Evaluation of 3D Feature Descriptors for Classification of Surface Geometries in Point Clouds

Georg Arbeiter\textsuperscript{1}, Steffen Fuchs\textsuperscript{1}, Richard Bormann\textsuperscript{1}, Jan Fischer\textsuperscript{1} and Alexander Verl\textsuperscript{1}

Abstract—This paper investigates existing methods for 3D point feature description with a special emphasis on their expressiveness of the local surface geometry. We choose three promising descriptors, namely Radius-Based Surface Descriptor (RSD), Principal Curvatures (PC) and Fast Point Feature Histograms (FPFH), and present an approach for each of them to show how they can be used to classify primitive local surfaces such as cylinders, edges or corners in point clouds. Furthermore these descriptor-classifier combinations have to hold an in-depth evaluation to show their discriminative power and robustness in real world scenarios. Our analysis incorporates detailed accuracy measurements on sparse and noisy point clouds representing typical indoor setups for mobile robot tasks and considers the resource consumption to assure real-time processing.

I. INTRODUCTION

Perception of the environment is crucial for the accomplishment of tasks by mobile service robots. Both for navigation and manipulation, a 3D representation of the robot’s surroundings is inevitable.

Current mobile service robots, such as the Care-O-bot\textsuperscript{©} 3, are designed to interact in everyday environments. The great diversity of such unstructured environments and the objects in them makes it difficult to provide models of all relevant objects and teaching every situation will never be achievable. Thereby adding semantic information in a more generic way to the sensor data can help the robot to perceive the complex world with more flexibility and handle new and unexpected situations more reliably.

With the continuous increase of computational capacities research areas dealing with 3D cognition problems become more and more appealing. Furthermore the introduction of the Microsoft Kinect camera, as being the first real time 3D sensing device in the low-cost section, caused a major boost for the development of applications using 3D perception. Thus a variety of 3D feature descriptor algorithms have evolved in the recent past. Promising similar properties and having common applications, there is yet no comparative evaluation of these methods available.

In this paper we investigate existing 3D feature descriptors that can be used to classify local surface geometries in point clouds. These local features use the information provided by a point’s $k$ closest neighbors to represent this point in a more discriminative geometrical way. Interpreting these estimated feature values by applying a specific classifier allows us to assign a label to each point that defines on which surface type the point lies. For the purpose of this work we differentiate between the following five basic surface types: plane ($P$), edge ($E$), corner ($Co$), cylinder ($Cy$), sphere ($S$).

These descriptor-classifier combinations have to hold against a series of test scenarios and will be evaluated in terms of accuracy and computation time. This evaluation is intended to show what capabilities and limitations each feature descriptor has regarding its potential to “classify the world”. The scenario point clouds are exclusively acquired from PrimeSense cameras. This puts a special requirement on each descriptor to sustain the device’s typical noise and quantization errors [1].

The remainder of the paper is structured as follows: Section II provides an overview of existing descriptors and current related work. Section III explains about the methods used for feature estimation. The specific approaches to interpret these descriptor values are presented in Section IV. In Section V we show the implementation details for our benchmark setup. Results are presented and discussed in Section VI.

II. RELATED WORK

In the recent past many feature types for point clouds have been proposed and most of them addressed problems of object recognition and point cloud registration. Some of them were ported successfully from the 2D domain, such as RIFT [2], others like spin images [3] or curvature maps [4] were adopted from the 3D mesh department.

Another popular family are descriptors belonging to the feature histograms. Inspired by the work of [5] the Point Feature Histograms (PFH) [6] were deployed for geometrical surface description and later on refined in terms of computation time under the name Fast Point Feature Histograms (FPFH) [7]. Further modifications exist as Global Fast Point Feature Histograms (GFPFH) [8] and Viewpoint Feature Histograms (VFH) [9] which put their emphasis on object recognition in a more global manner.

Spin Images and 3D Shape Contexts [10] are popular descriptors for object recognition tasks. Unique Shape Context [11] presents an improvement of the later one in terms of accuracy and reduced memory consumption. However, they are sensitive to sensor noise and require densely sampled data [12]. The RIFT descriptor and intensity-domain spin images [2] only work with intensity information provided for every point of the point cloud which is the case for most laser scanner systems but not for PrimeSense cameras. Also intensity values are stronger related to the surface texture

\footnote{The authors are with the Institute for Manufacturing Engineering and Automation, Fraunhofer IPA, 70569 Stuttgart, Germany <first name>.<last name> at ipa.fraunhofer.de www.ipa.fraunhofer.de}
than to the actual geometry, which does not really help to classify local shapes.

Two approaches relying on surface normal estimation are Principal Curvatures (PC) (provided by [13]) and the Radius-Based Surface Descriptor (RSD) [14]. Both derive local surface information from point normals in a local neighborhood. They have a strong potential provided that the normal estimation is robust against noise.

[6] and [15] propose two concepts of surface classification using PFH and RSD but both of them were tested with laser scanners only. Besides of those mentioned previously the majority of proposals either tackle the less generic problem of specific object recognition and fitting using large predefined data sets. Or they discuss the methods that focus on problems involving simple plane segmentation (e.g. [16]) while ignoring other shape types.

In the majority of the work mentioned, a comparative evaluation to other feature types is not performed. Most of the times, the descriptive power of the features is only shown in sample images instead of quantitative results and the scene selection is not sufficient both in variety and quantity.

In contrast, the work presented here intends to evaluate certain feature descriptors against each other. Key characteristics of the descriptors have to be (1) real-time processing, (2) robustness against noise, (3) no use of intensity data (as the data comes from a PrimeSense device), (4) the ability to describe local surface geometries and (5) an efficient open-source implementation. Regarding these prerequisites, we select RSD, FPFH and PC for an in-depth evaluation.

III. FEATURE DESCRIPTORS

The following section gives an overview of the investigated feature descriptors and presents a short explanation of their principals and characteristics. A prerequisite for all feature estimation algorithms is a point cloud \( P = \{p_1, p_2, ..., p_n\} \), with \( n \) feature points \( p_i \) where each feature point is a subset of \( m \) feature values \( p_i = \{f_1, f_2, ..., f_m\} \). In our case every feature point at least consists of the values \( p_i = \{p_i, n_i\} \), where

\[
p_i = [x_i, y_i, z_i]^T
\]

represents the 3D position vector of \( p_i \) and

\[
n_i = [n_{xi}, n_{yi}, n_{zi}]^T
\]

the local surface normal vector of \( p_i \). Since examining the best normal estimation algorithm is not part of this work, though a good representation of the surface normals is key to all algorithm investigated here. Therefore we went with the method suggested by [17] which performs a Principal Component Analysis on surrounding points, where the direction of the third component represents the surface normal.

In the following the \( k \) surrounding points, also called local neighborhood, of a point \( p_i \) are referred to as a subset \( P^k \) of points \( p_j \) (\( j \in \{1...k\} \)), where \( \|p_i - p_j\|_2 \leq r \) with \( \|\cdot\|_2 \) being the Euclidean distance and \( r \) a defined sphere radius. Collecting this set of points is an essential part of the algorithms described here and is carried out by using the same implementation of a fixed radius search for every descriptor.

After running the descriptor algorithm, each point \( p_i \) of the point cloud \( P \) is extended to \( p_i = \{p_i, n_i, d_i\} \), where \( d_i \) represents the estimated values of the used descriptor.

A. Radius-based Surface Descriptor

RSD as proposed in [14] describes the geometry of a point \( p_i \) by estimating the radius of the fitting curves to its local neighborhood \( P^k \). The feature values of each point consist of a maximum and minimum curvature radius taken from the distribution of normal angles by distance.

The problem of finding \( r_{\text{max}} \) and \( r_{\text{min}} \) can be solved by assuming that the relation between the distance \( d \) of two points and the angle \( \alpha \) between the points’ normals

\[
d(\alpha) = \sqrt{2r(1 - \cos \alpha)}
\]

can be simplified for \( \alpha \in [0, \pi/2] \) as

\[
d(\alpha) = r\alpha
\]

The estimated values at each point \( p_i \) finally are presented as \( d_i = [r_{\text{max}}, r_{\text{min}}] \).

B. Principal Curvatures

This feature describes the point’s local surface geometry as a measure of its maximum and minimum curvature along with a normalized vector indicating the first of the principal directions. This approach is very similar to the one RSD is based on and makes both descriptors closely related in terms of how they describe one point’s neighborhood. However the implementation of the PC estimation algorithm [13] differs somewhat from RSD.

All normals \( n_j \) of the neighborhood \( P^k \) are projected on the tangent plane of the surface defined by the normal \( n_q \) at the query point \( p_q \)

\[
m_j = (I - n_q \cdot n_q^T) \cdot n_j
\]

with \( I \) being a 3 \times 3 identity matrix. Computing the covariance matrix \( A \in \mathbb{R}^{3 \times 3} \) from all projections \( m_j \)

\[
A = \frac{1}{k} \sum_{j=1}^{k} (m_j - \bar{m})(m_j - \bar{m})^T
\]

where \( \bar{m} \) being the mean vector of all \( m_j \) and solving

\[
A \cdot x_l = \lambda_l \cdot x_l
\]

to find the non-zero eigenvectors \( x_l \) and their eigenvalues \( \lambda_l \), with \( l \in \{1, 2, 3\} \). If \( 0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3 \) then \( \lambda_3 \) corresponds to the maximum curvature \( c_{\text{max}} \) and \( \lambda_2 \) to the minimum curvature \( c_{\text{min}} \). Along with these values, the PC descriptor also provides the normalized eigenvector \( x_3 \) of the maximum curvature which results in the final representation of each point \( p_i = \{p_i, n_i, d_i\} \) with \( d_i = [c_{\text{max}}, c_{\text{min}}, x_3^T] \).
C. Fast Point Feature Histograms

Fast Point Feature Histograms [7] are a modification of Point Feature Histograms proposed in [6] and optimized in terms of computation time while retaining most of the discriminative power. A point’s FPFH is determined in two separated steps. In the first step, for each point $p_i$, a Simplified Point Feature Histogram (SPFH) is created by selecting the local neighborhood $P^k$. For every pair of points $p_i$ and $p_j$ ($i \neq j$, $p_i$ is the point with a smaller angle between its associated normal and the line connecting the points) in $P^k$, a Darboux $uvw$ frame ($u = n_i, v = (p_i - p_j) \times u, w = u \times v$) is defined. The angular variations of $n_i$ and $n_j$ are then calculated as

$$\cos(\alpha) = v \cdot n_j$$
$$\cos(\varphi) = \frac{(u \cdot (p_j - p_i))}{\|p_j - p_i\|_2}$$

$$\sigma = \text{atan2}(w \cdot n_j, u \cdot n_j)$$

stored in 11 bins for each angle normalized to 100, to form the 33 bin sized SPFH. In the second step all SPFHs in the neighborhood of $p_i$ are collected to form the actual FPFH:

$$\text{FPFH}(p_i) = \text{SPFH}(p_i) + \frac{1}{k} \sum_{j=1}^{k} \frac{1}{w_j} \cdot \text{SPFH}(p_j)$$

where $w_i = \|p_i - p_j\|_2$ is the applied weight depending on the distance to the query point $p_i$. The final descriptor values $d_i = [b_1, ..., b_33]$ are composed by the 33 bins of the weighted FPFH.

IV. Classifiers

The approaches presented in this section take the estimated values $d_i$ of the previously introduced descriptors to imply a certain class label. After classification the point cloud $P$ consists of feature points

$$p_i = \{p_i, n_i, d_i, l_i\}$$

where $l_i \in \{l_1, ..., l_k\}$ being one of the $k$ labels that was assigned.

A. Rules for RSD and PC

The idea behind the interpretation of RSD is based on the work of [15] which suggests to simply define several thresholds for the feature values of their proposed RSD descriptor to categorize surfaces. Based on several experiments with synthetic data, we applied a minor modified version of the originally proposed rule-set in favour to our requirements which results in slightly better differentiation of cylinder/sphere and edge/corner.

Since RSD and PC are based on the same geometrical approach by describing the highest and lowest curvature, this concept can also be transferred to classify values of PC (see Figure 1). In both cases edges and planes are located at opposite ends of one feature value (which is the minimum radius for RSD and maximum curvature for PC) and points in between are defined as curved. To distinguish further between curved points, another rule can be applied as a ratio between maximum and minimum values (being $r_{max}/r_{min}$ for RSD and $c_{max}/c_{min}$ for PC). The same principal works for corners and edges.

B. Support Vector Machine for FPFH

Support Vector Machines (SVM) is one of the supervised learning algorithms suggested in [6] to provide good results for FPFH classification. For being as close as possible to results proposed in [6] we also generated a variety of synthetical shape primitives featuring different sizes, point densities and noise levels. The noisy data was generated by adding random numbers to the X, Y and Z coordinates of each point according to the Gaussian distribution with a standard deviation $\sigma \in [0.0005, 0.002]$ ($[\sigma] = m$). We also differentiated between concave and convex types of edges, corners, cylinders and spheres.

Relying solely on synthetic training sets however did not show the results as expected which is caused by the fact that the original evaluation was performed on point clouds coming from LIDAR systems. The characteristics (in particular the quantization errors [1]) of point clouds acquired using a PrimeSense devices are very different to those using laser scanners and simulating these characteristics is more cumbersome. Therefore we additionally captured some real data scenes, labeled them manually and extracted the FPFH feature values for each class separately. The final training set composed as a medley of synthetic and real data was used to create a multi-class SVM in one-against-one manner.

V. BENCHMARK SETUP

In order to perform a meaningful evaluation, both scenario selection and measures have to be selected carefully. The scenarios should cover the range of desired applications and the measures have to be comparative and robust.

A. Scenarios

To provide results as close as possible to practical indoor applications, we exclusively used real data scenes for the evaluation. A total of 8 scenes which we equally separated into two range categories were captured with an ASUS Xtion PRO LIVE. The close range scenes represent a typical
setup for object identification and manipulation tasks and are evaluated up to a distance of 1.8 m. The far range scenes (see Figure 2) feature situations where an overview of the environment is needed, for example to find a certain drawer in the kitchen. Due to the quadratically increasing quanization error of the PrimeSense cameras, we restricted the distance to 3.0 m since everything beyond does not provide any useful information.

To provide the ground truth for every scene, we made use of the fact that PrimeSense cameras produce their point clouds organized. This allowed us to simply import depth and registered RGB images using a drawing program such as GIMP and colorize each pixel manually. Each class was represented by a particular RGB color code and then mapped back to point cloud\(^1\).

### B. Measuring Classification Accuracy

For classification tasks, the outcome of a classifier is commonly measured by comparing the expectations with the predicted results. For multi-class evaluation problems a typical representation is the confusion matrix \(A\), with \(A_{ij}\) for \(i, j \in \{1...l\} \) where \(k\) is the total number of labels and \(A_{ij}\) is the number of times a data point of the true label \(l_i\) was predicted as the label \(l_j\). In order to summarize each scene and to allow an easy comparison among them, we present our results using the following four measures. The \(micro\)-average results

\[
R_{\text{mic}} = P_{\text{mic}} = F_{\text{mic}} = \frac{\sum_i^k A_{ii}}{\sum_i^k \sum_j^k A_{ij}} = \frac{\sum_i^k tp_i}{\sum_i^k (tp_i + fn_i)}
\]

(11)

are the same for recall \(R_{\text{mic}}\), precision \(P_{\text{mic}}\) and F-measure \(F_{\text{mic}}\) and give the fraction of points predicted correctly to the total number of data points in the scene. Since our test scenarios represent typical indoor setups where the classes are not evenly balanced and the majority of points are located on planes (here: 75% - 95%), this measurement easily distorts the results to advantage for classifiers strong with planes. Therefore we also provide the three \(macro\)-averaged values for recall

\[
R_{\text{mac}} = \frac{1}{k} \sum_i^k A_{ii} = \frac{1}{k} \sum_i^k \frac{tp_i}{tp_i + fn_i}
\]

(12)

for precision

\[
P_{\text{mac}} = \frac{1}{k} \sum_i^k A_{ii} = \frac{1}{k} \sum_i^k \frac{tp_i}{tp_i + fp_i}
\]

(13)

and the \(F\)-measure

\[
F_{\text{mac}} = \left(1 + \beta^2\right) \cdot \frac{P_{\text{mac}} \cdot R_{\text{mac}}}{(\beta^2 \cdot P_{\text{mac}}) + R_{\text{mac}}}
\]

(14)

with \(\beta = 1\) as the harmonic mean of both. These values put an even weight on each class and give a more balanced result.

In addition to the investigation of all five classes we also examine the use case at every scene where only the discrimination of planes and edges from more complex shapes is required. For this purpose we consider edges and corners as being part of the same class (referred to as \(edges\)) as well as spheres and cylinders (referred to as \(curved\)) which reduces the evaluation problem to three classes.

### C. Implementation Details

All algorithms were implemented in C++ and investigated on an Intel Core i7-2600 CPU with 16 GB RAM, running Ubuntu 10.10 64 bit. Normal and feature estimation algorithms as well as Moving Least Squares smoothening were provided by PCL\(^2\) and OpenCV library\(^3\) was used to provide the implementation of the SVM.

### VI. RESULTS AND DISCUSSION

#### A. Computation Time

To investigate the computational complexity of each descriptor we measured the running time the estimation algorithms take to compute an entire point cloud consisting of 232,412 points depending on their local neighborhood radius. Since these measurements depend very much on the system they are running on, Figure 3 presents the measurements in percentage relative to PC as it turns out to be the fastest. In average RSD needs about 13 % and FPFH about 157 % longer than PC.

#### B. Accuracy

The outcome of each algorithm heavily depends on the correct adjustment of the configuration parameters for each individual scene. Whereas the parameters suggested in [6] work well for more accurate devices such as laser scanners, we could not obtain satisfying results with these for our setup and needed a different configuration. While small normal/feature radii tend to capture many details of the

---

\(^1\)The data set is available at http://www.care-o-bot-research.org/contributing/data-sets

\(^2\)http://pointclouds.org/

\(^3\)http://opencv.willowgarage.com/

---
scene, greater radii are more robust against sensor noise. In order to accomplish an evaluation close to practical use cases, we selected two different sets of configuration parameters. One set was used for all close range scenarios, the other one for all far range scenes. These values were determined by first testing a wide range of parameter combinations on every scene and then selecting the best trade-off for each category. Table I gives the final two parameter sets.

For the far range scenes we also found it beneficial to perform surface smoothing beforehand. For this purpose we used the Moving Least Squares method provided by PCL to apply a third order polynomial fitting after normal estimation.

The pictures in Table V visualize the outcome of the algorithms on all scenes. Table II presents the corresponding accuracy values while Table IV summarizes all scenes separated by classes.

The discriminating power of FPFH, as it is proposed by [18], comes in handy where multiple objects of various shapes dominate the scene. In particular in close-ups with very low noise and quantization errors the FPFH can play its card to label sharp corner and edges and to differentiate correctly between the curved objects. We confirm this fact by looking at the average values (Table II) and pictures (Table V) of the close kitchen and close table scenes, where it works quite satisfying through all classes and outperforms PC and RSD. Especially in the sphere category FPFH matches the point much more reliably than the others do which clearly makes FPFH the winner at these two scenes (in terms of micro average as well as macro average). Table IV proofs good results for the sphere class as well.

Most of the other scenes are dominated by planes, edges and corners with a few curved objects in them which is probably the most common setup as it can be found indoors. PC makes the best shape compared to the other three descriptors as it has advantages in a robust detection of edges and planes even in present of strong noise levels. The results look smooth and cleaner than they do for RSD and FPFH. Only the far kitchen scene troubles all descriptors. Most of the points of this scene are 2.5 m and further away from the sensor which suggests another modification of configuration parameters. By reducing the problem to a three class categorization (3c) the overall results (Table II) stay

### TABLE I

<table>
<thead>
<tr>
<th>Configuration parameters of each algorithm for the two distance categories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RSD</strong></td>
</tr>
<tr>
<td>close range</td>
</tr>
<tr>
<td>$r_{n} = 0.03$</td>
</tr>
<tr>
<td>$r_{f} = 0.03$</td>
</tr>
<tr>
<td>$r_{min, low} = 0.035$</td>
</tr>
<tr>
<td>$r_{min, high} = 0.08$</td>
</tr>
<tr>
<td>$x_{(C, S)} = 4.75$</td>
</tr>
<tr>
<td>$x_{(E, C)} = 3.5$</td>
</tr>
<tr>
<td>far range</td>
</tr>
<tr>
<td>$r_{n} = 0.045$</td>
</tr>
<tr>
<td>$r_{f} = 0.045$</td>
</tr>
<tr>
<td>$r_{min, low} = 0.038$</td>
</tr>
<tr>
<td>$r_{min, high} = 0.09$</td>
</tr>
<tr>
<td>$x_{(C, S)} = 4.75$</td>
</tr>
<tr>
<td>$x_{(E, C)} = 3.5$</td>
</tr>
</tbody>
</table>

a neighborhood radius for normal estimation (in m)
b neighborhood radius for feature estimation (in m) c lower threshold on the min radius separating edge/curved d lower threshold on the max curvature separating plane/curve e higher threshold on the min radius separating curved/plane f lower threshold on the max curvature separating curves/edge g ratio of max/min values separating cylinder/sphere h ratio of max/min values separating edge/corner

The pictures in Table V visualize the outcome of the algorithms on all scenes. Table II presents the corresponding accuracy values while Table IV summarizes all scenes separated by classes.

Fig. 3. Running time in seconds of the feature estimation algorithm depending on the selected neighborhood radius

### TABLE II

<table>
<thead>
<tr>
<th>Evaluation results for particular scenes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scene</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>kitchen close</td>
</tr>
<tr>
<td>kitchen far</td>
</tr>
<tr>
<td>office close</td>
</tr>
<tr>
<td>office far</td>
</tr>
</tbody>
</table>

Presented in terms of micro-average (which is the same for precision, recall and F-measure), macro-average recall, macro-average precision and macro-average F-measure.

(3c) refers to the 3 class evaluation

### TABLE III

<table>
<thead>
<tr>
<th>CHANGE OF THE F-MEASURE FROM FIVE-CLASS TO THREE-CLASS CATEGORIZATION IN PERCENTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scene</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>kitchen close</td>
</tr>
<tr>
<td>kitchen far</td>
</tr>
<tr>
<td>office close</td>
</tr>
</tbody>
</table>

Very similar to Table II, Table IV presents the results of the 3 class categorization for the same scenes. The overall results (Table II) stay very similar to Table II, Table IV presents the results of the 3 class categorization for the same scenes.
almost the same while naturally the absolute results improve for every algorithm. Table III shows these improvements of the F-measure relative to the results of the five-class categorization for all scenes. One can easily see that PC is the candidate with highest benefit for this case in most of the scenes which again proofs its strength for planes and edges.

The close relation of RSD and PC can be found in many of the test scenarios. Both have the characteristic to label points close to an edge as cylinders and have trouble to label curved objects correctly. However, RSD seems to be more affected by noisy data than PC, especially on planes. According to the accuracy values in Table II RSD places second in most of scenes where PC performs best.

TABLE IV
SHOWS THE ACCURACY RESULTS PER CLASS OVER ALL EVALUATED SCENES WITH A TOTAL OF ABOUT 1.7 MIO. POINTS.

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision RSD</th>
<th>F-measure RSD</th>
<th>Recall PC</th>
<th>F-measure PC</th>
<th>Recall FPFH</th>
<th>F-measure FPFH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane</td>
<td>2,892 979 964</td>
<td>976 722 460</td>
<td>721 429 777</td>
<td>721 429 777</td>
<td>721 429 777</td>
<td>721 429 777</td>
</tr>
<tr>
<td>Edge</td>
<td>248 156 275</td>
<td>734 686 547</td>
<td>731 689 402</td>
<td>731 689 402</td>
<td>731 689 402</td>
<td>731 689 402</td>
</tr>
<tr>
<td>Sphere</td>
<td>986 657 355</td>
<td>540 416 648</td>
<td>157 100 441</td>
<td>157 100 441</td>
<td>157 100 441</td>
<td>157 100 441</td>
</tr>
<tr>
<td>Cylinder</td>
<td>972 660 566</td>
<td>146 282 292</td>
<td>125 155 903</td>
<td>125 155 903</td>
<td>125 155 903</td>
<td>125 155 903</td>
</tr>
<tr>
<td>Corner</td>
<td>980 123 991</td>
<td>352 465 318</td>
<td>126 194 141</td>
<td>126 194 141</td>
<td>126 194 141</td>
<td>126 194 141</td>
</tr>
<tr>
<td>Edges+Corner</td>
<td>263 367 293</td>
<td>791 748 804</td>
<td>395 499 429</td>
<td>395 499 429</td>
<td>395 499 429</td>
<td>395 499 429</td>
</tr>
<tr>
<td>Curved</td>
<td>999 135 115</td>
<td>551 566 508</td>
<td>168 218 188</td>
<td>168 218 188</td>
<td>168 218 188</td>
<td>168 218 188</td>
</tr>
</tbody>
</table>

Along with the performance in accuracy and computation time another important matter is the flexibility. RSD and PC both are easy to configure and it is very straight forward to adjust them on scenes with different focus. FPFH however always requires the whole process of acquiring and labelling sample data to create a trained model which is cumbersome.

While the results might look satisfying at first, non of the algorithms can actually hold up to the requirements of the test scenarios. Both have the characteristic to label points close to an edge as cylinders and have trouble to label curved objects correctly. However, RSD seems to be more affected by noisy data than PC, especially on planes. According to the accuracy values in Table II RSD places second in most of scenes where PC performs best.

VIII. CONCLUSION

In this paper, we presented an in-depth evaluation of feature point descriptors on a variety of real-world scenarios. Both computation time and geometric surface classification accuracy have been measured and compared.

FPFH certainly has the potential to precisely classify complex shapes. Our experiments however showed that it has particular trouble dealing with the typical characteristics of PrimeSense cameras and to compensate that, exhaustive adjustments and training is required. RSD and PC both show very similar habits. However PC turns out to be more robust against sensor noise and classifies almost every scene much smoother than RSD does. In particular for the plane-edge-curved categorization tasks PC cuts a fine figure as long as it is restricted to an acceptable range.

VIII. ACKNOWLEDGEMENTS

This research was financed by the research program "Effiziente Produktion durch IKT" of the Baden-Württemberg Stiftung, project "ATLAS".

REFERENCES

<table>
<thead>
<tr>
<th>Ground truth</th>
<th>RSD</th>
<th>PC</th>
<th>FPFH</th>
</tr>
</thead>
<tbody>
<tr>
<td>kitchen close</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>kitchen far</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>table close</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>table far</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>office close</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>office far</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>cupboard close</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>cupboard close</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

The first column shows the manually labeled scene ground truth, the others the classification outcome for RSD, PC and FPFH. Point colors: light blue: plane, red: edge, yellow: corner, green: cylinder, dark blue: sphere, gray: ignored for evaluation.