Voting-Based Pose Estimation for Robotic Assembly Using a 3D Sensor

Changhyun Choi*, Yuichi Taguchi†, Oncel Tuzel†, Ming-Yu Liu†‡, and Srikumar Ramalingam†
*Georgia Institute of Technology †Mitsubishi Electric Research Labs (MERL) ‡University of Maryland
*cchoi@cc.gatech.edu †{taguchi,oncel,mliu,ramalingam}@merl.com

***Abstract*— We propose a voting-based pose estimation algorithm applicable to 3D sensors, which are fast replacing their 2D counterparts in many robotics, computer vision, and gaming applications. It was recently shown that a pair of oriented 3D points, which are points on the object surface with normals, in a voting framework enables fast and robust pose estimation. Although oriented surface points are discriminative for objects with sufficient curvature changes, they are not compact and discriminative enough for many industrial and real-world objects that are mostly planar. As edges play the key role in 2D registration, depth discontinuities are crucial in 3D. In this paper, we investigate and develop a family of pose estimation algorithms that better exploit this boundary information. In addition to oriented surface points, we use two other primitives: boundary points with directions and boundary line segments. Our experiments show that these carefully chosen primitives encode more information compactly and thereby provide higher accuracy for a wide class of industrial parts and enable faster computation. We demonstrate a practical robotic bin-picking system using the proposed algorithm and a 3D sensor.**

I. Introduction

In robotics, pose estimation refers to the estimation of 6-degree-of-freedom (6-DoF) object pose using sensor measurements (e.g., images, 3D point clouds) and prior knowledge (e.g., a 3D model) of the object. Pose estimation plays a major role in many robotics applications such as bin-picking, localization, and 3D reconstruction.

2D Images: Until recently, pose estimation was primarily done using 2D images because cameras are cost effective and allow fast image acquisition. The main problem is to match the 2D features with their corresponding 3D features in the model. This becomes challenging due to changes in illumination, rotation, scale and partial viewpoint changes in the image space. Furthermore, some views of the object can theoretically lead to ambiguous poses. In order to handle these challenges, several invariant feature descriptors [1], [2] were used to find the correspondences between an input image and a database of images, where the keypoints are matched with the 3D coordinates and stored offline [3], [4].

Depth Edges: Most industrial parts are textureless and one has to rely heavily on the edges in the images. When boundaries of an object are used, a set of edge templates of an object is often known a priori, and the templates are searched in query image edge maps. Following the seminal paper on chamfer distance [5], there were several useful variants that incorporate edge orientation [6], [7] or employ hierarchical representation [8]. Intensity-based edge detection often gives too many edge pixels where only a few of them are useful edges coming from depth discontinuities. Raskar et al. [9] introduced multi-flash camera (MFC) to directly estimate depth edges by casting shadows from multiple flashes. Depth edges from MFC were successfully used in many pose estimation algorithms [10], [11].

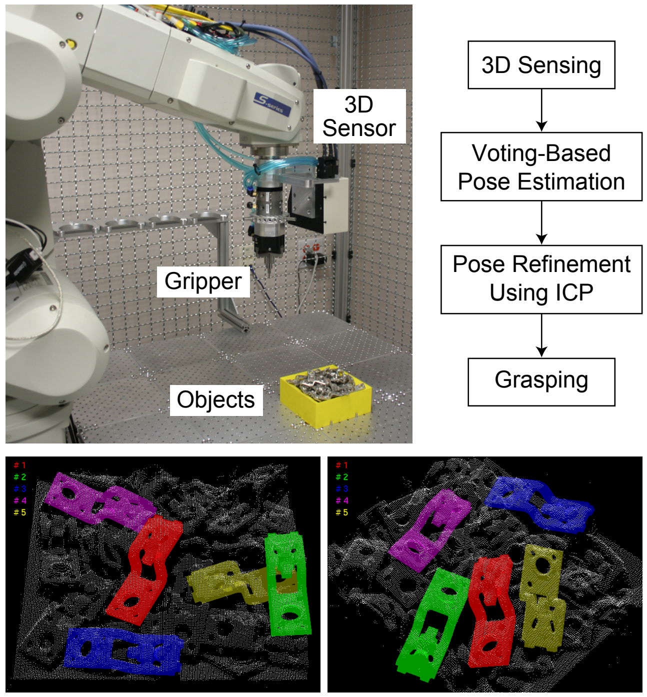


Fig. 1. **System overview.** (Top-left) Setup of our bin-picking system. The system uses a 3D sensor attached on a robot arm to grasp an object randomly placed in a bin. (Top-right) Algorithm flowchart. (Bottom) Pose estimation results. Best five pose estimates are superimposed on the scanned 3D point cloud. Note that the scenario exhibits a lot of challenges such as noise, missing data, clutter, and occlusions.

3D Data: As 3D sensors are becoming more and more cost effective on the commercial front, researchers are motivated to develop robust and faster algorithms for 3D sensors. Pose estimation would involve the correspondences between 3D features in the data and the model. In contrast to 2D images, 3D data are largely invariant to the geometric and photometric changes described above. The main challenge is to solve the correspondence problem in the presence of sensor noise, occlusions, and clutter. The size, distributions of surface normals and boundaries of an object are critical in registering the sensor data with the model. Several feature descriptors and matching algorithms have been proposed [12],

[13], [14], [15], [16]. These descriptors are invariant to rigid body transformation, but sensitive to noise and occlusion. Furthermore, they require dense point clouds, which may not be available.

RANSAC, Clustering, and Voting: Pose estimation is feasible with various kinds of correspondences between 3D sensor data and the model: 3 point correspondences [17], 2 line correspondences [18], and 6 points to 3 or more planes [19]. Typically these correspondences are used in a hypothesize-and-test framework such as RANSAC to compute the pose. Alternatively, the pose can be retrieved from the mode of the hypothesized pose distribution either using a Hough voting scheme [20] or clustering [21] in the parameter space.

These approaches suffer from two problems when only 3D sensor data are available without images or other priors: (1) *geometric primitives* such as points, lines, and planes are not very discriminative individually and are combinatorial to match; (2) it is difficult to achieve computational speed without doing any prior computations on the model. In this paper, we consider geometric primitives in a pair, which we refer to as *pair feature* [22], [23], [24]. The pair feature is more discriminative than individual primitives, which reduces the complexity of the matching task.

In [25], depth differences between pairs of points in range data are used in a classification framework for human pose estimation. In [23], a pair feature is defined by using surface points with their normals as primitives. An object is represented by a set of pair features in a hash table for fast retrieval. Random points are sampled from the sensor data and each pair votes for a particular pose. The required pose corresponds to the one getting the largest number of votes. Our paper can be seen as a generalization of this method for a larger class of objects using additional primitives that better exploit the boundary information in 3D data.

Note that the above algorithms detect objects in 3D data and provide their *coarse* poses. The coarse poses can be further refined using an iterative-closest point (ICP) algorithm [26].

Contributions: Surface points with normals are good to register objects that have rich variations in surface normals. However, they are not very efficient in representing many industrial and real-world objects that are mostly planar. To this end, we propose several novel pair features that exploit the depth discontinuities in 3D data. The main contribution of this paper is a comprehensive study and development of highly informative and compact features to model a 3D object by using its surface normals, boundaries, and their geometric relationships. We exploit a voting framework to efficiently compute poses with the compact features and apply the pose estimation algorithm to a practical bin-picking system using a 3D sensor.

II. System Overview

Fig. 1 (top-left) shows the setup of our bin-picking system. Our system uses a 3D sensor attached on a 6-axis industrial robot arm to estimate the poses of objects randomly placed in a bin. The 3D sensor is based on structured light using an infrared laser and provides 3D data as depth maps of 640×480 pixels. The 3D sensor is calibrated with respect to the robot arm, thereby allowing grasping and picking of an object using the estimated pose.

Fig. 1 (top-right) shows the algorithm flowchart. Our system scans the bin of objects using the 3D sensor. Given a 3D CAD model of a target object, our voting-based algorithm (described in Section III) performs detection and pose estimation of the target object using the scanned 3D point cloud. This provides multiple coarse pose hypotheses. The system selects several top pose hypotheses and individually refines them using a variant of ICP algorithm [26]. The refinement algorithm renders the CAD model using the current pose estimate and generates 3D points for the model by sampling the surface of the rendered model. It then computes the closest 3D point in the scanned point cloud for each 3D point in the model and updates the pose estimate using the 3D point correspondences.

After refinement, the registration error is given by the average distance between the corresponding scene and model points. The registration error could be high when the coarse pose computed by the voting algorithm is incorrect, or when a part of the object is missing due to occlusion from other objects. If the registration error is small and the estimated pose is safely reachable by the robot arm, the system grasps the object.

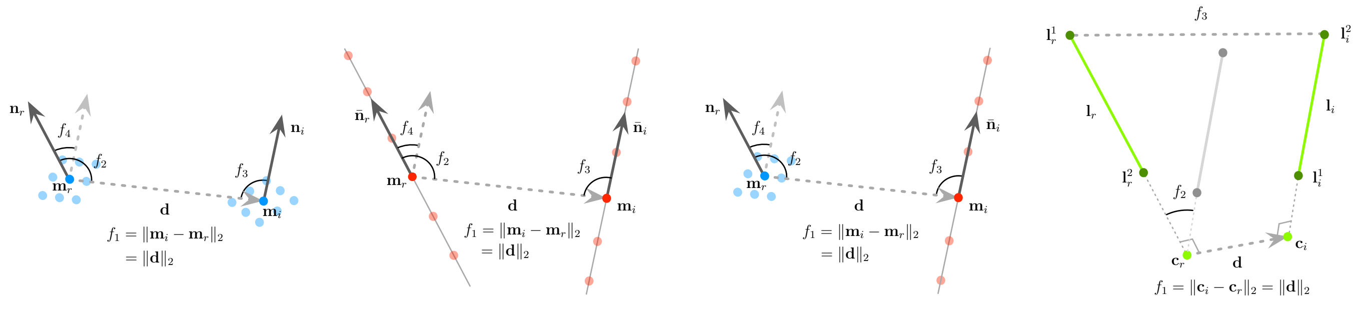
Please watch the accompanying video to see our bin-picking system in action.

III. Voting-Based Pose Estimation

We use oriented points (points with orientations) and line segments as our geometric primitives. We denote a pair feature based on a pair of oriented points on the object surface [23] as *S2S*. We propose three novel pair features: (1) a pair of oriented points on the object boundary (*B2B*), (2) a combination of an oriented point on the object surface with an oriented point on the object boundary (*S2B*), and (3) a pair of line segments on the object boundary (*L2L*). Note that the pair features are defined asymmetrically and we denote the first primitive in the pair as *reference* and the second as *referred*. To retrieve the pose of the object, it is necessary to establish correspondences between pair features from the scene and the model. We use various geometric constraints from the pair of oriented points or line segments as their descriptor. The correspondences between the scene and model pair features are then established by matching their descriptors.

A. Pair Features

1) S2S — Surface-to-Surface: Drost et al. [23] defined a pair feature using two points on the object surface and their normals. Given an oriented point from the 3D scene and a corresponding primitive from the object model, the 3D pose can be recovered up to a planar rotation by aligning point locations and their normals. To resolve the rotation ambiguity and recover the full 6-DoF pose, at least one correspondence



(a) S2S **(b)** B2B **(c)** S2B **(d)** L2L

Fig. 2. **Pair features for voting-based pose estimation. (a-c)** Point pair feature descriptors, $\mathbf{F}_{\mathsf{S2S}}, \mathbf{F}_{\mathsf{B2B}}$, and $\mathbf{F}_{\mathsf{S2B}}$, are defined by the relative position f_1 and orientations f_2, f_3, and f_4 of a pair of oriented points $(\mathbf{m}, \mathbf{n})$ where blue points indicate surface points with surface normal vectors and red points denote boundary points with directions. **(d)** The line pair feature descriptor $\mathbf{F}_{\mathsf{L2L}}$ is defined by the distance f_1 and the acute angle f_2 between two (infinite) lines, and the maximum distance between the two line segments f_3.

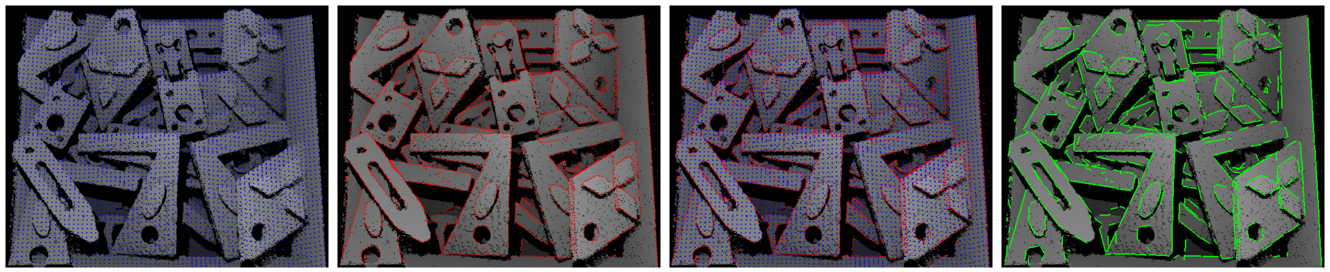


Fig. 3. **Geometric Primitives $\mathcal{M}$ for the pair features.** From left to right: Surface points and normals for *S2S*, boundary points and directions for *B2B*, their combination for *S2B*, and 3D boundary line segments for *L2L*. While surface points are obtained by subsampling the original scan, 3D boundary line segments are estimated using a RANSAC-based algorithm, and the boundary points are then obtained by subsampling along the line segments.

between two pairs of scene and model primitives is necessary. Let $\{(\mathbf{m}_r, \mathbf{n}_r), (\mathbf{m}_i, \mathbf{n}_i)\}$ denote the pair feature where $\mathbf{m}_r$ and $\mathbf{m}_i$ are the reference and referred points on the object surface, and $\mathbf{n}_r$ and $\mathbf{n}_i$ are their normals respectively. The associated descriptor with the *S2S* pair feature was given by

$$\mathbf{F}_{\mathsf{S2S}} = (f_1, f_2, f_3, f_4)^{\mathsf{T}} \tag{1}$$
$$= (\|\mathbf{d}\|_2, \angle(\mathbf{n}_r, \mathbf{d}), \angle(\mathbf{n}_i, \mathbf{d}), \angle(\mathbf{n}_r, \mathbf{n}_i))^{\mathsf{T}}, \tag{2}$$

where $\mathbf{d}$ is the vector from the reference point to the referred point, and $\angle(\mathbf{v}_1, \mathbf{v}_2) \in [0; \pi)$ represents the angle between two vectors. The first component of the descriptor, $f_1 = \|\mathbf{m}_i - \mathbf{m}_r\|_2 = \|\mathbf{d}\|_2$, represents the Euclidean distance between the two surface points. The second and third components, f_2 and f_3, are angles between the vector $\mathbf{d}$ and the surface normal vectors $\mathbf{n}_r$ and $\mathbf{n}_i$, respectively. The last component f_4 is the angle between the two normal vectors. The *S2S* feature is illustrated in Fig. 2(a).

When the object spans a wide range of surface normals, the *S2S* feature provides a good description of the object. However, it fails for shapes that do not span a rich set of surface normals. Unfortunately, most industrial parts are planar and have a very small set of normal directions. Additionally, due to noise in 3D data, it is difficult to estimate the normals accurately in high curvature regions on the surface.

2) B2B — Boundary-to-Boundary: We define *B2B*, a new point pair feature based on two points on the object boundary (depth edges). In contrast to surface points, boundary points do not have well defined normals. Therefore, we fit line segments to boundary points and use their directions as orientations.

We use a 3D extension of the 2D line fitting approach presented in [11]. First we compute the edges in range images using the Canny edge detector [27]. Points from the edge map are randomly sampled and 3D lines are fit locally using RANSAC. By iteratively finding and removing line segments with maximum inliers, we recover all line segments. These line segments are further refined using least squares.

After line fitting, we uniformly sample boundary points on the 3D line segments. In Fig. 2(b), the red points show the boundary points on two 3D line segments. We define *B2B* feature descriptor $\mathbf{F}_{\mathsf{B2B}} \in \mathbb{R}^4$ as

$$\mathbf{F}_{\mathsf{B2B}} = (f_1, f_2, f_3, f_4)^{\mathsf{T}} \tag{3}$$
$$= (\|\mathbf{d}\|_2, \angle(\bar{\mathbf{n}}_r, \mathbf{d}), \angle(\bar{\mathbf{n}}_i, \mathbf{d}), \angle(\bar{\mathbf{n}}_r, \bar{\mathbf{n}}_i))^{\mathsf{T}}. \tag{4}$$

This descriptor is equivalent to $\mathbf{F}_{\mathsf{S2S}}$ except that $\bar{\mathbf{n}}_r$ and $\bar{\mathbf{n}}_i$ are directions of the 3D lines. Note that the directions are not uniquely determined, therefore we consider two possible directions, $\bar{\mathbf{n}}$ and $-\bar{\mathbf{n}}$, when we use the *B2B* feature.

Object boundaries are highly informative. Compared to *S2S*, *B2B* provides more concise modeling since there are fewer boundary points than surface points. Additionally, the orientations from local line segments are more robust to noise compared to surface normals.

3) S2B — Surface-to-Boundary: A pair feature only based on boundary points is not very reliable for objects with high curvature. For example, any point on the surface of a spherical object can potentially become a depth edge based on the pose, whereas depth edges on a polyhedral object are more stable and always appear on plane intersections. To jointly and efficiently model both planar and curved objects, we propose *S2B*, a heterogeneous pair feature using an oriented surface point and an oriented boundary point. As shown in Fig. 2(c), we define *S2B* feature descriptor $\mathbf{F}_{\mathsf{S2B}} \in \mathbb{R}^4$ as

$$\begin{aligned}\mathbf{F}_{\mathsf{S2B}} &= (f_1, f_2, f_3, f_4)^{\mathsf{T}} &(5)\\ &= (\|\mathbf{d}\|_2, \angle(\mathbf{n}_r, \mathbf{d}), \angle(\bar{\mathbf{n}}_i, \mathbf{d}), \angle(\mathbf{n}_r, \bar{\mathbf{n}}_i))^{\mathsf{T}}. &(6)\end{aligned}$$

4) L2L — Line-to-Line: We propose *L2L*, a pair feature using two 3D line segments. This pair feature is particularly efficient for polyhedral objects and objects having long boundary line segments, since the number of line segments is fewer than that of surface points or boundary points. Let $\mathbf{c}_r$ and $\mathbf{c}_i$ be the closest points on the infinite lines that contain the 3D line segments, and $\{\mathbf{l}_r^1, \mathbf{l}_r^2, \mathbf{l}_i^1, \mathbf{l}_i^2\}$ denote the end points of line segments, as shown in Fig. 2(d). We define *L2L* feature descriptor $\mathbf{F}_{\mathsf{L2L}} \in \mathbb{R}^3$ as

$$\begin{aligned}\mathbf{F}_{\mathsf{L2L}} &= (f_1, f_2, f_3)^{\mathsf{T}} &(7)\\ &= (\|\mathbf{c}_i - \mathbf{c}_r\|_2, \angle_a(\mathbf{l}_r^2 - \mathbf{l}_r^1, \mathbf{l}_i^2 - \mathbf{l}_i^1), d_{\max})^{\mathsf{T}}, &(8)\end{aligned}$$

where $\angle_a(\mathbf{v}_1, \mathbf{v}_2) \in [0; \frac{\pi}{2}]$ represents the acute angle between two vectors, and

$$d_{\max} = \max(\|\mathbf{l}_i^1 - \mathbf{l}_r^1\|_2, \|\mathbf{l}_i^1 - \mathbf{l}_r^2\|_2, \|\mathbf{l}_i^2 - \mathbf{l}_r^1\|_2, \|\mathbf{l}_i^2 - \mathbf{l}_r^2\|_2).$$

The first and second components are the distance and angle between the two infinite lines, while the last component represents the maximum distance between the two line segments. The maximum distance $d_{\max}$ is computed by finding the maximum of the all possible distances between an end point in one line segment and an end point in the other. Using $d_{\max}$ is helpful to prune false matches between two line segments having similar distance and angle (e.g., any pair of coplanar orthogonal lines have the same distance and angle). However, line segments usually break into several fragments during the line fitting procedure due to sensor noise, occlusion, etc. As a result, the end points of line segments are usually unstable. Thus we use a bigger quantization step for this component of the descriptor. Note that we discard pairs of parallel lines since the closest points cannot be uniquely determined.

Recently, it has been shown that the minimum and maximum distances between line segments are very effective in pruning the search space in correspondence problems [28]. Note that we can also use the minimum distance between two line segments in building the *L2L* feature. However, this was sensitive due to breaking of line segments in our experiments.

B. Object Representation

As shown in [23], we globally model an object using a set of all possible pair features computed from the object model. Once this set is determined, we calculate pair features in the scene point cloud and match them with the set of the model pair features.

The pair feature representation of a target object is constructed offline. We first obtain geometric primitives $\mathcal{M}$: surface points for *S2S*, boundary points for *B2B*, both surface and boundary points for *S2B*, and 3D lines for *L2L*. These primitives can be calculated from either 3D scanned data with known calibration between the sensor and the object, or synthetic depth data rendered from a known CAD model. With these primitives $\mathcal{M}$, all possible pair features, $(\mathbf{m}_r, \mathbf{m}_i) \in \mathcal{M}^2$ for *S2S*, *B2B*, or *S2B* and $(\mathbf{l}_r, \mathbf{l}_i) \in \mathcal{M}^2$ for *L2L*, are calculated.

For efficient feature matching, we store the set of pair features of the model in a hash table data structure $\mathcal{H}$, as in [23]. We quantize the pair feature descriptors and use them as the key for the hash table. Pair features that have similar descriptors are inserted together in the same bin and matching/voting can be done in constant time. Note that it is important to define the quantization levels appropriately; using very large step sizes reduces discriminative power of the descriptors, whereas using very small step sizes makes the algorithm sensitive to noise.

C. Voting Scheme for S2S, B2B, and S2B Features

After computing pair features and constructing a hash table structure, we find pose hypotheses by calculating rigid body transformations between a scene pair feature and a set of corresponding model pair features. To make this search efficient, we adopt a voting scheme. A naïve approach would require voting in the 6-DoF pose space, which is not computationally efficient. Instead, Drost et al. [23] proposed a voting scheme that reduces the voting space to a 2D space using *local coordinates*. First, a scene point pair $(\mathbf{s}_r, \mathbf{s}_i) \in \mathcal{S}^2$, where $\mathcal{S}$ is the set of primitives from the scene, is searched in the hash table $\mathcal{H}$, and a corresponding model point pair $(\mathbf{m}_r, \mathbf{m}_i) \in \mathcal{M}^2$ is found. Then, the reference points of the pairs, $\mathbf{s}_r$ and $\mathbf{m}_r$ are aligned in an intermediate coordinate system, as shown in Fig. 4 (left). To fully align the pair, the referred points, $\mathbf{s}_i$ and $\mathbf{m}_i$, should be aligned by rotating the object around the normal. After the planar rotation angle α is calculated, the local coordinates are defined by the pair of the reference model point and the planar rotation angle $(\mathbf{m}_r, \alpha)$. The transformation from $(\mathbf{m}_r, \mathbf{m}_i)$ to $(\mathbf{s}_r, \mathbf{s}_i)$ is given by

$$\mathbf{s}_i = \mathbf{T}_{s\to g}^{-1}\mathbf{R}_{\mathbf{x}}(\alpha)\mathbf{T}_{m\to g}\mathbf{m}_i, \qquad (9)$$

where $\mathbf{R}_{\mathbf{x}}(\alpha)$ is the rotation around the $\mathbf{x}$-axis with angle α, $\mathbf{T}_{s\to g}$ and $\mathbf{T}_{m\to g}$ are the transformations from the scene and model coordinate systems to the intermediate coordinate system, respectively.

In the voting phase, a given reference scene point $\mathbf{s}_r$ and every other point $\mathbf{s}_i$ are paired, and then the model pair features $(\mathbf{m}_r, \mathbf{m}_i)$ which are similar to the scene pair feature $(\mathbf{s}_r, \mathbf{s}_i)$ are searched in the hash table $\mathcal{H}$ using their descriptors. For every matching $(\mathbf{m}_r, \mathbf{m}_i)$, the rotation angle α is computed and then votes are cast in the 2D space of $(\mathbf{m}_r, \alpha)$. After all the matchings are voted, the elements that

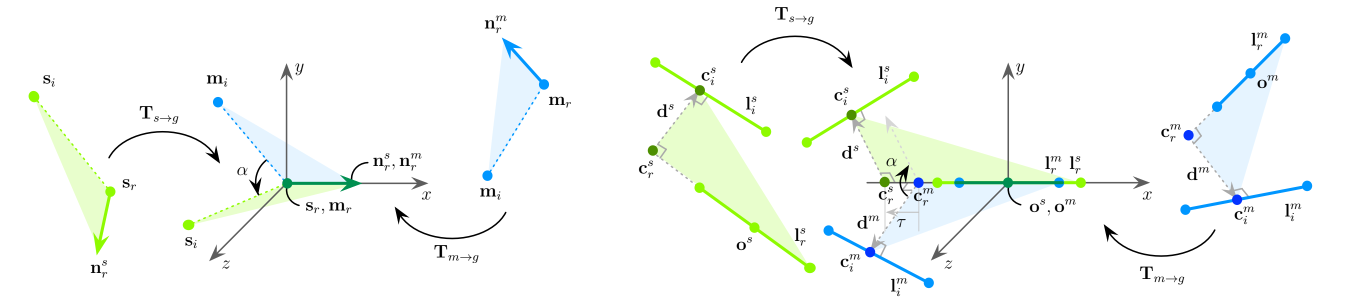


Fig. 4. **Aligning pair features via an intermediate coordinate system.** (Left) Transformation for the point pair features *S2S*, *B2B*, and *S2B*. By $\mathbf{T}_{s\to g}$, the scene reference point $\mathbf{s}_r$ is moved to the origin and its orientation (normal or direction) $\mathbf{n}_r^s$ is aligned to the $\mathbf{x}$-axis. The model reference point is similarly transformed by $\mathbf{T}_{m\to g}$, such that the positions and orientations of the reference points are aligned. The referred points $\mathbf{s}_i$ and $\mathbf{m}_i$ are then aligned by a rotation with angle α around the $\mathbf{x}$-axis. Thus a 2D space $(\mathbf{m}_r, \alpha)$ is used for voting. (Right) Transformation for the line pair feature *L2L*. By $\mathbf{T}_{s\to g}$, the scene reference line $\mathbf{l}_r^s$ is aligned to the $\mathbf{x}$-axis and its middle point $\mathbf{o}^s$ is moved to the origin. The model reference line is similarly transformed by $\mathbf{T}_{m\to g}$ such that the reference lines are aligned. The referred lines $\mathbf{l}_i^s$ and $\mathbf{l}_i^m$ are then aligned by a rotation with angle α and a translation with τ along the $\mathbf{x}$-axis. Thus a 3D space $(\mathbf{o}^m, \alpha, \tau)$ is used for voting.

have votes exceeding a threshold are selected as valid pose candidates, and then the transformation between the model and scene coordinate systems is computed using (9). This voting scheme is applicable to *S2S*, *B2B*, and *S2B*, since the point pair features are fundamentally equivalent. However, the *L2L* feature, defined by a pair of line segments, requires a specialized voting scheme.

D. Voting Scheme for L2L Feature

As described earlier, the end points of line segments are not stably determined. We therefore build a voting scheme for the *L2L* feature based on the infinite lines that contain the 3D line segments, which is robust to the fragmentation of line segments.

Similar to the point pair features, the voting scheme for the *L2L* feature is based on aligning two pair features in an intermediate coordinate system. As illustrated in Fig. 4 (right), the reference line $\mathbf{l}_r^s$ and the referred line $\mathbf{l}_i^s$ from the scene are transformed by $\mathbf{T}_{s\to g}$ in order to align $\mathbf{l}_r^s$ to the $\mathbf{x}$-axis and to align the middle point $\mathbf{o}^s$ to the origin. Similarly, $\mathbf{l}_r^m$ and $\mathbf{l}_i^m$ are transformed via $\mathbf{T}_{m\to g}$. Still there are two degrees of freedom to fully align the line pairs. As in the point pair features, the first one is the rotation around the $\mathbf{x}$-axis; this angle α is determined from the angle between $\mathbf{d}^s$ and $\mathbf{d}^m$. The other degree of freedom is the translation along the $\mathbf{x}$-axis; this corresponds to the displacement between the closest points $\mathbf{c}_r^m$ to $\mathbf{c}_r^s$, denoted as τ. Therefore, we use a 3D space $(\mathbf{o}^m, \alpha, \tau)$ for voting using the *L2L* feature. The transformation from $(\mathbf{l}_r^m, \mathbf{l}_i^m)$ to $(\mathbf{l}_r^s, \mathbf{l}_i^s)$ can be computed as

$$\mathbf{l}_i^s = \mathbf{T}_{s\to g}^{-1}\mathbf{T}_{\mathbf{x}}(\tau)\mathbf{R}_{\mathbf{x}}(\alpha)\mathbf{T}_{m\to g}\mathbf{l}_i^m, \tag{10}$$

where $\mathbf{T}_{\mathbf{x}}(\tau)$ is the translation along the $\mathbf{x}$-axis with τ.

E. Pose Clustering

In the voting scheme explained in the previous sections, raw pose hypotheses are obtained by thresholding in the voting space. Since an object is modeled by a large set of pair features, it is expected to have multiple pose hypotheses each for different reference primitives, points $\mathbf{m}_r$ or lines $\mathbf{l}_r^m$, supporting the same pose. Thus, it is required to aggregate similar poses from different reference primitives [23]. Although there are several methods for clustering in 3D rigid body transformation space $SE(3)$ such as mean shift on Lie groups [21], these methods are usually computationally prohibitive for time critical applications. Here we adopt an agglomerative clustering approach which is very efficient.

We first sort the raw pose hypotheses in decreasing order of the number of votes. From the highest vote, we create a new cluster. If the next pose hypothesis is close to one of the existing clusters, the hypothesis is added to the cluster and the cluster center is updated as the average of the pose hypotheses within the cluster. If the next hypothesis is not close to any of the clusters, it creates a new cluster. The proximity testing is done with fixed thresholds in translation and rotation. Distance computation and averaging for translation are performed in the 3D Euclidean space, while those for rotation are performed using quaternion representation. After clustering, the clusters are sorted in decreasing order of the total number of votes which determines confidence of the estimated poses.

IV. Experimental Results

In this section, we present an extensive evaluation of the proposed methods on synthetic and real data. We also evaluate the performance of our bin-picking system described in Section II.

A. Synthetic Data

To compare the performance of the four pair features, we generated 500 synthetic scenes in which six objects (Fig. 5) were drawn with randomly selected poses. We ensured that these random poses do not lead to physically infeasible overlapping objects by checking the intersection of their bounding boxes. We rendered the scenes with OpenGL by setting the parameters of the rendering camera based on the calibration parameters of our 3D sensor. For every object the correct pose is stored in a ground truth database for

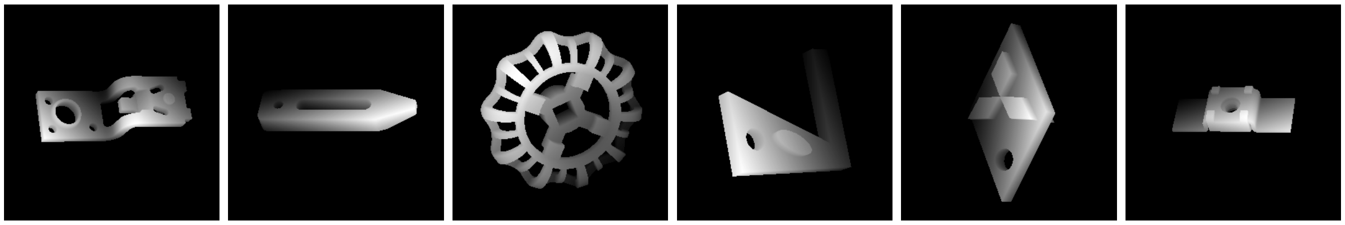


Fig. 5. **Test Objects.** The 3D CAD models of these test objects are used to create the model pair features for the voting algorithm and to generate synthetic dataset. From left to right: **Circuit Breaker**, **Clamp**, **Wheel**, **Γ-Shaped**, **Logo**, and **Weld Nuts**.

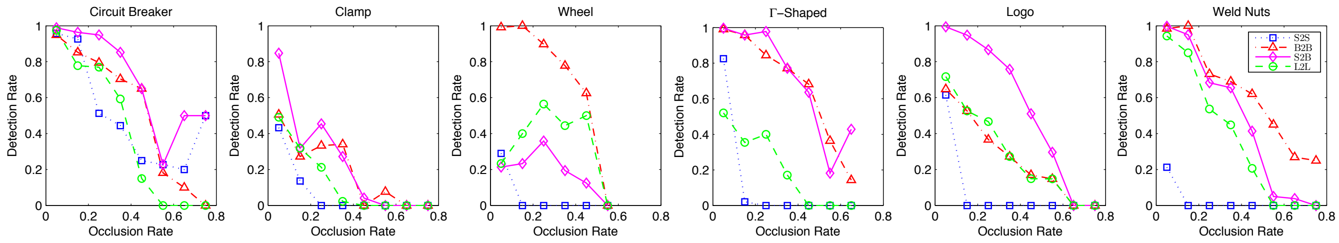


Fig. 6. **Detection rates against occlusion rates for the synthetic dataset.** Performance of the four methods decreases as occlusion rate increases. Although the performance depends on objects, *B2B* and *S2B* features generally outperform the other features.

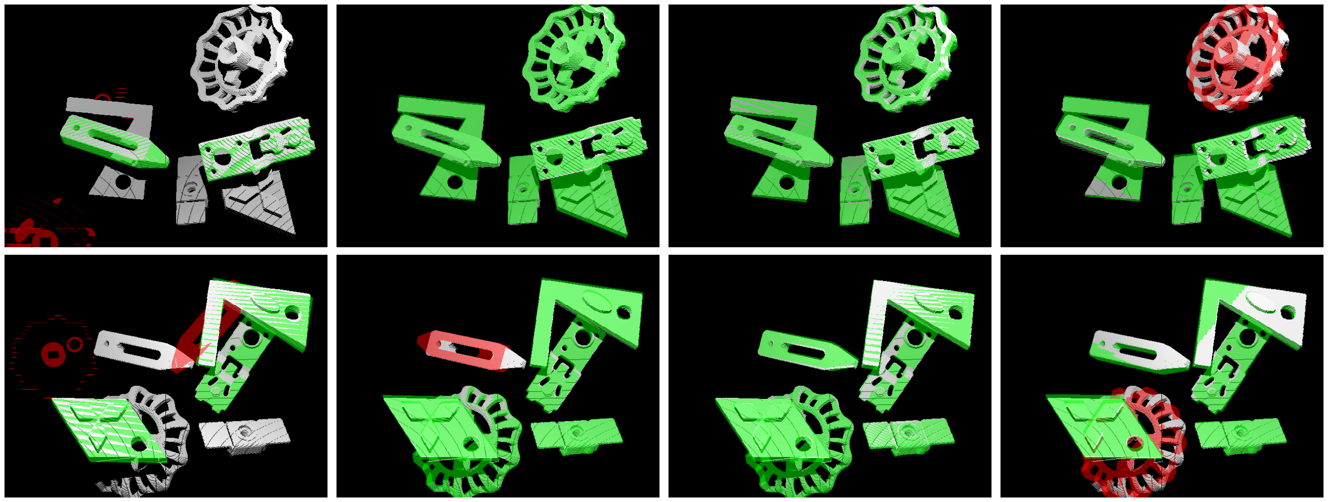


Fig. 7. **Two example scenes from the** 500 **synthetic scenes.** From left to right: Results using *S2S*, *B2B*, *S2B*, and *L2L* features. Correct and incorrect poses are depicted as green and red renderings respectively.

experimental validation. Note that we identify and account for the object symmetries during our experiments.

As shown in Fig. 7, objects in the synthetic scene severely occlude each other and the degree of occlusion is various over the 500 test scenes. We quantify the occlusion rate and study the detection performance for different occlusion rates. We follow the occlusion definition of [14]:

$$\text{occlusion} = 1 - \frac{\text{model surface area in the scene}}{\text{total model surface area}}. \quad (11)$$

We performed the voting-based pose estimation using each pair feature and considered only the pose that got the maximum number of votes. The estimated pose was then compared with the ground truth. If the errors in translation and rotation were within 5 mm and 5°, we counted it as a true positive; otherwise it was regarded as a false positive.

Fig. 6 shows the detection rate at different occlusion rates for each of the six objects. For Wheel and Weld Nuts objects, the *B2B* feature outperforms the other pair features, while the *S2B* feature shows better results for other objects. Since each object possesses different geometric characteristics, the performance of the four pair features on different objects slightly varies; nevertheless, our boundary-based pair features (*B2B*, *S2B*, and *L2L*) show better performance than the *S2S* feature. The reason why the *S2S* feature reports inferior results is that pairs of surface points in the same planar region of the object can correspond to any planar region in the scene. As shown in the leftmost column of Fig. 7, planar surfaces of several objects are fitted to the background plane in the scene.

The boundary-based pair features are not only more discriminative, but also more efficient. Table I shows average numbers of pair features in the synthetic scenes and relative processing times where the time of the fastest method, *B2B*, is shown as one. Voting using the *S2S* feature requires a much larger number of pair features than voting using boundary-

TABLE I

AVERAGE NUMBERS OF PAIR FEATURES IN THE SYNTHETIC SCENE DATASET AND RELATIVE PROCESSING TIME.

Feature	Number of Features	Relative Proc. Time[†]
S2S [23]	23040000 ($= 4800 \times 4800$)	3.21
B2B	2616953 ($\approx 1618 \times 1618$)	**1.00**
S2B	7689280 ($\approx 4800 \times 1602$)	1.20
L2L	121058 ($\approx 348 \times 348$)	1.03

[†] The fastest method, *B2B*, is shown as one.

based pair features. Although the number of the *L2L* features is the smallest, average processing time per a line pair takes more because of the higher-dimensional voting space and more complex transformation via the intermediate coordinate system.

B. Real Data

We tested the voting-based pose estimation for real 3D data scanned with our 3D sensor. The ground truth poses of the objects are manually identified. Fig. 8 shows results for each of the four pair features. The scene on the upper row contains multiple instances of four of our test objects. The objects occlude each other and make the scene highly cluttered. The displayed pose corresponds to the best pose hypothesis computed for each of the four objects. In the result of using the *S2S* feature, two estimated poses are false positives. Similar to the results for synthetic data, the planar area of Clamp object caused several false pose estimates. As shown in the lower row, we also tested the four pair features in the scene which has multiple instances of Circuit Breaker object. For comparison, we rendered top six pose hypotheses obtained using each pair feature. Although in general all pair features provide good performance for this object as shown in the synthetic experiments, the *L2L* feature has three false positives in this case, which are the flipped poses of the ground truth poses. These poses have high similarities except the small differences inside the object. The *L2L* feature is not very robust to such small differences, since the directions of line segments become unstable for short line segments.

C. Bin-Picking System Performance

Pose Estimation Accuracy: To quantitatively estimate the accuracy of our bin-picking system, we used a single Circuit Breaker object placed on a plane. We scanned it from different locations by moving the robot arm, estimated at each location the pose of the object in the robot coordinate system, and computed pose estimation errors as absolute differences from their median. We selected 100 random sensor locations such that the object is centered in the field of view of the 3D sensor. The locations were within 20° from the $\mathbf{z}$-axis of the robot coordinate system with a distance to the object of 330 mm. For pose estimation, we used the voting algorithm with the *L2L* feature followed by ICP-based pose refinement.

Fig. 9 shows the histograms of pose estimation errors. Table II describes their average for each translation and rotation around each axis. They demonstrate the consistent pose estimation results of our system with average absolute errors of less than 0.3 mm for all (X, Y, Z) translations and less than 0.3° for all (A, B, C) rotations.

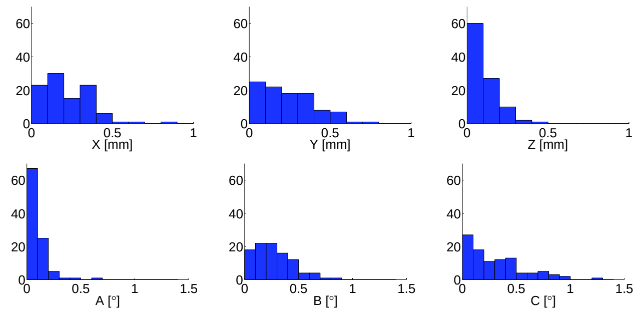


Fig. 9. **Histograms of pose estimation errors** for each translation (X, Y, Z) and rotation around each axis (A, B, C). The errors are computed as absolute differences from their median.

TABLE II

AVERAGE ABSOLUTE POSE ESTIMATION ERRORS.

X [mm]	Y [mm]	Z [mm]	A [°]	B [°]	C [°]
0.22	0.24	0.09	0.09	0.27	0.30

TABLE III

PICKUP SUCCESS RATE.

Total Trial	Success	Failure	Success Rate
344	338	6	98.3%

Pickup Success Rate: We measured the pickup success rate of our system by placing 36 Circuit Breaker objects randomly in a bin as shown in Fig. 1. We used the *B2B* feature and used 20% of the total number of boundary points for voting (each 3D scan included $\sim$ 3000 boundary points). The system performed ICP-based pose refinement for the best 5 poses computed with the voting algorithm, and picked up a single object as described in Section II for each cycle. We refilled the bin when the system detected no pickable objects or the system continuously picked up a predefined number (15) of objects. The system picked up 10.2 objects on average in a continuous process.

As shown in Table III, our system achieved a success rate of more than 98% over 344 trials. All 6 failures were due to occlusion of the gripping location of the object. The estimated object poses were correct even in these failure cases.

Processing Time: In the above bin-picking experiments, the voting-based pose estimation algorithm using the *B2B* feature (including 3D line fitting, voting, and pose clustering) took around 500 msec. The refinement process using ICP required around 100 msec for each pose hypothesis. The system was implemented on an Intel Core i7-2600 PC with C++. As shown in the accompanying video, the entire pose estimation process can be performed during robot motion, avoiding the wait time for the robot.

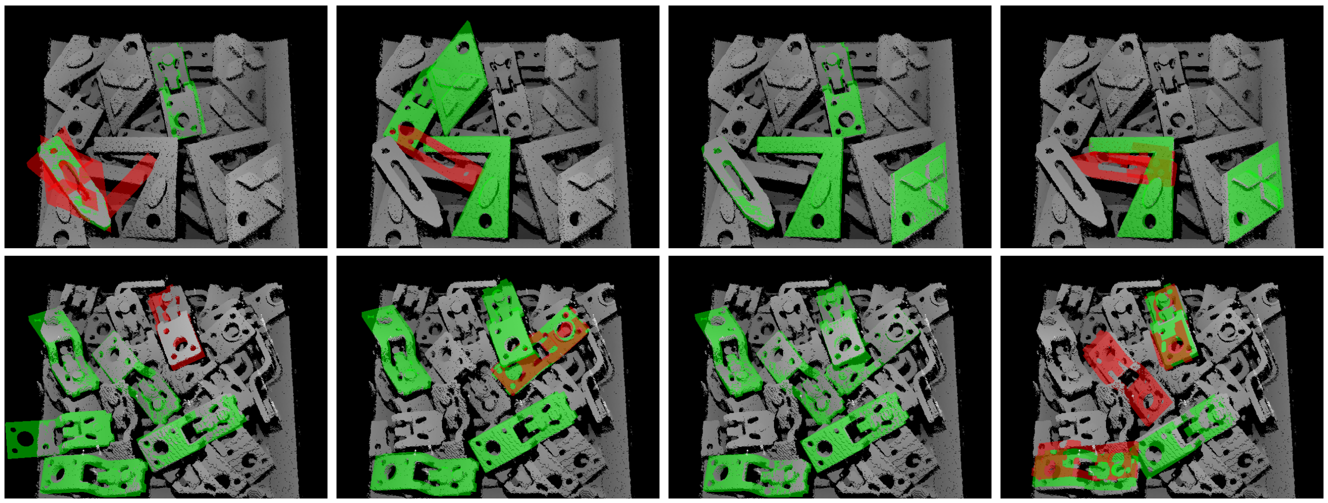


Fig. 8. **Two example scenes from the real scans.** From left to right: Results using *S2S*, *B2B*, *S2B*, and *L2L* features. The scene on the upper row includes multiple instances of our test objects. The scene on the lower row contains multiple instances of Circuit Breaker object. Our algorithm can reliably estimate poses of the object even when there are multiple objects and the scene is highly cluttered.

V. Conclusions

We developed a family of pair features using oriented surface points, oriented boundary points, and boundary line segments to model a wide variety of objects. We used the pair features in a voting framework for robust and efficient pose estimation. We showed that the pair features based on the object boundary are more compact and informative, thereby leading to higher accuracy and faster computation. We demonstrated a bin-picking system with pickup success rate of more than 98% and pose estimation error less than 0.3 mm and 0.3° for translations and rotations.

Acknowledgments: We thank Jay Thornton, Haruhisa Okuda, and Makito Seki for their helpful comments, and John Barnwell and Yasuo Kitaaki for their support in building the experimental setup. This work was done at MERL with support from MERL. Changhyun Choi and Ming-Yu Liu contributed to the work while they were at MERL.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *CVIU*, vol. 110, no. 3, pp. 346–359, 2008.
[3] I. Gordon and D. Lowe, "What and where: 3D object recognition with accurate pose," in *Toward Category-Level Object Recognition*, 2006, pp. 67–82.
[4] C. Choi and H. I. Christensen, "Robust 3D visual tracking using particle filtering on the SE(3) group," in *ICRA*, 2011.
[5] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *IJCAI*, vol. 2, 1977, pp. 659–663.
[6] C. Olson and D. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Trans. Image Proc.*, vol. 6, no. 1, pp. 103–113, 1997.
[7] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *CVPR*, 2010, pp. 1696–1703.
[8] D. M. Gavrila, "A bayesian, exemplar-based approach to hierarchical shape matching," *PAMI*, pp. 1408–1421, 2007.
[9] R. Raskar, K. Tan, R. Feris, J. Yu, and M. Turk, "Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging," *ACM Trans. Graphics*, vol. 23, pp. 679–688, 2004.
[10] A. Agrawal, S. Yu, J. Barnwell, and R. Raskar, "Vision-guided robot system for picking objects by casting shadows," *IJRR*, vol. 29, no. 2–3, pp. 155–173, 2010.
[11] 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, pp. 2028–2035.
[12] F. Stein and G. Medioni, "Structural indexing: Efficient 3-D object recognition," *PAMI*, vol. 14, no. 2, pp. 125–145, 1992.
[13] C. Dorai and A. Jain, "COSMOS-A representation scheme for 3D free-form objects," *PAMI*, vol. 19, no. 10, pp. 1115–1130, 1997.
[14] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *PAMI*, vol. 21, no. 5, pp. 433–449, May 1999.
[15] A. Mian, M. Bennamoun, and R. Owens, "On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes," *IJCV*, vol. 89, no. 2, pp. 348–361, 2010.
[16] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *ICRA*, 2009, pp. 3212–3217.
[17] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, 1987.
[18] Z. Zhang and O. Faugeras, "Determining motion from 3D line segment matches: A comparitive study," in *BMVC*, 1990.
[19] S. Ramalingam, Y. Taguchi, T. Marks, and O. Tuzel, "P2Π: A minimal solution for the registration of 3D points to 3D planes," in *ECCV*, 2010.
[20] D. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.
[21] O. Tuzel, R. Subbarao, and P. Meer, "Simultaneous multiple 3D motion estimation via mode finding on lie groups," in *ICCV*, 2005, pp. 18–25.
[22] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification," in *3DIM*, Oct. 2003, pp. 474–481.
[23] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *CVPR*, 2010.
[24] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *PAMI*, pp. 1584–1601, 2006.
[25] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, 2011.
[26] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *PAMI*, pp. 239–256, 1992.
[27] J. Canny, "A computational approach to edge detection," *PAMI*, vol. 8, no. 6, pp. 679–698, Nov. 1986.
[28] S. Ramalingam, S. Bouaziz, P. Sturm, and P. Torr, "The light-path less traveled," in *CVPR*, 2011.