Persistence-based Pooling for Shape Pose Recognition

Thomas Bonis[1], Maks Ovsjanikov[2], Steve Oudot[1], and Frédéric Chazal[1]

[1] Inria Saclay, DataShape team
[2] Laboratoire d'Informatique de l'Ecole Polytechnique

Abstract. In this paper, we propose a novel pooling approach for shape classification and recognition using the bag-of-words pipeline, based on topological persistence, a recent tool from Topological Data Analysis. Our technique extends the standard max-pooling, which summarizes the distribution of a visual feature with a single number, thereby losing any notion of spatiality. Instead, we propose to use topological persistence, and the derived persistence diagrams, to provide significantly more informative and spatially sensitive characterizations of the feature functions, which can lead to better recognition performance. Unfortunately, despite their conceptual appeal, persistence diagrams are difficult to handle, since they are not naturally represented as vectors in Euclidean space and even the standard metric, the bottleneck distance is not easy to compute. Furthermore, classical distances between diagrams, such as the bottleneck and Wasserstein distances, do not allow to build positive definite kernels that can be used for learning. To handle this issue, we provide a novel way to transform persistence diagrams into vectors, in which comparisons are trivial. Finally, we demonstrate the performance of our construction on the Non-Rigid 3D Human Models SHREC 2014 dataset, where we show that topological pooling can provide significant improvements over the standard pooling methods for the shape pose recognition within the bag-of-words pipeline.

Keywords: Shape recognition, Bag-of-Words, Topological Data Analysis

1 Introduction

In the recent years, databases of 3-dimensional objects have been getting larger and larger. In order to automatically process these databases, many algorithms relying on retrieval have been proposed. However, for certain tasks, classification techniques can be more efficient. Efficient classification pipelines have been proposed for images and some elements of these techniques such as the bag-of-words methods [1] or feature learning using deep network architectures [2] have been used to perform retrieval and shape comparison. Traditionally, the bag-of-words method relies on extracting an unordered collection of descriptors from the shapes we consider, which are then quantized into a set of vectors called "words". The information given by this quantization process is then summarized using a

pooling scheme, which produces a vector usable by standard learning algorithms. Ideally, all the steps of this framework should be robust to transformations of the shape: translations, rotations, changes of scale, etc. Modern bag-of-words approaches for 3D-shapes usually rely on a pooling method called sum-pooling [1] which consists in taking the average of the value of each words across the shape.

Since its introduction for image processing in [3], the bag-of-words pipeline, which we present in Section 4, has been improved in various ways. Here, we focus on the pooling part of the framework. Apart from the traditional sum-pooling approach, a popular pooling method, called *max-pooling* introduced in [4], consists in taking the maximum of the value for each visual word. Several works have highlighted the improvement in accuracy obtained using this pooling scheme as well as its compatibility with the linear kernel for learning purposes, [4, 5]. The strength of max pooling is due in part to its remarkable robustness properties. One of the main assumptions made in the bag-of-words approach is that the "word" values that compose the output of the encoding step, are, for a given class and a given word, i.i.d random variables. Refinements of the max-pooling scheme have been proposed under this assumption: for instance [6] proposed to consider the k highest values for each words to estimate the probability of at least k features being present in the object. However, the independence assumption of the word functions is unrealistic; for 3D shapes close vertices tend to have similar word functions, as illustrated in Figure 1. Thus, in this example, the generalization proposed by [6] ends up capturing the same feature multiple times and providing multiple redundant values. On the other hand, pooling on different parts of an image [7] and 3D shape [8, 9] has been proposed to take advantage of spatial information, an approach known as Spatial Pyramid Matching. This approach has drastically improved the performance of the bag-of-words procedures on multiple datasets, although it contradicts the identically distributed assumption, and lacks proper robustness guarantees.

In this work, we propose to see the word functions not as a unordered collection of random values but as a random function defined on the vertices of a graph (in our case, the mesh of the shape). Following this approach, we propose to use persistent homology to capture information regarding the global structure of the word functions which is not available for the traditional max-pooling approach.

Persistent homology was first introduced in the context of Topological Data Analysis under the name *size theory* [10]. It was later generalized to higher dimensions as *persistent homology* theory [11, 12]. The 0-dimensional persistent homology of the superlevel-sets of a function encodes the prominence of the peaks of the function into a collection of points in the plane, called a *persistence diagram*. These diagrams enjoy strong robustness properties [13–15]. One option to compare persistence diagrams is to use a distance between diagrams such as the bottleneck distance and to use nearest-neighbor algorithms as it was done by [16]. However, in this work, we aim at being able to use classification algorithms such as SVM or logistic regression that requires a Hilbert space structure, which

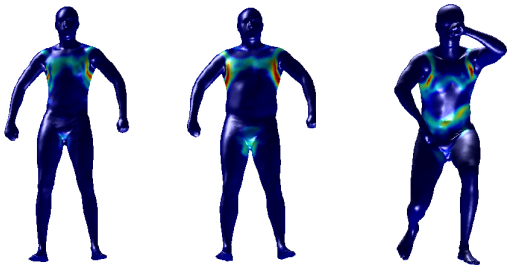

Fig. 1. Example of a word function obtained on two different shapes in the same pose and for the two different poses.

is not the case of the space of persistence diagrams. One approach to tackle this issue is to make use of the "kernel trick" by using a positive-definite kernel in order to map the persistence diagrams into a Hilbert space. As recently shown by Reininghaus et al. [17], one cannot rely on natural distances such as the Wasserstein distance to build traditional distance-based kernels such as the Gaussian kernel. This led the authors to propose another kind of kernel. A major limitation of their approach, however, is that these types of kernel are non-linear and the complexity of the classification becomes linear with the size of the training set which causes scalability issues. Another approach to directly embed persistence diagrams into a Hilbert Space was proposed in [18]. However this embedding is highly memory-consuming as it maps a single diagram into a set of functions and is not appropriate for dealing with large datasets.

In this work, we propose to perform pooling by computing the persistence diagrams of each word function. We then map these persistence diagrams into $\mathbb{R}^d$ for some reasonable value of d -< 20- by considering the peaks with highest prominence. Since we provide a direct mapping of persistence diagrams into $\mathbb{R}^d$, we can use it for the pooling stage for the bag-of-words procedure and achieve good performance with respect to the classification phase. We call this pooling approach Topological Pooling. Since it relies on persistence diagrams, this method is stable with respect to most transformations the shape can undergo: translations, rotations, etc., as long as the descriptors used in input are also invariant to these transformations. Moreover, we show that this pooling approach is robust to perturbations of the descriptors. Finally we demonstrate the validity of our approach compared to both sum-pooling and max-pooling by performing pose recognition on the SHREC 2014 dataset.

2 The bag of words pipeline

The bag-of-words pipeline consists of three main steps: feature extraction, coding and pooling. Here we describe each step briefly taking a functional point of view,

and we also introduce the notations we will need to define our new pooling method. We will assume that the input to the pipeline is a set of M 3D-shapes G_i represented as triangle meshes with vertices V_i.

Feature extraction aims at deriving a meaningful representation of the shape: the feature function denoted as $\mathcal{F}_i : V_i \to \mathbb{R}^N$. It is usually done by computing local descriptors (such as HKS [19], SIHKS [20], WKS [21], Shape-net features [2], etc.) on each vertex of the mesh.

The purpose of *coding* is to decompose the values of the $\mathcal{F}_i$ by projecting them on a set of points $W = (w_k)_{k\in[|1,K|]} \in \mathbb{R}^N$ called a *codebook*. This allows to replace each feature function by a family of functions $(C_{i,k} : V_i \to \mathbb{R})_{k\in[|1,K|]}$, called the *word functions*. In other words, for a coding procedure $Coding$ and codebook W, the C_i are defined through

$$C_{i,k}(V_i) = Coding(\mathcal{F}_i(V), W).$$

There exist various coding methods, such as Vector Quantization [22], Sparse Coding [4], Locally Constrained Linear Coding [23], Fisher Kernel [24] or Super-vector [25]. The codebook is usually computed using K-means but supervised codebook learning methods [23], [5] generally achieve better accuracy. In the Sparse Coding approach, the one we use in this paper, W and C are computed on the training set following

$$\min_{(C_{i,k})_{i,k},W} \sum_{i=1}^{M} \sum_{x\in V_i} \|(\mathcal{F}_i(x)) - \sum_{k=1^K} C_{i,k}((\mathcal{F}_i(x)))w_k\|_2^2 + \lambda\|C_i((\mathcal{F}_i(x)))\|_1,$$

with constraint $\|w_i\| \leq 1$ and regularization parameter λ. During the testing phase, the optimization is only performed on C with the codebook already computed.

The *pooling* step aims at summarizing properties of the family $(C_{i,k})_{i,k}$ and representing them through a compact vector $(\mathcal{P}_i)_{1\leq i\leq M}$ which can then be used in standard learning algorithms such as the SVM (Support Vector Machine). Usually, the pooling method depends on the coding scheme. For Vector Quantization, one traditionally uses sum-pooling:

$$\begin{aligned}\mathcal{P}_i &= (SumPool(C_{i,1}), ..., SumPool(C_{i,K})) \\ &= (\sum_{x\in V_i} C_{i,1}(\mathcal{F}_i(x)), ..., \sum_{x\in V_i} C_{i,K}(\mathcal{F}_i(x))).\end{aligned}$$

Max-pooling was introduced along the Sparse Coding scheme by Yang *et al.* in [4]. With this pooling technique, we summarize a function by its maximum:

$$\begin{aligned}\mathcal{P}_i &= (MaxPool(C_{i,1}), ..., MaxPool(C_{i,K})) \\ &= (\max_{x\in V} C_{i,1}(\mathcal{F}_i(x)), ..., \max_{x\in V} C_{i,K}(\mathcal{F}_i(x))).\end{aligned}$$

It is interesting to note that the max-pooling approach is more robust than the sum-pooling. Indeed, it is robust to usual transformations the shape can

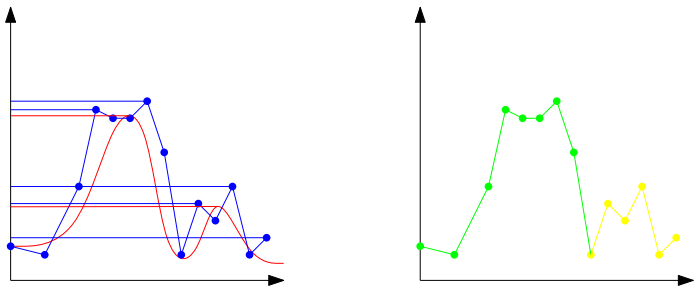

Fig. 2. A function f (red), a noisy approximation $\tilde{f}$ of f (blue) and their respective local maxima. Despite having a lot of local maxima, $\tilde{f}$ only has two "prominent peaks" (green and yellow).

undergo: translations, rotations, changes of scales, etc. However, it is still quite limited as it summarizes a whole function by a single value. A natural idea is to not limit ourselves to the global maximum of the function but rather to capture all local maxima. On the other hand, in this naive form, the method results in a very unstable pooling vector since arbitrarily small perturbations of the word functions can create many local maxima, as shown in Figure 2. Thus, a pooling approach consisting of taking the highest k local maxima is not stable. On the other hand, in the example shown in Figure 2, we can see that, while there are a lot of local maxima for the noisy function, both functions show only two "prominent peaks". These notions of "peak" and "prominence" are properly defined in the 0-dimensional persistent homology framework which provides us with tools to derive a robust pooling method.

3 Introducing 0-dimensional persistent homology

0-persistent homology provides a formal definition of prominence and measures the prominence of each peak of a function f, with the promise that the most prominent ones are stable under small perturbations of f. We provide a brief overview of the computation of 0-dimensional persistent homology for the superlevel-sets of a function defined on a graph, and invite the reader to consult [11] for a more general introduction.

Let f be a function defined on the vertices of a finite graph $G = (V, E)$. In 0-dimensional persistent homology, one focuses on the evolution of the connectivity of the subgraphs F_α of G induced by the superlevel-sets of f : $F_\alpha = (\{v \in V \mid f(v) \geq \alpha\}, \{(u, v) \in E \mid \min(f(u), f(v)) \geq \alpha\})$, as α decreases from $+\infty$ to $-\infty$, as shown in figure 3. A vertex v is a *local maximum* if, for any edge (v, u) in E, we have $f(u) \leq f(v)$. A peak p corresponds to a local maximum $b_p = f(v_p)$ of f. We say that p is *born* at b_p, see figure 3.(b). For a local maximum v_p, let $C(v_p, \alpha)$ be the connected component of v_p in F_α and let d_p be the largest value of α such that the maximum of f over $C(v_p, \alpha)$ is larger than b_p, we say that p *dies* at d_p. Intuitively, a peak dies when its connected component gets merged with the

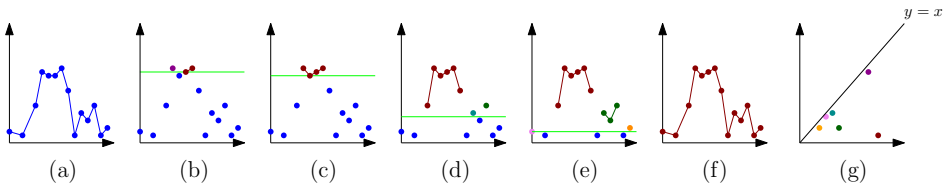

Fig. 3. Evolution of the connectedness of the superlevel-sets F_α of a function f in blue (a) as α (green) decreases from $+\infty$ to $-\infty$ (b-f). This evolution is then encoded in a persistence diagram (g).

one of another peak that has a higher maximum. Thus, there exists a vertex u_p which connects the two components such that $f(u_p) = d_p$. u_p is called a *saddle*, see figure 3.(c). The "prominence" of p is then the difference $b_p - d_p$. The peak corresponding to the global maximum of f dies when α reaches the minimum value of f on G[3]. Thus, a peak of f can be described by the couple (b_p, d_p). The set of such points (with multiplicity) in the plane is called a *persistence diagram*, denoted Δ_f, see figure 3.(g).

Persistence diagrams are endowed with a natural metric called the *bottleneck distance*. The definition of this metric involves the notion of partial matching. A partial matching M between two diagrams Δ_1 and Δ_2 is a subset of $\Delta_1 \times \Delta_2$ such that each point of Δ_1 and Δ_2 appears at most once in M. The bottleneck cost $C(M)$ of a partial matching M between two diagrams Δ_1 and Δ_2 is the infimum of $\delta \geq 0$ that satisfy the following conditions:

- For any $(p_1, p_2) \in M$, $||p_1 - p_2||_\infty \leq \delta$, and
- For any other point (b, d) of Δ_1 or Δ_2, $b - d \leq 2\delta$.

The bottleneck distance between two diagrams D_1 and D_2, is then defined as:

$$d_B(\Delta_1, \Delta_2) = \inf_\delta \{\delta \mid \exists M, C(M) \leq \delta\}$$

Intuitively, the bottleneck distance can be seen as the cost of a minimum perfect matching between persistence diagrams (with possibility to match points to the diagonal $y = x$), where the cost is the length of the longest line, see figure 4. A remarkable property of persistence diagrams, proven by [13] and [15], is their robustness with respect to perturbations of f. Given two functions f and g defined on some graph G, we have:

$$d_B(\Delta_f, \Delta_g) \leq ||f - g||_\infty = \sup_{v \in V} |f(v) - g(v)| \tag{1}$$

In other words, if we compare the diagrams of a function f and of a noisy version of a function $\tilde{f}$ then each point $p \in D_{\tilde{f}}$ can either be matched to a point of D_f or it has a low prominence, see figure 4.

[3] This point is slightly different from the traditional persistent homology framework. Usually, the death value of the peak corresponding to the global maximum is set to $-\infty$.

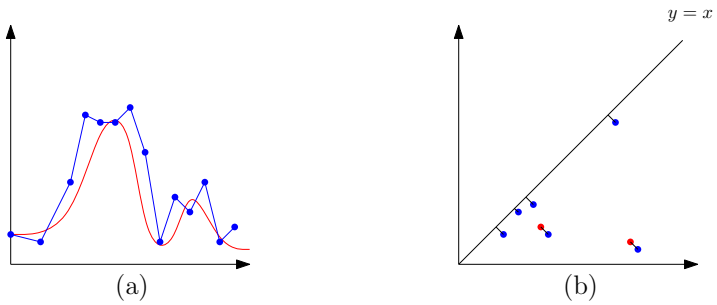

Fig. 4. (a): A real-valued function f (red) and a noisy approximation $\tilde{f}$ of f (blue).(b): Their respective persistence diagrams have close bottleneck distance.

Computation As 0-dimensional persistence encodes the evolution of the connectivity of the superlevel-sets of a function, computing it can be done using a simple variant of a Union-find algorithm; in practice we use Algorithm 1 described by Chazal et al. [26], with parameter τ set to infinity. This algorithm has close to linear complexity in the number of vertices of the meshes; more precisely it has complexity $O(|V|\log(|V|) + |V|\alpha(|V|))$ where α is the inverse of the Ackermann function.

4 Using persistence diagrams for pooling.

As we previously mentioned at the end of Section 4, a simple idea to enhance the max-pooling approach is to consider the values of multiple local maxima. However, this can be highly unstable under small perturbations of the word functions. As we saw in Section 3, we can use persistence diagrams to deal with this issue. Given a persistence diagram Δ, we define the *prominence* p of a point $(b, d) \in \Delta$ by $p = b - d$; in other words, the prominence corresponds to the lifespan of a peak during the computation of the persistence diagram. Given a function f on a graph G, we define the infinite-dimensional Topological Pooling vector of f with $i-th$ coordinate given by

$$TopoPool(f)_i = p_i(\Delta_f),$$

where $p_i(\Delta_f)$ is the i-th highest prominence of the points of Δ_f if there is at least i points in Δ_f and 0 otherwise. Since the stability of persistence diagrams given in Equation 1 implies the stability of the prominence of the points of Δ_f, such a construction yields some stability for our pooling scheme.

Proposition 1 *Let G be a graph and f and g two functions on a graph G with vertices V. Then, for any integer n, and any $0 < k < n$,*

$$|TopoPool(f)_k - TopoPool(g)_k| \leq 2 \sup_{x \in V} |f(x) - g(x)|$$

Of course, in practice we cannot use an infinite-dimensional vector and we simply consider a truncation of this vector keeping n first coordinates, we denote such a truncated pooling vector "TopoPool-n". Using the notations of Section , given some $n > 0$, the pooling vectors $(\mathcal{P}_i)_{1 \leq i \leq M}$ we consider are

$$\mathcal{P}_i = (TopoPool - n(C_{i,1}(\mathcal{F}_i(x))), ..., TopoPool - n(C_{i,K}(\mathcal{F}_i(x)))).$$

5 Experiments

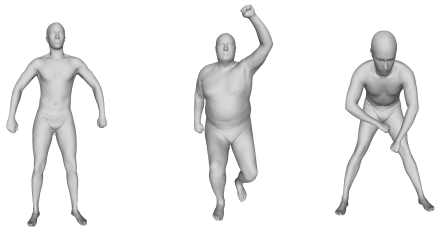

Fig. 5. The real SHREC 2014 dataset

In this section we evaluate the sum-pooling, the max-pooling and our topological pooling approaches on the SHREC 2014 dataset "Shape Retrieval of Non-Rigid 3D Human Models" [27], which we modify by applying a random rotation to each 3D shape. The dataset is composed of 400 meshes of 40 subjects taking 10 different poses and we wish to classify each of these meshes with respect to the pose taken by the subject. We consider both SIHKS features [20] and curvature-based features corresponding to the unary features from [28] and composed of 64 values corresponding to the curvatures, the Gaussian curvature, the mean curvature ... The coding step is performed using Sparse Coding [4] and the computation are performed performed using the SPAMS toolbox [29]. The learning part is done using a Support Vector Machine. We use 3 shapes per class for the training set, 2 for the validation set and 5 for the testing set. We compare the traditional sum-pooling with our TopoPool-n with different values for n -remark that $n = 1$ is equivalent to max-pooling- and under different codebook sizes. As a baseline, we also display the results obtained using a rigid Iterated Closest Point (ICP) [30] and a 1-nearest neighbour classification, which aims at iteratively minimizing the distance between two point clouds through rigid deformations. In our case it corresponds to finding the correct rotation to align the shapes as two shapes in a similar pose are close, however the approach can fail if it gets stuck in a local minimum and is not able to recover the correct rotation. We run the experiment a hundred times, selecting the training and testing sets at random. We display the mean accuracy over the multiple runs in Table 5.

Pooling / Codebook size	40	60	80	100	120	140	160	180	200
SIHKS features									
Sum-Pooling	0.53	0.56	0.60	0.60	0.58	0.62	0.61	0.60	0.60
TopoPool-1	0.46	0.55	0.53	0.54	0.58	0.59	0.63	0.64	0.64
TopoPool-5	0.69	0.71	0.69	0.70	0.73	0.70	0.74	0.73	0.72
TopoPool-10	0.70	0.71	0.71	0.69	0.72	0.71	0.73	0.74	0.72
TopoPool-15	0.72	0.73	0.71	0.70	0.74	0.71	0.74	**0.75**	0.71
TopoPool-20	0.72	0.73	0.70	0.72	0.73	0.72	0.73	0.75	0.73
Curvature features									
Sum-Pooling	0.80	0.80	0.84	0.85	0.88	0.88	0.87	0.88	0.89
TopoPool-1	0.39	0.56	0.56	0.57	0.64	0.69	0.69	0.73	0.76
TopoPool-5	0.63	0.79	0.80	0.80	0.82	0.85	0.86	0.87	0.86
TopoPool-10	0.74	0.85	0.85	0.86	0.86	0.87	0.89	0.89	0.88
TopoPool-15	0.78	0.85	0.87	0.87	0.88	0.89	0.89	**0.90**	0.90
TopoPool-20	0.79	0.88	0.88	0.88	0.88	0.89	0.90	0.90	0.89
ICP	0.55								

Table 1. Mean accuracy obtained on the SHREC 2014 dataset.

The first noticeable fact about our experiments being the overall better results obtained by our Topological Pooling scheme compared to the max-pooling and to the sum-pooling for the SIHKS features. In the case of curvature features, Topological Pooling and sum-pooling gives similar accuracy results for large codebooks but in the case of smaller codebooks, Topological pooling gives much better results. It is interesting to notice that the gap between the different pooling scheme decreases as the size of the codebook increases. Indeed, the smaller the codebook, the richer each word function in terms of topology -and thus the richer each persistence diagrams will be-.

Regarding the running time of our experiment in the case of SIHKS features, online testing using the bag-of-words procedure with the largest codebook to a given shape takes around 40 seconds, where most of the time is devoted to computing the SIHKS. On the other hand, performing ICP between two shapes takes 6 seconds, thus the online testing time for a single shape with ICP is 6 times the cardinality of our training set seconds; in our case 5 minutes. On the other hand, with the ICP approach requires no offline training while the bag of words requires to compute the codebook, perform the whole bag-of-words pipeline on each training shape and compute the SVM which takes roughly 45 minutes. Overall we have to classify 350 shapes, the bag-of-words approach requires 4 hour and a half while the ICP approach requires more than a day.

6 Conclusion

In this paper, we proposed to use the canonical graph structure on shapes to capture neighborhood information between the different feature vectors. We then built discrete "word functions" on this graph instead of following the traditional approach of considering a collection of independent "word" vectors. We then

proposed to consider new pooling features making use of this new information and generalizing the classical max-pooling approach by using the critical points of the "word functions". We proposed to use 0-dimensional persistent homology to ensure stability of a pooling output relying on these features. Finally, we designed a new pooling method relying on these new features and we experimentally showed that these features are efficient in a pooling context.

Acknowledgements. This work was supported by ANR project TopData ANR-13-BS01-0008. First author was supported by the French Délégation Générale de l'Armement (DGA). Second author was supported by Marie-Curie CIG-334283-HRGP, a CNRS chaire dexcellence, a chaire Jean Marjoulet from Ecole Polytechnique, and a Faculty Award from Google Inc.

References

1. A. M. Bronstein, M. M. Bronstein, L. J. Guibas, M. Ovsjanikov, Shape google: Geometric words and expressions for invariant shape retrieval, ACM Trans. Graph. 30 (2011) 1–20.
2. J. Masci, D. Boscaini, M. M. Bronstein, P. Vandergheynst, Shapenet: Convolutional neural networks on non-euclidean manifolds.
URL http://arxiv.org/abs/1501.06297
3. F.-F. Li, P. Perona, in: In CVPR, 2005, pp. 524–531.
4. J. Yang, K. Yu, Y. Gong, T. Huang, Linear spatial pyramid matching using sparse coding for image classification, in: in IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2009.
5. Y.-L. Boureau, F. Bach, Y. LeCun, J. Ponce, Learning mid-level features for recognition, in: In Proc. CVPR, 2010.
6. L. Liu, L. Wang, X. Liu, In defense of soft-assignment coding, ICCV '11, 2011, pp. 2486–2493.
7. S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2, CVPR '06, 2006, pp. 2169–2178.
8. R. J. Lpez-Sastre, A. Garca-Fuertes, C. Redondo-Cabrera, F. J. Acevedo-Rodrguez, S. Maldonado-Bascn, Evaluating 3d spatial pyramids for classifying 3d shapes., Computers and Graphics 37 (2013) 473–483.
9. C. Li, A. B. Hamza, Intrinsic spatial pyramid matching for deformable 3d shape retrieval, IJMIR 2 (2013) 261–271.
10. A. Verri, C. Uras, P. Frosini, M. Ferri, On the use of size functions for shape analysis, Biological Cybernetics 70 (1993) 99–107.
11. H. Edelsbrunner, J. Harer, Computational Topology - an Introduction., American Mathematical Society, 2010.
12. A. Zomorodian, G. Carlsson, Computing persistent homology, Discrete Comput. Geom 33 (2005) 249–274.
13. D. Cohen-Steiner, H. Edelsbrunner, J. Harer, Stability of persistence diagrams, in: Proc. 21st ACM Sympos. Comput. Geom., 2005, pp. 263–271.
14. F. Chazal, D. Cohen-Steiner, L. J. Guibas, M. Glisse, S. Y. Oudot, Proximity of persistence modules and their diagrams, in: Proc. 25th ACM Sympos. Comput. Geom., 2009.

15. F. Chazal, V. de Silva, M. Glisse, S. Oudot, The structure and stability of persistence modules (2012).
URL http://arxiv.org/abs/1207.3674
16. C. Li, M. Ovsjanikov, F. Chazal, Persistence-based structural recognition, in: CVPR, 2014, pp. 2003–2010.
17. J. Reininghaus, S. Huber, U. Bauer, R. Kwitt, A Stable Multi-Scale Kernel for Topological Machine Learning, in: CVPR, 2015.
18. P. Bubenik, Statistical topology using persistence landscapes, JMLR 16 (2015) 77–102.
19. J. Sun, M. Ovsjanikov, L. Guibas, A concise and provably informative multi-scale signature based on heat diffusion, in: Proceedings of the Symposium on Geometry Processing, SGP '09, 2009, pp. 1383–1392.
20. M. M. Bronstein, I. Kokkinos, Scale-invariant heat kernel signatures for non-rigid shape recognition, in: In Proc. CVPR, 2010.
21. H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Comput. Vis. Image Underst. 110 (2008) 346–359.
22. G. Salton, M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, Inc., New York, NY, USA, 1986.
23. J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, in: in IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2010.
24. F. Perronnin, J. Sánchez, T. Mensink, Improving the fisher kernel for large-scale image classification, in: ECCV, 2010, pp. 143–156.
25. X. Zhou, K. Yu, T. Zhang, T. S. Huang, Image classification using super-vector coding of local image descriptors, in: ECCV, 2010.
26. F. Chazal, L. J. Guibas, S. Y. Oudot, P. Skraba, Persistence-based clustering in riemannian manifolds., J. ACM 60 (2013) 41.
27. D. e. a. Pickup, SHREC'14 track: Shape retrieval of non-rigid 3d human models, EG 3DOR'14, 2014.
28. E. Kalogerakis, A. Hertzmann, K. Singh, Learning 3D Mesh Segmentation and Labeling, ACM Transactions on Graphics 29.
29. J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding, J. Mach. Learn. Res. 11 (2010) 19–60.
30. P. J. Besl, N. D. McKay, A method for registration of 3-d shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 239–256.