3D Pose Estimation of Daily Objects Using an RGB-D Camera

Changhyun Choi and Henrik I. Christensen
Center for Robotics & Intelligent Machines
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
{cchoi,hic}@cc.gatech.edu

***Abstract*— In this paper, we present an object pose estimation algorithm exploiting both depth and color information. While many approaches assume that a target region is cleanly segmented from background, our approach does not rely on that assumption, and thus it can estimate pose of a target object in heavy clutter. Recently, an oriented point pair feature was introduced as a low dimensional description of object surfaces. The feature has been employed in a voting scheme to find a set of possible 3D rigid transformations between object model and test scene features. While several approaches using the pair features require an accurate 3D CAD model as training data, our approach only relies on several scanned views of a target object, and hence it is straightforward to learn new objects. In addition, we argue that exploiting color information significantly enhances the performance of the voting process in terms of both time and accuracy. To exploit the color information, we define a color point pair feature, which is employed in a voting scheme for more effective pose estimation. We show extensive quantitative results of comparative experiments between our approach and a state-of-the-art.**

I. Introduction

Object recognition and 6-DOF pose estimation are important tasks in robotic perception. For the last decade, stable keypoint descriptors [1], [2] have led to successful progress on object recognition. As these keypoint descriptors are invariant to changes in illumination and geometric transformation, keypoint correspondences over different images can be reliably determined. For robotic manipulation, 3D coordinates of keypoints are generally required as an object model so that full 6-DOF object pose can be recovered. These keypoint coordinates can be calculated via structure from motion [3] or back-projecting 2D keypoints to 3D CAD model [4].

The keypoint descriptors are suitable for textured objects, but a large number of daily objects still lack texture. For the less textured object, edge feature is preferred since it corresponds to the object boundaries. A common approach is that a set of edge image templates of an object is known *a priori*, and in testing phase the template images are matched with a given query edge image. In classic computer vision, the chamfer [5] and Hausdorff [6] distances were proposed as robust metric, and they were further enhanced by considering edge orientation [7], [8]. A common method to extract edge feature from an image is image gradient-based method, such as Canny edge detector [9]. However, this method often results in unnecessary edges coming from surface texture or non-Lambertian reflectance. To find useful edges from depth discontinuities, the multi-flash camera [10] was introduced to determine depth edges by casting shadows from multiple flashes and was successfully employed in several robotic pose estimation algorithms [11], [12].

Fig. 1. **Overview.** Our approach exploits both geometric shape and color information in point clouds for robust object pose estimation. The estimated pose of each object is depicted in color object point cloud. Since our approach does not hinge upon the planar segmentation, it can be applied in highly cluttered environment. (Best viewed in color)

As RGB-D sensors, which provide depth as well as color information in real-time, have recently been introduced at low cost, 3D information-based pose estimation can be more feasible than ever. Compared to 2D images, 3D data are more invariant to the geometric changes. The iterative-closest point (ICP) algorithm [13] is well-known for the registration of 3D point clouds, but it requires a good initial pose estimate. Rusu et al. [14] proposed the Viewpoint Feature Histogram (VFH) that encodes four angular distributions of surface normals on a segmented point cloud. As the VFH is not robust to occlusion and does not allow full pose estimation, the Clustered Viewpoint Feature Histogram (CVFH) was recently presented [15]. Lai et al. [16] proposed a tree structure for scalable object recognition and pose estimation, but the pose estimation is limited in that it can only estimate 1-DOF rotation of the object pose. Although these approaches can recognize object pose efficiently, they hinge upon perfect

segmentation from the background. All of these approaches are applicable for well structured table-top manipulation, but they are not robust for cluttered environments.

For general object pose estimation, it is required to match an object model with a scene directly. Like local image keypoints, several local invariant features have been proposed based on the distribution of surface normal around a point [17], surface curvature [18], spin image [19], and relative angles between neighboring normals [20]. While these features are invariant to rigid body transformation, they are sensitive to noise and resolution difference of point clouds.

Drost et al. [21] defined a pair feature using two points on surfaces and their normals. In the learning phase, a set of pair features from an object is calculated and saved in a hash table for fast retrieval. In the testing phase, points are randomly sampled from the sensor data, and each pair matched with pairs in the learned model votes for a pose hypothesis. After the voting process, a set of high votes over a certain confidence level are aggregated to form possible object pose hypotheses. The pair feature can be seen as a successor of the surflet pairs [22], and using a hash table for fast matching is also presented in [23]. This approach was recently enhanced by incorporating the visibility context [24] or considering object boundary information [25]. There are also several modified Hough transforms for 3D object pose estimation [26] using the SRT distance [27].

The surface point pair feature is well suited to recognize objects that have rich variations in surface normals. However, it is not very efficient in representing planar or self-symmetric objects because a lot of different point pairs fall into the same hash slot. Although this ambiguity could be solved via the voting process where different pose hypotheses are aggregated separately, this certainly degrades its efficiency. Moreover, when there are a large amount of background clutter in a test scene, a lot of the point pair features come from the clutter. If surface shapes of our object model and the clutter are similar each other, it is highly likely to have false feature matches and consequently results in false pose estimates. As such, we need to prune unnecessary feature matching for more efficient and accurate pose estimation. We exploit the RGB color information to prune potentially false matches based on the color similarity. To be more robust to illumination changes, the HSV (Hue, Saturation, and Value) color space is considered. By using these additional dimensions, the casted votes in an accumulator space are more likely to contribute to true pose hypotheses. Furthermore, the voting process is more efficient since unnecessary votes are skipped. These arguments are verified in the following experimental section.

II. RGB-D Pose Estimation

In this section, our pose estimation algorithm is presented. Staring by explaining the point pair features of [21], we introduce our new color point pair feature. After explaining how the new feature is employed during object learning, the voting scheme will be described.

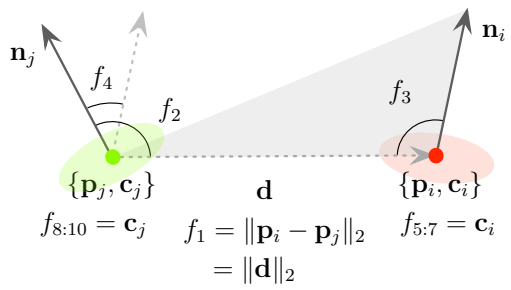

Fig. 2. **Color Point Pair Feature (CPPF).** The feature is defined by the relative position (f_1), relative orientations (f_2, f_3, and f_4), and colors ($f_{5:7}$ and $f_{8:10}$) of the pair of color oriented points $\{(\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i), (\mathbf{p}_j, \mathbf{n}_j, \mathbf{c}_j)\}$.

A. Point Pair Features

A pair feature of two oriented points has been employed in shape-based object recognition [22], [21]. Let $\{(\mathbf{p}_i, \mathbf{n}_i), (\mathbf{p}_j, \mathbf{n}_j)\}$ denote the pair feature where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the *reference* and *referred* points on the object surface, and $\mathbf{n}_r$ and $\mathbf{n}_i$ are their normals respectively. The point pair feature $\mathbf{F}_{\mathsf{PPF}} \in \mathbb{R}^4$ is defined by

$$\mathbf{F}_{\mathsf{PPF}} = \mathtt{PPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j) \tag{1}$$

$$= \begin{pmatrix} \|\mathbf{d}\|_2 \\ \angle(\mathbf{n}_i, \mathbf{d}) \\ \angle(\mathbf{n}_j, \mathbf{d}) \\ \angle(\mathbf{n}_i, \mathbf{n}_j) \end{pmatrix} \tag{2}$$

where $\mathbf{d} = \mathbf{p}_i - \mathbf{p}_j$, and $\angle(\mathbf{v}_1, \mathbf{v}_2) \in [0; \pi)$ represents the angle between two vectors. The first dimension, $\|\mathbf{d}\|_2 = \|\mathbf{p}_i - \mathbf{p}_j\|_2$, represents the Euclidean distance between the two surface points. The second and third components are angles between the vector $\mathbf{d}$ and the surface normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$, respectively. The last component is the angle between the two normal vectors. This feature effectively encodes geometric constraints of point cloud surfaces so that efficient matching between model and scene point clouds is possible, especially when these point clouds contain rich surface normal variations.

B. Color Point Pair Features

Even though the PPF might be suitable for objects having rich variations in surface normals, it is generally not discriminative enough to describe planar or self-symmetric objects. Hence it is required to augment the pair feature so that the feature will be more effective to these types of objects. The color point pair feature $\mathbf{F}_{\mathsf{CPPF}} \in \mathbb{R}^{10}$ is defined by concatenating two 3D color vectors of the points:

$$\mathbf{F}_{\mathsf{CPPF}} = \mathtt{CPPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j, \mathbf{c}_i, \mathbf{c}_j) \tag{3}$$

$$= \begin{pmatrix} \mathtt{PPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j) \\ \mathbf{c}_i \\ \mathbf{c}_j \end{pmatrix} \tag{4}$$

Algorithm 1: BuildHashTable($\mathcal{M}$)

Data: $\mathcal{M} = \{(\mathbf{p}_1^m, \mathbf{n}_1^m, \mathbf{c}_1^m), \cdots, (\mathbf{p}_{N_m}^m, \mathbf{n}_{N_m}^m, \mathbf{c}_{N_m}^m)\}$
Result: $\mathcal{H}$
Params: $\delta, \theta, \boldsymbol{\sigma}$

$\mathcal{H} \leftarrow \{\phi\}$
for $i \leftarrow 1$ **to** N_m **do**
  for $j \leftarrow 1$ **to** N_m **do**
    if $i \neq j$ **then**
      $\mathbf{F} \leftarrow \texttt{CPPF}(\mathbf{p}_i^m, \mathbf{p}_j^m, \mathbf{n}_i^m, \mathbf{n}_j^m, \mathbf{c}_i^m, \mathbf{c}_j^m)$ (4)
      $\mathcal{I} \leftarrow \texttt{HashIndex}(\mathbf{F}, \delta, \theta, \boldsymbol{\sigma})$ (6)
      $\alpha_m \leftarrow \texttt{PlanarRotAngle}(\mathbf{n}_i, \mathbf{p}_i^m, \mathbf{p}_j^m)$ (§II-D)
      $\mathcal{H} \leftarrow \texttt{HashInsert}(\mathcal{H}, \mathcal{I}, \{\alpha_m, i\})$

Algorithm 2: RGB-D Pose Estimation

Data: $\mathcal{H}, \mathcal{M}, \mathcal{S} = \{(\mathbf{p}_1^s, \mathbf{n}_1^s, \mathbf{c}_1^s), \cdots, (\mathbf{p}_{N_s}^s, \mathbf{n}_{N_s}^s, \mathbf{c}_{N_s}^s)\}$
Result: $\mathcal{P} = \{\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_{N_p}\}$
Params: $N_p, \gamma_s, \tau_v, \delta, \theta, \boldsymbol{\sigma}$

if $\mathcal{H} = \{\phi\}$ **then**
  $\mathcal{H} \leftarrow \texttt{BuildHashTable}(\mathcal{M}, \delta, \theta, \boldsymbol{\sigma})$ ⟨1⟩
$\mathbf{A} \leftarrow \mathbf{0}_{N_m \times \lfloor \frac{\pi}{\theta} \rfloor}$
$\mathcal{P} \leftarrow \{\phi\}$
for $h \leftarrow 1$ **to** $\lfloor N_s \cdot \gamma_s \rfloor$ **do**
  $i \leftarrow \texttt{RandomSample}(N_s)$
  for $j \leftarrow 1$ **to** N_s **do**
    if $i \neq j$ **then**
      $\mathbf{F} \leftarrow \texttt{CPPF}(\mathbf{p}_i^s, \mathbf{p}_j^s, \mathbf{n}_i^s, \mathbf{n}_j^s, \mathbf{c}_i^s, \mathbf{c}_j^s)$ (4)
      $\mathcal{I} \leftarrow \texttt{HashIndex}(\mathbf{F}, \delta, \theta, \boldsymbol{\sigma})$ (6)
      $\alpha_s \leftarrow \texttt{PlanarRotAngle}(\mathbf{n}_i^s, \mathbf{p}_i^s, \mathbf{p}_j^s)$
      while $\{\alpha_m, i_m\} \leftarrow \texttt{HashSearch}(\mathcal{H}, \mathcal{I})$ **do**
        $\alpha \leftarrow \alpha_m - \alpha_s$ (11)
        $i_\alpha \leftarrow \lfloor \frac{\alpha}{\theta} \rfloor$
        $\mathbf{A}(i_m, i_\alpha) \leftarrow \mathbf{A}(i_m, i_\alpha) + 1$
  $\mathbf{T}_{s \to g} \leftarrow \texttt{InterTransform}(\mathbf{p}_i^s, \mathbf{n}_i^s)$
  $\{\mathcal{V}_{.i_m}, \mathcal{V}_{.i_\alpha}\} \leftarrow \texttt{PickHighVotes}(\mathbf{A}, \tau_a)$
  foreach $v \in \mathcal{V}$ **do**
    $\mathbf{T}_{m \to g} \leftarrow \texttt{InterTransform}(\mathbf{p}_{v.i_m}^m, \mathbf{n}_{v.i_m}^m)$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \texttt{GetPose}(\mathbf{T}_{s \to g}, \mathbf{T}_{m \to g}, v.i_\alpha)$ (8)
$\mathcal{P} \leftarrow \texttt{PoseClustering}(\mathcal{P}, N_p)$ (§II-E)

where $\mathbf{c}_i$ and $\mathbf{c}_j \in \mathbb{R}^3$ are color vectors. For generality, each color channel is normalized as $c \in [0; 1]$. Fig. 2 illustrates the CPPF.

C. Object Learning

Given a set of object point clouds, an object representation is learned globally by calculate all possible CPPF from the training data. While several approaches [21], [25] require known CAD model of the object for training, our approach directly learns from scanned point clouds of the object. Thus our approach can easily learn new objects although their CAD models are not available. Once the set of CPPF features are calculated, it is saved in a hash table data structure $\mathcal{H}$ for efficient feature matching [23], [21].

To use the CPPF as the key for hash table, we need to quantize the feature descriptors:

$$\mathcal{I} = \texttt{HashIndex}(\mathbf{F}_{\mathsf{CPPF}}, \delta, \theta, \boldsymbol{\sigma}) \tag{5}$$

$$= \begin{pmatrix} \lfloor \frac{\|\mathbf{d}\|_2}{\delta} \rfloor \\ \lfloor \frac{\angle(\mathbf{n}_i, \mathbf{d})}{\theta} \rfloor \\ \lfloor \frac{\angle(\mathbf{n}_j, \mathbf{d})}{\theta} \rfloor \\ \lfloor \frac{\angle(\mathbf{n}_i, \mathbf{n}_j)}{\theta} \rfloor \\ \lfloor \mathbf{c}_i \oslash \boldsymbol{\sigma} \rfloor \\ \lfloor \mathbf{c}_j \oslash \boldsymbol{\sigma} \rfloor \end{pmatrix} \tag{6}$$

where $\delta \in \mathbb{R}, \theta \in \mathbb{R}, \boldsymbol{\sigma} \in \mathbb{R}^3$ are quantization levels for distance, angle, and color vectors, respectively. The symbol $\oslash$ denotes component-wise division.

With this index $\mathcal{I}$ of the feature $\mathbf{F}_{\mathsf{CPPF}}$, necessary information for pose estimation is saved in an entry of the hash table $\mathcal{H}$. By storing the features in $\mathcal{H}$, similar CPPFs are grouped together in the same slot, and matching with scene CPPFs can be done in constant time on average.

The object learning process is presented in Algorithm 1 where referred equations and sections are marked as $(\cdot)$ and $(\S\cdot)$ in the comments area, respectively. Given object model point clouds $\mathcal{M}$, the algorithm returns the learned hash table $\mathcal{H}$. The N_m designates the number of points in $\mathcal{M}$. The α_m is the intermediate angle that will be explained in Section II-D. The quantization parameters $\delta, \theta, \boldsymbol{\sigma}$ are important to set. While choosing very large levels reduce discriminative power of the feature, using very small levels make the algorithm sensitive to noise. In our experiment, we empirically found that $\delta = 2$ (mm), $\theta = 10°$, and $\boldsymbol{\sigma} = (0.1, 0.1, 0.4)^\mathsf{T}$ work well. For the color quantization levels $\boldsymbol{\sigma}$, we use the HSV color space. The V (value or intensity) channel is not generally invariant to illumination changes, and hence a larger level (0.4) is used. Although we empirically found these parameters, searching optimal parameters possibly for each object would be interesting future work.

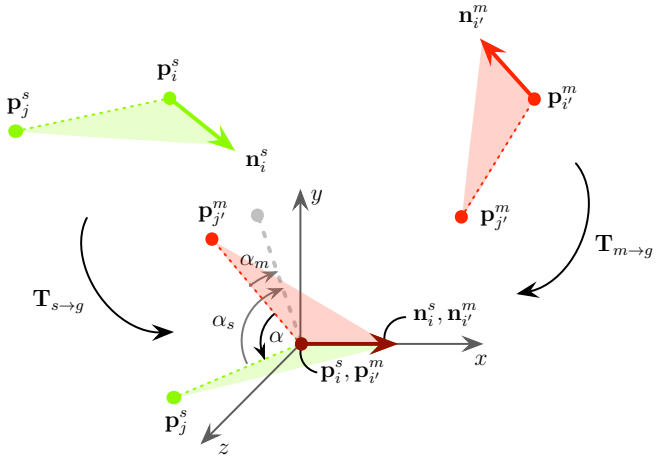

Fig. 3. **Aligning pair features in the intermediate coordinate system.** By $\mathbf{T}_{s \to g}$, the scene reference point $\mathbf{p}_i^s$ is moved to the origin and its orientation (normal or direction) $\mathbf{n}_i^s$ is aligned to the $\mathbf{x}$-axis. The model reference point is similarly transformed by $\mathbf{T}_{m \to g}$, such that the positions and orientations of the reference points are aligned. The referred points $\mathbf{p}_j^s$ and $\mathbf{p}_{j'}^m$ are then aligned by a rotation with angle α around the $\mathbf{x}$-axis.

Fig. 4. **Test Objects.** Ten daily objects were chosen. Each object model is learned by combining multiple views of object point clouds. From left to right: **Clorox**, **Flash**, **Kuka Mug**, **Milk**, **MVG Book**, **Orange Juice**, **Pringles**, **Starbucks Mug**, **Tide**, and **Wrench**.

D. Voting Scheme

Let's assume that we found a correct match of CPPFs between scene and model point clouds. As described in Fig. 3, we can align two normal vectors $\{\mathbf{n}_i^s, \mathbf{n}_{i'}^m\}$ of the two reference points $\{\mathbf{p}_i^s, \mathbf{p}_{i'}^m\}$ in an intermediate coordinate system. The alignment of two reference points constrains 3-DOF translation and the alignment of the two normals further constrains 2-DOF rotation. Therefore, there is only 1-DOF rotation ambiguity $\alpha \in \mathbb{R}$ around the $\mathbf{x}$-axis of the intermediate coordinate system. Once the α is determined by the two vectors $\mathbf{p}_j^s - \mathbf{p}_i^s$ and $\mathbf{p}_{j'}^m - \mathbf{p}_{i'}^m$, we can recover the pose of the object, $\mathbf{P} \in SE(3)$, which is the full 6-DOF rigid body transformation from the model coordinate system to the scene coordinate system via

$$\mathbf{P} = \mathbf{T}_{m\to s} \tag{7}$$

$$= \mathbf{T}_{s\to g}^{-1}\mathbf{R}_{\mathbf{x}}(\alpha)\mathbf{T}_{m\to g} \tag{8}$$

where $\mathbf{R}_{\mathbf{x}}(\alpha)$ is the rotation around the $\mathbf{x}$-axis with angle α, $\mathbf{T}_{s\to g} \in SE(3)$ and $\mathbf{T}_{m\to g} \in SE(3)$ are the transformations from the scene and model coordinate systems to the intermediate coordinate system, respectively. For a quick verification, the referred points $\{\mathbf{p}_j^s, \mathbf{p}_{j'}^m\}$ can be aligned by $\mathbf{P}$ as

$$\mathbf{p}_{j'}^s = \mathbf{P}\mathbf{p}_{j'}^m \tag{9}$$

$$= \mathbf{T}_{s\to g}^{-1}\mathbf{R}_{\mathbf{x}}(\alpha)\mathbf{T}_{m\to g}\mathbf{p}_{j'}^m. \tag{10}$$

It is possible to choose any arbitrary intermediate coordinate system, but a trivial choice is choosing the sensor coordinate system.

Unfortunately, the aforementioned assumption of a correct correspondence between two CPPFs is not always valid. In reality, there is a nontrivial amount of similar geometric and colored surfaces between the actual object and cluttered background point clouds. Due in part to sensor noise and to illumination changes, it happens that these similar regions result in incorrect pose hypotheses. To address this issue, a voting process is performed so that it finds the most likely pose hypothesis from the bin earned the maximum number of votes [21]. Since there is two associations $\{\mathbf{p}_{i'}^m, \alpha\}$ given a scene CPPF, we need to create a 2D accumulator space $\mathbf{A} \in \mathbb{R}^{N_m \times \lfloor \frac{\pi}{\theta} \rfloor}$ for the voting process. Each corresponding model CPPF searched from the hash table $\mathcal{H}$ casts a vote in the space $\mathbf{A}$ such that high votes in $\mathbf{A}$ are highly likely to be valid poses. Note that the α could be calculated online, but it is more efficient if α_m, the angle between the vector $\mathbf{p}_{j'}^m - \mathbf{p}_{i'}^m$ and the upper $\mathbf{xy}$ half-plane, is pre-calculated and saved in $\mathcal{H}$. Then all α for every corresponding model features can be determined by one calculation of α_s and cheap minus operations as

$$\alpha = \alpha_m - \alpha_s. \tag{11}$$

Algorithm 2 deliberatively describes the voting process. Referred equations, algorithms, and sections are cited as $(\cdot)$, $\langle\cdot\rangle$, and $(\S\cdot)$ in the comments area, respectively. As inputs, it takes the hash table $\mathcal{H}$, the object model point clouds $\mathcal{M}$, and N_s points of the test scene point cloud $\mathcal{S}$. It then returns N_p pose hypotheses $\mathcal{P}$ as outputs. Both the sampling ratio γ_s of scene points and the threshold for voting τ_v control the trade-off between speed and accuracy. In our experiment, we consider $N_p = 10$ pose hypotheses and examine all scene points $\gamma_s = 1.0$. We usually set $\tau_v = 10$ but slightly tune depending on the size of the object. The `RandomSample(`N`)` returns a random number between 1 and N without repetition, and `InterTransform(`$\mathbf{p}, \mathbf{n}$`)` calculates the aligned transform using given set of point $\mathbf{p}$ and normal $\mathbf{n}$. Lastly, `PoseClustering(`$\mathcal{P}, N_p$`)` clusters the raw pose hypotheses $\mathcal{P}$ together in a set of N_p grouped poses that will be explained in the following section.

E. Pose Clustering

In Algorithm 2, the pose hypotheses $\mathcal{P}$ appended per each outer voting loop are directly resulted from the CPPFs of the i-th reference scene point and remaining referred scene points. It means that another reference scene point also may result in pose hypotheses similar to some of the $\mathcal{P}$. This fact is valid until the object is a rigid body. Therefore, a post-processing is required so that similar poses are grouped together to give more stable results [21], [24], [25]. Since advanced clustering methods such as mean shift [28], [27] are computationally expensive, we employ an efficient agglomerative clustering. The function `PoseClustering(`$\mathcal{P}, N_p$`)` takes un-clustered pose hypotheses $\mathcal{P}$ as an input, and sort them in decreasing order of the number of votes. Starting by creating a new cluster with the pose hypothesis having the highest votes, similar poses are grouped together. If a pose is far from the existing clusters, a new cluster is created. The distance testing of between poses is based on a fixed thresholds in translation and rotation. When the clustering is finished, the cluster are sorted again, and top N_p pose clusters are returned.

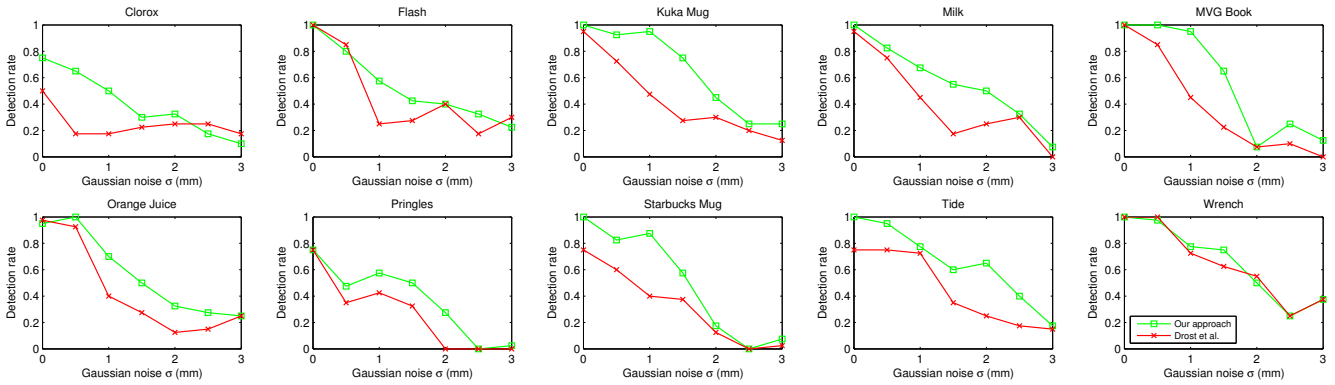

Fig. 5. **Detection rates against Gaussian noise σ.** As σ increases, the performance of both approaches decrease. But in general our approach clearly outperforms Drost et al. [21] in every objects.

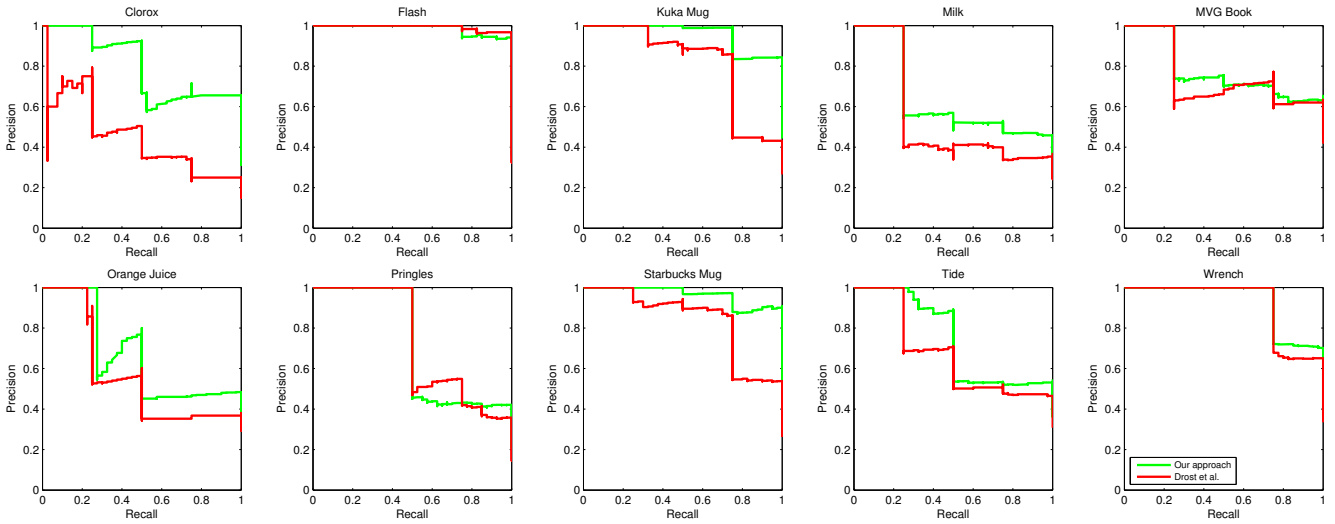

Fig. 6. **Precision-recall curves for the noise experiment.** These curves were drawn by varying the threshold on the total number of votes of the N_p pose results. It turns out that our CPPF is more robust to Gaussian noise than the PPF of Drost et al. [21].

III. Experimental Results

In this section, we present a set of comparative experiments between Drost et al. [21] and our approach. First, we compare the performance of the two approaches with respect to Gaussian noise in Section III-A. Second, the performance of the two approaches are evaluated in highly cluttered scenes in Section III-B.

As test objects, 10 daily objects were chosen as shown in Fig. 4. For object training data, multiple views of point clouds were registered. To efficiently register the multiple point clouds, a chess board was placed in a planar background so that initial transformations between views are determined. After refining the registration using ICP [13], the object is segmented from the planar background. We combined 4 scans per object. The Kinect sensor was used to scan both object and test scene point clouds. All model and scene point clouds are subsampled in 2.5D depth images per 4 pixels and 7 pixels respectively.

A. Gaussian Noise

To examine the performance with respect to noise, we added Gaussian noise to object point clouds with different standard deviations as shown in Fig. 8. Each object has 4 model point clouds, and the range of standard deviations are $0.0, 0.5, \cdots, 3.0$ (mm). For statistically meaningful results, 10 different test clouds were generated for each test. Thus the total number of tested point clouds are $4 \times 7 \times 10 = 280$. With the known pose of each object point cloud, we counted the number of true positives, which are within 10 (mm) for translation and 10° for rotation.

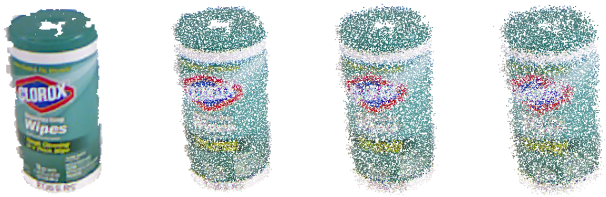

Fig. 8. **Adding Gaussian noise in object point clouds.** To compare performance of CPPF with that of PPF, noisy object point clouds were generated. From left to right: $\sigma = 0.0$, $\sigma = 1.0$, $\sigma = 2.0$, and $\sigma = 3.0$ (mm).

Fig. 5 presents detection rates with respect to the different Gaussian noise. Not surprisingly, the performance of both

Fig. 7. **Selected pose estimation results of Drost et al. [21] (second row) and our approach (third row).** The first row shows the color image of scanned scenes. Correct pose estimates are depicted in color point clouds in second and third rows. In cluttered scenes, [21] works poorly because the low dimensional PPF feature does not give good matches between the model and the scene. Using the color information, CPPF is more discriminative so that pose results from the voting scheme result in the true positive poses. (Best viewed in color)

approaches decrease as the noise level σ increases. However, the graphs clearly show that our approach outperforms Drost et al. [21] in every case. As the original object point clouds are perturbed by the noise, it happens that the PPF and CPPF have false feature matches. These false matches result in wrong pose hypotheses, and thus the final detection results are worsen. However, the CPPF is more discriminative than the PPF mainly due to the color channels. Since a large number of false matches are skipped based on the color similarity, the detection rates are much better than the PPF. The results are also shown as precision-recall curves in Fig. 6. The precision-recall curves were generated by varying the threshold value on the total number of votes of the first pose cluster having the maximum number of votes. The performance differences slightly vary over the test objects, but the overall trend is that our approach shows better detection performance against the Gaussian noises. It is more clear in "Clorox", "Kuka Mug", "Orange Juice", and "Starbucks Mug".

B. Cluttered Scene

As shown in the previous section, both PPF and CPPF work well when there is no clutter. But in real scenarios, robots often need to work in highly cluttered environments in which segmenting each object from the background is impossible like the scenes in Fig. 7. For the test scenes, we put random subsets of our test objects in a paper box with random poses. All other objects not in our test objects were placed as clutter. We captured 31 test scenes, and 7 of them were captured by changing illumination with a non-white lamp. The scene of the right most column in Fig. 7 is one example. For evaluation, the ground truth pose of the objects were carefully annotated. We first performed both pose estimation approaches on the test scenes and ran ICP [13] algorithm starting from the pose results. If the refined poses are close enough to the true pose, we saved them for quantitative analysis. The criterion for correct pose is that differences between a pose and the corresponding ground truth pose are within 15 (mm) for translation and 15° for rotation. Note that we chose slightly bigger thresholds than the thresholds used in the noise experiment because even well estimated ground truth still possesses small amount of errors.

Fig. 7 shows selected pose estimation results from the 31 scenes. The images in the first row are test scene images captured from the Kinect, and the second and third rows represents estimated poses of Drost et al. [21] and our approach in the scene point clouds respectively. For clear visualization, only correct poses are depicted in color point clouds. Please note that any pose refinement processes were not performed for fair comparison between two approaches, though the additional refinements would enhance the final pose accuracy. While [21] recognized at most one object pose per scene, our approach recalled at least more than half of the test objects. It is more clear from precision-recall curves in Fig. 9. Similar to Fig. 6, these precision-recall curves were drawn by varying the threshold value on the number of votes of the best pose cluster. While the performance of our approach is promising, the performance of Drost et al. [21] is extremely not encouraging for these cluttered scenes. Especially, it did not report any true positive poses in some objects such as,"Flash", "Kuka Mug", "Starbucks Mug", and "Wrench". For other objects, the recalls do not reach to 50%, and the precisions are no more than 10%.

Both approach returns $N_p = 10$ pose results, and these multiple poses are sorted in decreasing order of the number

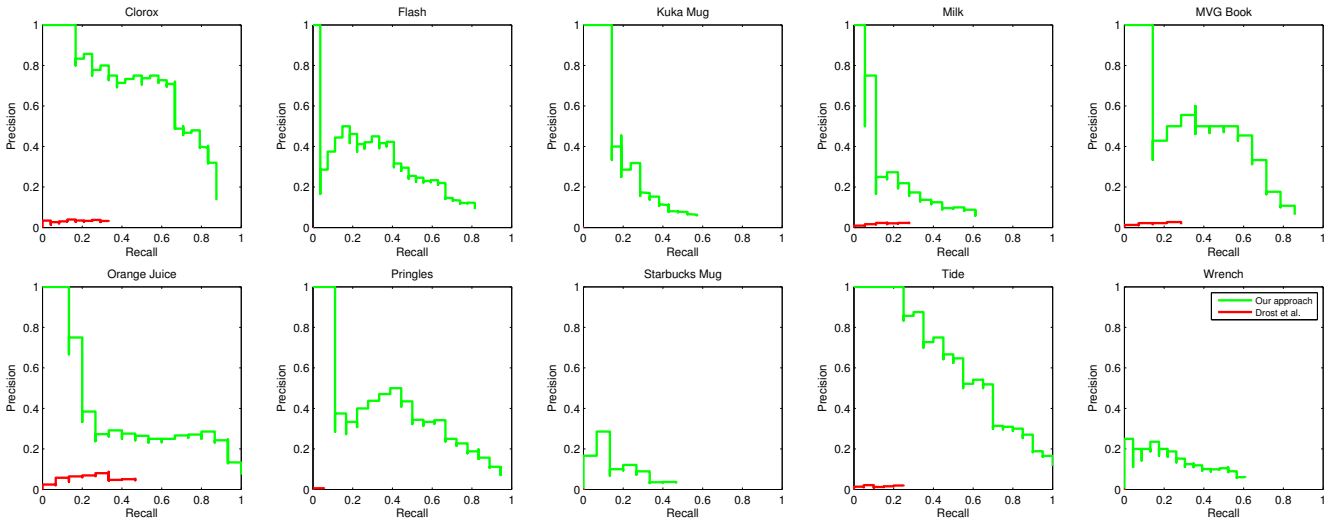

Fig. 9. **Precision-recall curves for cluttered scene experiments.** While our approach reports good precision as well as high recall, Drost et al. [21] works poorly in highly cluttered background.

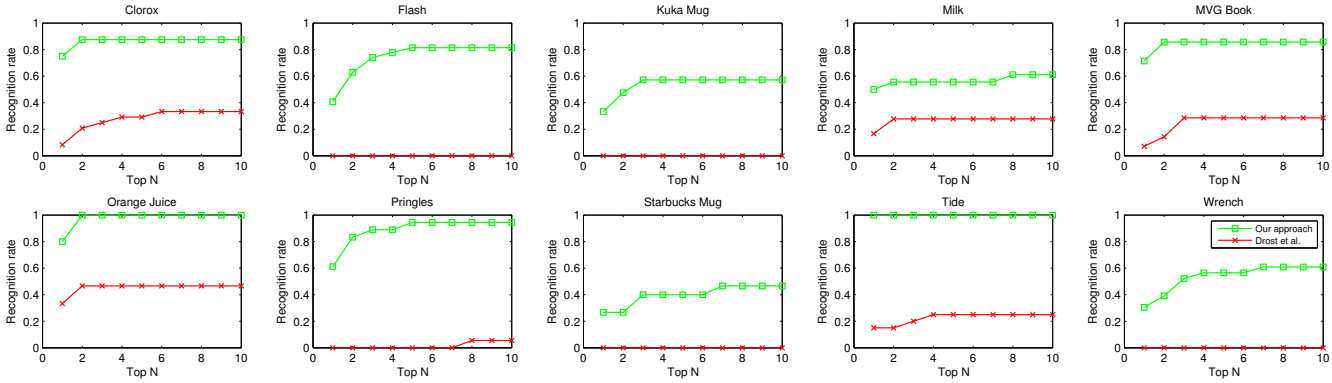

Fig. 10. **Recognition rate of top *N* pose results.** If the ground truth pose is within the top N poses, it is counted as a true positive. While [21] reports poor recognition rates, our approach shows outstanding performance.

TABLE I

AVERAGE AND RELATIVE PROCESSING TIME.

Algorithm	**Average Time**† (sec)	**Relative Time**
Drost et al. [21]	49.208	1.52
Our	**32.379**	**1.00**

† Run on a standard desktop, Intel Core2 Quad CPU with 4G RAM.

of votes. Thus it is interesting to examine the recognition rate with respect to the top N poses. The recognition rate of top N poses on each object is presented in Fig. 10. If the ground truth pose is within the top N poses, we counted it as a true positive. According to Fig. 10, the recognition rates increase as the considered number of results N increases. For both results, the recognition rates are mostly saturated around $N = 3$, which means that the pose hypothesis having higher number of votes is more likely to be the true positive. Note that the best rate of [21], "Orange Juice", is no higher than 50%, while more than half of our results exceed 80% recognition rates.

Our approach is not only more accurate but also more efficient. Our algorithm is efficient due to the sparsity of correspondences between CPPFs. While the PPFs from similar geometric model surfaces are grouped into the same hash slot, the CPPFs from the surfaces are fallen to multiple slots based on the color characteristics. Thus the CPPFs are more broadly distributed in the hash table than the PPFs. This fact leads to more efficient voting algorithm since the number of actual voting is way lower. In addition, since the number of raw pose hypotheses is also lower, the pose clustering process is consequently more efficient. The average processing time of both approaches, which is required to estimate the pose of each object, is shown in Table I, where our approach is 1.52 times faster than the work of Drost et al. [21] on average.

IV. CONCLUSIONS

We presented a voting-based pose estimation algorithm by combining geometric and color information from an RGB-D camera. Our approach learned each object model by scanning multiple point clouds of the object, hence it does not assume an accurate 3D CAD of the object. We have shown a set of comparative experimental results between our approach and the state-of-the-art and verified that our approach outperformed the compared one in terms of both time and accuracy.

V. ACKNOWLEDGMENTS

This work has in part been sponsored by the Boeing Corporation. The support is gratefully acknowledged.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 48–55.

[4] C. Choi and H. I. Christensen, "Robust 3D visual tracking using particle filtering on the SE(3) group," in *Proceedings of IEEE International Conference on Robotics and Automation (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 *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, 1977, pp. 659–663.

[6] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 850–863, 1993.

[7] C. Olson and D. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Transactions on Image Processing*, vol. 6, no. 1, pp. 103–113, 1997.

[8] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1696–1703.

[9] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[10] 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 Transactions on Graphics*, vol. 23, pp. 679–688, 2004.

[11] A. Agrawal, S. Yu, J. Barnwell, and R. Raskar, "Vision-guided robot system for picking objects by casting shadows," *International Journal of Robotics Research*, vol. 29, no. 2–3, pp. 155–173, 2010.

[12] 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 *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2028–2035.

[13] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 239–256, 1992.

[14] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[15] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, and G. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 585–592.

[16] K. Lai, L. Bo, X. Ren, and D. Fox, "A scalable tree-based approach for joint object and pose recognition," in *AAAI Conference on Artificial Intelligence*, 2011.

[17] F. Stein and G. Medioni, "Structural indexing: Efficient 3-D object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 125–145, 1992.

[18] C. Dorai and A. Jain, "COSMOS-A representation scheme for 3D free-form objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1115–1130, 1997.

[19] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, May 1999.

[20] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3212–3217.

[21] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[22] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification," in *Proceedings of International Conference on 3-D Digital Imaging and Modeling (3DIM)*, Oct. 2003, pp. 474–481.

[23] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1584–1601, 2006.

[24] E. Kim and G. Medioni, "3D object recognition in range images using visibility context," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 3800–3807.

[25] C. Choi, Y. Taguchi, O. Tuzel, M. Liu, and S. Ramalingam, "Voting-Based pose estimation for robotic assembly using a 3D sensor," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012 (To appear).

[26] O. Woodford, M. Pham, A. Maki, F. Perbet, and B. Stenger, "Demisting the hough transform for 3D shape recognition and registration," in *Proceedings of British Machine Vision Conference (BMVC)*, 2011.

[27] M. Pham, O. Woodford, F. Perbet, A. Maki, B. Stenger, and R. Cipolla, "A new distance for scale-invariant 3D shape recognition and registration," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011.

[28] O. Tuzel, R. Subbarao, and P. Meer, "Simultaneous multiple 3D motion estimation via mode finding on lie groups," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2005, pp. 18–25.