# Convex Bricks: A New Primitive for Visual Hull Modeling and Reconstruction

Chari, V.; Agrawal, A.; Taguchi, Y.; Ramalingam

TR2012-033    May 2012

**Abstract**

Industrial automation tasks typically require a 3D model of the object for robotic manipulation. The ability to reconstruct the 3D model using a sample object is useful when CAD models are not available. For textureless objects, visual hull of the object obtained using silhouette- based reconstruction can avoid expensive 3D scanners for 3D modeling. We propose convex brick (CB), a new 3D primitive for modeling and reconstructing a visual hull from silhouettes. CB's are powerful in modeling arbitrary non-convex 3D shapes. Using CB, we describe an algorithm to generate a polyhedral visual hull from polygonal silhouettes; the visual hull is reconstructed as a combination of 3D convex bricks. Our approach uses well studied geometric operations such as 2D convex decomposition and intersection of 3D convex cones using linear programming. The shape of CB can adapt to the given silhouettes, thereby significantly reducing the number of primitives required for a volumetric representation. Our framework allows easy control of reconstruction parameters such as accuracy and the number of required primitives. We present an extensive analysis of our algorithm and show visual hull reconstruction on challenging real datasets consisting of highly non-convex shapes. We also how real results on pose estimation of an industrial part in a bin-picking system using the reconstructed visual hull.

*IEEE International Conference on Robotics and Automation (ICRA)*

# Convex Bricks: A New Primitive for Visual Hull Modeling and Reconstruction

Visesh Chari*, Amit Agrawal†, Yuichi Taguchi†, and Srikumar Ramalingam†
*INRIA Rhône-Alpes †Mitsubishi Electric Research Labs (MERL)
*visesh.chari@inrialpes.fr †{agrawal,taguchi,ramalingam}@merl.com

***Abstract*— Industrial automation tasks typically require a 3D model of the object for robotic manipulation. The ability to reconstruct the 3D model using a sample object is useful when CAD models are not available. For textureless objects, *visual hull* of the object obtained using silhouette-based reconstruction can avoid expensive 3D scanners for 3D modeling. We propose convex brick (CB), a new 3D primitive for modeling and reconstructing a visual hull from silhouettes. CB's are powerful in modeling arbitrary non-convex 3D shapes. Using CB, we describe an algorithm to generate a polyhedral visual hull from polygonal silhouettes; the visual hull is reconstructed as a combination of 3D convex bricks. Our approach uses well-studied geometric operations such as 2D convex decomposition and intersection of 3D convex cones using linear programming. The shape of CB can adapt to the given silhouettes, thereby significantly reducing the number of primitives required for a volumetric representation. Our framework allows easy control of reconstruction parameters such as accuracy and the number of required primitives. We present an extensive analysis of our algorithm and show visual hull reconstruction on challenging real datasets consisting of highly non-convex shapes. We also show real results on pose estimation of an industrial part in a bin-picking system using the reconstructed visual hull.**

## I. Introduction

Shape reconstruction from silhouettes or occluding contours is an important and classical problem in computer vision with applications in diverse fields such as virtual reality, computer graphics, 3D modeling, and robotics. Several robotic tasks like grasping [13], [26], visual servoing [22], [7], and autonomous navigation [12] require the use of 3D CAD models. When prior CAD models are not available, or when available models have to be modified to incorporate new parts, algorithms for 3D reconstruction from images may be used to accomplish the task. The *visual hull* of an object obtained using silhouette based reconstruction is well-suited as a 3D model, especially for texture-less and industrial parts. It can avoid expensive 3D scanners for 3D modeling, since it only requires 2D images that can be captured using the camera mounted on the robot arm.

As defined by Laurentini [15], visual hull is the maximal object shape that is consistent with the silhouettes of the object. In principle, it can be obtained by the intersection of back-projected visual cones of all the silhouettes. Baumgart [1] showed how to build polyhedral visual hull by representing each occluding contour using polygons and computing the 3D polyhedral intersections of resulting visual cones. Volume based approaches reconstruct the 3D volume of the object, usually by discretization of the 3D space into

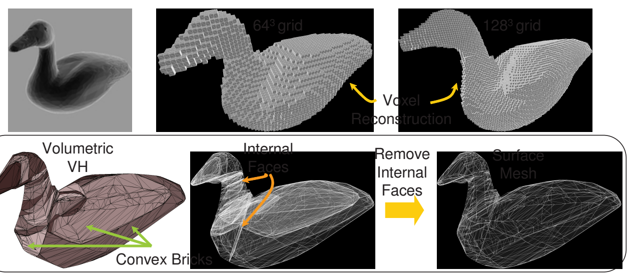

Fig. 1. VH reconstruction on Duck dataset using 10 silhouettes. Voxel carving starting from $64^3/128^3$ grid results in $21784/160372$ voxels, with $6503/26080$ voxels on the surface respectively. To obtain the same precision as $128^3$ grid, our volumetric reconstruction requires only 109 convex bricks (shades of red-brown) with 1551 vertices. Surface mesh can be easily obtained by removing internal planes using our representation.

voxels [5], [23], [28], [35], [27], [33]. In voxel carving (VC), each voxel is projected onto the images and is removed if it lies outside the silhouette. The *shape* of voxel remains fixed (cube), and a desired precision is achieved by increasing the resolution of the volumetric representation via reducing the voxel size. Since voxel shape remains fixed, recent research in VC has focused primarily on efficient implementation, e.g., using GPU's [30].

In this paper, we demonstrate that a richer volumetric representation can be obtained by modifying the voxel shape. We propose *convex brick* as a new 3D primitive for volumetric visual hulls (VH). A convex brick (CB) can be described as a voxel whose shape is not fixed, but depends on the given set of input silhouettes. Intuitively, if one is also allowed to modify the shape of the primitive, it can better represent a visual hull with smaller number of primitives. Consider Figure 1, where a large number of voxels are required for volumetric representation of a simple shape. Even if voxels inside the VH are discarded, 26080 surface voxels are required to obtain a precision of 1 pixel (maximum deviation of re-projected VH from silhouettes). In contrast, only 109 convex bricks are required for volumetric reconstruction, reducing the number of primitives by orders of magnitude. Notice that the CB representation does not have the 'blockiness' or quantization artifacts. In addition, we simultaneously obtain a surface representation by simply removing internal faces between convex bricks. Thus, our approach bridges the gap between volumetric and surface based representations, by providing both.

Using CB, we present an algorithm to reconstruct polyhe-

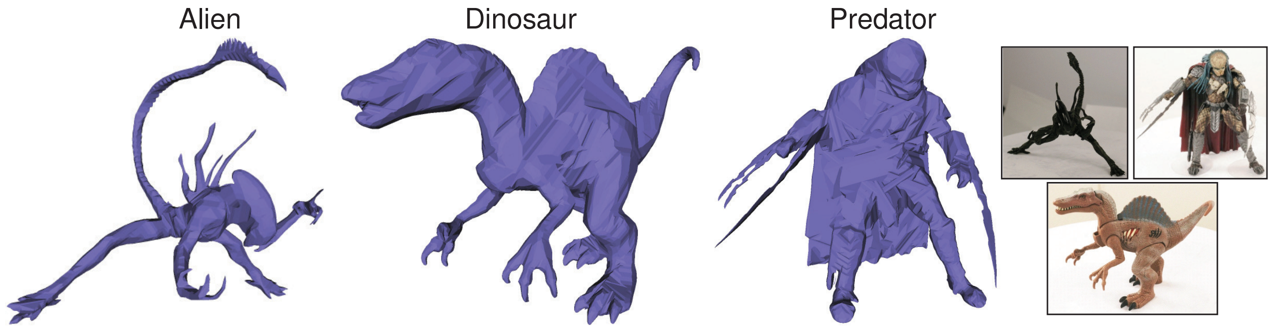

Fig. 2. High quality visual hull reconstruction on real data using convex bricks. The shapes are complex and highly non-convex. From left to right: Number of CB's: 23107, 21120, and 35932. Number of vertices: 106872, 101325, and 156243. Data courtesy Yasutaka Furukawa.

dral visual hulls from polygonal silhouettes. Our approach is similar in spirit to VC. We project each convex brick onto silhouettes and compute 2D intersections. However, the intersections in 2D are lifted back to 3D to continually *refine* the shape of convex bricks, rather than keeping the shape fixed to a cube. This allows us to have a representation independent of any 3D discretization. The 2D intersections are decomposed into convex partitions, leading to 3D intersections between convex polyhedras. These can be precisely computed using linear programming, thereby avoiding degeneracies in general volume-volume intersections. Voxels can be seen as a special case of CB. As shown in Figure 2, our approach generates detailed VH for highly complex and non-convex shapes.

**Contributions:** Our paper has the following contributions:

- We introduce the concept of convex bricks as a 3D primitive for representing and reconstructing VH.
- We describe an algorithm to reconstruct polyhedral VH for non-convex shapes using convex bricks by only employing convex 3D intersections.
- We demonstrate the flexibility of convex bricks in controlling the precision as well as the number of primitives for VH reconstruction.

**Benefits and Limitations:** A representation using CB has following benefits: (a) Unlike voxels, the representation is independent of any 3D discretization. Both the size and shape of CB's depend on the inherent *non-convexity* of the shape, rather than on the surface area as in VC [35]. (b) CB representation produces high quality reconstruction using lower number of primitives than VC. (c) It can provide both volumetric and surface information, while previous approaches result in either of two. (d) Using CB, a 3D convex decomposition of the shape is obtained automatically, useful for applications that require such decompositions. To the best of our knowledge, ours is the first paper to propose silhouette-dependent voxels as a 3D primitive and to describe their applicability for volumetric visual hulls. While our algorithm is slower than VC, it is not a drawback in a real application, since the CAD model generation has to be done once and can be done offline.

## A. Related Work

The problem of reconstructing surfaces from occluding contours was first considered by Koenderink [14], who attempted to infer 3D curvature properties from occluding contours. Most of the approaches for visual hull reconstruction can be divided into two broad categories: (a) volume-based approaches and (b) surface-based approaches. While volume-based approaches reconstruct the 3D volume, surface-based approaches [8], [9], [24], [18], [31], [34] attempt to reconstruct the elements of object surface such as surface patches or individual strips.

Boyer and Franco [2] improved voxel carving by performing a conforming Delaunay triangulation of visual hull surface points obtained along viewing edges. However, their approach also either keeps or discards each tetrahedron based on the projection of its centroid and does not modify its shape. Unlike 3D modeling approaches such as [6] that use parameterized primitives and recover the parameters using images, convex bricks are not parameterized.

Lazebnik *et al.* [16] showed that the visual hull surface is a projective topological polyhedron made of curve edges and faces connecting them. The visual hull is computed via locating frontier and triple points by finding a sparse set of corresponding points observed by pairs and triplets of cameras. Several approaches assume local smoothness and compute rim/frontier points using epipolar constraints based on second-order approximation of the surface [14], [36]. However, the orientations reverse at frontier points leading to an approximate topology. Recent state-of-the-art approach by Franco and Boyer [8] has shown high quality visual hull reconstruction by first retrieving the viewing edges. Local connectivity and orientation information is then used to incrementally build a mesh using epipolar correspondences. A final walk-through is also required to identify the planar contours for each face of the polyhedron. Such approaches can face difficulties in presence of segmentation and calibration errors; an epipolar line corresponding to a viewing edge may not intersect the silhouette. Thus, explicit handling of such cases is required, either by modifying local geometry or silhouettes. In contrast, such errors are implicitly handled in volume-based approaches such as ours. Furthermore, since typical outputs of such approaches are surface meshes, they lag behind volumetric outputs like ours in representational power. For example, it is well known that convex objects like CB's are well suited to answer many geometric queries like ray intersection, point-in-polygon etc.

Apart from these two class of methods, other approaches to visual hull reconstruction have been proposed. Brand *et al.* [3] proposed an algebraic solution that exploits the differential structure of the manifold and estimates the set of tangent planes and points of contact to the surface corresponding to 2D contour tangents. Matusik *et al.* [25] proposed image-based visual hull for fast image-based rendering. However, their approach does not reconstruct a 3D visual hull. A Bayesian approach enforcing a class-specific shape prior was proposed in [10]. Along with silhouette information, photo-consistency has also been used to improve the 3D shape [9], [32], [11], [37], although we focus solely on using silhouette information.

Several applications on non-convex polyhedra have efficient solutions if the polyhedron is first decomposed into convex pieces. These include geometric problems such as computing the Minkowski sum of polyhedras, point-in-polyhedra test [17], as well as applications such as collision detection and motion planning [20]. VH obtained using our approach can be directly used without any post-processing for these applications. In addition, approximate convex decomposition [19] of the shape is often desired, since exact decomposition into a minimum number of pieces is NP-hard. We also show how to achieve approximate convex decomposition by controlling the precision of VH as well as number of primitives in our algorithm.

## II. Visual Hull Reconstruction

Consider an object observed by $N$ pinhole cameras with known calibration. Let $\mathcal{I}_i$ denote the set of images and $\mathcal{C}_i$ the silhouette in each image. We assume that the silhouette in each image is a set of polygonal contours. The viewing cone $\mathcal{V}_i$ associated with the silhouette $\mathcal{C}_i$ is the closure of the set of rays passing through points inside $\mathcal{C}_i$ and through the center of projection of $\mathcal{I}_i$. The visual hull $(\mathcal{VH})$ is defined as the intersection of all viewing cones:

$$\mathcal{VH} = \cap \mathcal{V}_i. \tag{1}$$

Our approach to visual hull (VH) reconstruction consists of (a) initializing a coarse model and (b) refining it by partitioning into convex bricks using given silhouettes. Any initialization such as a bounding box can be used. Note that a convex decomposition of 3D shape is not unique. In Section V, we discuss initialization techniques based on VC, resulting in a different convex decomposition.

**Initialization:** Instead of using a bounding box, we initialize using $\mathcal{CVH}$, the apparent (view-dependent) convex hull of the shape [38]. This is obtained by computing the 2D convex hull of each silhouette and finding the intersection of all convex visual cones. $\mathcal{CVH}$ can be computed by solving a single linear program, and is a tighter convex approximation compared to the bounding box.

### A. Refinement by Partitioning into Convex Bricks

The silhouettes are processed one by one and each silhouette is used to refine the current 3D model by potentially decomposing it into multiple convex bricks. Without

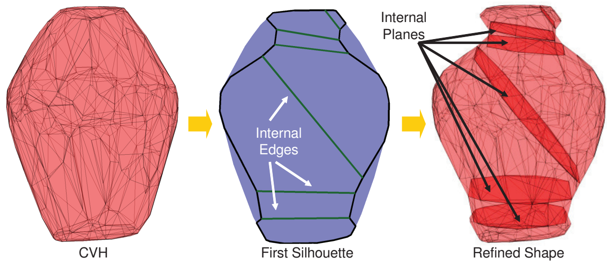

Fig. 3. Refinement of $\mathcal{CVH}$ using the first silhouette. Intersection of the projected $\mathcal{CVH}$ (blue) with the silhouette results in the same silhouette. It is partitioned into $m = 7$ convex polygons using internal edges (green). $\mathcal{CVH}$ is refined into 7 convex bricks. The internal edges result in internal faces.

loss of generality, let us assume that we have $K_i$ convex bricks $B_1 \dots B_{K_i}$ describing the object after processing $i^{th}$ silhouette. Thus, $K_0 = 1$ and $B_0 = \mathcal{CVH}$. Each silhouette is processed only once.

Intuitively, we wish to intersect the visual cone of each silhouette with the current model to remove 3D regions that project outside the silhouette. However, since silhouettes are typically non-convex, such general 3D intersections become tricky. We now show how to achieve such 3D intersections using only convex 3D intersections.

**Silhouette Processing:** Given a new silhouette $\mathcal{C}_i$, each convex brick $B_j$ is processed independently. We wish to find the 3D intersection of $B_j$ with the visual cone corresponding to the current silhouette. We first project $B_j$ onto the silhouette resulting in a 2D convex polygon $\mathcal{P}(B_j)$, where $\mathcal{P}$ denotes the projection operator. Next, we compute the intersection $\mathcal{S}_{ij}$ of $\mathcal{P}(B_j)$ with the given silhouette $\mathcal{C}_i$:

$$\mathcal{S}_{ij} = \mathcal{P}(B_j) \cap \mathcal{C}_i. \tag{2}$$

However, the intersection $\mathcal{S}_{ij}$ can be non-convex. We then find a convex decomposition of $\mathcal{S}_{ij}$, resulting in $m_{ij}$ convex 2D polygons. Thus,

$$\mathcal{S}_{ij} = \bigcup_{k=1}^{m_{ij}} \mathcal{T}_{ijk}, \tag{3}$$

where $\mathcal{T}_{ijk}$'s denote the convex partitions of $\mathcal{S}_{ij}$. Each partition $\mathcal{T}_{ijk}$ define a *convex* visual cone, which can be intersected with $B_j$ by solving a small linear program. Thus, we have simplified the intersection of each convex brick $B_j$ with the visual cone of the given non-convex silhouette by only using convex 3D intersections. Figure 3 demonstrates this procedure for the first silhouette used in the Vase dataset, where the $\mathcal{CVH}$ is refined and partitioned into 7 convex bricks. Figure 4 shows processing at a later stage, where a convex brick is only refined, but not partitioned.

Achieving 3D intersections via 2D intersections in the image plane is also used by previous approaches [24], [25], [29], [31]. Note that if the convex brick is projected inside the silhouette, it is not refined since the current silhouette does not provide any new information for that brick.

### B. Obtaining Surface Mesh

The convex decomposition of $\mathcal{S}_{ij}$ will introduce new edges, referred to as *internal edges*, since they are not part of

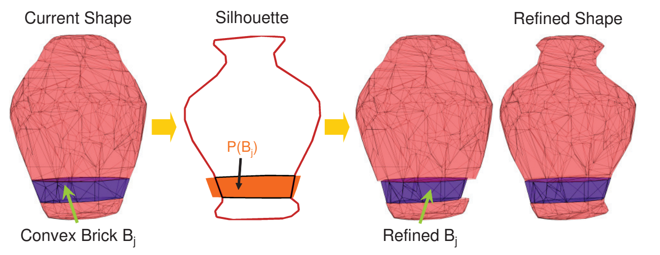

Fig. 4. Refinement of convex brick $B_j$. The projection of $B_j$ is intersected with the silhouette (red) resulting in a convex polygon (black). Convex brick is refined without getting partitioned. Shape obtained after processing all convex bricks with this silhouette is also shown.

the original silhouette. The internal edges result in internal faces in the reconstructed VH (Figure 3). Interestingly, there are no internal vertices, *i.e.*, none of the vertices of the reconstructed VH will fall inside *all* silhouettes. The reason is as follows. After the first silhouette is processed, all convex bricks have vertices that project on that silhouette. Further, the 2D convex decomposition does not introduce new vertices. Thus, for every additional view, the vertices of the refined convex bricks will project on the current or the previous silhouettes. This implies that one can obtain a surface mesh by simply removing the internal faces, which are marked during the processing. Thus, CB representation can provide both volumetric and surface information. Figures 1 and 5 show examples. In Section V, we show that initialization using VC results in internal vertices in reconstructed VH.

## C. Differences & Similarities with Voxel Carving

Similar to VC, the convex brick is refined only when it falls on the silhouette. In VC, a voxel is divided into 8 smaller voxels by keeping the shape fixed to a cube, while CB is divided into smaller CB's using 2D convex decomposition. Given a silhouette, a voxel may need to be divided several times to reach a given accuracy. In contrast, a convex brick needs to be divided only once for a given silhouette. This is because any non-convex 2D polygon can be decomposed into multiple convex partitions. Hence, CB refinement using a non-convex silhouette can be done via convex intersection with back-projected convex partitions. Thus, for piecewise-linear *polygonal* silhouettes, the projection of the reconstructed VH matches the polygonal silhouettes[1].

As discussed in [8], since VH shape is stable under small perturbations, using bounded polygonal approximations of occluding contours will have little impact on the global shape. Our approach can work with *any* polygonal approximation of contours. This is useful in practice, since in presence of noise and discretization, viewing cones never exactly exhibit tangency property [8].

We define the visual hull precision $\alpha$ as the average of the maximum deviation of the re-projected visual hull from each silhouette. We use the real dataset provided by

[1]To clarify, the projected VH is not guaranteed to match the underlying continuous (smooth) silhouette.

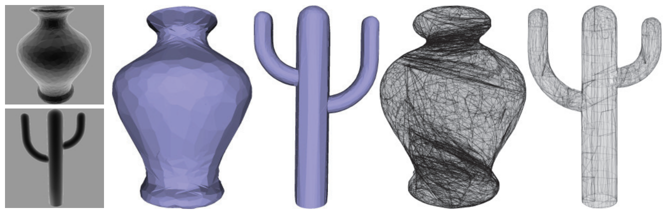

Fig. 5. (Left) Original mesh of Vase and Cactus. (Middle) Reconstructed VH using 30 images with 1667 and 766 convex bricks for Vase and Cactus respectively. (Right) Surface mesh with 15283 and 8999 vertices respectively.

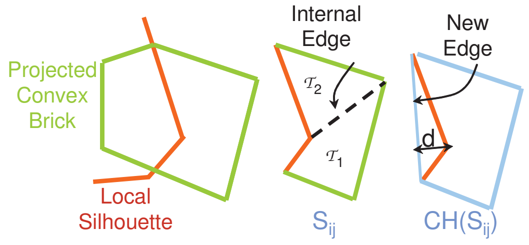

Fig. 6. Convex brick projected onto a silhouette results in the intersection $S_{ij}$, which has two convex partitions. Ideally, the brick will be refined into two smaller bricks. For precision control, the maximum distance $d$ of the silhouette from the convex hull of $S_{ij}$ is computed. If $d < \alpha$, convex brick is not divided, only refined using the new edges in $CH(S_{ij})$.

Yasutaka Furukawa[2], which provides silhouettes for several challenging non-convex shapes. For synthetic datasets, we render the silhouettes from the available mesh model[3]. We fit lines (tolerance of 1 pixel) to pixel-discretized contour points resulting in piecewise-linear polygonal silhouettes. Thus, the maximum deviation of the projected polyhedral VH with continuous smooth silhouettes will be 1 pixel. Figure 2 shows high quality reconstruction on challenging real dataset consisting of several non-convex shapes. Figures 5 and 1 show reconstruction on the synthetic datasets along with obtained surface mesh.

## D. Controlling the Precision of Visual Hull

Voxel carving provides a natural way to control the VH precision via voxel resolution at the finest level. We demonstrate a novel way to control VH precision by locally controlling the convex decomposition of projected convex bricks. Figure 6 depicts the idea where $\mathcal{S}_{ij}$ will ideally be decomposed into two convex partitions. To enable precision control, we compute the convex hull of $\mathcal{S}_{ij}$, denoted as $CH(\mathcal{S}_{ij})$. Given a precision $\alpha$, if the silhouette lies within a distance $d < \alpha$ of $CH(\mathcal{S}_{ij})$, $\mathcal{S}_{ij}$ is not divided into convex partitions. Subsequently, the convex brick $B_j$ is not divided, but only refined using the new edges in $CH(\mathcal{S}_{ij})$. Thus, locally convex approximation of the shape is obtained. Note that when the silhouette itself is locally convex, $\mathcal{S}_{ij}$ is convex. Thus, $\mathcal{S}_{ij} = CH(\mathcal{S}_{ij})$ and $d$ equals zero. This implies that there is no approximation for convex bricks falling onto locally convex regions of the silhouette.

Figure 7 shows VH reconstruction using varying $\alpha$ for Dinosaur dataset (image resolution $2000 \times 1500$). Even when

[2]http://www.cs.washington.edu/homes/furukawa/research/visual_hull/
[3]http://www.cs.caltech.edu/~njlitke/meshes/toc.html

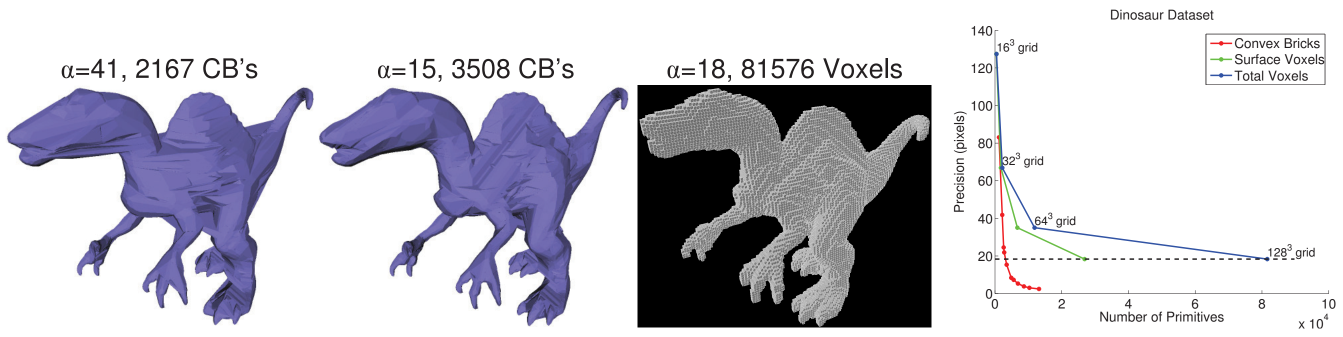

Fig. 7. (Left) Reconstructed VH using different precision. Even when $\alpha$ is large, the obtained VH retains high frequency details to a large extent. (Right) Number of primitives versus precision for VH reconstruction using convex bricks and voxels.

$\alpha$ is large, our reconstruction provides high frequency shape details. Since not all convex bricks are divided, precision control results in lesser number of convex bricks (compare to 21120 bricks for full precision). In comparison, VC starting with $128^3$ grid results in a precision of 18 pixels using 81576 voxels (26858 on surface). By controlling the precision, our approach automatically results in an approximate convex decomposition of the VH. Thus, the VH can be directly used without any post-processing for applications such as collision detection and motion planning that require an approximate convex decomposition.

## III. Implementation Details

Our algorithm requires following well-studied geometric operations: (a) 2D polygon intersection, (b) 2D convex decomposition, and (c) 3D convex intersections. We use the General Polygon Clipper (GPC) library[4] for 2D polygon intersections. For 2D convex decomposition, we use the CGAL[5] implementation which has a complexity of $O(n \log n)$ for a polygon with $n$ vertices.

A convex polytope can be represented as the intersection of supporting half-spaces (H-representation) or as a minimal set of extreme vertices (V-representation). Each representation can be obtained from the other via vertex/facet enumeration. We represent each convex brick using both H-representation (for 3D intersections) and V-representation (for projection). Since half-spaces can be written as linear inequalities, intersection of two convex 3D polytopes is equivalent to computing the reduced set of inequalities from the union of their inequalities. For vertex enumeration from half-spaces and reducing the set of inequalities, we use the *lrs* software[6]. To refine a convex brick, we compute the back-projected planes given by the individual 2D convex partitions, add the corresponding linear inequalities to the current convex brick inequalities, and compute the reduced set. The rest of our implementation is in un-optimized Matlab and runs on a 64-bit PC with 4GB RAM. For all our results, we verified that the reconstruction is a watertight mesh by loading the mesh into CGAL and testing for polyhedron validity and closedness. Run-time (full precision) is less than 3 minutes for simple shapes (cactus, vase, duck) using 30 views and $30-40$ minutes for complex shapes shown in Figure 2 using 24 views. For Dinosaur dataset, run-time reduces to 6.5 minutes for $\alpha = 83$ and 8.6 minutes for $\alpha = 15$ (Figure 7). In general, each convex brick has less than 10 supporting planes in the final shape.

[4]http://www.cs.man.ac.uk/~toby/alan/software/
[5]http://www.cgal.org/
[6]http://cgm.cs.mcgill.ca/~avis/C/lrs.html

## IV. Analysis

Now we analyze various aspects of our algorithm and demonstrate the flexibility of our approach in controlling the precision of VH as well as the number of primitives used for reconstruction.

**Number of CB's:** We first consider the number of convex bricks $K_i$ after processing $i$ silhouettes. Figure 8 plots $K_i$ for 6 different datasets. Initially the number of CB's may increase rapidly, since the first few views provide a lot of new information for visual hull refinement. For example, after the first view is processed, the number of convex bricks directly correspond to the number of convex partitions of the first silhouette. During the first few views, convex bricks will project over larger silhouette regions and each will be refined into larger number of bricks. However, after few initial views, the increase in number of CB's is almost linear as empirically observed from the plots. As more views are processed, CB's become smaller and project to locally small regions of silhouette. In addition, a majority of them will also fall inside the silhouette and will not be divided.

**View Ordering:** The order in which views are processed also affects the final number of convex bricks. Two heuristics seems relevant, where the new view is (a) maximally different or (b) minimally different from previous views. Figure 8 (right) plots $K_i$ for these two ordering along with several random orderings for Dinosaur dataset. The first view is same for all orderings. When successive views are maximally different, refined convex bricks become more localized after each view, reducing the final number of CB's. When successive views are minimally different, refinement is slow resulting in convex bricks covering larger silhouette regions. We performed extensive tests and in our experience using the heuristic of maximally different views works well, both for synthetic and real datasets. For simplicity, the maximal different view is chosen by maximizing the distance of the new viewpoint with all previous viewpoints, although other approaches can be used.

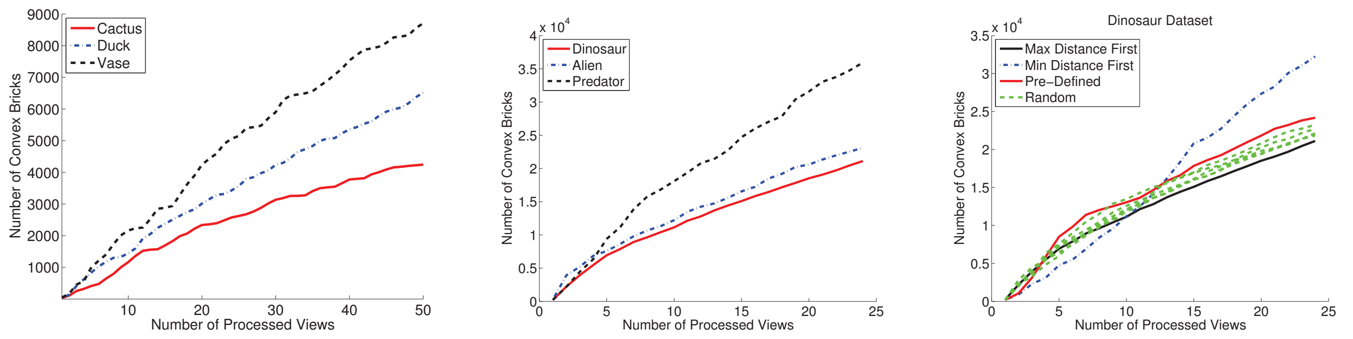

Fig. 8. Plot of number of convex bricks after each silhouette is processed for synthetic (left) and real datasets (middle). Increase in bricks is almost linear as more views are processed. (Right) Using a maximally different view-ordering results in smaller number of convex bricks. 'Pre-defined' refers to ordering provided in the dataset.

**Precision vs Number of Primitives:** Figure 7 (right) shows a comparison of the number of primitives required for reconstructing VH for a given precision using voxels and convex bricks. We run VC using various grid sizes ($16^3, 32^3, \ldots$). For each reconstruction, we compute the corresponding precision $\alpha$ and the required number of voxels. Then we run our approach with the same $\alpha$ and note the number of CB's required as well as the resulting precision[7]. Figure 7 shows that for similar precision, the number of CB's required is much smaller. Conversely, for the same number of primitives, convex bricks result in better precision. Note that precision improves much more rapidly with increase in convex bricks compared to increase in voxels.

**Controlling the Number of CB's:** Can we achieve a reconstruction by specifying the number of CB's? Such a control is easily achievable in our approach. As soon as the number of convex bricks reaches a specified number $T$, we stop dividing the convex bricks, but keep on refining them as described in Section II-D. This essentially amounts to setting $\alpha$ to a large value once $T$ CB's are obtained. Figure 9 shows Dinosaur VH using a fixed number of CB's. Even when the number of CB's is small, the reconstruction captures the topological information to a large extent. Interestingly, reconstruction using a *single* convex brick results in the apparent (view-dependent) 3D convex hull of the shape. This is a better approximation than using a single voxel (bounding box). Thus, similar to precision control, specifying the maximum number of convex bricks also results in an approximate convex decomposition of the reconstructed shape. For Dinosaur dataset, run-time reduces to 1.7 minutes for $T = 200$ and 5.1 minutes for $T = 1000$.

## V. Hybrid Approach using Voxel Carving

The main algorithm described in Section II-A starts with $\mathcal{CVH}$, a single convex brick. However, our approach can also use a voxel-based initialization obtained using VC. Since each voxel is convex, it can be regarded as a convex brick and each voxel can be refined as described in Section II-A *independently*. Figure 11 compares the reconstructions obtained using a $32^3$ voxel initialization with $\mathcal{CVH}$ initialization. The

[7]The final precision obtained using CB's will be smaller than $\alpha$ used.

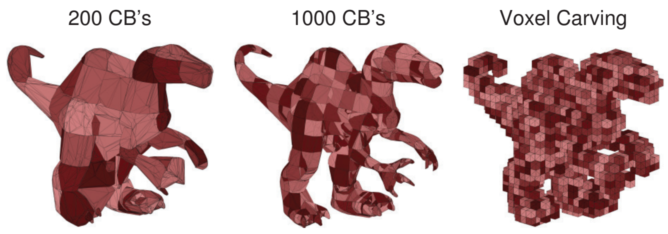

Fig. 9. Visual hull reconstruction of Dinosaur using fixed number of CB's (each is colored differently). Both reconstructions are approximate convex decomposition of the shape. VC output is also shown using 2102 voxels with 1659 voxels on the surface.

volumetric VH is shown using transparency, which reveals the difference in the internal structure of convex bricks.

Interestingly, a voxel initialization produces a *different* convex decomposition of the resulting VH compared to $\mathcal{CVH}$ initialization; the reconstructed VH now also contains internal vertices. Figure 10 depicts the reason. Compared to Section II-A, where all vertices of convex partitions are *on* the silhouette, the projected voxel vertices modify the 2D convex decomposition by adding new points *inside* the silhouette. Thus, convex bricks get more localized after processing the first silhouette. Notice the shape of convex bricks in Figure 11; voxel initialization avoids long thin slivers that might arise due to the convex decomposition of the silhouette when internal vertices are not added. In addition, intersection of several projected voxels with the silhouette may be convex, not requiring any partitions (Figure 10). This helps in reducing the number of convex bricks as shown in Figure 11. However, using a fine voxel grid initialization results in a large number of convex bricks to start with and may not provide a benefit.

Voxel initialization also helps in reducing the number of vertices in the final surface mesh. The intersection of projected convex bricks with the silhouette results in new vertices *on* the silhouette, increasing the number of vertices on the surface. This increase is smaller with voxel initialization, since a fraction of new vertices are added inside the silhouette during convex decomposition. To obtain the surface mesh, we remove all vertices (and their corresponding edges) whose projection lie inside all silhouettes, along with internal planes. For Dinosaur dataset, the number of vertices

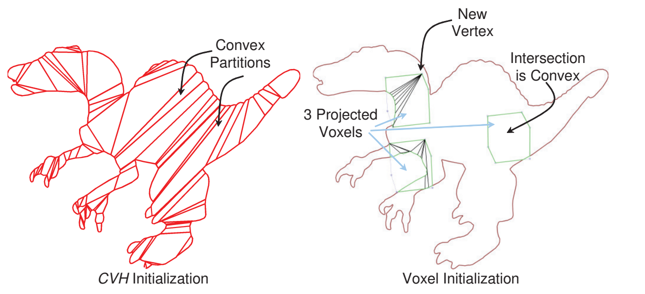

Fig. 10. (Left) Convex decomposition of the first silhouette for $\mathcal{CVH}$ initialization. All vertices lie on the silhouette. (Right) Voxel-based initialization essentially modify the convex partitions (black) by inserting new vertices inside the silhouette corresponding to the voxel projection. This better localizes the convex bricks.

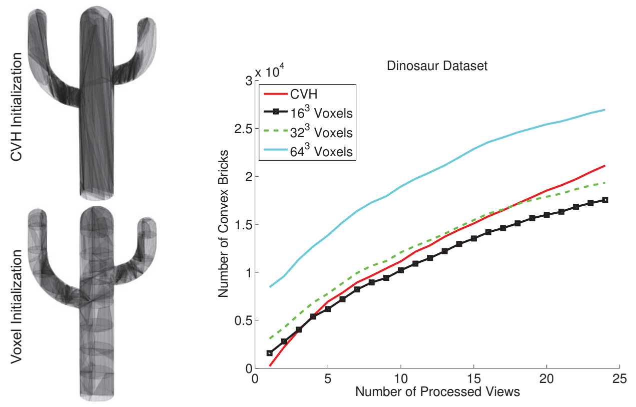

Fig. 11. (Left) Difference in volumetric VH with and without voxel initialization. (Right) Comparison of the number of convex bricks after processing each silhouette for different voxel-based initializations.

reduces from 101325 to 77603 using $32^3$ voxel initialization.

In general, we obtain a larger number of vertices compared to a pure surface-based approach [8]. The extra vertices lie on internal planes and are caused by the intersection of projected internal planes with the silhouette during the processing. In future, we plan to remove these extra vertices in post-processing.

## VI. Application to Bin Picking

The 3D model reconstructed using our approach can be used for several applications in robotics, including grasping, servoing, and navigation. Underlying all these applications is the ability of the robot to estimate the pose of the object of interest given some measurements.

Here we demonstrate the use of our algorithm for bin-picking applications. The objective of bin picking is to perform detection and pose estimation of objects that are randomly placed in a cluttered bin and to grasp an object using its estimated pose with a robot arm. The object detection and pose estimation typically require a 3D model of the target object, which we wish to reconstruct using the object itself.

**Visual Hull Reconstruction:** Figure 12 illustrates a pipeline of our system for the bin-picking application. We attached a camera to a robot arm and calibrated it with respect to the robot in order to capture a set of images of the object from multiple locations with known camera poses. We obtained mask images from the captured images (Figure 12 (left)) by using background subtraction. We then reconstructed a visual hull from the silhouettes extracted from the mask images using our CB algorithm. We used 34 views to reconstruct the visual hull shown in Figure 12 (center), consisting of 10466 CB's and 128506 vertices.

**Bin Picking:** We validated the accuracy of the reconstructed visual hull model in a bin-picking system presented in [21]. The system uses a multi-flash camera to robustly extract depth edges for the bin of objects. The system then performs detection and pose estimation of objects by using fast directional chamfer matching (FDCM) algorithm [21] that matches the extracted depth edges with template edge images computed from a 3D model of the object. The estimated pose is further refined by rendering the 3D model of the object and using a variant of iterative-closest point (ICP) algorithm for the edge points. In [21], a hand-made 3D CAD model was used for generating the template images of FDCM and for the pose refinement. We replaced it with our reconstructed visual hull model in the system and observed a similar pose estimation accuracy. Figure 12 (right) shows a pose estimation result using the FDCM algorithm. We also achieved a pickup success rate of 95% over 100 trials, which is comparable to the case when we use the hand-made 3D CAD model.

## VII. Discussions

We conceptualized and explored the notion of convex bricks as *silhouette-dependent* voxels for volumetric visual hull representation and reconstruction. Convex bricks provide a richer volumetric representation along with providing surface information. In addition to high quality reconstruction, they also provide a 3D convex decomposition useful for post-processing. Convex bricks offer tunable visual hull reconstructions by controlling the precision and the number of required primitives. We demonstrated that the reconstructed 3D model can be used in practical robotics applications such as pose estimation.

Our current unoptimized Matlab implementation is slower than voxel carving; however our approach is highly parallelizable, since each convex brick is refined independently. Fast inside/outside decision for projected convex brick may be possible using distance transforms and occlusion maps, which would improve run-time. The 2D polygon intersection algorithm can be replaced by a faster line clipping algorithm. In addition, incremental linear programming can further reduce the computation cost, since each new silhouette only adds few inequalities for each convex brick. Linear time algorithms such as [4] can also be used for convex 3D intersections.

We believe that convex brick as a 3D primitive opens up promising avenues for further research. Several inference algorithms in vision use super-pixels instead of pixels for computational benefits and better spatial support. Similarly, inference algorithms on graphs constructed using convex bricks as nodes have the potential to provide computational

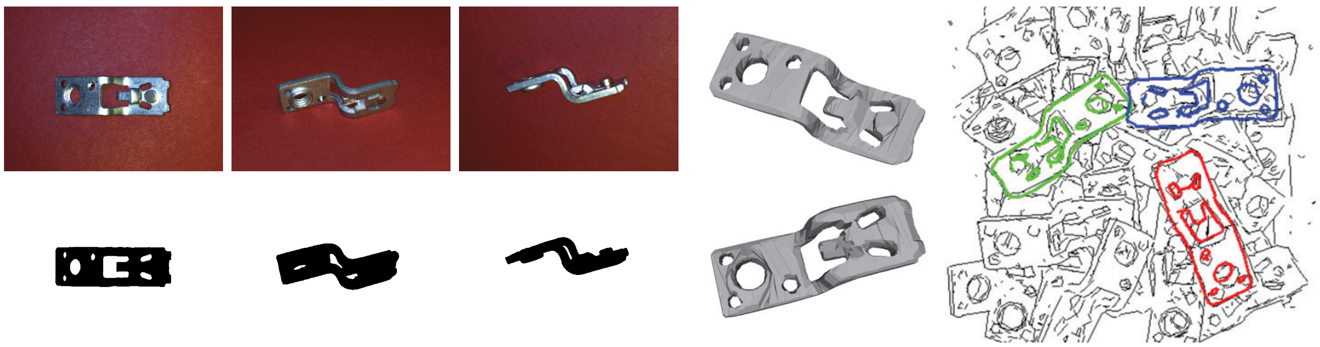

Fig. 12. (Left) Examples of input images and their corresponding mask images from which the silhouettes are extracted. (Center) Reconstructed visual hull. (Right) Pose estimation results in a bin-picking system presented in [21]. We used the reconstructed visual hull to generate template edge images for the fast directional chamfer matching (FDCM) algorithm. We used a multi-flash camera to extract depth edges from the scene and estimated object poses using FDCM. Best three estimated poses (red, green, and blue) are superimposed on the depth edges extracted from the scene.

benefits over voxels for applications such as 3D segmentation/recognition. Binary space partitioning (BSP) trees are used for representing 2D/3D convex partitions. A BSP-tree based data structure for convex bricks will be useful in reconstruction and processing of the visual hull. It would provide connectivity information between different parts of the 3D model. Finding the best 2D convex decomposition that minimize the number of convex bricks and/or surface vertices also remains an interesting future work.

**Acknowledgments:** We thank Jay Thornton, Oncel Tuzel, and Haruhisa Okuda for their helpful comments and feedback. This work was done while Visesh Chari was an intern at MERL with support from MERL.

## References

[1] B. G. Baumgart. *Geometric modeling for computer vision*. PhD thesis, Stanford U., Oct 1974. STAN-CS-74-463.
[2] E. Boyer and J. Franco. A hybrid approach for computing visual hulls of complex objects. In *CVPR*, volume I, pages 695–701, 2003.
[3] M. Brand, K. Kang, and D. Cooper. Algebraic solution for the visual hull. In *CVPR*, volume 1, pages 30–35, 2004.
[4] B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *Siam J. Comput.*, 21(4):671–696, Aug. 1992.
[5] K. M. Cheung. *Visual hull construction, alignment and refinement for human kinematic modeling, motion tracking and rendering*. PhD thesis, CMU, 2003.
[6] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.
[7] R. Feddema, J.T.; Simon. Visual servoing and CAD-driven microassembly. In *ICRA*, 1998.
[8] J. Franco and E. Boyer. Efficient polyhedral modeling from silhouettes. *PAMI*, 31:414–427, Mar. 2009.
[9] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. *IJCV*, 81:53–67, 2009.
[10] K. Grauman, G. Shakhnarovich, and T. Darrell. A bayesian approach to image-based visual hull reconstruction. In *CVPR*, volume 1, pages 187–194, 2003.
[11] C. Hernandez and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *CVIU*, 96:367–392, 2004.
[12] G. Horn, J.; Schmidt. Continuous localization for long-range indoor navigation of mobile robots. In *ICRA*, 1995.
[13] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz. Real-time CAD model matching for mobile manipulation and grasping. In *IEEE International Conference on Humanoid Robots*, 2009.
[14] J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.
[15] A. Laurentini. The visual hull concept for the silhouette-based image understanding. *PAMI*, 16:150–162, 1994.
[16] S. Lazebnik, E. Boyer, and J. Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. In *CVPR*, volume 1, pages 156–161, Dec. 2001.
[17] J. Li, W. Wang, and E. Wu. Point-in-polygon tests by convex decomposition. *Computer and Graphics*, 2007.
[18] M. Li, M. Magnor, and H.-P. Seidel. Hardware-accelerated visual hull reconstruction and rendering. In *Graphics Interface 2003*, pages 65–71, 2003.
[19] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra. In *Proc. ACM Symp. Solid and Physical Modeling*, 2007.
[20] H. Liu, W. Liu, and L. Latecki. Convex shape decomposition. In *CVPR*, pages 97–104, June 2010.
[21] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, A. Agrawal, and H. Okuda. Pose estimation in heavy clutter using a multi-flash camera. In *ICRA*, 2010.
[22] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2D-3D model-based approach. In *ICCV*, 1999.
[23] W. Martin and J. Aggarwal. Volumetric description of objects from multiple views. *PAMI*, 5(2):150–158, 1983.
[24] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Proc. Eurographics Workshop on Rendering*, 2001.
[25] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH*, pages 369–374, 2000.
[26] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *IROS*, 2007.
[27] H. Noborio, S. Fukuda, and S. Arimoto. Construction of the octree approximating three dimensional objects by using multiple views. *PAMI*, 10(6):769–782, Nov. 1988.
[28] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 40:1–29, 1987.
[29] A. Rivers, F. Durand, and T. Igarashi. 3d modeling with silhouettes. *ACM Trans. Graph*, 29:109:1–109:8, July 2010.
[30] A. Schick and R. Stiefelhagen. Real-time gpu-based voxel carving with systematic occlusion handling. In *Proc. 31st DAGM Symposium on Pattern Recognition*, pages 372–381, 2009.
[31] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21(3):53–61, 2001.
[32] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: a maximum-flow formulation. In *ICCV*, volume 1, pages 349–356, Oct. 2005.
[33] S. Srivastava and N. Ahuja. Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics and Image Processing*, 49:68–84, Jan. 1990.
[34] S. Sullivan and J. Ponce. Automatic model construction, pose estimation, and object recognition from photographs using triangular splines. *PAMI*, 20(10):1091–1096, 1998.
[35] R. Szeliski. Rapid octree construction from image sequences. *CVIU*, 58(1):23–32, 1993.
[36] R. Vaillant and O. Faugeras. Using extremal boundaries for 3-D object modeling. *PAMI*, 14(2):157–173, 1992.
[37] G. Vogiatzis, C. Hernandez, P. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photoconsistency. *PAMI*, 2007.
[38] Y. Wexler. View synthesis using convex and visual hulls. In *BMVC*, 2001.