3D Object Classification via Part Graphs

Florian Teich[a], Timo Lüddecke[b] and Florentin Wörgötter[c]
III. Physikalisches Institut, Georg-August Universität Göttingen, Friedrich-Hund-Platz 1, Göttingen, Germany

Keywords: 3D Classification, Segmentation, Graphs.

Abstract: 3D object classification often requires extraction of a global shape descriptor in order to predict the object class. In this work, we propose an alternative part-based approach. This involves automatically decomposing objects into semantic parts, creating part graphs and employing graph kernels on these graphs to classify objects based on the similarity of the part graphs. By employing this bottom-up approach, common substructures across objects from training and testing sets should be easily identifiable and may be used to compute similarities between objects. We compare our approach to state-of-the art methods relying on global shape description and obtain superior performance through the use of part graphs.

1 INTRODUCTION

Classifying objects based on their shape and appearance is essential to many applications such as autonomous vehicles, assistive robots and scene understanding. For example in robotics, it is crucial for 3D shape classification algorithms to make correct predictions about what class an object belongs to. Driven by the trend of 2.5D- and 3D-devices becoming cheaper, the focus on 3D vision tasks has increased recently. State-of-the-art methods (Qi et al., 2017b; Kanezaki et al., 2018) usually treat the object as a whole, reduce the global shape to a global 3D feature - explicitly or implicitly encoded - and employ classification techniques such as Multi-Layer-Perceptrons (Qi et al., 2017a), single Fully-Connected layer (Li et al., 2018; Kanezaki et al., 2018) or Support Vector Machines (SVMs) (Yang et al., 2017; Su et al., 2015; Yu et al., 2018; Wang et al., 2019a) to learn distinguishing different object classes. This global approach fails to make explicit use of semantic substructures of the objects: their parts. Object concepts like tables often consist of parts, i.e. a *tabletop* and a few *legs*. Decomposing objects into their parts and reasoning about the object classes using these parts might show much better generalization than standard techniques that only use global 3D descriptors. Substructures already seen in the training set can be exploited to increase the certainty that the object belongs to a specific class. For instance, during training time only table instances with four table legs may be seen, the test set may consist of tables with many more legs. In this case, systems that make use of part graphs are able to identify common structures in the tables from the training and the test set and associate these instances from the same class easily, whereas classical methods may fail.

In his theory of Recognition-by-Components (Biederman, 1987), Biederman theorizes that humans perceive objects as compositions of volumetric primitives - *geons* - and recognize objects based on them in a bottom-up manner. Following this theory, we present a pipeline to classify 3D shapes by first decomposing an object into its parts, creating attributed part graphs and finally classifying these graphs to predict the shapes' classes.

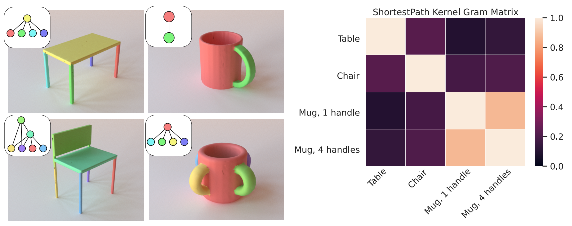


Figure 1: Objects are automatically segmented and part graphs are created. Shape descriptors of parts serve as node attributes in the graph. Relying on this, graph kernels can be used to estimate similarity of objects.

[a] https://orcid.org/0000-0001-6708-7233
[b] https://orcid.org/0000-0002-1643-7827
[c] https://orcid.org/0000-0001-8206-9738

Teich, F., Lüddecke, T. and Wörgötter, F.
3D Object Classification via Part Graphs.
DOI: 10.5220/0010232604170426
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 417-426
ISBN: 978-989-758-488-6

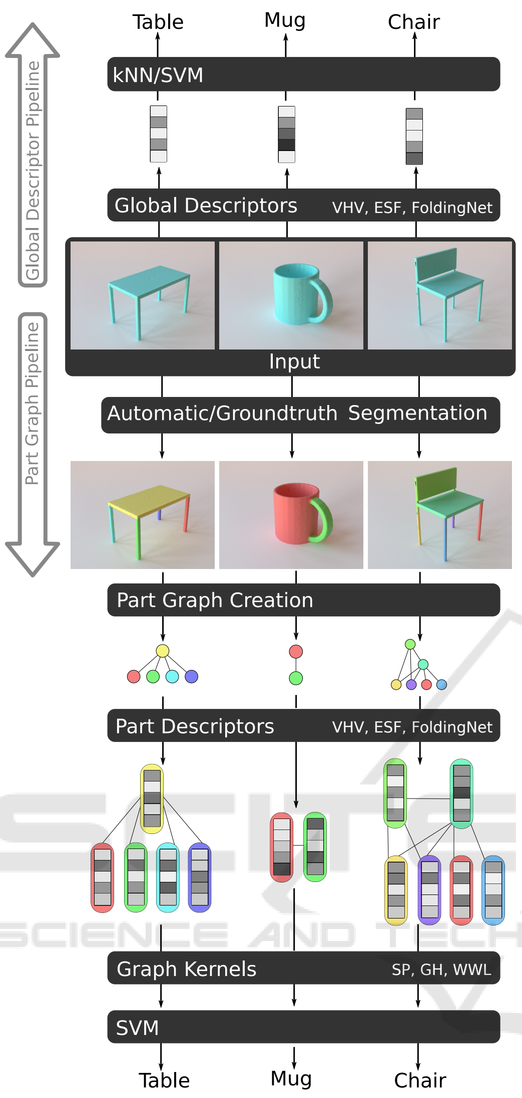


Figure 2: Overview for global descriptor pipeline and the part graph pipeline. For the former, global descriptors are extracted from the input shapes (here either VFH, ESF or FoldingNet), which are fed to the SVM. For the part graph pipeline, the input shape is first decomposed into its parts, using groundtruth- or automatic segmentation. Subsequently, the part graph is created and the parts' descriptors are extracted (again, either VFH, ESF or FoldingNet descriptors). Finally, graph kernels (ShortestPath, GraphHopper or Wasserstein Weisfeiler-Lehman) are used to compute the kernel matrix between all graphs to train a SVM.

2 METHOD OVERVIEW

For the decomposition of objects into parts, various segmentation algorithms already exist that try to find part boundaries on the surface to result in a part clustering of the object. Afterwards, a graph is created where nodes represent part instances of the object and edges indicate spatial adjacency of these part instances. This process is illustrated in Fig. 2. Different objects may contain a different number of parts, therefore it is not straightforward how to compute similarities between the obtained part graphs. In this work we employ graph kernels to solve this problem. Furthermore, we compare ground truth segmentations and automatic segmentations that result in different part-graphs and employ various 3D shape descriptors in order to benchmark our approach against classical classification methods as k-Nearest-Neighbor (kNN) and Support Vector Machines on the global shape descriptors.

The underlying hypothesis of this work is that an explicit decomposition into parts helps in identifying untypical objects. Our contributions are:

- A comparison of different graph kernels on part graphs,
- the identification of useful node attributes (3D descriptors) for the part graphs,
- a benchmark against global shape descriptor classifiers.
- and the estimation of the potential of this method when an optimal segmentation is present.

3 RELATED WORKS

3.1 Classical 2D Image Classification

Object classification is one of the central problems in Computer Vision. Given a set of possible classes and a query object, the task is to decide which class the object belongs to. Most traditional image classifiers start by extracting features such as SIFT (Lowe, 1999) or SURF (Bay et al., 2006) from the image. In different images, a varying amount of keypoints and patches may be detected and reduced to their feature vectors, making it non-trivial to compare two sets (of different sizes) of feature vectors extracted from different images. The Bag-of-visual-words (Sivic and Zisserman, 2003) method tackles this issue by identifying codewords (cluster centroids of the feature vectors found in the training set). All feature vectors detected inside an image are then classified as codewords and the image can be reduced to the histogram of codewords present inside. A classifier, e.g. an SVM (Cortes and Vapnik, 1995) can then be employed to classify a query image based on its fixed size feature histogram.

3.2 3D Shape Classification

Ideas and concepts from 2D classification have been widely adopted in 3D shape classification.

In (Bronstein et al., 2011), a set of 3D feature descriptors were used to create bag-of-visual-words for shape classification and retrieval, analogous to 2D image classifiers (Sivic and Zisserman, 2003; Fei-Fei and Perona, 2005).

Maturana et al. (Maturana and Scherer, 2015) use a 3D CNN to classify voxelized representations of 3D shapes. Their method suffers from discretization of the input shapes and long training times due to the additional dimension compared to 2D CNNs.

The Multi-view method in (Su et al., 2015) renders the 3D object from different views and applies a CNN on these input images. The activations from the different views are merged by using a max-pool operation and sent to the second CNN. At testing time, activations in the last layer of the first CNN for all different views are evaluated and each is fed to a one-vs-rest linear SVMs. The predictions of the SVMs are summed up and the highest scoring class is used as the final class prediction. This multi-view method leads to highly accurate classifiers in many cases, often surpassing performance of pure 3D methods (Qi et al., 2016).

Another popular 3D object classification method is the Deep Neural Network architecture PointNet (Qi et al., 2017a). PointNet transforms the input point cloud into a high dimensional feature space and applies Max-Pooling to reduce the point set to a global feature vector, followed by a Multi-Layer-Perceptron (MLP) to predict the final object class.

3.3 3D Descriptors

Classical 3D feature descriptors include Viewpoint Feature Histogram (VFH) (Rusu et al., 2010), Ensemble of Shape Functions (ESF) (Wohlkinger and Vincze, 2011), Signature of Histograms of Orientations (SHOT) (Tombari et al., 2010), and the Spherical Harmonic Shape Descriptor (SPH) (Kazhdan et al., 2003). These descriptors capture several properties of the shape as e.g. pairwise distances of points or angles between their normal vectors and encode this information in histograms.

In the recent past, more methods started to use Deep Neural Networks to extract global 3D features from shapes. Neural Networks are trained on classification (Su et al., 2015) or reconstruction (Wu et al., 2016; Achlioptas et al., 2017; Yang et al., 2017) tasks. After training, a forward pass in the network for any new query 3D shape can be used to extract the activations inside the last hidden layer of the network to obtain the global 3D shape feature. It is important to note that the resulting descriptors may contain negative elements and the values of the individual elements are not bounded, whereas in classical 3D histogram descriptors, values between 0 and 1 are obtained, usually summing up to 1 if histograms are normalized.

3.4 3D Segmentation

In recent years, 3D mesh data became more accessible and 3D benchmark datasets were released for tasks such as classification and part segmentation (Chen et al., 2009; Wu et al., 2015; Chang et al., 2015; Yi et al., 2016). In (Chen et al., 2009), the authors compare various automatic and manual segmentation approaches on a set of meshes. The meshes are clean and do not contain any background or clutter. The challenge of the dataset is to segment a given object into its parts. Additionally, the authors provide human-annotated segmentations of the meshes and suggest metrics to measure segmentation performance.

The Randomized Cuts method, proposed by Golovinskiy and Funkhouser (Golovinskiy and Funkhouser, 2008) clusters the 3D shape multiple times such that edges, where clusters meet, can be identified as segmentation boundaries if enough clusterings voted for these boundaries. The shape diameter function in (Shapira et al., 2008) is used to estimate the local diameter of the objects' volume at various locations to cluster the shape into regions of varying diameters. In (Lien and Amato, 2007), shapes are decomposed into roughly convex subgroups resulting in a partitioning of the object. Apart from classic approaches, data-driven methods (Qi et al., 2017a; Kalogerakis et al., 2010; Le et al., 2017; George et al., 2018) were developed. These methods are usually trained to identify part boundaries (Le et al., 2017) in order to cluster the object or to densely classify all vertices/faces of the object into specific part classes (Qi et al., 2017a; George et al., 2018; Kalogerakis et al., 2010).

3.5 3D Shape Classification using Graphs

Graph Convolutional Networks (GCN) (Kipf and Welling, 2016) were explored as a way to generalize deep neural networks to graphs. Due to the inherent representation of surface meshes as connected vertices, GCNs can easily be used on these 3D shapes (Monti et al., 2017; Hanocka et al., 2019). Even on unorganized point clouds, methods emerged that create adjacencies between the points to create such graph structures and make it possible to use graph convolutions on these structures (Wang et al., 2019b). In all these works, the nodes of the graphs that are processed represent low-level entities, usually points of the point cloud or vertices of the mesh. In our work, nodes represent parts - geometric and semantic entities that capture entire substructures of the objects.

The work most similar to methods in this paper is from Schoeler (Schoeler and Wörgötter, 2015). In their work, artificially created 3D pointclouds of household tools are classified to determine their affordances (e.g. hit, poke, sieve, cut, contain, etc.). Objects are first segmented by an automatic segmentation method and multi-class SVMs are trained on descriptors from parts to predict object class and affordances. To compare part graphs at testing time, Schoeler defines custom distance measures between part- and pose-signatures. The dataset used in (Schoeler and Wörgötter, 2015) is quite small (144 objects) and partially homogeneous (objects only consisting of volumetric cuboids, cylinders, cones and spheres). Furthermore, the work in (Schoeler and Wörgötter, 2015) lacks an analysis of different graph kernels and shape descriptors from deep neural networks.

4 METHOD

4.1 Overview

Our proposed method tries to tackle the task of object classification in a bottom-up manner. First, the object is decomposed into its parts by using the automatic 3D mesh segmentation algorithm proposed by Au et al. (Au et al., 2011). In the remainder of this article, we call this method automatic segmentation to distinguish it from ground truth segmentation. Using this decomposition, a part graph is created for each object by connecting parts via edges when they are spatially close to each other. For each node, a 3D feature descriptor is extracted on the underlying part. Finally, similarities between all graphs are computed to create a kernel matrix and train an SVM, as shown in Fig. 1. We constrain our analysis to SVMs which have shown State-of-the-art results recently (Wang et al., 2019a). This work features a comparison of different descriptors.

4.2 Object to Part Decomposition

In order to decompose objects into parts, we rely on the algorithm from (Au et al., 2011). The segmentation algorithm assumes a 2-Manifold, that contains no defects as vertices or edges occupying the same space, as input. The idea is to define fields on the surface with high field variation in concave surface regions and less in planar and convex regions. A field is defined by a source and a sink vertex, drawn from the set of extreme points on the mesh. These extreme points are ”mesh vertices located at prominent parts” (Au et al., 2011), usually at extremities. From the created fields, isolines are drawn and a subset of these lines is selected as hypothetical part-boundaries. Using this algorithm, we obtain segmented (decomposed) objects.

4.3 3D Descriptors

Our methods allows using any feature descriptor. In this work, we use three different descriptors: VFH, ESF and FoldingNet-Descriptor. Viewpoint Feature Histogram (VFH) (Rusu et al., 2010) and Ensemble of Shape Functions (ESF) are classical 3D descriptors working on pointclouds without any further requirements, whereas FoldingNet is a neural network-based descriptor, which requires training on an external dataset.

4.3.1 Ensemble of Shape Functions (ESF)

ESF (Wohlkinger and Vincze, 2011) combines three descriptors, namely D2, D3 and A3, covering distances, angles and area of point pairs or triplets sampled from the point cloud. These statistics can be normalized and binned to create histograms. For each of the three descriptors, three histograms are created (by categorizing each sampled pair/triplets into inside/outside/in-between the surface). An additional modification of the D2 histogram is appended, resulting in 10 histograms, each having 64 bins. Thus, the final shape descriptor has a length of 640 and the elements are between 0 and 1, summing up to 10 in total due to the normalization of the histograms.

4.3.2 Viewpoint Feature Histogram (VFH)

VFH extends the popular Fast Point Feature Histogram (FPFH) (Rusu et al., 2009) descriptor that captures histograms for angular features. We used the VFH implementation from the Point Cloud Library (PCL) (Rusu and Cousins, 2011). This implementation also contains an additional histogram that covers the distances between the centroid and the individual points of the point cloud. The final descriptor has 308 elements, all between 0 and 1.

4.3.3 FoldingNet

The FoldingNet descriptor (Yang et al., 2017) is trained in a autoencoder-like manner. During training, the autoencoder learns to represent pointclouds by a fixed-size vector (1024 elements). The encoder part reduces the point cloud to such a vector and the decoder reconstructs the pointcloud. As reconstruction loss, the Chamfer distance is used. We train the autoencoder on classes of the ShapeNet dataset that are not inside the subset of the 16 here selected classes and augment the data by applying random rotations. After 200 epochs, we stop the training procedure and use the encoder part of the network to generate FoldingNet descriptors for the objects. All input point clouds during training and testing are centered at the origin and scaled to a unit sphere.

All these three description methods permit computing local descriptors for the individual object parts as well as global descriptors for entire shapes. The global descriptor is combined with classification methods (kNN and SVM) but also appended to each local part descriptor in order to capture local as well as global properties inside the part graph.

4.4 Graph Kernels for Part-graph Classification

Given object segmentations, part graphs may be created by representing object parts as nodes and adjacency of parts by edges inside the graph. Segmentations may be provided by automatic methods such as the concavity-aware method in Section 4.2 or by human annotations of the object parts. Since each node is attributed to a 3D descriptor of the part it is representing, in the following we will focus on methods for continuous attributes only and dismiss discrete label attributes. In fact, we append the global shape descriptor of an object to each node inside its part graph. Thus, nodes have attributes of length 616 (VFH), 1280 (ESF) or 2048 (Foldingnet).

In our experiments we consider different graph kernels for continuous attributes : ShortestPath (Borgwardt and Kriegel, 2005), GraphHopper (Feragen et al., 2013) and Wasserstein Weisfeiler-Lehman (Togninalli et al., 2019). Graph kernels are used to determine similarity between two graphs. A tuple (V,E) of a set of vertices V and edges $E \subseteq \{\{u,v\} \subseteq V | u \neq v\}$ is considered a graph G. In order to compute the similarity of two graphs, graph kernels often break down the two graphs into substructures as e.g. subgraphs, paths, walks, nodes or edges. On pairs of such substructures, another kernel may be directly used to compute a measure of similarity (Kriege et al., 2020). In the following paragraphs, we showcase the three graph kernels that are employed in our experiments. For further information about graph kernels we refer to Kriege et al. (Kriege et al., 2020) who compiled a survey paper including many more graph kernel methods.

4.4.1 ShortestPath Graph Kernel (SP)

For the ShortestPath graph kernel, graphs are decomposed into shortest paths, which can be compared against each other whether paths have the same length. First, the graphs have to be transformed into shortest path graphs. The set of nodes stays unchanged, but edges in the shortest path graph S are created between all nodes that are connected by a path in the original graph G. The edges in S are labeled by the distances of the shortest paths they represent. Computing shortest paths for all pairs of vertices is usually accomplished by using the Floyd-Warshall algorithm (Floyd, 1962). Having transformed two graphs G_i, G_j into their shortest path graphs S_i, S_j, the shortest path kernel is defined as:

$$k(S_i,S_j) = \sum_{e_i \in E_i} \sum_{e_j \in E_j} k_{path}(e_i,e_j), \tag{1}$$

where E_i, E_j are the sets of edges inside S_i, S_j.

$$\begin{aligned} k_{path}(e_i,e_j) = k_v(v_i,v_j) * k_e(e_i,e_j) * k_v(u_i,u_j) + \\ k_v(v_i,u_j) * k_e(e_i,e_j) * k_v(u_i,v_j), \end{aligned} \tag{2}$$

where $e_i = \{v_i,u_i\}$ and $e_j = \{v_j,u_j\}$. For the implementation of the edge kernel k_e, the Dirac function is used to compare the lengths of the two edges. For k_v, we used different node kernels for the descriptors, either a linear kernel (in case of the FoldingNet descriptor):

$$k_v(u,v) = u^T * v, \tag{3}$$

or the histogram-intersection (in case of VFH & ESF):

$$k_v(u,v) = \sum_{k=1}^{m} \min(u_k,v_k). \tag{4}$$

The ShortestPath kernel can be calculated between all graphs at training time. The results are normalized as:

$$\frac{k(G_i,G_j)}{\sqrt{k(G_i,G_i)k(G_j,G_j)}}. \tag{5}$$

The resulting matrix can be used by an SVM to find decision boundaries between each pair of classes (”OVO”) or one class vs. the rest of classes (”OVR”).

During testing time, the new ShortestPath kernel between the new graph and all graphs inside the training set is calculated to result in a feature vector that can be classified by the SVM to obtain a class prediction.

4.4.2 GraphHopper (GH)

The GraphHopper (Feragen et al., 2013) kernel considers the length of the path as well as distances between visited nodes' attributes along the path.

$$k(G_i, G_j) = \sum_{\pi \in P} \sum_{\pi \in P'} k_{path}(\pi, \pi'), \tag{6}$$

with sets of shortest paths P, P' from G_i and G_j. Path kernel k_{path} is only calculated if the length of both paths is identical:

$$k_{path}(\pi, \pi') = \begin{cases} \sum_{j=1}^{|\pi|} k_v(\pi(j), \pi'(j)), & \text{if } |\pi| = |\pi'|, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Here, k_v is the node kernel. In (Feragen et al., 2013), the authors show that this GraphHopper kernel can be efficiently implemented, such that the computational complexity is considerably lower compared to the ShortestPath kernel.

4.4.3 Wasserstein Weisfeiler-Lehman (WWL)

The recently developed Wasserstein Weisfeiler-Lehman (WWL) graph kernel (Togninalli et al., 2019) makes use of two existing concepts: the Weisfeiler-Lehman Framework (Shervashidze et al., 2011) and the Wasserstein metric (Vaserstein, 1969). The Wasserstein distance, also known as the earth mover's distance can be seen as the minimum cost to turn probability distribution P into probability distribution Q, when interpreting them as piles of dirt and considering moving amounts of dirt a cost.

The Weisfeiler-Lehman (WL) algorithm can be applied iteratively to a graph by substituting the original node labels of each of the nodes by creating a multiset label for each of them by appending neighboring labels in a sorted manner to the original node label. Subsequently, the resulting multiset label is reduced to a novel label and the next iteration may begin. The Weisfeiler-Lehman subtree kernel can be used on labeled (non-attributed) graphs. It compares shared subtrees between two graphs by using the WL algorithm explained above. In (Togninalli et al., 2019), the authors provide a novel approach to extending the WL subtree kernel to continuous attributes. Furthermore, Togninalli et al. (Togninalli et al., 2019) suggest a graph embedding based on the concatenation of all nodes across all WL iterations. Using this graph embedding, distances between (continuous) nodes or graphs can be defined. The authors chose the euclidean distance for node features and the Wasserstein distance for calculating distances between entire graphs such that kernel matrices can be constructed.

5 EXPERIMENTS

5.1 Object Classification

We are comparing the classification performance of various graph kernels on automatically segmented objects as well as groundtruth segmentations and compare this with global shape classifiers.

Experimental Setup. For the experiments, a subset of the ShapeNet dataset is used. The dataset from (Kalogerakis et al., 2017) provide consistent groundtruth part annotations, such that this oracle-segmentation can be compared to the automatic segmentation explained in 4.2. The used dataset contains objects from 16 different classes. We used a smaller subset (40-55 instances per class) in order to speed up the experiments. The segmentation algorithm in 4.2 requires the objects to be individual separated components. However, the majority of shapes in the ShapeNet dataset consists of multiple connected components. Thus, a preprocessing step to generate 2-Manifolds from the set of all shapes is used (Huang et al., 2018). 100 random splits of the objects were created, each consisting of 20% Test Data and 80% Training Data. For the ShortestPath and GraphHopper kernels, we use the implementation from the GraphKernelLibrary ("GraKeL") (Siglidis et al., 2018), for the WWL graph kernel, we use the official Python implementation of the authors.

Table 1 shows the classification performance under different settings. For each of the three descriptors VFH, ESF and FoldingNet, eight different classification methods were trained. The latter are grouped into global description-based and graph-based.

- kNN (global): a k-Nearest-Neighbor classifier with distance metric $\ell 1$ and $k = 10$, using the feature descriptor obtained from the global shape.
- SVM (global): a linear C-SVM classifier using the feature descriptor obtained from the global shape.
- SP and GH, automatic segmentation: a C-SVM using the kernel matrix obtained from applying the ShortestPath/GraphHopper Graph-Kernel to the dataset of part graphs generated by the

Table 1: Classification accuracy of tested methods. The columns refer to different descriptors.

Classifier		VFH	ESF	FoldingNet
Global descriptor-based Pipeline				
kNN		62.26 ± 2.86	73.73 ± 2.95	92.43 ± 1.87
SVM		65.32 ± 3.46	79.89 ± 2.92	95.84 ± 1.58
Graph-based Pipeline				
GH	automatic seg.	71.96 ± 3.00	78.32 ± 3.04	95.43 ± 1.57
SP	automatic seg.	75.38 ± 3.13	78.68 ± 3.12	92,81 ± 1.82
WWL	automatic seg.	**76.86 ± 2.85**	**82.46 ± 2.82**	**96.49 ± 1.41**
GH	groundtruth seg.	82.31 ± 2.78	85.88 ± 2.51	96.60 ± 1.41
SP	groundtruth seg.	84.16 ± 2.28	85.67 ± 2.76	94.86 ± 1.48
WWL	groundtruth seg.	84.33 ± 2.77	89.26 ± 2.58	97.38 ± 1.11

concavity-aware automatic segmentation method from (Au et al., 2011).

- SP and GH, groundtruth segmentation: a C-SVM using the kernel matrix obtained from applying the ShortestPath/GraphHopper graph kernel to the dataset of part graphs generated by the annotated groundtruth segmentations provided in (Kalogerakis et al., 2017).
- WWL, automatic segmentation: a Krein SVM (Loosli et al., 2015) using the kernel matrix obtained from applying the Wasserstein Weisfeiler-Lehman graph kernel to the dataset of part graphs generated by the concavity-aware automatic segmentation method from (Au et al., 2011).
- WWL, groundtruth segmentation: a Krein SVM using the kernel matrix obtained from applying the Wasserstein Weisfeiler-Lehman graph kernel to the dataset of part graphs generated by the annotated groundtruth segmentations provided in (Kalogerakis et al., 2017).

For VFH and ESF, only the kNN classifier is used in the original papers (Rusu et al., 2010). Since the graph classification is using SVMs after calculating the kernel matrix, we included the global descriptor SVM to have a more similar baseline. For all SVMs, grid search was applied to optimize the parameter C individually.

The performance of the kNN classifier is low for all three descriptors, compared to the global descriptor SVM and the graph kernel SVMs. Table 1 also shows that classification accuracy is much higher when employing the graph kernels using automatic segmentation, in comparison to a global descriptor SVM classifier for the two histogram descriptors VFH and ESF. For the FoldingNet descriptor, only GH and WWL graph kernels show better performance than the global descriptor SVM baseline. The WWL graph kernel performance is - in all cases - higher than SP and GH, both for the automatic, as well as for the groundtruth segmentations.

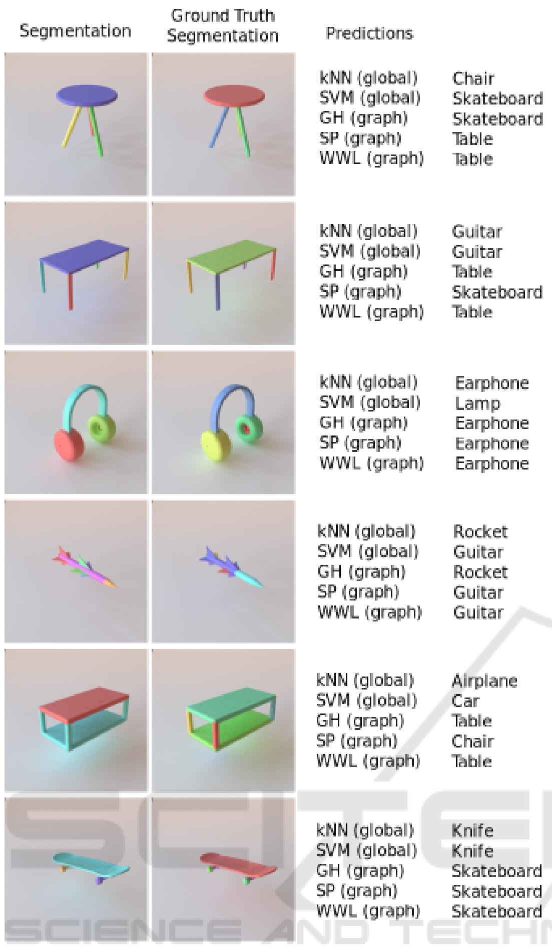
Furthermore, employing the graph kernels on groundtruth segmentations of objects always shows improvements over part graphs from automatic segmentations. The results of the groundtruth segmentation may be seen as an upper bound of what would be possible to achieve by using good automatic segmentation algorithms.

5.2 Object Classification on Out-of-Distribution Objects

We created 50 artificial objects in total from three different object classes (chairs, mugs, tables) depicted in Figure 4. These objects are designed to be peculiar and hence differ from the objects in the original ShapeNet dataset: mugs now have more than one handle (up to four) and chairs and tables contain more than four legs (up to 10).

With this experiment we want to assess the generalization performance of both classical classifiers using global shape features and the graph kernel methods. As training set, all objects from the first experiment are used. For the test, the 50 newly created objects are used.

The results in Table 2 show that the FoldingNet descriptor generalizes well to the new shapes, contrary to VFH and ESF. All three graph kernel methods improve the accuracy of the classification compared to the global descriptor SVM in case of the FoldingNet descriptor. The ShortestPath kernel surpasses the global 3D feature SVM in all three cases, whereas the GraphHopper classifier only shows superior results in case of VFH and FoldingNet. The WWL kernel outperforms all other methods only for the ESF descriptor and achieves the same accuracy as



kNN (global)	Chair
SVM (global)	Skateboard
GH (graph)	Skateboard
SP (graph)	Table
WWL (graph)	Table
kNN (global)	Guitar
SVM (global)	Guitar
GH (graph)	Table
SP (graph)	Skateboard
WWL (graph)	Table
kNN (global)	Earphone
SVM (global)	Lamp
GH (graph)	Earphone
SP (graph)	Earphone
WWL (graph)	Earphone
kNN (global)	Rocket
SVM (global)	Guitar
GH (graph)	Rocket
SP (graph)	Guitar
WWL (graph)	Guitar
kNN (global)	Airplane
SVM (global)	Car
GH (graph)	Table
SP (graph)	Chair
WWL (graph)	Table
kNN (global)	Knife
SVM (global)	Knife
GH (graph)	Skateboard
SP (graph)	Skateboard
WWL (graph)	Skateboard

Figure 3: Qualitative classification results of global classifiers and graph classifiers based on automatic segmentations using the VFH descriptor. Groundtruth segmentation is shown for comparison of segmentations. Note, GH, SP and WWL predictions are based on automatic segmentation from (Au et al., 2011), not groundtruth segmentations.

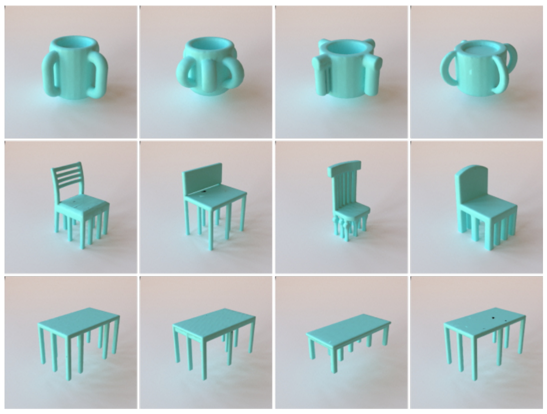


Figure 4: Artificial Object examples.

Table 2: Classification accuracy of tested methods on artificial objects. The columns refer to different descriptors.

Method	VFH	ESF	Folding-Net
Global descriptors			
SVM	72	80	84
Graph-based			
GH, autom. seg.	78	76	96
SP, autom. seg.	**90**	84	**98**
WWL, autom. seg.	76	**88**	**98**

ShortestPath on the FoldingNet descriptor. These results may indicate that the graph kernel methods make better use of substructures in order to predict the final object classes. Thus, these graph kernel methods are recommended to boost classification performance.

6 CONCLUSIONS

This work investigated the utility of representing objects in terms of their substructures. To this end, a comparison on three different graph kernels operating on automatically extracted object parts was conducted. Furthermore, three 3D feature extractors that describe these parts were analyzed regarding their suitability for object classification. A comparison against global (whole-object) descriptors suggests that this part-based approach indeed improves performance, especially if the Wasserstein Weisfeiler-Lehman kernel is used. Our results on part graphs using groundtruth segmentations indicate further potential for improvement by making use of better segmentation algorithm (as soon as this is available). We confirmed the finding of parts being useful with an even stronger effect on a novel dataset of peculiar objects that deviate from standard object class concepts.

Since our approach comes with additional processing steps (mesh cleanup, segmentation, individual descriptor extraction for each object part), the use of this pipeline is recommended in cases where classification accuracy is required to be maximized and additional computation cost is less critical.

REFERENCES

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. (2017). Representation learning and adversarial generation of 3d point clouds. *arXiv preprint arXiv:1707.02392*.

Au, O. K.-C., Zheng, Y., Chen, M., Xu, P., and Tai, C.-L. (2011). Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1125–1134.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg. Springer Berlin Heidelberg.

Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115.

Borgwardt, K. M. and Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE.

Bronstein, A. M., Bronstein, M. M., Guibas, L. J., and Ovsjanikov, M. (2011). Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30(1):1–20.

Chang, A. X., Funkhouser, T. A., Guibas, L. J., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012.

Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3).

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE.

Feragen, A., Kasenburg, N., Petersen, J., de Bruijne, M., and Borgwardt, K. (2013). Scalable kernels for graphs with continuous attributes. In *Advances in neural information processing systems*, pages 216–224.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345.

George, D., Xie, X., and Tam, G. K. (2018). 3d mesh segmentation via multi-branch 1d convolutional neural networks. *Graphical Models*, 96:1–10.

Golovinskiy, A. and Funkhouser, T. (2008). Randomized cuts for 3d mesh analysis. In *ACM SIGGRAPH Asia 2008 papers*, pages 1–12. ACM New York, NY, USA.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D. (2019). Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12.

Huang, J., Su, H., and Guibas, L. (2018). Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*.

Kalogerakis, E., Averkiou, M., Maji, S., and Chaudhuri, S. (2017). 3d shape segmentation with projective convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3779–3788.

Kalogerakis, E., Hertzmann, A., and Singh, K. (2010). Learning 3d mesh segmentation and labeling. In *ACM SIGGRAPH 2010 papers*, pages 1–12. Citeseer.

Kanezaki, A., Matsushita, Y., and Nishida, Y. (2018). Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019.

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2003). Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kriege, N. M., Johansson, F. D., and Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, 5(1):1–42.

Le, T., Bui, G., and Duan, Y. (2017). A multi-view recurrent neural network for 3d mesh segmentation. *Computers & Graphics*, 66:103–112.

Li, J., Chen, B. M., and Lee, G. H. (2018). So-net: Self-organizing network for point cloud analysis. *arXiv preprint arXiv:1803.04249*.

Lien, J.-M. and Amato, N. M. (2007). Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 121–131.

Loosli, G., Canu, S., and Ong, C. S. (2015). Learning svm in kreĭn spaces. *IEEE transactions on pattern analysis and machine intelligence*, 38(6):1204–1216.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee.

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE.

Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J. (2010). Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162. IEEE.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE.

Schoeler, M. and Wörgötter, F. (2015). Bootstrapping the semantics of tools: Affordance analysis of real world objects on a per-part basis. *IEEE Transactions on Cognitive and Developmental Systems*, 8(2):84–98.

Shapira, L., Shamir, A., and Cohen-Or, D. (2008). Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561.

Siglidis, G., Nikolentzos, G., Limnios, S., Giatsidis, C., Skianis, K., and Vazirgiannis, M. (2018). Grakel: A graph kernel library in python. *arXiv preprint arXiv:1806.02193*.

Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953.

Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., and Borgwardt, K. (2019). Wasserstein weisfeiler-lehman graph kernels. In *Advances in Neural Information Processing Systems*, pages 6436–6446.

Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer.

Vaserstein, L. N. (1969). Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72.

Wang, C., Pelillo, M., and Siddiqi, K. (2019a). Dominant set clustering and pooling for multi-view 3d object recognition. *arXiv preprint arXiv:1906.01592*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019b). Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12.

Wohlkinger, W. and Vincze, M. (2011). Ensemble of shape functions for 3d object classification. In *2011 IEEE international conference on robotics and biomimetics*, pages 2987–2992. IEEE.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

Yang, Y., Feng, C., Shen, Y., and Tian, D. (2017). Foldingnet: Interpretable unsupervised learning on 3d point clouds. *CoRR*, abs/1712.07262.

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. (2016). A scalable active framework for region annotation in 3d shape collections. *SIGGRAPH Asia*.

Yu, T., Meng, J., and Yuan, J. (2018). Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 186–194.