Robotics and Autonomous Systems 75 (2016) 595–613

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

RGB-D object pose estimation in unstructured environments[☆]

Changhyun Choi[a,*], Henrik I. Christensen[b]

[a] Computer Science & Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[b] Institute for Robotics & Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA

HIGHLIGHTS

- A point pair feature describing both geometric shape and photometric color and its application to a voting-based 6-DOF object pose estimation.
- Parallel algorithms significantly accelerating the voting process on modern GPU architectures.
- Effective in unstructured scenes in which the prevailing segmentation-based approaches may not be applicable.

ARTICLE INFO

Article history:
Available online 9 October 2015

Keywords:
Pose estimation
RGB-D
Voting scheme
Hough transform
Range sensing
GPU
Parallelization
Bin-picking

ABSTRACT

We present an object pose estimation approach exploiting both geometric depth and photometric color information available from an RGB-D sensor. In contrast to various efforts relying on object segmentation with a known background structure, our approach does not depend on the segmentation and thus exhibits superior performance in unstructured environments. Inspired by a voting-based approach employing an oriented point pair feature, we present a voting-based approach which further incorporates color information from the RGB-D sensor and which exploits parallel power of the modern parallel computing architecture. The proposed approach is extensively evaluated with three state-of-the-art approaches on both synthetic and real datasets, and our approach outperforms the other approaches in terms of both computation time and accuracy.

1. Introduction

As robotic systems are getting deployed from well controlled space to unstructured environments, they are required to operate in highly cluttered scenes. A typical scenario involves pick-and-place tasks, where a robot has to detect the presence of an object in clutter, to pick it up, and to move to a particular location. An important challenge is thus designing a system that can detect and estimate poses for a diverse set of objects. Following the pioneering work by [1], there have been a large number of efforts to tackle the 6 degrees of freedom (6-DOF) pose estimation problem. In spite of the considerable attention and efforts, most of the developed techniques are limited as they perform poorly in the presence of clutter. In addition, most of approaches are computationally challenged due to the high dimensional problem space, and thus the algorithms scarcely perform fast enough for general robotic manipulation tasks.

For the 6-DOF pose estimation problem, it is a typical assumption that 3D object models are known *a priori*. With the 3D object representation, various sensor modalities, such as monocular cameras, stereo rigs, and laser rangefinders, were employed to calculate the rigid body transform from the object coordinate system to the sensor coordinate system. While monocular cameras are relatively cheap and popular, these sensors lose depth information via the perspective projection. Stereo cameras provide depth information, but the depth information is not reliable for textureless regions as the depth is typically estimated via the disparity determined by texture distances. Laser rangefinders provide accurate depth in exchange for low resolution, high cost, and slow frame rate. In addition to these sensors, RGB-D sensor is a good alternative as it overcomes aforementioned restrictions. This sensor can be regarded as a combination of a monocular camera and an active illumination depth sensor which estimates more reliable depth on textureless regions than the stereo camera. This affordable sensor provides depth data along with color image in real-time speed (30 fps). It is thus of interest to investigate how this sensor can be utilized for the 6-DOF object pose estimation problem.

[☆] This paper was presented in part at the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vila Moura, Algarve, Portugal (Choi and Christensen, 2012).
[*] Corresponding author.
E-mail address: cchoi@csail.mit.edu (C. Choi).
URL: http://people.csail.mit.edu/cchoi/ (C. Choi).

http://dx.doi.org/10.1016/j.robot.2015.09.020
0921-8890/© 2015 Elsevier B.V. All rights reserved.

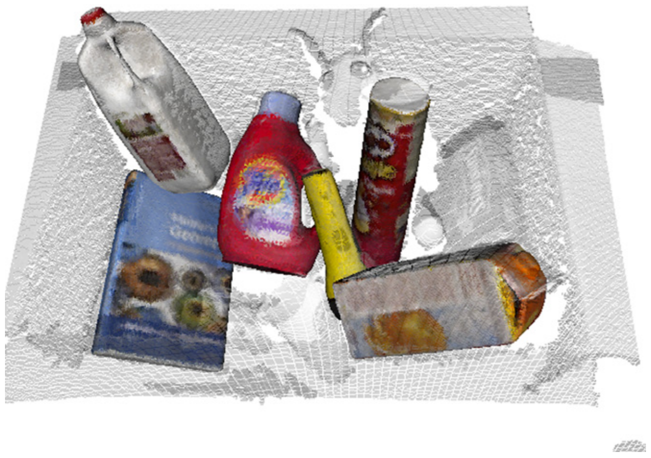


Fig. 1. Overview. Our approach exploits both geometric shape and photometric color information in point clouds for robust object pose estimation. The estimated pose of each object is depicted as a color mesh model. Since our approach does not hinge upon the planar segmentation, it can be applied in highly cluttered environments wherein the planar segmentation might not be applicable.

In this work, we tackle the longstanding robotic perception problem, which is estimating the pose of a known object in heavy clutter. To fully take advantage of the new RGB-D camera, we propose the color point pair feature (CPPF) that selectively encodes the geometric surface shape and the photometric color characteristics. Fig. 1 shows a teaser that depicts the pose estimation results of our approach in a highly cluttered background.

This paper is organized as follows. In Section 2, various pose estimation approaches are revisited, and their advantages and limitations are discussed. After our main contributions are explained in Section 3, the RGB-D pose estimation approach is prudently described in Section 4, where the point pair feature is revisited in Section 4.1, and our color augmented pair feature is then introduced in Section 4.2. We explain the parallelized object learning algorithm which calculates a set of features from an object point cloud in Section 4.3. The voting process on a GPU is shown in Section 4.4, followed by the pose clustering in Section 4.5. Experimental results on various synthetic and real datasets are reported and discussed in Section 5.

2. Related work

2.1. 2D keypoint descriptors

Object recognition and 6-DOF pose estimation are important tasks in robotic perception. For the last decade, stable keypoint descriptors [2,3] have led to successful progress on object recognition. As these keypoint descriptors are invariant to changes in illumination and moderate geometric transformations, keypoint correspondences across different images can be reliably determined. For robotic manipulation, as an object model a set of 3D coordinates of the keypoints is generally required so that a full 6-DOF object pose can be recovered by a set of keypoint correspondences. These keypoint coordinates can be calculated via structure from motion [4] or back-projecting 2D keypoints to a 3D CAD model [5].

2.2. 2D edges and template matching

The keypoint descriptors are suitable for textured objects, but a large number of daily or industrial objects lack texture. For less textured objects, edge features are preferred since they correspond to the geometric shape or boundaries of the objects. A common approach is so-called "Template Matching"; a set of edge image templates of an object is known *a priori*, and in the testing phase the templates are matched with a given edge image. In classic computer vision, the chamfer [6] and Hausdorff [7] distances were proposed as robust metrics, and they were further enhanced by considering edge orientations [8,9]. Common methods to extract edge features from an image are image gradient-based, such as the Canny edge detector [10]. However, these methods often result in unnecessary edges from surface texture or non-Lambertian reflectance. Meaningful edges are obtained from depth discontinuities; [11] introduced the multi-flash camera to detect depth edges by casting shadows from multiple flashes. The sensor was successfully employed in several bin-picking system [12,13].

Beyond the depth discontinuous edges, there were efforts to exploit dense depth information for better recognition performance. [14] utilized the depth data to detect edges from depth discontinuities and adopted a template matching of the signed distance transformed images. [15] combined the image gradient from RGB channels and the surface normal vectors calculated from the depth channel. The work employs a large number of templates for each object to take into account the variability of shape, and it uses the modern SSE instructions for parallel computation in CPU. While the initial work learns object templates online, the later work [16] automatically generates templates from 3D mesh models. Although this work can rapidly detect objects from RGB-D scenes, it is not free from the limitations of template matching: high false positive rates, low accuracy, and not reliable detection outside the learned viewpoints.

2.3. 3D global descriptors

RGB-D sensor provides depth as well as color information in real-time. As the sensor is available at low cost, 3D information-based pose estimation can be more feasible than ever. Unlike 2D images, 3D data maintains depth information which plays a crucial role in registration and pose estimation. The iterative-closest point (ICP) algorithm [17] has been well-known for the registration of 3D point clouds, but it requires a good initial pose estimate to guarantee a good registration. The viewpoint feature histogram (VFH) proposed in [18] 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 6-DOF pose estimation, the clustered viewpoint feature histogram (CVFH) overcoming the previous limitations was subsequently presented [19]. This work further extended to combining local keypoint descriptor, shape, and color features [20], and [21] also reported a similar multimodal approach. For scalable object recognition and pose estimation, [22] proposed a tree structure, yet 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 the planar segmentation that removes a known background plane. Thus all of these approaches are applicable for well structured table-top manipulation. However, they are not robust for complex or cluttered backgrounds.

2.4. 3D local descriptors

For object pose estimation in unknown background shapes or clutter, we cannot rely on the object segmentation. And hence it is required to hypothesize a set of possible transformations between an object model and a scene. Like local image keypoints, several local invariant feature descriptors have been proposed based on the distribution of surface normal around a point [23], surface curvature [24], spin image [25], SHOT descriptor [26], 3D shape context [27],[1] and relative angles between neighboring normals [29]. The descriptors are then compared to generate a set of correspondences between the model descriptors and the descriptors sampled from the given scene. Since the descriptors are locally determined, it is common to use a hypothesize-and-test framework, such as RANSAC [30], in order to estimate a set of consistent rigid body transformations. While these features are invariant to rigid body transformations, they are often sensitive to noise, the resolution difference of point clouds, and the parameters of the descriptors. As a complete review of existing 3D descriptors is beyond the scope of this paper, the reader is referred to [31,32].

2.5. *Point pair features*

Another avenue of pose estimation is to employ voting processes in which all possible pose hypotheses are aggregated in a lower dimensional space. An efficient voting scheme with a point pair feature, defined by two points on surfaces and their normals, was appeared in [33]. In the learning phase, a set of pair features from an object point cloud is computed and saved in a hash table for fast retrieval. In the testing phase, a pair of points is sampled from the sensor data, and the pair matched to that of the pairs in the learned model votes for a pose hypothesis. After the voting process, a set of high votes over a certain confidence level is aggregated to form a possible object pose hypothesis. The pair feature can be seen as a successor of the surflet pairs [34], and using a hash table for fast matching was also appeared in [35,36]. This approach was recently enhanced by incorporating the visibility context [37] or considering object boundary information [38]. There are also several variants of Hough transforms [39] using the SRT distance [40].

The voting algorithm is composed of an amount of repetitive and independent operations, and thus the algorithm is inherently possible to be parallelized. Modern graphics processing unit (GPU) provides massive parallel computation power beyond rendering purposes, and hence it is a natural idea to design a parallel algorithm to be accelerated on the GPU. One good example of parallel algorithms in computer vision is the particle filter [41], which is straightforward to be parallelized as the likelihood evaluation of each particle is independent of the other particles. [42] showed a preliminary design of a particle filter on a GPU for a simple 2D object tracking, and some approaches presented GPU accelerated particle filters for 3D object pose tracking, for which they render a 3D object model to a GPU and a CUDA kernel directly accesses the rendering results to calculate importance weights of the particles [43–45]. The aforementioned work of [14] was parallelized on a GPU so that the distance transform and the downhill simplex optimization are accelerated. [46] showed a parallel implementation of [33] for an object-based camera localization and mapping. Their approach is similar to our proposed approach in the sense that the parallelization was achieved by employing a GPU parallel library, AMD Bolt C++ library [47].

The point pair feature was also employed in a sampling strategy [48] based on RANSAC [30]. Instead of performing the voting process, the work samples two points from the given scene and immediately calculate the transformation between the model and scene pair features. This approach can recognize the object of interest efficiently in relatively simple scenes, but it usually suffers from low recall rate unless the number of RANSAC iterations is big enough. Moreover, it excessively reduces the number of model point pairs by only considering a set of pairs having a certain point distance and by reducing the number of features on populated hash slots. It may reduce computation time but certainly degrade the recognition performance.

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 ameliorated 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 may come from the clutter. If surface shapes of our object model and the clutter are similar to 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 prevent 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 are more likely to contribute to true pose hypotheses. Moreover, the voting process is more efficient since unnecessary votes are disregarded. Employing color information for computational perception is not new; a color variant of SIFT feature [49] is one example. [50,51] used color channels for planar-patch data association. The SHOT local descriptor [26] was also augmented with color information in order to represent local shape and texture information [52].

3. Contributions

Our main contributions are as follows:

- **CPPF**: We present the color point pair feature which enables the voting-based pose estimation to be more efficient. Although employing color information is not novel idea, to the best of our knowledge it is the first effort to augment the point pair feature by utilizing color. We further discuss why it is more accurate and efficient, and the color quantization learning from data is also presented.
- **Parallelization**: Inspired by the parallel operation of the voting scheme, we present a set of parallel algorithms for pose estimation. Our approach is significantly accelerated on a GPU so that it processes an RGB-D scene in about one second.
- **Datasets**: We present an extensive dataset for the RGB-D pose estimation problem with ground truth pose information. The dataset is composed of synthetic and real RGB-D scenes along with 10 object mesh models, and it is designed for various evaluations, such as noise, multiple objects single instance, single object multiple instances, and highly cluttered scenes.

The preliminary result of this work was presented in [53]. The major improvements in this work compared to the previous work are as follows:

- **Object model**: While the preliminary work used a set of partial point clouds registered in a common object coordinate frame, this work employs 3D polygonal meshes as object models (Fig. 6). The former is restricted in terms of view coverage, the latter is more general as it allows to recognize at any viewpoints.
- **Parallelization**: We tackle the inherent computational complexity of the voting procedure by decomposing the computation into parallel operations. To process one RGB-D scene, [53] takes more than 30 s, whereas this work only takes about 1 s.
- **Evaluations**: While [53] was only compared with Drost et al. [33], this work is rigorously evaluated with Papazov et al. [48] and Hinterstoisser et al. [16] as well as Drost et al. [33]. GPU-accelerated version of Drost et al. [33] is also compared (please see Section 5).

[1] 3D shape context descriptor [27] is a 3D extension of the 2D shape context [28].

```
Algorithm 1: RGB-D Pose Estimation
  Data: $\mathcal{M} = \{(\mathbf{p}_1^m, \mathbf{n}_1^m, \mathbf{c}_1^m), \cdots, (\mathbf{p}_{N_m}^m, \mathbf{n}_{N_m}^m, \mathbf{c}_{N_m}^m)\}$, $\mathcal{S} = \{(\mathbf{p}_1^s, \mathbf{n}_1^s, \mathbf{c}_1^s), \cdots, (\mathbf{p}_{N_s}^s, \mathbf{n}_{N_s}^s, \mathbf{c}_{N_s}^s)\}$
  Result: $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \cdots, (\mathbf{P}_{\widehat{N_p}}, v_{\widehat{N_p}})\}$
  Params: $N_v, \gamma_s, \delta, \theta, \sigma, N_c$
1: $\{\mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{D}_h, \mathcal{I}_s^r\} \leftarrow$ ObjectLearning($\mathcal{M}$)                    ⟨2⟩
2: $\{\mathcal{V}, i_v\} \leftarrow$ Voting($\mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{D}_h, \mathcal{I}_s^r$)           ⟨3⟩
3: $\mathcal{P} \leftarrow$ ComputePoses($\mathcal{M}, \mathcal{S}, \mathcal{V}, i_v$)                               ⟨4⟩
4: $\mathcal{P} \leftarrow$ ClusterPoses($\mathcal{P}$)                                       ⟨5⟩
```

- **Color quantization step learning**: Whereas [53] found the color quantization step empirically, this paper presents an automatic step learning from a set of synthetic data. (please see Section 5.8)

4. RGB-D pose estimation

In this section, our RGB-D pose estimation algorithm is presented. By revisiting the point pair features of [33] in Section 4.1, we introduce our color point pair feature in Section 4.2. In Section 4.3, the parallel object learning algorithm is presented, and the voting process and pose clustering are explained in Sections 4.4 and 4.5, respectively. The overall algorithm is shown in Algorithm 1 which takes an object model point cloud $\mathcal{M}$ and an scene point cloud $\mathcal{S}$ as inputs and returns a set of pose hypotheses $\mathcal{P}$ as an output. The more details of the parameters and the algorithms, referred as $\langle\cdot\rangle$ in the comments area, will be described in the subsequent sections.

4.1. Point pair feature

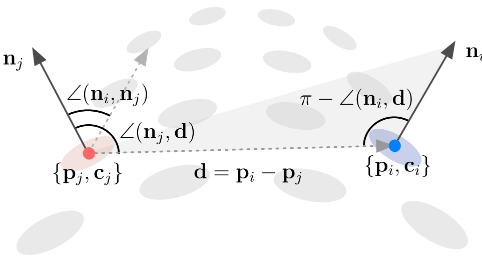
The point pair feature (PPF) is defined by pairing two oriented points, and it has been employed in shape-based object recognition [34,33]. Let $\{(\mathbf{p}_i, \mathbf{n}_i), (\mathbf{p}_j, \mathbf{n}_j)\}$ denote the pair of two oriented points where $\mathbf{p}_i$ and $\mathbf{p}_j \in \mathbb{R}^3$ are the *reference* and *referred* points on the object surface, and $\mathbf{n}_i$ and $\mathbf{n}_j \in \mathbb{R}^3$ are their normals respectively. The point pair feature vector $\mathbf{F}_{\mathsf{PPF}} \in \mathbb{R}^4$ is then defined as

$$\mathbf{F}_{\mathsf{PPF}} = \mathrm{PPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j) = \begin{pmatrix} \|\mathbf{d}\|_2 \\ \pi - \angle(\mathbf{n}_i, \mathbf{d}) \\ \angle(\mathbf{n}_j, \mathbf{d}) \\ \angle(\mathbf{n}_i, \mathbf{n}_j) \end{pmatrix} \tag{1}$$

where $\mathbf{d} = \mathbf{p}_i - \mathbf{p}_j \in \mathbb{R}^3$, and $\angle(\mathbf{v}_1, \mathbf{v}_2) \in [0, \pi]$ represents the angle between two vectors, $\mathbf{v}_1$ and $\mathbf{v}_2 \in \mathbb{R}^3$. The first dimension, $\|\mathbf{d}\|_2 = \|\mathbf{p}_i - \mathbf{p}_j\|_2 \in \mathbb{R}^+$, represents the Euclidean distance between the two 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.

4.2. Color point pair feature

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 (CPPF) vector $\mathbf{F}_{\mathsf{CPPF}} \in \mathbb{R}^{10}$ is defined by concatenating two 3D color vectors of the points:



$\mathbf{n}_j$
$\mathbf{n}_i$
$\angle(\mathbf{n}_i, \mathbf{n}_j)$
$\pi - \angle(\mathbf{n}_i, \mathbf{d})$
$\angle(\mathbf{n}_j, \mathbf{d})$
$\{\mathbf{p}_j, \mathbf{c}_j\}$
$\mathbf{d} = \mathbf{p}_i - \mathbf{p}_j$
$\{\mathbf{p}_i, \mathbf{c}_i\}$

Fig. 2. The color point pair feature (CPPF). The feature is defined by the pair of two color orientated points $\{(\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i), (\mathbf{p}_j, \mathbf{n}_j, \mathbf{c}_j)\}$ where each dimension is determined by the Euclidean distance between two points ($\|\mathbf{p}_i - \mathbf{p}_j\|_2$), angles between surface normals and distance vector ($\pi - \angle(\mathbf{n}_i, \mathbf{d})$, $\angle(\mathbf{n}_j, \mathbf{d})$, $\angle(\mathbf{n}_i, \mathbf{n}_j)$), and colors ($\mathbf{c}_i, \mathbf{c}_j$).

```
Algorithm 2: ObjectLearning($\mathcal{M}$)
  Data: $\mathcal{M}$
  Result: $\mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{D}_h, \mathcal{I}_s^r$
  Params: $\delta, \theta, \sigma$
1: $\mathcal{H} \leftarrow \mathbf{0}_{N_m \times N_m}$
2: $\mathcal{A} \leftarrow \mathbf{0}_{N_m \times N_m}$
3: for $i \leftarrow 1$ to $N_m$ do
4:     for $j \leftarrow 1$ to $N_m$ do
5:         if $i \neq j$ then
6:             $\mathbf{F} \leftarrow$ CPPF($\mathbf{p}_i^m, \mathbf{p}_j^m, \mathbf{n}_i^m, \mathbf{n}_j^m, \mathbf{c}_i^m, \mathbf{c}_j^m$)      (2)
7:             $\mathbf{k} \leftarrow$ HashKey($\mathbf{F}, \delta, \theta, \sigma$)                      (3)
8:             $\kappa \leftarrow$ BitEncodeCPPF($\mathbf{k}$)                          (4)
9:             $\alpha_m \leftarrow$ PlanarRotAngle($\mathbf{n}_i, \mathbf{p}_i^m, \mathbf{p}_j^m$)          (§4.4)
10:            $\mathcal{H}(i, j) \leftarrow \kappa$
11:            $\mathcal{A}(i, j) \leftarrow \alpha_m$
12: $\{\mathcal{H}_s, \mathcal{I}_o^s\} \leftarrow$ gpu::Sort($\mathcal{H}$)
13: $\{\mathcal{H}_r, \mathcal{D}_h\} \leftarrow$ gpu::Reduce($\mathcal{H}_s$)
14: $\mathcal{I}_s^r \leftarrow$ gpu::ExclusiveScan($\mathcal{D}_h$)
```

$$\mathbf{F}_{\mathsf{CPPF}} = \mathrm{CPPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j, \mathbf{c}_i, \mathbf{c}_j) = \begin{pmatrix} \mathrm{PPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j) \\ \mathbf{c}_i \\ \mathbf{c}_j \end{pmatrix} \tag{2}$$

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. When it comes to using color information, it is crucial to choose a proper color space to be less variable to illumination changes. The original color information from the RGB-D sensor is in the RGB color space. However, the RGB values tend to change much as illumination intensity varies. As an alternative, HSV (Hue, Saturation, and Value) color space was well studied and proven to be more invariant than the RGB space with respect to illumination changes. Thus we adopt the HSV color space in our experiments.

4.3. Object learning

In the object learning phase, an object representation is learned globally by calculating all possible CPPFs from an object point cloud. When there are N_m points in the point cloud $\mathcal{M}$, $N_m \times N_m$ CPPFs are calculated including N_m self pairs. Once the set of CPPF features are calculated, it is saved in a data structure for the later feature matching. Hash tables have been widely adopted for the purpose due to its fast search time [35,33,37,38,48]. To use the CPPF as the key for hash table, we first need to quantize the feature vector as

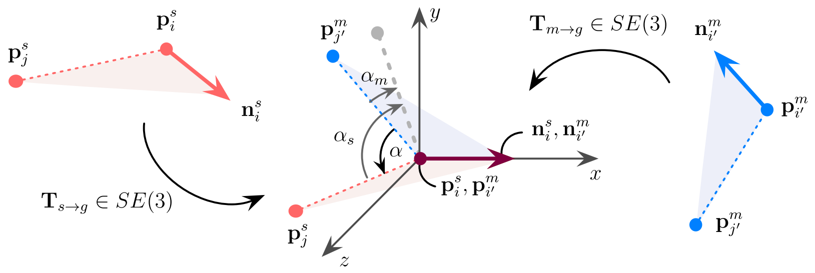


Fig. 3. Feature alignment in the intermediate coordinate system. By aligning the reference points and normal vectors to the origin and the **x**-axis of the coordinate system, respectively, the displacement between scene and model CPPFs is reduced to the 1-DOF rotation α around the **x**-axis. (Please see Section 4.4 for details.)

$$\mathbf{k} = \mathtt{HashKey}(\mathbf{F}_{\mathrm{CPPF}}, \delta, \theta, \boldsymbol{\sigma}) \tag{3}$$
$$= \begin{pmatrix} \left\lfloor \dfrac{\|\mathbf{d}\|_2}{\delta} \right\rfloor \\ \left\lfloor \dfrac{\pi - \angle(\mathbf{n}_i, \mathbf{d})}{\theta} \right\rfloor \\ \left\lfloor \dfrac{\angle(\mathbf{n}_j, \mathbf{d})}{\theta} \right\rfloor \\ \left\lfloor \dfrac{\angle(\mathbf{n}_i, \mathbf{n}_j)}{\theta} \right\rfloor \\ \lfloor \mathbf{c}_i \oslash \boldsymbol{\sigma} \rfloor \\ \lfloor \mathbf{c}_j \oslash \boldsymbol{\sigma} \rfloor \end{pmatrix}$$

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. A key of the CPPF, $\mathbf{k} \in \mathbb{Z}^{10}$, is then bit-encoded as

$$\kappa = \mathtt{BitEncodeCPPF}(\mathbf{k}) \tag{4}$$

where the 64-bit key $\kappa \in \mathbb{Z}$ is encoded as in Fig. 4.

Although the hash table search is designed for constant time search $O(1)$, the performance of the hash tables highly depends on the choice of hash function. Even though we carefully design the hash function, lots of point pairs are inserted in a relatively small number of hash slots due to the symmetric regions of objects. As a result, each hash search does not guarantee the optimal search time. In our approach, we employ the modern parallel computing architecture which is called GPU. For computational acceleration on the GPU, it is required to maintain the data in a simple structure so that a large number of threads can efficiently process the data in parallel. In this respect, we maintain the set of keys in an array storage and use parallel operations for faster search later.

The object learning process is presented in Algorithm 2 where referred equations and sections are marked as $(\cdot)$ and $(\S\cdot)$ in the comments area, respectively. Given an object model point cloud $\mathcal{M}$, the algorithm returns the reduced hash keys $\mathcal{H}_r$ with the intermediate angle array $\mathcal{A}$ and other data required for the later voting process. The N_m designates the number of points in $\mathcal{M}$, and the α_m is the intermediate angle stored in $\mathcal{A}$ that will be explained in Section 4.4. The for-loop in line 3 is parallelized so that each thread takes care of each (i, j) pair. Once the keys are calculated and saved in $\mathcal{H}$, they are first sorted $(\mathcal{H}_s)$ and then reduced so that duplicated keys are removed. By reducing, the sorted keys $\mathcal{H}_s$ are shrunk to the reduced keys $\mathcal{H}_r$ with the array of duplication numbers $\mathcal{D}_h$. The $\mathcal{D}_h$ is further utilized to come up with the indices from the reduced array to the sorted array $\mathcal{I}_s^r$. Similarly, the $\mathcal{I}_o^s$ has the indices from the sorted array to the original array. These two arrays of indices are necessary to map from the reduced array $\mathcal{H}_r$ to the original array $\mathcal{H}$ for the voting process. The parallel primitive operations, such as sort, reduce, and exclusive scan, are available via GPU C++ template libraries, such as NVIDIA Thrust library [54] or AMD Bolt library [47].

The quantization parameters δ, θ, $\boldsymbol{\sigma}$ are important to set. While choosing very large levels reduces the discriminative power of the feature, using very small levels makes the algorithm sensitive to noise. In Section 5.8, we show how the color quantization parameters $\boldsymbol{\sigma}$ are learned from data.

4.4. Voting scheme

Let us 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 **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

$$\begin{aligned} \mathbf{P} &= \mathbf{T}_{m\to s} \\ &= \mathbf{T}_{s\to g}^{-1}\mathbf{R}_{\mathbf{x}}(\alpha)\mathbf{T}_{m\to g} \end{aligned} \tag{5}$$

where $\mathbf{R}_{\mathbf{x}}(\alpha)$ is the rotation around the **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

$$\begin{aligned} \mathbf{p}_{j'}^s &= \mathbf{P}\mathbf{p}_{j'}^m \\ &= \mathbf{T}_{s\to g}^{-1}\mathbf{R}_{\mathbf{x}}(\alpha)\mathbf{T}_{m\to g}\mathbf{p}_{j'}^m. \end{aligned}$$

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 [33]. The usual approach is employing two dimensional accumulator space for the voting process, in which the rows are the model reference points and the columns are discretized bins of α [33,38]. Though it is possible to allocate a big memory space in CPU memory, it is technically impossible to assign the big voting space to each thread in GPU. As a workaround, a big chunk of global memory $\mathcal{V}$, where the maximum size is N_v, is allocated and every votes are accumulated in $\mathcal{V}$. Algorithm 3 carefully describes the voting process. Given the model point cloud $\mathcal{M}$, scene point cloud $\mathcal{S}$, and other data obtained

Algorithm 3: Voting($\mathcal{M}$, $\mathcal{S}$, $\mathcal{A}$, $\mathcal{I}_o^s$, $\mathcal{H}_r$, $\mathcal{D}_h$, $\mathcal{I}_s^r$)

```
Data: $\mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{I}_s^r, \mathcal{D}_h$
Result: $\mathcal{V}, i_v$
Params: $N_v, \gamma_s, \delta, \theta, \boldsymbol{\sigma}$
1: $\mathcal{V} \leftarrow \mathbf{0}_{1\times N_v}$
2: $i_v \leftarrow 0$
3: for $i \leftarrow 1$ to $N_s$ do
4:     for $j \leftarrow 1$ to $N_s$ do
5:         if $i \neq j$ and $\|\mathbf{p}_i^s - \mathbf{p}_j^s\| < \gamma_s$ then
6:             $\mathbf{F} \leftarrow$ CPPF($\mathbf{p}_i^s, \mathbf{p}_j^s, \mathbf{n}_i^s, \mathbf{n}_j^s, \mathbf{c}_i^s, \mathbf{c}_j^s$)    (2)
7:             $\kappa \leftarrow$ HashKey($\mathbf{F}, \delta, \theta, \boldsymbol{\sigma}$)    (3)
8:             $\alpha_s \leftarrow$ PlanarRotAngle($\mathbf{n}_i^s, \mathbf{p}_i^s, \mathbf{p}_j^s$)    (§4.4)
9:             $i_r \leftarrow$ BinarySearch($\mathcal{H}_r, \kappa$)
10:            if $i_r \neq \emptyset$ then
11:                $N_d \leftarrow \mathcal{D}_h(i_r)$
12:                $i_s \leftarrow \mathcal{I}_s^r(i_r)$
13:                for $d \leftarrow 1$ to $N_d$ do
14:                    $i_o \leftarrow \mathcal{I}_o^s(i_s + d)$
15:                    $\alpha_m \leftarrow \mathcal{A}(i_o)$
16:                    $\alpha \leftarrow \alpha_m - \alpha_s$    (7)
17:                    $\widehat{i_v} \leftarrow$ AtomicAdd($i_v$, 1)
18:                    $\mathcal{V}(\widehat{i_v}) \leftarrow$ BitEncodeVote($i, \frac{i_o}{N_m}, \lfloor\frac{\alpha}{\theta}\rfloor$)    (6)
```

Algorithm 4: ComputePoses($\mathcal{M}$, $\mathcal{S}$, $\mathcal{V}$, i_v)

```
Data: $\mathcal{M}, \mathcal{S}, \mathcal{V}, i_v$
Result: $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \cdots, (\mathbf{P}_{N_p}, v_{N_p})\}$
Params: $\delta, \theta, \boldsymbol{\sigma}$
1: $\mathcal{V}_s \leftarrow$ gpu::Sort($\mathcal{V}, i_v$)
2: $\{\mathcal{V}_r, \mathcal{D}_v\} \leftarrow$ gpu::Reduce($\mathcal{V}_s$)
3: for $i \leftarrow 1$ to Length($\mathcal{V}_r$) do
4:     $v \leftarrow \mathcal{V}_r(i)$
5:     $i_s \leftarrow$ ScenePointIndex($v$)
6:     $i_m \leftarrow$ ModelPointIndex($v$)
7:     $i_\alpha \leftarrow$ AlphaIndex($v$)
8:     $\mathbf{T}_{s\to g} \leftarrow$ InterTransform($\mathbf{p}_{i_s}^s, \mathbf{n}_{i_s}^s$)
9:     $\mathbf{T}_{m\to g} \leftarrow$ InterTransform($\mathbf{p}_{i_m}^m, \mathbf{n}_{i_m}^m$)
10:    $\mathcal{P} \leftarrow \mathcal{P} \cup \{$GetPose($\mathbf{T}_{s\to g}, \mathbf{T}_{m\to g}, i_\alpha$), $\mathcal{D}_v(i)\}$    (5)
```

from Algorithm 2, each reference scene point $\mathbf{p}_i^s$ is paired with the other scene points $\mathbf{p}_j^s$. When the Euclidean distance between these two points are within the given search radius γ_s, the CPPF from the two points is calculated and is searched in the model keys $\mathcal{H}_r$ via the binary search. Since $\mathcal{H}_r$ is already sorted and reduced, the binary search is done in $O(\log|\mathcal{H}_r|)$ where $|\mathcal{H}_r|$ is the number of reduced keys. If the corresponding key is found, the number of duplicated keys N_d is looked up via $\mathcal{D}_h$ and the model point index of the sorted array i_s is found from $\mathcal{I}_s^r$. Finally, the original index in the original key array i_o is determined from $\mathcal{I}_o^s$, and this index is used to refer the pre-calculated intermediate angle from $\mathcal{A}$. The set $\{i, \frac{i_o}{N_m}, \lfloor\frac{\alpha}{\theta}\rfloor\}$ comprises a vote for a pose hypothesis, and for the later computation each vote is bit-encoded as

$$\nu = \texttt{BitEncodeVote}\left(i, \frac{i_o}{N_m}, \left\lfloor\frac{\alpha}{\theta}\right\rfloor\right) \tag{6}$$

where the 64-bit vote $\nu \in \mathbb{Z}$ is encoded as in Fig. 5, i and $\frac{i_o}{N_m}$ are indices of scene and model reference points, respectively, and the last term $\lfloor\frac{\alpha}{\theta}\rfloor$ is the discretized angle of α. To avoid race condition in the array $\mathcal{V}$, the atomic add operation is employed. 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 **xy** half-plane, is pre-calculated and saved in $\mathcal{A}$ [33]. Then all α for every corresponding

Algorithm 5: ClusterPoses($\mathcal{P}$)

```
Data: $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \cdots, (\mathbf{P}_{N_p}, v_{N_p})\}$
Result: $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \cdots, (\mathbf{P}_{\widehat{N_p}}, v_{\widehat{N_p}})\}$
Params: $N_c$
1: $\mathcal{P} \leftarrow$ gpu::SortDescending($\mathcal{P}$)
2: $prev \leftarrow next \leftarrow 1$
3: $\widehat{N_p} \leftarrow 0$
4: while $1 \le next \le N_p$ and $\widehat{N_p} < N_c$ do
5:     $next \leftarrow$ gpu::Partition($\{\mathcal{P}(i) \mid prev \le i \le N_p\}, \mathcal{P}(prev)$)
6:     $\mathcal{P}(\widehat{N_p}) \leftarrow$ gpu::Reduce($\{\mathcal{P}(i) \mid prev \le i < next\}$)
7:     $\widehat{N_p} \leftarrow \widehat{N_p} + 1$
8: $\mathcal{P} \leftarrow$ gpu::SortDescending($\mathcal{P}, \widehat{N_p}$)
```

model features can be determined by one calculation of α_s and minus operations as

$$\alpha = \alpha_m - \alpha_s. \tag{7}$$

The set of votes $\mathcal{V}$ is then used to compute the final pose hypotheses in Algorithm 4 which shows how a set of pose hypotheses is obtained. Given the votes array $\mathcal{V}$ and its associated size i_v, it calculates a set of poses and vote numbers $\mathcal{P}$, in which each element is the pair of a pose hypothesis $\mathbf{P}_i \in SE(3)$ and its number of votes $v_i \in \mathbb{Z}$. As in Algorithm 2, $\mathcal{V}$ is sorted and then reduced to avoid redundant calculations on the same votes. In this process, duplicated elements in $\mathcal{V}$ are removed and the number of duplication is updated in $\mathcal{D}_v$. Inside the for-loop, each unique vote element is bit-decoded so that the index of scene reference point i_s (i in Fig. 5), the index of model reference point i_m ($\frac{i_o}{N_m}$ in Fig. 5), and the index of alpha i_α ($\lfloor\frac{\alpha}{\theta}\rfloor$ in Fig. 5) can be obtained. From the indices, the intermediate transforms $\mathbf{T}_{s\to g}$ and $\mathbf{T}_{m\to g}$ are recovered, and finally the pose estimate is calculated via Eq. (5). The sorting and reducing are performed via the parallel library, and the for-loop is parallelized on GPU.

4.5. Pose clustering

In Algorithm 4, it is important to note that each pair of pose and vote number in $\mathcal{P}$ is calculated from the pair of i_sth scene point and i_mth model point. Since the object has N_m model points, many pairs in $\mathcal{P}$ should represent a consistent pose hypothesis. For instance, if a half of N_m points are visible in the given scene, then the ideal number of pairs in $\mathcal{P}$ should be $\frac{N_m}{2}$. But, in reality, the number tends to be much more or less due mainly to scene noise, the occlusion ratio of the object, false matches from clutter, and multiple instances of the same object. To take into account these cases, we need to aggregate similar pose hypotheses. Since advanced clustering methods such as mean shift [55,40] are computationally expensive, we employ an efficient agglomerative clustering. While [33,37,38] have done the similar clustering, we further enhance this process on GPU so that hundreds of thousands elements in $\mathcal{P}$ can be clustered in parallel.

The clustering process on GPU is shown in Algorithm 5. It takes unclustered pose hypotheses $\mathcal{P}$ as an input, and it sorts $\mathcal{P}$ in descending order of the number of votes v_i to make sure that the elements are grouped together to the several most likely pose hypotheses. The algorithm aggregates pose hypotheses until the number of clustered pose hypotheses $\widehat{N_p}$ reaches the maximum number of clusters N_c or every elements are grouped together. The main operation in here is the partition operation which reorders the elements close to the compared pose $\mathcal{P}(prev)$. The similar elements to the compared one are then placed between $prev$ and $next$, and they are aggregated to result in a pose hypothesis $\mathcal{P}(\widehat{N_p})$ by the reduce operation. After the clustering, it ends with the final descending sorting on the clustered pose hypotheses so that the most likely pose hypothesis comes first.

Elements	$\lfloor \frac{\mathbf{c}_j(3)}{\boldsymbol{\sigma}(3)} \rfloor$	$\lfloor \frac{\mathbf{c}_j(2)}{\boldsymbol{\sigma}(2)} \rfloor$	$\lfloor \frac{\mathbf{c}_j(1)}{\boldsymbol{\sigma}(1)} \rfloor$	$\lfloor \frac{\mathbf{c}_i(3)}{\boldsymbol{\sigma}(3)} \rfloor$	$\lfloor \frac{\mathbf{c}_i(2)}{\boldsymbol{\sigma}(2)} \rfloor$	$\lfloor \frac{\mathbf{c}_i(1)}{\boldsymbol{\sigma}(1)} \rfloor$	$\lfloor \frac{\angle(\mathbf{n}_i,\mathbf{n}_j)}{\theta} \rfloor$	$\lfloor \frac{\angle(\mathbf{n}_j,\mathbf{d})}{\theta} \rfloor$	$\lfloor \frac{\pi-\angle(\mathbf{n}_i,\mathbf{d})}{\theta} \rfloor$	$\lfloor \frac{\Vert\mathbf{d}\Vert_2}{\delta} \rfloor$
Bits	63 59	58 54	53 49	48 44	43 39	38 34	33 28	27 22	21 16	15 0

Fig. 4. Bit-encoded 64-bit CPPF key. The quantized CPPF descriptor **k** in Eq. (3) is bit-encoded so that the discretized distance $\lfloor \frac{\Vert\mathbf{d}\Vert_2}{\delta} \rfloor$, discretized angles $\lfloor \frac{\pi-\angle(\mathbf{n}_i,\mathbf{d})}{\theta} \rfloor$, $\lfloor \frac{\angle(\mathbf{n}_j,\mathbf{d})}{\theta} \rfloor$, $\lfloor \frac{\angle(\mathbf{n}_i,\mathbf{n}_j)}{\theta} \rfloor$, and the discretized color values $\lfloor \mathbf{c}_i \oslash \boldsymbol{\sigma} \rfloor$, $\lfloor \mathbf{c}_j \oslash \boldsymbol{\sigma} \rfloor$ are stored in a 64-bit key.

Elements	i	$\frac{i_o}{N_m}$	$\lfloor \frac{\alpha}{\theta} \rfloor$
Bits	63 32	31 6	5 0

Fig. 5. Bit-encoded 64-bit vote. Each vote is composed of three values: the index of scene reference point i, the index of model reference point $\frac{i_o}{N_m}$, and the quantized α angle $\lfloor \frac{\alpha}{\theta} \rfloor$. These values are encoded in a 64-bit vote, and a set of votes is further processed to result in a set of pose hypotheses.

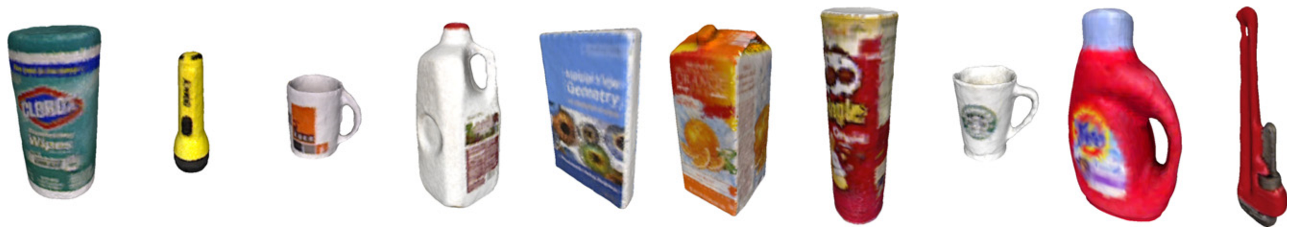


Fig. 6. Polygonal mesh models of the test objects. Ten daily objects were chosen. Each object model is obtained by combining multiple views of object point clouds, followed by the Poisson reconstruction algorithm [56]. From left to right: **Clorox**, **Flash**, **Kuka Mug**, **Milk**, **MVG Book**, **Orange Juice**, **Pringles**, **Starbucks Mug**, **Tide**, and **Wrench**.

5. Experimental results

In this section, we present a set of comparative experiments in which our approach is compared with Drost et al. [33], Papazov et al. [48], and Hinterstoisser et al. [16]. We briefly explain how we build the object models from an RGB-D camera in Section 5.1. Section 5.2 describes our experiment setup, such as the implementations of the approaches, the chosen parameters associated with the implementations, and the machine specifications used for the evaluations. We start the evaluations with a synthetic dataset to compare the performance of the approaches with respect to Gaussian noise in Section 5.3. The performance of the approaches are further evaluated in two sets of synthetic cluttered scenes: multiple objects single instance setting in Section 5.4 and single object multiple instances setting in Section 5.5. In Section 5.6, the approaches are compared in a set of highly cluttered real RGB-D scenes, captured by placing the random selections of the target objects and several other objects as clutter in a paper box. We compare the computation time of the approaches and discuss the reasons of the different efficiency in Section 5.7. Lastly, learning of color quantization step is explained in Section 5.8.

5.1. Object models

As test objects, 10 daily objects were chosen and their polygonal mesh models were generated as shown in Fig. 6. To generate the mesh models, we used an RGB-D camera to capture multiple RGB-D views containing one of the objects. For simplicity, ARTags [57] were employed to register the multiple RGB-D scenes, followed by a planar segmentation which removes the background plane to result in the segmented object of interest. The mesh models were then obtained by running the Poisson reconstruction algorithm [56]. Since our approach requires color information of the objects, color attributes of the point clouds were transferred to their closest vertices in the mesh models so that the mesh models maintain the color information.

As we already explained in Section 4, the object model $\mathcal{M}$ in Algorithm 1 is a model point cloud, and hence we need to convert the mesh models to model point clouds. For the conversion, we employ a sampling approach that randomly samples a set of points from the faces of an object mesh model with the probability proportional to the areas of the faces. Since our approach requires color information, the color attribute of each point is also determined by averaging the color values of the three vertices of the randomly selected face. To generate sufficient samples, we sample 1,000,000 points per object mesh and subsample using a voxel grid with the leaf size 5 mm. By averaging color values in each voxel, we can avoid the texture aliasing which is undesirable artifact when we sample from a high frequency texture. As a large number of CAD models are available from web nowadays, such as the Google 3D Warehouse [58] or the KIT object models web database [59], our approach can easily learn any object as long as its 3D model is available from the database. Even if the 3D model is not available, we can generate the model as we built the models for our test objects (Fig. 6).

5.2. Experiment setup

For experiments, we generated a set of synthetic and real datasets. The synthetic datasets were generated by rendering the polygonal mesh models (Fig. 6) in OpenGL. The projection matrix in OpenGL was set in accordance with the known intrinsic parameters of the RGB-D camera, the ASUS Xtion Pro in our experiments, so that the rendered scenes simulate the captured scenes from the RGB-D camera. The color and depth buffers from each rendering were accessed to save as an RGB-D pcd file which is a *de facto* standard file format of point cloud in the Point Cloud Library (PCL) [60]. When we save the point cloud, the pose values of the rendered objects were also saved as ground truth pose files for quantitative evaluations. More detailed descriptions of the dataset generation will be stated in each following section.

Our approach is compared with Drost et al. [33], Papazov et al. [48], and Hinterstoisser et al. [16]. The approach using spin images [25] is a possible baseline, but it is not considered here since both Drost et al. [33] and Papazov et al. [48] reported better results. We use our own implementation of Drost et al. [33], and its GPU accelerated version is also used to compare the computation time with our approach in Section 5.7. Papazov et al. [48] contributed their implementation to the PCL, so it is chosen for this evaluation. The work of Papazov et al. [48] has two important parameters which determines its performance significantly: the pair width d constraining the first dimension (the distance between two points in a pair which is equivalent to $\|\mathbf{d}\|_2$ in Eq. (1)) of learned PPFs and the probability P_M of recognizing the model in a single iteration. The authors suggested to use about half the maximum distance between visible points for the value of d. Thus we calculate the maximum point distance in each model $d^m_{max} \in \mathbb{R}^+$ as

$$d^m_{max} = \max_{i,j\in[1,N_m]} \|\mathbf{p}^m_i - \mathbf{p}^m_j\|_2 \tag{8}$$

where N_m is the number of model points $\mathcal{M}$ and $\mathbf{p}^m_i, \mathbf{p}^m_j \in \mathbb{R}^3$ are the i, jth model points. Then d is set as

$$d = \gamma \cdot d^m_{max} \tag{9}$$

where γ is the ratio. Although the authors recommended $\gamma = 0.5$, we evaluated the performance of Papazov et al. [48] with different values $\gamma = 0.1, 0.3, 0.5, 0.7, 0.9$, and we found that $\gamma = 0.3$ is the best value. The other parameter P_M decides the number of RANSAC iterations. The default value of P_M is 0.0125, but when we evaluated with this value, the random sampling approach of Papazov et al. [48] barely reported true positives. The number of RANSAC iterations with $P_M = 0.0125$ is

$$\begin{aligned} N &= \frac{\ln(1-P_S)}{\ln(1-P_M)} \\ &= 366.1062 \end{aligned} \tag{10}$$

where $P_S = 0.99$ is the probability of recognizing the model in N trials. Since the approach searches for pose hypotheses only 366 times, the chance of having true positive recognitions is very low. We reduced the parameter as low as $P_M = 0.0005$ so that the number of iterations is sufficiently high as

$$\begin{aligned} N &= \frac{\ln(1-0.99)}{\ln(1-0.0005)} \\ &= 9208.04 \end{aligned} \tag{11}$$

which means the approach of Papazov et al. [48] will randomly search for the pose hypotheses about 9000 times. In this experiments, we set $\gamma = 0.3$ and $P_M = 0.0005$ for Papazov et al. [48].

The LINEMOD in [15] is a template matching approach for pose estimation. Since each template encodes the shape information of an object with a certain pose, it is generally required to have a large set of templates to take into account the shape variations. While the work of [15] learned the templates with a manual interaction from a user, the later work of Hinterstoisser et al. [16] exploited 3D CAD models to generate the templates automatically. Following the setup in Hinterstoisser et al. [16], we used 15° and 10 cm for rotation and depth step sizes. We generated multiple templates for each object by rotating the 3D object model with 15° step in $\mathbf{x} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\mathbf{y} \in (-\pi, \pi]$, and $\mathbf{z} \in (-\pi, \pi]$ axes, which results in both in-plane and out-of-plane rotations. To address scale changes depending on the distance between the sensor and the object, templates were generated with six levels of depth: from 40 to 90 cm with 10 cm step size. Therefore, the total number of templates per each object is $13 \times 24 \times 24 \times 6 = 44{,}928$. Please note that Hinterstoisser et al. [16] sampled only the upper hemisphere of the objects, whereas we sampled the whole sphere as objects in our datasets are randomly placed. This template matching approach requires a large number of templates, which takes an amount of time to generate.[2] As opposed to the time consuming template generation, learning CPPFs is very efficient as our model for each object $\mathcal{M}$ is just a subsampled 3D point cloud and building a hash table is parallelized in GPU.[3] In Hinterstoisser et al. [16], they did not explicitly explain how to recover the set of $SE(3)$ poses from the template matching, and thus we used the coarse pose estimation shown in [61] which estimates the rotation and translation from both the corresponding template's rotation information and the similar triangles. Hinterstoisser et al. [16] employed two post-processing stages in which false positive detections are removed via checking color value followed by the ICP refinement [17]. In this experiment, the additional color checking is not performed. Please note that Hinterstoisser et al. [16] used RGB channels to calculate the color gradient features when they create templates and applied the color-based post-verification for higher precision. To compare the performance of the template matching with that of other approaches, we decide not to perform this refinement exclusively for Hinterstoisser et al. [16] approach. Instead, we run LINEMOD with the ICP as the estimated poses from template matching are approximated and thus geometric refinement is strongly necessary. It will be shown, however, in the following sections that running the ICP refinement would not always enhance final pose estimates. For the ICP parameters, we set 1 cm for the max correspondence distance, 50 for the maximum number of iterations, and 10^{-6} for the transformation epsilon. We chose the LINEMOD implementation in the OpenCV [62] and the ICP implementation in the PCL.[4]

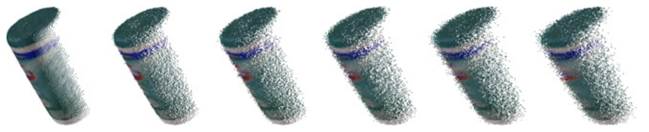


Fig. 7. Adding Gaussian noise in the synthetic noise dataset. To simulate the noise of RGB-D cameras, Gaussian noise is added in the direction of the camera ray. From left to right: $\sigma = 0, 2, 4, 6, 8, 10$ mm.

All approaches except Papazov et al. [48] are deterministic in the sense that the approaches always result in the same pose estimates if identical object model and scene point cloud are given. Papazov et al. [48] is, however, stochastic as it employs the RANSAC algorithm. For statistically meaningful results, we ran 10 trials for Papazov et al. [48] and averaged over the results of the multiple trials, while our approach, Drost et al. [33], and Hinterstoisser et al. [16] with and without the ICP were run only once.

We evaluate the performance of the five approaches quantitatively using the ground truth information. If the difference between an estimated pose and its corresponding ground truth pose is less than 15 mm for translation and 10° for rotation, it is counted as a true positive. As some objects are self-symmetric, symmetry of each object is also taken into account. For example, "Clorox" is a cylinder shape, so any rotation in the axis of symmetry is ignored for the comparison.

For the parameters in Algorithm 2, we set $\delta = 10$ mm, $\theta = 6°$. The color quantization steps $\boldsymbol{\sigma}$ are learned automatically as will be shown in Section 5.8. We set the maximum size of the global vote memory $N_v = 40{,}000{,}000$ and the search radius $\gamma_s = 1.0 \cdot d^m_{max}$ in Algorithm 3. The maximum number of pose clusters is set as $N_c = 10$ in Algorithm 5.

All the experiments were evaluated in a standard desktop computer with an Intel Core2 Quad CPU Q9300, 8G RAM, and an off-the-shelf GPU, NVIDIA GeForce GTX 590 with CUDA 4.1.

[2] In our experiment, it takes about 1.76 h, on average, to generate 44,928 templates.

[3] In our experiment, it takes less than 50 ms as will be shown in Section 5.7.

[4] There is another implementation of LINEMOD in the PCL, but in our evaluation it reported much worse performance than the LINEMOD implementation in the OpenCV.

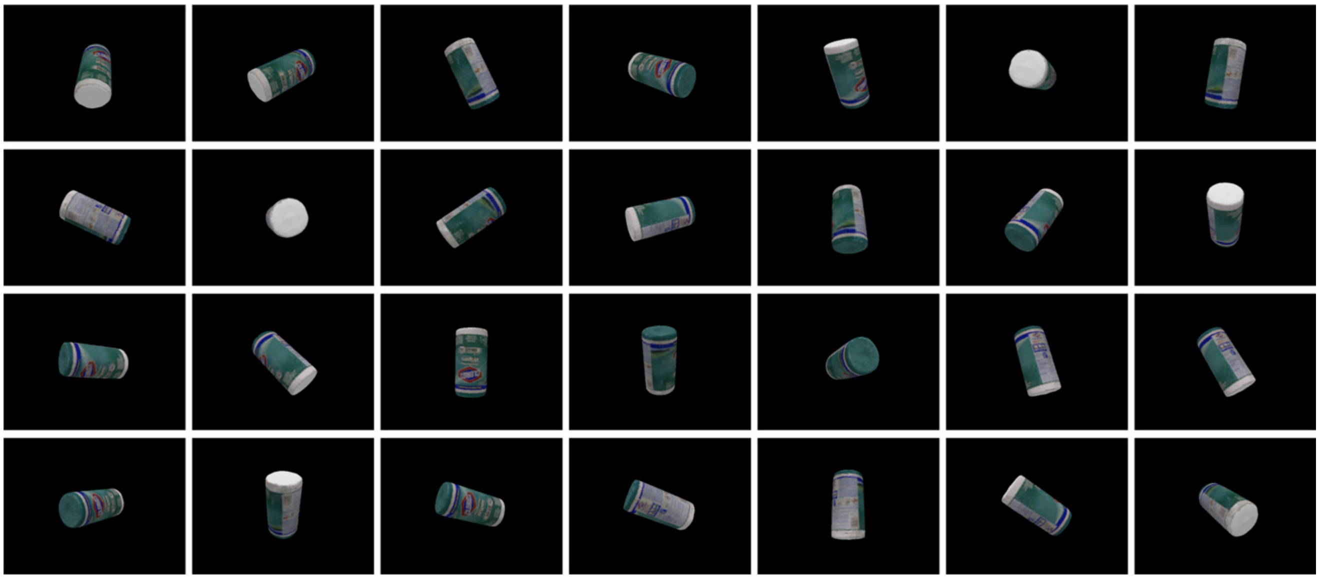


Fig. 8. Some "Clorox" examples of the synthetic noise dataset. To evaluate the performance of five approaches with respect to Gaussian noise, a large set of synthetic dataset was generated. For each object, its mesh model was drawn in OpenGL with a fixed translation (0.5 m apart from the virtual camera) yet with random rotations. Some scenes of "Clorox" object are shown here.

5.3. Gaussian noise

To examine the performance with respect to noise, we generate a set of synthetic noise scenes by adding Gaussian noise with different standard deviations as shown in Fig. 7. To mimic the noise of RGB-D cameras, the Gaussian noise is added in the direction of the camera ray as follows

$$\tilde{\mathbf{p}}^s = \mathbf{p}^s + \frac{\mathbf{p}^s}{\|\mathbf{p}^s\|_2} \cdot n, \quad n \sim \mathcal{N}(0, \sigma^2) \tag{12}$$

where $\tilde{\mathbf{p}}^s, \mathbf{p}^s \in \mathbb{R}^3$ are noisy and noise free points in the synthetic scene respectively, $\frac{\mathbf{p}^s}{\|\mathbf{p}^s\|_2}$ is the unit vector of the camera ray, and $n \in \mathbb{R}$ is the noise from the zero-mean Gaussian distribution with the standard deviation σ. The range of the standard deviations is $0, 2, \ldots, 10$ mm, which are 6 different standard deviations. For statistically meaningful results, 50 different test clouds were generated with random rotations for each object, as some of "Clorox" scenes are shown in Fig. 8. Thus the total number of tested point clouds for 10 test objects are $10 \times 6 \times 50 = 3000$.

Fig. 10 presents recognition rates with respect to the six different amount of Gaussian noise. As we would expect, the recognition rates of the five approaches decrease as the noise level σ increases. Except some cases, our approach reports the best recognition performance in general, and slightly worse performance is shown by Drost et al. [33]. Hinterstoisser et al. [16] shows moderate recognition rates, while Papazov et al. [48] reports the worst recognition performance overall. Hinterstoisser et al. [16] is better or comparable to our approach in the two mug objects, "Kuka Mug" and "Starbucks Mug". It is possibly due to the fact that the templates of Hinterstoisser et al. [16] capture distinctive boundaries of the objects. According to the results of Hinterstoisser et al. [16] with and without the ICP, the ICP refinement helps to increase the recognition rate, but only for smaller Gaussian noise levels. As the standard deviation σ increases, the additional ICP process even worsens due mainly to the false point data association on the noisy point cloud. It is worth noting that the performance of Hinterstoisser et al. [16] is relatively less affected by the Gaussian noise. It is because that approach relies on template matching in which 2D templates are matched against the input scene image. Since the Gaussian noise is added only to the depth channel, the RGB channels are not affected at all, and thus the approach relatively less degenerates. However, our approach still shows better recognition rates than Hinterstoisser et al. [16] in most cases.

The results are also shown as precision–recall curves in Fig. 9. The curves were generated by varying the threshold value on the score of the pose estimates: the number of votes for both our approach and Drost et al. [33], the visibility term μ_V which is the ratio of the model surface area matched to the scene in Papazov et al. [48], and the template matching score in Hinterstoisser et al. [16]. As a performance measure, average precision (AP) is shown at the end of each curves.[5] AP value less than 0.01 is omitted for clear visualization. According to the precision–recall graphs, our approach outperforms other approaches in most cases with some minor exceptions. In both mug objects, Hinterstoisser et al. [16] shows better or comparable precision and recall to our approach. In most cases, Papazov et al. [48] shows poor recall, mostly lower than 30%, except "Milk" and "Tide". Even in "Tide", its recall is at best about 50%, while our approach at least reports about 80% in all cases. Drost et al. [33] exhibits comparable performance to our approach in some objects, and yet our approach shows higher precision in general. From the results of Hinterstoisser et al. [16] with and without the ICP, we can notice that the ICP enhances both precision and recall in general, but it degrades in "Flash", "Kuka Mug", and "Starbucks Mug" objects which are relatively small objects.

The noise dataset only has one object per scene without any backgrounds, and thus all points are corresponding to the test object. In these relatively simple scenes, the performance difference between our approach and other approaches is not so distinctive, but we will see the gap as we evaluate them in more challenging datasets in the following sections.

5.4. Synthetic scenes: Multiple objects single instance (MOSI)

As we discussed in Section 5.3, when the scene is quite simple in the sense that only one object exists in an uncluttered background, pose estimation is quite straightforward. However,

[5] AP is a measure of the area under the precision–recall curve, and it is widely used in object recognition literature to compare performance of various approaches [63].

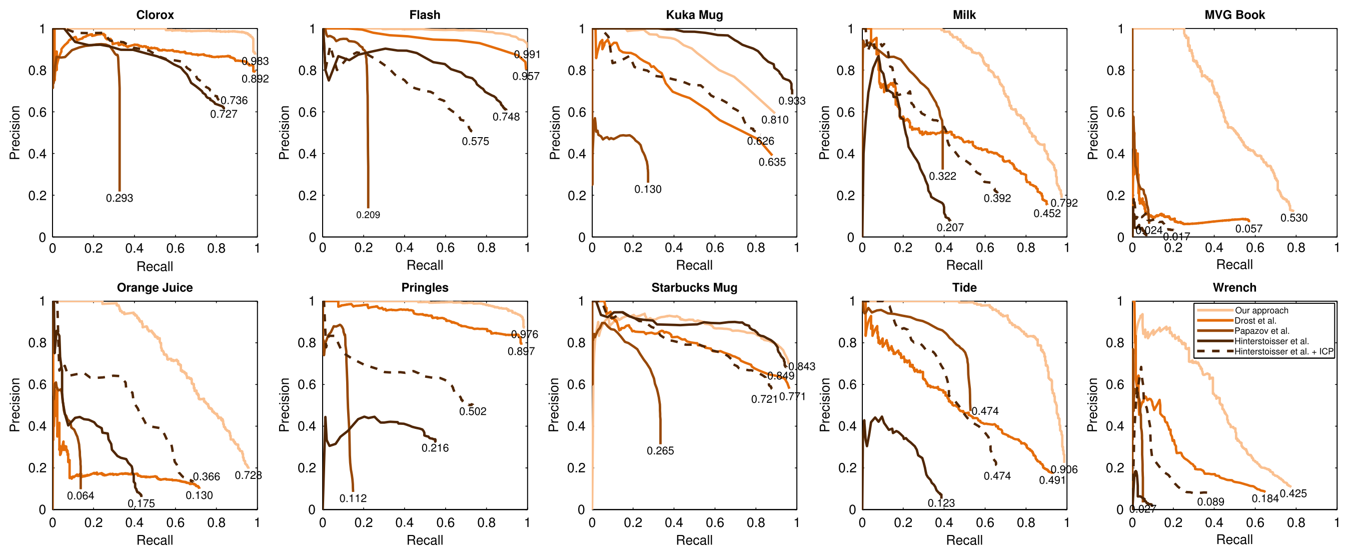


Fig. 9. Precision–recall curves of the noise experiment. Average Precision (AP) value is presented at the end of each curve. Our approach outperforms the other approaches in most cases. The performance of Drost et al. [33] is quite similar to that of ours in some cases, but in general our approach reports higher precision. According to the results of Hinterstoisser et al. [16] with and without the ICP, the ICP algorithm generally enhances both precision and recall but not always. Note that the work of Papazov et al. [48] is suffered from low recall; while it reports about 50% recall at best, our approach shows at least about 80% recall in every case.

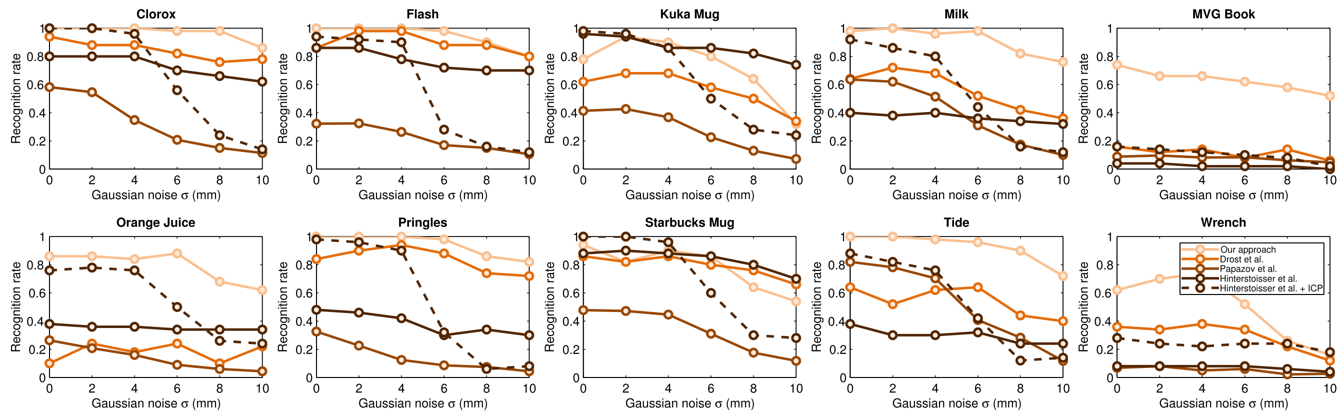


Fig. 10. Recognition rates against Gaussian noise $\boldsymbol{\sigma}$. As σ increases, the performance of the five approaches decreases. Though the performance of the approaches slightly varies, overall our approach outperforms the four compared approaches in most objects.

as environments are getting more cluttered the number of possible pose hypotheses needed to hypothesize and to verify is exponentially increasing with the number of points from cluttered backgrounds. To evaluate the performance of five approaches with respect to clutter, we generated 50 synthetic scenes where random selections of our test objects were placed in either a white laundry basket or a wire basket (Fig. 11), whose 3D models were downloaded from Google 3D Warehouse [58]. Since single instance of each object is considered, we call this setup as multiple objects single instance (MOSI).

Fig. 11 presents selected pose estimation results from the 50 scenes. The first row shows the color images of the synthetic scenes, and the second to sixth rows present the detection results of Hinterstoisser et al. [16] without and with the ICP, Papazov et al. [48], Drost et al. [33], and our approach, respectively. For clear visualization, only correct pose estimates are displayed with color mesh models in the monochrome scene point clouds. Please note that except Hinterstoisser et al. [16] with the ICP any pose refinement processes were not performed for the rest of approaches, though the additional refinements would enhance the final pose accuracy. According to the qualitative results, it is clear that the recognition performance of our approach is superior to the other approaches. In the cluttered scenes, Hinterstoisser et al. [16] without the ICP does not detect any of objects. The main reason of the discouraging performance is due to the limitation of the template matching. Please recall that the templates were generated with the 15° rotational and 10 cm depth steps. Even though we employ the significant number of templates,[6] the templates do not cover entire variations in the appearance of the object. So the detections from the template matching are often not accurate enough to be considered as true positives. The ICP refinement certainly helps in this case. Hinterstoisser et al. [16] with the ICP reports several true positive detections in several scenes. Papazov et al. [48] shows poor performance as it only recognizes one object per scene at best. The reason of the substandard performance may be due to the limited number of searches in the sampling approach. Note that we set the parameter P_M in Papazov et al. [48] small enough to ensure a sufficiently large number of iterations. The number of iterations, 9208 in Eq. (11), might be sufficient for simple scenes, such as the noise experiment dataset in Section 5.3, but it may not be sufficiently

[6] As mentioned in Section 5.2, 44,928 templates were used in our experiments.

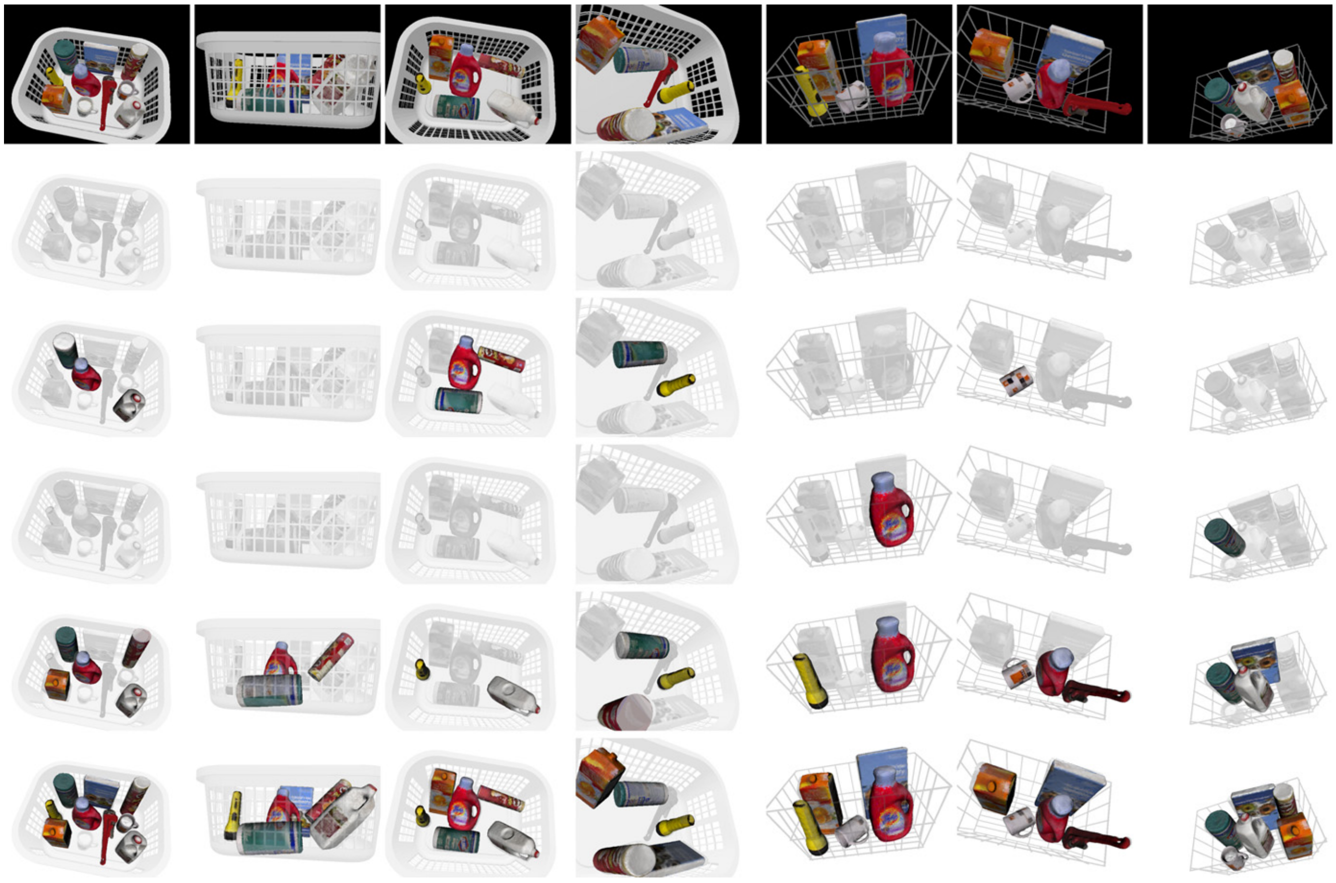


Fig. 11. Selected pose estimation results of Hinterstoisser et al. [16] without the ICP (second row), Hinterstoisser et al. [16] with the ICP (third row), Papazov et al. [48] (fourth row), Drost et al. [33] (fifth row), and our approach (sixth row) in the MOSI dataset. The first row shows the color images of the synthetic scenes. Correct pose estimates are depicted as color mesh models in the second to sixth rows. In these cluttered scenes, Hinterstoisser et al. [16] without the ICP barely detects the target objects due to the coarse sampling of templates. By the additional ICP refinement, it reports some true positive detections, but it is much inferior to both Drost et al. [33] and our approach. Papazov et al. [48] also works poorly mainly due to the relatively insufficient number of searches for these challenging environments. Drost et al. [33] shows better results than Papazov et al. [48] via the voting process that considers a much large number of pose hypotheses, but this approach still misses a number of true positive detections. Thanks to the additional color information encoded in CPPFs, our approach shows the best performance among the five approaches in these cluttered environments.

large enough for these complex scenes. Drost et al. [33] seems more encouraging compared to Papazov et al. [48], but it still fails to recognize some objects of which our approach with the CPPF can successfully recognize and estimate the poses. Due mainly to more discriminative power of the CPPF, our approach outperforms the other approaches.

Fig. 12 presents precision–recall graphs of the MOSI experiment. Like Fig. 9, the graphs were drawn by varying the threshold value on the score of the pose estimates. The precision–recall curves clearly show the distinguished performance of our approach, as it reports about 100% recall near 100% precision in some objects, such as “Clorox”, “Flash”, “Pringles”, and “Tide”. Drost et al. [33] also shows good performance especially in “Clorox”, “Milk”, “Pringles”, and “Tide”, yet it is inferior compared to our approach in terms of both precision and recall. Papazov et al. [48] exhibits moderate performance in some objects, such as “Clorox” or “Tide” which are relatively bigger size and having rich variations in surface normals. Whereas Hinterstoisser et al. [16] with the ICP is comparable to Papazov et al. [48], the approach without the ICP is the worst approach in this dataset.

The five approaches return $N_c = 10$ pose results in maximum, and these multiple poses are sorted in decreasing order of the score. Thus it is of interest to examine the recognition rate with respect to the top N poses. Fig. 13 presents the recognition rates of the top N pose estimates. The recognition rates are calculated by the ratio of true positives which are counted if the ground truth pose is within the top N poses. The recognition rates thus increase monotonously as N increases. According to the plots, the same story is discovered; the recognition rates of our approach are highest among the five approaches, followed by Drost et al. [33]. In both Figs. 12 and 13, it is clear that the ICP improves the recognition rates a lot for Hinterstoisser et al. [16].

5.5. Synthetic scenes: Single object multiple instances (SOMI)

Both experiments of Sections 5.3 and 5.4 were designed to detect one instance of each object at a time. In many real scenarios, however, it happens to exist more than one instance of an object. One of the most popular scenarios, especially in manufacturing, is the bin-picking in which a bunch of identical objects, needed to be picked, are placed in a pile [13,12,38,64,65]. Even in our daily lives, identical objects are often placed in environments, such as dishes and silverware in a shelf or identical milk bottles in a refrigerator. As such, it is of interest to evaluate the performance of five approaches in the multiple instances setting, which is called as single object multiple instances (SOMI).

When there are multiple identical objects in a scene, it tends to have an amount of the similar local features, and hence grouping these features and verifying multiple hypotheses calculated from the features are generally required. The voting-based approaches are especially preferred in this setting, since it is designed to search all possible pose hypotheses and to group together in a set of consistent pose hypotheses via the voting process. For the

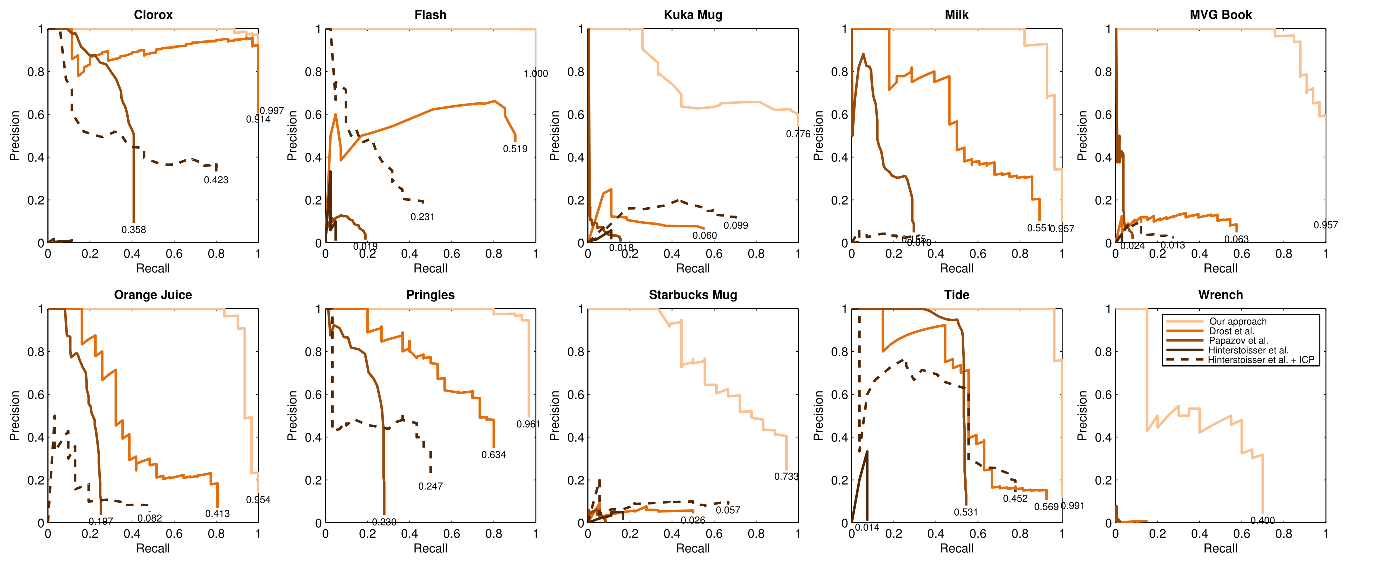


Fig. 12. Precision–recall curves of the multiple objects single instance (MOSI) experiment. Our approach significantly outperforms the four other approaches in both precision and recall. Hinterstoisser et al. [16] without the ICP is the worst approach in this MOSI setting, but with the help of the ICP refinement its recognition performance is enhanced as good as that of Papazov et al. [48].

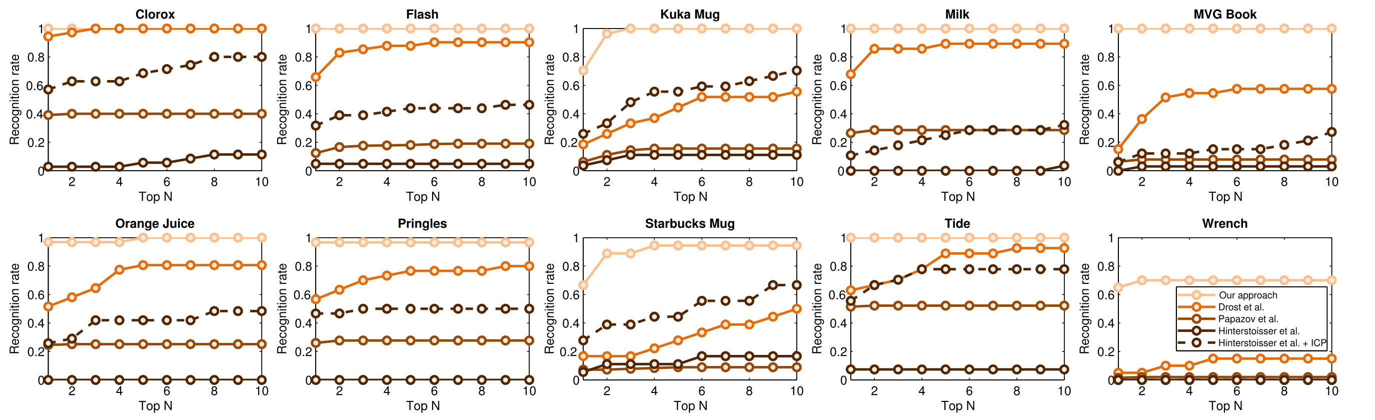


Fig. 13. Recognition rates of top N pose results in the MOSI experiment. If the ground truth pose is within the top N poses, it is counted as a true positive. Our approach shows the best recognition rates in every object.

evaluation in the SOMI setting, we generated 10 synthetic scenes for each object where randomly chosen numbers of instances, between 3 and 10 instances of the object, were randomly placed in the white laundry basket. Like the previous experiment, $N_c = 10$ maximum number of clusters was used.

Some of the experimental results on the SOMI dataset are presented in Fig. 14. Similar to Fig. 11, the color image of the synthetic scenes are shown in the first row, and their corresponding pose estimation results by Hinterstoisser et al. [16] without and with the ICP, Papazov et al. [48], Drost et al. [33], and our approach are presented in the second to sixth rows respectively. Please note that our approach can handle this multiple instances setting very well. Our approach even reports three true positive detections in the "Pringles" scene, which is very challenging due to the occlusion of the grid in the laundry basket.

When we look at the precision–recall curves in Fig. 15, it is quite clear that our approach significantly outperforms both Hinterstoisser et al. [16] and Papazov et al. [48] and still better than Drost et al. [33] in most cases. The recognition performance of our approach and Drost et al. [33] is similar in "Flash" object. As we saw in the MOSI experiment, Hinterstoisser et al. [16] with the ICP in "Kuka Mug" object shows the recognition performance comparable to our approach. Other than that, the approach suffers from low precision and recall.

In the recognition rates of top N results (Fig. 16), we can notice that the superior performance of our approach in most cases. Since each scene in the SOMI dataset contains up to 10 instances of the tested object, the recognition rates should monotonously increase as N increases. The recognition performance of both our approach and Drost et al. [33] increase as we expect, whereas Hinterstoisser et al. [16] without the ICP and Papazov et al. [48] do not. It implies that Hinterstoisser et al. [16] without the ICP and Papazov et al. [48] are not encouraging in this multiple instances scenario. It is also worth noting that in this SOMI experiment the performance of Papazov et al. [48] is comparable to the performance of Hinterstoisser et al. [16] with the ICP, while it is worse in the MOSI experiment. It is probably due to the multiple instances of the test object. As there are more than one instance, the sampling approach of Papazov et al. [48] may have better chances to have true positive detections.

5.6. Real cluttered scenes

So far, we have evaluated the five pose estimation approaches in synthetic datasets which were generated by rendering 3D mesh models in OpenGL. However, it is necessary to compare the performance of the approaches in real RGB-D scenes which

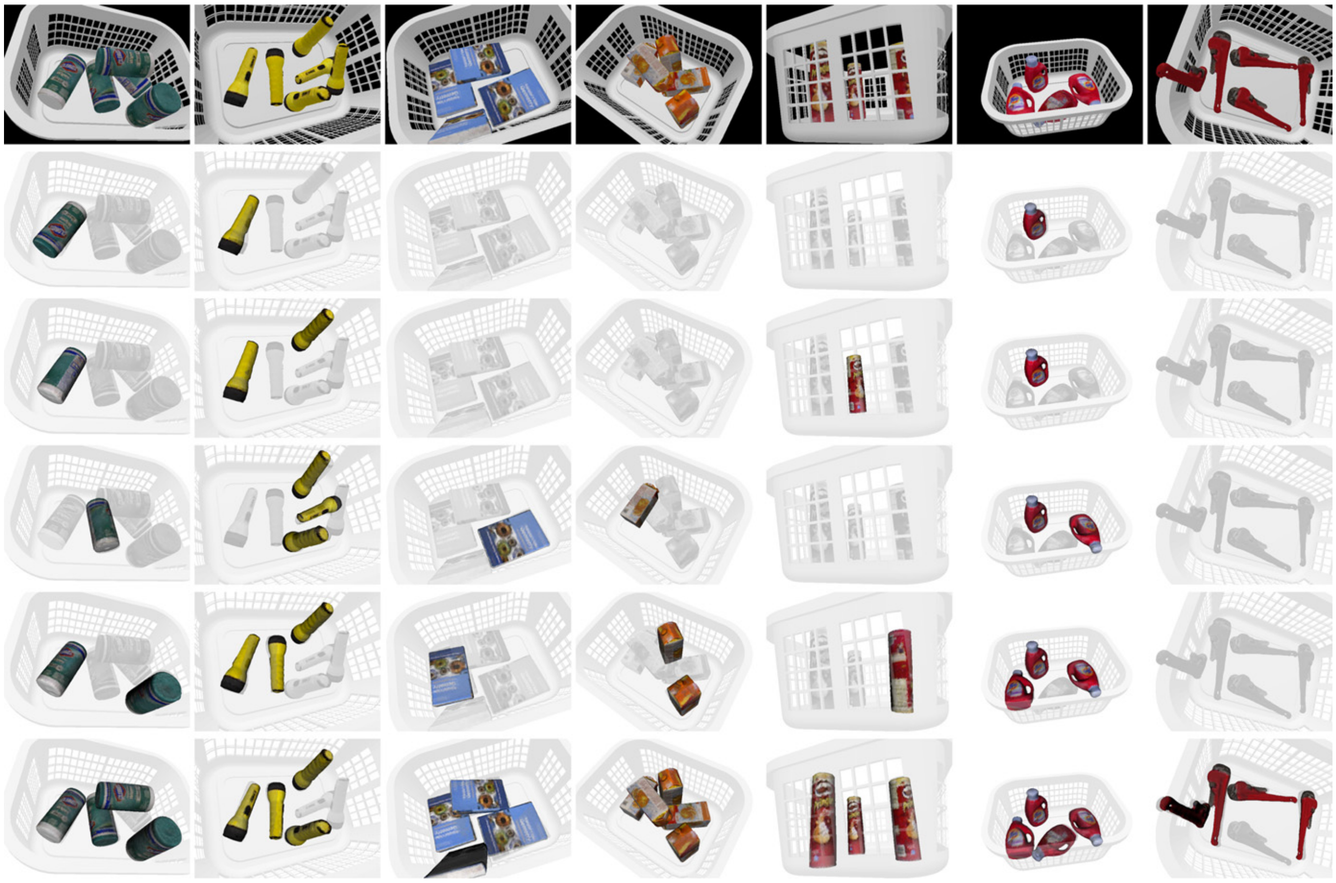


Fig. 14. Selected pose estimation results of Hinterstoisser et al. [16] without the ICP (second row), Hinterstoisser et al. [16] with the ICP (third row), Papazov et al. [48] (fourth row), Drost et al. [33] (fifth row), and our approach (sixth row) in the SOMI dataset. Our approach can handle the multiple instances scenario well, whereas Hinterstoisser et al. [16] and Papazov et al. [48] fail to recognize multiple instances in most cases.

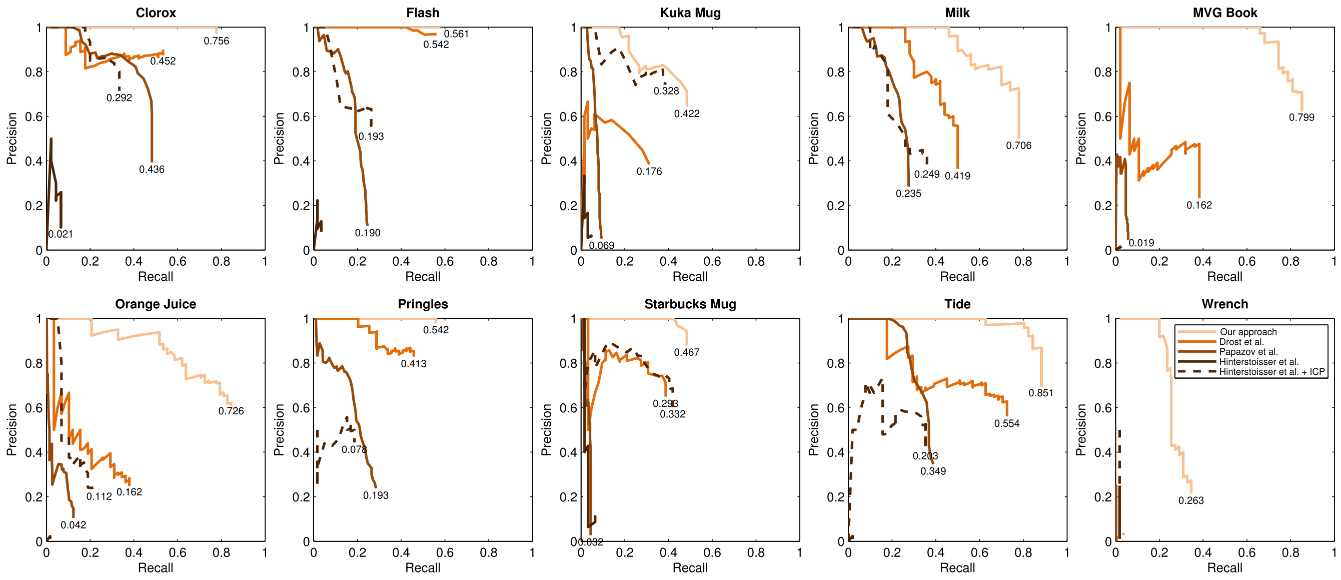


Fig. 15. Precision–recall curves of the single object multiple instances (SOMI) experiment. It is clear that our approach is superior to both Papazov et al. [48] and Hinterstoisser et al. [16] and still better than Drost et al. [33] in general.

contain the real sensor noise as well as background clutter. For the more realistic 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 additionally placed as clutter to make more challenging dataset. 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. 17 is one example. This dataset is much more challenging than the synthetic datasets in the previous sections, since additional clutter and the target objects are superimposed each other. For quantitative evaluation, the ground truth poses of the objects were carefully annotated. To annotate them, we first performed the five pose estimation approaches on

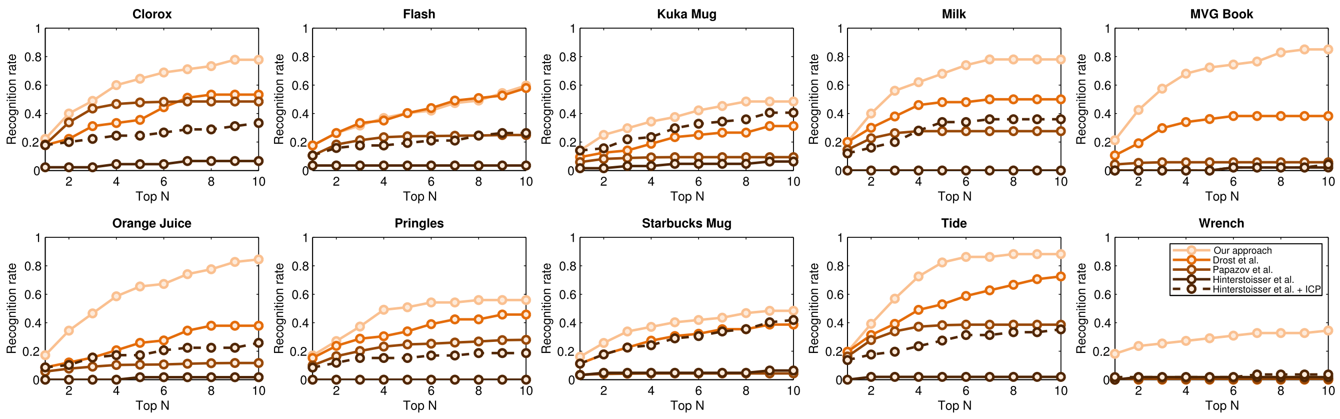


Fig. 16. Recognition rates of top N pose results in the SOMI experiment. As N increases, the recognition rates have to monotonously increase, because each scene has up to 10 instances of each object. While our approach and Drost et al. [33] perform as we expected, Hinterstoisser et al. [16] without the ICP and Papazov et al. [48] fail to increase the recognition rates, implying that they are not suitable for this multiple instances setting.

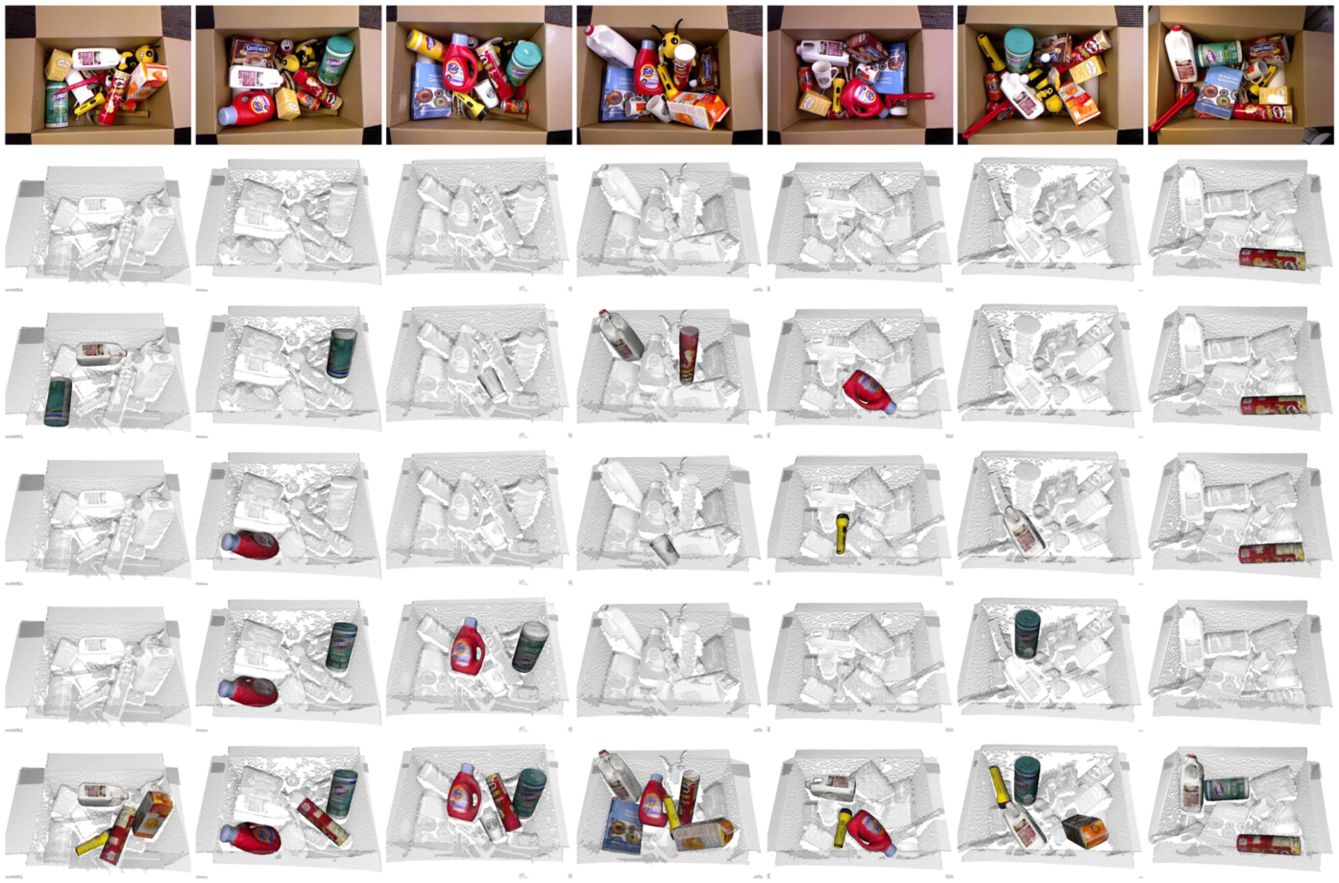


Fig. 17. Selected pose estimation results of Hinterstoisser et al. [16] without the ICP (second row), Hinterstoisser et al. [16] with the ICP (third row), Papazov et al. [48] (fourth row), Drost et al. [33] (fifth row), and our approach (sixth row) in the real cluttered dataset. The first row shows the color image of the scanned RGB-D scenes. In these cluttered scenes, neither of approaches recall more than half of the objects except our approach. Hinterstoisser et al. [16] without the ICP does not detect any objects, whereas it can recall one or two objects with the help of the ICP algorithm. Papazov et al. [48] is also suffered from low recall in the cluttered scenes, and hence it detects at best one object per scene. Drost et al. [33] 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 are more likely to be the true positive poses.

the test scenes and ran the ICP [17] algorithm starting from the estimated pose results. When none of the approaches recognized the object, we manually aligned the corresponding object model to the scene point cloud and then ran the ICP algorithm to obtain the ground truth pose. If the refined pose was close enough to the true pose, we saved it for quantitative analysis. We use the same criterion from the previous experiments for counting true positives.

Fig. 17 shows selected pose estimation results from the 31 scenes. The images in the first row are test scene images captured from an RGB-D camera, and the second to sixth rows represent estimated poses of Hinterstoisser et al. [16] without and with the ICP, Papazov et al. [48], Drost et al. [33], and our approach in the scene point clouds respectively. While Hinterstoisser et al. [16], Papazov et al. [48], and Drost et al. [33] estimate at best two object poses per scene, our approach recalls more than half of the test objects in most scenes. Precision–recall curves of these approaches

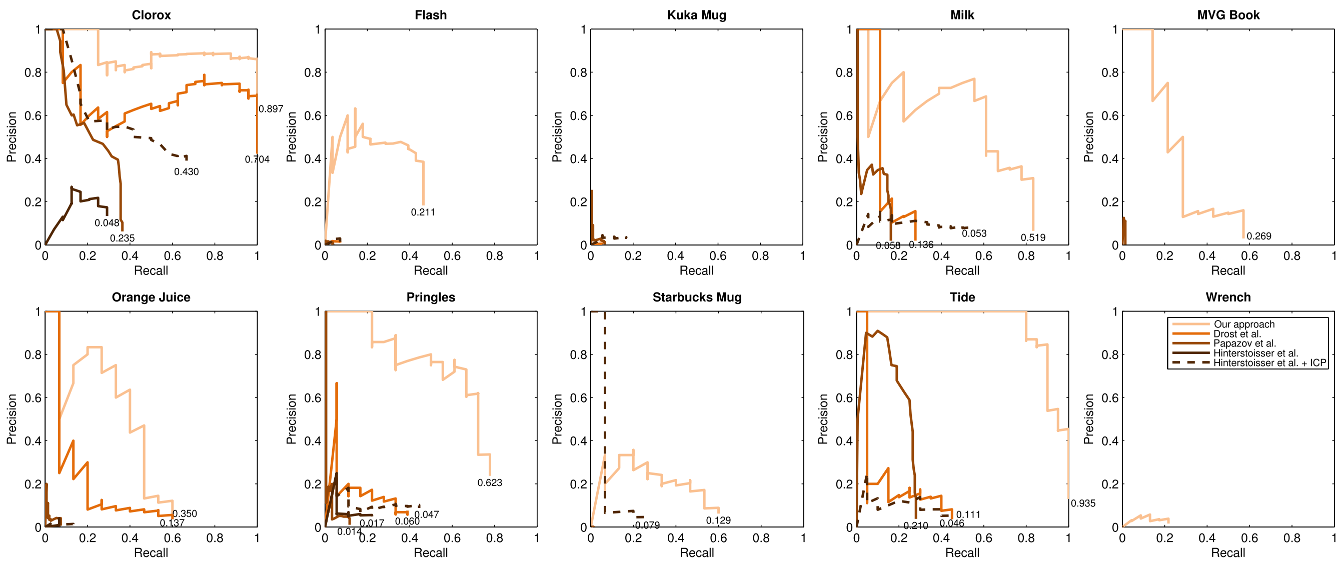


Fig. 18. Precision–recall curves for the real cluttered scene experiments. While our approach reports good precision as well as high recall, other approaches report substandard results in the highly cluttered backgrounds.

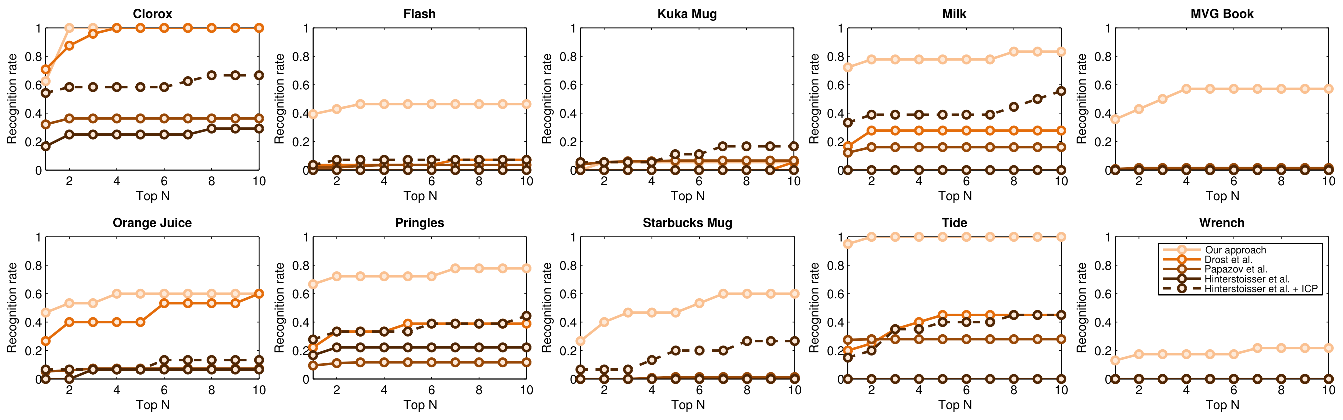


Fig. 19. Recognition rates of top N pose results in the real cluttered scene experiments. Hinterstoisser et al. [16] and Papazov et al. [48] report poor recognition rates, whereas our approach shows outstanding performance in most cases.

are presented in Fig. 18. Whereas the performance of our approach is promising, the performance of both Hinterstoisser et al. [16] and Papazov et al. [48] is extremely not encouraging for these cluttered scenes. Especially, they barely report true positive poses in some objects such as,“Flash”, “Kuka Mug”, “MVG Book”, and “Wrench”. Drost et al. [33] shows better performance than these approaches, yet it is not enough for this cluttered real dataset. In comparison with Fig. 12 showing precision–recall curves for the synthetic MOSI experiment, the performance of five approaches in this dataset is worse because of noise from the real RGB-D sensor and extra clutter. But the overall trend in the performance of the five approaches is quite similar in the sense that our approach exhibits the best performance and Hinterstoisser et al. [16] without the ICP results in the worst performance.

The recognition rates of top N poses on each object is presented in Fig. 19. According to the plots, the recognition rates increase as the considered number of results N increases. In “Clorox” and “Tide”, our approach shows nearly perfect performance as the first or second pose estimates are always true positives. Due to the difficulty in the scenes, the performance of all the approaches is degraded, but our approach still shows outstanding performance compared to the other approaches.

5.7. Computation time

Our approach is not only more accurate but also more efficient. The average processing time of the each approach, which is the amount of time required to estimate the pose of each object per scene, is shown in Table 1. We also report the time of our GPU implementation of Drost et al. [33] to compare the differences between the CPPF and the PPF. The pre-processings in the third and fourth columns of the table include subsampling and normal estimation. Given a model or scene point cloud, it subsamples the given cloud using a voxel grid that averages multiple points in a set of voxels to result in a set of subsampled points. Since we are using the color information, the RGB color attributes are also aggregated. As all approaches, except Hinterstoisser et al. [16] without the ICP, require oriented points,[7] normal estimation is performed on both model and scene point clouds. The pre-processing is running in CPU, and thus the computation time of the pre-processing is nearly the same across the six approaches. For the subsampling, the same voxel size (10 mm) for the 10 test objects was used, so we can notice that the pre-processing time of the each object model increases proportional to the size of the objects; the bigger

[7] Oriented points mean the points having associated surface normal vectors.

Table 1
Average computation time of the approaches on the real dataset.

Object	Approaches[a]	Prep. model	Prep. scene	Model2GPU[d],[c]	Scene2GPU[d]	Main comp.	ICP	Total (s)[b]
Clorox	Our + GPU	0.013 ± 0.001	0.086 ± 0.005	0.035 ± 0.003	0.006 ± 0.002	0.996 ± 0.104	–	$\mathbf{1.134 \pm 0.106}$
	[33] + GPU	0.013 ± 0.002	0.087 ± 0.006	0.029 ± 0.003	0.005 ± 0.001	1.479 ± 0.120	–	1.613 ± 0.124
	[33]	0.013 ± 0.007	0.088 ± 0.008	1.278 ± 0.055	–	26.366 ± 4.865	–	27.745 ± 4.891
	[48]	0.012 ± 0.001	0.088 ± 0.005	–	–	7.369 ± 1.979	–	7.469 ± 1.981
	[16]	–	–	–	–	14.290 ± 4.739	–	14.290 ± 4.739
	[16] + ICP	0.012 ± 0.002	0.087 ± 0.005	–	–	14.818 ± 4.821	0.784 ± 0.692	15.701 ± 4.916
Flash	Our + GPU	0.008 ± 0.001	0.086 ± 0.005	0.018 ± 0.006	0.004 ± 0.001	0.248 ± 0.057	–	$\mathbf{0.363 \pm 0.058}$
	[33] + GPU	0.008 ± 0.001	0.084 ± 0.005	0.017 ± 0.001	0.004 ± 0.000	0.968 ± 0.016	–	1.080 ± 0.016
	[33]	0.008 ± 0.001	0.085 ± 0.004	0.271 ± 0.007	–	4.444 ± 0.531	–	4.807 ± 0.535
	[48]	0.008 ± 0.001	0.086 ± 0.005	–	–	1.624 ± 0.256	–	1.718 ± 0.257
	[16]	–	–	–	–	27.091 ± 5.831	–	27.091 ± 5.831
	[16] + ICP	0.008 ± 0.001	0.086 ± 0.005	–	–	28.229 ± 6.109	0.291 ± 0.174	28.615 ± 6.153
Kuka Mug	Our + GPU	0.008 ± 0.001	0.084 ± 0.004	0.017 ± 0.002	0.004 ± 0.001	0.201 ± 0.022	–	$\mathbf{0.314 \pm 0.024}$
	[33] + GPU	0.008 ± 0.001	0.084 ± 0.005	0.017 ± 0.002	0.004 ± 0.001	0.262 ± 0.023	–	0.374 ± 0.027
	[33]	0.008 ± 0.001	0.084 ± 0.004	0.210 ± 0.004	–	1.426 ± 0.148	–	1.727 ± 0.151
	[48]	0.008 ± 0.001	0.086 ± 0.005	–	–	2.653 ± 0.445	–	2.748 ± 0.445
	[16]	–	–	–	–	27.478 ± 7.147	–	27.478 ± 7.147
	[16] + ICP	0.008 ± 0.002	0.087 ± 0.004	–	–	28.572 ± 7.224	0.245 ± 0.129	28.912 ± 7.232
Milk	Our + GPU	0.011 ± 0.001	0.086 ± 0.005	0.026 ± 0.002	0.004 ± 0.001	0.312 ± 0.063	–	$\mathbf{0.439 \pm 0.066}$
	[33] + GPU	0.013 ± 0.004	0.086 ± 0.006	0.030 ± 0.004	0.005 ± 0.002	0.825 ± 0.075	–	0.960 ± 0.083
	[33]	0.011 ± 0.001	0.084 ± 0.004	3.059 ± 0.049	–	27.447 ± 4.250	–	30.602 ± 4.256
	[48]	0.012 ± 0.001	0.086 ± 0.004	–	–	6.248 ± 1.380	–	6.346 ± 1.382
	[16]	–	–	–	–	13.155 ± 4.102	–	13.155 ± 4.102
	[16] + ICP	0.012 ± 0.002	0.086 ± 0.005	–	–	13.748 ± 4.149	1.130 ± 0.710	14.977 ± 4.253
MVG Book	Our + GPU	0.011 ± 0.001	0.083 ± 0.004	0.025 ± 0.002	0.004 ± 0.001	0.401 ± 0.087	–	$\mathbf{0.524 \pm 0.089}$
	[33] + GPU	0.012 ± 0.001	0.085 ± 0.004	0.024 ± 0.007	0.006 ± 0.014	1.001 ± 0.049	–	1.128 ± 0.057
	[33]	0.011 ± 0.001	0.085 ± 0.004	1.606 ± 0.024	–	39.460 ± 7.268	–	41.161 ± 7.273
	[48]	0.011 ± 0.001	0.086 ± 0.005	–	–	2.890 ± 0.490	–	2.987 ± 0.492
	[16]	–	–	–	–	13.205 ± 3.109	–	13.205 ± 3.109
	[16] + ICP	0.011 ± 0.001	0.087 ± 0.006	–	–	13.744 ± 3.173	0.677 ± 0.508	14.519 ± 3.070
Orange Juice	Our + GPU	0.011 ± 0.001	0.085 ± 0.005	0.025 ± 0.003	0.004 ± 0.001	0.629 ± 0.140	–	$\mathbf{0.753 \pm 0.141}$
	[33] + GPU	0.012 ± 0.002	0.087 ± 0.004	0.030 ± 0.006	0.006 ± 0.002	1.715 ± 0.022	–	1.851 ± 0.023
	[33]	0.011 ± 0.001	0.085 ± 0.005	1.314 ± 0.034	–	25.694 ± 4.768	–	27.104 ± 4.770
	[48]	0.011 ± 0.001	0.084 ± 0.004	–	–	4.856 ± 1.084	–	4.951 ± 1.085
	[16]	–	–	–	–	13.272 ± 3.928	–	13.272 ± 3.928
	[16] + ICP	0.011 ± 0.001	0.086 ± 0.005	–	–	13.892 ± 4.093	1.046 ± 0.585	15.035 ± 4.125
Pringles	Our + GPU	0.010 ± 0.001	0.086 ± 0.004	0.021 ± 0.002	0.004 ± 0.002	0.389 ± 0.059	–	$\mathbf{0.509 \pm 0.062}$
	[33] + GPU	0.011 ± 0.002	0.087 ± 0.005	0.028 ± 0.004	0.006 ± 0.002	1.004 ± 0.017	–	1.135 ± 0.020
	[33]	0.009 ± 0.001	0.084 ± 0.004	0.719 ± 0.019	–	28.450 ± 6.196	–	29.263 ± 6.202
	[48]	0.010 ± 0.001	0.086 ± 0.006	–	–	2.588 ± 0.466	–	2.683 ± 0.468
	[16]	–	–	–	–	15.478 ± 4.329	–	15.478 ± 4.329
	[16] + ICP	0.010 ± 0.001	0.087 ± 0.006	–	–	16.198 ± 4.421	0.557 ± 0.347	16.852 ± 4.548
Starbucks Mug	Our + GPU	0.008 ± 0.001	0.084 ± 0.004	0.018 ± 0.002	0.004 ± 0.001	0.227 ± 0.050	–	$\mathbf{0.341 \pm 0.052}$
	[33] + GPU	0.010 ± 0.002	0.088 ± 0.005	0.020 ± 0.002	0.005 ± 0.001	1.297 ± 0.081	–	1.418 ± 0.084
	[33]	0.008 ± 0.001	0.085 ± 0.004	0.370 ± 0.010	–	3.583 ± 0.382	–	4.046 ± 0.385
	[48]	0.008 ± 0.001	0.085 ± 0.004	–	–	2.230 ± 0.341	–	2.323 ± 0.341
	[16]	–	–	–	–	23.733 ± 6.328	–	23.733 ± 6.328
	[16] + ICP	0.009 ± 0.002	0.087 ± 0.005	–	–	24.676 ± 6.532	0.327 ± 0.154	25.099 ± 6.578
Tide	Our + GPU	0.012 ± 0.001	0.085 ± 0.005	0.031 ± 0.003	0.004 ± 0.001	0.446 ± 0.097	–	$\mathbf{0.578 \pm 0.097}$
	[33] + GPU	0.014 ± 0.002	0.086 ± 0.004	0.027 ± 0.004	0.005 ± 0.002	0.885 ± 0.011	–	1.017 ± 0.016
	[33]	0.012 ± 0.001	0.085 ± 0.004	4.739 ± 0.071	–	30.405 ± 4.256	–	35.241 ± 4.245
	[48]	0.012 ± 0.001	0.086 ± 0.004	–	–	6.432 ± 1.386	–	6.530 ± 1.388
	[16]	–	–	–	–	13.072 ± 4.281	–	13.072 ± 4.281
	[16] + ICP	0.013 ± 0.001	0.087 ± 0.005	–	–	13.714 ± 4.443	1.242 ± 0.676	15.056 ± 4.550
Wrench	Our + GPU	0.008 ± 0.001	0.086 ± 0.005	0.017 ± 0.002	0.004 ± 0.001	0.349 ± 0.047	–	$\mathbf{0.463 \pm 0.051}$
	[33] + GPU	0.009 ± 0.001	0.087 ± 0.005	0.023 ± 0.005	0.005 ± 0.001	1.721 ± 0.026	–	1.845 ± 0.028
	[33]	0.007 ± 0.001	0.086 ± 0.004	0.321 ± 0.011	–	20.017 ± 3.868	–	20.430 ± 3.869
	[48]	0.008 ± 0.001	0.087 ± 0.005	–	–	1.070 ± 0.176	–	1.164 ± 0.178
	[16]	–	–	–	–	30.508 ± 4.373	–	30.508 ± 4.373
	[16] + ICP	0.008 ± 0.001	0.086 ± 0.005	–	–	31.525 ± 4.442	0.211 ± 0.131	31.830 ± 4.483

[a] Our approach is compared with Drost et al. [33], Papazov et al. [48], Hinterstoisser et al. [16] with and without ICP [17].
[b] The most efficient computation time is indicated in **bold** type.
[c] Transfer time of model points to the GPU memory also includes the running time of the object learning algorithm shown in Algorithm 2.
[d] As parallel algorithms on GPU require model and scene data in GPU memory, the transfer times from CPU memory to GPU memory are reported as well.

objects, the more time to pre-process. Since our approach and the implementation of Drost et al. [33] + GPU are parallelized on GPU, we also report the transfer time of model and scene point clouds from CPU memory to GPU memory. In model point clouds, the transfer time includes the running time of the object learning algorithm (Algorithm 2). Please note that the model transfer and the object learning can be run only one time for each object. For Drost et al. [33] without GPU, only the object learning time in CPU is reported. The main computation time is for the rest of the pose estimation procedures. As Hinterstoisser et al. [16] with the ICP

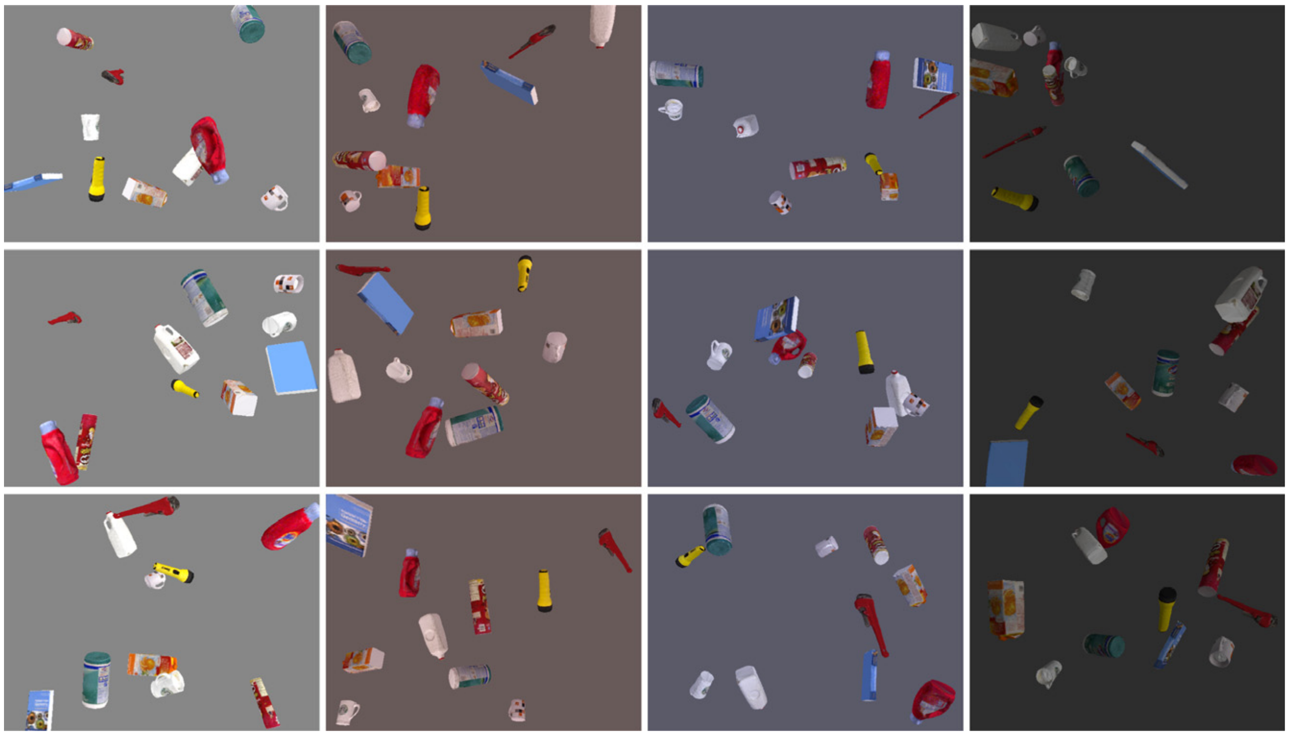


Fig. 20. Example images of the color quantization step learning dataset. The dataset consists of 50 scenes, where object mesh models are rendered with random poses, and ground truth poses. To find a robust quantization step to illumination variations, four different lighting conditions are considered: normal, red, blue, and dark lights. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

employs the ICP algorithm as a post-processing, the ICP time is also reported.

According to the table, the total computation time of our approach is the most efficient. Depending on the kind of objects, the total time varies. But the maximum runtime of our approach is within about 1 s, whereas the running time of Papazov et al. [48] and Hinterstoisser et al. [16] takes up to about 7 and 30 s, respectively, which are not instantaneous for real robotic manipulation tasks.

It is also worth comparing the computation time between our approach and Drost et al. [33]. Drost et al. [33] without GPU acceleration is extremely slow. For instance, it takes more than 40 s in "MVG Book". The computation time depends on the size of model point clouds. For those small object such as "Flash", "Kuka Mug", and "Starbucks Mug", the computation is done in several seconds. But it becomes not scalable as the model size increases. When Drost et al. [33] with GPU is considered, our approach is still about two times faster than Drost et al. [33]. Our algorithm is efficient due to the sparsity of correspondences between model and scene CPPFs. When the object model is learned, similar CPPFs are grouped together by the reduction parallel operation as explained in Section 4.3. Since our CPPF is more discriminative than the PPF of Drost et al. [33], our approach results in a smaller number of duplication for each key. The smaller number of duplication is directly related to the efficiency of the voting stage, as it determines the number of iterations of the for-loop in the 13-th line of Algorithm 3. The efficiency of our approach further comes from the fewer number of correspondences between model and scene CPPFs. Thanks to the color information, a set of mistakenly matched point pairs between model and scene can be ruled out if their color similarities are low, while the voting of Drost et al. [33] cannot help but processing the false matches. This facts enable our approach to be more efficient.

5.8. *Color quantization step learning*

A proper color quantization step

$$\boldsymbol{\sigma} = \begin{pmatrix}\sigma_h & \sigma_s & \sigma_v\end{pmatrix}^\top \in \mathbb{R}^3 \tag{13}$$

is crucial for the good performance of our CPPF-based pose estimation as it plays an important role to quantize the CPPF vector in Eq. (3). The default quantization step

$$\tilde{\boldsymbol{\sigma}} = \begin{pmatrix}0.25 & 0.25 & 1.00\end{pmatrix}^\top \tag{14}$$

works reasonably in general, but it is more preferable if the algorithm learns the parameter from data. In this section, we explain how the color quantization step is learned for each object from a synthetic dataset. To learn the step parameter, we generate a dataset of RGB-D scenes. Fig. 20 shows some example scenes among 50 scenes, where 10 object mesh models were rendered with random poses in front of a gray wall. To find a robust quantization step with respect to illumination variations, four different lighting conditions are considered: normal, red, blue, and dark lights. With the dataset, we run our approach with various quantization steps. We alter σ_h and σ_s in 6 different values: 0.1, 0.2, 0.25, 0.33, 0.5, and 1.0, while σ_v is fixed to 1.0 so that it is invariant to the variation on brightness. The total number of investigated steps for each object is thus $6 \times 6 = 36$, and the performance of each step is measured by AP of precision–recall curve on the dataset. Fig. 21 shows AP values with respect to color quantization steps in hue σ_h and saturation σ_s, where the highest AP value is depicted in its associated step. If there are multiple steps having the highest AP value, we choose the default step $\tilde{\boldsymbol{\sigma}}$ in Eq. (14) as shown in "Flash" and "Pringles".

Not surprisingly, there is no ideal quantization step that works well for all objects. Depending on the color distribution of each object, the best quantization step varies. Some objects are quite insensitive to the steps, such as "Flash", "Pringles", and "Tide". It is also worth noting that too coarse (1.0) and too fine (0.1) steps do

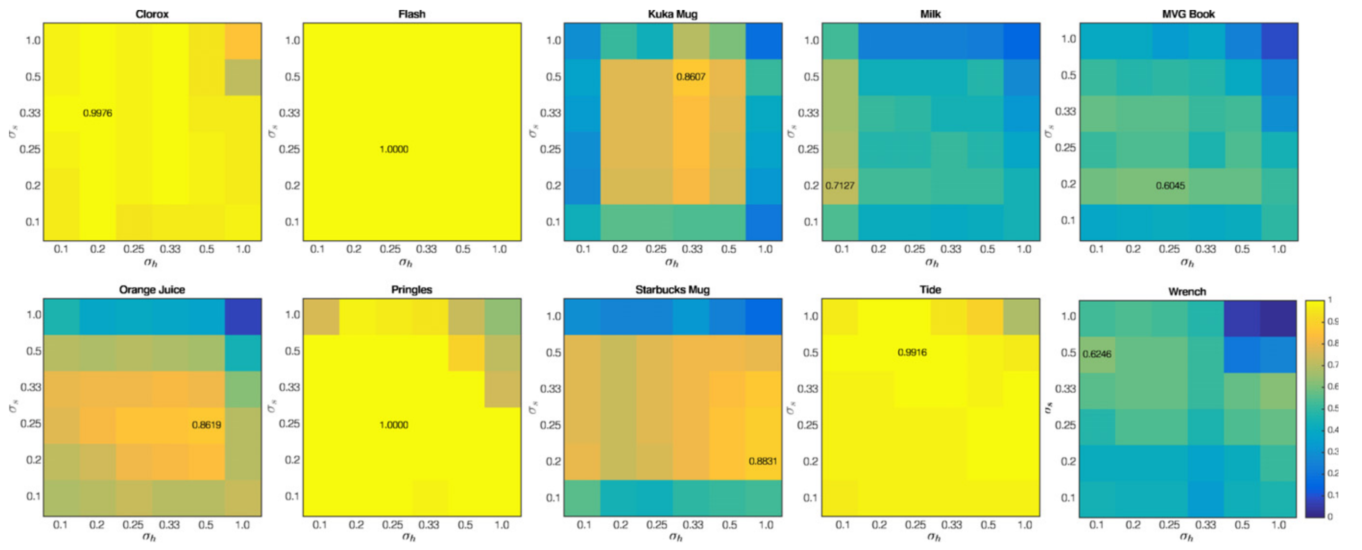


Fig. 21. Average precision (AP) values with respect to color quantization steps σ_h and σ_s. The value of σ_v is fixed to 1.0 for illumination invariance.

not work very well in general. Too coarse step means that color is barely considered in the quantization of the CPPF in Eq. (3). Note that the upper right corner of each plot is equivalent to the performance of the PPF-based pose estimation as 1.0 step means no quantization. If the step is too fine, the voting procedure might be too sensitive to illumination variations and thus results in less reliable pose estimates.

As we mentioned earlier, the upper right corner of each plot in Fig. 21 corresponds to the performance of the PPF approach. From these plots, it is clear that the performance of the PPF-based approach represents the lower bound of the performance of the CPPF-based approach. As we discussed in the previous sections, our CPPF-based approach is more accurate and faster than the PPF-based approach due mainly to the fact that matched pair features are more likely to be correct and sparser.

It is worth noting that even if an object is completely textureless it can still take advantage of the CPPF. The worst case would be the color of the object is identical to that of backgrounds, and in this case the performance of the CPPF will be the same as that of the PPF approach. But if the color of the object is different from that of the backgrounds, our CPPF approach would result in better results as false point pair from the backgrounds will be ignored in the voting phase thanks to the color channel. In that sense, our approach is general as it can be applicable to both textured and textureless objects.

The total time taken for learning the color quantization step for all objects is 22730.89 s, which is about 6 h and 20 min. If faster learning is required, we might employ a hill climbing algorithm starting from the default step in Eq. (14) as the function $AP(\sigma_h, \sigma_s)$ is approximately convex.

6. Conclusions

We presented a voting-based pose estimation algorithm by combining geometric shape and color information from an RGB-D camera. Our color point pair feature (CPPF) was employed in the efficient voting process which can recognize learned objects in highly cluttered environments. As the voting algorithm is inherently possible to be parallelized, we devised a set of highly parallelized algorithms that perform a large amount of repetitive operations in the multi-processors of a GPU. For quantitative evaluations, we generated extensive synthetic and real datasets on which our approach was systematically compared with three other state-of-the-art approaches. Our approach turned out to be more efficient as well as more robust than the compared approaches.

One of the possibilities for future work is to augment our CPPF to be more descriptive by appending additional information, such as principal curvature, boundary edges, etc. Another interesting direction would be to exploit contextual knowledge of environments so that pose estimation approaches adaptively change their algorithmic complexity depending on the scene complexity.

Acknowledgment

This work has in part been sponsored by the Boeing Corporation (12966BC–Wing Assembly). The support is gratefully acknowledged.

References

[1] L.G. Roberts, Machine perception of three-dimensional solids, in: Optical and Electrooptical Information Processing, MIT Press, 1963.
[2] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[3] H. Bay, A. Ess, T. Tuytelaars, L.V. Gool, Sp eeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.
[4] A. Collet, D. Berenson, S.S. Srinivasa, 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.
[5] C. Choi, H.I. Christensen, Robust 3D visual tracking using particle filtering on the special Euclidean group: A combined approach of keypoint and edge features, Int. J. Robot. Res. 31 (4) (2012) 498–519.
[6] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, 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.
[7] D.P. Huttenlocher, G.A. Klanderman, W.A. Rucklidge, Comparing images using the Hausdorff distance, IEEE Trans. Pattern Anal. Mach. Intell. (1993) 850–863.
[8] C. Olson, D. Huttenlocher, Automatic target recognition by matching oriented edge pixels, IEEE Trans. Image Process. 6 (1) (1997) 103–113.
[9] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, Fast directional chamfer matching, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2010, pp. 1696–1703.
[10] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.
[11] R. Raskar, K. Tan, R. Feris, J. Yu, M. Turk, Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging, ACM Trans. Graph. 23 (2004) 679–688.
[12] A. Agrawal, S. Yu, J. Barnwell, R. Raskar, Vision-guided robot system for picking objects by casting shadows, Int. J. Robot. Res. 29 (2–3) (2010) 155–173.
[13] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, A. Agrawal, 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.
[14] I.K. Park, M. Germann, M.D. Breitenstein, H. Pfister, Fast and automatic object pose estimation for range images on the GPU, Mach. Vis. Appl. 21 (5) (2010) 749–766.

[15] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, V. Lepetit, Gradient response maps for real-time detection of texture-less objects, IEEE Trans. Pattern Anal. Mach. Intell. 34 (5) (2012) 876–888.
[16] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, N. Navab, Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes, in: Proceedings of Asian Conference on Computer Vision, ACCV, 2012.
[17] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. (1992) 239–256.
[18] R.B. Rusu, G. Bradski, R. Thibaux, 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.
[19] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, G. Bradski, CAD-mo del recognition and 6DOF pose estimation using 3D cues, in: IEEE International Conference on Computer Vision Workshops, ICCV Workshops, 2011, pp. 585–592.
[20] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano, M. Vincze, Multimod al cue integration through hypotheses verification for RGB-D object recognition and 6DOF pose estimation, in: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2013, pp. 2104–2111.
[21] Z. Xie, A. Singh, J. Uang, K.S. Narayan, P. Abbeel, Multimod al blending for high-accuracy instance recognition, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2013, pp. 2214–2221.
[22] K. Lai, L. Bo, X. Ren, D. Fox, A scalable tree-based approach for joint object and pose recognition, in: AAAI Conference on Artificial Intelligence, 2011.
[23] F. Stein, G. Medioni, Str uctural indexing: Efficient 3-D object recognition, IEEE Trans. Pattern Anal. Mach. Intell. 14 (2) (1992) 125–145.
[24] C. Dorai, A. Jain, COSMOS—a representation scheme for 3D free-form objects, IEEE Trans. Pattern Anal. Mach. Intell. 19 (10) (1997) 1115–1130.
[25] A.E. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3D scenes, IEEE Trans. Pattern Anal. Mach. Intell. 21 (5) (1999) 433–449.
[26] F. Tombari, S. Salti, L. Di Stefano, Unique signatures of histograms for local surface description, in: Computer Vision—ECCV 2010, Springer, 2010, pp. 356–369.
[27] A. Frome, D. Huber, R. Kolluri, T. Bülow, J. Malik, Recogni zing objects in range data using regional point descriptors, in: Proceedings of European Conference on Computer Vision (ECCV), Springer, 2004, pp. 224–237.
[28] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Trans. Pattern Anal. Mach. Intell. (2002) 509–522.
[29] R.B. Rusu, N. Blodow, 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.
[30] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395.
[31] A. Mian, M. Bennamoun, R. Owens, Auto matic correspondence for 3D modeling: an extensive review, Int. J. Shape Model. 11 (2) (2005) 253.
[32] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, 3D object recognition in cluttered scenes with local surface features: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 36 (11) (2014) 2270–2287.
[33] B. Drost, M. Ulrich, N. Navab, 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.
[34] E. Wahl, U. Hillenbrand, 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, 2003, pp. 474–481.
[35] A.S. Mian, M. Bennamoun, R. Owens, Three-di mensional model-based object recognition and segmentation in cluttered scenes, IEEE Trans. Pattern Anal. Mach. Intell. (2006) 1584–1601.
[36] B. Matei, Y. Shan, H.S. Sawhney, Y. Tan, R. Kumar, D. Huber, M. Hebert, Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation, IEEE Trans. Pattern Anal. Mach. Intell. 28 (7) (2006) 1111–1126.
[37] E. Kim, 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.
[38] C. Choi, Y. Taguchi, O. Tuzel, M. Liu, 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.
[39] O.J. Woodford, M.-T. Pham, A. Maki, F. Perbet, B. Stenger, Demisting the Hough transform for 3D shape recognition and registration, Int. J. Comput. Vis. 106 (3) (2014) 332–341.
[40] M. Pham, O. Woodford, F. Perbet, A. Maki, B. Stenger, R. Cipolla, A new distance for scale-invariant 3D shape recognition and registration, in: Proceedings of IEEE International Conference on Computer Vision, ICCV, 2011.
[41] M. Isard, A. Blake, Condensation–conditional density propagation for visual tracking, Int. J. Comput. Vis. 29 (1) (1998) 5–28.
[42] A.S. Montemayor, J.J. Pantrigo, Á. Sánchez, F. Fernández, Particle filter on GPUs for real-time tracking, in: ACM SIGGRAPH 2004 Posters, 2004, p. 94.
[43] P. Azad, D. Munch, T. Asfour, R. Dillmann, 6-DoF model-based tracking of arbitrarily shaped 3D objects, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2011, pp. 5204–5209.
[44] O. Mateo Lozano, K. Otsuka, Real-time visual tracker by stream processing, J. Signal Process. Syst. 57 (2) (2009) 285–295.
[45] C. Choi, H.I. Christensen, RGB-D object tracking: A particle filter approach on GPU, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2013, pp. 1084–1091.
[46] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, A. Davison, SLAM++: Simultaneous localisation and mapping at the level of objects, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2013.
[47] AMD, Bolt C++ template library, 2012.
[48] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, D. Burschka, Rigid 3D geometry matching for grasping of known objects in cluttered scenes, Int. J. Robot. Res. 31 (4) (2012) 538–553.
[49] A.E. Abdel-Hakim, A. Farag, et al., CSIFT: A SIFT descriptor with color invariant characteristics, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, IEEE, 2006, pp. 1978–1983.
[50] K. Pathak, N. Vaskevicius, F. Bungiu, A. Birk, Utilizing color information in 3d scan-registration using planar-patches matching, in: Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), IEEE, 2012, pp. 371–376.
[51] E. Fernandez-Moral, J. Gonzalez-Jimenez, V. Arevalo, A compact planar-patch descriptor based on color, in: 2014 11th International Conference on Informatics in Control, Automation and Robotics, ICINCO, Vol. 02, 2014, pp. 296–302.
[52] F. Tombari, S. Salti, L.D. Stefano, A combined texture-shape descriptor for enhanced 3D feature matching, in: 2011 18th IEEE International Conference on Image Processing (ICIP), IEEE, 2011, pp. 809–812.
[53] C. Choi, H.I. Christensen, 3D pose estimation of daily objects using an RGB-D camera, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2012, pp. 3342–3349.
[54] J. Hoberock, N. Bell, Thrust: A parallel template library, version 1.7.0, 2010.
[55] O. Tuzel, R. Subbarao, P. Meer, Simultan eous multiple 3D motion estimation via mode finding on Lie groups, in: Proceedings of IEEE International Conference on Computer Vision (ICCV), Vol. 1, IEEE, 2005, pp. 18–25.
[56] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of Eurographics Symposium on Geometry Processing, 2006.
[57] M. Fiala, ARTag, a fiducial marker system using digital techniques, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Vol. 2, 2005, pp. 590–596.
[58] Google, Google 3D Warehouse, 2013.
[59] A. Kasper, Z. Xue, R. Dillmann, The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics, Int. J. Robot. Res. 31 (8) (2012) 927–934.
[60] R.B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2011, pp. 1–4.
[61] C. Choi, H. Christensen, 3D textureless object detection and tracking: An edge-based approach, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2012, pp. 3877–3884.
[62] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, 2008.
[63] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.
[64] M. Nieuwenhuisen, D. Droeschel, D. Holz, J. Stückler, A. Berner, J. Li, R. Klein, S. Behnke, Mobile bin picking with an anthropomorphic service robot, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2013.
[65] D. Buchholz, M. Futterlieb, S. Winkelbach, F.M. Wahl, Efficient bin-picking and grasp planning based on depth data, in: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2013, pp. 3245–3250.

Changhyun Choi is a postdoctoral associate in the Computer Science & Artificial Intelligence Lab (CSAIL) at Massachusetts Institute of Technology (MIT) working with Prof. Daniela Rus. He obtained a Ph.D. in Robotics at the School of Interactive Computing, College of Computing, Georgia Institute of Technology, wherein he was also affiliated with the Institute for Robotics and Intelligent Machines (IRIM). He was a research intern in the Imaging Group of Mitsubishi Electric Research Labs (MERL) in Cambridge, Massachusetts, an intern researcher at the Imaging Media Research Center (IMRC) at Korea Institute of Science and Technology (KIST), and an undergraduate researcher at the Intelligent Systems Research Center at Sungkyunkwan University. He holds a B.S. in Electrical and Computer Engineering from Sungkyunkwan University. His broad research interests are in visual perception for robotics, with a focus on object recognition and pose estimation, visual tracking, and 3D registration.

Henrik I. Christensen is the KUKA Chair of Robotics at the College of Computing Georgia Institute of Technology. He is also the executive director of the Institute for Robotics and Intelligent Machines (IRIM). Dr. Christensen does research on systems integration, human–robot interaction, mapping and robot vision. The research is performed within the Cognitive Robotics Laboratory. He has published more than 300 contributions across AI, robotics and vision. His research has a strong emphasis on "real problems with real solutions". A problem needs a theoretical model, implementation, evaluation, and translation to the real world. He is actively engaged in the setup and coordination of robotics research in the US (and worldwide). Dr. Christensen received the Engelberger Award 2011, the highest honor awarded by the robotics industry. He was also awarded the "Boeing Supplier of the Year 2012" with 3 other colleagues at Georgia Tech. Dr. Christensen is a fellow of American Association for Advancement of Science (AAAS) and Institute of Electrical and Electronic Engineers (IEEE). He received an honorary doctorate in engineering from Aalborg University 2014. He collaborates with institutions and industries across three continents. His research has been featured in major media such as CNN, NY Times, BBC, … He serves as a consultant to companies and government agencies across the world.