Learning Informative Point Classes for the Acquisition of Object Model Maps

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München, Germany
{rusu,marton,blodow,beetz}@cs.tum.edu

***Abstract*—This paper proposes a set of methods for building informative and robust feature point representations, used for accurately labeling points in a 3D point cloud, based on the type of surface the point is lying on. The feature space comprises a multi-value histogram which characterizes the local geometry around a query point, is pose and sampling density invariant, and can cope well with noisy sensor data. We characterize 3D geometric primitives of interest and describe methods for obtaining discriminating features used in a machine learning algorithm. To validate our approach, we perform an in-depth analysis using different classifiers and show results with both synthetically generated datasets and real-world scans.**

I. Introduction

Scene segmentation and interpretation in robotics is an important research topic. In particular, obtaining accurate and informative object models out of sensed data can greatly benefit applications such as robot manipulation or navigation. Most of the research on object and surface classification has been done in the areas of computer vision (e.g [1]), with only a few exceptions in the area of point cloud sensing and processing [2]–[4]. Creating object maps from point cloud based representations requires the recognition of geometric primitives, typically solved as an optimization problem (e.g. non-linear model fitting), but this approach has to cope with a big number of parameters (increased complexity). To deal with such large solution spaces, heuristic hypotheses generators can provide candidates which reduce the number of models that need to be verified.

In this paper, we investigate robust point feature representations using multi-value histograms that characterize the surface on which the points lie based on their local neighborhood, and present an in-depth analysis on their usage for efficient point cloud classification. These classifiers can be used as accurate and reliable labeling procedures which segment point clouds into candidate regions for parameterized geometric primitive fitting. Using point labels like: *point on cylinder*, *point on torus*, etc. and the geometric relationships between them, we speed up and improve the recognition and segmentation of objects in real world scenes.

Figure 1 presents a snapshot of our classification results for a scanned dataset representing an indoor kitchen environment. Notice how points on the two cups are labeled as cylinders (body) and edge (handle), and the surface they are lying on (table top) as planar.

Point cloud models have been studied under several aspects, and different feature or label types have been proposed. Two

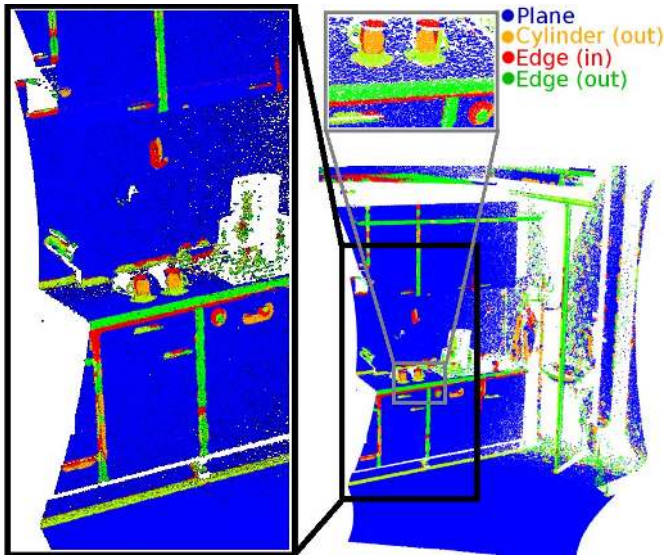

Fig. 1. Classification results for a dataset acquired in a kitchen environment. *In* and *Out* represent concave and convex surface types.

of the most widely used geometric point features are the underlying surface's estimated curvature and normal [5]. Both of them are considered *local features*, as they characterize a point using the information provided by the k closest neighbors of the point. Their estimated values are sensitive to sensor noise and the selection of the k neighbors, and they are not able to differentiate between points except at a very low level (e.g. points with a small curvature might lie on planes). Other proposals for point features include moment invariants [6], spherical harmonic invariants [7], and integral volume descriptors [8]. They have been successfully used in applications such as point cloud registration, and are known to be translation and rotation invariant, but they are not discriminative enough for determining the underlying surface type.

In general, descriptors that characterize points with a single value are not expressive enough to make the necessary distinctions for point-surface classification. As a direct consequence, most scenes will contain many points with the same or very similar feature values, thus reducing their informative characteristics. Alternatively, multiple-value point features such as curvature maps [9], or spin images [10], are some of the better local characterizations proposed for 3D meshes which got adopted for point cloud data. However, these representations

978–1–4244–2287–6/08/$25.00

require densely sampled data and are not able to deal with the amount of noise usually present in 2.5D scans.

We extend the work presented in [11] by computing local point feature histograms for each point in the cloud. We make an in-depth analysis of the points' signatures for different geometric primitives (i.e. planes, spheres, cylinders, tori, etc) and show classification results using different machine learning methods: Support Vector Machines (SVM), K-Nearest Neighbors (KNN) and K-Means clustering. The best classification results are obtained using Support Vector Machines which gave good results on other types of histogram-based classification, for images [12] as well as face-detection [13].

The remainder of the paper is organized as follows. Section II presents our implementation for computing informative point feature histograms. In Section III we analyze 3D geometric primitives from point cloud data, present our method for selecting the most discriminating point features from a given set, and discuss machine learning techniques for training a robust classifier. We discuss experimental results in Section IV, and conclude with Section V.

II. Point Feature Histograms

The goal of our work is to identify a feature space in which 3D points (obtained from real-world laser scans) lying on primitive geometric surfaces can be easily identified. For this purpose, the discriminating power of the feature space has to be high enough so that points on the same surface can be grouped in the same class, while points on different surfaces should be assigned to different classes. Furthermore, we require this feature space to be invariant to 3D rotations and translations, as well as insensitive to point cloud density and noise to a certain degree.

To formulate the feature space computational model, we first introduce the following notations:

- p_i is a 3D point having $\{x_i, y_i, z_i\}$ coordinates;
- n_i is a surface normal estimate at point p_i having a $\{nx_i, ny_i, nz_i\}$ direction;
- r is the radius of the sphere centered at a query point p_i, used for determining the nearest k neighbors of p_i (k-neighborhood) given a certain distance metric, i.e.
$$dist(p_i, p_j) \leq r, \text{ with } j \leq k;$$
- P_k^i is the set of points p_j, $(j \leq k)$, located in the k-neighborhood of a query point p_i;
- $\langle p_i,\ p_j \rangle$ describes the dot product between p_i and p_j;
- F^i is a feature histogram for the point p_i, represented as a multi-value array with each bin (F^i_{idx}) containing the percentage of point pairs from P_k^i which have a specific combination of feature values.

We propose the computation and usage of a histogram of values which encodes the local neighborhood's geometrical properties by generalizing the mean curvature at a point p, and provides an overall density and pose invariant multi-value feature. Given a set of points $P = \{p_1, ..., p_n\}$, the feature histograms are computed as follows:

1) for each point p_i in P, if normal n_i does not exist, then: i) select all P_{k_α} neighbors of p_i within a given radius r_α; ii) approximate n_i by the normal of the least-squares plane to the P_{k_α} surface using Principal Component Analysis; iii) use the existing viewpoint information v to re-orient n_i consistently[1]:
$$\text{if } \frac{\langle v - p_i,\ n_i \rangle}{\|v - p_i\|} < 0, \text{ then } n_i = -n_i$$
2) for each point p_i in P, select all P_{k_β} neighbors of p_i within a given radius $r_\beta > r_\alpha$;
3) for each pair of points p_{j_1} and p_{j_2} ($j_1 < k_\beta$, $j_1 \neq j_2$, $j_2 < j_1$) in P_{k_β}, and their estimated normals n_{j_1} and n_{j_2}, select a source p_s and target p_t, the source being the one having the smaller angle between the associated normal and the line connecting the points:
   if $\langle n_{j_1},\ p_{j_2} - p_{j_1} \rangle \leq \langle n_{j_2},\ p_{j_1} - p_{j_2} \rangle$
   then $p_s = p_{j_1},\ p_t = p_{j_2},\ n_s = n_{j_1},\ n_t = n_{j_2}$
   else $p_s = p_{j_2},\ p_t = p_{j_1},\ n_s = n_{j_2},\ n_t = n_{j_1}$
   and then define the Darboux frame (see Figure 2) with the origin in the source point as:
$$u = n_s,\ v = (p_t - p_s) \times u\ /\ ||p_t - p_s||,\ w = u \times v$$

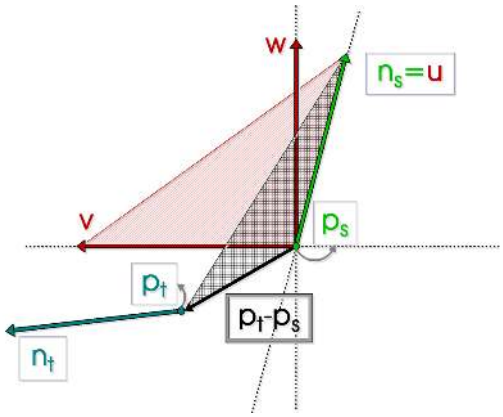

Fig. 2. The Darboux frame (vectors u, v and w) placed at p_s.

4) from p_s, p_t, n_s, and n_t, compute a set of 4 features that measure the angle differences between the points' normals and the distance vector between them and bin the values into a histogram [11]:

$$\left.\begin{aligned} f_0 &= \langle v, n_t \rangle \\ f_1 &= ||p_t - p_s|| \\ f_2 &= \langle u,\ p_t - p_s \rangle / f_1 \\ f_3 &= atan(\langle w, n_t \rangle, \langle u, n_t \rangle) \end{aligned}\right\} idx = \sum_{i=0}^{i \leq 3} \left[\frac{f_i \cdot d}{f_{i_{max}} - f_{i_{min}}} \right] \cdot d^i$$

where the []-operator denotes the integer part operation, d is the number of subdivisions of the features' maximum theoretical value range ($f_{i_{max}} - f_{i_{min}}$) and idx is the index of the histogram bin in which the point pair falls (F^i_{idx}). We increase the bin's value by 1, and normalize each bin with the

[1] if viewpoint information is unavailable (e.g. the dataset is not 2.5D), use an algorithm similar to [14]

total number of point pairs ($k \cdot (k+1)/2$) to achieve point density invariance.

The number of histogram bins that can be formed using these four geometric features is d^4. We empirically obtained good results by dividing the feature values in three parts, therefore obtaining a total of $3^4 = 81$ bins as the total number of combinations between the four features. Increasing the number of bins even further did not improve the results significantly, and since it increases exponentially by the power of 4, using more than 3 subdivisions would result in a large number of extra dimensions for each point (e.g. $4^4 = 256$D).

III. Learning 3D Geometric Primitives

Once the feature space is defined, we need to select a set of classes for labeling each point. The set is comprised of feature histograms for points lying on various 3D geometric surfaces. For our experiment, we have chosen the following geometric primitive shapes as class labels: plane, sphere, cylinder, cone, torus, edge and corner. The last two were selected due to the fact that data coming from indoor environments contains a high number of instances where they are present, hence it makes sense to differentiate between them and the rest.

To obtain histograms representing these shape primitives we apply the noise level found in our real-world laser scans to synthetically generated data. We use these histograms as training examples for machine learning techniques and test the classification both using marked (ground truth) point clouds and visually on the real-world dataset.

When computing the histograms, the selection of r_α and r_β should be adjusted to the size of the shapes that need to be detected and to the level of noise that is expected from the sensor. The value of r_α should be big enough to balance the effect of noise, but small enough to preserve the local nature of the computed normal. Similarly, the value of r_β should be big enough to capture enough information about the shape, but small enough to avoid too many surfaces in the neighborhood.

As we apply the method to indoor scenes with objects of every day use (e.g. cups, bottles, cereal boxes, oranges, etc), we generated the shape primitives used for training to have appropriate dimensions, and set $r_\alpha = 1.5cm$ and $r_\beta = 2.5cm$. These values are required for accurate normal estimation and distinctive feature generation for a cup's handle – the smallest shape (torus) we want to detect, given the noise level found in our scans (see subsection III-B).

A. Synthetically Generated Data

Since the chosen feature space requires consistently oriented normal information and bases its computational model on that, we need to account for two types of situations: i) when the estimated normals of a shape are oriented towards the concave (inner) part of the surface; and ii) when they are oriented towards the convex (outer) part. This does not mean that the feature space is variant to the object's pose, but that we will be able to identify the way the object is oriented to some degree (e.g. a corner of the room will be marked differently than a corner of a piece of furniture in the room – inner and outer corner respectively). Figure 3 presents the normal information for a cylinder in both the above mentioned cases.

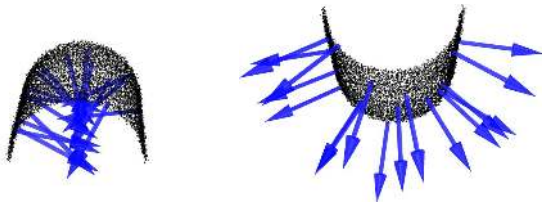

Fig. 3. Estimated and oriented surface normals for a cylinder pointing towards the concave (left) and convex (right) part of the surface.

Therefore, the number of classes has to be multiplied by two, with the exception of the plane. Figure 4 presents feature histograms for points lying on the selected geometric shapes.

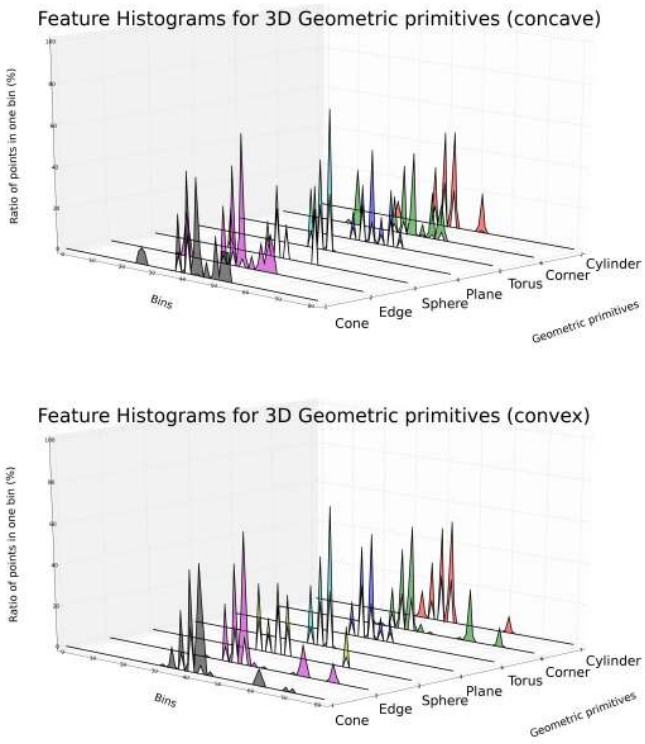

Fig. 4. Feature Histograms for points lying on several 3D geometric primitives: normals oriented towards concave (top) and convex (bottom) part.

The presented histograms were computed for shape primitives that were generated synthetically and are noiseless. The results show that the different geometrical properties of each surface produce unique signatures in the feature histograms space. We found that point density on the surface did not influence the feature histograms significantly, however the histograms for points that are on the edges of the generated shape primitives have small differences compared to those computed for points in the middle of the surface, producing the variations that can be seen in Figure 4.

B. Noise Analysis on Real-World Scans

An important requirement of the selected feature space is that it has to be able to cope with noisy datasets, acquired using real hardware sensors. To see whether the point histograms will be discriminating enough, we will analyze the level of noise in a scanned point cloud, add the same level of noise to our synthetic datasets, and compare the resulted histograms. The datasets have been acquired using SICK LMS laser sensors in an indoor kitchen environment [15].

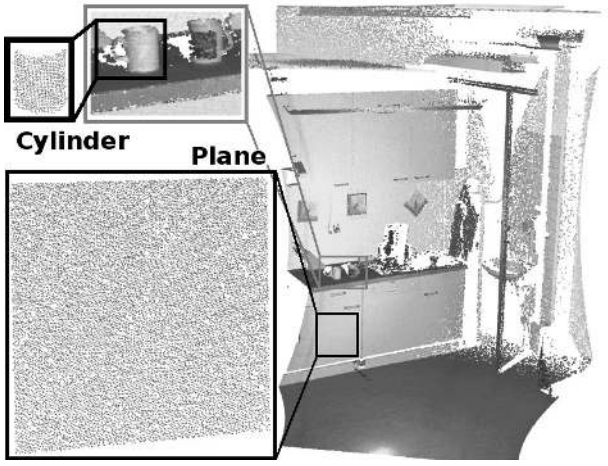

Fig. 5. Planar and cylindrical surface patches used for noise analysis.

For determining accurate noise models, we manually selected two portions of a scan (see Figure 5) to represent both linear shapes (e.g. planes) and non-linear (e.g. cylinders). To compute the noise parameters for the underlying shape model, we performed the following steps:

1) using a Sample-Consensus based method (MLESAC [16]), a search for the best support is performed;
2) a linear least-squares fit for the plane and a non-linear Levenberg-Marquardt optimization for the cylinder is used to refine the obtained model, and get the shape parameters;
3) by computing the distances from all the points in the dataset to the estimated shape model, and assuming their distribution to be Gaussian (see Figure 6), the mean and standard deviation of the noise is computed.

The left part of Figure 6 presents the distance distributions of the cylinder's points from the computed underlying shape model. Due to space constraints we are unable to show the resulted distance distribution for the planar patch, but the values obtained for mean and standard deviation were practically the same as the ones for the cylinder. In the figure, μ_{raw} and σ_{raw} represent the noise mean and standard deviation for the raw dataset. By adding the same level of noise to our synthetically generated datasets, and repeating the above computational steps, we obtained a new distance distribution with similar mean and standard deviation values, μ_{syn} and σ_{syn} (see the right part of Figure 6). The fitted Gaussian distribution is shown in black, the $\pm\sigma$ standard deviation in red, and the mean μ in blue.

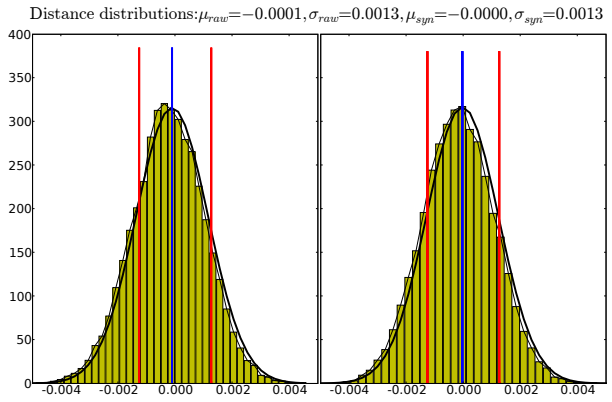

Fig. 6. Point to surface distance distributions for a point cloud representing a cylinder: raw scan (left), synthetic dataset with added noise (right).

Figures 7 and 8 show the feature histograms for points lying on planar and cylindrical surfaces: synthetically noiseless generated data on top, the noisy raw dataset in the middle, and the synthetic dataset with added noise on the bottom.

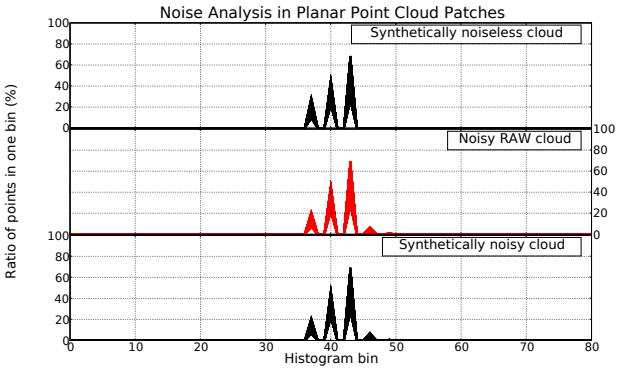

Fig. 7. Feature Histograms for points on a plane for synthetic noiseless data (top), raw point cloud data (middle) and synthetic noisy data (bottom).

C. Most Discriminating Features Selection

After obtaining enough datasets representing our geometric primitives of choice, and after their appropriate features have been computed and extracted, a selection of the most *relevant* features has to be performed. The selection is motivated by the fact that for a given shape, there could be tens of thousands to millions of feature histograms, which, if used directly, would render the learning problem costlier than needed. Therefore, a subset of features has to be extracted for each shape, which reduces the number of training examples but still preserves the discriminating power of the shape's histograms from other shapes. This can be viewed as a clustering problem, in the sense that the goal is to find the most relevant k clusters in

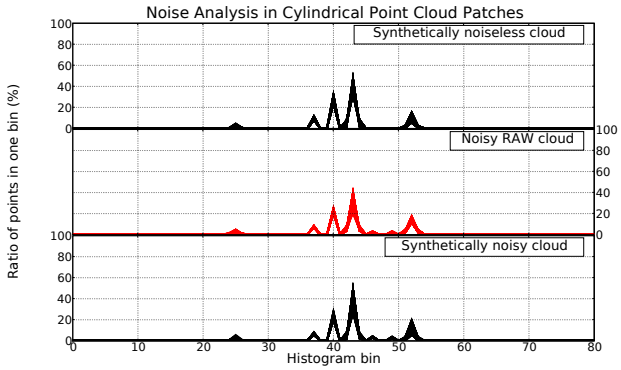

Fig. 8. Feature Histograms for points on a cylinder for synthetic noiseless data (top), raw point cloud data (middle) and synthetic noisy data (bottom).

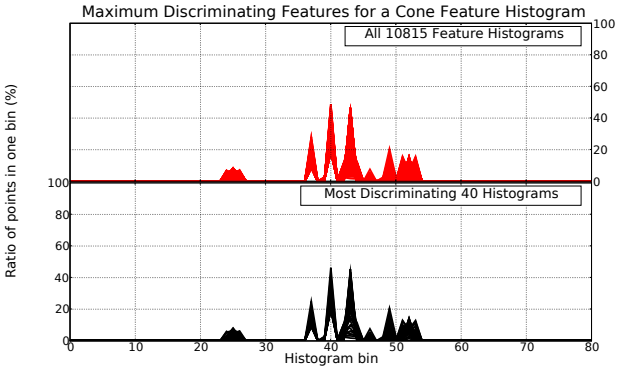

Fig. 9. Most discriminating 40 feature histograms for points lying on a concave conic surface. All 10815 feature histograms are shown on the upper part, and the resulted 40 feature histograms at the bottom.

the data, which taken together form the overall shape histogram without losing or modifying its discrimination. Therefore we could apply a direct clustering method such as K-Means, where k could be determined empirically. However, since our proposed feature space has 81 dimensions, it's very likely that the iterative clustering method will be trapped in local minima, and the optimal cluster centers will not be found. A solution is to reduce the dimensionality of the space and perform the clustering there. However, even in a lower dimensional space, the risk of converging to a false solution is directly related to the initial starting position of the clusters. Our solution consists in computing the mean μ-histogram of the shape, then creating a distance distribution from all the point feature histograms to it, divide it into equally distributed parts, and select a representative from each part as a starting solution for the k cluster centroids. In detail, the procedure for determining the most discriminating features is presented in the following computational steps:

1) for each shape candidate, compute the mean μ-histogram of the shape;
2) for each point in the shape, compute a distance metric D_m between the point's feature histogram and the mean μ-histogram of the shape;
3) arrange all the distance values in a distribution, divide it into k intervals, and uniformly sample an index value from each interval;
4) use the sampled distances as initializers for the k center clusters, and search for an optimal solution in the L2-Euclidean distance space with a K-Means like algorithm.

Our method gives similar results to [17]. For determining the optimal D_m, we have performed an in-depth analysis using multiple distance metrics and norms (such as L1-Manhattan, L2-Euclidean, Bhattacharyya, Chi-Square, Jeffries-Matusita, Kullback-Leibler) and have empirically discovered that the L1, Bhattacharyya and Chi-Square give the best results for our data. Analyzing distance metrics for histograms has already been performed in similar research initiatives such as [11], [18], and will not be covered again here.

Figure 9 presents the results of the selection process for a conic shape with 10815 feature histograms. Note that the number of k clusters was set to 40 just for the purpose of this example. In practice, we require the same number of training examples for each candidate shape. To achieve this, we first compute the feature histograms for all the shape primitives, and then we select $k = min(\alpha, nr^s{}_i)$, $i \leq n$, where nr^s is the number of feature histograms for shape s, n is the total number of primitive shape classes, and α is a maximum tolerated number of clusters set by the user.

Selecting the most discriminating features using our proposed K-Means clustering approach has the advantage that the learning time will be drastically reduced while the discriminating feature information will be preserved.

D. Supervised vs Semi-Supervised Learning

Our goal is to learn models for 3D geometric primitives (see Figure 4) and then apply these models to classify scanned data coming from sensors. Therefore, a decision has to be made on what type of learning method to use to achieve the best results. For supervised learning, the Support Vector Machines (SVM) are considered good classifiers, for their good performance on high dimensional spaces [12]. Given a set of training examples, the SVM learner attempts to find optimally separating hyperplanes which group sets of training examples belonging to the same class and separate those from different classes. The support vectors are usually found in a higher dimensional feature space where the data is mapped by making use of a kernel function. Choosing or designing a new kernel function is however a cumbersome process, and selecting the wrong kernel can lead to inadequate results.

However, by applying the most discriminating feature selection algorithm above, our training datasets are not voluminous, so trying out different kernels and optimally determining their best coefficients in a large parameter space is possible by training different models in parallel and selecting the best one. Besides the classical linear, sigmoid, and polynomial kernels, and motivated by the results in [12], we have selected the

family of Radial Basis Functions kernels:

$$K_{RBF}(x, y) = e^{-\gamma \cdot dist(x,y)}$$

By changing the distance metric, we obtain the following kernel functions:

$$dist = \sum_i |x_i - y_i|,\ K_{RBF} \rightarrow K_{RBF-Laplacian}$$

$$dist = \sum_i |x_i - y_i|^2,\ K_{RBF} \rightarrow K_{RBF-Gaussian}$$

$$dist = \sum_i \sqrt{|x_i - y_i|},\ K_{RBF} \rightarrow K_{RBF-Sublinear}$$

Both the Laplacian and sublinear kernels are shown to outperform the Gaussian kernel for histogram matching, as described in [1], [12].

Supervised learning techniques such as SVM have the advantage that while the model learning is slow (solved in our case through the use of most discriminating feature selection), the classification of new unseen instances is very fast.

Another category of learning techniques have been receiving increasing attention in classification problems lately: Semi-Supervised Learning techniques [19]. The problem of classifying new point features can be formulated differently, that is without directly specifying the target output classes. A rather effective and simple histogram classification method can be obtained by looking at different distance spaces and performing a K-Nearest Neighbor (KNN) search for the closest match, where $k = 1$. That is, for every shape training set, compute the mean μ-histogram, and assign the shape class to a point p, if the point's histogram p_i^f has the closest distance to it than to any other mean shape histogram. We evaluated the following distance metrics: Manhattan (L1), Euclidean (L2), Bhattacharyya, and Chi-Square (χ^2).

A similar method for classification is K-Means clustering, where instead of assigning the shape class directly as KNN does, an iterative search for the optimal k clusters is performed. Histograms belonging to the same class tend to be grouped together in the same cluster, which can be then verified whether it belongs to a required shape class by looking at its proximity to the shape's mean histogram. However, K-Means is sensitive to local minima and can easily be trapped based on how the cluster centroids are initialized. To account for this problem, in our implementation, we proceed as follows:

1) k is set to the the number of output shape classes, and for every shape class i, its k_i cluster centroid is initialized with the mean histogram of the geometric primitive;
2) a K-Means algorithm is run iteratively until convergence;
3) a distance metric between each obtained k_i cluster centroid and the mean histogram of each geometric primitive is computed and the cluster k is assigned to the closest shape class.

The last step is performed because we cannot be certain that the initial given centroids (and therefore labeling) is still the same at the end of the algorithm. For example the centroid cluster which was initialized with the mean histogram of a cylindrical shape could end up closer to the conical one, and viceversa.

IV. Discussions and Experimental Results

We have applied our algorithms to a synthetically generated scene to find out the accuracy of the classification results. Table 1 presents the results obtained using different kernels for Support Vector Machines [20], and various distance metrics for the K-Nearest Neighbors and K-Means clustering algorithms.

The best classification support for both noiseless and noisy data with Support Vector Machines was obtained using the RBF sublinear kernel, confirming the findings in [12]. The RBF Laplacian kernel gave similar results, albeit the RBF Gaussian kernel provided better results for noisy data. An interesting result was obtained using the linear kernel, which had the second best results for noiseless data. This means that when a lower computational cost is desired with small acceptable penalty losses in performance, a linear SVM could be used instead. Note that the numbers presented in the table are directly related to the synthetic dataset used – if ground truth would be available for real datasets, the outcome of the classification might result in bigger differences between the kernels. Figure 10 presents the classification results using the best parameters with each of the three methods (SVM, KNN, and KMeans) for our synthetic noiseless scene (left part) and for the scene with added noise (right part). In the legend presented at the bottom of Figure 10, we included two types of tori for detecting cups handles: one *rounded* type, close to the ideal torus, and another type more *flat*. Instantiations of the two different types of handles can be seen in Figure 11.

Table 1. Classification results

Method used	Noiseless	Noisy
SVM Linear kernel	95.17%	87.66%
SVM Polynomial kernel	94.39%	88.88%
SVM Sigmoid kernel	86.15%	83.44%
SVM RBF Gaussian kernel	94.55%	88.83%
SVM RBF Laplacian kernel	95.18%	88.14%
SVM RBF Sublinear kernel	**95.26%**	**89.55%**
KNN L1-Manhattan (μ-dist)	78.08%	78.03%
KNN L2-Euclidean (μ-dist)	67.22%	71.94%
KNN Bhattacharyya (μ-dist)	**87.11%**	**83.53%**
KNN Chi-Square (μ-dist)	83.64%	82.84%
K-Means (81D)	59.40%	55.24%
K-Means (81D) L1-Manhattan	**73.63%**	68.58%
K-Means (81D) L2-Euclidean	61.30%	68.58%
K-Means (81D) Bhattacharyya	**73.63%**	**70.74%**
K-Means (81D) Chi-Square	**73.63%**	68.58%

The KNN based classification gave very promissing results, especially when using the Bhattacharyya distance metric. This is due to the fact that the discriminating power of the feature histograms is high enough so that in certain distance spaces they can be easily separated. The presented results make the KNN method very attractive, especially for situations where a model does not have to be learned a priori.

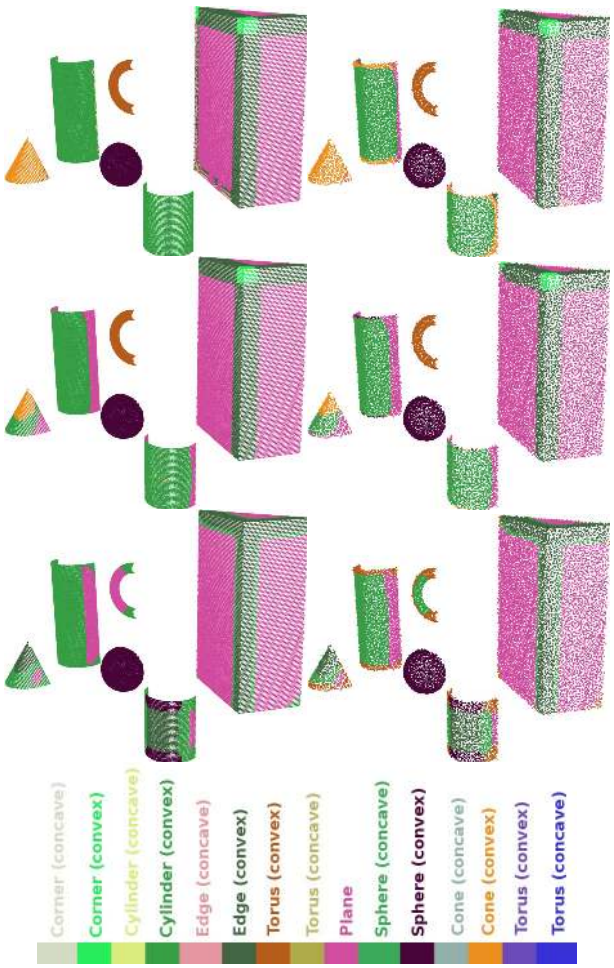

Fig. 10. Classification results for the synthetic scene. From left to right: noiseless and noisy synthetic scenes. From top to bottom: SVM sublinear kernel, KNN with Bhattacharyya distance, and K-Means with Bhattacharyya distance.

Not surprisingly, the K-Means clustering algorithm had the worst results of the three methods. Even though the k cluster centroids where initialized as close as possible to the true solution, the classical K-Means metric distance – the L2 Euclidean norm, is simply inadequate for such high dimensional spaces (81D) obtaining a rather poor 59.40% classification accuracy. The motivation for the 3^{rd} K-Means implementation step as explained in Section III-D can be observed in the last part of the table. Once the final k cluster centroids are set, we compare them again with the mean histograms of each shape and re-assign the cluster's label to the shape which is the closest, in a distance metric sense. The best results overall were obtained again with the Bhattacharyya distance.

Another aspect that needs to be verified is how the histograms classification copes with partial scans taken from different positions and angles. Figure 11 presents the classification results for 4 types of cups scanned from 3 different poses. In the first row, the acquired scans contain the cup handles as seen from a 45° angle and slightly from above; in the second row, the handles are precisely in the middle of the scan and are perpendicular to the viewpoint, while the position is above the cup almost looking down upon the inside bottom of the cup; and finally, in the third row, the handles are exactly at a 90° angle with the viewing direction, allowing them and the cup to be seen from the side. Our results indicate the following findings:

- all scans could identify the concave and convex cylindrical components of the cups with a reasonable error margin;
- the handles for the first 3 cups were identified as *flat* tori in situations where more than half of the handle was visible, and as *rounded* tori when seen from the side (e.g. the bottom row of the figure);
- the third cup had an unusual shape (i.e. not perfectly cylindrical) and has been classified as a composition between a normal concave cylinder (bottom), a convex cylinder (middle) and a convex torus (top), showing that the classification of more complex objects is possible, but has to be interpreted differently.

Fig. 11. Classification results for 4 types of cups scanned from different positions and angles.

To verify the robustness of the feature histograms on a more complex dataset where objects are grouped closer and occluding each another, we analyzed the scene presented in Figure 12 and obtained 90.26% classification results using a sublinear SVM kernel. Notice that our ground truth labels were expecting edges at the intersection of shape candidates, such as cylinder with plane, or sphere with plane, as well as torus with cylinder. Due to the radius r_β used, the torus representing the handle of the smaller cup was not classified as a torus, but as an edge. This leads to the conclusion that the overall classification accuracy could be improved if different levels of details are used (i.e. different radii r_α and r_β).

The results obtained by applying the same model to a real-world scene are shown in Figure 13. The top part of the figure presents the point cloud in intensity scale, and the bottom the classified results. Results obtained on raw laser scans acquired in an indoor kitchen environment have already been presented in Figure 1.

Fig. 12. A more complex noisy synthetic scene with 90.26% classification results using a sublinear SVM kernel.

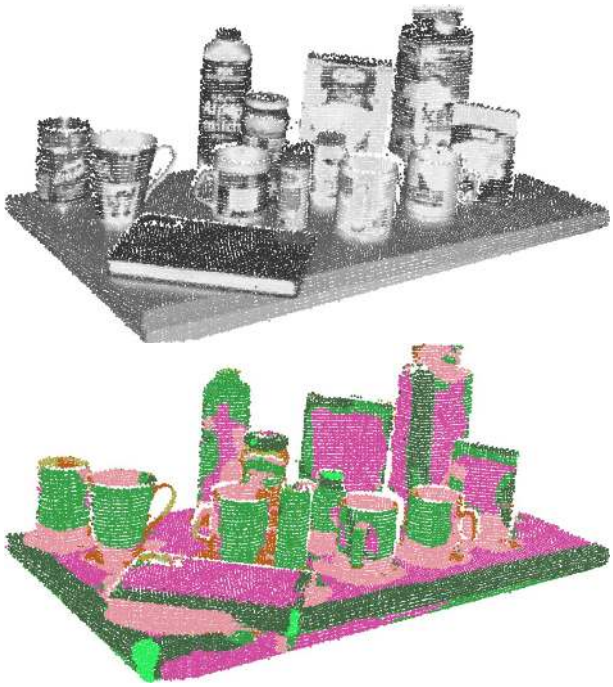

Fig. 13. A table-top with kitchen objects shown as an intensity image (top) and the results after classification (bottom).

V. Conclusion

We have presented a system for estimating informative point features as multi-value histograms which characterize the local geometrical properties of their underlying surface. The features are pose invariant and cope well with different dataset densities and noise levels.

By carefully selecting a machine learning classifier with an appropriate distance metric, and learning models of feature histograms for 3D shape primitives, we can robustly segment point cloud scenes into geometric surface classes. Our approach has been validated using synthetic datasets with ground-truth information and visually on real-world scans. The results show that the method works well in a wide range of situations, with the only tradeoff that the scale parameters (most importantly r_β) has to be chosen a priori by the user. As future work, we plan to investigate topics such as feature persistence over a range of radii to be able to select the correct parameters depending on the object type.

Acknowledgment

This work is supported by the CoTeSys (Cognition for Technical Systems) cluster of excellence.

References

[1] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, 2007, pp. 494–501.

[2] G. Taylor and L. Kleeman, "Robust Range Data Segmentation using Geometric Primitives for Robotic Applications," in *Proceedings of the 5th IASTED International Conference on Signal and Image Processing*, August 2003, pp. 467–472.

[3] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, June 2007.

[4] G. Biegelbauer and M. Vincze, "Efficient 3D Object Detection by Fitting Superquadrics to Range Image Data for Robot's Object Manipulation," in *ICRA*, 2007, pp. 1086–1091.

[5] A. M. and A. A., "On normals and projection operators for surfaces defined by point sets," in *Proceedings of Symposium on Point-Based Graphics 04*, 2004, pp. 149–155.

[6] F. Sadjadi and E. Hall, "Three-Dimensional Moment Invariants," *PAMI*, vol. 2, no. 2, pp. 127–136, 1980.

[7] G. Burel and H. Hénocq, "Three-dimensional invariants and their application to object recognition," *Signal Process.*, vol. 45, no. 1, pp. 1–22, 1995.

[8] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust Global Registration," in *Proc. Symp. Geom. Processing*, 2005.

[9] T. Gatzke, C. Grimm, M. Garland, and S. Zelinka, "Curvature Maps for Local Shape Comparison," in *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI' 05)*, 2005, pp. 246–255.

[10] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, May 1999.

[11] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-Pair-Relation Histograms: A Statistical 3D-Shape Representation for Rapid Classification," in *3DIM03*, 2003, pp. 474–481.

[12] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *Neural Networks, IEEE Transactions on*, vol. 10, no. 5, pp. 1055–1064, 1999.

[13] C. A. Waring and X. Liu, "Face detection using spectral histograms and SVMs." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 3, pp. 467–476, 2005.

[14] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992, pp. 71–78.

[15] R. B. Rusu, N. Blodow, Z. Marton, A. Soos, and M. Beetz, "Towards 3D Object Maps for Autonomous Household Robots," in *Proceedings of the 20th IEEE International Conference on Intelligent Robots and Systems (IROS), San Diego, CA, USA, Oct 29 - 2 Nov.*, 2007.

[16] P. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[17] N. Vasconcelos, "Feature Selection by Maximum Marginal Diversity: optimality and implications for visual recognition," *CVPR*, vol. 01, p. 762, 2003.

[18] G. Hetzel, B. Leibe, P. Levi, and B. Schiele, "3D Object Recognition from Range Images using Local Feature Histograms," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'01)*, vol. 2, 2001, pp. 394–399.

[19] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006. [Online]. Available: http://www.kyb.tuebingen.mpg.de/ssl-book

[20] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.