Going Further with Point Pair Features

Stefan Hinterstoisser[1(✉)], Vincent Lepetit[2], Naresh Rajkumar[1], and Kurt Konolige[1]

[1] Google, Mountain View, USA
hinterst@google.com, nareshkumar@google.com, konolige@google.com
[2] TU-Graz, Graz, Austria
lepetit@icg.tugraz.at

Abstract. Point Pair Features is a widely used method to detect 3D objects in point clouds, however they are prone to fail in presence of sensor noise and background clutter. We introduce novel sampling and voting schemes that significantly reduces the influence of clutter and sensor noise. Our experiments show that with our improvements, PPFs become competitive against state-of-the-art methods as it outperforms them on several objects from challenging benchmarks, at a low computational cost.

1 Introduction

Object instance recognition and 3D pose estimation have received a lot of attention recently, probably because of their importance in robotics applications [2,4,7,8,11,15,20,25]. For grasping and manipulation tasks, efficiency, reliability, and accuracy are all desirable properties that are still very challenging in general environments.

While many approaches have been developed over the last years, we focus here on the approach from Drost et al. [8], which relies on a depth camera. Since its publication, it has been improved and extended by many authors [2,5,6,14,22]. However, we believe it has not delivered its full potential yet: Drost's technique and its variants are based on votes from pairs of 3D points, but the sampling of these pairs has been overlooked so far. As a result, these techniques are very inefficient, and it usually still takes several seconds to run them.

Moreover, this approach is also very sensitive to sensor noise and 3D background clutter—especially if it is close to the target object: Sensor noise can disturb the quantization on which the approach relies heavily for fast accesses. Background clutter casts spurious votes that can mask the effects of correct ones. As a result, several other approaches [4,11,15,20] have shown significantly better performance on recent datasets [11,15].

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-46487-9_51) contains supplementary material, which is available to authorized users.

B. Leibe et al. (Eds.): ECCV 2016, Part III, LNCS 9907, pp. 834–848, 2016.
DOI: 10.1007/978-3-319-46487-9_51

Fig. 1. Several 3D objects are simultaneously detected with our method under different poses on cluttered background with partial occlusion and illumination changes. Each detected object is augmented with its 3D model, its 3D bounding box and its coordinate systems. For better visibility, the background is kept in gray and only detected objects are in color. (Color figure online)

In this paper, we propose a much better and efficient sampling strategy that, together with small modifications to the pre- and post-processing steps, makes our approach competitive against state-of-the-art methods: It beats them on several objects on recent challenging datasets, at a low computational cost.

In the remainder of this paper, we review related work, and describe in detail the method of Drost et al. [8]. We then introduce our contributions, which we compare against state-of-the-art methods in the last section (Fig. 1).

2 Related Work

The literature on object detection and 3D pose estimation is very broad. We focus here on recent work only and split them in several categories.

Sparse Feature-Based Methods. While popular for 3D object detection in color or intensity images several years ago, these methods are less popular now as practical robotics applications often consider objects that do not exhibit many stable feature points due to the lack of texture.

Template-Based Methods. Several methods are based on templates [11,20,26], where the templates capture the different appearances of objects under different viewpoints. An object is detected when a template matches the image and its 3D pose is given by the template. [11] uses synthetic renderings of a 3D object model to generate a large number of templates covering the full view hemisphere.

It employs an edge-based distance metric which works well for textureless objects, and refines the pose estimates using ICP to achieve an accurate 6D pose. Such template-based approaches can work accurately and quickly in practice. However, they show typical problems such as not being robust to clutter and occlusions.

Local Patch-Based Methods. [4,23] use forest-based voting schemes on local patches to detect and estimate 3D poses. While the former regresses object coordinates and conducts a subsequent energy-based pose estimation, the latter bases its voting on a scale-invariant patch representation and returns location and pose simultaneously. [3] also uses Random Forests to infer object and pose, but via a sliding window through a depth volume. In the experiment section, we compare our method against the recent approach of [4], which performs very well.

Point-Cloud-Based Methods. Detection of 3D objects in point cloud data has a very long history. A review can be found in [17]. One of the standard approaches for object pose estimation is ICP [16], which however requires an initial estimate and is not suited for object detection. Approaches based on 3D features are more suitable and are usually followed by ICP for the pose refinement. These methods include point pairs [8,16], spin-images [13], and point-pair histograms [21,24]. These methods are usually computationally expensive, and have difficulty in scenes with heavy clutter. However, we show in this paper that these drawbacks can be avoided.

The point cloud-based method proposed in [8] is the starting point of our own method, and we detail it in the next section.

3 "Drost-PPF" [8]

One of the most promising algorithms based on point clouds for matching 3D models to 3D scenes was proposed by Drost et al. [8]—in the rest of the paper, we will refer to it as *Drost-PPF*. Since our approach is based on it, we will describe it here in detail.

The authors of Drost-PPF coupled the existing idea of Point Pair Features (PPF) with a voting scheme to solve for the object pose and location simultaneously. More specifically, the goal of Drost-PPF is to establish for each scene point a correspondence to a model point and solve for the remaining degree of freedom to align the scene with the model data. This is done by using the relations between the corresponding points and all their neighboring points in model and scene space to vote in a Hough transform fashion for both unknowns.

Drost-PPF begins by extracting pairs of 3D points and their normals from the object's 3D model, to compute Point Pair Features (PPF) defined as a 4-vector:

$$\mathbf{F}(\mathbf{m}_1, \mathbf{m}_2, \mathbf{n}_1, \mathbf{n}_2) = [\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)]^\top , \tag{1}$$

where $\mathbf{m}_1$ and $\mathbf{m}_2$ are two 3D points and $\mathbf{n}_1$ and $\mathbf{n}_2$ their normals, $\mathbf{d} = \mathbf{m}_2 - \mathbf{m}_1$, and $\angle(\mathbf{a}, \mathbf{b})$ the angle between vectors $\mathbf{a}$ and $\mathbf{b}$. This feature vector is invariant to rigid motions, which allows the method to detect the objects under these transformations.

The PPFs are discretized and used as indices of a lookup table. Each bin of this lookup table stores the list of first model points $\mathbf{m}_i$ and the corresponding rotation angles ${\alpha_i}^{model}$ of all the PPFs that are discretized to the bin's index. In this context, ${\alpha_i}^{model}$ describes the rotation angle around the normal of the first model point $\mathbf{m}_i$ that aligns the corresponding point pair that is descretized to the bin's index to a fixed canonical frame—as described in [8]. The stored data will be used during the voting stage.

At run-time, Drost-PPF tries to establish for each scene point a correspondence to a model point. This is achieved by pairing each scene point with all other scene points, compute the corresponding PPFs and match them with similar PPFs computed from 3D points on the target objects. The latter step can be performed efficiently using the lookup table: The content of the bins of the lookup table indexed by the discretzed PPFs are used to vote for pairs made of model points and discretized rotation angles $\alpha_i = {\alpha_i}^{model} - \alpha^{scene}$ in a Hough transform fashion. In this context, α^{scene} is the rotation around the normal of the first scene point that aligns the current scene point pair with the fixed canonical frame. Once a peak in the Hough space is extracted, the 3D pose of the object can be easily estimated from the extracted scene and model point correspondence and its corresponding rotation α_i.

[22] used this method to constrain a SLAM system by detecting multiple repetitive object models. They devised a strategy towards an efficient GPU implementation. Another immediate industrial application is bin picking, where multiple instances of the CAD model is sought in a pile of objects [12]. The approach has also been used in robotic applications [1,19].

In addition, several extensions to [8] have been proposed. A majority of these works focused on augmenting the feature description to incorporate color [5] or visibility context [14]. [6] proposed using points or boundaries to exploit the same framework in order to match planar industrial objects. [7] modified the pair description to include image gradient information. There are also attempts to boost the accuracy and performance of the matching, without modifying the features. [9] made use of the special symmetric object properties to speed up the detection by reducing the hash-table size. [25] proposed a scene specific weighted voting method by learning the distinctiveness of the features as well as the model points using a structured SVM.

Drost-PPF has often been criticized for the high dimensionality of the search space [4], for its inefficiency [11], for being sensitive to 3D background clutter and sensor noise [18]. Furthermore, other approaches significantly outperform it on many datasets [11,15].

In the next section, we discuss in greater detail the shortcomings of [8] and propose several suitable feature sampling strategies that allow it to outperform all state-of-the-art methods [4,11,15,20] on the standard dataset of [11] and the very challenging occlusion dataset of [15], in terms of recognition rate. In addition, we show how we can speed up the approach to be significant faster than [8].

4 Method

We describe here our contributions to make PPFs more discriminative, and more robust to background clutter and sensor noise. We evaluate the improvement provided by each of these contributions, and compare their combinations against state-of-the-art methods in the next section.

4.1 Pre-processing of the 3D Models and the Input Scene

During a pre-processing stage, Drost-PPF subsamples the 3D points of the target objects and the input scene. The advantage is two-fold: This speeds up the further computations and avoids considering too many ambiguous point pairs: Points that are close to each other tend to have similar normals, and generate many non-discriminative PPFs. Drost-PPF therefore subsamples the points so that two 3D points have at least a chosen minimal distance to each other.

This however can lead to a loss of useful information when normals are actually different. We therefore keep pairs even with a distance smaller than the minimal distance if the angle between the normals is larger than 30 degrees, as these pairs are likely to be discriminative. Subsampling is then done as in Drost-PPF, but with this additional constraint.

4.2 Smart Sampling of Point Pairs

After sub-sampling, in Drost-PPF, every scene point is paired with every other scene point during runtime. The complexity is therefore quadratic in the number of points in the 3D scene. In order to reduce computation time, [8] suggests using only every m-th scene point as the first point, where m is often set to 5 in practice. While this improves runtime, the complexity remains quadratic and matching performance suffers because we remove information from the already sampled scene point cloud.

We propose a better way to speed up the computations without discarding scene points: Given a first point from the scene it should be only paired with other scene points that can belong to the same object. For example, if the distance between the two points is larger than the size of the object, we know that these two points cannot possibly belong to the same object and therefore should not be paired. We show below that this leads to a method that can be implemented much more efficiently.

A conservative way to do so would be to ignore any point that is farther away than d_{obj} from the first point of a pair, where d_{obj} be the diameter of the enclosing sphere of the target object, which defines a voting ball.

However, a spherical region can be a very bad approximation for some objects. In particular, with narrow elongated objects, sampling from a sphere with radius d_{obj} will generate many points on the background clutter if the object is observed in a viewing direction parallel to its longest dimension, as depicted in Fig. 2.

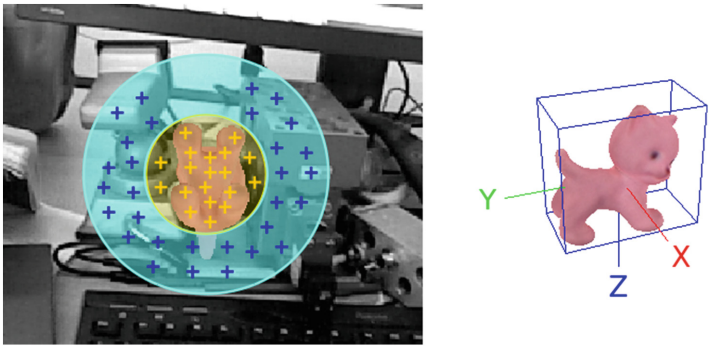

Fig. 2. Different voting spheres for a given first point projected into 2D. The yellow circle corresponds to a voting sphere with the smallest dimension of the object's bounding box as diameter. The blue circle corresponds to the voting sphere with the same diameter as the enclosing sphere. Points sampled in the yellow sphere are much more likely to lie on the object than points sampled in the blue sphere. (Color figure online)

In these cases we would like to use a smaller sampling volume where the ratio of scene points lying on the object compared to all other scene points is larger. However, we do not have any prior information on the pose of the object and the first scene point of the pair can lie anywhere on the object. It is therefore impossible to define a single volume that is smaller than the ball of radius d_{obj} without discarding pairs of scene points under certain object configurations that both lie on the target object.

We therefore opted for using consecutively two voting balls with different radiuses: A small one with radius $R_{\text{min}} = \sqrt{d_{\text{min}}^2 + d_{\text{med}}^2}$, where d_{min} is the smallest dimension of the object's bounding box and d_{med} is the median of its three dimensions, and the large conservative one with radius $R_{\text{max}} = d_{\text{obj}}$. It can be easily seen that R_{min} is the smallest observable expansion of the object. We will say that a point pair is accepted by a voting ball if the first point is at the center of the ball and its distance to the second point is smaller than the radius of the ball.

We first populate the accumulator with votes from pairs that are accepted by the small ball. We extract the peaks from the accumulator, which each corresponds to a hypothesis on the object's 3D pose and correspondence between a model and scene point, as in Drost-PPF. We then continue populating the accumulator with votes from pairs accepted by the large ball but were rejected by the small one. We proceed as before and extract the peaks to generate pose and point correspondence hypotheses. This way, under poses such as the one illustrated by Fig. 2, we can get peaks that are less polluted by background clutter during the first pass, and still get peaks for the other configurations during the second pass.

To efficiently look for pairs accepted by a ball of radius d, we use a spatial lookup table: For a given scene, we build a voxel grid filled with the scene points.

The number of voxels in each dimension is adapted to the scene points and can differ in x, y, and z dimensions. Each voxel in this grid has size d, and stores the indices of the scene points that lie in it. Reciprocally, for each scene point we also store the voxel index to which it belongs. Building up this voxel grid is a $O(n)$ operation. In order to extract for a given first point all other scene points that are maximally d away, we first look up the voxel the scene reference point belongs to and extract all the scene point indices stored in this voxel and in all adjacent voxels. We check each of these scene points if its distance to the scene reference point is actually smaller or equal than d.

The complexity of this method is therefore $O(nk)$ where k is usually at least one magnitude smaller than n, compared to the quadratic complexity of Drost-PPF, while guaranteeing that all relevant point pairs are considered.

4.3 Accounting for Sensor Noise When Voting

For fast access, the PPFs are discretized. However, sensor noise can change the discretization bin, preventing some PPFs from being correctly matched. We overcome this problem by spreading the content of the look-up table during the pre-processing of the model. Instead of storing the first model point and the rotation angles only in the bin indexed by the discretized PPF vector, we also store them in the (80) neighborhood bins indexed by the adjacent discretized PPFs (there are $3^4 = 81 - 1$ adjacent bins).

We face a similar problem at run-time during voting for the quantized rotation angles around the point normals. To overcome this, we use the same strategy as above and vote not only for the original quantized rotation angle but also for its adjacent neighbors.

However, as shown in Fig. 3, spreading also has a drawback: Because of discretization and spreading in the feature+rotation space, it is very likely that pairs made of close scene points have the same quantized rotation angle and are mapped to the same look-up table bin. They will thus vote for the same bin in the accumulator space, introducing a bias in the votes.

A direct method to avoid multiple votes would be to use a 3D binary array $(a_{i,j,k})$ for each scene point to "flag" if a vote with the i-th model point, j-th model point as first and second point respectively, and k-th quantized rotation angle around the normal has already been cast, and prevent additional votes for this combination.

Unfortunately, such an array would be very large as its size is quadratic with the number of model points, and we would need to create one for each of the scene point. We propose a much more tractable solution.

Instead of indexing a flag array by the pair of model points and corresponding rotation, we use an array b indexed by the quantized PPFs the point pairs generate. Each element of b is a 32-bit integer initialized to 0, and each bit corresponds to a discretized rotation angle α^{scene}. We simply set the bit corresponding to a quantized PPF and scene rotation to 1 the first time it votes, and to prevent further voting for the same combination of quantized PPF and scene rotation, even if it is generated again by discretization and spreading. Spreading

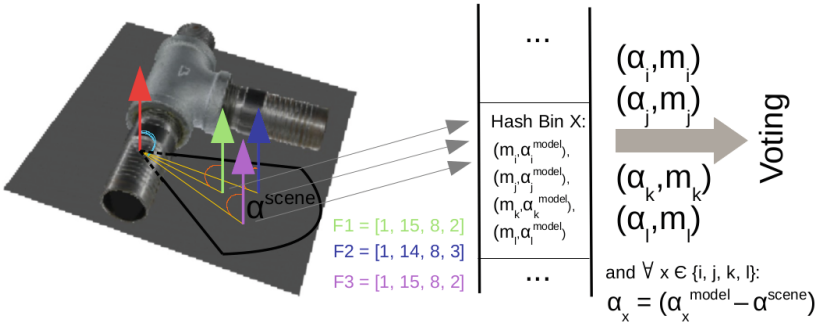

Fig. 3. A first scene point, in red, and three other scene points (green, blue and purple) build three point pair features. Because of discretization and spreading, all three point pair features correspond to the same scene rotation angle α^{scene}. Furthermore, feature discretization of F1 and F3 and spreading of F2 might lead to all three features being mapped to the same hash bin X which results in voting for the same combinations of model reference points and rotation angles $(\mathsf{m_i}, \alpha_i)$, $(\mathsf{m_j}, \alpha_j)$, $(\mathsf{m_k}, \alpha_k)$, $(\mathsf{m_l}, \alpha_l)$ several times. This artificially increases the voting for these specific combinations of model points and rotation angles and deteriorates the performance, especially in cases as shown in this figure where votes come from background. (Color figure online)

is achieved by generating for each discretized PPF created at run-time from the scene points the adjacent discretized PPFs, and treating them as the original discretized PPFs.

Note that we use here the scene rotations around the normals since it is only dependent on the scene point pair and not on the model points as shown in [8]. Thus, it is the same for all elements stored in one bin in the look-up table which allows us to leverage it to perform flagging in b.

This solution is more efficient than the direct method discussed above, as the number of possible entries is smaller in practice, thanks to the quantization of the PPFs. b is constant in size for all objects and scales linearly with the number of all possible quantizations of the PPFs. In practice, we quantize each angle and the distance of Eq. (1) into 22 and 40 bins respectively, yielding to $22^3 \times 40$ possible quantized PPFs. In our implementation, when the number of model points exceeds 650, the direct method takes significantly more time, with a typical slow-down of factor 3 for 1000 model points.

4.4 Generating Object Pose and Postprocessing

To extract object poses from the accumulator, Drost-PPF uses a greedy clustering approach. They extract the peaks from the accumulator, each corresponding to a hypothesis on the object's 3D pose and correspondence between a model and scene point, and process them in the same order as their numbers of votes, and assign them to the closest cluster if it is close enough, or otherwise create another cluster.

We found that this method is not always reliable especially in case of noisy sensors and background clutter. These result in spurious votes and the number of votes in the accumulator space is not necessarily a reliable indicator for the quality of a hypothesis.

Therefore, we propose a different cluster strategy that takes into account our voting strategy. We perform a bottom-up clustering of the pose hypotheses generated during voting with one voting ball. We allow hypotheses to join several clusters as long as their poses are similar to the one of the cluster center. We also keep track of the model points associated with each hypothesis and only allow a hypothesis to vote for a cluster if no other hypothesis with the same model point has voted for this cluster before. Thus, we avoid that ambiguous and repetitive geometric structures such as planar surfaces introduce biases.

For each of the few first clusters with the largest weights, we refine the estimated pose using projective ICP [10]. In practice, we consider the four first clusters for each of the two voting balls.

To reject the clusters that do not actually correspond to an object, we render the object according to the corresponding 3D pose, and count how many pixels have a depth close to the one of the rendering, how many are further away from the camera—and could be occluded, and how many are closer—and are therefore not consistent with the rendering. If the number of pixels that are closer is too large compared to the total number of pixels, we reject the cluster.

In practice, this threshold is set to 10 %. We also discard objects which are too much occluded. As a last check, we compute areas with significant depth or normal change in the scene, and compare them to the silhouette of the projected object: If the silhouette is not covered enough by depth or normal change, we discard the match. In practice, we use the same threshold that we use for occlusion check. We finally rank all remaining clusters according to how well they fit the scene points and return the pose of the best one only, or in case of multi-instance detection, the whole list of poses from all remaining clusters.

5 Experimental Results

We compare here our method against Drost-PPF, Linemod [11], DTTs [20], Birdal and Ilic [2], Bachmann et al. [4], and Krull et al. [15] on the ACCV dataset of [11] and on the Occlusion Dataset of [15].

For our method, we use the same parameters for all the experiments and objects, except for the post processing thresholds to account for the specificities of the occlusion dataset.

5.1 ACCV Dataset of [11]

We first tested our method on the standard benchmark from [11]. This dataset contains 16 objects with over 1100 depth and color images each, and provides the ground truth 3D poses for each object. We only evaluate our method for the non-ambiguous objects —we removed the bowl and the cup— because state-of-the-art approaches considered only these objects.

Table 1. Recognition rates according to the evaluation metric of [11] for different methods. We perform best for eight out of thirteen objects, while [4,11,20] use color data in addition to depth, and we only use depth data.

Approach	Our Appr	Linemod [11]	Drost et al. [8]	DTT [20]	Brachmann et al. [4]	Birdal and Ilic [2]
Sequence (#pics)			Matching score			
Ape (1235)	**98.5 %**	95.8 %	86.5 %	95.0 %	85.4 %	81.95 %
Bench V. (1214)	**99.8 %**	98.7 %	70.7 %	98.9 %	98.9 %	–
Cam (1200)	**99.3 %**	97.5 %	78.6 %	98.2 %	92.1 %	91.00 %
Can (1195)	**98.7 %**	95.4 %	80.2 %	96.3 %	84.4 %	–
Cat (1178)	**99.9 %**	99.3 %	85.4 %	99.1 %	90.6 %	95.76 %
Driller (1187)	93.4 %	93.6 %	87.3 %	94.3 %	**99.7 %**	81.22 %
Duck (1253)	**98.2 %**	95.9 %	46.0 %	94.2 %	92.7 %	–
Box (1253)	98.8 %	**99.8 %**	97.0 %	**99.8 %**	91.1 %	–
Glue (1219)	75.4 %	91.8 %	57.2 %	**96.3 %**	87.9 %	–
Hole P. (1236)	**98.1 %**	95.9 %	77.4 %	97.5 %	97.9 %	–
Iron (1151)	98.3 %	97.5 %	84.9 %	98.4 %	**98.8 %**	93.92 %
Lamp (1226)	96.0 %	97.7 %	93.3 %	**97.9 %**	97.6 %	–
Phone (1224)	**98.6 %**	93.3 %	80.7 %	88.3 %	86.1 %	–

[2] does not evaluate all objects and does not use refinement with ICP. For the approach of Bachmann et al. [4], we use the numbers reported for synthetic training without a ground plane, since adding a ground plane during training artificially adds additional knowledge about the test set.

Like [4,11,20] we only use features that are visible from the upper hemisphere of the object. However, differently to these methods and similarly to [2,8] we still allow the object to be detected in any arbitrary pose. Thus, we are solving for a much larger search space than [4,11,20].

In addition, as in [2,8], we only use the depth information and do not make use of the color information as [4,11,20]. However, as shown in Table 1, we perform best on eight objects out of thirteen.

5.2 Occlusion Dataset of [15]

The dataset of [11] is almost free of occlusion. Hence, to demonstrate robustness with respect to occlusions, we tested our method on the occlusion dataset of [15]. It is much more noisy and includes more background clutter than previous occlusion datasets [16]. It includes over 1200 real depth and color images with 8 objects and their ground truth 3D poses.

As Table 2 shows, our approach performs better for five objects while the method of Krull et al. [15] performs better on three objects. On average we perform 3.3 % better than Krull that was the state-of-the-art on this dataset.

Table 2. Recognition rates on the occlusion dataset [15] according to the evaluation metric of [15]. Our approach performs better for five objects out of eight. On average we perform 3.3 % better than Krull et al. [15], while Krull et al. [15] uses color data in addition to depth and a ground plane during training. We only use depth data.

Approach	Our approach	Linemod [11]	Brachmann et al. [4]	Krull et al. [15]
Sequence (#pics)		Matching score		
Ape (952)	**81.4 %**	49.8 %	62.6 %	77.9 %
Can (1143)	**94.7 %**	51.2 %	80.2 %	86.6 %
Cat (655)	55.2 %	34.9 %	50.0 %	**55.6 %**
Driller (1044)	86.0 %	59.6 %	84.3 %	**93.6 %**
Duck (911)	**79.7 %**	65.1 %	67.6 %	71.9 %
Box (770)	**65.6 %**	39.6 %	8.5 %	35.6 %
Glue (462)	52.1 %	23.3 %	62.8 %	**67.9 %**
Hole Puncher (1156)	**95.5 %**	67.2 %	63.2 %	94.8 %
Average	**76.3 %**	54.4 %	67.3 %	73.0 %

While we only use depth data, [15] also uses color data. Moreover, the method in [15] was trained with an added a ground plane during training[1], which gives an extra advantage.

5.3 Computation Times

Our approach takes between $0.1s$ and $0.8s$ to process a 640×480 depth map. Like Drost-PPF, the sub-sampling during pre-processing depends on the object diameter; denser sampling is used for smaller objects, which increases the processing time.

On average it is about 6 times faster than Drost-PPF, while being significantly more accurate. Moreover, our method could be implemented on GPU, as [22] did, for further acceleration.

5.4 Worst Case Runtime Discussion

In this section we discuss the worst case runtime behavior. Despite our smart sampling strategy presented in Sect. 4.2, the worst case runtime behavior is still $O(n^2)$ where n is the number of subsampled points in a scene. However, this is not a problem in practice as this happens only if all observed scene points lie in a sphere with radius $\sqrt{3} \times d_{obj}$ ($\sqrt{3}$ comes from taking the diagonal of voxels of width d_{obj} into account). This is because our spatial look-up table only gives back points that fall into the same voxel or adjacent voxels of the spatial look-up table as our candidate point falls into. When there are points outside

[1] Private email exchange with the authors.

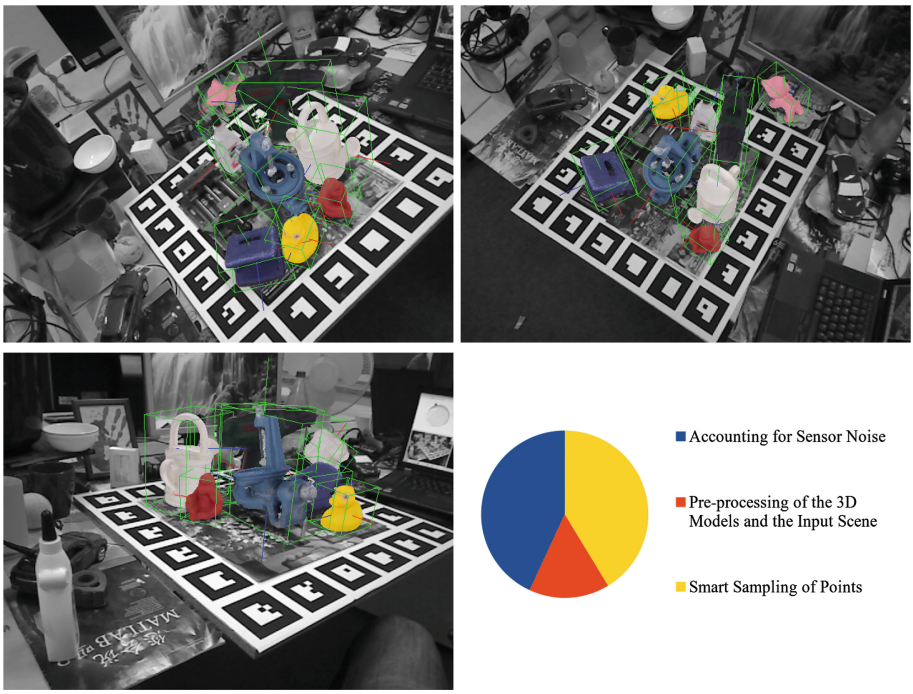

Fig. 4. Three images of the dataset of [15] with at least seven out of nine objects correctly detected. Note the strong background clutter and heavy occlusion. Lower Right: Contribution of each proposed step to the matching scores compared to [8]. Accounting for sensor noise and our sampling of points are almost equal, while our proposed pre-processing of the 3D model and the input scene has the smallest but still significant effect.

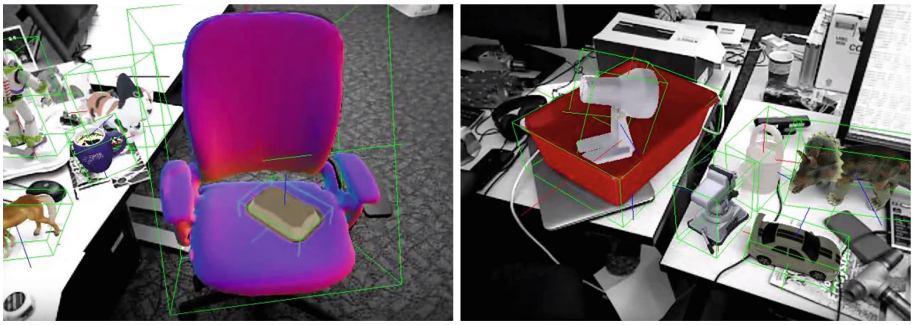

Fig. 5. Several 3D objects are simultaneously detected with our method under different poses on cluttered background with partial occlusion and illumination changes. Each detected object is augmented with its 3D model, its 3D bounding box and its coordinate systems. For better visibility, the background is kept in gray.

this sphere, these points do not vote for this candidate point thus resulting in a runtime $O(nk)$ with $k < n$.

Taking normals into account for subsampling increases the number of subsampled points by a factor typically smaller than two. Therefore the number of points falling into one voxel of the spatial look-up table is not very large and due to (self-) occlusion the overall number of visble points is fairly small, typically around 1000 or less. However, even 1000 points are easily handled in a $O(n^2)$ manner and in such a case matching is done quite quickly. For instance, matching for a close-up view of the chair seen in Fig. 5 is usually done in less than 200 ms.

More problematic are scenes where subsampled points on an object are only a tiny fraction of the overall number of points (the overall number of points can easily exceed $15k$). However, in these cases our spatial look-up table kicks in and the run-time goes from $O(n^2)$ for Drost et al. [8] to $O(kn)$ where $k \ll n$. k is often over twenty times smaller than n.

In short, in practice, the complexity of our approach is significantly better than Drost's [8].

5.5 Contribution of Each Step

We also performed experiments on the occlusion dataset [15] with all eight objects to evaluate the influence on the matching score of each of the steps we proposed compared to [8].

To do so, we first ran our implementation of the original Drost-PPF method, and computed the average matching score $\overline{S_{\text{Drost}}}$ for all eight objects on the dataset of [15]. We then turned on all our contributions and computed the new average matching score $\overline{S_{\text{Ours}}}$. The gain g_c from each contribution alone is computed by computing the average matching score $\overline{S_c}$ with only this contribution turned on, and taking $g_c = \overline{S_c}/(\overline{S_{\text{Ours}}} - \overline{S_{\text{Drost}}})$.

As Fig. 4 shows, accounting for sensor noise is with 43.1 % the most important part, directly followed by smart sampling of points with 41.3 % and finally our contribution to pre-processing of the 3D model and the input scene by 15.6 %.

6 Conclusion

We have shown that by cleverly sampling features and by adding feature spreading to account for sensor noise, we can boost the method from Drost et al. [8] to outperform state-of-the-art approaches in object instance detection and pose estimation, including those that use additional information such as color cues.

References

1. Beetz, M., Klank, U., Kresse, I., Maldonado, A., Mosenlechner, L., Pangercic, D., Ruhr, T., Tenorth, M.: Robotic roommates making pancakes. In: Humanoid Robots (2011)
2. Birdal, T., Ilic, S.: Point pair features based object detection and pose estimation revisited. In: IEEE International Conference on 3D Vision (2015)
3. Bonde, U., Badrinarayanan, V., Cipolla, R.: Robust instance recognition in presence of occlusion and clutter. In: European Conference on Computer Vision (2014)
4. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6D object pose estimation using 3D object coordinates. In: Schiele, B., Tuytelaars, T., Fleet, D., Pajdla, T. (eds.) ECCV 2014, Part II. LNCS, vol. 8690, pp. 536–551. Springer, Heidelberg (2014)
5. Choi, C., Christensen, H.: 3D pose estimation of daily objects using an RGB-D camera. In: International Conference on Intelligent Robots and Systems (2012)
6. Choi, C., Taguchi, Y., Tuzel, O., Liu, M.L., Ramalingam, S.: Voting-based pose estimation for robotic assembly using a 3D sensor. In: International Conference on Robotics and Automation (2012)
7. Drost, B., Ilic, S.: 3D object detection and localization using multimodal point pair features. In: 3D Imaging, Modeling, Processing, Visualization and Transmission (2012)
8. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: efficient and robust 3D object recognition. In: Conference on Computer Vision and Pattern Recognition (2010)
9. de Figueiredo, R.P., Moreno, P., Bernardino, A.: Fast 3D object recognition of rotationally symmetric objects. In: Sanches, J.M., Micó, L., Cardoso, J.S. (eds.) IbPRIA 2013. LNCS, vol. 7887, pp. 125–132. Springer, Heidelberg (2013)
10. Fisher, R.: Projective ICP and stabilizing architectural augmented reality overlays. In: International Symposium on Virtual and Augmented Architecture (2001)
11. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Matsushita, Y., Rehg, J.M., Hu, Z., Lee, K.M. (eds.) ACCV 2012, Part I. LNCS, vol. 7724, pp. 548–562. Springer, Heidelberg (2013)
12. Holz, D., Nieuwenhuisen, M., Droeschel, D., Stuckler, J., Berner, A., Li, J., Klein, R., Behnke, S.: Active recognition and manipulation for mobile robot bin picking. In: Röhrbein, F., Veiga, G., Natale, C. (eds.) Gearing Up and Accelerating Cross-fertilization Between Academic and Industrial Robotics Research in Europe. Springer, Berlin (2014)
13. Johnson, A., Hebert, M.: Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Trans. Pattern Anal. Mach. Intell. (1999)
14. Kim, E., Medioni, G.: 3D object recognition in range images using visibility context. In: International Conference on Intelligent Robots and Systems (2011)
15. Krull, A., Brachmann, E., Michel, F., Yang, M.Y., Gumhold, S., Rother, C.: Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In: International Conference on Computer Vision (2015)
16. Mian, A., Bennamoun, M., Owens, R.: Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans. Pattern Anal. Mach. Intell. (2006)
17. Mian, A., Bennamoun, M., Owens, R.: Automatic correspondence for 3D modeling: an extensive review. Int. J. Shape Model. (2005)

18. Mohamad, M., Rappaport, D., Greenspan, M.: Generalized 4-points congruent sets for 3D registration. In: IEEE International Conference on 3D Vision (2014)
19. Nieuwenhuisen, M., Droeschel, D., Holz, D., Stuckler, J., Berner, A., Li, J., Klein, R., Behnke, S.: Mobile bin picking with an anthropomorphic service robot. In: International Conference on Robotics and Automation (2013)
20. Rios-Cabrera, R., Tuytelaars, T.: Discriminatively trained templates for 3D object detection: a real-time scalable approach. In: International Conference on Computer Vision (2013)
21. Rusu, R., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3D registration. In: International Conference on Robotics and Automation (2009)
22. Salas-Moreno, R., Newcombe, R., Strasdat, H., Kelly, P., Davison, A.: SLAM++: simultaneous localisation and mapping at the level of objects. In: Conference on Computer Vision and Pattern Recognition (2013)
23. Tejani, A., Tang, D., Kouskouridas, R., Kim, T.-K.: Latent-class hough forests for 3D object detection and pose estimation. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part VI. LNCS, vol. 8694, pp. 462–477. Springer, Heidelberg (2014)
24. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: Maragos, P., Paragios, N., Daniilidis, K. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 356–369. Springer, Heidelberg (2010)
25. Tuzel, O., Liu, M.-Y., Taguchi, Y., Raghunathan, A.: Learning to rank 3D features. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part I. LNCS, vol. 8689, pp. 520–535. Springer, Heidelberg (2014)
26. Wohlhart, P., Lepetit, V.: Learning descriptors for object recognition and 3D pose estimation. In: Conference on Computer Vision and Pattern Recognition (2015)