Support Vector Machines for Classification of Geometric Primitives in Point Clouds

M. Caputo, K. Denker, M. Franz, P. Laube, G. Umlauf

Institute for Optical Systems, University of Applied Sciences Constance, Germany

Abstract.
Classification of point clouds by different types of geometric primitives is an essential part in the reconstruction process of CAD geometry. We use support vector machines (SVM) to label patches in point clouds with the class labels tori, ellipsoids, spheres, cones, cylinders or planes. For the classification features based on different geometric properties like point normals, angles, and principal curvatures are used. These geometric features are estimated in the local neighborhood of a point of the point cloud. Computing these geometric features for a random subset of the point cloud yields a feature distribution. Different features are combined for achieving best classification results. To minimize the time consuming training phase of SVMs, the geometric features are first evaluated using linear discriminant analysis (LDA).
LDA and SVM are machine learning approaches that require an initial training phase to allow for a subsequent automatic classification of a new data set. For the training phase point clouds are generated using a simulation of a laser scanning device. Additional noise based on an laser scanner error model is added to the point clouds. The resulting LDA and SVM classifiers are then used to classify geometric primitives in simulated and real laser scanned point clouds.
Compared to other approaches, where all known features are used for classification, we explicitly compare novel against known geometric features to prove their effectiveness.

1 Introduction

Since laser scanners and similar devices are wide spread and have become less expensive, the need for automatic CAD reconstruction methods increases. Originally such devices were used only for quality control in the manufacturing process, but today also for reverse engineering and reconstruction of technical and non-technical components. For the reconstruction of geometric primitives like cones, planes, or spheres, detection is usually implicit by fitting different primitive types and deciding based on error thresholds. In noisy point clouds these often RANSAC-based methods [FB81] may fit the wrong geometric primitive because it yields the smallest error. Explicit classification can solve this phenomenon. Denker et al. [DHR*13] proposed a system for online reconstruction, in which a fixed set of heuristic rules based on estimation of local differential geometric properties is used for primitive detection. Cylinders, spheres, and planes

are the only supported primitive types of the system and additional methods for the detection of other primitives such as cones or tori are needed.

Our system classifies the geometry of small patches of a point cloud using machine learning methods. In machine learning input samples classified by learning to discriminate different classes within feature spaces from training samples. Thus, these machine learning algorithms require feature vectors of a fixed length as input. The computation of these feature vectors from raw 3d point data is called feature extraction.

Feature extraction in point clouds and other 3d data is an important topic, that has been addressed from various perspectives. Most of the existing work is either intended for meshed data or for 3d shape recognition in object databases. It can be used to detect complex objects like cars, lamp posts or parking meters in 3d scans of urban environments [GKF09] or to create object recognition for the purpose of robot-object interaction [HHYR12]. Describing such 3D-shapes is possible by a variety of geometric features extracted from point clouds. Simple point features like shape distributions can be used to measure the similarity between different 3d shapes [OFCD02]. The distributions are represented as histograms sampled from angles, distances, areas, and volumes of random point tuples. Considering the local neighborhood of points, features based on point pairs and their normals are introduced in [WHH03]. Additional distributions for identifying shapes contained in point clouds, based on distances, are proposed in [MS09]. Another neighborhood based method for shape description uses curvatures and curvature directions. Hetzel et al. [HLLS01] use histograms based on normals and curvatures to identify objects in range images. In [RMBB08] point neighborhoods are used to describe a 16 dimensional feature histogram for point cloud segmentation.

Recently, geometric features are used for classification based on machine learning methods. Endoh et al. [EYO12] use locally linear embedding to learn object shapes. They explore clustering to reduce the number of required training samples. The impact of supervised, semi-supervised, and unsupervised dimension reduction as a learning method for shape-classification is studied by in [YTSO08] for the surflet pair feature of [WHH03]. Using support vector machines with point features as well as curvature features to classify surfaces is proposed in [AFB*12] for a small set of geometric primitives including edges and corners. The mentioned feature-based methods only use a small subset of possible point could features. It is also assumed that using all features leads to high discriminative power in classification and tests of particular feature and feature-combination performance are missing.

Our approach is to take a small patch of the original point cloud, extract its geometric features and use a pre-trained machine learning algorithm to detect which geometric primitive is most likely represented by the point cloud patch. This approach needs no fixed heuristic rules or thresholds and can be extended to additional primitives, provided enough training data exist. We explicitly test single and combined feature performance for detection.

2 Methods

To apply machine learning algorithms such as linear discriminant analysis (LDA) or support-vector machines (SVM) a training phase is required. In the training phase the LDA and SVM are exposed to a large set of pre-classified point clouds to learn to discriminate between the different classes. The information about these classes is implicitly represented in distributions of geometric features exacted form the local geometry in the point cloud. Acquiring real scanned point clouds for the training is difficult, because scanning a sufficiently large number of training point clouds would require a substantial amount of time and a large number of different real world models. Therefore, the training point clouds are generated using a laser scanning simulation with a built-in error model.

In this section first the geometric features are described, Section 2.1. Based on the feature histograms the machine learning approaches are presented in Section 2.2. To generate the necessary training point clouds a scan simulation is used (Section 2.3), showing the resulting feature spaces in Section 2.4.

2.1 Geometric features and feature histograms

The feature histograms used for LDA or SVM are required to have a fixed length. To achieve this, the extracted geometric features are arranged in histograms with 64, 96, 128, 256, or 512 bins. These histograms are usually normalized and might be cropped to eliminate the influence of outliers.

The first four geometric features we used for comparison depend only on the location of the points in the point cloud and are adopted from [OFCD02].

F.1 *Point angles* are computed as the angles between two vectors spanned by three random points.
F.2 *Point distances* δ are computed by the Euclidean distance between two random points.
F.3 *Triangle areas* are computed from the square root of the triangle area of three random points.
F.4 *Tetrahedron volumes* are computed from the cubic root of the tetrahedron volumes V of four random points $\mathbf{p}_1, \ldots, \mathbf{p}_4$, i.e.

$$V = |(\mathbf{p}_1 - \mathbf{p}_4) \cdot ((\mathbf{p}_2 - \mathbf{p}_4) \times (\mathbf{p}_3 - \mathbf{p}_4))|/6.$$

The points required for these features are mutually different, uniformly distributed random points from the point cloud. The resulting feature histograms have 64 bins and are normalized to $[0, 180]$ for F.1 and to $[0, 1]$ for the others.

Geometric features that do not only depend on point locations are normal angles and normal directions.

F.5 *Normal angles* are given by angles between normals at two random points.
F.6 *Normal directions* are coordinates of the normalized normal at all points.

Thus, the feature histogram for F.6 is the concatenation of the three 32 bin histograms for the individual coordinates ranging from -1 to 1.

Geometric features that depend on the curvature are defined as follows:

F.7 *Curvature angles* are computed as the angles between the two corresponding principal curvature directions $\mathbf{v}_1$, $\mathbf{v}_2$ at two random points.
F.8 *Curvature directions* are given by the six coordinates of two normalized principal curvature directions $\mathbf{v}_1$, $\mathbf{v}_2$ at all points. Optionally, each coordinate can be weighted by the absolute value of the principal curvatures κ_1 or κ_2.
F.9 *Curvature differences* are computed from the absolute differences of the principal curvatures κ_1, κ_2 and Gaussian $K = \kappa_1\kappa_2$ and mean curvature $H = (\kappa_1 + \kappa_2)/2$ at two random points, optionally weighted by distance.

The geometric feature F.7 results in two 64-bin histograms normalized to $[0, 180]$ concatenated to one 128-bin histogram. The feature histogram for F.8 is the concatenation of the six 32-bin histograms for the individual coordinates ranging from -1 to 1. For the feature histogram for F.9 the four 32-curvature-bin are cropped to the 0.05 to 0.95 percentile range, normalized, and concatenated.

In order to combine the classification capabilities of individual geometric features, they can be combined into more general features. In [WHH03] combined feature based on two surflet pairs are proposed. These surflet pairs are defined as point-normal-pairs $(\mathbf{p}_1, \mathbf{n}_1)$ and $(\mathbf{p}_2, \mathbf{n}_2)$ with normalized normals $\mathbf{n}_1, \mathbf{n}_2$. From two surflet pairs a local, right-handed, orthonormal frame is computed

$$\mathbf{u} = \mathbf{n}_1, \qquad \mathbf{v} = ((\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{u})/\|(\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{u}\|, \qquad \mathbf{w} = \mathbf{u} \times \mathbf{v}.$$

This frame yields three geometric attributes

$$\alpha = \arctan(\mathbf{w} \cdot \mathbf{n}_2, \mathbf{u} \cdot \mathbf{n}_2), \qquad \beta = \mathbf{v} \cdot \mathbf{n}_2, \qquad \gamma = \mathbf{u}(\mathbf{p}_2 - \mathbf{p}_1)/\|\mathbf{p}_2 - \mathbf{p}_1\|.$$

Together with the point distance δ these attributes define the surflet pair feature:

F.10 *Surflet pairs* are computed as the tuple $(\alpha, \beta, \gamma, \delta)$ for two random points.

Thus, the surflet pair feature yields a 128-bin histogram normalized to $[0, 1]$.

To construct other combined features we combined those features that proved effective as individual features. The combined feature histograms are concatenated from the histograms of the individual features.

F.11 *Triple combinations* are combined from the three best geometric features depending on point, normal, and curvature information: F.4, F.5, F.7.
F.12 *Simple surflet combinations* are combined from F.4, F.5, and F.10.
F.13 *Extended surflet combinations* are combined from F.10 and F.11.

Remark 1. For comparison we tested nine additional geometric features, e.g. the shape index [KvD92]. However, all these features proved less effective in the experiments with a true positive classification rate below 0.5. These additional geometric features are given in Appendix A.

Normal and principal curvature estimation For the computation of the geometric features normals, principal curvatures and principal curvature directions at random points in the point cloud must be estimated.

To estimate a local normal at a point $\mathbf{p}$, the set P of its 100 closest neighbors is determined. Computing a principal component analysis (PCA) for P yields the eigenvector corresponding to the smallest eigenvalue of the covariance matrix. This eigenvector $\mathbf{n_p}$ is used to estimate the normal at point $\mathbf{p}$.

To estimate the principal curvatures and principal curvature directions at $\mathbf{p}$ the polynomial fitting of osculating jets of [CP05] is used. The points of P are approximated by a bi-variate height function $z(x, y)$ over the estimated tangent plane defined by $\mathbf{n_p}$. Then, $z(x, y)$ is computed by the truncated Taylor expansion of order n using the *n-jet* J_n

$$z(x,y) = J_n(x,y) + o\left(\|(x,y)\|^n\right), \qquad J_n(x,y) = \sum_{k=0}^{n}\sum_{i=0}^{k} \frac{b_{k-i,i}}{(k-i)!i!} x^{k-i} y^i.$$

The n-jet has the same differential geometric properties up to order n as the underlying surface. Since J_n is only approximated, it yields approximated differential geometric quantities. The $(n+1)(n+2)/2$ jet coefficients $b_{k-i,i}$ are computed by least squares approximation. They are used to approximate the Weingarten map of the surface. Its eigenvalues and eigenvectors yield estimates for the principal curvatures κ_1, κ_2 and principal curvature directions $\mathbf{v}_1$, $\mathbf{v}_2$. For the implementation we used the CGAL-library [CGA].

2.2 Machine learning

Two machine learning algorithms were used: *Linear Discriminant Analysis* (LDA) and a d-class *Support Vector Machine* (SVM) for open and closed classification.

Supervised learning methods are designed to classify new data with respect to a large body of pre-classified training data. The training data are represented as feature vectors which are chosen to separate the different classes from each other. These features live in a high-dimensional feature space F. In supervised learning the objective is to compute hypersurfaces separating the classes in feature space. These hypersurfaces are then used to classify new input data by deciding on which side of the hypersurfaces the data reside. In case of SVMs the additional objective is to maximize the margin between two classes in relation to their separating hypersurface. Figure 1 illustrates this concept in case of separating hyperplanes in a two-dimensional feature space.

Linear discriminant analysis The training data consist of feature data $\mathbf{x}_i \in F$ that are manually assigned to the correct class $y_i \in \{1, \ldots, d\}$ for d-class classification. So, the training data are given by

$$\mathcal{X}_d = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n) |\ \mathbf{x}_i \in F, y_i \in \{1...d\}\}.$$

For LDA the data are assumed to be separable by a hyperplane which is represented by its normal ω, the *weight vector*, and *bias* b as

$$\omega^t \mathbf{x} + b = 0,$$

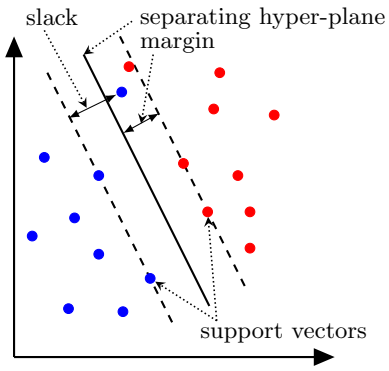


Fig. 1: Separating hyperplane, margin, support vectors, and slack for a two dimensional feature space.

for $\mathbf{x} \in F$. For only two classes this results in a decision function $f_{\omega,b}(\mathbf{x}) = \text{sign}(\omega^t \mathbf{x} + b)$ that assigns the class labels ± 1 to a previously unseen test data point $\mathbf{x}$. Assuming a normal distribution with means μ_1 and μ_2 and common co-variance Σ for the two classes, the weight vector and bias are given by

$$\omega = \Sigma^{-1}(\mu_1 - \mu_2), \qquad b = (\mu_1 + \mu_2)/2.$$

This concept can be extended to d-class classification (see below), [DHS01].

Support vector machine An SVM has the additional objective to maximize the margin between two classes. This results in finding ω and b such that

$$\Phi(\omega) = \omega^t \omega / 2 \tag{1}$$

is minimized with respect to the constraints

$$y_i(\omega^t \mathbf{x}_i + b) \geq 1, i = 1, \ldots, n, \tag{2}$$

see [CST00]. The minimum of (1) subject to (2) is computed by dualization using Lagrange multipliers. The Kuhn-Tucker-conditions imply that the weight vector ω can be represented in terms of the Lagrange multipliers $\alpha = (\alpha_1, \ldots, \alpha_n)$ and the training data as $\omega = \sum_{i=1}^{n} \alpha_i \, y_i \, \mathbf{x}_i$. So, instead of minimizing (1) the resulting Lagrangian

$$\Phi^*(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \, y_i y_j \, \mathbf{x}_i^t \mathbf{x}_j. \tag{3}$$

is maximized with respect to the constraints

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \qquad \alpha_i \geq 0, i = 1, \ldots, n.$$

If in the solution $\alpha_i \neq 0$, the corresponding $\mathbf{x}_i$ is called *support vector*.

For noisy data requiring linear separability is too restrictive. Thus, *slack variables* ξ_i are used to allow for some data inside the margin, see Fig. 1. The objective function to minimize additionally penalizes excessive slack variables

$$\Phi(\omega) = \frac{1}{2}\omega^t\omega + C\sum_{i=1}^{n}\xi_i \tag{4}$$

with respect to the constraints

$$y_i(\omega^t\mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, \ldots, n. \tag{5}$$

The minimum of (4) subject to (5) is again computed via dualization and Lagrange multipliers yielding the maximization problem (3) with the additional constraints $\alpha_i \leq C, i = 1, \ldots, n$.

For some problems, the classes cannot be separated by hyperplanes, but only by nonlinear hypersurfaces. So, the scalar product $\mathbf{x}_i^t\mathbf{x}_j$ is replaced by a *kernel* $K(\mathbf{x}_i, \mathbf{x}_j)$. The kernel is chosen such that it corresponds to a scalar product in a high-dimensional feature space. For our tests we used the *Gaussian RBF Kernel*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2).$$

With this kernel the classifier for new data $\mathcal{X} \in F$ is given by $f_{\alpha,b}(\mathbf{x}) = \text{sign}(m_{\alpha,b}(\mathbf{x}))$ with the *data margin*

$$m_{\alpha,b}(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

Multi-class classification For d-class classification the SVM concept is extended using the *one-versus-all* approach. Given the d-class training data $\mathcal{X}_d$ the one-versus-all approach uses d binary SVMs. Each binary SVM is trained to separate one class from all others. These d binary SVMs can be used in two different ways to classify new data $\mathbf{x} \in F$. For *closed SVM classification* (cSVM) the class with the largest data margin $m_{\alpha,b}(\mathbf{x})$ is selected to be the class $\mathbf{x}$ belongs to. For *open SVM classification* (oSVM) the class with the largest non-negative data margin $m_{\alpha,b}(\mathbf{x})$ is selected. If all margins are negative, $\mathbf{x}$ is not classified.

For more details on SVM based learning methods refer to [CST00,SS01]. For our implementation we used the SHARK library [IHMG08].

Model selection Model selection refers to the process of selecting the best parameters C and γ for the SVM. For this optimization usually a grid search in the two-dimensional (C, γ)-space is used. We used a two-dimensional non-uniform grid on which all pairwise parameter combinations are evaluated. To compare the different models, k-fold cross validation is used. We used $k = 5$. First, the training data $\mathcal{X}$ are divided into k equally sized sub-sets $\mathcal{S}_i, i = 1, \ldots, k$. Then,

for each grid point the SVM is trained k-times on each training set $\mathcal{T}_i = \mathcal{X} \setminus \mathcal{S}_i$. After the training on $\mathcal{T}_i$ the SVM is tested with the data from $\mathcal{S}_i$ yielding a true-positive-rate. For each grid point the k true-positive-rates are averaged. The best (C, γ)-combination is given by the grid point with the largest true-positive-rate. Optimization results are shown in Appendix A.

2.3 Laser scanning simulation and training data generation

For the training a large set of training point clouds is required which is generated by a laser scanning simulation. This simulation is based on tracing the rays of a virtual laser line probe onto virtual geometric primitives. It simulates the behavior of real laser scanners by sweeping fans of rays from a scan position over the geometric primitive. To define the position and pose of the laser probe two spheres around the centroid of the geometric primitive are constructed with radii of 8 and 0.02 units. Each scan position and pose is defined by a random point $\mathbf{p}_1$ on the outer sphere, a view direction $\mathbf{s}_1 = \mathbf{p}_2 - \mathbf{p}_1$ by a random point $\mathbf{p}_2$ on the inner sphere, and a random view-up vector $\mathbf{s}_2 \perp \mathbf{s}_1$. The plane perpendicular to $\mathbf{s}_2$ contains the reference fan, that is a cone of rays with aperture angle φ_1 and axis $\mathbf{s}_1$. The reference fan contains φ_1/φ_3 equidistant rays. To generate additional fans at $\mathbf{p}_1$ the reference fan is rotated around $\mathbf{s}_3 = \mathbf{s}_2 \times \mathbf{s}_1$ by φ_2-angle steps. $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{s}_2$ are uniformly distributed. This setup is shown in Figure 3.

Fig. 2: Point cloud patch of a sphere.

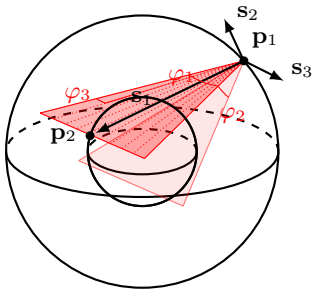


Fig. 3: The setup of the simulation of a laser scanner.

Additionally, the laser scanner model includes two types of systematic errors. The first error affects the scanner position $\mathbf{p}_1$ by a random offset adding the same error to all points of one fan. The second error affects the distance measure for each individual scanned point. Both errors are normally distributed with zero mean and standard deviation of 0.0125 and 0.0025, respectively.

There are six classes of geometric primitives: planes, cylinders, cones, spheres, ellipsoids, and tori. The training data were generated by extracting point cloud patches from a simulated scan of a complete primitive. Each point cloud patch was defined to contain all points of the point cloud within a cube of edge length

0.7 centered at a random point of the point cloud. Figure 2 shows an example point cloud patch.

We use 9600 point cloud patches with equal distribution from one of the six primitive classes as training data. 7680 patches are used for training, 1920 are used to evaluate the method by the true-positive-rate. The geometric parameters for the primitives are normally distributed with mean and standard deviation as in Table 1. To compute the geometric features from the point cloud patches often random point pairs, triplets, or quadruplets were chosen. In these cases 2^{17} feature values were sufficient to yield stable feature histograms.

Primitive parameter	Mean	Standard deviation	Primitive parameter	Mean	Standard deviation
Plane radius	3.0	0.5	Sphere radius	2.0	0.4
Cylinder height	3.0	0.5	Ellipsoid radius	2.0	0.4
Cylinder radius	2.0	0.4	Torus radius	5.0	0.6
Cone height	3.0	0.5	Torus tube radius	2.0	0.4
Cone radius	2.0	0.3			

Table 1: Properties of normal distribution of primitive parameters.

2.4 Feature space visualization

To better understand the classification properties of individual SVMs the separating hyper-surfaces are visualized. Because the feature spaces have dimensions between 64 and 512, PCA is used to reduce the features spaces to three dimensions. The SVMs were then trained on the dimension reduced training data. For visualization the training data are visualized by colored points: blue for planes, green for cylinders, red for cones, cyan for spheres, magenta for ellipsoids, and yellow for tori. For each binary SVMs of the one-versus-all approach the scalar field of the data margin was computed. Using the marching cubes algorithm [LC87] a meshes of the boundary surface was extracted and rendered in the 3d feature space. Examples are shown in Figure 4.

3 Results

The results of the application of LDA and SVMs for geometric primitive recognition are evaluated with respect to the classification results in terms of true-positive-rates.

In this section we show classification results of the geometric features in Section 2.1 used with LDA- and SVM-classification. All results are based on the same training data of point cloud patches from the six primitive classes. Table 2 shows the confusion matrix for the closed d-class SVM for the simple surflet combination feature F.12. In the confusion matrix the rows are the primitive class of the test data and the columns are the detected classes, e.g. in the first

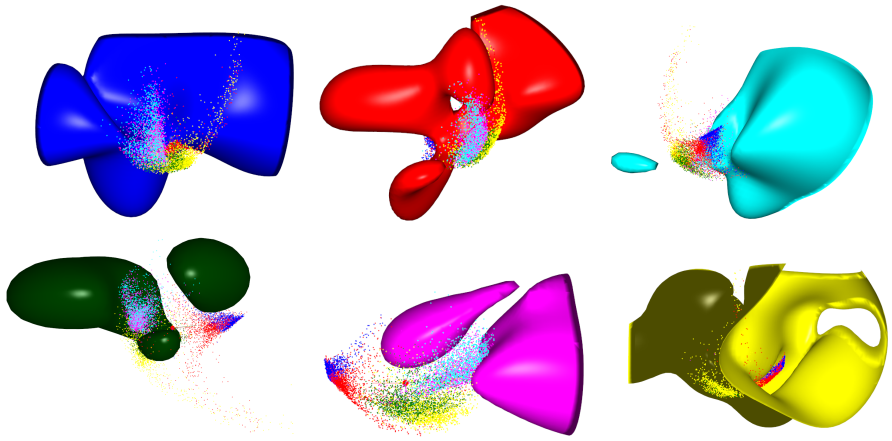
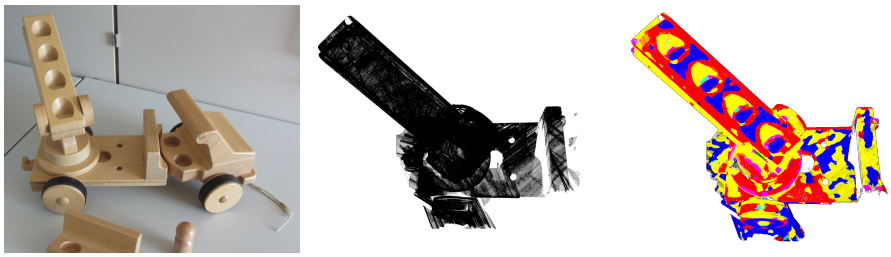


Fig. 4: Visualization of the separating hyper-surface of the SVM to classify planes (blue), cones (red), spheres (cyan), cylinders (green), ellipsoids (magenta), and tori (yellow) from different perspectives in the 3d feature space of the simple surflet combinations feature F.12.

row there are 311 planar point clouds, of which 304 are classified as planes and 7 are classified as cones. Table 3 shows the true-positive-rates for all features.

The visualization of the reduced feature spaces in Figure 4 allows, much like the confusion matrix, to identify two or more classes which may not be separated properly. This is the case if two or more colors are mixed inseparable in one region of the feature space.

Figure 5(c) shows real scan data colored according to the detected primitives. For coloring the simple surflet combinations feature F.12 in a closed SVM



(a) Wooden toy firetruck. (b) Point cloud of scanned firetruck. (c) Point cloud of firetruck colored by primitive class.

Fig. 5: A real scan showing a part of a wooden toy firetruck colored by primitive class using the simple surflet combinations feature F.12: planes (blue), cylinders (green), cones (red), spheres (cyan), ellipsoids (magenta), tori (yellow).

	F.12 Simple surflet combi.						F.10 Surflet pairs						F.5 Normal angles					
	Planes	Cyl.	Cones	Spheres	Ellips.	Tori	Planes	Cyl.	Cones	Spheres	Ellips.	Tori	Planes	Cyl.	Cones	Spheres	Ellips.	Tori
Planes	304	0	7	0	0	0	305	0	6	0	0	0	297	0	14	0	0	0
Cyl.	0	298	2	0	0	18	0	299	2	0	0	17	0	291	3	0	0	24
Cones	6	22	294	0	0	3	31	20	266	1	0	7	54	30	230	0	0	11
Spheres	0	0	0	302	16	0	0	0	0	281	37	0	0	0	0	282	36	0
Ellips.	0	0	0	83	240	0	0	0	1	81	238	3	0	0	0	98	208	17
Tori	0	76	1	0	1	247	0	75	4	0	3	243	0	110	3	2	13	197

Table 2: Confusion matrices for features F.12, F.10 and F.5 used with a six-class cSVM in heat-map coloring.

Feature Histogram	LDA	cSVM	oSVM	Feature Histogram	LDA	cSVM	oSVM
F.1 Point angles	0.643	0.713	0.680	F.7 Curvature angles	0.628	0.695	0.603
F.2 Point dist.	0.336	0.366	0.134	F.8 Curvature dir.	0.312	0.443	0.302
F.3 Triangle areas	0.594	0.644	0.558	F.9 Curvature differences	0.356	0.420	0.185
F.4 Tet. vol.	0.612	0.758	0.733	F.10 Surflet pairs	0.720	0.850	0.841
F.5 Normal angles	0.678	0.784	0.755	F.11 Triple combi.	0.765	0.840	0.820
F.6 Normal dir.	0.432	0.582	0.533	F.12 Simple surflet combi.	0.779	0.878	0.866
				F.13 Ext. surflet combi.	0.803	0.865	0.848

Table 3: Test results as true positive rate for all learning algorithms and features of our artificial data set.

4 Discussion

The discussion of the results is split into one section on the classification results and one section on the quality and influence of the training data.

4.1 Classification results

The geometric features of Section 2.1 categorized as point-based, normal-based, curvature-based, and combined features are discussed separately.

With a true-positive-rate of 0.758 the tetrahedron volume feature F.4 has the highest true-positive-rate among point-based features. While performing well for planes it confuses ellipsoids with spheres and tori with cylinders. The reasons for this seems to be that the training data for those classes are very similar, see Table 1, and point-based features do not capture curvature information sufficiently.

The normal angles feature F.5 has a classification rate of 0.784 and by that the highest classification rate among normal-based features. It best performs for planes and cylinders but is weak for classifying ellipsoids and tori, see Table 2.

With 0.695 the curvature angle feature F.7 has the highest classification rate of the curvature-based features. It best classifies planes and has the weakest performance for ellipsoids and tori. This is due to the fact that the training data for curved primitives are relatively uniform, see Table 1. This effect can be observed in all confusion matrices, where basically the same primitives are confused by all features.

The combined features perform almost equally well. The surflet pair feature F.10 performs best for planes, cylinders, and spheres, and has better performance for cones, ellipsoids, and tori than F.4 or F.7. However, it confuses cylinders with tori, see Table 2. The simple surflet combination feature F.12 performs best among all described features. It also separates spheres and ellipsoids as well as cylinders and tori best, see Table 2.

For all features cSVM results were superior to LDA and oSVM results. The reason for this is that oSVM only labels results with non-negative margin and LDA can only handle linear separation.

Using cSVM with the simple surflet combination feature F.12 we colored a real scan of a wooden toy firetruck, see Figure 5. Because of static patch size in the training process the point cloud was scaled to 10% of its original size. The three main colors are red, blue, and yellow which correspond to cones, planes, and tori, respectively. While planes and tori are often labeled right, transitions between different primitives are classified as cones. Taking into account the cone hypersurface from Figure 4 one can see that the cone class border is very close to all other feature classes. This results in frequent miss-classification as cones.

4.2 Training data

One disadvantage of using machine learning for primitive classification is the need for a large set of training data. Since there is no simple way to collect and produce enough real scans for training we uased simulated scans. However, our scan simulator generates scans that have different characteristics compared to real scans. They are different in density and uniformity of the scanned point clouds. Distribution of the scan lines across the surface of the scanned object are different for each human operator. Usually real laser scanners are manually guided and therefore distances between the scan lines vary. The scan simulator rotates the fan by a defined angle and generates uniform distances between different scan lines.

The error model of our simulator only covers errors in the measured distance and in the position of the laser or the camera. Error caused by specular reflections are not simulated, but can occur for polished objects. This could be solved by using real scans for the training or extending the scan simulator. Improving the scan simulator could be done by using real random scan paths instead of simple random points for sweeping and including an error model for surface reflections.

Patches extracted from point clouds typically contain a few hundred points. For training, those patches were extracted from point clouds containing only one primitive. When classifying real scenes, patches will often contain not only a part of one, but multiple or no primitives at all. In open classification those

patches might remain unclassified. In closed classification they would be classified to the wrong class. Decreasing the size of the input patch would make classification of smaller patches possible. The smallest possible patch would be the local neighborhood of one point.

We estimate curvatures and normals using the 100-nearest neighbors of a point. For our simulated point clouds this value was sufficient to yield stable values for normals and curvatures. In case the point cloud is very dense, more points are needed to cover a sufficiently large patch for estimation. The same applies for sparse point clouds were too many neighbors result in insufficient neighborhoods. Using small neighborhoods in very noisy point clouds would lead to high approximation errors. Because features based on normals and curvatures heavily rely on point neighborhood, they are not scale invariant.

5 Conclusion

We present a method for primitive recognition based on point cloud classification using support vector machines. Our set of features is normal-, point-, and curvature-based. Based on simulated scans a large set of training data was generated to train and compare results of LDA and open/closed SVM classification. The geometric features were compared with respect to true-positive-rates and optimal SVM parameters. Resulting classifiers were applied to a real scan.

We have evaluated the discriminative power of different features and feature combinations for primitive classification. Especially closed SVM classification used with a combined feature is showing promising results in classifying primitives. Results of curvature based features did not meet our expectations. This might be because of high similarity between some classes within the training set. By using our method as a pre-processing step in fitting, deciding the geometric primitive for noisy point clouds can be improved. This might also lead to less iterations for iterative fitting methods.

For future work we intend to detail the pros and cons of various geometric features and their dependence on the distribution of the training data for SVM classification. To generate simulated scans that match real scans as close as possible is another aspect we plan to investigate. Additionally we plan to extend the machine learning methods to multi-output SVMs and other methods from machine learning to allow for a lager set of geometries to recognize.

References

[AFB*12] Arbeiter G., Fuchs S., Bormann R., Fischer J., Verl A.: Evaluation of 3d feature descriptors for classification of surface geometries in point clouds. In *Inter. Conf. on Intelligent Robots and Systems* (2012), pp. 1644–1650.

[CGA] CGAL, Computational Geometry Algorithms Library. www.cgal.org.

[CP05] Cazals F., Pouget M.: Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design 22*, 2 (2005), 121–146.

[CST00] Cristianini N., Shawe-Taylor J.: *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[DHR*13] Denker K., Hagel D., Raible J., Umlauf G., Hamann B.: On-line reconstruction of CAD geometry. In *Inter. Conf. on 3D Vision* (2013), pp. 151–158.

[DHS01] Duda R. O., Hart P. E., Stork D. G.: *Pattern Classification.* Wiley, 2001.

[EYO12] Endoh M., Yanagimachi T., Ohbuchi R.: Efficient manifold learning for 3d model retrieval by using clustering-based training sample reduction. In *Inter. Conf. on Acoustics, Speech and Signal Proc.* (2012), pp. 2345–2348.

[FB81] Fischler M. A., Bolles R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[GKF09] Golovinskiy A., Kim V. G., Funkhouser T.: Shape-based recognition of 3D point clouds in urban environments. *Inter. Conf. on Comp. Vision* (2009).

[HHYR12] Hwang H., Hyung S., Yoon S., Roh K.: Robust descriptors for 3d point clouds using geometric and photometric local feature. In *Inter. Conf. on Intelligent Robots and Systems (IROS)* (2012), pp. 4027–4033.

[HLLS01] Hetzel G., Leibe B., Levi P., Schiele B.: 3d object recognition from range images using local feature histograms. In *CVPR* (2001), pp. 394–399.

[IHMG08] Igel C., Heidrich-Meisner V., Glasmachers T.: Shark. *Journal of Machine Learning Research 9* (2008), 993–996.

[KvD92] Koenderink J. J., van Doorn A. J.: Surface shape and curvature scales. *Image Vision Comput. 10*, 8 (1992), 557–565.

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87* (1987), pp. 163–169.

[MS09] Mahmoudi M., Sapiro G.: Three-dimensional point cloud recognition via distributions of geometric distances. *Graphical Models 71*, 1 (2009), 22–31.

[OFCD02] Osada R., Funkhouser T., Chazelle B., Dobkin D.: Shape distributions. *ACM Transactions on Graphics 21*, 4 (2002), 807–832.

[RMBB08] Rusu R. B., Marton Z. C., Blodow N., Beetz M.: Persistent point feature histograms for 3d point clouds. In *10th Inter. Conf. on Intelligent Autonomous Systems* (2008).

[SS01] Schoelkopf B., Smola A. J.: *Learning with Kernels.* MIT Press, 2001.

[WHH03] Wahl E., Hillenbrand U., Hirzinger G.: Surflet-pair-relation histograms: A statistical 3d-shape representation for rapid classification. In *3DIM* (2003), IEEE, pp. 474–482.

[YTSO08] Yamamoto A., Tezuka M., Shimizu T., Ohbuchi R.: SHREC '08 entry: Semi-supervised learning for semantic 3d model retrieval. In *Inter. Conf. on Shape Modeling and Applications* (2008).

A Additional geometric features and model selection

For comparison additional geometrical features were tested, see Table 4. For optimization results of the SVM model selection refer to Table 5.

A.1 *Centroid distances* are computed as distance of random points to the bounding box centroid, [OFCD02].

A.2 *Cube cell count* is computed by subdividing the point cloud's bounding box into equally sized cells and counting the points in each cell. This feature is not invariant to rotation.

A.3 *K-Median points* are computed by clustering the points into k clusters, such that the sum of distances of points in the cluster to their median is minimized. The coordinates of the resulting medians are concatenated to one feature vector of size $3k$.

A.4 The moduli of *principal curvatures* κ_1, κ_2 as in Section 2.1 yield two concatenated histograms.

A.5, A.6, and A.7 *mean curvatures* $H = (\kappa_1 + \kappa_2)/2$, *Gauss curvatures* $K = \kappa_1\kappa_2$, and *curvature ratios* $|\kappa_1/\kappa_2|$.

A.8 *Curvature changes* are computed as the absolute difference between a random point's principal curvatures and those of its nearest neighbor and yield two concatenated histograms.

A.9 *Shape indices* as in [KvD92].

The histograms of the geometric features A.5, A.6, A.7, and A.8 are cropped to range between the 0.05 and 0.95 percentiles. All these features are less effective in the experiments with a true-positive-rate below 0.5.

Feature Histogram	LDA	cSVM	oSVM	Feature Histogram	LDA	cSVM	oSVM
A.1 Centroid distances	0.417	0.465	0.153	A.6 Gauss curvature	0.214	0.241	0.021
A.2 Cube cell count	0.340	0.449	0.358	A.7 Curvatures ratio	0.286	0.302	0.035
A.3 K-Median points	0.164	0.240	0.015	A.8 Curvature change	0.260	0.277	0.014
A.4 Principal curvatures	0.271	0.301	0.034	A.9 Shape index	0.296	0.302	0.034
A.5 Mean curvature	0.237	0.268	0.030				

Table 4: True positive rate of additional geometric features.

Feature Histogram	C	γ	Feature Histogram	C	γ
F.1 Point angles	10000	0.4	F.12 Simple surflet combinations	1000	0.1
F.2 Point distances	10000	0.6	F.13 Extended surflet combinations	50	0.1
F.3 Triangle areas	10000	0.25	A.1 Centroid distances	20000	0.01
F.4 Tetrahedron volumes	1000	2	A.2 Cube cell count	100	0.1
F.5 Normal angles	20000	0.25	A.3 K-Median points	0.01	0.01
F.6 Normal directions	50	0.5	A.4 Principal curvatures	0.1	2
F.7 Curvature angles	100	0.1	A.5 Mean curvature	0.1	5
F.8 Curvature directions	5	0.1	A.6 Gaussian curvature	0.1	20
F.9 Curvature differences	100	0.6	A.7 Curvatures ratio	0.1	5
F.10 Surflet pairs	10000	0.3	A.8 Curvature change	0.01	1.25
F.11 Triple combinations	50	0.1	A.9 Shape index	0.1	5

Table 5: SVM parameters C and γ for best classification results for each feature.