Signature of Topologically Persistent Points for 3D Point Cloud Description

William J. Beksi and Nikolaos Papanikolopoulos

***Abstract*— We present the Signature of Topologically Persistent Points (STPP), a global descriptor that encodes topological invariants of 3D point cloud data. These topological invariants include the zeroth and first homology groups and are computed using persistent homology, a method for finding the features of a topological space at different spatial resolutions. STPP is a competitive 3D point cloud descriptor when compared to the state of art and is resilient to noisy sensor data. We demonstrate experimentally on a publicly available RGB-D dataset that STPP can be used as a distinctive signature, thus allowing for 3D point cloud processing tasks such as object detection and classification.**

I. Introduction

The automatic recognition of shapes in 3D data continues to attract growing interest in the research community. Shape recognition is significant to applications such as shape matching and retrieval, object classification, 3D reconstruction, manipulation and grasping, and robot localization and navigation. In recent times, the availability of low-cost 3D sensors (RGB-D, stereo, time-of-flight, etc.) has strongly motivated the development of this technology.

Shape recognition is performed by either a local or global approach. Local descriptors rely on keypoints extracted from surfaces. The aim of descriptors using local methods is to try to single out points that are distinctive in order to allow for effective description and matching. Within the local neighborhood of each keypoint, geometric information is encoded to obtain a compact representation of the input data invariant up to a predefined transformation (translation, rotation, scaling, point density variations, etc.). Global descriptors encode object geometry. They are not computed for individual points, but for a whole cluster of points that represents an object.

Although there has been substantial progress on extracting and encoding discriminative geometric information, only recently have researchers started looking into the topological structure of the data as an additional source of information. The emergence of topological data analysis (TDA) [1] has led to the creation of computational tools capable of identifying topological structure. Since that time, researchers have shown that TDA can capture characteristics of the data that other methods often fail to provide. Contemporary applications of TDA include, but are not limited to, gait recognition [2], activity recognition [3], facial recognition [4], digital forensics [5], discriminating among breast cancer subtypes [6], mesh segmentation [7], 3D point cloud segmentation [8], [9], and many more.

The authors are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA. Emails: {beksi, npapas}@cs.umn.edu.

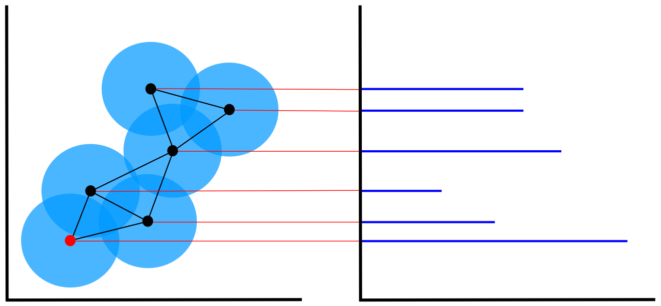
Fig. 1. The evolution of a scale parameter defining the neighborhood radius about each point (left) and the corresponding barcode diagram (right). When the neighborhoods of two points overlap, one dies while the other survives. At the end of this process, the lone surviving point (red) is a point of infinite persistence and is the representative of the set of points which forms a connected component. This procedure of computing topological persistence over varying spatial resolutions is known as a *filtration.*

Persistent homology is a particularly useful tool in TDA [10]. It captures the birth and death times of topological features (e.g. connected components, holes, and higher dimensional analogs), at multiple scales. This information can be visualized through a barcode diagram: a collection of horizontal line segments in a plane where the horizontal axis corresponds to a scale parameter and the vertical axis represents an arbitrary ordering of the homology generators, Fig. 1. A key feature of persistent homology is its stability; small changes in the input data lead to small changes in the associated barcode diagrams [11].

Our contribution is the introduction of a novel global descriptor STPP (Signature of Topologically Persistent Points) which is based on computing the persistence of the zeroth and first homology groups of a 3D point cloud. This signature is used to create a feature vector where the birth and death of the generators of homology correspond to the evolution of the number of connected components and the number of holes in the dataset. STPP can be computed quickly and efficiently on point cloud data, requires no preprocessing (sampling, hole filling, surface normal calculation, etc.) and uses a single tuning parameter. Moreover, it can cope with noisy datasets without compromising on performance.

This paper is structured as follows. Related work is described in Section II. Section III states the problem followed by our approach to computing a signature of topologically persistent points in Section IV. The experimental setup and results are presented and discussed in Section V. We conclude in Section VI.

II. Related Work

The problem of shape analysis has generated a great amount of research. Shape representation and description is an area of shape analysis with many important applications. In this section, we review relevant global descriptors for describing shapes composed of 3D point cloud data followed by developments in shape analysis using topological persistence that has led to this work.

Rusu et al. generalize the fast point feature histograms (FPFH) idea to create a descriptor that captures the relationship of local geometric parts in whole objects [12]. This descriptor, termed the global fast point feature histogram (GFPFH), is used for scene interpretation in mobile manipulation scenarios. Their work is followed by a 3D point cloud descriptor, called the viewpoint feature histogram (VFH) descriptor, which incorporates both geometry and viewpoint [13].

In [14], a multimodal perception system consisting of hierarchical object geometric categorization and appearance classification for personal robots introduces the global radius-based surface descriptor (GRSD). This descriptor is derived from the radius-based surface descriptor [15]. It can generalize over objects with similar geometry thus limiting the possibilities of corresponding an object instance to its 3D point cloud cluster.

The clustered viewpoint feature histogram (CVFH) descriptor, based on the VFH descriptor, is described by Aldoma et al. [16]. The main idea behind the semi-global CVFH descriptor is to take advantage of object parts obtained by a depth sensor and use them to build a coordinate system similar to the VFH descriptor. It improves upon issues that make VFH sensitive to missing point cloud data due to partial occlusions, segmentation, or sensor artifacts.

Wohlkinger and Vincze introduce an ensemble of shape functions (ESF) descriptor in [17]. This global shape descriptor is built on three distinct shape functions that describe distance, angle, and area distributions on the surface of a partial point cloud. ESF allows for real-time classification of objects sensed with an RGB-D sensor based on learning from synthetic CAD models.

Kanezaki et al. present the concatenated voxelized shape and color histograms descriptor, (Con)VOSCH [18]. It combines the GRSD and the circular color cubic higher-order local auto correlation descriptors (C^3-HLAC). The descriptor is designed to facilitate object classification by considering geometric and visual features in a unified manner.

The authors of [19] propose a global object recognition pipeline utilizing a 3D semi-global describer called oriented, unique, and repeatable CVFH (OUR-CVFH). OUR-CVFH employs a method to estimate semi-global unique reference frames (SGURF) computed on the surface of an object as seen from a single viewpoint. By exploiting the orientation provided by these reference frames, OUR-CVFH efficiently encodes the geometrical properties of an object surface.

A global covariance-based point cloud descriptor for object detection and recognition is put forward by Fehr et al. [20]. The descriptor is constructed by forming the covariance matrix from an RGB-D feature vector. The authors show that covariance descriptors are computationally fast and provide a compact (low dimensionality) representation of a 3D point cloud object.

In contrast to the aforementioned global descriptors, STPP encodes the *topological* information of a 3D point cloud. Our work is inspired by an early study of shape description and classification via persistent homology [21]. Additional inspiration comes from the results of Li et al. [22] where persistence diagrams built from functions defined on objects serve as compact and informative descriptors for images and shapes.

III. Problem Statement

Given a topological space $X = \{x_0, \ldots, x_{m-1}\} \in \mathbb{R}^3$ where $x_0, \ldots, x_{m-1}$ are the points in a point cloud acquired by an RGB-D sensor, our goal is to compute a topologically persistent description of the point cloud. To do this, we first approximate the topology of the space using a simplicial complex. Next, we compute the persistent homology of the complex up to a maximum scale value. Finally, we record the births and deaths of the generators of homology.

IV. Signature of Topologically Persistent Points

We formulate a solution to the problem outlined in III by creating a pipeline that: 1) Models the input space as a *Vietoris-Rips* complex; 2) Uses an incremental algorithm to compute the topological persistence of the zeroth and first generators of homology through a *filtration*; 3) Forms a feature vector comprised of the birth-death pairing of the homology generators. The end result is the capability to distinguish between noisy point cloud datasets, sensed by a depth sensor, using only topological features. In the proceeding subsections we describe the details of computing a topologically persistent signature for a 3D point cloud dataset. To do so requires a few definitions from algebraic topology.

A. Constructing a Simplicial Complex Representation

1) Simplices: A k-simplex σ is the convex hull of $k+1$ affinely independent vertices $v_0, \ldots, v_k \in \mathbb{R}^n$. We call k the dimension of a k-simplex. Note that a 0-simplex corresponds to a point, a 1-simplex is a finite line segment, and a 2-simplex is a triangle in $\mathbb{R}^n$. A face τ of σ is the convex hull formed by the subset $\{v_0, \ldots, v_k\}$ of the $k+1$ vertices. For example, the face of a line segment corresponds to its two end points.

2) Simplicial Complexes: A simplicial complex K is a non-empty set of simplices such that if $\sigma \in K$ and τ is a face of σ, then $\tau \in K$. In addition, the intersection of any two simplices $\sigma, \sigma' \in K$ is a face of both σ and σ'. We denote $|K|$ as the set of points in $\mathbb{R}^n$ that are contained in the union of all simplices in K. The set $|K|$ is a topological space with the subspace topology from $\mathbb{R}^n$. A subset of simplices $L \subset K$ that is itself a simplicial complex is called a subcomplex

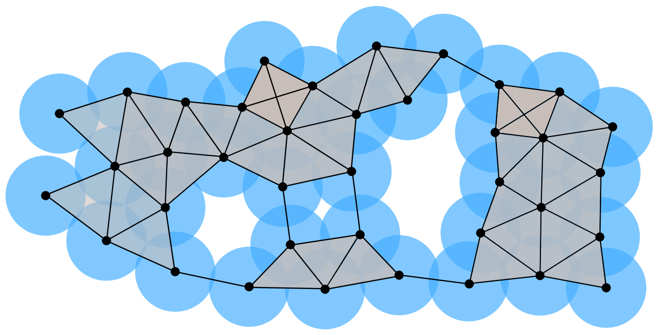

Fig. 2. A Vietoris-Rips complex generated on a point cloud consisting of 38 points.

of K. The conditions that define a simplicial complex ensure that the assembly of simplices is well-defined. A simplicial complex generalizes both the notion of a geometric graph and a triangulation.

3) Vietoris-Rips Complex: To compute the topological invariants of a point cloud, we need to make sure that the topology of the simplicial complex representation is close to the topology of the underlying space. We do this using a Vietoris-Rips complex, Fig. 2. Given a set of points $\{x_0, \ldots, x_{m-1}\} \in \mathbb{R}^n$ and fixed radius ϵ, the Vietoris-Rips complex of the topological space is defined as

$$K = \{\sigma \subset \{x_0, \ldots, x_{m-1}\} \mid \text{dist}(x_i, x_j) \leq \epsilon, \forall x_i \neq x_j \in \sigma\}.$$

In this definition, dist is the Euclidean distance metric and the points of σ are pairwise within ϵ of each other.

B. Persistent Homology Computation

1) Chains, Cycles, Boundaries: For a simplicial complex $K \in \mathbb{R}^3$, a k-chain is a subset of k-simplices in K. The addition of chains is defined with integer coefficients modulo 2. In other words, we sum two k-chains, c and d, by taking the symmetric difference of the two sets,

$$c + d = (c \cup d) - (c \cap d).$$

This operation is commutative. The group C_k is the set of all k-chains with the addition operator and the empty set defined as the zero element of C_k. For every integer k there is a chain group, however for a complex in $\mathbb{R}^3$ only $0 \leq k \leq 3$ are non-trivial.

The collection of $(k-1)$ dimensional faces forms the boundary, $\partial_k(\sigma)$, of a k-simplex σ and is a $(k-1)$-chain. By taking the sum of the boundaries of the simplices in a k-chain we have the boundary of the k-chain, $\partial_k(c) = \sum_{\sigma \in c} \partial_k(\sigma)$. A homomorphism, $\partial_k : C_k \rightarrow C_{k-1}$, is defined for each boundary operator. The collection of boundary operators on the chain groups form a chain complex,

$$\emptyset \rightarrow C_3 \xrightarrow{\partial_3} C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} \emptyset.$$

The kernel of ∂_k, ker $\partial_k = \{c \in C_k \mid \partial_k(c) = \emptyset\}$, is the set of k-chains with empty boundary. The image of ∂_k, img $\partial_k = \{d \in C_{k-1} \mid \exists c \in C_k : d = \partial_k(c)\}$, is the set of $(k-1)$-chains that are boundaries of k-chains. A k-chain in the kernel of ∂_k is a k-cycle and a $(k-1)$-chain in the image of ∂_k is a k-boundary. Together, the sets Z_k of k-cycles and B_k of k-boundaries with addition form subgroups of C_k. Note that the boundary of a vertex is the empty set and since K is a complex in $\mathbb{R}^3$ there are no non-empty 3-cycles or 3-boundaries, i.e. $Z_3 = B_3 = \emptyset$.

2) Homology Groups: The k-cycle group factored by the k-boundary group defines the kth homology group, $H_k = Z_k / B_k$. Elements of H_k are the homology classes $c + B_k = \{c + b \mid b \in B_k\}$, for all $c \in Z_k$. A subset of H_k generates a vector space if every element is the sum of elements in the subset. A minimal generating set creates a basis for H_k. The size of the basis corresponds to the rank of the homology group. We express the rank of the kth homology group as the kth Betti number of K, $\beta_k = \text{rank } H_k$, where

$$\text{rank } H_k = \text{rank } Z_k - \text{rank } B_k.$$

That is, β_k corresponds to the number of k-dimensional holes in a topological space. Although there exists a Betti number for each integer k, only ones for $0 \leq k \leq 2$ are non-zero for complexes in $\mathbb{R}^3$. It follows that for $k = 0$, β_0 is the number of connected components in K and for $k = 1$, β_1 is the number of holes in K.

3) Filtrations: To compute persistent homology, an increasing sequence of topological spaces known as a filtration is created. A filtration can be described as an evolution of the growth of a complex. It contains an ordering of the simplices in a subcomplex which we call a *filter*. Formally, if $f : X \rightarrow \mathbb{R}$ is defined on a topological space X, then each sublevel set $X_r = f^{-1}((-\infty, r])$ yields a topological space X_r where $X_r \subset X'_r$ and $r \leq r'$.

As the spatial range r increases, homological features are created (born) and can disappear (die). We work with a filtration of finite simplicial complexes in $\mathbb{R}^3$,

$$\emptyset \subset K_0 \subset K_1 \subset \ldots \subset K_l = K_\infty,$$

where l is the maximum threshold for constructing the complex.

C. Incremental Algorithm

1) Overview: Our approach to computing a topologically persistent signature of a 3D point cloud is based on the scheme described in [23]. It uses the ordering of the simplices in a filter to compute persistence incrementally. In this subsection, we present the essential background of the algorithm.

Suppose the sequence of σ_i, for $0 \leq i \leq l$, is a filter and let the sequence $K_i = \{\sigma_j \mid 0 \leq j \leq i\}$, for $0 \leq i \leq l$, be the corresponding filtration. We need to determine whether a $(k+1)$-simplex σ_i belongs to a $(k+1)$-cycle in K_i. When $k + 1 = 0$, this is trivial since every vertex belongs to a 0-cycle. For edges, we keep track of the connected components of the complex where each component is represented by its vertex set. An edge belongs to a 1-cycle if and only if its two endpoints belong to the same component.

After settling the cycle question the algorithm labels each simplex. A $(k+1)$-simplex σ_i is *positive* if it belongs to a $(k+1)$-cycle, otherwise it's labeled *negative*. Let $\beta_k = \beta_{k,i}$

be the kth Betti number of K_i, and let $\text{pos}_k = \text{pos}_{k,i}$ and $\text{neg}_k = \text{neg}_{k,i}$ be the number of positive and negative k-simplices in K_i. Then,

$$\beta_k = \text{pos}_k - \text{neg}_{k+1},$$

for $0 \leq k \leq 2$. In other words, the Betti number β_k is the number of k-simplices that create k-cycles minus the number of $(k+1)$-simplices that destroy k-cycles by creating k-boundaries. This pairing between positive k-simplices and negative $(k+1)$-simplices describes the persistence of non-bounding cycles in homology groups. Furthermore, it provides a signature that expresses the expansion of the filtration K_i.

2) Computation: To compute the persistence of 0-cycles we use a union-find data structure. For each edge, we perform two find operations on the end points to determine the point set representatives. A union operation is then performed based on the youngest (shortest-lived) 0-cycle representative.

Computing persistent 1-cycles is complicated by the fact that there can be multiple unpaired edges at any time. To compute the persistence of 1-cycles, we use a list of points, edges, and triangles created during the simplicial complex construction, and maintain a linear array $T[0, \ldots, s-1]$. The function of this array is similar to a hash table. For a pair of simplices (σ_i, σ_j), the index j of the matching negative simplex identified by the algorithm is stored in $T[i]$. Additionally, a set of positive 1-simplices Λ^i defining the cycle created by σ_i and destroyed by σ_j is also stored in $T[i]$. Each simplex in the filter has an entry in the array, but information is only stored in the entries of the positive simplices.

The search for a 1-cycle proceeds as follows. Assume that σ_j is a negative 2-simplex and suppose that the algorithm arrives arrives at $T[j]$. We search for the youngest 1-simplex, in the set of positive 1-simplices that represent the boundary of σ_j, by successively probing entries in T from right to left until we find the appropriate one. More specifically, starting with a set Λ equal to the positive boundary 1-simplices, we let $i = \max(\Lambda)$ be the index of the youngest member of Λ. If $T[i]$ is unoccupied, then we end the search and store Λ and j in $T[i]$. If $T[i]$ is occupied, then it contains a set Λ^i representing a permanently stored 1-cycle. Therefore, we add Λ and Λ^i by taking the symmetric difference of the two sets, and get a new Λ that represents a 1-cycle homologous to the old one. This procedure repeats until we find an empty slot in the array.

When searching for 1-cycles the incremental algorithm makes many redundant computations. To improve the performance of the algorithm, we introduce two procedures for speeding up the search for 1-cycles. These procedures, store-cycles and update-cycles, are shown in Alg. 1-3.

3) Runtime: With a union-find data structure the search for 0-cycles can be done in nearly constant time. By using weighted merging for union and path compression for find, the amortized time per operation is $\mathcal{O}(A^{-1}(n))$, where n is the number of 0-simplices and $A^{-1}(n)$ is the very slowly growing inverse of the Ackermann function. The cycle search

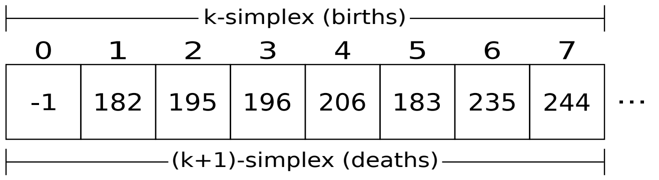

Fig. 3. The birth-death pairings of the homology generators are encoded in a feature vector where the indices of the vector correspond to the birth of a k-simplex and the entries of the vector correspond to the index of a destroying $(k+1)$-simplex. Unpaired simplices, denoted by -1 entries, represent k-dimensional holes.

approach introduced by Edelsbrunner et al. has a runtime that is cubic in the number of simplices. However, with our proposed procedures the algorithm operates in nearly linear time thus making the computation of persistent 1-cycles on large 3D datasets feasible.

Algorithm 1 COMPUTE-PERSISTENCE

```
1: for all $\sigma^j$ do
2:     $\Lambda = \{\sigma \in \partial_{k+1}(\sigma^j) \mid \sigma$ positive$\}$
3:     store-cycles($\Lambda$)
4: end for
```

D. Topologically Persistent Feature Vector

1) Overview: A barcode or persistence diagram encapsulates a concise description of the topological changes that occur during a filtration. Intuitively, a k-dimensional hole born at time b and filled at time d gives rise to a point (b, d) in the kth persistence diagram or an interval in the kth barcode diagram. Therefore, a persistence diagram is a multiset of points in $\mathbb{R}^2$ while a barcode diagram is an equivalent multiset of intervals in $\mathbb{R}$.

The use of distances between barcode/persistence diagrams has received much attention lately in applications [24]–[26] where they serve as topological proxies for the input data. Distances between the diagrams serve as measures of the similarity between datasets. These distances can be expressed as a bottleneck or Wasserstein distance between two planar point sets using the L_∞ metric.

2) Construction: Although the distance between barcode/persistence diagrams has been shown to measure the similarity between some datasets, we observe that a single scalar value is not enough to discriminate between massive 3D point clouds. In contrast to the work mentioned in the previous subsection, we construct a vector of topologically persistent features as follows.

The output of Alg. 1 is a sorted birth and death pairing between a k-simplex and $(k+1)$-simplex, respectively. We encode this information in a feature vector where the indices of the vector correspond to the birth of a k-simplex and the entries of the vector correspond to the index of a destroying $(k+1)$-simplex, Fig. 3. This pairing is unique, e.g. a 0-simplex is terminated by exactly one 1-simplex and a 1-simplex is terminated by exactly one 2-simplex. An unpaired simplex is a simplex of infinite persistence and represents a k-dimensional hole.

Algorithm 2 STORE-CYCLES

```
1:  $\Lambda = \text{sort}(\Lambda)$, $i = \max(\Lambda)$, $\Lambda^c = \{\emptyset\}$
2:  if $T[i]$ is unoccupied then
3:      for $n = 0$ to $|\Lambda| - 2$ do
4:          if $T[\Lambda(n)]$ is unoccupied then
5:              $\Lambda^c = \Lambda^c + \Lambda(n)$
6:          else if $T[\Lambda(n)]$.cycle is nonempty then
7:              $\Lambda^c = \Lambda^c + T[\Lambda(n)]$.cycle
8:          else
9:              continue
10:         end if
11:         $T[\Lambda(n)]$.update $\leftarrow i$
12:     end for
13:     $T[i]$.cycle $\leftarrow \Lambda^c$
14:     $T[i]$.death $\leftarrow j$
15:     $\sigma^j \leftarrow$ negative
16:     update-cycles($i, 0, T[i]$.update)
17: else
18:     for $n = 0$ to $|\Lambda| - 2$ do
19:         if $T[\Lambda(n)]$ is unoccupied then
20:             $\Lambda^c = \Lambda^c + \Lambda(n)$
21:         else if $T[\Lambda(n)]$.cycle is nonempty then
22:             $\Lambda^c = \Lambda^c + T[\Lambda(n)]$.cycle
23:         end if
24:     end for
25:     $\Lambda = \Lambda^c + T[i]$.cycle
26:     if $\Lambda$ is nonempty then
27:         $i = \max(\Lambda)$
28:         if $T[i]$ is unoccupied then
29:             for $n = 0$ to $|\Lambda| - 2$ do
30:                 $T[\Lambda(n)]$.update $\leftarrow i$
31:             end for
32:             $T[i]$.cycle $\leftarrow \Lambda - i$
33:             $T[i]$.death $\leftarrow j$
34:             $\sigma^j \leftarrow$ negative
35:             update-cycles($i, 0, T[i]$.update)
36:         end if
37:     end if
38: end if
```

Algorithm 3 UPDATE-CYCLES

```
1: if update is nonempty then
2:     for $n = 0$ to $|\text{update}| - 1$ do
3:         update-cycles($i$, update($n$), $T$[update($n$)].update)
4:     end for
5: end if
6: $T$[update($n$)].cycle $\leftarrow T$[update($n$)].cycle$+T[i]$.cycle$+i$
```

Fig. 4. A subset of objects from the RGB-D Object Dataset [27] used for the experiments.

V. Experimental Results

In this section, we construct and evaluate a processing pipeline for computing topological signatures as described in Section IV. The experiments were performed using the RGB-D Object Dataset [27] where we focus on the task of category classification. Construction of the simplicial complexes and computing topological persistence were done on a 64-bit GNU/Linux machine with a single CPU core.

1) Setup: The RGB-D Object Dataset consists of 300 objects divided into 51 categories and provides roughly 250,000 point clouds. Following the experimental procedure in [27], we subsample the dataset by taking every fifth point cloud. This gives us approximately 45,000 point clouds upon which we run classification experiments. To perform category recognition, we randomly leave one object out from each category for testing and train the classifiers on the point clouds of the remaining objects. Classification is performed using an SVM classifier and RBF kernel [28]. The accuracies are averaged over 10 trials.

2) Experiment 1: In this experiment, we compare STPP against five different global 3D point cloud descriptors: VFH, GRSD, CVFH, OUR-CVFH, and ESF. Implementations of each of these descriptors are publicly available in the Point Cloud Library (PCL)[1]. We compute STPP features using a single step filtration as follows.

The Vietoris-Rips complex representation of the data is computed by first performing a nearest neighbors search about each point up to a radius of 3.5 mm using a kd-tree. Next, we create an edge list where an edge exists between two neighboring points. We then proceed to create a triangle list by finding all three cliques in the edge graph. Once we have the edge and triangle lists we can compute the signature of the point cloud using the incremental algorithm (Alg. 1).

The last stage of the pipeline uses the output of the incremental persistence computation to form a feature vector. The indices of the vector range over the sorted 0-simplices and 1-simplices while the entries consist of the indices of the destroying 1-simplices and 2-simplices. We also construct feature vectors for each of the five global descriptors on the subsampled data. These feature vectors are then used to train separate classifiers for comparison as shown in Table I.

3) Experiment 2: This experiment considers the affect of exposing the STPP descriptor to different noise levels. To compare the effect of noise on the descriptor we randomly perturb all the points by $\delta \in [-0.0005, 0.0005]$, $\delta \in [-0.001, 0.001]$, and $\delta \in [-0.003, 0.003]$. This perturbation essentially deforms the surface of the point cloud. We then recompute the signatures and rerun the category classification experiments as described above.

[1]http://www.pointclouds.org

TABLE I
OBJECT CLASSIFICATION RESULTS

Descriptor	Category Accuracy	Number of Features
GRSD	8.89 $\pm$0.55	4
STPP	23.32 $\pm$1.58	2
VFH	23.61 $\pm$1.92	4
OUR-CVFH	26.91 $\pm$1.64	6
CVFH	29.39 $\pm$1.35	5
ESF	39.60 $\pm$1.01	6

4) Discussion: In these experiments we compare the performance of state of the art global geometrical descriptors with STPP, a global topological descriptor. To highlight the discriminative power of STPP in performing category classification we create separate feature vectors for the generators of homology: 0-cycles (Betti 0) and 1-cycles (Betti 1). The overall accuracy of the Betti 0 and Betti 1 feature vectors on the dataset is 14.16% $\pm$1.46 and 18.34% $\pm$1.48, respectively. Table I reports the overall accuracy of combining 0-cycles and 1-cycles (Betti $0+1$) as features. We see that combining the birth-death pairing of the homology generators increases the classification accuracy of the descriptor. We also note that in terms of the number of features used, STPP uses two features while other methods make use of four to six features. As for robustness to noise, the average accuracy of STPP is 22.57% for $\delta \in [-0.0005, 0.0005]$, 21.24% for $\delta \in [-0.001, 0.001]$, and 16.28% for $\delta \in [-0.003, 0.003]$.

VI. CONCLUSION

This paper presents STPP, a new 3D point cloud descriptor that uses persistent homology to compute a topological signature based on the birth-death pairing of 0-cycles and 1-cycles. To show the feasibility of STPP, we implemented a pipeline that computes and compares the topological signature of 3D point cloud objects against geometrical-based descriptors. We showed that the classification performance of STPP is competitive, it inherently deals with noisy datasets, and is theoretically well-founded.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation through grants #CNS-1338042, #IIS-1427014, #CNS-1439728, #CNS-1531330 and #CNS-1544887.

REFERENCES

[1] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, no. 2, pp. 255–308, 2009.

[2] J. Lamar-León, E. B. Garcia-Reyes, and R. Gonzalez-Diaz, "Human gait identification using persistent homology," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2012, pp. 244–251.

[3] J. L. Leon, R. Alonso, E. G. Reyes, and R. G. Diaz, "Topological features for monitoring human activities at distance," in *International Workshop on Activity Monitoring by Multiple Distributed Sensing*. Springer, 2014, pp. 40–51.

[4] Z. Zhou, Y. Huang, L. Wang, and T. Tan, "Exploring generalized shape analysis by topological representations," *Pattern Recognition Letters*, vol. 87, pp. 177–185, 2017.

[5] A. Asaad and S. Jassim, "Topological data analysis for image tampering detection," in *International Workshop on Digital Watermarking*. Springer, 2017, pp. 136–146.

[6] N. Singh, H. D. Couture, J. Marron, C. Perou, and M. Niethammer, "Topological descriptors of histology images," in *International Workshop on Machine Learning in Medical Imaging*. Springer, 2014, pp. 231–239.

[7] P. Skraba, M. Ovsjanikov, F. Chazal, and L. Guibas, "Persistence-based segmentation of deformable shapes," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, 2010, pp. 45–52.

[8] W. J. Beksi and N. Papanikolopoulos, "3D point cloud segmentation using topological persistence," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5046–5051.

[9] ——, "3D region segmentation using topological persistence," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[10] H. Edelsbrunner and J. Harer, *Computational topology: an introduction*. American Mathematical Soc., 2010.

[11] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Discrete & Computational Geometry*, vol. 37, no. 1, pp. 103–120, 2007.

[12] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and segmenting objects for mobile manipulation," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2009, pp. 47–54.

[13] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 2155–2162.

[14] Z.-C. Marton, D. Pangercic, R. B. Rusu, A. Holzbach, and M. Beetz, "Hierarchical object geometric categorization and appearance classification for mobile manipulation," in *IEEE-RAS International Conference on Humanoid Robots*, 2010, pp. 365–370.

[15] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D modelling of novel objects from a single view," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 3700–3705.

[16] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "Cad-model recognition and 6dof pose estimation using 3d cues," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 585–592.

[17] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 2987–2992.

[18] A. Kanezaki, Z.-C. Marton, D. Pangercic, T. Harada, Y. Kuniyoshi, and M. Beetz, "Voxelized shape and color histograms for rgb-d," in *IROS Workshop on Active Semantic Perception*, 2011.

[19] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6DOF pose estimation," in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*. Springer, 2012, pp. 113–122.

[20] D. Fehr, W. J. Beksi, D. Zermas, and N. Papanikolopoulos, "Covariance based point cloud descriptors for object detection and recognition," *Computer Vision and Image Understanding*, vol. 142, pp. 80–93, 2016.

[21] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas, "Persistence barcodes for shapes," *International Journal of Shape Modeling*, vol. 11, no. 02, pp. 149–187, 2005.

[22] C. Li, M. Ovsjanikov, and F. Chazal, "Persistence-based structural recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1995–2002.

[23] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," *Discrete and Computational Geometry*, vol. 28, no. 4, pp. 511–533, 2002.

[24] J. Gamble and G. Heo, "Exploring uses of persistent homology for statistical analysis of landmark-based shape data," *Journal of Multivariate Analysis*, vol. 101, no. 9, pp. 2184–2199, 2010.

[25] A. Adcock, D. Rubin, and G. Carlsson, "Classification of hepatic lesions using the matching metric," *Computer vision and image understanding*, vol. 121, pp. 36–42, 2014.

[26] C. Gu, L. Guibas, and M. Kerber, "Topology-driven trajectory synthesis with an example on retinal cell motions," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2014, pp. 326–339.

[27] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1817–1824.

[28] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.