TUTORIAL

©DIGITAL VISION

Point Cloud Library

Three-Dimensional Object Recognition and 6 DoF Pose Estimation

By Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze

With the advent of new-generation depth sensors, the use of three-dimensional (3-D) data is becoming increasingly popular. As these sensors are commodity hardware and sold at low cost, a rapidly growing group of people can acquire 3-D data cheaply and in real time.

With the massively increased usage of 3-D data for perception tasks, it is desirable that powerful processing tools and algorithms as well as standards are available for the growing community. The Point Cloud Library (PCL) [1] aims at providing exactly these. It is a collection of state-of-the-art algorithms and tools to process 3-D data. The library is open source and licensed under Berkeley Software Distribution (BSD) terms and, therefore, free to use for everyone. The PCL project brings together researchers, universities, companies, and individuals from all around the world, and it is rapidly becoming a reference for anyone interested in 3-D processing, computer vision, and robotic perception. The PCL core is structured in smaller libraries offering algorithms and tools for specific areas of 3-D processing, which can be combined to efficiently solve common problems such as object recognition, registration of point clouds, segmentation, and surface reconstruction, without the need of reimplementing all parts of a system needed to solve these subtasks. In other words, the tools and algorithms provided by PCL allow researchers and companies to better concentrate on their specific areas of expertise.

In this article, we focus on 3-D object recognition and pose estimation. Specifically, our goal is to recognize rigid objects from a single viewpoint and estimate their position and orientation in the real world. The objects used to train the system are represented as 3-D meshes, and the real objects are sensed using a depth sensor such as the Kinect. In particular, we review several state-of-the-art 3-D shape descriptors aimed at object recognition, which are included in PCL and are publicly available.

As a good descriptor in the end is a small part of an object recognition system, we present two entire pipelines

Digital Object Identifier 10.1109/MRA.2012.2206675
Date of publication: 10 September 2012

1070-9932/12/$31.00©2012IEEE

for our recognition system built using bits and pieces available in PCL: one relies on global descriptors that require the notion of objects and hence deploy a specific preprocessing step based on segmentation while the other one uses local descriptors that are computed locally around key points and, thus, do not require a presegmentation step. Throughout the article, the advantages and disadvantages of the different methods used are presented to provide the reader with the necessary insights for designing the features of a recognition system to be used in a specific front-end application. The tutorial also provides specific guidelines on how to use PCL primitives for the various methods presented.

Finally, an experimental evaluation is performed to demonstrate the applicability and the effectiveness of the presented methods for the task at hand. Moreover, both training and ground truth test data sets are publicly available together with a set of tools to perform evaluations on these data sets.

Local Descriptors

Generally, 3-D local descriptors are developed for specific applications such as registration, object recognition, and local surface categorization. For such applications, each point is associated with a descriptor describing the local geometry of a point.

Signature of Histograms of Orientations

The signature of histograms of orientation (SHOT) descriptor [2] encodes a signature of histograms representing topological traits, making it invariant to rotation and translation and robust to noise and clutter. The descriptor for a given key point is formed by computing local histograms incorporating geometric information of point locations within a spherical support structure. For each spherical grid sector, a one-dimensional histogram is constructed by accumulating point counts of the angle between the normal of the key point and the normal of each point belonging to the spherical support structure. The final descriptor is formed by orderly juxtaposing all histograms together according to the local reference frame.

Discrete quantization of the sphere introduces a boundary affect when used in combination with histograms, resulting in abrupt changes from one histogram bin to another. Therefore, quadrilinear interpolation is applied to each accumulated element, resulting in an evenly distribution into adjacent histogram bins. Finally, for better robustness toward point-density variations, the descriptor is L_∞-normalized. The dimensionality of the used signature is 352. Algorithm 1 gives some guidelines on how to compute SHOT descriptors using PCL. In this algorithm, first, an object of the class pcl::SHOTEstimation is created using the template parameters of our input data (cloud and normals). A kd-tree is provided to the estimator to perform NN searches. Indices represent the key points from the input cloud where SHOT descriptors should be computed. The radius search represents the size around each key point that will be described. Finally, calling the compute method returns a point cloud with as many descriptors as key points.

Algorithm 1. Computing SHOT descriptors in PCL.

```
pcl::SHOTEstimation< ... > shot;
shot.setSearchMethod(tree); //kdtree
shot.setIndices(indices); //keypoints
shot.setInputCloud(cloud); //input
shot.setInputNormals(normals); //normals
shot.setRadiusSearch(0.06); //support
shot.compute(*shots); //descriptors
```

Fast Point Feature Histogram

Descriptors such as point feature histogram (PFH) [3], fast PFH (FPFH) [4], and viewpoint feature histogram (VFH) [5] can be categorized as geometry-based descriptors. These descriptors represent the relative orientation of normals, as well as distances, between point pairs. Point pairs are generated between a point with coordinates p_i and the points in its local neighborhood with coordinates p_j. These point pairs are represented with the angles α, ϕ, and θ, computed in a reproducible fixed coordinate frame. The coordinate frame is constructed using the surface normal n_i of p_i, the vector $(p_i - p_j)/(\|p_i - p_j\|_2)$ and the cross product of these two vectors:

$$u = n_i, \tag{1}$$

$$v = u \times \frac{p_i - p_j}{\|p_i - p_j\|_2}, \tag{2}$$

$$w = u \times v. \tag{3}$$

With these, we can calculate the angles α, ϕ, and θ as follows:

$$\alpha = v^T \cdot n_i, \tag{4}$$

$$\phi = u^T \cdot \frac{p_i - p_j}{\|p_i - p_j\|_2}, \tag{5}$$

$$\theta = \arctan(w^T \cdot n_i, u^T \cdot n_i). \tag{6}$$

In the case of PFH, the normal orientation angles are computed for all point pairs of p and its neighborhood. To form the descriptor, the estimated values are binned into a histogram of size 33, representing the divisions of the feature space. The FPFH descriptor can be computed using PCL using Algorithm 1 and simply replacing the pcl::SHOTEstimation instantiation with a pcl:FPFHEstimation instantiation.

Algorithm 2. Computing 3-D SC descriptors in PCL.

```
pcl::ShapeContext3DEstimation< ... > dsc;
dsc.setSearchMethod (tree);
dsc.setIndices (indices);
dsc.setInputCloud (in);
dsc.setInputNormals (normals);
dsc.setRadiusSearch (radius_);
dsc.setMinimalRadius (radius_/10.f);
dsc.setPointDensityRadius (radius_/5.0);
dsc.compute (*dscs);
```

3-D Shape Context

The 3-D shape context (SC) descriptor [6] is an extension of the 2-D SC descriptor [7] into 3-D. The descriptor for a given key point is computed utilizing a 3-D spherical grid, centered on the key point, and superimposed over all points in the point cloud. Within the support structure, a set of sectors is defined by means of subdivisions along the azimuth, elevation, and radial dimensions, equally spaced for the first two dimensions and logarithmically spaced for the radial dimension. The final descriptor is built from the subdivision of the grid structure and is represented in bins of a 3-D histogram. Each bin of the histogram is associated with a value and computed as the weighted sum of the number of points falling in the corresponding grid sector. Additionally, the computed weight is inversely proportional to the local point density and the bin volume.

The north pole of the 3-D grid is aligned with the normal computed on the key point being described, while no principal direction is defined over the normal plane. This remaining degree of freedom is dealt with by taking into account as many orientations of the grid as the number of azimuth subdivisions, leading to several local reference frames and, in turn, to multiple descriptions for the same key point to deal with rotations and translations of the point cloud. Algorithm 2 shows how to compute the 3-D SC descriptor using PCL. Again, observe the similarity between this algorithm and the one presented in Algorithm 1, requiring the user to set an additional two parameters.

Unique SC

The unique SC (USC) [8] descriptor is an extension of the 3-D SC descriptor [6]. It defines a local reference frame for each key point so as to provide two principal directions over the normal plane that uniquely orientate the 3-D grid associated with each descriptor. This reduces the required memory footprint and possible ambiguities during the successive matching and classification stage because it avoids multiple descriptions for the same key point. In particular, the local reference frame is defined as in [2]. Once the 3-D spherical grid is uniquely oriented according to the local reference frame, the 3-D histogram yielding the final descriptor is built following the approach proposed in [6]. To compute USC descriptor using PCL, a variant of Algorithm 2 can be used, obtained by replacing pcl::ShapeContext3DEstimation for pcl::UniqueShapeContext.

Radius-Based Surface Descriptor

Radius-based surface descriptor (RSD) describes the geometry of points in a local neighborhood by estimating their radial relationships. The radius can be estimated by assuming each point pair to lie on a sphere and finding its radius by exploiting that the relation between the distance of the points d and the angle between the two point normals α is described by

$$d(\alpha) = \sqrt{2r}\sqrt{1-\cos(\alpha)} \approx r\alpha + r\alpha^3/24 + O(\alpha^5). \quad (7)$$

This relation turns out to be nearly linear for $\alpha \in (0, \pi/2)$ with the radius r (in meters) as the scaling factor. To accurately estimate the minimum and maximum radii r in a neighborhood, linear regression is applied on the minimum and maximum (α, d) pairs. The radius takes its extreme value, which is infinity, for a planar surface and lower values for surfaces with a higher curvature. As an example, using the minimum and maximum radius of a neighborhood allows to distinguish between spheres and cylinders. For a cylinder, the minimum radius will have a value close to the actual cylinder radius whereas the maximum value will be infinite. A detailed description and analysis of the method can be found in [9].

In our experiments, we use the local feature for matching; to do so, we use the complete histogram as the feature with 289 values. However, roughly half are nonzero due to the limited variability of nearby surface normals that are estimated by the current method in PCL. The RSD descriptor can be computed using PCL using Algorithm 1 and simply replacing the pcl::SHOTEstimation instantiation for a pcl::RSDEstimation instantiation.

Spin Images

Spin images (SIs) [10] can be used for point clouds with normal information available for every point. The approach operates on a rejection strategy, computing the difference of the point normal and the source normal n and considers only points with the difference smaller than a specific threshold. Furthermore, two distances are computed for each of the points: the distance from n and the distance from the source point along n. As a final step, a histogram is formed by counting the occurrences of different discretized distance pairs. In our implementation, we used a 153-dimensional descriptor. The SI descriptor can be computed in PCL using Algorithm 1 and simply replacing the pcl::SHOTEstimation instantiation for pcl::SpinImageEstimation.

Global Descriptors

The global object descriptors are high-dimensional representations of object geometry and were engineered for the purpose of object recognition, geometric categorization, and shape retrieval. They are only usually calculated for subsets of the point clouds that are likely to be objects.

Algorithm 3. Computing a global PFH descriptor in PCL on an input cloud.

```
pcl::PFHEstimation<...>pfh;
pfh.setSearchMethod(tree);
pfh.setInputCloud(input);
pfh.setInputNormals(normals);
pfh.setRadiusSearch(cloud_radius);
pfh.setIndices(indices);
pfh.compute(*pfh_signature);
```

These object candidates are obtained through a preceding segmentation step in the processing pipeline.

Point Feature Histogram

As explained earlier, PFH [3] is constructed by calculating the angles α, ϕ, and θ for all the point pairs in a neighborhood and binning them into a histogram. Generally, the number of bins required for a particular division of the features space into *div* divisions for each dimension is div^D, where D is the dimensionality of the feature space.

In the original article, in addition to the normal angles, the distance between point pairs was used, which turned out not to be a discriminative feature. The PCL implementation does not use this feature; therefore, the required number of bins only grows as a power of three. As a consequence, it becomes computationally feasible to divide the dimensions into five divisions, which leads to a 125-dimensional feature vector.

This feature space captures the correlations between the normal angles of point pairs in a neighborhood and was used both as a local and a global feature, and various other features are based on it. Here we use it as a global feature by considering every point pair in a cluster for building the histogram. Algorithm 3 shows how to compute a PFH descriptor using PCL. To obtain a global description of the input cloud, setRadiusSearch needs to be set according to the maximum distance between any two points on the input cloud, and indices contain only a single value in the range of the number of points.

Viewpoint Feature Histogram

The viewpoint feature histogram (VFH) introduced by Rusu et al. [5] is related to the FPFH feature. Here, the angles α, ϕ, and θ are computed based on a point's normal and the normal of the point cloud's centroid p_c. The viewpoint-dependent component of the descriptor is a histogram of the angles between the vector $(p_c - p_v)/(\|p_c - p_v\|_2)$ and each point's normal. It makes the histogram useful for joint object recognition and pose estimation, the purpose for which it was originally developed. The other component is a simplified point feature histogram (SPFH) estimated for the centroid of the point cloud and an additional histogram of the distances of the points in the cloud to the cloud's centroid. In the PCL implementation, each of those four histograms have 45 bins and the viewpoint-dependent component has 128 bins, totaling 308 bins. Algorithm 4 shows how to compute a VFH descriptor using PCL. Because of the global nature of VFH, no key points need to be provided, and the output of the estimator will only be a single descriptor encoding the geometry of the whole input cloud and its viewpoint.

Algorithm 4. Computing a VFH descriptor in PCL on an input cloud.

```
pcl::VFHEstimation<...>vfh;
vfh.setSearchMethod(tree);
vfh.setInputCloud(input);
vfh.setInputNormals(normals);
vfh.compute(*vfh_signature);
```

Clustered Viewpoint Feature Histogram

The clustered viewpoint feature histogram (CVFH) presented in [11] is an extension to VFH, based on the idea that objects have a certain structure that allows to split them in a certain number N of disjoint smooth regions. Each of these smooth regions is then used independently to compute a set of N VFH histograms. For instance, the kth VFH histogram uses the centroid p_{c_k} and normal average n_{c_k} computed by averaging points and normals belonging to region k to build the coordinate frame from which the normal distributions of the object are computed. The smooth regions for CVFH are easily computed by 1) removing points on the object with high curvatures, indicating noise or borders between smooth regions and 2) performing region growing on the remaining points on *xyz* and normal space. CVFH outputs a multivariate representation of the object of interest and the histogram k is additive as long as p_{c_k} and n_{c_k} remain unchanged. This allows the feature to be more robust to segmentation artifacts, locally noisy surfaces, and occlusions as long as at least one of the k regions is still visible (see Figure 1). The authors proposed to make the feature scale dependent by avoiding the normalization over the total amount of points

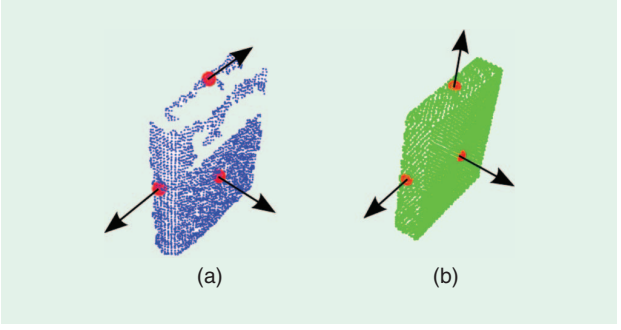

Figure 1. Centroids and normal averages for three regions of a book. (a) Point cloud obtained from the Kinect. (b) Similar viewpoint of the book used to train the recognition system.

Algorithm 5. Computing CVFH descriptors in PCL for an input cloud.

```
pcl::CVFHEstimation< ... > cvfh;
cvfh.setSearchMethod(tree);
cvfh.setInputCloud(input);
cvfh.setInputNormals(normals);
cvfh.setEPSAngleThreshold(angle);
cvfh.setCurvatureThreshold(max_curv);
cvfh.setNormalizeBins(false);
cvfh.compute(*cvfh_signatures);
```

on the object, allowing CVFH to differentiate between equally shaped objects from different sizes and, under occlusions, the shape of histograms only changes locally. The drawback is that the point clouds used for training and recognition need to share a common resolution so that the number of points describe the size of the object. From a computational point of view, the bottleneck to compute CVFH is the region-growing step with a cost $n\log(n)$, where n is the amount of points of the object to be described. Algorithm 5 shows how to compute the CVFH descriptors using PCL. Because of the semiglobal nature of CVFH, the number of resulting descriptors might be more than one depending on the geometry of the cloud to be encoded.

Ensemble of Shape Functions

The ensemble of shape function (ESF) descriptor introduced in [12] is an ensemble of ten 64-bin-sized histograms of shape functions describing characteristic properties of the point cloud. The shape functions consist of angle, point distance, and area shape functions. A voxel grid serves as an approximation of the real surface and is used to separate the shape functions into more descriptive histograms representing point distances, angles, and areas, either on the surface, off the surface, or both.

As shown in Figure 2, the ESF descriptor can be efficiently calculated directly from the point cloud with no preprocessing necessary, such as smoothing, hole filling, or surface normal calculation, and handles data errors such as outliers, holes, noise, and coarse object boundaries gracefully. Algorithm 6 shows how to compute an ESF descriptor using PCL. The ESF algorithms differ from the other feature algorithms as it does not require the use of normals to describe the cloud.

Figure 2. Calculation of the shape functions on an example point cloud of a mug. Point distances classified into on the surface, off surface, and both.

Algorithm 6. Computing an ESF descriptor in PCL on an input cloud.

```
pcl::ESFEstimation< ... > esf;
esf.setInputCloud(input);
esf.compute(*esf_signature);
```

Global RSD

Global RSD (GRSD) is the global version of the RSD local feature, which describes the transitions between different surface types (and free space) for an object, constructed by using the estimated principle radii for labeling the local surface types. As described in [9], it is based on a voxelization of the input point cloud, and it differs from the global FPFH (GFPFH) feature by eliminating the need for a complicated local surface classifier through the use of RSD as described earlier. Additionally, RSD is computed only once per voxel cell, drastically reducing the number of neighborhoods that need to be analyzed.

For this work, we counted the transitions between surface types around occupied voxels instead of along lines between occupied voxels. This modified version, noted as GRSD, makes the computation of the transition histogram linear in the number of cells. Together with the computation of RSD for each cell, the computation is linear in the number of points.

As the feature is independent of resolution, the normalization of the histogram can be avoided, making it dependent on scale and additive. By using five local surface types and considering free space, the number of used GRSD histogram bins is 20.

Recognition Pipelines

In this section, we will present how 3-D shape descriptors can be used in combination with other components to build a full recognition system, outputting the objects present in a range image together with their position and orientation (see Figure 3). Without loss of generality, this section follows our specific scenario in which the representation of world objects is given by 3-D meshes or CAD models. However, it also applies to other scenarios, as the training stage of a system is the only affected part in the pipelines. First, we present common components of both the global and local descriptors-based pipelines and then move to specific parts of each processing pipeline.

Training

To train the system, the meshes representing our objects are transformed to partial point clouds, simulating the input that will be given by a depth sensor. For this purpose, a virtual camera is uniformly placed around the object on a bounding sphere with a radius large enough to enclose the object. To obtain uniform sampling around the object, the sphere is generated using an icosahedron as starting shape and subdividing each triangular face with four equilateral triangles. This is done recursively for each face until the desired level of

recursion is reached. The level of recursion indicates how many triangles will shape the approximated sphere. The resulting triangles are used to place the camera at their baricenter, and a partial view of the mesh is obtained by sampling the depth buffer of the graphic card. There are two main parameters that govern this process: 1) the level of recursion and 2) the resolution of the synthetic depth images. According to our experiments, one level of recursion is enough to sample the view space (80 views are generated), and a resolution of 150 $\times$ 150 gives an appropriate level of detail over the object. This functionality is part of the visualization library of PCL. After the views are generated for each model, the desired 3-D feature is computed on the partial views and the descriptors are stored for future use. Moreover, a transformation between model and view coordinates is also saved. The inverse of such a transformation allows to transform the view to model coordinates, and, therefore, the full 3-D model can be obtained by applying the transformation for each view and fusing the transformed views together. Finally, in the case of global 3-D features, a camera roll histogram [11] to retrieve a full 6-DoF pose is also computed for each view and stored, as detailed in "Correspondence Grouping" section. Figure 4 shows the block diagrams for the local and global 3-D pipelines, respectively, which can be found in PCL and were used for the experimental results presented in this tutorial.

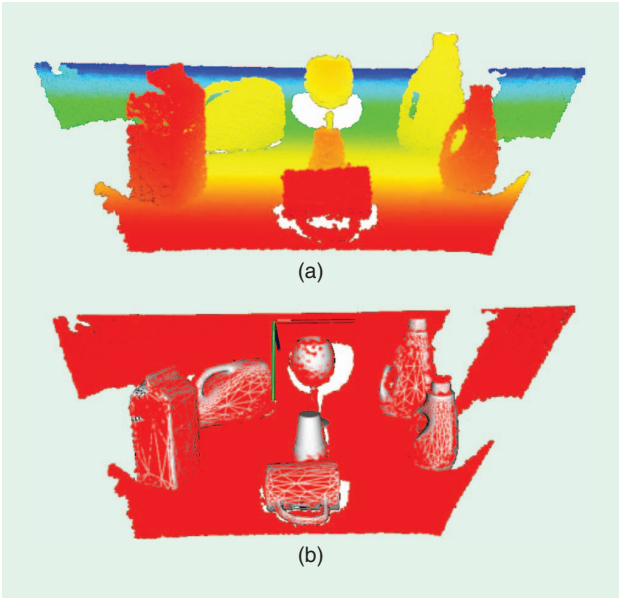

Figure 3. (a) Point cloud obtained from Kinect, colors from red to blue representing increasing depth and (b) recognition results given by the local recognition pipeline. The recognition hypotheses have been postprocessed using the hypotheses verification stage proposed in [13]. The corresponding 3-D meshes of the recognized objects are overlaid as wireframes after being transformed by the 6-DoF pose given by the recognition processing pipeline.

Recognition Pipeline for Local Descriptors

Key Point Extraction

The first step for every 3-D perception pipeline based on local descriptors is represented by the extraction of 3-D key points from data. The topic of 3-D key point detection is new, and recently, there have been several proposals in literature addressing this problem [14]. The main characteristics of a key point detector are repeatability and distinctiveness (other than computational efficiency). The former property deals with the capability of a detector to accurately extract the same key points under a variety of nuisances, while the latter is the ability to extract key points that can be easily described, matched, and/or classified, highly characterizing a surface or a specific object.

Importantly, the distinctiveness of a key point detector depends on the subsequent description stage: a set of key points can be more or less salient depending on the traits of the local descriptors applied on them. For this reason, to conduct a fair comparison among local descriptors, we have decided not to adopt any particular technique; instead, we use a uniform key point sampling over the point cloud, analogously to what was proposed in the 3-D object recognition pipeline of [10]. Specifically, in our experiments, key points are sampled using 3-D voxelization with a width of 15 mm.

Description and Matching

Once key points are extracted, they are associated to a local description. A common trait of all local descriptors is the definition of a local support used to determine the subset of neighboring points around each key point that will be used to compute its description. For fairness, the support size is set, for all methods, to the same value (6 cm in our experiments—see Figure 5. As for the other parameters

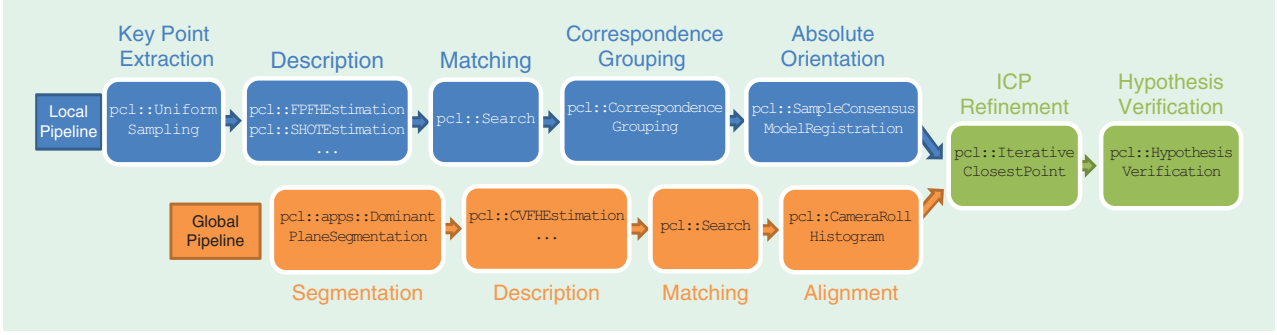

Figure 4. Block diagrams for the PCL local and global 3-D pipelines.

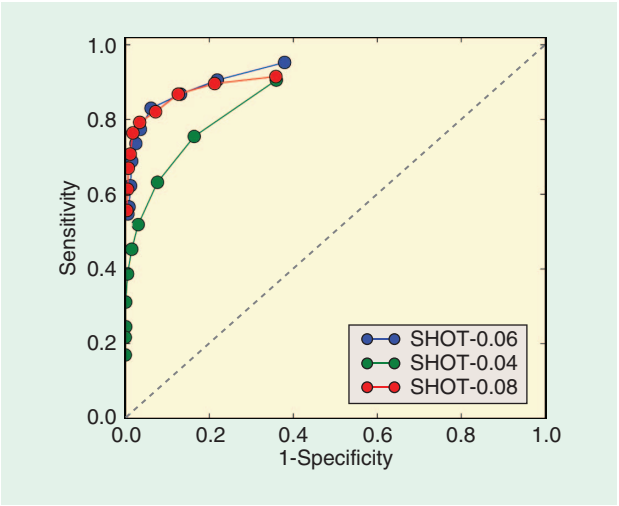

Figure 5. Recognition performance (without pose) using different support sizes (4, 6, and 8 cm) for the SHOT descriptor evaluated on the test data set without occlusions. From the figure, a support size of 4 cm does not seem to hold enough descriptiveness, while 6 and 8 cm sizes deliver similar results. Therefore, a choice of 6 cm seems reasonable to better deal with occlusions and clutter without affecting descriptiveness.

used for each specific technique, we have resorted to those proposed by the authors of each method.

Once descriptors are computed for the current scene and each model in the library, they need to be matched to yield point-to-point correspondences. To handle the case of multiple instances of the same model in each scene, each scene descriptor is matched against the descriptors of all models in our database (and not vice versa). More specifically, descriptors are compared using the Euclidean distance, and correspondences are set between each scene descriptor and its nearest neighbor (NN) in the model database. A matching threshold is commonly used to discard correspondences pairing descriptors that lie far apart in the descriptor space. In contrast, we do not prune any correspondence in our pipeline (equivalent to using an infinite value for the threshold), leaving the task of selecting consistent model hypotheses to the following correspondence grouping stage. This has the beneficial effect of eliminating one sensible parameter from the pipeline. Finally, efficient approximated matching schemes are deployed to speed up the matching stage: in particular, the fast approximate NN (FLANN) algorithm was used [15].

Correspondence Grouping

As a result of the matching stage, point-to-point correspondences are determined by associating pairs of model-scene descriptors that lie close in the descriptor space. A relatively common approach within 3-D object recognition methods is represented by an additional stage, usually referred to as *correspondence grouping*, where correspondences are discarded by enforcing geometrical consistency between them. More specifically, by assuming that the transformation between the model and its instance in the current scene is rigid, the set of correspondences related to each model is grouped into subsets, each one holding the consensus for a specific rotation and translation of that model in the scene. Subsets whose consensus is too small are discarded, simply by thresholding based on their cardinality.

A few methods were proposed in literature for this task (see [10], [16], and [17] and citations therein). In this comparison, we have used an approach similar to that proposed in [18], where an iterative algorithm based on simple geometric consistency of pairs of correspondences was proposed. In our approach, all correspondences are clustered into subsets geometrically consistent, i.e., starting from a seed correspondence $c_i = \{p_i^m, p_i^s\}$ (p_i^m and p_i^s being, respectively, the model and scene 3-D key points associated with c_i) and looping over all correspondences that were not yet grouped, the correspondence $c_j = \{p_j^m, p_j^s\}$ is added to the group seeded by c_i if the following relation holds:

$$\left| \left\| p_i^m - p_j^m \right\|_2 - \left\| p_i^s - p_j^s \right\|_2 \right| < \varepsilon, \tag{8}$$

with ε being a parameter of this method, intuitively representing the consensus set dimension. After parameter tuning, we set ε to 8 mm in our experiments.

As previously mentioned, a threshold needs to be deployed to discard correspondence subsets supported by a small consensus: a minimum of three correspondences is needed to compute the 6-DoF pose transformation. As this threshold trade-offs the number of correct recognitions (true positives) for the number of wrong recognition [false positives (FPs)], we have used it to span the reported ROC curves concerning the local pipeline.

Random Sample Consensus and Absolute Orientation

The previous geometric validation stage, although generally effective in discarding a high number of geometrically inconsistent correspondences, does not guarantee that all correspondences left in each cluster are consistent with a unique 6-DoF pose, i.e., with a unique 3-D rigid rotation and translation of the model over the scene. For this reason, it is useful to include an additional step based on a M-estimator, such as random sample consensus (RANSAC) whose aim is to additionally reduce the number of left correspondences in each cluster by eliminating those not consistent with the same 6-DoF pose. The model on which the consensus of the estimator is iteratively built is generally obtained by means of an absolute orientation algorithm. Given a set of exact correspondences between two point clouds, it determines the 3×3 rotation matrix ($\tilde{\mathbf{R}}$) and the 3-D translation vector ($\tilde{\mathbf{T}}$), which define the rigid transformation that best explains them. Because no outliers are assumed in the correspondence set, this is usually solved via least square minimization: given a set of N exact correspondences $c_1, \ldots, c_N$, $\tilde{\mathbf{R}}$ and $\tilde{\mathbf{T}}$ are obtained as

$$\arg\min_{\mathbf{R},\mathbf{T}} \sum_{i=1}^{N} \left\| p_i^s - \mathbf{R} \cdot p_i^m - \mathbf{T} \right\|_2^2. \tag{9}$$

Between the several absolute orientation algorithms in literature [19], we used the one proposed in [20], where the rotational component of the transformation is represented as a unit quaternion, given its effectiveness demonstrated in the comparison presented in [19].

Recognition Pipeline for Global Descriptors

Segmentation

Because global features require the notion of object, the first step in the recognition pipeline involves a segmentation of the scene to extract the objects in it. Segmentation in itself is a well-researched topic and several techniques exist [21]–[24]. In this case, we chose a simple but effective method based on the extraction of a dominant scene plane and a Euclidean clustering step (similar to flood filling) on the remaining points after dominant plane removal. The clustering step is guided by a threshold t, which indicates how close two points are required to be to belong to the same object. Therefore, for a successful segmentation, the method requires different objects to be at least t far away from each other. The method has a strong prior because of its assumption of a dominant plane like a table or a floor. Nevertheless, such an assumption is commonly used in robotic scenarios where the structure of human-made environment is exploited. PCL provides several components to perform such a task and the details of this segmentation method can be found in [25].

Description and Matching

The output of segmentation represents the potential objects in the scene. The shape and geometry of each of these objects is described by means of a proper global descriptor and represented by a single histogram (except for CVFH where multiple histograms might be obtained as presented in "Clustered Viewpoint Feature Histogram" section). In all cases, the histogram(s) are independently compared against those obtained in the training stage, and the best N NNs are retrieved. The N NNs represent the N most similar views in the training set according to the description obtained by means of the desired global feature and the metric used to compare the histograms. $N > 1$ neighbors instead of the single NN are usually retrieved, as the results can be greatly improved through pose estimation and postprocessing. Nevertheless, a descriptive and distinctive feature helps to efficiently prune lots of unfeasible candidates.

The histogram matching is done using Fast Library for Approximate Nearest Neighbors (FLANN) by means of a brute force search, and the distance between histograms is computed using the L1 metric (except for CVFH where the metric proposed in [11] is used) as it is more robust than L2 when dealing with outliers. Brute force might be used in this case without affecting global performance as the number of histograms in the training set is much smaller than in the case of local descriptors.

Camera Roll Histogram and 6-DoF Pose

The partial view candidates obtained by matching an object cluster to the training set can be aligned using their respective centroids obtaining a 5-DoF pose. A last degree of freedom involving a rotation about the roll axis of the camera needs to be computed for a full 6-DoF pose. Global features are usually invariant to this rotation due to the fact that rotations about the roll axis of the camera do not result in changes of the geometry of the visible part of the object (assuming no occlusions). Therefore, we adopt the approach presented in [11], where the roll rotation is obtained by means of the camera roll histogram and an optimization step that efficiently aligns two histograms. The rotation obtained can be then applied to the 5-DoF pose to obtain a full 6-DoF pose. This step is applied for each of the N candidates obtained from the previous step.

Postprocessing

Both the local and the global pipelines can undergo an optional postprocessing stage to improve the outcome of the recognition. One first step within this stage is usually represented by applying iterative closest point (ICP) to the recognition hypotheses, with the aim of refining the estimated 6-DoF pose. A following step, usually referred in literature as hypotheses verification, aims at reducing the number of FPs while retaining correct recognitions. This is generally carried out by leveraging on geometrical cues that can be efficiently computed once the model hypotheses have been aligned to the scene. Typical cues are the percentage of supporting points (i.e., model points that are close to scene points), as well as the percentage of outliers (number of visible points belonging to the models that do not have a counterpart within the scene points). Currently, PCL contains an implementation of the hypothesis verification algorithm proposed in [13]. Figure 3 shows an example where the recognition hypotheses are postprocessed using this method. Other verification strategies have been proposed in the literature [17], [26].

Another hypothesis verification method especially conceived for global pipelines [11] is based on reordering the N candidates list using a metric based on the number of inliers between the object cluster in the scene and the aligned training views. The number of inliers can be efficiently computed by counting how many points in the object cluster have a neighbor on the candidate views within a certain distance that we set to 5 mm. From Figure 6, it is clear that postprocessing and reordering the candidates help to obtain straighter recognition accumulated rate graphs. The higher the difference between first rank and nth rank, more the advantage a feature can obtain from such a postprocessing.

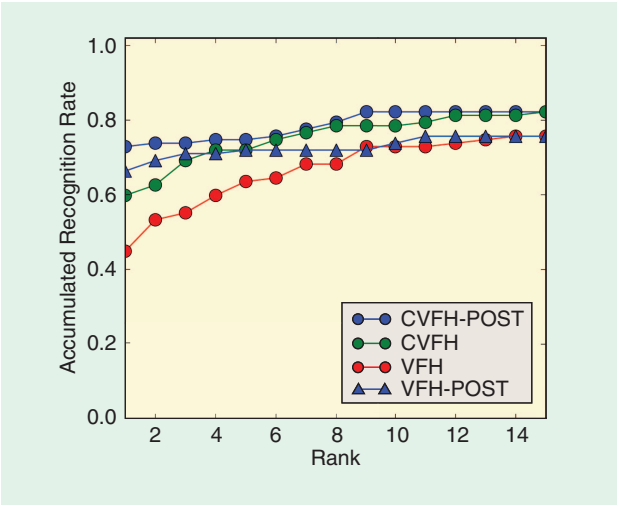

Figure 6. Accumulated recognition rate for VFH and CVFH both with and without postprocessing.

Evaluation of CAD Recognition and 6-DoF Pose Estimation

Training Data Set

The database was constructed using physical objects that are generally available from major retailers. The objects can be divided into three categories: for the first two categories, all objects were obtained from a single retailer (IKEA and Target, respectively) while the third category contained a set of household objects commonly available in most retail stores.

Test Data Set

For the experimental evaluation, we have collected several scenes and labeled them with ground truth models and 6-DoF pose. The data set has been split into two different sets: one where the objects in the scene can be segmented using the aforementioned segmentation method and without occlusions and the second set where the objects might be touching each other and present different degrees of occlusions. The sets contain 31 and 23 scenes, respectively. The full 3-D model meshes, the scene point clouds obtained from the Kinect, and the ground truth annotations can be downloaded from http://users.acin.tuwien.ac.at/aaldoma/datasets/RAM.zip .

Experiments

To better demonstrate the practical applicability of the presented features and pipelines, an evaluation has been performed on the aforementioned data sets. Specifically, we have carried out several experiments regarding recognition and pose estimation. Due to the segmentation step required by global features, they were evaluated only on the first data set while local features were evaluated on both data sets. Figure 7 presents a subset of the objects that were used in the data set.

Figures 8 and 9 show the ROC curves concerning the comparison among local features. From these results, it can be observed that the SHOT and USC features consistently outperform other approaches on both data sets. Three-dimensional SC and FPFH performs as well reasonably good although FPFH performance tends to sink under the presence of occlusions and nearby objects, as it can be clearly observed in Figure 9.

It is also worth noting that, from a practical point of view, the dimensionality of the descriptor also plays an important role. For instance, USC standard size of 1,980 required about 6 GB of memory when the whole training descriptors were loaded. For larger data sets, this implies the need of out-of-core solutions to perform the feature-matching stage, this having an impact also on the efficiency of the recognition pipeline. FPFH presents an important advantage regarding this factor due to its low dimensionality, being just made of 33 bins. The SHOT descriptor, with

Figure 7. A subset of the objects used in the recognition experiments. The picture shows a screenshot of the point cloud obtained by the Kinect sensor with registered RGB information.

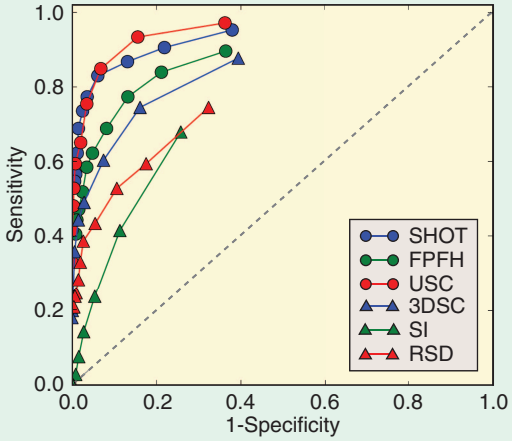

Figure 8. ROC curve for recognition (no pose) using local features in the set without occlusions spanning on the size of the consensus set.

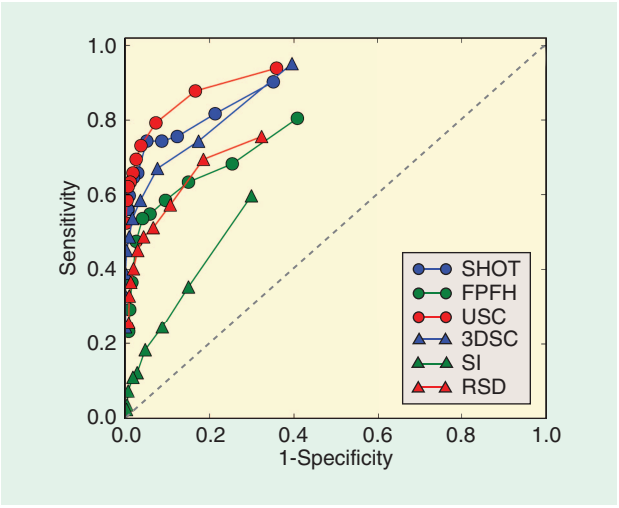

Figure 9. ROC curve for recognition (no pose) using local features in the set with occlusions spanning on the size of the consensus set.

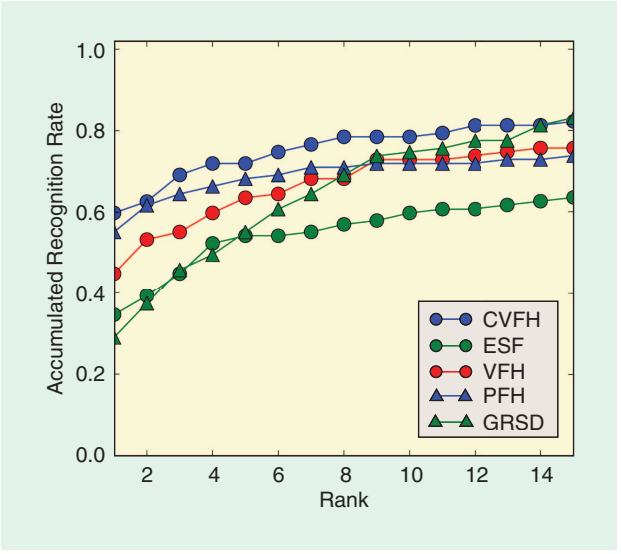

Figure 10. Accumulated recognition rate for global features.

a dimensionality of 352, represents a good tradeoff between recognition accuracy and memory footprint, as well as in terms of matching efficiency.

Another important factor is represented by the amount of false positives (FPs) being yielded by the local recognition pipeline. This is especially true when the recognition thresholds are set low enough to maximize the number of correct recognitions. From Figures 8 and 9, this seems to be a common trait among all local features and motivates the use of strong hypotheses verification strategies to reduce the number of FPs while retaining correct recognitions.

For what concerns global features, the results are presented in Figures 6 and 10 and Table 1. As stated earlier, global features were only evaluated on the test data set without occlusions, where the objects can be segmented by means of dominant plane extraction and Euclidean clustering. Figure 10 presents a direct comparison among all global features. Features using normal information to describe the surface of the objects seem to present a clear advantage over ESF in this data set. By explicitly taking into account the size of the objects and the viewpoint information, CVFH is the best performing feature.

Figure 6 presents how the recognition results can be improved by means of a suitable postprocessing over a reduced number of candidates obtained directly out from the matching stage. As observed in Figure 10, GRSD recognition accuracy improves rapidly when several candidates are taken into account; however, the viewpoint of the candidates appears to be incorrect more often as seen from the root-mean-square error (RMSE) of the feature, which is about twice bigger as that of the other features (see Table 1).

The effect of the descriptor dimensionality is less important among global features due to the small number of descriptors required to describe a set of objects. This also results in an important speedup on the matching stage. Because of the segmentation step, the number of FPs is directly related to the recognition capacity of the feature. In other words, the amount of recognitions (false or true positives) is bounded by the number of objects found during the segmentation step. Nevertheless, global features might as well benefit from hypotheses verification stages to filter those objects that were incorrectly recognized.

Finally, we would like to mention that a direct, fair, and accurate comparison between global and local features/pipelines is difficult due to their different characteristics and assumptions. However, it is clear that in scenarios where segmentation becomes challenging or occluded objects need to be recognized, global features should be avoided. Conversely, in controlled environments where speed becomes an important factor, global features might represent a good choice.

Classification of Unknown Objects

A related problem is that of the generalization of the descriptors to unknown objects of similar shape. While a descriptive feature is preferred if many views of all objects are known a priori, some robustness to small changes is required if similar objects are to be detected (or the same objects under different noise levels or resolution). To provide the reader with an intuition of how well the

Table 1. Average RMSE for the different global features among those correctly recognized.

Feature	Average RMSE [Number of kTP]
CVFH	0.000262 [88]
VFH	0.000197 [81]
GRSD	0.000432 [85]
PFH	0.000220 [79]

Table 2. Classification accuracy on the RGB-D object data set.

k-NN	VFH	GRSD	PFH	(G)SHOT
Metric	[308d]	[20d]	[125d]	[352d]
L1	56.44 ± 2.66	43.93 ± 2.20	53.00 ± 1.64	55.73 ± 1.52
L2	52.75 ± 2.33	43.10 ± 2.20	51.31 ± 1.86	48.82 ± 1.42
Hellinger	57.20[1]	45.91 ± 2.00	55.95 ± 1.48	54.63 ± 1.75
χ^2	56.3 ± 2.86	45.95 ± 1.99	55.63 ± 1.49	55.63 ± 1.79

[1]Building the index took ~12 h for this metric, so it was evaluated for only one instance of the data set.

presented (global) features cope with these changes, we compared the results on a standard data set to previously published results and obtained similar results using a simple *k*-NN classifier.

For the empirical evaluation of the features for classification, we followed the method used by Lai et al. to allow a comparison of results. For training and evaluating the classifiers, we used each of the more than 200,000 scans of 300 objects from 51 categories from the database. The task is to learn a model of the categories in the data set, which generalizes to objects and were not part of the training set. For estimating the performance of the features, we evaluated each of them on ten different partitionings of the data set into training data and test data. To obtain a partitioning, all views of one of the objects from each category was chosen at random and removed from the data set. The removed objects formed the test partition and the remaining objects the training partition.

Before estimating the features, outliers were removed from the point clouds. We considered all point outliers whose mean distance to the 50 nearest points deviates more than three standard deviations from the global distance mean. The point clouds were further subsampled using a voxel grid filter with a grid size of 5 mm to speed up subsequent computations. An important step in estimating all of the point signatures is the estimation of point normals. In PCL, the estimation is done via principal component analysis on the covariance matrix obtained from the neighborhood of a point, and sign disambiguation is done by orienting all of the normals toward the viewpoint. For all the features under investigation, all points with a distance smaller or equal to 2 cm are considered to lie within the neighborhood of a point. We used the FLANN for our *k*-NN implementation with a K-means index with default parameters.

In Table 2, we give the mean accuracy as well as the accuracy's standard deviation for all the features and classifiers used. As CVFH computes a descriptor for each object part, it is not a fixed-length feature for a complete object, so integrating it for this classification task would not be straightforward. Thus, we replaced it with SHOT for this experiment, by computing a single SHOT feature for each object and using it as a global feature (we computed SHOT for the centroid using the complete cloud as the neighborhood).

As a comparison, the results reported in [27] are 53.1 ± 1.7 using linear support vector machines (SVM) as classifier, 64.7 ± 2.2 using a Gaussian kernel SVM, and 66.8 ± 2.5 using random forests—all using efficient match kernel [28] features using random Fourier sets computed on SIs. Thus, the features themselves perform well for 3-D shape-based object categorization but more advanced classifiers are needed for obtaining better results.

Conclusion

We have presented a survey of local and global 3-D shape features publicly available in the PCL. Moreover, we argued that a good feature is a small but important piece of any object recognition system and, therefore, presented two different complete pipelines (using local and global features, respectively) built up from different modules available in PCL.

We have experimentally shown how the different pipelines can be used to recognize objects and estimate their 6-DoF pose in real scenes obtained with the Kinect sensor. The global features have also been evaluated on a public data set for the task of object classification providing good results with a simple NN classifier.

We believe that the use of 3-D shape-based features and recognition pipelines able to yield 6-DoF poses have a strong value in many robotic scenarios and applications and hope that the presented results and available tools will be useful for the reader when designing a 3-D perception system.

Acknowledgments

The authors thank Kevin Lai for granting them access to the RGB-D object data set [27] and Florian Seidel for his help regarding the evaluation in the "Classification of Unknown Objects" section. Special thanks to the PCL community (see http://pointclouds.org/about.html) as well as to the funding institutions of the authors of this article.

References

[1] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China, May 9–13, 2011, pp. 1–4.

[2] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proc. 11th European Conf. Computer Vision (ECCV '10)*, 2010, pp. 356–369.

[3] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc 21st IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Nice, France, Sept. 22–26, 2008, pp. 3384–3391.

[4] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3-D registration," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Kobe, Japan, 2009, pp. 3212–3217.

[5] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3-D recognition and pose using the viewpoint feature histogram," in *Proc. 23rd IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 2010, pp. 2155–2162.

[6] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Proc. 8th European Conf. Computer Vision (ECCV 04)*, 2004, pp. 224–237.

[7] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 4, pp. 509–522, 2002.

[8] F. Tombari, S. Salti, and L. Di Stefano, "Unique shape context for 3-D data description," in *Proc. ACM Workshop on 3-D Object Retrieval (3-D OR '10)*, 2010, pp. 57–62.

[9] Z. C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2D-3-D Categorization and classification for multimodal perception systems," *Int. J. Robot. Res.*, vol. 30, no. 11, pp. 1378–1402, Sept. 2011.

[10] A. E. Johnson, "Spin-images: A representation for 3-d surface matching," Robotics Inst., Carnegie Mellon Univ, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-97-47, Aug. 1997.

[11] A. Aldoma, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, M. Vincze, and G. Bradski, "CAD-model recognition and 6 DOF pose estimation using 3D cues," in *Proc. ICCV 2011, 3D Representation and Recognition (3D RR11)*, Barcelona, Spain, 2011, pp. 585–592.

[12] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3-D object classification," in *Proc. IEEE Int. Conf. Robotics and Biomimetics (IEEE-ROBIO)*, 2011, pp. 2987–2992.

[13] C. Papazov and D. Burschka, "An efficient RANSAC for 3-D object recognition in noisy and occluded scenes," in *Proc. 10th Asian Conf. Computer Vision (ACCV)*, 2010, pp. 135–148.

[14] F. Tombari, S. Salti, and L. Di Stefano, "Performance evaluation of 3-D key point detectors," *Int. J. Comput. Vis.*, to be published.

[15] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Computer Vision Theory and Application (VISAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[16] F. Tombari and L. Di Stefano, "Hough voting for 3-D object recognition under occlusion and clutter," *IPSJ Trans. Comput. Vis. Appl. (CVA)*, vol. 4, pp. 20–29, Mar. 2012.

[17] A. S. Mian, M. Bennamoun, and R. A. Owens, "On the repeatability and quality of key points for local feature-based 3-D object retrieval from cluttered scenes," *Int. J. Comput. Vis.*, vol. 89, no. 2–3, pp. 348–361, 2009.

[18] H. Chen and B. Bhanu, "3D free-form object recognition in range images using local surface patches," *Pattern Recog. Lett.*, vol. 28, no. 10, pp. 1252–1262, 2007.

[19] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-D rigid body transformations: A comparison of four major algorithms," *Machine Vision Appl.*, vol. 9, no. 5-6, pp. 272–290, 1997.

[20] B. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Amer. A*, vol. 4, no. 4, pp. 629–642, 1987.

[21] N. Bergström, M. Björkman, and D. Kragic, "Generating object hypotheses in natural scenes through human-robot interaction," in *Proc.IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sept. 2011, pp. 827–833.

[22] A. K. Mishra and Y. Aloimonos, "Visual segmentation of 'simple' objects for robots," in *Proc.Robotics: Science and Systems (RSS)*, 2011.

[23] C. C. Fowlkes, D. R. Martin, and J. Malik, "Local figure-ground cues are valid for natural images," *J Vis.*, vol. 7, no. 8, pp. 1–9, 2007.

[24] D. Comaniciu, P. Meer, and S. Member, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 603–619, 2002.

[25] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3-D Point cloud maps for mobile manipulation in human environments," in *Proc. IEEE/RSJ Int. Conf.Intelligent Robots and Systems (IROS)*, St. Louis, MO, Oct. 11–15, 2009.

[26] P. Bariya and K. Nishino, "Scale-hierarchical 3-D object recognition in cluttered scenes," in *Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1657–1664.

[27] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *Proc. Int. Conf.Robotics and Automation (ICRA)*, 2011, pp. 1817–1824.

[28] L. Bo and C. Sminchisescu, "Efficient match kernels between sets of features for visual recognition," in *Proc. NIPS*, 2009.

Aitor Aldoma, Vision4Robotics Group, Automation and Control Institute, Vienna University of Technology, Vienna, Austria. E-mail: aa@acin.tuwien.ac.at.

Zoltan-Csaba Marton, Intelligent Autonomous Systems, Technische Universität München, Germany. E-mail: marton@cs.tum.edu.

Federico Tombari, Computer Vision Lab, DEIS, University of Bologna, Italy. E-mail: federico.tombari@unibo.it.

Walter Wohlkinger, Vision4Robotics Group, Automation and Control Institute, Vienna University of Technology, Vienna, Austria. E-mail: ww@acin.tuwien.ac.at.

Christian Potthast, Robotic Embedded Systems Laboratory (RESL) and Department of Computer Science, University of Southern California (USC), Los Angeles, CA 90089, USA. E-mail: potthast@usc.edu.

Bernhard Zeisl, Computer Vision and Geometry Group, ETH Zurich, Switzerland. E-mail: zeislb@inf.ethz.ch.

Radu Bogdan Rusu, Open Perception, Menlo Park, CA, USA. E-mail: rusu@openperception.org.

Suat Gedikli, Willow Garage, Menlo Park, CA, USA. E-mail: gedikli@willowgarage.com.

Markus Vincze, Vision4Robotics Group, Automation and Control Institute, Vienna University of Technology, Vienna, Austria. E-mail: vm@acin.tuwien.ac.at.