MULTI-SCALE APPROACHES FOR THE MATCHING DISTANCE ESTIMATION

A. CERRI, B. DI FABIO, G. JABŁOŃSKI, AND F. MEDRI

Abstract. This paper deals with the concepts of persistence diagram and matching distance. These are two of the main ingredients of *Topological Persistence*, which has proven to be a promising framework for shape comparison. Persistence diagrams are descriptors providing a signature of the shapes under study, while the matching distance is a metric to compare them. One drawback in the application of these tools is the computational cost for the evaluation of the matching distance. The aim of the present paper is to introduce a new framework for the approximation of the matching distance, which does not affect the reliability of the entire approach in comparing shapes, and extremely reduces the computational cost. This is shown through experiments on 3D-models.

1. Introduction

Interpreting and comparing shapes are challenging issues in computer vision, computer graphics and pattern recognition [18, 20]. Topological Persistence – including Persistent Homology [13] and Size Theory [3, 14] – has proven to be a successful comparison/retrieval/classification (hereafter CRC) scheme.

In a nutshell, the basic idea for dealing with the CRC task is to define a measure of the (dis)similarity between the shapes in a given database. This can be done by extracting a battery of shape descriptors – the so-called *persistence diagrams* – from each element in the database, capturing meaningful shape properties. Thus, the problem of assessing the (dis)similarity between two shapes can be recast into the one of comparing the associated persistence diagrams according to the *matching* (or *bottleneck*) *distance*, a proven stable distance between these descriptors. This process defines a metric over the database, that can be used for CRC purposes. In general, a given persistence diagram may come from different shapes. This can be interpreted as an equivalence with respect to the properties captured by that descriptor.

Such an approach has been successfully used in a number of concrete problems concerning shape comparison and retrieval [7, 9, 12]. However, defining a (dis)similarity metric in the case of large databases can lead to a considerable computational cost. The bottleneck in this procedure can be identified in the evaluation of the matching distance.

The contribution of this paper. Reducing the computational costs in defining a (dis)similarity metric within a database of shapes is definitely a desirable target. This would enable us to further improve the persistence CRC framework and apply it to a wider class of concrete problems.

Key words and phrases. Persistence diagram, shape analysis, dissimilarity criterion.

In [8], the authors introduce a multi-scale strategy for the approximation of a matching distance-based (dis)similarity metric. For each pair of persistence diagrams associated to different elements in a database, the idea is to compute a rough estimation of their matching distance, which is faster to obtain than its exact computation. This estimation is based on a "dissimilarity criterion" that will be explained in detail in Section 3. It has been implemented using an algorithmic procedure which allows for a progressive refinement of such an estimation, whenever it is not sufficient to distinguish between persistence diagrams (and hence shapes) which are too similar.

The aim of the present paper is to continue that work, by extending it in two main respects:

- From the algorithmic viewpoint, we introduce a new scheme generalizing the one proposed in [8]. It is based on a randomized strategy to apply the aforementioned dissimilarity criterion. The outcome is a more flexible tool allowing us to obtain statistically better performances in terms of classification results.
- As for experiments, we enlarged their setting by considering a new dataset of triangle meshes and new batteries of persistence diagrams, obtaining even in this case satisfactory results.

The present work is organized as follows. In Section 2 we recall the necessary definitions and results needed in the rest of the paper. In Section 3 we describe the multi-scale construction of a matching distance-based (dis)similarity metric presented in [8], together with the dissimilarity criterion which is formalized in Theorem 3.1. Section 4 and Section 5 are devoted to present the schema we have developed for the application of the dissimilarity criterion. The experimental setting is described in Section 6, ranging from the chosen datasets to the selected batteries of persistence diagrams. In Section 7 experimental results are provided. A final discussion about the obtained results concludes the paper (Section 8).

2. Background notions

In the classical formulation of *persistence* [13], the shape of an object is usually studied by choosing a topological space X, and a function $\varphi : X \to \mathbb{R}$, called a *filtering (or measuring) function*. The role of φ is to describe the properties considered relevant to analyze the shape of the object represented by X [7, 14]. By considering the sublevel sets induced on X by the variation of φ, we can define a family of subspaces $X_u = \varphi^{-1}((-\infty, u])$, $u \in \mathbb{R}$, nested by inclusion, i.e. a filtration of X. Focusing on the occurrence of important topological events along this filtration, such as the appearance and disappearance of connected components, tunnels and voids, it is possible to obtain a global description of the considered shape. This information can be encoded in an algebraic structure known in the literature as *ordinary persistent homology groups* and defined as follows. Given $u \leq v \in \mathbb{R}$, we consider the inclusion of X_u into X_v. This inclusion induces a homomorphism of homology groups $H_k(X_u) \to H_k(X_v)$ for every $k \in \mathbb{Z}$. Its image consists of the k-homology classes that live at least from $H_k(X_u)$ to $H_k(X_v)$ and is called the *kth persistent homology group of (X, φ) at (u, v)*. If X satisfies some mild conditions [6] (which will be assumed to hold throughout the paper) this group is finitely generated: In this case, we denote its rank by $\beta_k^{u,v}(X, \varphi)$.

Persistence diagrams. A simple and compact description of the persistent homology groups of (X, φ) is provided by the corresponding persistence diagrams. These are multi-sets of points lying in the half-plane $\overline{\Delta^+} = \{(u, v) \in \mathbb{R}\times\mathbb{R} : u \leq v\}$. For each point, the u-coordinate represents the *birth* (in terms of the values of the filtering function) of a topological feature, whereas the v-coordinate represents its *death*. The distance of a point from the diagonal $\Delta = \{(u, v) \in \mathbb{R}\times\mathbb{R} : u = v\}$ represents the *lifespan* of the associated topological feature. Having this interpretation in mind, we can rank topological features with bounded lifespan by importance, according to the length of their life. The basic assumption here is that the longer a feature survives, the more meaningful or coarse the feature is for shape description. Vice-versa, noise and shape details are characterized by a shorter life. A persistence diagram can be formally defined via the notion of *multiplicity* [13, 15]. In what follows, the symbol Δ^+ denotes the set $\{(u, v) \in \mathbb{R} \times \mathbb{R} : u < v\}$.

Definition 2.1 (Multiplicity)**.** Let $k \in \mathbb{Z}$ and $(u, v) \in \Delta^+$. The *multiplicity* $\mu_k(u, v)$ of (u, v) is the finite non-negative number defined by

$$\lim_{\varepsilon\to 0^+} \left(\beta_k^{u+\varepsilon,v-\varepsilon}(X, \varphi) - \beta_k^{u-\varepsilon,v-\varepsilon}(X, \varphi) - \beta_k^{u+\varepsilon,v+\varepsilon}(X, \varphi) + \beta_k^{u-\varepsilon,v+\varepsilon}(X, \varphi)\right).$$

Definition 2.2 (Persistence Diagram)**.** The persistence diagram $D_k(X, \varphi)$ is the multiset of all points $(u, v) \in \Delta^+$ such that $\mu_k(u, v) > 0$, counted with their multiplicity, union the points of Δ, counted with infinite multiplicity.

We will call *proper points* the points of a persistence diagram lying on Δ^+.

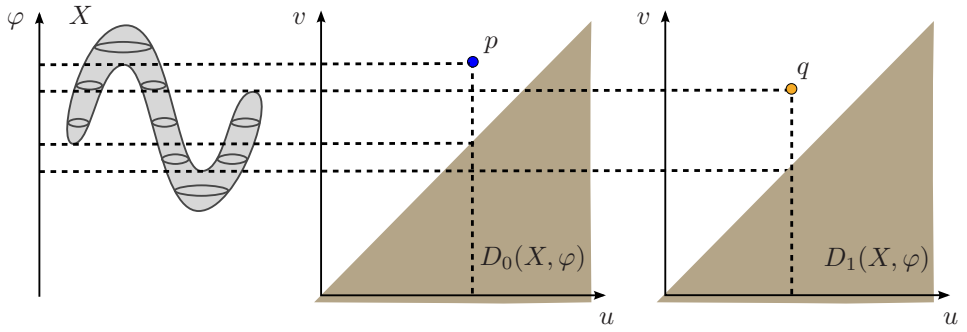

FIGURE 1. (*a*) The height function φ on the space X, and the associated persistence diagrams $D_0(X, \varphi)$ and $D_1(X, \varphi)$.

Figure 1 shows an example of persistence diagrams for $k = 0, 1$. The surface $X \subset \mathbb{R}^3$ is filtered by the height function φ. $D_0(X, \varphi)$ has only one proper point as well as $D_1(X, \varphi)$. The abscissa of p (q, respectively) corresponds to the level at which a new connected component (tunnel, respectively) is born along the filtration, while its ordinate identifies the level at which this connected component (tunnel, respectively) merges with the existing one (is closed on one side, respectively). To see, for instance, that $\mu_0(p) = 1$, letting $p = (\bar{u}, \bar{v})$, it is sufficient to observe that, for every $\varepsilon > 0$ sufficiently small, it holds that $\beta_0^{\bar{u}+\varepsilon,\bar{v}-\varepsilon}(X, \varphi) = 2$, $\beta_0^{\bar{u}-\varepsilon,\bar{v}-\varepsilon}(X, \varphi) = \beta_0^{\bar{u}+\varepsilon,\bar{v}+\varepsilon}(X, \varphi) = \beta_0^{\bar{u}-\varepsilon,\bar{v}+\varepsilon}(X, \varphi) = 1$, and apply Definition 2.1. In a analogous way, it can be observed also that $\mu_1(q) = 1$.

Matching distance. Two persistence diagrams can be compared by means of the matching distance, which measures the cost of finding a correspondence between their points. In doing this, the cost of taking a point p to a point p' is measured as the minimum between the cost of moving one point onto the other and the cost of moving both points onto the diagonal. In particular, the matching of a proper point p with a point of Δ can be interpreted as the destruction of the point p. See Figure 2 for an example.

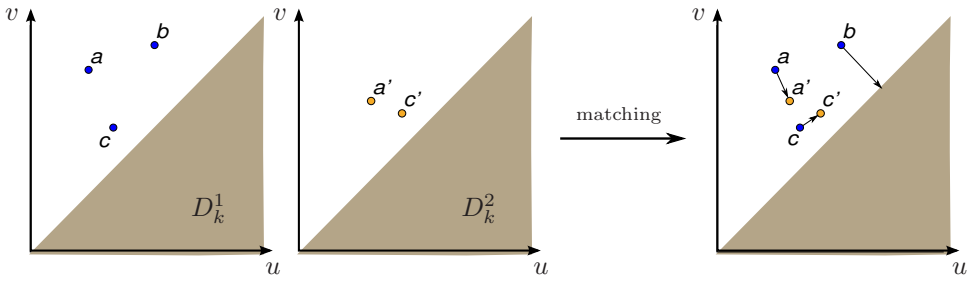

FIGURE 2. The matching between D_k^1 and D_k^2 realizing their matching distance.

Definition 2.3 (Matching Distance)**.** Let D_k^1, D_k^2 be two persistence diagrams. The matching distance $d_{match}\left(D_k^1, D_k^2\right)$ is defined as

$$d_{match}(D_k^1, D_k^2) = \min_{\sigma} \max_{p \in D_k^1} d(p, \sigma(p)),$$

where σ varies among all the bijections between D_k^1 and D_k^2 and

$$d\left((u,v),(u',v')\right) = \min\left\{\max\left\{|u-u'|, |v-v'|\right\}, \max\left\{\frac{v-u}{2}, \frac{v'-u'}{2}\right\}\right\} \tag{1}$$

for every $(u,v),(u',v') \in \overline{\Delta^+}$.

The main interest in this metric is due to the fact that persistence diagrams are robust with respect to the matching distance. A visual interpretation of this property is given in Figure 3, where the 0th persistence diagram of a woman surface model filtered by the height function is considered (a) together with a noisy version of it (b). In both diagrams, the two points which are farthest from the diagonal represent the components born once the filtration includes the woman's hands (they do not touch the rest of the body) and dying at the height of the armpits.

Looking at Figure 3, the stability of persistence diagrams with respect to the matching distance can be explained as follows: Small changes in the considered filtering function produce only small changes in the position of points far from the diagonal, and possibly produce variations close to the diagonal.

More formally, let us fix a homology degree, and consider two filtering functions $\varphi, \psi : X \to \mathbb{R}$. If we measure the distance between φ and ψ by the L_∞-norm, and the one between the corresponding persistence diagrams $D_k(X,\varphi)$ and $D_k(X,\psi)$ by the matching distance, the stability result bounds the latter distance by the former, i.e. $d_{match}(D_k(X,\varphi), D_k(X,\psi)) \leq \|\varphi - \psi\|_\infty$ [11, 13].

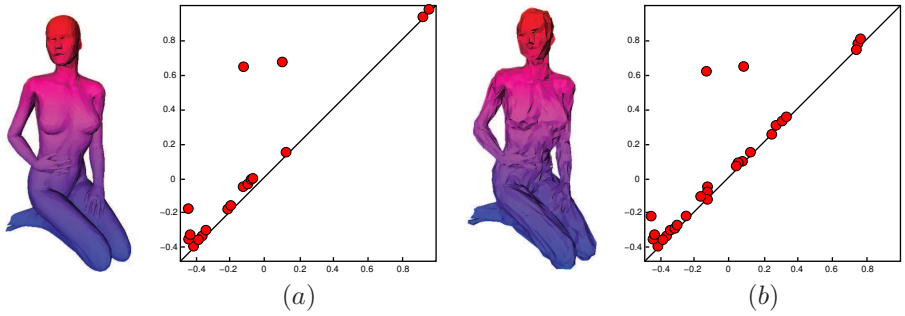

FIGURE 3. A woman surface model filtered by the height function and its $0th$-persistence diagram (a). A noisy version of the filtered surface model and its $0th$-persistence diagram (b). The surface model is part of the non-Rigid Shape Benchmark [4].

3. A dissimilarity criterion

Computationally, the evaluation of the matching distance between two persistence diagrams takes $O(h^{2.5})$ [10], being h the total amount of their proper points.

As stressed before, in CRC applications involving large databases, computing the matching distance for any possible shape comparison can imply a high computational cost. In fact, noisy or detailed shape models can produce persistence diagrams with a large number of proper points. Our goal is to reduce this computational complexity by considering, at first, only a rough estimation of the metric induced by the matching distance over a database, to be possibly refined whenever it is not sufficient to distinguish between different shapes.

The key point here is the observation that, in most cases, realizing that two shapes are very dissimilar does not require to compute the *exact* matching distance between the associated persistence diagrams. Deciding, e.g., whether an elephant is different from an ant requires only a first glance at the two animals. In our framework, such a "first glance" could be equivalent to a rough estimation of the matching distance – and hence faster than its exact computation – between the persistence diagrams associated with the "elephant shape" and the "ant shape", respectively. On the contrary, a different level of accuracy could be necessary to distinguish, e.g., the "wolf shape" from the "German shepherd shape". This would lead to a sharper estimation of the matching distance between the associated persistence diagrams, possibly to its actual computation.

In light of these considerations, we propose a multi-scale construction of our matching distance-based (dis)similarity metric.

Let D_k be a persistence diagram. For every $p = (u, v) \in \Delta^+$ and every $\delta > 0$, let $\mathcal{Q}_\delta(p)$ be the open square centered at p of side equal to 2δ, and let us denote by $\sharp(\mathcal{Q}_\delta(p), D_k)$ the number of points of D_k contained in $\mathcal{Q}_\delta(p)$. These notation will be maintained throughout the paper.

Theorem 3.1 (Dissimilarity criterion)**.** *Let D_k^1, D_k^2 be two persistence diagrams for which a point $p = (u, v) \in \Delta^+$ and two real numbers $\delta, \varepsilon > 0$ exist, such that $\mathcal{Q}_{\delta+\varepsilon}(p) \subset \Delta^+$ and $\sharp(\mathcal{Q}_\delta(p), D_k^1) - \sharp(\mathcal{Q}_{\delta+\varepsilon}(p), D_k^2) > 0$. Then $d_{match}(D_k^1, D_k^2) \geq \varepsilon$.*

Proof. The assumption $\sharp(\mathcal{Q}_\delta(p), D_k^1) > \sharp(\mathcal{Q}_{\delta+\varepsilon}(p), D_k^2)$ implies that, for every bijection $\sigma : D_k^1 \to D_k^2$ there exists at least one proper point $\bar{q} = (\bar{u}, \bar{v}) \in D_k^1$ such

that $\bar{q} \in \mathcal{Q}_\delta(p)$ and $\sigma(\bar{q}) = \bar{q}' = (\bar{u}', \bar{v}') \in D_k^2$, with $\bar{q}' \notin \mathcal{Q}_{\delta+\varepsilon}(p)$. Then, from (1) it holds that

$$d(\bar{q}, \bar{q}') \geq \min\left\{\varepsilon, \max\left\{\frac{\bar{v}-\bar{u}}{2}, \frac{\bar{v}'-\bar{u}'}{2}\right\}\right\} \geq \min\left\{\varepsilon, \frac{\bar{v}-\bar{u}}{2}\right\} = \varepsilon. \tag{2}$$

Indeed, in (2), the first inequality holds because both $|\bar{u} - \bar{u}'|$ and $|\bar{v} - \bar{v}'|$ are not smaller than the difference between the semi-sides of $\mathcal{Q}_\delta(p)$ and $\mathcal{Q}_{\delta+\varepsilon}(p)$; the second inequality is obvious; the equality follows from both the facts that $\bar{v} - \bar{u} > (v-\delta) - (u+\delta)$, being $(\bar{u}, \bar{v}) \in \mathcal{Q}_\delta(p)$ and $(u+\delta, v-\delta) \in \Delta^+$ the bottom right vertex of $\mathcal{Q}_\delta(p)$, and $(v-\delta-\varepsilon)-(u+\delta+\varepsilon) \geq 0$, i.e. $(v-\delta)-(u+\delta) \geq 2\varepsilon$, being $(u+\delta+\varepsilon, v-\delta-\varepsilon) \in \overline{\Delta^+}$ the bottom right vertex of $\mathcal{Q}_{\delta+\varepsilon}(p)$. Hence $\max_{q \in D_k^1} d(q, \sigma(q)) \geq \varepsilon$ for every bijection σ and, by Definition 2.3, the claim is proved. □

Figure 4 shows an example of Theorem 3.1 in action. Figures 4 $(a)-(b)$ represent two persistence diagrams, say D_k^1 and D_k^2, respectively. In Figure 4 (c) the two multisets of points are overlapped, and the two squares $\mathcal{Q}_\delta(p)$ and $\mathcal{Q}_{\delta+\varepsilon}(p)$ are depicted. As can be seen, it holds that $\sharp(\mathcal{Q}_\delta(p), D_k^1) - \sharp(\mathcal{Q}_{\delta+\varepsilon}(p), D_k^2) = 1$. Hence, by Theorem 3.1 we get that surely $d_{match}(D_k^1, D_k^2) \geq \varepsilon$.

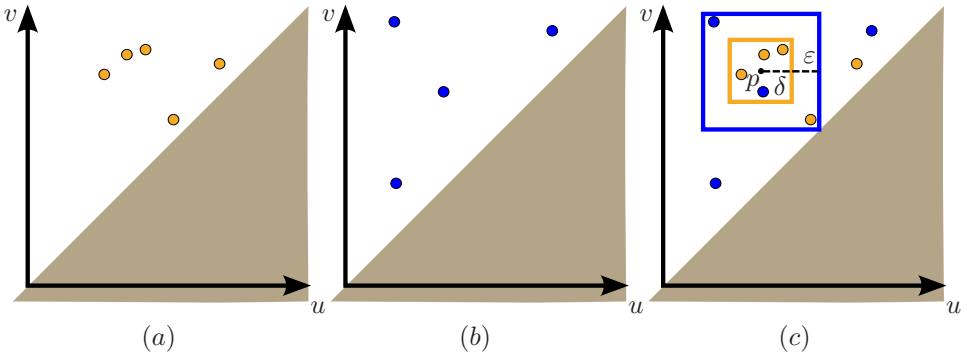

FIGURE 4. $(a)-(b)$ Two persistence diagrams D_k^1 and D_k^2. (c) The overlapping of D_k^1 and D_k^2, and the two squares $\mathcal{Q}_\delta(p)$ and $\mathcal{Q}_{\delta+\varepsilon}(p)$ for a certain $p \in \Delta^+$.

The issue here is to find a suitable way to apply Theorem 3.1, so to improve our CRC framework. To this end, two different schemes have been designed, which we describe in Section 4 and 5, respectively.

We conclude the section with a remark which will be useful later.

Remark 3.2. Definition 2.3 implies that $d_{match}(D_k^1, D_k^2) \leq (V-U)/2$, with $U = \min_{(u,v)\in L} u$, $V = \max_{(u,v)\in L} v$ and $L = D_k^1 \cup D_k^2$. Indeed, $(V-U)/2$ upper bounds the cost of the bijection between D_k^1 and D_k^2, taking all the points of L onto Δ. Since d_{match} is realized by the cheapest bijection between D_k^1 and D_k^2, we have the claim.

4. A Refinement Prefixed Grid Scheme

This section is devoted to review the scheme introduced in [8] to estimate from the bottom the matching distance between two persistence diagrams by virtue of Theorem 3.1. Moreover, we describe the pseudo-code and the computational

complexity of each algorithm. We called this procedure a Refinement Prefixed Grid Scheme (RPGS).

An implementation of RPGS is Algorithm 1, which takes as input the lists A and B of proper points of two persistence diagrams, and a parameter N which is a natural number. It runs a number of iterations equal to N. Starting from a prefixed grid as in Figure 5(a), during each iteration, a finer and finer grid is created on a triangular region $\mathrm{T} \subset \overline{\Delta^+}$ with vertices $(U-\omega, U-\omega), (U-\omega, V+\omega), (V+\omega, V+\omega)$, containing all the points belonging to A and B, being U and V as in Remark 3.2, and ω an arbitrarily small positive real number. At each iteration n, the algorithm produces $n(n+1)/2$ small squares with side equal to $(n+5)th$ part of the side of T. It then evaluates Theorem 3.1 on each small square compared with the square having its same center and side three times greater. The algorithm returns the maximum value for which Theorem 3.1 holds. Algorithm 1 makes use of two different subroutines. The first one is $Matrix(i,j)$, which simply generates a two dimensional matrix $0_{i\times j}$. The second one is $CountPoints(S,p,q)$ (Algorithm 2), whose output is the sum of the entries of the 3×3 submatrix $S[p-1,p,p+1;q-1,q,q+1]$, that is, the number of points of the largest square into which we are going to evaluate the theorem. An example of RPGS in action is shown in Figure 5.

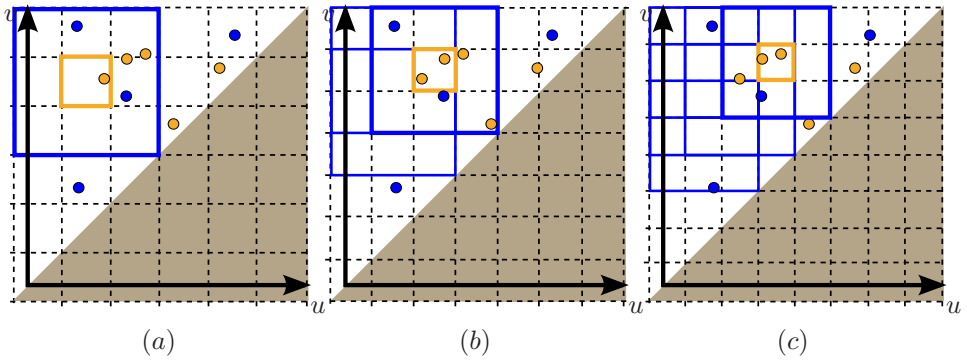

FIGURE 5. RPGS in action: three steps $(a)-(c)$ are necessary to find squares in which Theorem 3.1 holds.

Computational complexity. Set $h = |A| + |B|$. The computational complexity C of RPGS can be formalized as

$$\mathrm{C}(h,N) = c_1 + \sum_{n=1}^{N}\left(c_2 + 2c_3(n+5)^2 + c_4\cdot h + \sum_{p=2}^{n+4}\sum_{q=p+3}^{n+4} c_5\right),$$

with $c_4\cdot h$ the cost of lines $9-16$, $c_3(n+5)^2$ the cost of lines $7-8$, c_3 and c_4 being constants as well as c_1 (lines $1-3$), c_2 (lines $5-6$) and c_5 (lines $19-29$), in Algorithm 1.

Making some simple mathematical manipulations we obtain that

$$\mathrm{C}(h,N) = c_1 + N(c_2 + c_4\cdot h) + 2c_3\cdot\sum_{n=1}^{N}(n+5)^2 + \sum_{n=1}^{N}\sum_{p=1}^{n+3}\sum_{q=1}^{n-p+1} c_5.$$

Algorithm 1 RPGS(A, B, N)

```
Res ⇐ 0
ω ⇐ (V − U)/10
Side ⇐ V − U + 2ω
for n = 1 to N do
    t ⇐ 5 + n
    sSide ⇐ Side/t
    qA ⇐ Matrix(t, t)
    qB ⇐ Matrix(t, t)
    for all a ∈ A do
        (i, j) ⇐ ⌈(a − U + ω)/sSide⌉
        qA(i, j) ⇐ qA(i, j) + 1
    end for
    for all b ∈ B do
        (i, j) ⇐ ⌈(b − U + ω)/sSide⌉
        qB(i, j) ⇐ qB(i, j) + 1
    end for
    for p = 2 to (t − 1) do
        for q = (p + 3) to (t − 1) do
            QA ⇐ CountPoints(qA, p, q)
            QB ⇐ CountPoints(qB, p, q)
            r1 ⇐ (QA < qB(p, q))
            r2 ⇐ (QB < qA(p, q))
            if (r1 or r2) and (Res < sSide) then
                Res ⇐ sSide
            end if
        end for
    end for
end for
return Res
```

Algorithm 2 CountPoints(S, p, q)

```
for i = (p − 1) to (p + 1) do
    for j = (q − 1) to (q + 1) do
        Res ⇐ Res + S(i, j)
    end for
end for
return Result
```

Now, by counting the total number of squares on which the theorem is evaluated on a run of the algorithm, which is

$$\sum_{n=1}^{N}\sum_{p=1}^{n+3}\sum_{q=1}^{n-p+1} 1 = \sum_{n=1}^{N}\sum_{p=1}^{n+3}(n-p+1) = \sum_{n=1}^{N}\frac{n(n+1)}{2} = \frac{N^3+3N^2+2N}{6},$$

we can conclude that the computational complexity of RPGS is $O(N^3)$.

5. A Randomized Bisection Squares Scheme

In this section we are going to elaborate on a more generic procedure for the application of Theorem 3.1 to approximate the matching distance between two persistence diagrams. Starting from the most general formulation, we will analyze the main building blocks, showing how to configure them to obtain different schemes of computation.

General Scheme. Algorithm 3 shows the skeleton of the most general scheme.

Algorithm 3 GeneralScheme(A, B, R, S, IQS, CIR)

```
1: r ⇐ r_1 ⇐ r_2 ⇐ 0
2: H ⇐ A ∪ B
3: Q ⇐ IQS(H)
4: for i = 1 to R do
5:     p ⇐ (CIR(x_min(H), x_max(H)), CIR(y_min(H), y_max(H)))
6:     d ⇐ p_y − p_x
7:     if d > c ∗ (V − U) then
8:         δ ⇐ CIR(0, d)
9:         for i = 1 to S do
10:            η ⇐ CIR(0, δ)
11:            if #(Q_η(p), A) − #(Q_δ(p), B) > 0 then
12:                r_1 ⇐ δ − η
13:            end if
14:            if #(Q_η(p), B) − #(Q_δ(p), A) > 0 then
15:                r_2 ⇐ δ − η
16:            end if
17:            r ⇐ max(r, r_1, r_2)
18:        end for
19:    end if
20: end for
21: return r
```

The overall idea can be sketched as follows. We first choose some points inside the triangular area T defined as in Section 4 (first **for** loop, Line 5). For each of these points p, we build a first square, say K^p, centered at p (first **if-then** construct, Line 8). Then, other squares, say $\{L^p_j\}$, are taken concentrically inside each K^p (second **for** loop, Line 10). Finally, we evaluate Theorem 3.1 on each possible pair (K^p, L^p_j) (Lines 11-15). Before going on, let us first disambiguate the pseudo-code and explain the followings key points:

- Line 1 initializes return values of Theorem 3.1 applications. In particular r will store the best result while r_1 and r_2 will memorize the temporary values related to the two possible applications;
- Line 3 initializes any structure which is necessary to make membership tests ($IQS(H)$ stands for *"Initialize Query Structure on the set H"*), and in particular to count how many proper points of a given set (A or B) belong to a given squared area;
- The value c appearing in Line 7, which represents the discrete step we use in our measurement, is a scalar in the interval $]0, m[\subset \mathbb{R}$, with m arbitrarily small, while the whole expression stands for *"Take p only if it is over the main diagonal by at least c times $V - U$"*;

- Finally, the expression $CIR(m, M)$ (that is, *"Choose In Range m..M"*) in Lines 5, 8 and 10 denotes the choice of a value inside the interval $[m, M] \subset \mathbb{R}$.

Computational complexity. From a computational point of view, Algoritm 3 generates R "outer" squares, each one containing S smaller, "inner" squares, and tries to evaluate at each step if Theorem 3.1 holds, keeping the maximum value of the estimate. It is already possible to identify the main parameters involved:

- $\mathbf{A}, \mathbf{B} \Leftarrow$ lists of proper points ($H = A \cup B$ and $h = |H|$);
- $\mathbf{R} \Leftarrow$ the number of outer squares;
- $\mathbf{S} \Leftarrow$ the number of inner squares;
- $\mathbf{CIR} \Leftarrow$ the way we choose the side length and the center of each square;
- $\mathcal{Q} \Leftarrow$ the way we make membership tests.

Having introduced the above parameters, we can give a first formulation of the overall computational complexity $C(h, R, S)$, which appears to be:

$$C(h, R, S) = I_{\mathcal{Q}}(h) + E_{out}(R) \cdot E_{int}(S) \cdot E_{\mathcal{Q}}(h), \tag{3}$$

with $I_{\mathcal{Q}}(h)$ the cost of building a structure to make membership tests, $E_{out}(R)$ the cost of evaluating R outer squares, $E_{int}(S)$ the cost of evaluating S inner squares, and $E_{\mathcal{Q}}(h)$ the cost of executing a query on such a structure.

In the following subsections we will discuss in detail each computational cost appearing in (3).

$\mathbf{E_{int}}$(S): Choosing the best pair - A bisection scheme. The first block we are going to explain is the one which realizes the search of the pair of squares (inner and outer) optimizing the estimate of the matching distance. Let us start with the following remark:

Remark 5.1. Assume that the side 2δ and position p of the outer square is fixed. If Theorem 3.1 holds for this square compared with some inner square, then there is only one optimal side's measure of the inner square, such that ε (semi-difference of square's sides) is maximal and $\#(\mathcal{Q}_{\delta-\varepsilon}(p), A) - \#(\mathcal{Q}_{\delta}(p), B) > 0$ (Theorem 3.1 holds). This is a straightforward consequence of the monotonicity of $\#(\mathcal{Q}_{\delta-\varepsilon}(p), A) - \#(\mathcal{Q}_{\delta}(p), B)$ with respect to ε.

We can exploit information from Remark 5.1 to formulate an algorithm for finding the optimal measure for an inner square's side with a fixed outer square. One can use the bisection method [17] to find the maximal difference in the length of sides of the squares up to a given error τ. Such τ value would depend on S with regard to the number of steps of the algorithm, and on a real interval to calculate the effective value. Without further constraints, the best choice of τ would be $\tau = \frac{\delta-\eta}{S}$.

The choice of a bisection scheme to find the optimal pair of outer/inner squares provides an improvement in the overall computational complexity from $O(S)$ down to $O(\log_2 S)$ (which means that we are evaluating $O(S)$ smaller squares in $O(\log_2 S)$ steps). The pseudo-code related to this part can be found in Algorithm 4.

It is interesting to observe that this method can be easily transposed to the opposite situation, when the inner square is fixed and the bigger one may vary, which is what Algorithm 5 achieves. Finally, we can provide a scheme which does not degrade the overall complexity and use both these approaches, constant outer square first, then constant inner square, and this is Algorithm 6.

Algorithm 4 BisectionInnerSquare$(p, \delta, \eta, \tau, A, B)$

$b \Leftarrow \eta, e \Leftarrow \delta$
while $e - b > \tau$ **do**
  $\gamma \Leftarrow (b + e)/2$
  if $\#(\mathcal{Q}_\gamma(p), A) - \#(\mathcal{Q}_\delta(p), B) > 0$ **then**
    $e \Leftarrow \gamma$
  else
    $b \Leftarrow \gamma$
  end if
end while
return $\delta - \gamma$

Algorithm 5 BisectionOuterSquare$(p, \delta, \eta, \tau, A, B)$

$b \Leftarrow \eta, e \Leftarrow \delta$
while $e - b > \tau$ **do**
  $\gamma \Leftarrow (b + e)/2$
  if $\#(\mathcal{Q}_\eta(p), A) - \#(\mathcal{Q}_\gamma(p), B) > 0$ **then**
    $b \Leftarrow \gamma$
  else
    $e \Leftarrow \gamma$
  end if
end while
return $\gamma - \eta$

Algorithm 6 BisectionSquare$(p, \delta, \eta, \tau, A, B)$

$d_m \Leftarrow d_M \Leftarrow 0$
if $\#(\mathcal{Q}_\delta(p), A) - \#(\mathcal{Q}_\delta(p), B) > 0$ **then**
  $d_m \Leftarrow$ BisectionInnerSquare$(p, \delta, \eta, \tau, A, B)$
end if
if $\#(\mathcal{Q}_\eta(p), A) - \#(\mathcal{Q}_\eta(p), B) > 0$ **then**
  $d_M \Leftarrow$ BisectionOuterSquare$(p, \delta, \eta, \tau, A, B)$
end if
$d \Leftarrow \max(d_m, d_M)$
Return d

The introduction of the bisection method in the general scheme can be carried out simply by replacing lines 8-18 with the code:

$\delta \Leftarrow CIR(0, d)$
$\eta \Leftarrow CIR(0, \delta)$
$\tau \Leftarrow \frac{\delta - \eta}{S}$
$r_1 \Leftarrow$ BisectionSquare$(p, \delta, \eta, \tau, A, B)$
$r_2 \Leftarrow$ BisectionSquare$(p, \delta, \eta, \tau, B, A)$
$r \Leftarrow \max(r, r_1, r_2)$

The new part is responsible for choosing sides of squares, as well as calling the bisection method. In this case, in the overall computational complexity 3, we have $E_{int} = O(\log_2(S))$.

$\mathbf{E_{out}(R)}$: Choosing outer squares - A random scheme. The procedure to choose outer squares is quite different from the bisection scheme we have just presented. Indeed, the only constraint we have to satisfy in choosing outer squares is that their bottom-right vertex must be contained in the triangular region T. This requirement directly contributes to the computational complexity, which cannot be better than linear in the number of outer squares we want to consider, meaning that $E_{out}(R) = O(R)$.

We propose in Algorithm 7 the Randomized Bisection Square Scheme (RBSS) to choose outer squares. Such choice is determined by selecting the squares' centers (line 5 in the **for** loop). We observe that a first optimization of our scheme can be obtained by taking only squares containing at least one proper point from $A \cup B$: We will show in Section 7 how we achieve this.

The optimal estimates for inner and outer squares are computed separately as explained in the previous subsection. In Lines 8-9 we use the procedure $Random(b, e)$ which returns random numbers from the interval $[b, e]$. For a point $p = (p_x, p_y)$, the measure of inner squares' sides centered at p is set randomly up to 30% of the maximal possible length, which is $d = p_y - p_x$. The measure of the outer squares' sides is set also randomly to at least 70% of d.

Algorithm 7 RBSS(A, B, R, IQS)

1: $r \Leftarrow r_1 \Leftarrow r_2 \Leftarrow 0$
2: $H \Leftarrow A \cup B$
3: $\mathcal{Q} \Leftarrow IQS(H)$
4: **for** $i = 1$ **to** R **do**
5: $\quad p \Leftarrow (\text{Random}(x_{\min}(H), x_{\max}(H)), \text{Random}(y_{\min}(H), y_{\max}(H)))$
6: $\quad d \Leftarrow p_y - p_x$
7: $\quad$ **if** $d > c * (V - U)$ **then**
8: $\quad\quad \delta \Leftarrow d \cdot \text{Random}(0.7, 1)$
9: $\quad\quad \eta \Leftarrow d \cdot \text{Random}(0, 0.3)$
10: $\quad\quad \tau \Leftarrow \frac{\delta - \eta}{S}$
11: $\quad\quad r_1 \Leftarrow \text{BisectionSquare}(p, \delta, \eta, \tau, A, B)$
12: $\quad\quad r_2 \Leftarrow \text{BisectionSquare}(p, \delta, \eta, \tau, B, A)$
13: $\quad\quad r \Leftarrow \max(r, r_1, r_2)$
14: $\quad$ **end if**
15: **end for**
16: **return** r

$\mathbf{I_{\mathcal{Q}}(h), E_{\mathcal{Q}}(h)}$: Making membership tests - kD-Trees. We need a fast method to make a query about number of points in a given square of Δ^+. Using kD-trees [1] is probably one of the best possible choices, as this allows for fast retrieval of points in a given square. Moreover, it is linear in terms of memory usage (with respect to the number of points) and the kD-tree construction does not increase the overall computational complexity, being $O(y \log_2 y)$, with y the cardinality of the set on which the kD-Tree is built. In our context two kD-trees are needed, one for points of A and one for points of B. Therefore, by using the kD-tree structure in Algorithm 7 we can update the overall complexity 3 as

$$C(h, S, R) = O(h \log_2 h) + O(R) \cdot O(\log_2(S)) \cdot O(h),$$

with $O(H)$ the cost of executing a query on a kD-tree.

$\mathbf{I}_{\mathcal{Q}}(\mathbf{h}), \mathbf{E}_{\mathcal{Q}}(\mathbf{h})$: Making membership tests - Pre-Computed number of points. Instead of using kD-trees, we could take into account pre-computed arrays to retrieve number of points in squares. Such an approach requires that the edges of the considered squares lie on a known grid. A further restriction is that vertices have integer coordinates, so we can use a two dimensional array. Under these assumptions, the number of points in the square with opposite vertices in (a, b) and $(a + k, b + k)$, respectively, is

$$\#points = SUM(a + k, b + k) - SUM(a + k, b) - SUM(a, b + k) + SUM(a, b),$$

where $SUM(u, v)$ stores the number of points in the rectangle of $\mathbb{R}^2$ having one vertex in $(0, 0)$ and the opposite one in (u, v). See Figure 6 for a visual interpretation of this relation.

Let us now assume we have an array SUM with entries $SUM(i, j)$, $i, j \in \mathbb{Z}$. In order to compute each entry in SUM, we divide the triangular region T by the smallest possible grid. The number of the grid cells is set to λ^2. It is then possible to count the number of points in the considered squares using Algorithm 8.

Algorithm 8 Sum(A, λ)

1: $\tau \Leftarrow \frac{V-U}{\lambda}$
2: **for** every $a \in A$ **do**
3: $(i, j) \Leftarrow \lceil \frac{a-U}{\tau} \rceil$
4: $SUM(i, j) \Leftarrow SUM(i, j) + 1$
5: **end for**
6: **for** $i = 1$ to λ **do**
7: **for** $j = 1$ to λ **do**
8: $SUM(i, j) \Leftarrow SUM(i, j) + SUM(i - 1, j) + SUM(i, j - 1) - SUM(i - 1, j - 1)$
9: **end for**
10: **end for**

First we count the number of points in the cells of the grid. Then we proceed row by row starting from the bottom, and in every row we go from the left to the right. For every entry $SUM(i, j)$ we add the number of points in the grid cell and the number of points in the rectangles which have vertices just below and to the left. Finally, we subtract the number of points in the rectangle with vertex $(i - 1, j - 1)$ (line 8).

Using this approach the computational complexity of RBSS turns out to be

$$C(\lambda, R) = O(\lambda^2) + O(R) \cdot O(\log_2(\lambda)) \cdot O(1),$$

with $O(1)$ the cost of executing a query on Algorithm 8. We remark that, when evaluating the computational complexity, the parameter λ has an effect on the range of the error τ. More precisely, by choosing the Sum scheme, the best possible choice of τ is $\tau = \frac{V-U}{\lambda}$.

Comparison with the matching distance computational cost. We conclude this section with some final remarks on the computational complexity of Algorithm 7, by comparing the usage of kD-trees with the one of the Sum procedure (Algorithm 8). We start from kD-trees. In this case, the computational complexity $C(h, S, R)$ has been evaluated as

$$C(h, S, R) = O(h \log_2 h) + O(R) \cdot O(\log_2(S)) \cdot O(h).$$

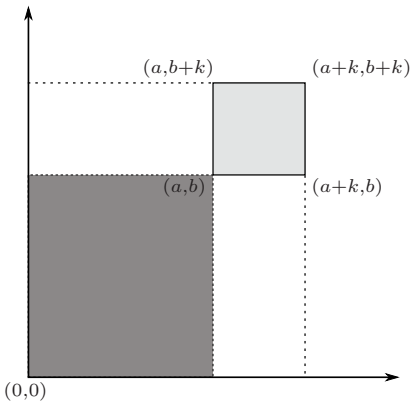

FIGURE 6. Computing number of points in the square.

It is not grater than the cost of evaluating the matching distance between two persistence diagrams only if $O(R) \cdot O(log_2(S)) \cdot O(h) \leq O(h^{2.5})$, or equivalently

$$O(R) \cdot O(log_2(S)) \leq O(h^{1.5}).$$

From the above relation we can deduce that, in order to keep $C(h, S, R)$ lower than the computational complexity of the matching distance, once we fix S, R can range between 1 and $h^{1.5}$, while, fixing R, S can range between 1 and $2^{h^{1.5}}$.

We can now analyze the usage of the Sum procedure. In this case, the RBSS computational complexity has been evaluated as

$$C(\lambda, R) = O(\lambda^2) + O(R) \cdot O(\log_2(\lambda)). \tag{4}$$

Hence, in order to have a computational complexity asimptotically lower than that of the matching distance, the following inequalities shall be verified:

$$\begin{cases} O(\lambda^2) \leq O(h^{2.5}) \\ O(R) \cdot O(\log_2(\lambda)) \leq O(h^{2.5}) \end{cases} \tag{5}$$

These reasonings lead us to conclude that optimal values for λ would be the in the range between 2 and $h^{1.25}$, while for R between $\frac{h^{2.5}}{\log_2 h^{1.25}}$ and $h^{2.5}$.

The previous considerations about S, λ and R are summarized in Table 1, assuming that the minimum possible value for R to be taken is 1.

SUM	λ from 2 to $h^{1.25}$	R from 1 to $h^{2.5}$
kD-tree	S from 2 to $2^{h^{1.5}}$	R from 1 to $h^{1.5}$

TABLE 1. Admissible values for the parameters S, R and λ.

To conclude we observe that, as in the case of RGPS, we can ensure that the computational complexity of RBSS is smaller than the complexity of classical computation of matching distance. We can do this simply by forcing the parameter R

to be smaller than either $h^{1.5}$ or $h^{2.5}$, according to the usage of kD-trees and the SUM scheme. Depending on the data, varying the ratio between S and R in the former case, as well as the ratio between λ and R in the latter one, would give a more accurate estimate for the matching distance computation.

6. Experimental setting

Our goal is to validate the theoretical framework introduced in the previous Sections 4 and 5. Through some experiments on persistence diagrams for 0th homology degree (a.k.a. *formal series* [14]), associated with 3D-models represented by triangle meshes, we will prove that our algorithms allow us to reduce the computational complexity in defining a matching distance-based metric over a given database, without greatly affecting the goodness of results (in terms of database classification).

The datasets. To test the proposed framework we considered two datasets of 3D-surface mesh models. As a first one, we opted for the Non-Rigid World Benchmark [4] (from now on Db1). This database contains 148 three-dimensional models, such as, e.g., cats, dogs, wolves, horses, lions and gorillas, in a variety of poses for non-rigid, shape similarity experiments. Figure 7 shows five models belonging to the "cat" class (first row), together with some representatives for other classes in the database (second row). The second database used in our experiments is the one

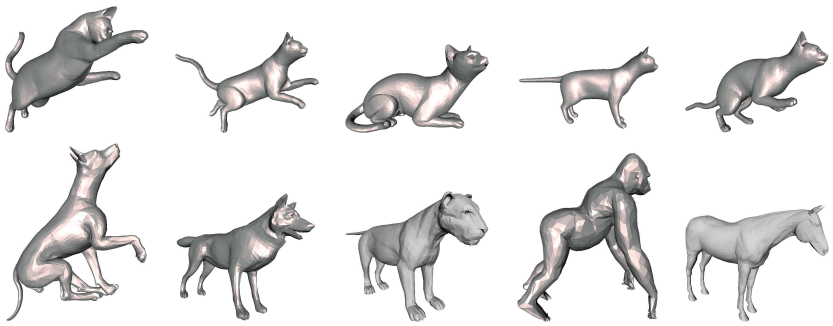

Figure 7. Models from the "cat" class (first row), and some representatives from other classes of the Non-Rigid Shape Benchmark.

introduced in [2], consisting of 228 3D-surface mesh models. This database (from now on Db2) is divided into 12 classes, each one containing 19 elements obtained as follows: A null model (cat0, david0, dog0,..., victoria0, wolf0) taken from the Non-Rigid World Benchmark is considered together with six non-rigid, possibly non-metric-preserving deformations applied to it at three different strength levels. An example of the transformations and their strength levels is given in Figure 8.

The filtering functions. To select the considered filtering functions we followed [2]. In that paper, the authors exploit the modularity of the persistence framework to study under different perspectives the models belonging to the two datasets. Indeed, persistence diagrams inherit their invariance properties (with respect to groups of transformations) directly from the corresponding filtering functions. Therefore, to obtain different invariance properties, it is sufficient to change the filtering

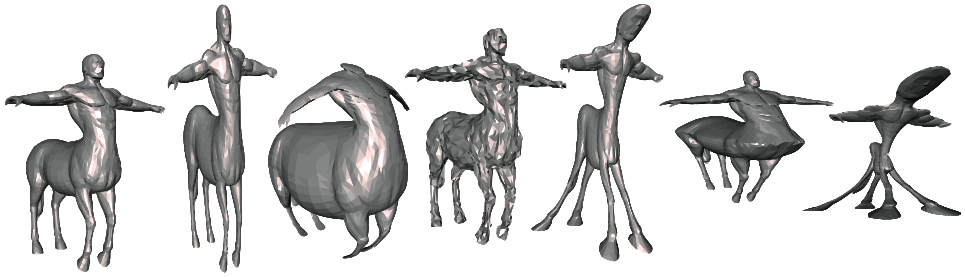

FIGURE 8. The null model "Centaur0" and the 3^{rd} strength level for each deformation.

function. In particular, the ones chosen in [2] fit the different purposes the two datasets are designed for: Db1 is suited to analyze non-rigid shape similarity, while Db2 has been created to deal with noise and other deformations which do not preserve the metric properties of shapes (e.g. the Riemannian metric).

According to the previous reasonings, for Db1 we considered two filtering functions which are well known to be robust with respect to non-rigid shape changes. The first one, which we denote by φ_{HK}, is chosen to be the heat kernel signature [5, 19], computed using the first 10 eigenfunctions of the Laplace-Beltrami operator and a fixed time t = 1000, and the second one, φ_G, the integral geodesic distance [16]. The invariance to scale comes from the a priori normalization of the models.

As for Db2, to define the considered filtering functions we proceeded as follows: For each triangle mesh M of vertices $\{P_1, \ldots, P_n\}$, the center of mass B is computed, and the model is normalized to be contained in a unit sphere. Further, a vector $\vec{w}$ is defined as

$$\vec{w} = \frac{\sum_{i=1}^{n}(P_i - B)\|P_i - B\|}{\sum_{i=1}^{n}\|P_i - B\|^2}.$$

We can think to the vector $\vec{w}$ as a generalization of the center of mass: Its computation is rotation and translation invariant, so that its relative position with respect to the corresponding triangle mesh does not change when rigid movements are taken into account. Moreover, the considered models are sufficiently generic (no point-symmetries occur, etc) to ensure that the vector $\vec{w}$ is well-defined over the all database, as well as its orientation is stable. Three filtering functions $\varphi_L, \varphi_P, \varphi_B$ are computed on the vertices of M: φ_L is the distance from the line parallel to $\vec{w}$ and passing through B, φ_P is the distance from the plane orthogonal to $\vec{w}$ and passing through B, and φ_B is the distance from B (see Figure 9 as an example). The values of φ_L, φ_P and φ_B are then normalized so that they range in the interval $[0, 1]$. These filtering functions are translation and rotation invariant, as well as scale invariant because of a priori normalization of the models.

The experiments. Suppose to have fixed a database, say Db, and a scheme, say Scm. For each corresponding filtering function φ, we can induce a metric over Db by computing the matching distances $d_{ij}^{\varphi} = d_{match}(D_0(M_i, \varphi), D_0(M_j, \varphi))$ for every $i, j = 1, \ldots, |\text{Db}|$, with |Db| the number of models in Db. To approximate such a metric, we apply Scm to get a lower bound for each d_{ij}^{φ}, say Res_{ij}^{φ}. This procedure is controlled by a threshold, $thresh^{\varphi}$, obtained as follows: For every class in the database, 4 elements are (randomly) selected, and an average of the matching distances on this small subset is evaluated. The final value of $thresh^{\varphi}$ is then the

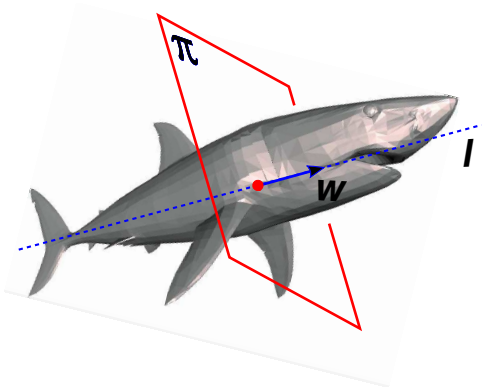

FIGURE 9. A shark model depicted with its center of mass, and the associated vector $\vec{w}$, which define the filtering functions $\varphi_L, \varphi_P, \varphi_B$.

average over all the classes in the database. In this perspective, the value $thresh^{\varphi}$ represents the average matching distance between two elements of the same class.

Now, if $Res_{ij}^{\varphi} > thresh^{\varphi}$, then we can assume that the shapes of M_i and M_j are quite dissimilar (compared with respect to φ) and therefore it is sufficient to have just an estimation of d_{ij}^{φ}: We opted for $((V-U)/2+Res_{ij}^{\varphi})/2$, with V and U taken as in Remark 3.2. In plain words, our estimation is the average between the lower bound (according to Theorem 3.1) and the upper bound (according to Remark 3.2) of d_{ij}^{φ}. If $Res_{ij}^{\varphi} \leq thresh^{\varphi}$, then the exact value of d_{ij}^{φ} is computed. The overall process is described in Algorithm 9.

Algorithm 9 MetricApprox$(A, B, Exp, thresh)$

```
Res = Scm(A, B, Exp)    %Scm is either RPGS or RBSS
if Res > thresh then
    Val = [(V − U)/2 + Res]/2
else
    Val = d_match(A, B)
end if
return Val
```

7. Experimental results

Experiments with RPGS. As seen in Section 4, the computational complexity of RPGS is a function of the number of iterations N and is shown to be $O(N^3)$. We want to apply this scheme to pay not more than computing matching distances for all the pairs of persistence diagrams presented in our databases. Indeed, we recall that computing d_{match} costs $O(h^{2.5})$, where h is the number of proper points in each comparison. We set $N = h^{Exp}$, with Exp a parameter we make vary in such a way that the inequality $Exp \leq \frac{2.5}{3}$ is verified. With this choice, we can ensure that RPGS has a computational complexity asymptotically lower than the one of the matching distance.

Tables 2 and 3 show how we applied RPGS to the two considered datasets. In the first column of Table 2 (Table 3, respectively), from top to bottom, we display

the average precision/recall (PR) graphs induced by φ_{HK}, and φ_G (φ_L, φ_P and φ_B), respectively, when considering the computation of the matching distances on the whole database Db1 (Db2, respectively) and on some subparts of it after running RPGS, with Exp set at different values. As can be seen, our approximation strategy does not affect so much the PR performances even in the displayed worst cases.

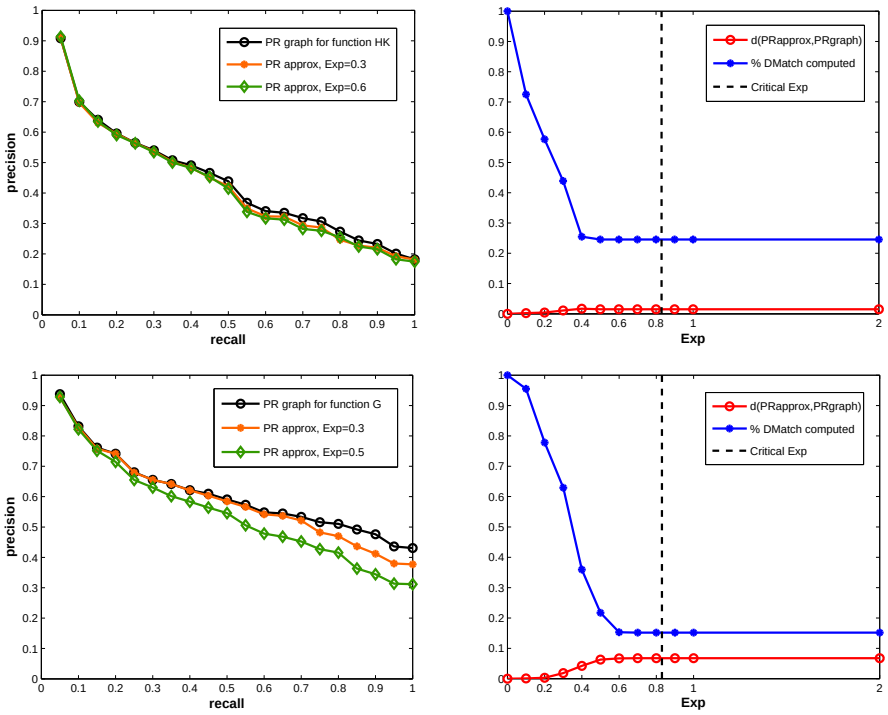

TABLE 2. First column: PR graphs related to φ_{HK} and φ_G computing d_{match} on the whole database (black), and on subparts of it (PR approx) by virtue of RPGS for two different values of Exp (shaded); Second column: varying Exp, how the percentage of d_{match} computed and the distance between PR graph and PR approx vary.

Table 2 (Table 3, respectively), second column, gives a more general overview of the obtained results. From top to bottom, each graph shows the reduction in the computational costs – in terms of the percentage of computed matching distances used to build the metric approximations – and an evaluation of the PR performances according to the chosen values of Exp, for the filtering functions φ_{HK}, and φ_G (φ_L, φ_P and φ_B), respectively. In particular, for a given value of Exp the evaluation of results is expressed as the average L_1-distance between the PR graph associated to that value Exp and the one obtained by computing all the matching distances between the elements in the database. The "critical Exp" depicted in all plots represents the value of Exp for which the cost of applying RPGS equals the one of computing the matching distance between two persistence diagrams.

As our plots show, it is possible to greatly reduce the computational costs by approximating the matching distance-based metric over the database, obtaining PR graphs which are quite close to the best possible.

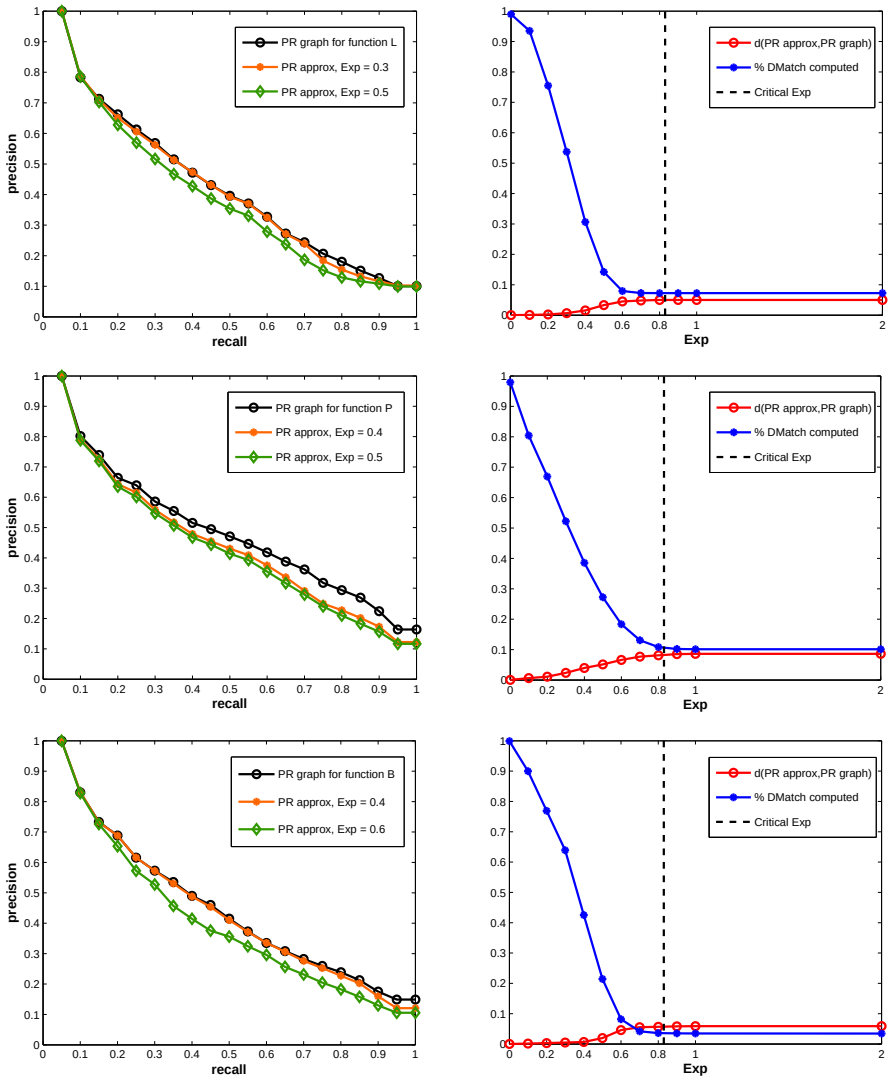

TABLE 3. First column: PR graphs related to φ_L, φ_P and φ_B computing d_{match} on the whole database (black), and on subparts of it (PR approx) by virtue of RPGS for two different values of Exp (shaded); Second column: varying Exp, how the percentage of d_{match} computed and the distance between PR graph and PR approx vary.

Experiments with RBSS. We performed similar experiments for RBSS. Given two lists A and B of proper points, we considered the Sum method (Algorithm 8) to make membership tests (see Algorithm 7, line 3). As for the choice of outer squares, we optimized Algorithm 7 by replacing Line 5 with the following:

$$p \Leftarrow \text{ random point from } A \cup B.$$

In other words, outer squares are randomly chosen in such a way that they are centered at proper points.

Having made use of the Sum method, by (4), the computational complexity of RBSS can be expressed as a function of R, i.e. the number of outer squares, and λ, being λ^2 the number of cells in the finest possible grid covering the triangular region T. Then, we fixed the value of λ in such a way that the number of cells in the grid and belonging to T is approximately $h = |A| + |B|$, that is, $\lambda = \sqrt{2h}$.

Finally, we set c (Algorithm 7, line 8) accordingly.

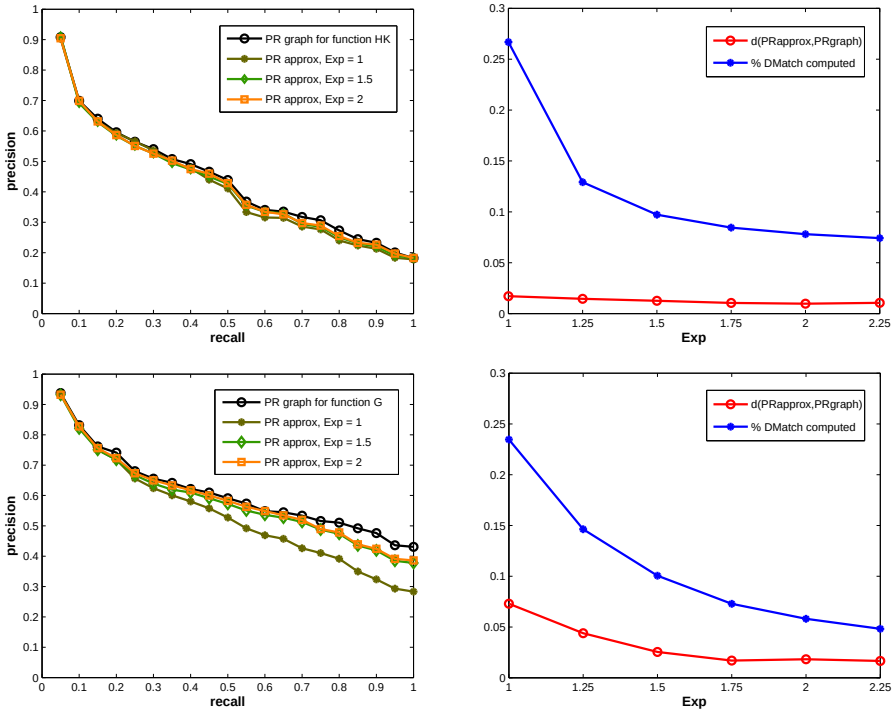

TABLE 4. First column: PR graphs related to φ_{HK} and φ_G computing d_{match} on the whole database (black), and average PR graphs computed on subparts of it (PR approx) by virtue of RBSS for three different values of Exp (shaded); Second column: varying Exp, how the percentage of d_{match} computed and the distance between PR graph and PR approx vary.

Similarly to the case of RPGS, we want to apply RBSS to have an estimation from the bottom of d_{match}, possibly paying less than the computational cost of the matching distance. To achieve this, by (5), we have to assure that $O(R) \cdot O(\log_2(\lambda)) \leq O(h^{2.5})$. Indeed, having fixed λ in such a way that the first inequality in (5) is verified, it is sufficient to require that $R \leq \frac{h^{2.5}}{\log_2(\sqrt{2h})}$.

In our experiments we set $R = \frac{h^{Exp}}{\log_2(\sqrt{2h})}$, letting Exp range from 1 (i.e., the number of outer squares is proportional to the number of the considered proper points) to 2.25. We observe that $Exp = 2.5$ correspond to a limit situation, in which the computational cost of RBSS approaches the one of computing d_{match}.

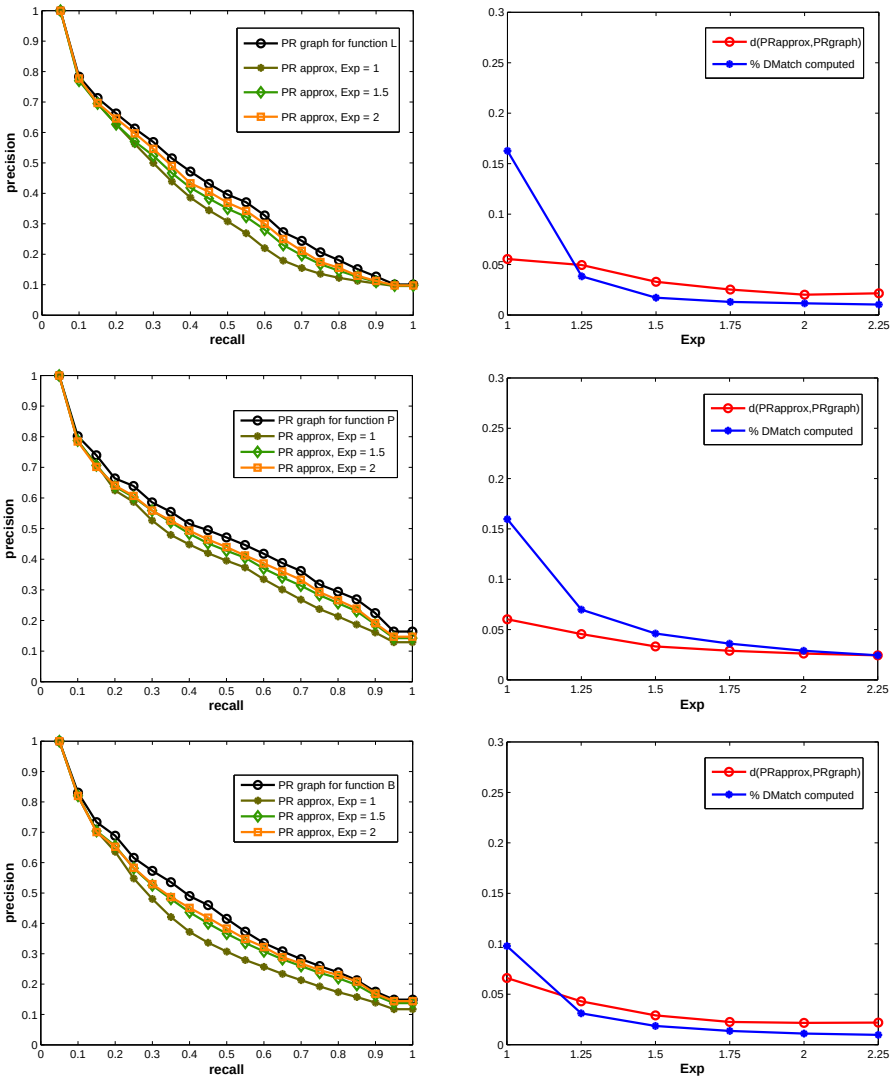

TABLE 5. First column: PR graphs related to φ_L, φ_P and φ_B computing d_{match} on the whole database (black), and average PR graphs computed on subparts of it (PR approx) by virtue of RBSS for three different values of Exp (shaded); Second column: varying Exp, how the percentage of d_{match} computed and the distance between PR graph and PR approx vary.

Tables 4 and 5 show our results. We remark that, due to intrinsic randomness of RBSS, we repeated the same experiments more than once, observing however almost identical performances. Nevertheless, we decided to present here experimental results representing the average of all considered experimental instances.

In the first column of Table 4 (Table 5, respectively), from top to bottom, we display the average PR graphs induced by φ_{HK}, and φ_G (φ_L, φ_P and φ_B), respectively, when considering the computation of the matching distances on the whole database

Db1 (Db2, respectively) and on some subparts of it after running RBSS, with Exp set at different values. Even for RBSS, our approximation strategy produces good results in terms of PR performances, if compared with the ones corresponding to the metrics induced by d_{match}.

Table 4 (Table 5, respectively), second column, analyzes the obtained results under a different viewpoint. From top to bottom, each graph shows the reduction in the computational costs – in terms of the percentage of computed matching distances used to build the metric approximations – and an evaluation of the PR performances according to the chosen values of Exp, for the filtering functions φ_{HK}, and φ_G (φ_L, φ_P and φ_B), respectively. In particular, for a given value of Exp the evaluation of results is expressed as the average L_1-distance between the PR graph associated to that value Exp and the one obtained by computing all the matching distances between the elements in the database.

8. Discussion

In this paper we continued the work started in [8]. We presented two multi-scale strategies, RGPS and RBSS, for the evaluation of a matching distance-based (dis)similarity metric induced on a database of shapes. The capabilities of the proposed frameworks in CRC applications have been validated through experiments on two datasets of 3D models represented by triangle meshes, whose shapes have been analyzed by considering five batteries of persistence diagrams.

The obtained results show that, using our multi-scale approaches, it is possible to provide an approximation of the metric induced by the matching distance between persistence diagrams without compromising the goodness of results – in terms of retrieval performance – and greatly reducing the computational costs characterizing the exact evaluation of the matching distance.

In particular, experiments on RBSS produced results which are, in our opinion, even surprising. Indeed, as displayed by Tables 4 and 5, a decreasing in the number of the actually computed matching distances corresponds in general to an increasing in the goodness of results (in terms of retrieval performance). This is actually shown by PR graphs we obtained by computing d_{match} only on a subpart of the considered datasets, as well as by the L_1-distance between these PR graphs and the ones obtained by computing all possible matching distances.

Having such experimental evidence in a hand, we may argue that RBSS can achieve high accuracy level in approximating the matching distance. On the other hand, we do not have any theoretical results confirming our intuition. This is certainly an open problem we plan to address in the next future.

Moreover, it would be interesting to explore the capability of RBSS in other theoretical settings, e.g., selecting the outer squares according to some probability distribution built on the mutual positions of the considered proper points.

References

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sept. 1975.

[2] S. Biasotti, A. Cerri, P. Frosini, and D. Giorgi. A new algorithm for computing the 2-dimensional matching distance between size functions. *Pattern Recognition Letters*, 32(14):1735 – 1746, 2011.

[3] S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Comput. Surv.*, 40(4):1–87, 2008.
[4] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 1 edition, 2008.
[5] A. M. Bronstein, M. M. Bronstein, M. Ovsjanikov, and L. J. Guibas. Shape google: geometric words and expressions for invariant shape retrieval. *ACM Trans. Graphics (TOG)*, 30(1):1–20, 2011.
[6] F. Cagliari and C. Landi. Finiteness of rank invariants of multidimensional persistent homology groups. *Appl. Math. Lett.*, 24(4):516–518, 2011.
[7] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas. Persistence barcodes for shapes. *IJSM*, 11(2):149–187, 2005.
[8] A. Cerri, B. D. Fabio, and F. Medri. Multi-scale approximation of the matching distance for shape retrieval. In M. Ferri, P. Frosini, C. Landi, A. Cerri, and B. D. Fabio, editors, *CTIC*, volume 7309 of *Lecture Notes in Computer Science*, pages 128–138. Springer, 2012.
[9] F. Chazal, D. Cohen-Steiner, L. J. Guibas, F. Mémoli, and S. Oudot. Gromov-hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum*, 28(5):1393–1403, 2009.
[10] M. d'Amico, P. Frosini, and C. Landi. Using matching distance in size theory: A survey. *Int. J. Imag. Syst. Tech.*, 16(5):154–161, 2006.
[11] M. d'Amico, P. Frosini, and C. Landi. Natural pseudo-distance and optimal matching between reduced size functions. *Acta. Appl. Math.*, 109:527–554, 2010.
[12] B. Di Fabio, C. Landi, and F. Medri. Recognition of occluded shapes using size functions. In *Lecture Notes In Computer Science*, volume 5716, pages 642–651, 2009.
[13] H. Edelsbrunner and J. Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010.
[14] P. Frosini and C. Landi. Size theory as a topological tool for computer vision. *Pattern Recogn. and Image Anal.*, 9:596–603, 1999.
[15] P. Frosini and C. Landi. Size functions and formal series. *Appl. Algebra Engrg. Comm. Comput.*, 12(4):327–349, 2001.
[16] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01*, pages 203–212, 2001.
[17] D. Kincaid and E. Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. Pure and applied undergraduate texts. American Mathematical Society, 2002.
[18] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. PAMI*, 22(12), 2000.
[19] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proc. SGP '09*, pages 1383–1392, 2009.
[20] J. Tangelder and R. Veltkamp. A survey of content-based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471, 2008.

ARCES, Università di Bologna, Italia
E-mail address: andrea.cerri2@unibo.it

ARCES, Università di Bologna, Italia
E-mail address: barbara.difabio@unibo.it

Institute of Computer Science, Jagiellonian University, Poland
E-mail address: grzegorz.jablonski@uj.edu.pl

Dipartimento di Scienze dell'Informazione, Università di Bologna, Italia
E-mail address: filippo.medri@gmail.com