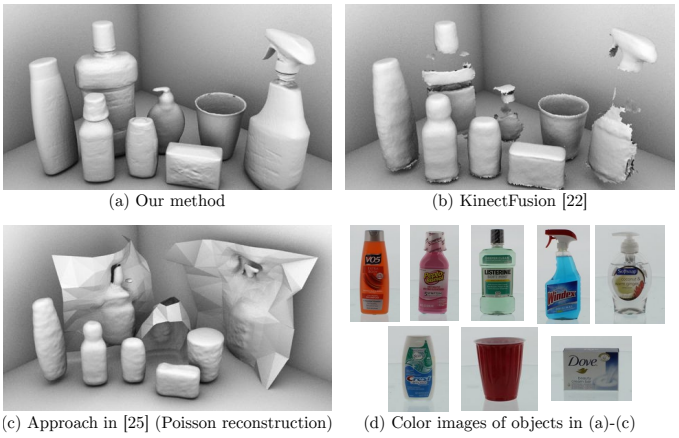
I. Introduction and Related Work

The arrival of the Kinect has sparked large interest in 3D scanning. A highly accurate 3D scanner could have wide impacts in robot vision, particularly instance and object recognition and pose detection, robot grasping, virtual reality, graphics, and online shopping. Variants on the recently proposed KinectFusion algorithm have become particularly popular in reconstruction due to the method's ability to reconstruct objects in realtime while recovering surface details [22], [30]. As a depth sensor receives streaming depth images, KinectFusion (1) calibrates the current depth map using frame-to-model iterative closest point (ICP), (2) updates a truncated signed distance function (TSDF) that stores averaged depth readings, and (3) constructs a mesh using the marching cubes algorithm [19]. Recently published variants of KinectFusion discuss methods to account for the nonlinear distortions introduced by consumer-grade depth sensors [32], [33], [26].

While KinectFusion primarily uses depth cues in reconstruction, popular stereo techniques employ color cues in reconstruction. In particular, multiview stereo approaches currently obtain state-of-the-art results in reconstruction purely from multiple calibrated RGB images [12], [6], [13], [16], [21]. Most such methods produce a 3D point cloud, which is then used to compute a mesh representing the scene. There exist several approaches to compute this point cloud, such as plane-sweeping [8], stereo-matching [28], and patch growing [24].

In reconstructing a single object, one popular approach involves constructing the object's visual hull and iteratively deforming the hull towards multiview-stereo-based point clouds to extract fine details [12], [6], [11]. In particular, the visual hull attempts to reconstruct an object purely from silhouettes captured from multiple views [4]. Noticing that each object silhouette backprojects to a cone, the visual hull intersects these cones to form a description of the real object's shape. Because the visual hull of an object is the envelope of all its possible circumscribed cones, the object must fully lie within its visual hull (see [23] for a proof and Figure 3 for a visualization in 2D).

Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA, 94709



(a) Our method (b) KinectFusion [22]

(c) Approach in [25] (Poisson reconstruction) (d) Color images of objects in (a)-(c)

Fig. 1. Collections of scanned objects reconstructed from (a) the method in this paper, (b) KinectFusion [22], and (c) the previous approach in [25], and (d) colored versions of these objects. Back row, left to right: VO5 volumizing shapoo, Listerine, Softsoap Hand Soap (Coconut and Warm Ginger), red cup, Windex. Front row, left to right: Pepto Bismol, Crest Complete Minty Fresh Toothpaste, Dove Soap. Note that this image does not represent a scanned scene, but a collection of individually scanned objects to conserve space.

Although multi-view stereo, visual hull, and KinectFusion-style approaches perform well in specific settings, they have pitfalls. While multi-view stereo approaches perform very well with highly-textured objects, the poor clouds resulting from lack of texture can lead to sparse and inaccurate clouds [12], [6], [13]. While the visual hull can recover the rough shape of an object even with objects with little texture, it fails to recover concavities in an object, which cannot be represented via calibrated silhouette data [23]. While KinectFusion-style approaches can perform well in the "concavity and low-texture" regime, they fail when working with objects that are translucent, transparent, or highly specular; the visual hull can provide better shape estimates here [22], [26].

Little work has been published on combining each method's strengths. Steinbrucker et. al. [27] discuss an ap-

(a) Ortery Photobench, Perspective

(b) Canon and Carmine Unit

(c) Ortery Photobench, Side

Fig. 2. Our scanning setup consisting of (a), (c) the Ortery Photobench and (b) 5 Canon Rebel t3i + Carmine 1.09 units mounted together. We described a method to jointly calibrate all 10 sensors in [25].

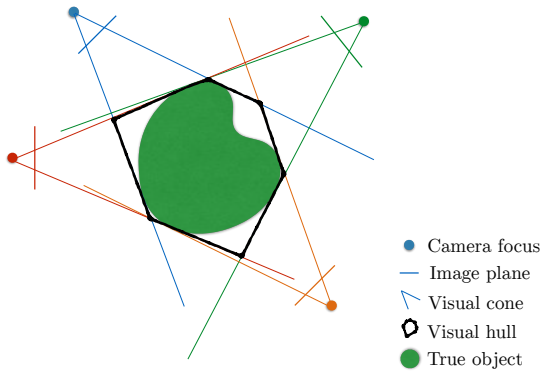


Fig. 3. The red, green, blue, and orange have different views of the solid green object, and thus different silhouettes. Intersecting the cones formed by the silhouettes forms the visual hull, an approximation to the true object's surface. Although the visual hull provides better surface approximations as the number of cameras increases, the visual hull cannot recover the concavity in the true object.

proach to jointly use RGB with depth data to improve camera calibration associated with KinectFusion, resulting in higher quality scene scans. Xu et. al. [31] present an approach that closely refines an object's depth data boundaries purely using silhouette information. As these approaches rely heavily on existing depth data from the Kinect, they do not work well in reconstructing objects that are translucent, transparent, or highly specular. Another work discusses a method to improve Kinect depth reading fidelity by fusing stereo information from both the IR and RGB sensors; although this approach reconstructs bits and pieces of translucent, transparent, and specular objects, the improved data alone is not sufficient to create high quality 3D models due to the large variances in depth estimates [7].

Contributions. This paper presents a reconstruction method that capitalizes on the strengths of the RGB and depth modalities. Specifically, we take in as input a set of calibrated RGB and depth images and produce a high-resolution object mesh. By fusing silhouette and depth information, our method recovers detailed models for challenging objects that are difficult to reconstruct using either modality alone. We evaluate our results on the BigBIRD dataset, discussed in Section II. Shown in Figure 1, our method outperforms competing approaches including the original BigBIRD models [25] and KinectFusion [22].

II. The BigBIRD Dataset

The BigBIRD dataset [25] consists of high quality scanned data for 125 household objects. The data collection system consists of 5 high resolution (12.2 MP) Canon Rebel T3 DSLR cameras and 5 PrimeSense Carmine 1.09 short-range depth sensors. After mounting each Carmine to a DSLR using a platform designed by RGBDToolkit [1], each DSLR is mounted to an arm (the Ortery MultiArm 3D 3000) (see Figures 2b, c).

Data collection for a single object entails placing an object on the turntable (the Ortery Photobench 260) (Figure 2a) and running a single command; the full process is otherwise automated. The turntable rotates in increments of 3 degrees while each of the 5 Canon Rebel T3/Carmine units captures RGB/depth images. This yields a total of 600 point clouds, 600 high-resolution, and 600 low-resolution RGB images from 5 polar $\times 120$ azimuthal views. The 5 cameras are calibrated with each other by placing a chessboard in various orientations, detecting chessboard corners, and running a global bundle adjustment optimizer (described in [25]). Calibrating the top camera with the turntable involves running another bundle adjustment optimizer using a chessboard on the turntable.

III. Unifying Image and Range Sensor Data

At a high level, our reconstruction pipeline (1) computes object segmentations from the high resolution RGB images (Section III-A), (2) computes a visual hull using the segmented, calibrated RGB images (Section III-B, III-C), (3) computes a KinectFusion model using the calibrated depth data (Section III-C), (4) refines the original raw depth maps using the visual hull and KinectFusion models and merges these refined depth maps into a point cloud (Sections III-D, III-E), and (5) forms an object mesh by fusing this merged point cloud with the visual hull (Section III-F).

Our experiments (Section IV) illustrate that our approach capitalizes on the strengths of both the KinectFusion and the visual hull approaches to recover accurate shape models even for objects with concavities and translucent parts.

A. Computing Object Segmentations

Given an object, we first describe a method to extract silhouettes for each of its 600 high-resolution RGB images, which we use to compute the visual hull. We first manually segment *only the first view* of each of the 5 Canon DSLR cameras. Specifically, after running Simple Linear Iterative Clustering (SLIC) [2] to tile an image with superpixels, we manually select the superpixels belonging to the object. The interface, with a sample object, is shown in Figure 4(a); users select computed SLIC superpixels (marked with yellow boundaries) belonging to the object. This process takes under 2 minutes per object (for all 5 segmentations) and is the only

(a) Interactive Segmentation (b) A&H Detergent (c) VO5 Shampoo (d) Softsoap Handsoap

Fig. 4. The interactive segmenter, (a), original images, (b-d top row), and automatically computed segmentations learned from manual segmentations (b-d bottom row). After tiling the image with superpixels using SLIC (separated by yellow boundaries in (a)), users can quickly select superpixels belonging to the object (pink regions). Selecting the last orange superpixel in (a) would complete the segmentation process. Although our learned background models misclassify chessboard regions in (b-d) and produces noisy segmentations for translucent objects (e.g., the Softsoap dispenser pump), our reconstruction scheme still recovers very good object shape (Section IV). Best viewed in color.

manual step in our pipeline. *Note that the user only has to manually segment* 5 *objects, and not all* 600*; we show how to automatically compute the rest of the segmentations below.* In particular, we use 5 segmentations, one per DSLR camera.

Using only 5 *manually segmented images*, we can recover segmentations for all views: per manual segmentation, we construct a dataset $\{(\text{pix}_i, y_i)\}_i$ where $y_i = 1$ denotes an object-pixel and $y_i = 0$ denotes a background pixel, and pix_i denotes the color of the pixel in LAB space. We run k-means ($k = 20$) on the background pixels, i.e. $\{\text{pix}_i | y_i = 0\}$, initialized with k-means++ [3]. For *all* pixels in the manual segmentation, we store the Euclidean distance to the closest mean; next, we use the manual labels to compute a threshold T such that pixels closer than T to a cluster center are classified as "background" while the rest are classified as "foreground." Specifically, we choose T by maximizing the number of correctly predicted pixels; because the number of pixels is on the order of a few million, we can optimally select T by considering all possible distances that the pixels take on.

For the remaining 119 images captured from each of the 5 cameras, we then use the corresponding background model to classify each pixel as object or background. Finally, we mark superpixels as object-superpixels if they contain greater than 30% coverage of object-pixels, and background-superpixels otherwise. Figures (b-d) show examples of marked object superpixels with a magenta tint; we deliberately use a low threshold of 30% to recover object superpixels belonging to white/translucent/transparent objects. Though this introduces false positive object superpixels (e.g. see marked chessboard superpixels in (b-d)), the visual hull carves these regions away, as the same false-positive superpixel rarely appears in many camera views. Armed with these silhouettes and the calibration information per image provided in the BigBIRD dataset, we can then construct the visual hull.

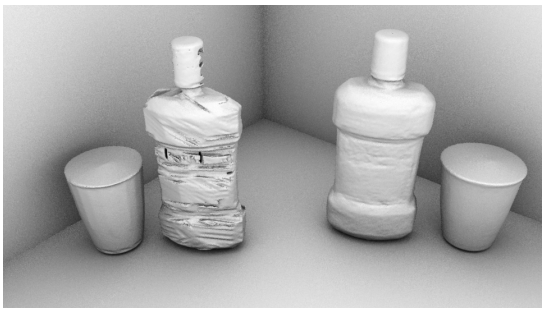


Fig. 5. Comparisons of the hard visual hull (left two objects), using the method in [9] and the soft visual hull, using the method in Section III-B (right two objects).

B. Computing Visual Hull Models

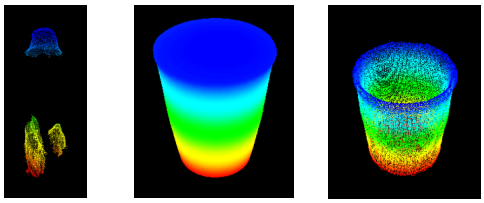
Many variations on computing visual hulls exist [9], [18], [20], [17], [10]. In this paper, we consider the method discussed in [9], which offers good tradeoffs between speed and accuracy: we (1) define a function $F(x) = 1$, if $x \in \mathbb{R}^3$ falls in all silhouettes, and 0 otherwise, (2) compute an initial point x_0 such that $F(x_0) = 1$, i.e. an initial point that lies on the visual hull, and (3) run an implicit surface polygonizer to recover the visual hull mesh, using x_0 as an initialization. We can compute x_0 by considering the 3D back-projections of the bounding boxes per silhouette; repeatedly sampling points within the intersection of these bounding boxes eventually yields a valid x_0 (see [6] for how to compute this intersection). For step (3), we use the publicly available Blumenthal polygonizer with marching tetrahedra [5], [29].

This visual hull approach leads to excessive object carving arising from object pixels being labeled as background pixels during segmentation (see Figure 5, left objects). Such errors typically arise when the object either has bright, white colors or white specularities near segmentation boundaries. To ameliorate this problem, we account for silhouette noise by instead polygonizing the surface $G(x) = 1$ if $x \in \mathbb{R}^3$ falls in $1 - \varepsilon$ of all silhouettes, and 0 otherwise (we set $\varepsilon = 0.1$ in our experiments).

Despite being a crude approximation, the right two objects in Figure 5 demonstrate that this approach works well in practice; the hard visual hull recovers finer surface detail for the cup, but heavily over-carves the Listerine bottle due to segmentation errors. The soft visual hull forgives the segmentation errors in the Listerine bottle that cause excessive carving, but smooths the cup's surface details. We use the soft visual hull, as it generally conforms better to an object's shape. As expected, both methods fail to recover the cup's concavity.

C. Computing Calibrated KinectFusion Models

We can jointly use the depth camera data to recover object concavities; since individual depth maps are inherently noisy, we fuse the depth maps into a single mesh using a variant of the KinectFusion algorithm. The KinectFusion algorithm



(a) Palmolive detergent, KinectFusion mesh vertices

(b) Red cup, visual hull mesh vertices

(c) Red cup, KinectFusion mesh vertices

Fig. 6. Challenges associated with the depth modalities we attempt to blend. Point clouds recovered from KinectFusion meshes can often have large gaps of missing space [22]. These gaps could be misleading due to missing depth readings arising from object transparencies (see (a), featuring the KinectFusion mesh vertices of the translucent Palmolive dishwashing soap). Gaps could also be legitimate, recovering object concavities such as the cup's concavity in (c). Although the visual hull can fill in illegitimate gaps, it can also introduce hallucinated points which cover important concavities, such as the cup's concavity in (b). Section III-D and III-E discuss methods to capitalize on each modality's strengths and reason through the inaccuracies that each modality introduces.

assigns poses to each incoming camera frame via frame-to-model, point-to-plane ICP [22]; because KinectFusion requires slow camera pose movements, and the BigBIRD dataset has 5 cameras in relatively far away locations (see Figure 2c), we use the camera poses that the BigBIRD dataset provides per Carmine rather than frame-to-model ICP. Separate experiments indicated that the provided poses provide more reliable poses over those provided by frame-to-model ICP (particularly in cases where depth maps have gaps due to transparencies). The provided poses also allow us to use depth information from all 5 cameras. We employ a CPU implementation of KinectFusion that represents the TSDF structure using an octree; specifically, we adopt the implementation used in [33].

D. Refining the Original Depth Maps

To ultimately fuse the visual hull and KinectFusion models, we aim to construct a dense cloud whose points lie on the surface of the object and deform the visual hull towards this cloud. In particular, this dense cloud's points will be a subset of the union of the visual hull and KinectFusion mesh vertices. Selecting this subset is nontrivial, since the visual hull introduces hallucinated vertices, e.g., the top of the red cup (Figure 6b). Further, KinectFusion models can contain large empty spaces in regions with few depth readings (Figure 6a). Exacerbating the problem, gaps in the KinectFusion model could either be due to legitimate gaps such as object concavities (Figure 6c) or illegitimate gaps due to object transparencies (Figure 6a). Ultimately, we construct the desired cloud by refining the raw depth maps.

We summarize our algorithm to fuse these depth cues in Algorithm 1. We now explain how Algorithm 1 assigns a refined depth to a single pixel (i, j) given z-buffered depths for the visual hull and KinectFusion maps vh and kf and a raw depth map raw, for a single camera c in angle a. The following 4 cases correspond to the 4 cases described in Algorithm 1.

Algorithm 1 Refining the Original Depth Maps

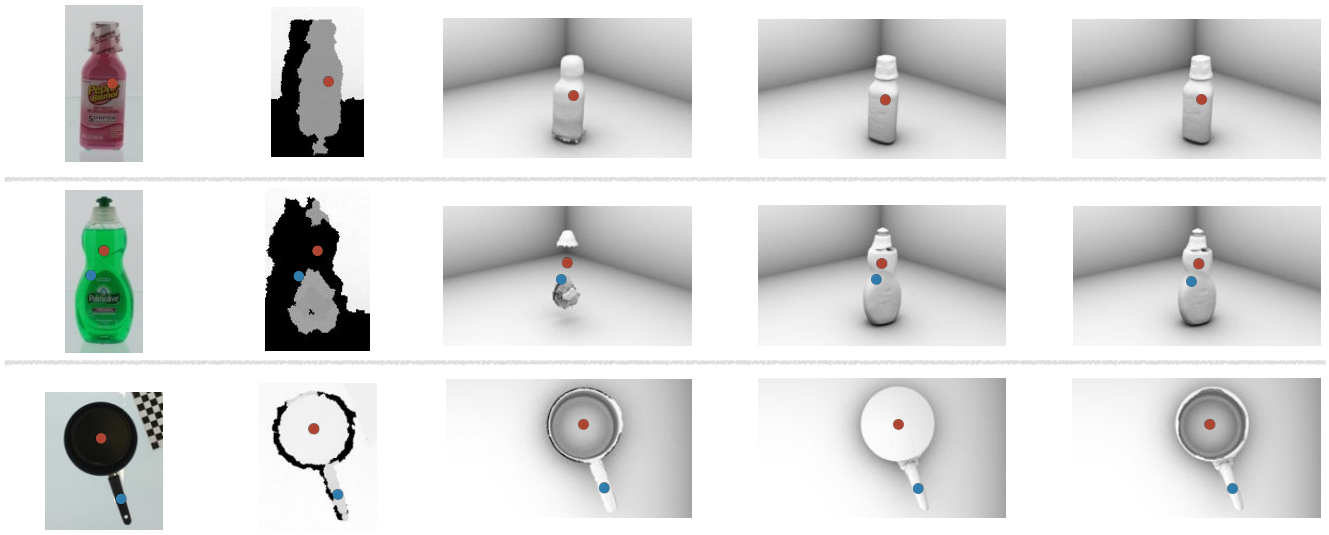
$C \leftarrow$ list of depth cameras whose depth maps to refine
$A \leftarrow$ list of turntable angles per depth camera
$VH \leftarrow$ the soft visual hull mesh
$KF \leftarrow$ the KinectFusion mesh
$cloud \leftarrow$ empty point cloud
for c in C **do**
  for a in A **do**
    $raw \leftarrow$ raw depths for camera c, angle a
    $vh \leftarrow VH$'s z-buffered depths in camera c, angle a
    $kf \leftarrow KF$'s z-buffered depths in camera c, angle a
    $nr \leftarrow$ number of rows in raw
    $nc \leftarrow$ number of columns in raw
    $refined \leftarrow$ empty array with nr rows and nc cols
    for $0 \leq i < rows$ **do**
      for $0 \leq j < cols$ **do**
        if $vh[i,j] = \infty$ **then**
          $refined[i,j] = \infty$ ▷ Case 1
        else if $raw[i,j] = \infty$ **or** $kf[i,j] = \infty$ **then**
          $refined[i,j] = vh[i,j]$ ▷ Case 2
        else if $|vh[i,j] - kf[i,j]| < 1$ mm **then**
          $refined[i,j] = vh[i,j]$ ▷ Case 3
        else ▷ Case 4
          $refined[i,j] = \max\{vh[i,j], kf[i,j]\}$
        end if
      end for
    end for
    $newcloud \leftarrow$ point cloud generated from $refined$
    $cloud$.appendPointsNotAtInfinity($newcloud$)
  end for
end for
return $cloud$

Case 1: vh does not project onto (i, j). We assume that the recovered mesh strictly lies within the visual hull, so if $vh[i, j] = \infty$, we use a refined depth of ∞. The remainder of the cases assume that vh projects onto (i, j).

Case 2: Either raw or kf has a missing depth at (i, j). Because vh projects onto (i, j), Algorithm 1 interprets this case as the Carmine missing readings due to transparencies. For example, this happens for the red point in the second row of Figure 7, the translucent Palmolive dishwashing liquid, where the soft visual hull is reliable while the KinectFusion and raw depth maps have missing depths. In this case, Algorithm 1 prescribes returning the visual hull's depth.

For the blue point in the same row, the raw depth map returns a missing depth while the KinectFusion mesh actually returns a depth reading; note that the KinectFusion depth reading is spurious since the depth reading represents a point that actually lies "inside" the object rather than on the object's surface. Specifically, the KinectFusion mesh returns a false depth reading here, since this point is directly visible to the camera when it isn't supposed to be (namely because transparent surfaces are not reconstructed). According to Algorithm 1, we return the visual hull's depth, as desired.

Case 3: vh and kf return readings closer than 1 mm. In



(a) Object color images (b) Raw depth maps (c) KinectFusion meshes (d) Soft visual hull meshes (e) Our method

Fig. 7. Three concrete cases of the intuition behind merging visual hull and KinectFusion depth information. Section III-D explains how Algorithm 1 operates on (1) the red corresponding points in the first row (Pepto Bismol, a simple, opaque object), (2) the red and blue corresponding points in the second row (Palmolive dishwashing container, an object with major translucencies), and (3) the red and blue corresponding points in the third row (black pot, an object, an object with a major concavity). Our method takes the best pieces of the KinectFusion and soft visual hull meshes; this is particularly evident in the third row, where our method recovers the pot's concavity from the KinectFusion mesh and the refined handle from the soft visual hull.

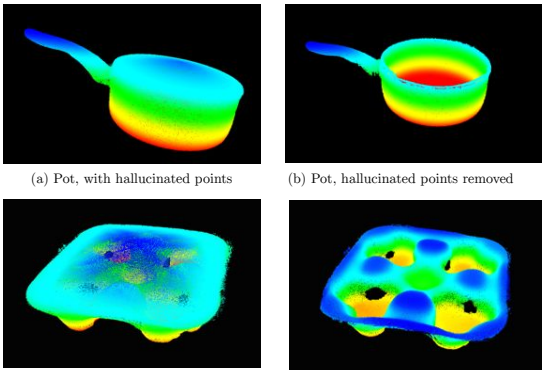
this case, we opt to use *vh*'s depths, since surfaces recovered by the visual hull tend to be more refined. For example, shown in the first row of Figure 7, the KinectFusion and soft visual hulls for the Pepto Bismol container are both fairly reliable. Algorithm 1 returns the visual hull reading for the red point.

Case 4: *vh* and *kf* return readings farther than 1 mm. In this final case, we opt to return the maximum depth between *vh* and *kf*, as this likely implies the presence of a concavity. We address this case in the third row of Figure 7, which presents a black pot with a large concavity. In the red point, both KinectFusion and soft visual hulls return valid depth readings, but these readings differ by more than 1 mm. Taking the maximum depth reading gives us the KinectFusion depth, allowing us to properly recover the concavity.

Algorithm 1 opts to choose *vh* depths for the blue point, which falls under Case 3. This leads to a more refined handle, showing that Algorithm 1 can pick and choose parts of *vh* and *kf* based on reliability.

E. Eliminating Hallucinated Points

Algorithm 1 has a shortcoming, namely that it always assigns a finite depth to points that fall within *vh*. Due to segmentation errors and the soft visual hull threshold, it is possible for points to fall within the soft visual hull, but not fall within the true object. Because the Carmine and raw depth maps indicate such points to be outside the object, Algorithm 1 treats these hallucinated points as part of Case 2, generating hallucinated points. As an example, Algorithm 1 frequently hallucinates points where the visual hull fills in concavities (see Figure 8). Figure 9 presents a visualization



(a) Pot, with hallucinated points (b) Pot, hallucinated points removed

(c) Cup holder, with hallucinated points (d) Cup holder, hallucinated points removed

Fig. 8. Point clouds of a black pot (a) immediately after applying Algorithm 1 and (b) after applying our hallucination removal scheme. We similarly show point clouds of an object with more complex concavities (a paper cup holder) in (c) and (d). See Figure 11 for color images of these objects.

for why this happens for a cup: camera A's cone does not fully carve away the top of the cup, leaving the sliver of points covering the concavity.

We eliminate hallucinated points as follows: for each point *P* generated by the visual hull in the cloud generated by Algorithm 1, project *P* onto each refined depth map *D*. If *P*'s depth in *D*'s frame is strictly smaller than the current depth value in *D*, then discard *P*.

Figure 9 explains why this method works: camera A introduces a hallucinated point at the green dot. We can use camera B to resolve this discrepancy; projecting the green dot onto camera B's image plane shrouds the purple point, which camera B originally sees with Algorithm 1 (see Camera B's refined depth map). Eliminating this green dot resolves the

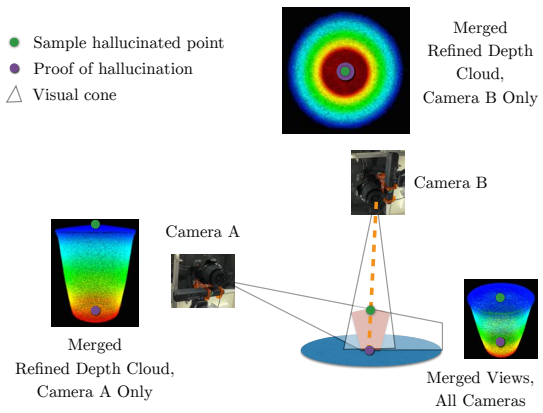


Fig. 9. Camera A's viewing cone does not fully carve away the visual hull, leaving a sliver of hallucinated points; the green dot shows a sample hallucinated point (see Camera A's refined depth map). Discussed in Section III-E, the existence of the purple dot along the orange ray allows us to detect the green dot as a hallucinated point.

contradiction that arises from the purple dot indicating free space along the orange ray.

Because we assume that the Carmines do not usually hallucinate poins, we iterate only through points generated by the visual hull. Further, as long as the KinectFusion model provides depth readings in concave regions of the object, the hallucinated points that "cover up" the concavity will be eliminated. See Figure 8b, d for merged clouds after hallucination removal of two sample objects. Finally, to remove stray points in this de-hallucinated cloud, we remove all points that do not have at least 5 neighbors within a radius of 1 mm.

F. Fusing the Unified Cloud and Soft Visual Hull

We now fuse the soft visual hull computed in Section III-B and "de-hallucinated" cloud computed in Section III-E into a single model. Similar to the visual hull formulation, for $x \in \mathbb{R}^3$, we define a function $F(x) = 1$ if x projects within all silhouettes *and* x's nearest neighbor to a point in the de-hallucinated cloud lies within a maximum distance $r = 1$ mm. After finding an initial x_0 on the surface by repeatedly sampling from the intersection of the back-projected silhouette bounding boxes, we run the Bloomenthal polygonizer to extract a mesh.

In practice, the polygonizer generates many triangles with poor aspect ratio, e.g. slivers.[1] Triangles with aspect ratios close to 1 are desirable, since this leads to cleaner meshes that are more easily editable in 3D software due to their connectivity properties; not explored in this paper, this also leads to cleaner texture maps [6]. To improve triangle aspect ratios while losing little surface detail, we alternate between (1) applying $\sqrt{3}$-subdivision without smoothing [15] and (2) applying an edge decimation procedure. Figure 10 shows a sample mesh before and after two iterations of this subdivision-decimate process.

[1]The aspect ratio of a triangle is defined as the ratio of the circumradius to twice the inradius; e.g., the aspect ratio of an equilateral triangle is 1.

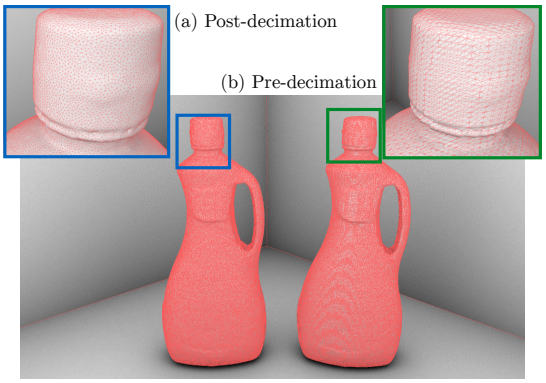


Fig. 10. Zooms (a) and (b) are meshes obtained after fusing together the visual hull and dense cloud after hallucination removal (Section III-F). (b) shows the mesh triangles before applying our decimation procedure. (a) shows the triangles afterwards. Note the substantially improved triangle quality, without the loss of surface geometry quality.

IV. Experiments

Figure 11 presents reconstructions of 19 distinct objects that fall in 3 categories: (1) simple and easy to reconstruct (relatively opaque, without concavities), (2) objects with at least one major concavity, and (3) objects with major translucencies or transparencies. Each row presents a different object category while each column presents reconstructions obtained from a different algorithm: the Poisson reconstruction method from [25] (PR), the hard visual hull method from [9] (VH), KinectFusion [22] (KF), and our approach. Images do not represent scanned scenes, but a collection of individually scanned objects to conserve space.

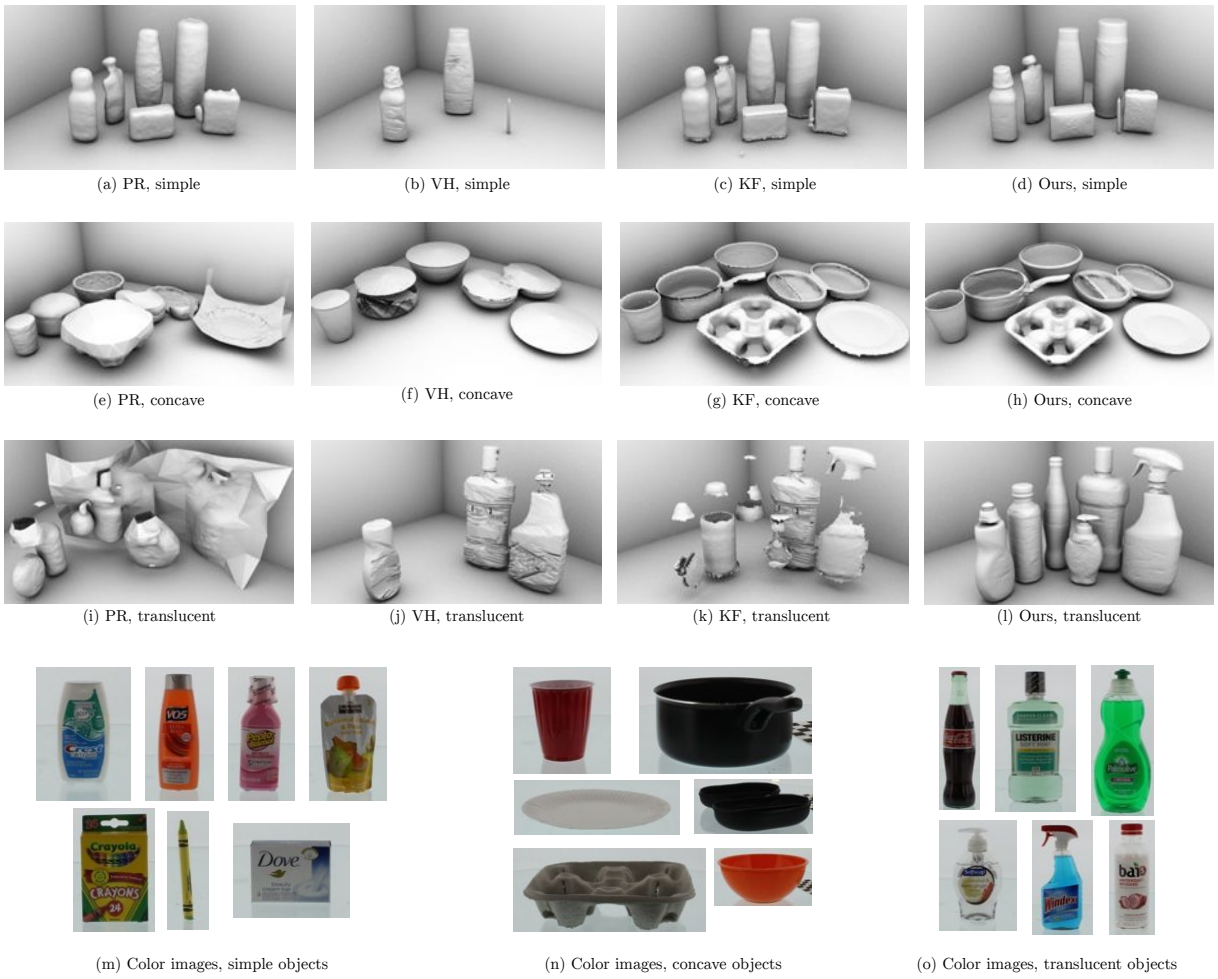
A. Simple Objects

Our method outperforms the other three competing methods in reconstructing simple objects. PR and KF tend to oversmooth surface details; further, neither method produces satisfactory reconstructions of the crayon. While PR produces closed meshes, KF often does not, evidenced by the noisy polygons towards the bottom of the objects. Although VH recovers the crayon, it overcarves the VO5 shampoo and Pepto Bismol, and fails to reconstruct the other objects due to excessive carving.

Our approach preserves the best aspects of each method: we recover surface details without excessive carving, evidenced by the properly reconstructed crayon and cap details for the Pepto Bismol, spray adhesive, and shampoo, while retaining closed meshes.

B. Objects with Concavities

In reconstructing concave objects, PR and VH miss surface details and hallucinate polygons. PR produces hallucinated polygons that cover concavities for the red cup, paper cup holder, carrying tote, and black pot as well as extraneous polygons around the paper plate. Due to excessive carving, VH fails to recover the paper cup holder, mangles the pot's surface, and fails to recover the pot's handle. KF and our method both recover all concavities. However, our method produces more refined models: our pot's handle is more



(a) PR, simple (b) VH, simple (c) KF, simple (d) Ours, simple

(e) PR, concave (f) VH, concave (g) KF, concave (h) Ours, concave

(i) PR, translucent (j) VH, translucent (k) KF, translucent (l) Ours, translucent

(m) Color images, simple objects (n) Color images, concave objects (o) Color images, translucent objects

Fig. 11. Collections of scanned objects reconstructed by competing methods. Rows 1-3 present objects that (1) are "simple", i.e. have few concavities, and are mostly opaque), (2) have major concavities, and (3) have major transparencies/translucencies, respectively. Columns 1-4 present reconstructions produced by the Poisson reconstruction-based method in the original BigBIRD [25], [14] (PR), the visual hull method in [9] (VH), KinectFusion [22] (KF), and our approach, respectively. Color images of simple, concave, and translucent/transparent objects are presented in (m)-(n), respectively. Each image does not present a scanned scene, but a collection of individually scanned objects to conserve space.

clearly defined, and the edges of our objects are not as rough due to our closed meshes.

C. Objects with Translucencies

PR produces many hallucinations in reconstructing translucent objects in an effort to reconstruct areas with few depth points. Unfortunately, this yields nearly unrecognizable reconstructions. Again, VH overcarves objects, entirely missing 3 objects. For the remaining 3, VH misses the top halves of the Palmolive and Windex bottles and overcarves the Listerine bottle. KF recovers bits and pieces of all objects, but still misses large regions due to the translucencies.

Our method recovers the majority of objects, including tiny surface details: we recover the opening to the Palmolive bottle, indentations on the Coca-Cola bottle where the label is present, and bottle caps of the Coca-Cola and Dragon Fruit juice bottles, all details on the order of 1 mm or less. Further, our method properly recovers objects that are white, with large clear regions, shown by the Softsoap Hand soap and Bai5 Sumatra Dragon Fruit juice.

Our approach is not perfect however, as it fails to recover regions of the Windex's pipe and Palmolive bottle. Our automatic segmentations fail to recover the superpixels corresponding to these regions, as our segmentation method classifies the white/translucent colors of the pipes as part of the background.

D. Quantitative Measurements

Quantitatively measuring errors associated with the BigBIRD dataset is nontrivial, as there exists no ground truth data. We inspect 4 objects that can be decomposed into a few primitives: a Pringles can, an almonds container, a Dove soap box, and a 3M spray (Figure 12). We model the Pringles, almonds, and 3M spray containers using 3 cylinders stacked on each other and the Dove soap box using a rectangular prism. We determined the appropriate dimensions per primitive using calipers that are accurate to 0.1 mm. After fitting the appropriate ground truth model to each reconstructed object via point-to-plane ICP, we obtain the RMS errors in Table I; we compare our method to the

(a) Almonds Can (b) Dove Soap Box (c) Pringles Can (d) 3M Spray

Fig. 12. The Almonds Can, Dove Soap Box, Pringles Can, and 3M spray, which we use for quantitative measurements. We model the Almonds, Pringles, and 3M Cans using 3 cylinders: the cap, the container, and the bottom. We model the Dove Soap Box using a rectangular prism.

Primitive Fitting RMS Errors (mm)				
	PR [25]	SVH	KF [22]	OUR METHOD
Pringles	0.566	**0.541**	0.850	0.563
Dove Soap	0.995	0.981	1.123	**0.948**
Almond Can	0.339	0.303	0.662	**0.294**
3M Spray	2.018	1.971	2.189	**1.958**

TABLE I

RMS ERRORS FROM FITTING GROUND TRUTH MODELS TO RECONSTRUCTIONS FROM POISSON RECONSTRUCTION (PR), THE SOFT VISUAL HULL (SVH), KINECTFUSION (KF), AND OUR METHOD.

soft rather than the hard visual hull, as the latter heavily overcarves the Dove box and Almond Can. Thanks to our well-calibrated cameras, most RMS errors are under 2 mm. In the case of the Dove Soap, Almond Can, and 3M spray our method produces reconstructions with the least RMSE; with the Pringles, our method comes in a close second place.

V. CONCLUSION

This paper reasons through the advantages and shortcomings of the KinectFusion and visual hull techniques, and arrives at a highly effective method to fuse these models together. Individually, the KinectFusion algorithm does poorly in reconstructing objects with major translucencies but reconstructs concavities, while the visual hull does poorly in reconstructing concavities but reconstructs regions with major translucencies. Our method exploits the complementary nature of KinectFusion and visual hull to produce a unified algorithm that properly recovers objects with concavities and translucencies.

VI. ACKNOWLEDGEMENTS

KN was supported by an NSF Graduate Fellowship (2013 – 2014) and by an NDSEG Fellowship (2014 – 2015). AS was supported by an NDSEG Fellowship. This research was funded in part by ONR through a Young Investigator Program award. We thank NVIDIA for donating a Tesla K40 GPU. We thank Animesh Garg for useful discussions.

REFERENCES

[1] Rgbdtoolkit. http://www.rgbdtoolkit.com.
[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels. Technical report, 2010.
[3] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.
[4] B. G. Baumgart. Geometric modeling for computer vision. Technical report, DTIC Document, 1974.
[5] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.
[6] Hernández C.E. and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *CVII*, 96(3):367–392, 2004.
[7] W. C. Chiu, U. Blanke, and M. Fritz. Improving the kinect by cross-modal stereo. In *BMVC*, volume 1, page 3, 2011.
[8] R. T. Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363, 1996.
[9] K. Forbes, A. Voigt, N. Bodika, et al. Visual hulls from single uncalibrated snapshots using two planar mirrors. In *Pattern Recognition Association of South Africa*, 2004.
[10] J. Franco, E. Boyer, et al. Exact polyhedral visual hulls. In *BMVC*, volume 1, pages 329–338, 2003.
[11] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. In *ECCV*, pages 564–577. 2006.
[12] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 32(8):1362–1376, 2010.
[13] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, pages 3121–3128, 2011.
[14] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing*, 2006.
[15] Leif Kobbelt. 3-subdivision. In *Computer Graphics and Interactive Techniques*, pages 103–112, 2000.
[16] P. Labatut, J-P. Pons, and R. Keriven. Robust and efficient surface reconstruction from range data. In *CGF*, volume 28, pages 2275–2290, 2009.
[17] A. Ladikos, S. Benhimane, and N. Navab. Efficient visual hull computation for real-time 3d reconstruction using cuda. In *CVPR Workshop*, pages 1–8, 2008.
[18] S. Lazebnik, Y. Furukawa, and J. Ponce. Projective visual hulls. *IJCV*, 74(2):137–165, 2007.
[19] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987.
[20] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Computer Graphics and Interactive Techniques*, pages 369–374, 2000.
[21] P. Mücke, R. Klowsky, and M. Goesele. Surface reconstruction from multi-resolution sample points. In *VMV*, pages 105–112, 2011.
[22] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011.
[23] D. Schneider. Visual hull. Technical report, The University of Edinburgh, 2009.
[24] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, volume 1, pages 519–528, 2006.
[25] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *ICRA*, 2014.
[26] J. Smisek, M. Jancosek, and T. Pajdla. 3d with kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. 2013.
[27] F. Steinbrucker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *ICCV-W*, pages 719–722, 2011.
[28] C. Strecha, R. Fransens, and L. Van Gool. Wide-baseline stereo from multiple views: a probabilistic account. In *CVPR*, volume 1, pages I–552, 2004.
[29] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics*, 23(4):583–598, 1999.
[30] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS-W RGB-D: Advanced Reasoning with Depth Cameras*, 2012.
[31] D. Xu, J. Cai, T. J. Cham, P. Fu, and J. Zhang. Kinect-based easy 3d object reconstruction. In *Advances in Multimedia Information Processing–PCM 2012*, pages 476–483. 2012.
[32] Q. Zhou and V. Koltun. Simultaneous localization and calibration: Self-calibration of consumer depth cameras.
[33] Q. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *ICCV*, pages 473–480, 2013.