Object Detection in Cluttered Environments with Sparse Keypoint Selection

Viktor Seib and Dietrich Paulus
University of Koblenz-Landau
Universitätsstr. 1, 56070 Koblenz, Germany
{vseib,paulus}@uni-koblenz.de

Abstract

In cases such as mobile robotic applications with limited computational resources, traditional approaches might be preferred over neural networks. However, open source solutions using traditional computer vision are harder to find than neural network implementations. In this work we address the task of object detection in cluttered environments in point clouds from RGB-D cameras. We compare several open source implementation available in the Point Cloud Library and present a novel and superior solution for this task. We further propose a novel sparse keypoint selection approach that combines the advantages of uniform sampling and a dedicated keypoint detection algorithm. Our extensive evaluation shows the validity of our approach, which also improves the results of the compared methods. All code is available on our project repository: https://github.com/vseib/point-cloud-donkey.

1. Introduction

The application of deep learning methods for computer vision tasks is omnipresent. In recent years neural networks have been defining the state-of-the-art in object classification and detection, despite high computational demands for training and the need for vast amounts of annotated data. With the release of affordable RGB-D cameras, point clouds became a modality of high research interest. Several large-scale benchmarks featuring point clouds have been introduced. ModelNet [28] offers synthetic point clouds, whereas the Washington RGB-D [8] and BigBird [23] data sets provide real-world colored point clouds from RGB-D cameras for classification.

The necessity to process point clouds is increasing due to LiDAR sensors and increased interest in autonomous cars. Our research focuses on object detection in cluttered scenes for mobile robotic applications. Due to the current relevance of point cloud processing we focus on this modality. In our case, we process point clouds from RGB-D cameras.

Mobile robots have a very limited energy budget and limited computational resources. Every increase in power consumption reduces the total time of operation. Therefore, it is desirable to have robots equipped with a single computing device that has reasonable power demands. Usually, this results in using a notebook computer without a GPU. This is were traditional approaches have their strength. Inference (and even training) of traditional approaches is viable on portable computers already available on mobile robotic platforms. Further, traditional approaches require considerable less data and time for training. Trading off these advantages for a lower accuracy compared to neural networks might be acceptable in many cases for robotic applications.

A key component to successfully detecting objects in cluttered point clouds are sophisticated algorithms for hypothesis generation (HG) and hypothesis verification (HV). These tasks are sometimes addressed individually (HG: [24], [20] and HV: [13], [1]) or simultaneously ([3], [30]). With numerous pretrained models and implementations available, deep learning has become easily accessible – more than traditional approaches have been in the past. Luckily, some implementations of traditional approaches are available as part of the Point Cloud Library (PCL) [18].

However, when examining these approaches more carefully, we notice that they often address object instance retrieval in scenes [24], [1], [30]. This means that given on object point cloud the algorithm is capable of finding all occurring instances and determine their poses in a given scene point cloud. Application scenarios for this task are bin picking or retrieving objects from a shelf. In contrast, we are interested in generalized object detection, where the actual object instances are not available at test time. Instead, the scene point cloud is searched for objects based on previously extracted information, for example a codebook of local feature descriptors.

Generalized object detection amongst clutter is more challenging than object retrieval. We do not have the exact object shape that is searched for, but only the object's features. Further, we want to find all relevant objects from the data set by only processing the scene once. This means that we might obtain multiple instances of multiple objects,

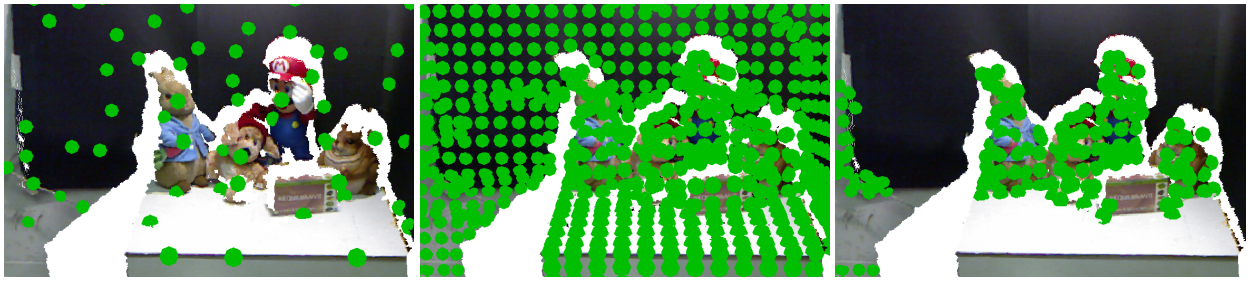

Figure 1. Keypoints detected with the Intrinsic Shape Signatures approach (left) and with a uniform voxel grid (center). Our proposed keypoint selection method (right) discards most of the keypoints not belonging to the objects of interest.

while dealing with occlusions and clutter. Object retrieval in scenes on the other hand, obtains multiple instance of only one class when the scene point cloud is processed.

To achieve satisfactory results we argue that the hypothesis generation and verification steps must be aided during keypoint extraction. Salient keypoint are essential for this purpose, yet for reasons of computational efficiently many approaches resort to using uniform sampling. We therefore make the following contributions: We present a point cloud processing pipeline designed for object detection in cluttered environments. Further, we propose a novel, sparse keypoint filtering that combines the computational efficiency of uniform sampling with the repeatability of a dedicated keypoint detection algorithm (Figure 1). As a third contribution, we evaluate and compare available object detection implementations. Our extensive evaluation demonstrates that the presented contributions are superior to the existing methods.

The remainder of this work is structured as follows: After discussing related work in Section 2 we present our pipeline for generalized object detection in Section 3. A detailed description of our contributions for keypoint selection follows in Section 4 and HG and HV are described in Section 5. An extensive evaluation presented in Section 6 shows the validity of our approach. The results are discussed in Section 7, while Section 8 concludes the paper.

2. Related Work

This section on related work is divided in topics relevant to compare the proposed method with the existing implementations in the PCL. We highlight important aspects of our work and their relevance compared to related approaches.

2.1. Data Representation

Early approaches dealing with 3-D data mostly used datasets available as meshes [7], [19]. Meshes can be converted to point clouds by simply taking the mesh vertices or by sampling the mesh surface to obtain a denser point cloud. With the advent of affordable RGB-D cameras, point clouds were explicitly addressed in the experiments [24], [1]. The work presented in [30] performs experiments on both types of data. According to [30] the results on point clouds are always worse. This is attributed to meshes being a richer data structure that allows to extract more meaningful information. While this is true, especially due to the vertex and face information, real-world sensors such as RGB-D cameras and LiDAR always provide point clouds. We therefore focus on point clouds in this study to allow for real world applications of our research.

2.2. Keypoints and Features

Traditional approaches are often based on extracting local feature descriptors at selected keypoint positions. Popular examples for local features on point clouds include FPFH [17], SHOT [25] and the RoPS [6] descriptor. In their experiments on point clouds Zhou *et al*. [30] examined the SHOT and RoPS descriptors and found that using SHOT led to better results in terms of the applied metrics. The keypoint positions can be selected with a dedicated algorithm, such as ISS [29] or KPQ [12] that find unique and repeatable positions on the input data. However, since computing these positions implies an additional computational overhead, many approaches resort to extracting features on a random [24] or uniform grid [19].

Descriptors obtained from an input point cloud are matched in a nearest-neighbor fashion with a previously created codebook. Grid sampling and nearest-neighbor matching provoke erroneous feature correspondences. To alleviate these effects, only matches below a certain feature distance threshold can be accepted. This threshold can be fixed [24] or adapt automatically to the number of hypotheses [30].

2.3. Hypothesis Generation and Classification

Hypothesis generation and especially hypothesis verification is rarely explicitly addressed. In the Implicit Shape Models (ISM) [9] of Leibe *et al*. hypotheses for object locations are generated by voting using the generalized Hough-transform [2]. The votes are cast based on relative vectors obtained during the training phase. This is in contrast to e.g.

the work of McCann and Lowe [11] where hypotheses are simply generated by the frequency of occurrence of feature matches. Later work adapted the hypothesis generation of Leibe *et al*. to 3-D data, treating the maxima resulting from Hough-voting as an implicit hypothesis verification step [7], [19], [20]. These methods are suitable for point cloud classification, but struggle with many false positive detections when exposed to cluttered point clouds. Apart from using the generalized Hough-transform other methods apply correspondence grouping [3], [1] or pose clustering [13], [4] to generate hypotheses for object locations and poses.

2.4. Hypothesis Verification and Detection

Hypothesis verification methods are mostly centered around object and scene alignment using the Iterative Closest Points (ICP) algorithm and some thresholds and similarity measures. After generating hypotheses with correspondence grouping, Chen *et al*. [3] remove some outliers based on point distances. Subsequently, they run the ICP algorithm with randomly selected correspondences for several times and select the hypothesis with lowest registration error. Mian *et al*. [13] align the coordinate frames of the object and scene to obtain a coarse registration. This is further refined with ICP on a downsampled scene. If certain quality measures are satisfied, the object model is pruned for occlusion and ICP is run again on the full-resolution scene. Aldoma *et al*. [1] use the method of Chen *et al*. [3] to generate hypotheses. They propose a verification scheme based on global optimization over geometric and visibility cues. Finally, the more recent work by Zhou *et al*. [30] emphasizes the generation of strong hypotheses by a self-adapted measure with a double RANSAC step. The thresholds in the hypothesis generation step adapt to the occlusion levels and clutter of the scene to obtain the best possible hypotheses. These are refined by employing ICP twice with different thresholds and validation checks.

2.5. Open Source Implementations

One of the objectives of this work is to evaluate available open source algorithms for generalized object classification and detection. To facilitate the comparison, we focus on implementations using the PCL as a common library. The PCL provides implementations of the algorithms of Tombari *et al*. [24], Aldoma *et al*. [1] along with the hypothesis generation of Chen *et al*. [3]. Further, we carefully reimplement the approach of Zhou *et al*. [30] using the PCL. Although mainly targeting object detection, all of these methods can be used for classification of isolated objects by simply regarding the strongest hypothesis as the classification result.

We also examine two implementations of the ISM algorithm for point cloud classification, namely the approach of Knopp *et al*. [7] (available in the PCL), as well as the approach of Seib *et al*. [20] (implemented using the PCL).

Special care is taken by adapting the algorithms of Tombari *et al*., Aldoma *et al*. and Zhou *et al*. from instance retrieval to generalized object detection. Specifically, instead of an object point cloud and scene point cloud, the input consists only of the scene point cloud along with a previously learned codebook. The codebook contains extracted feature descriptors, along with their original keypoint positions, local reference frames and relative center vectors if required by the respective method. Whenever the original algorithm performs point cloud registration, we construct sparse point clouds based on the keypoint positions of the matched features in the scene and the codebook. These sparse point clouds are used for the ICP algorithm.

2.6. Our Approach

Our goal is finding a generalized object detection approach for cluttered point clouds. We want this method to run efficiently on standard mobile computers without a GPU to be usable on mobile robotic platforms. Achieving this goal is a challenging task and is not possible with the available implementations so far. We therefore propose the following novel modifications to enhance the standard object processing pipelines.

Similar to Zhou *et al*. [30] we see a fault tolerant hypothesis generation process as essential for the following step of verification. Zhou *et al*. use a self-adapted measure to find reasonable threshold for hypothesis generation. Contrary, our proposed keypoint selection algorithm combines the speed of grid sampling with the repeatability from dedicated keypoint detection algorithms. The features from the selected keypoints produce more relevant correspondences and improve the total detection results.

With the established correspondences we generate hypotheses in a continuous 3-D Hough-space similar to Seib *et al*. [20]. However, different than Seib *et al*. we exploit the local reference frame from feature extraction to find a full 6-D pose estimate, making our pipeline suitable for object detection. The complete point cloud processing pipeline proposed in this work, along with all modifications applied to available implementations, is released to the public and available on our project repository.

3. 3-D Detection Pipeline

This section briefly introduces our complete object processing pipeline that is suitable for object classification and detection. The pipeline is shown in Figure 2.

3.1. Offline Stage

The offline stage (training) is used to create a spatial codebook of local feature descriptors. All 3-D objects or 2.5-D object views of a data set's training split are used to extract keypoints on a uniform grid. For each keypoint we compute SHOT [25] or CSHOT [26] descriptors depending

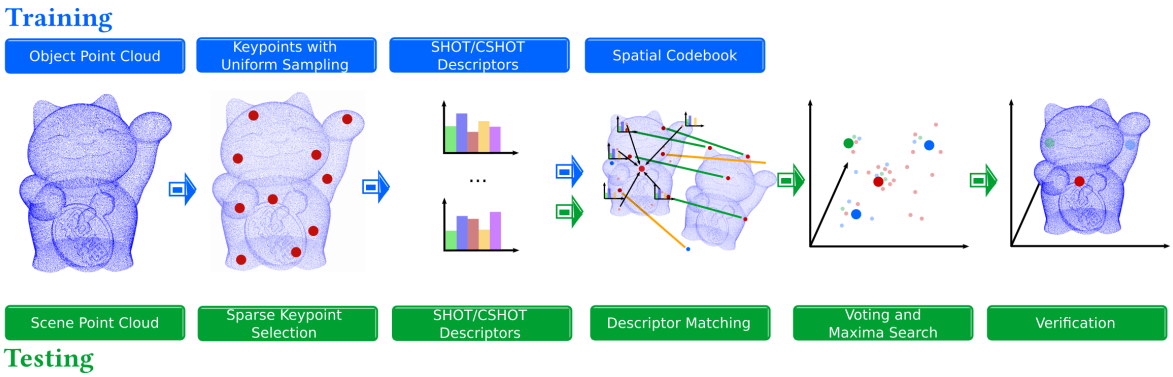

Figure 2. Our complete point cloud processing pipeline. The training stage (blue) is used to create a spatial codebook, which is used in the testing stage (green) for object classification and detection.

on whether the data contains color cues or not. Each descriptor is associated with a local reference frame, a center vector pointing to the object's centroid, a class id label and a vector indicating the extent of the object's bounding box.

3.2. Online Stage

During the online stage (testing) the pipeline is given the created codebook from training and an object for classification or a scene for object detection. Keypoints are extracted on the input point cloud using our proposed sparse keypoint selection method (Section 4). Each computed descriptor is matched with the codebook using Chi-Squared distance. With SHOT and CSHOT being histogram descriptors we found that Chi-Squared produces better results than Euclidean distance which is frequently used for matching. The established feature correspondences are used to create hypotheses for object locations by casting votes in a continuous 3-D Hough-space. We use a separate voting space for each object class to filter the hypotheses (Section 5).

4. Keypoint Selection and Matching

Keypoint detection algorithms [29], [12] are capable of finding salient and repeatable points to compute descriptors. Despite their proven capability of selecting good keypoints, many approaches sample keypoints on a grid [19], [24], [30], [20]. While grid sampling produces less reliable keypoints that encompass more false descriptor matches, it is much faster than using a dedicated keypoint selection algorithm. One of the reasons why dedicated keypoint detection algorithms are slow, is their general working procedure. They have to *densely* evaluate the surface of the point cloud to find salient regions according to a given measure.

In contrast, we propose a *sparse* evaluation to obtain descriptive keypoints. We first sample keypoints on a grid and then only evaluate the sparse sampled regions instead of the complete point cloud surface. This allows us to find descriptive keypoints with only a small computational overhead compared to grid sampling. Moreover, the smaller amount of resulting keypoints speeds ups the following pipeline steps and the increased keypoint saliency and repeatability promotes more correct matches.

We use a voxel grid with the size of g along each axis. The centroid of all points falling into a grid cell is regarded as the keypoint location. Our experiments have shown that for metric data from RGB-D cameras a value in the range of $g = 0.02\,\text{m}$ provides a good trade-off between accuracy and number of points to evaluate in the following step.

In the second step we apply a quality measure based on the properties of the local point neighborhood of the selected grid points. The assumption is that the local curvature of the point cloud is a good saliency measure. This also allows to discard homogeneous surfaces such as tables or walls that produce many unreliable keypoints. For each sampled keypoint $\hat{\mathbf{p}}_j$ we find the set of points defining its neighborhood $\mathcal{N}$ in the radius given by g, obtaining a local surface $\mathcal{S}_j = \{\mathbf{p}_i\} = \mathcal{N}_g \hat{\mathbf{p}}_j$. We continue by computing the Gaussian curvature $K = \kappa_1 \kappa_2$ on the local surface $\mathcal{S}_j$ to find a quality measure for the keypoint $\hat{\mathbf{p}}_j$. In this case κ_1 and κ_2 are the principal curvatures of $\mathcal{S}_j$.

After evaluating all keypoints, we keep a certain percentage of the "best" keypoints according to our quality measure. Other strategies might be using a fixed threshold for K or automatically selecting a threshold based on a histogram of all values K. For an example, see Figure 1, where 75% of all initial keypoints were discarded. Note that almost all remaining keypoints concentrate on the objects.

The selected keypoint locations are used to compute feature descriptors. These descriptors are matched with a codebook obtained during the training stage. We found that using a matching threshold mostly degrades the performance and simply match each feature from the scene to its nearest neighbor in the codebook.

5. Hypothesis Generation and Verification

The Hough-transform is a widely used method for object detection [2]. Many related approaches use a discrete

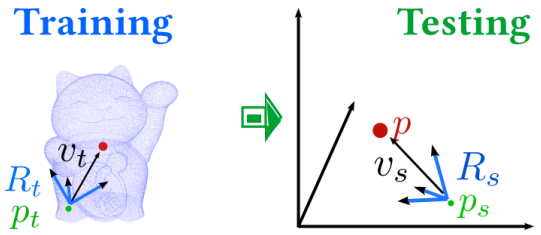

Figure 3. The center vector $\mathbf{v}_t$ from training is converted with the local reference frames R_t and R_s into the global frame and casts a vote at position $\mathbf{p}$ from its keypoint location $\mathbf{p}_s$.

Hough-space for hypothesis generation [19], [24], [30]. However, we opted for a continuous Hough-space, following the ideas of Leibe *et al.* [9] and Seib *et al.* [20]. By using a continuous Hough-space we avoid discretization and boundary effects that are typical for discrete Hough-spaces.

Hypothesis generation is based on established feature correspondences between the input at the testing stage and the codebook from training. Data from the codebook is indicated by a subscript $_t$ (training) and data from the input at testing time is indicated by a subscript $_s$ (scene). With each correspondence we have a pair of local reference frames R_t, R_s (represented as 3×3 rotation matrices) and a pair of descriptors $\mathbf{f}_t, \mathbf{f}_s$. We further have the keypoint location $\mathbf{p}_s$ where $\mathbf{f}_s$ was computed from (and $\mathbf{p}_t$ from training). The codebook additionally gives us the object class id C belonging to feature descriptor $\mathbf{f}_t$, the center vector $\mathbf{v}_t$ and the bounding box extent vector $\mathbf{b}$. The object class id C indicates which Hough-space will be used to generate a hypothesis from this feature correspondence.

A hypothesis is generated by casting a vote into the corresponding Hough-space of class C. Each vote is generated by obtaining the rotation matrix

$$R = R_s^T R_t \tag{1}$$

which allows us to transform the center vector $\mathbf{v}_t$ from the reference frame in training to the reference frame of the Hough-space used in the testing stage:

$$\mathbf{v}_s = R\mathbf{v}_t \,. \tag{2}$$

Thus, a vote is cast at position $\mathbf{p} = \mathbf{p}_s + \mathbf{v}_s$ into the Hough-space C. This process is visualized in Figure 3. After the voting procedure we have $|C|$ 3-D voting spaces each with object center hypotheses of one class C. Note that the number of classes $|C|$ is known from the training phase.

The final step for hypothesis generation is to find the maxima in the voting space. Since we have continuous voting spaces we can not simply check the relevant bins for maxima locations. Instead, we regard each voting space as a sparse probability density function and use the mean-shift algorithm for maxima detection [5]. Due to the sparse nature of the voting spaces that is aided by separate voting for each class, our maxima search is very time efficient. The mean-shift algorithm returns a set of hypothetical object locations. After non-maxima suppression, the density at each remaining position is estimated with a Gaussian kernel K as

$$\hat{m}(\mathbf{x}) = \frac{1}{nh^3} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{p}_i}{h}\right) , \tag{3}$$

where $\hat{m}$ indicates a maximum at position $\mathbf{x}$ with n votes at positions $\mathbf{p}_i$. The bandwidth parameter h is similar to the bin size in a discrete voting space and indicates the radius where the n votes were extracted from.

The proposed sparse keypoint selection already helps in removing keypoints that lead to erroneous correspondences. Still, some false matches might still be present in the final hypothesis. We therefore apply RANSAC based on keypoint pairs $\{\mathbf{p}_t, \mathbf{p}_s\}$ from the correspondences of each maximum to remove outliers. After outlier removal we update the density estimation at each maximum location (Equation (3)). Finally, the bounding box of each hypothesis is obtained by averaging the extent vectors $\mathbf{b}$ of each correspondence in the reference frame of the hypothesis.

6. Experiments

We perform different experiments to demonstrate the validity of the proposed techniques. First and isolated from the presented object detection pipeline, we evaluate the proposed keypoint selection approach on the RandomViews and Kinect data sets [27], see below for details.

Additionally, we evaluate two application domains, object classification and object detection, of our object processing pipeline (Section 3) and the pipelines presented in Section 2.5. For this purpose, we use synthetic data sets with 3-D objects as well as real-world 2.5-D data captured with RGB-D cameras. In all cases data is represented as point clouds. More details on the employed data sets are given in the next section.

For a comparative evaluation with other approaches we employ the code published by the respective authors. Where the code is not available we carefully reimplement the approaches using the PCL and all available information from the respective publications. Using the PCL for all object processing pipelines allows for a better comparison of all approaches in terms of performance metrics and runtime. In all of the following experiments we use SHOT or CSHOT feature descriptors and the same keypoint sampling grid with $g = 0.02\,\text{m}$.

6.1. Data Sets

To evaluate the classification on synthetic datasets we choose the Princeton Shape Benchmark (PSB) [22] and the ModelNet data sets [28]. The latter data set is primarily used to benchmark neural networks and is available in two

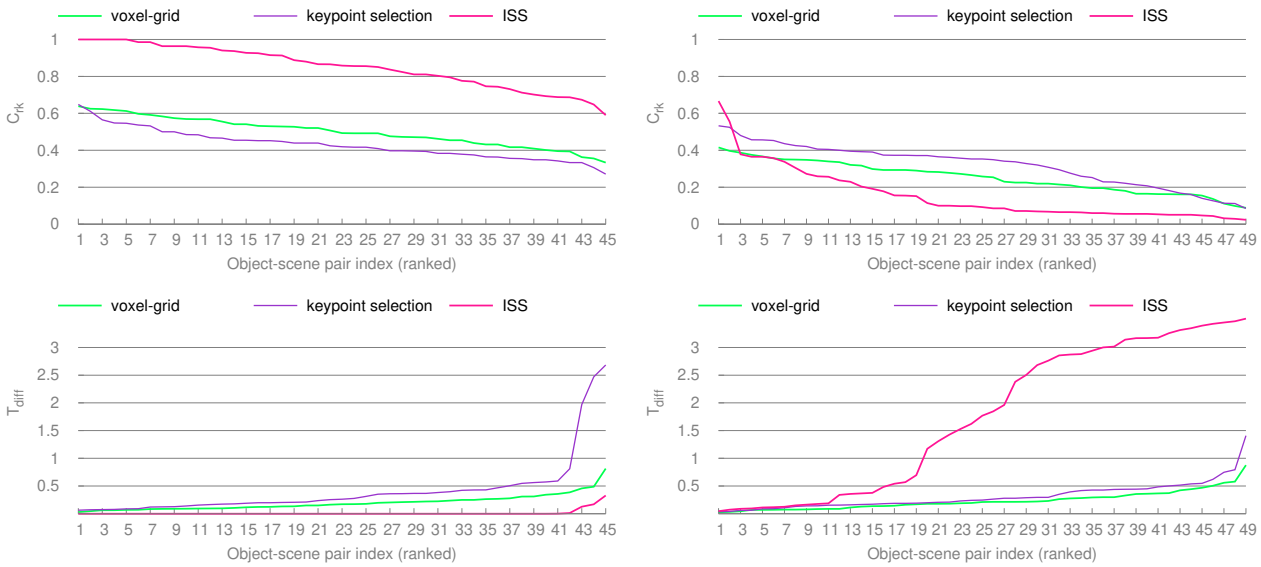

Figure 4. Evaluation results for the RandomViews data set without clutter (left) and the Kinect data set with clutter (right). The proposed keypoint selection method shows its benefits in data with clutter by producing more correct keypoint matches than the other methods. Note that higher values are better for the C_{rk} metric, while lower values are better for the T_{diff} metric.

variants: with 10 (MN10) and with 40 (MN40) different object classes. These classification data sets consist of isolated 3-D objects without clutter. Since these data sets are available as meshes, we take the original vertices and sample each surface to obtain a denser point cloud.

Further, we test classification on real world data from RGB-D cameras, namely on the Washington RGB-D (WASH) [8] and the BigBird (BIGB) [23] data sets. These data sets contain original data with noise and holes.

For the detection benchmark we employ the typical data sets used in this area [27]. We use RandomViews (RAV) and Kinect (KIN) to obtain viable hyperparameters individually for each of the evaluated pipelines. RandomViews is composed of isolated objects and is used exemplary for a data set without clutter. On the other hand, the Kinect data set contains objects of interest, unknown objects along with background data and represents a data set with clutter. An example scene from Kinect data set is shown in Figure 1.

6.2. Keypoint Detection

We perform a similar evaluation as proposed by Prakhya *et al*. [15] for their BSHOT descriptor to evaluate keypoint selection and matching using the RandomViews and Kinect data sets [27]. These data sets consist of isolated objects (RandomViews) and object views (Kinect), along with several scenes comprising 2 to 5 objects each. In total there are 49 object-scene pairs in the Kinect data set, while the number of object-scene pairs in the RandomViews data set is much larger. To keep the evaluation concise, we limit the object-scene-pairs to 45 using only one object per scene.

Note that in contrast to the classification and detection evaluation, keypoint matching is evaluated by presenting one 3-D object (or one 2.5-D object view) and a scene containing this object to the algorithm. We then extract keypoints on the scene and object and compute feature descriptors. In the next step reciprocal correspondences between the object and scene descriptors are established. Finally, we use RANSAC to filter outliers and estimate a 3-D transformation T_h of the object. The quality of the estimated transformation hypothesis T_h is assessed by computing the difference T_{diff} between the ground truth transformation T_g and T_h with the Euclidean metric

$$T_{\text{diff}} = \sqrt{\sum_{i=1}^{4}\sum_{j=1}^{4}\left(T_{h_{ij}} - T_{g_{ij}}\right)^2}. \tag{4}$$

The quality of the descriptor matching is assessed with

$$C_{rk} = \frac{c_r}{k_m} \tag{5}$$

as the ratio between the RANSAC correspondences c_r and the number of keypoints extracted from the object k_m. These two metrics are used to compare keypoints extracted on a uniform grid, our proposed keypoint selection method, as well as a dedicated keypoint detection algorithm. For the latter, we choose the Intrinsic Shape Signatures (ISS) [29].

The evaluation results are presented in Figure 4. The values along the axis of abscissae are ranked from best to worst experiment for each graph individually. This allows for an easier comparison of the individual methods. For these experiments we setup our proposed keypoint selection method

to retain the best 50% of the keypoints found by the voxel grid. In general we observe that ISS is superior to the other methods when no clutter is present (Figure 4, left). It allows for significantly more correct keypoint matches and almost errorless transformation matrices. Naturally, when no clutter is present in the data our proposed keypoint selection throws away good keypoints and its performance degrades compared to the voxel grid method.

However, real data is expected to be noisy and to contain some clutter. In this case, shown on the right of Figure 4 with the Kinect data set, the ISS keypoint method is inferior to the other two. On the other hand, by selecting the best keypoints our method is superior to the other approaches in terms of correctly matched keypoints, while at the same time producing similar errors for the computed transformation matrices as the voxel-grid keypoints.

We further measured the time for keypoint computation across multiple runs of both data sets. When normalizing the time for ISS keypoint computation to 1.0, our proposed method takes a relative time of 0.06, while the voxel-grid keypoints take a relative time of 0.01. This small increase in processing time is compensated for in later pipeline steps as a much lower number of features needs to be processed.

In summary, our proposed keypoint selection method is much faster than a dedicated keypoint algorithm while at the same time providing better results in data amongst clutter. On the other hand, in artificial data without noise or clutter our proposed methods shows no benefits in terms of keypoint matching quality.

6.3. Object Classification

The evaluation metric used for classification is accuracy, defined as the ratio of correctly classified objects and the total number of objects in the data set. We carefully choose the best hyperparameters for the evaluation of each individual method. Approaches focusing on hypothesis verification and object detection introduced in Section 2 are also evaluated here. Since no object pose is necessary for classification, we omit the hypothesis verification steps of these approaches and regard the hypothesis with the highest weight as the classification result. Using the open source pipelines for classification allows for a sanity check in terms of whether the adjustments made to the methods are viable.

Table 1 shows the classification results for all open source approaches investigated here. Additionally, below the separating line we add two neural networks and another traditional approach that is not available as PCL-based implementation for comparison. The neural networks, PointNet [16] and KCNet [21] outperform the traditional approaches on the MN10 and MN40 data sets. The same holds for the method of Li *et al*. [10] for RGB-D data.

The ISM 3-D approach available in the PCL [7] has a very long training time and inferior results and is therefore excluded from further evaluations. The method of Chen *et al*. [3] is the hypothesis generation approach used by Aldoma *et al*. [1]. The former is used for classification, the latter will be examined in Section 6.4 for detection.

Table 1. Accuracy achieved by the evaluated approaches on different data sets. From the PCL-based methods (above the separating line), the approach of Seib *et al*. performs best. Still, neural networks and Li *et al*. achieve the best overall results (below the line).

Method	PSB	MN10	MN40	WASH	BIGB
Knopp [7]	0.37	0.25	-	-	-
Chen [3]	0.62	0.78	0.73	0.63	0.45
Tombari [24]	0.65	0.82	0.77	0.80	0.63
Zhou [30]	0.64	0.84	0.78	0.73	0.60
Seib [20]	**0.68**	**0.86**	**0.81**	**0.85**	**0.76**
Our	0.65	0.81	0.77	0.83	0.71
Qi [16]	-	-	0.89	-	-
Shen [21]	-	**0.97**	**0.96**	-	-
Li [10]	-	-	-	**0.94**	**0.88**

The other approaches [24], [30], [20] have a similar performance on data sets with isolated objects (PSB, MN10, MN40). Regarding data from RGB-D cameras (WASH and BIGB data sets), [30] falls behind [24] and [20] which both use center vectors for hypothesis generation. The approach of Seib *et al*. [20] is the only one using a continuous Hough-space, which we adapt for our own pipeline. However, we omit the "codebook cleaning" and "global features" parts of their pipeline to simplify the overall approach and to avoid introducing more hyperparameters. Even with these parts omitted, our pipeline is still competitive and promises good results with an additional hypothesis verification step for object detection. One final note on the WASH data set: Li *et al*. used the complete data set in their evaluation, while we used a balanced version of WASH, which has the same number of classes, but only exactly three instances per class.

6.4. Object Detection

We apply the F_1 score as evaluation metric for object detection, which is defined as a harmonic mean of precision (pr) and recall (re):

$$F_1 = \frac{2 \cdot \mathrm{pr} \cdot \mathrm{re}}{(\mathrm{pr} + \mathrm{re})} \,. \tag{6}$$

Since this metric relies on true and false positive and negative detections, we regard a detection as true positive if the found location is within a radius of d_{max} of the ground truth object position. Considering the small object size and the metric data of the applied data sets, we set $d_{max} = 0.05\,\mathrm{m}$. This is factor 2.5 of the keypoint extraction grid and similar to the quality distance measure often defined as "twice the mesh resolution" in other work [30].

Table 2. Detection results and average times to process a single scene. Best results are shown in bold typeset.

Method	RAV	KIN	RAV time	KIN time
Aldoma [1]	0.08	0.02	2.0 s	3.5 s
Tombari [24]	0.42	0.41	2.1 s	2.2 s
Zhou [30]	0.32	0.07	2.2 s	2.1 s
Our (baseline)	**0.70**	**0.44**	2.0 s	1.8 s

Evaluations are performed on a mobile computer with an Intel i7-2760QM processor, 10 GB RAM and Ubuntu Linux 18.04. This is an outdated processor which is weaker than modern processors that can be expected to be used on a mobile robot. For instance, the CPU benchmark [14] indicates that a modern i3 processor for mobile devices has more than double the computing power of the evaluating system.

Despite our efforts of adapting these methods and a careful and time consuming search for best parameters, the results were disappointing. As can be seen from Table 2, the pipeline of Aldoma *et al*. [1] is not suitable for object detection. Our PCL-based implementations of Tombari *et al*. [24] and Zhou *et al*. [30], perform better, but the latter fails when clutter is present in the data.

We attribute this bad performance to the use of the ICP algorithm in many hypothesis verification approaches. With the task being object detection in contrast to retrieval, the sparse keypoint point clouds are much harder to register for the ICP algorithm. Further, some hypothesis verification methods (especially the one proposed by Aldoma *et al*.) employs thresholds of ratios between inlier, outlier and occluded points. With only sparse clouds and a low total number of points, such thresholds become less robust.

Our own pipeline as described in Section 3 and without the contribution of Section 4 performs best and serves as our own baseline for the following evaluations. Table 3 shows the effect of sparse keypoint selection (SKS) when applied to the baseline method. The numbers behind SKS indicate the percentage of all initially computed grid keypoints that were discarded. All variants of SKS improve the results compared to the baseline on both data sets. This demonstrates that reducing ambiguous keypoint locations and thus the number of wrong correspondences boosts the detection performance. Additionally, as points get discarded, the processing times are significantly reduced and make up for the small overhead of computing our keypoint metrics.

As a final evaluation we plug in the SKS into the adapted PCL pipelines. The results are shown in Table 4. Using the proposed keypoint selection method improves the results for two of the pipelines on the cluttered Kinect data set. However, SKS has no benefit when used with the clean RandomViews data set.

Table 3. Detection results and average times per scene with our contributions. Using the proposed sparse keypoint selection (SKS) improves the results in all cases (best results in bold).

Method	RAV	KIN	RAV time	KIN time
Our (baseline)	0.70	0.44	2.0 s	1.8 s
+ SKS-25	0.77	0.45	1.9 s	1.5 s
+ SKS-50	**0.83**	**0.51**	1.6 s	1.1 s
+ SKS-75	**0.83**	0.45	1.2 s	0.9 s

Table 4. Detection results and times achieved with adapted implementations and our contributions (improved results in bold).

Method + SKS-50	RAV	KIN	R. time	K. time
Aldoma [1]	**0.09**	0.01	1.5 s	1.5 s
Tombari [24]	0.35	**0.44**	1.4 s	1.1 s
Zhou [30]	0.23	**0.17**	1.8 s	1.7 s

7. Discussion

The evaluation in the previous section has proven the validity of our contributions for point cloud classification and object detection. Further, the proposed keypoint selection helps to increase the detection rate of other methods. The proposed keypoint filtering decreases the number of feature correspondences available to the following pipeline steps. In some cases, for example, data set without clutter, too many correspondences might be discarded and the total detection results decrease (Table 4). Nevertheless, in case of our own pipeline all variants of sparse keypoint selection have improved the results (Table 3).

One should keep in mind that our point cloud processing pipeline was build with the task of object detection in mind. The comparative approaches from the PCL on the other hand are designed for object retrieval. Although taking great care in their adaptation to object detection, we could not make them work as initially anticipated. By releasing our code we therefore hope to make a contribution to the community interested in traditional computer vision methods for point cloud processing.

8. Summary

In this work we evaluated point cloud processing pipelines available in the PCL for the task of object detection. Our own approach has proven superior to the existing methods in this task. Its results were further improved with the contributed keypoint selection method. We have further shown that combining our contributions with existing methods also improves the results.

References

[1] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3d object recognition. In *European conference on computer vision*, pages 511–524. Springer, 2012. 1, 2, 3, 7, 8

[2] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. 2, 4

[3] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, 2007. 1, 3, 7

[4] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010. 3

[5] Keinosuke Fukunaga and Larry D Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975. 5

[6] Yulan Guo, Ferdous A Sohel, Mohammed Bennamoun, Jianwei Wan, and Min Lu. Rops: A local feature descriptor for 3d rigid objects based on rotational projection statistics. In *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, pages 1–6. IEEE, 2013. 2

[7] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *European Conference on Computer Vision*, pages 589–602. Springer, 2010. 2, 3, 7

[8] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE international conference on robotics and automation*, pages 1817–1824. IEEE, 2011. 1, 6

[9] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7, 2004. 2, 5

[10] Chi Li, Austin Reiter, and Gregory D Hager. Beyond spatial pooling: fine-grained representation learning in multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4913–4922, 2015. 7

[11] Sancho McCann and David G Lowe. Local naive bayes nearest neighbor for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3650–3656. IEEE, 2012. 3

[12] Ajmal Mian, Mohammed Bennamoun, and Robyn Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2):348–361, 2010. 2, 4

[13] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1584–1601, 2006. 1, 3

[14] Passmark Software. CPU Benchmarks. https://www.cpubenchmark.net/, 2021. 8

[15] Sai Manoj Prakhya, Bingbing Liu, and Weisi Lin. B-shot: A binary feature descriptor for fast and efficient keypoint matching on 3d point clouds. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1929–1934. IEEE, 2015. 6

[16] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 7

[17] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 2

[18] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011. 1

[19] Samuele Salti, Federico Tombari, and Luigi Di Stefano. On the use of implicit shape models for recognition of object categories in 3d data. In *Asian Conference on Computer Vision*, pages 653–666. Springer, 2010. 2, 3, 4, 5

[20] Viktor Seib, Nick Theisen, and Dietrich Paulus. Boosting 3d shape classification with global verification and redundancy-free codebooks. In *VISIGRAPP (5: VISAPP)*, pages 257–264, 2019. 1, 3, 4, 5, 7

[21] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018. 7

[22] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Proceedings Shape Modeling Applications, 2004.*, pages 167–178. IEEE, 2004. 5

[23] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE international conference on robotics and automation*, pages 509–516. IEEE, 2014. 1, 6

[24] Federico Tombari and Luigi Di Stefano. Object recognition in 3d scenes with occlusions and clutter by hough voting. In *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*, pages 349–355. IEEE, 2010. 1, 2, 3, 4, 5, 7, 8

[25] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proc. of the European Conf. on computer vision*, ECCV'10, pages 356–369. Springer-Verlag, 2010. 2, 3

[26] Federico Tombari, Samuele Salti, and Luigi Di Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. In *Proc. of the International Conference on Image Processing (ICIP)*, pages 809–812. IEEE, 2011. 3

[27] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance evaluation of 3d keypoint detectors. *International Journal of Computer Vision*, 102(1-3):198–220, 2013. 5, 6

[28] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d

shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1, 5

[29] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696. IEEE, 2009. 2, 4, 6

[30] Wei Zhou, Caiwen Ma, and Arjan Kuijper. Hough-space-based hypothesis generation and hypothesis verification for 3d object recognition and 6d pose estimation. *Computers & Graphics*, 72:122–134, 2018. 1, 2, 3, 4, 5, 7, 8