Computers & Graphics 37 (2013) 460–472

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on 3D Object Retrieval

Data-aware 3D partitioning for generic shape retrieval ☆, ☆☆

Ivan Sipiran [a,*], Benjamin Bustos [a], Tobias Schreck [b]

[a] KDW+PRISMA Research Group, Department of Computer Science, University of Chile, Chile
[b] Visual Analytics Group, Department of Computer and Information Science, University of Konstanz, Germany

ARTICLE INFO

Article history:
Received 24 October 2012
Received in revised form
4 April 2013
Accepted 5 April 2013
Available online 1 May 2013

Keywords:
Mesh partitioning
Optimization matching
Shape retrieval

ABSTRACT

In this paper, we present a new approach for generic 3D shape retrieval based on a mesh partitioning scheme. Our method combines a mesh global description and mesh partition descriptions to represent a 3D shape. The partitioning is useful because it helps us to extract additional information in a more local sense. Thus, part descriptions can mitigate the semantic gap imposed by global description methods. We propose to find spatial agglomerations of local features to generate mesh partitions. Hence, the definition of a distance function is stated as an optimization problem to find the best match between two shape representations. We show that mesh partitions are representative and therefore it helps to improve the effectiveness in retrieval tasks. We present exhaustive experimentation using the SHREC'09 Generic Shape Retrieval Benchmark.

© 2013 The Authors. Published by Elsevier Ltd. All rights reserved.

1. Introduction

Three-dimensional objects are a valuable resource in many fields such as engineering and medicine. They can represent the shape of a real object in a suitable way in order to be used by computers. The versatility of this representation has resulted in an increasing interest of the scientific community in several related topics. For instance: shape analysis, shape processing, modeling applications, and so on. In addition, it is currently possible to find massive and publicly available 3D data. For example the Google Sketchup collection, for which its use is becoming a common practice. For these reasons, the search for efficient and effective tools for this kind of data is imperative in order to support future applications.

In particular, the content-based similarity search of 3D objects has received much attention in recent years. This can be performed without relying on additional information for searching, only using the provided shapes. Additionally, many fields (for example medicine [1,2], CAD/CAM [3], etc.) have benefited from the large amount of approaches proposed to overcome the problem of 3D matching. Nevertheless, the problem remains challenging and it is far from being completely solved. Moreover, part of the problem resides in the possibility of defining a suitable similarity measure between 3D models.

In this paper, we consider the problem of generic shape retrieval. A common approach to facing this problem is to compute an intermediate representation (feature vectors or graphs, for instance) and subsequently defining the similarity of two objects as the similarity of their representations. In this direction, there are methods that exploit the visual similarity, the statistical properties of 3D measures, or the possibility of defining transform functions on the data, just to name a few. However, one of the most critical problems is the semantic gap. That is, the intermediate representation may not be able to capture all the needed information of a shape and therefore the effectiveness of searching may be seriously affected.

A previous study by Bustos et al. [4] showed that some features could well represent certain classes of objects and furthermore, some features could be complimentary in representing a shape. This is because algorithms cover only a part of the possible spectrum of characteristics such as shape, silhouette, or intrinsic properties. Thus, a natural extension of classic approaches was the combination of features for improving the effectiveness of retrieval. Approaches in this direction have been previously presented by Bustos et al. [5], Vranic [6], and Papadakis et al. [7], all of them with promising results. However, the semantic gap is still latent in this approach as any possible combination of features could not represent important characteristics to discriminate between objects.

A more recent approach is the combination of global and part-based information. The idea is to combine features extracted from an entire object with features extracted from parts of an object. Some techniques have been presented so far by Li and Johan [8], Bustos et al. [9], and Schreck et al. [10]. All these techniques share a common aspect: the part-based features come from a fixed partitioning of the objects. Although it was possible to improve the effectiveness with respect to using only global features, the fixed

☆This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial-No Derivative Works License, which permits non-commercial use, distribution, and reproduction in any medium, provided the original author and source are credited.

☆☆To comment on this article, please join the discussion on the Collage Authoring Environment Google Group https://groups.google.com/group/collage-authoring-environment.

* Corresponding author. Tel.: +56 2 29784972.
E-mail addresses: iasipiranm@gmail.com, isipiran@dcc.uchile.cl (I. Sipiran), bebustos@dcc.uchile.cl (B. Bustos), tobias.schreck@uni-konstanz.de (T. Schreck).

0097-8493/$ - see front matter © 2013 The Authors. Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.04.002

Fig. 1. Two globally dissimilar chairs. Note that the chair at right is taller than the left one. Nevertheless, it is possible to find similarities between their parts, which can be exploited to improve the similarity measure between the two objects.

partitioning limits the possibility of having truly distinctive parts. This opens up a question on how to define a new kind of partitioning dependent on the shape information.

We believe that the use of local features can enhance the use of global features in shape retrieval. That is, we are trying to mitigate the effect of the semantic gap. For instance, a common fact is having two objects with different appearance in the same class. Obviously, a global feature could differ in those objects. However, in a local sense, it is still possible to find correspondences between parts, so we can take advantage of this fact to improve the similarity measure (see Fig. 1). Therefore, the discriminative power of local features combined with global features could help to improve the effectiveness in the similarity search.

In this paper, we propose a shape retrieval method using a data-aware partitioning algorithm. Our idea is to exploit the local characteristics of objects to determine discriminative parts. Thus, each object is represented by its global feature and a set of features extracted from parts. The partitioning method relies on finding robust local features (namely keypoints) on the object's surface and subsequently determining the parts where there is a high concentration of keypoints (for instance, a human shape commonly has many features located in hands, feet, and head). Beyond techniques which made use of the bag of features approach to aggregate local descriptors for retrieval, our method is the first attempt in combining global and local features found in a data-adaptive way for generic shape retrieval.

Our main contribution is three-fold:

- We propose a model partitioning algorithm based on local features. Regions on the surface with high concentration of local features will be selected as parts.
- We combine the global feature with features obtained from parts and define a combined distance to assess the similarity. The distance between global features is performed as usual. The distance between sets of parts is stated as an optimization problem. In addition, we propose a geometrical consistency criterion which can be formulated within the same optimization problem.
- We evaluate our approach using a well-known, established benchmark dataset and appropriate performance measures.

Our approach is a generic, simple framework by which global and local descriptors can be combined in a data-adaptive way. The approach is able to provide on average, an improvement over the retrieval effectiveness of state of the art global descriptors. A careful, systematic analysis of the results is performed to assess in detail the magnitude of the improvement, relating it with global-only methods, and identifying classes of models for which the method is particularly effective. We test our approach using a state-of-the-art local interest point detector with desirable properties in combination with two robust view-based descriptors. Our approach is flexible in that it can accommodate further, possibly application-specific, object segmentation and description schemes, if needed.

Our paper is organized as follows. Section 2 briefly presents the state of the art in generic shape retrieval. Section 3 describes our partitioning algorithm based on local features. Section 4 is devoted to the matching methods and the definition of our similarity measure. Section 5 describes our experiments and presents the discussion of our results. Finally, Section 6 draws the conclusions.

2. Related work

The interest in 3D model retrieval has resulted in a large amount of proposed techniques to overcome the problem. One of the most studied approaches is to convert a 3D model into a more convenient representation for comparison, for example feature vectors. Then, the comparison can be done by defining a distance between those representations. For generic shape retrieval, this approach has received attention due to the efficiency of computing distances between vectors. In this section, we provide a brief description of the state of the art related to descriptors for generic shape retrieval and possible combinations to improve the performance. For a comprehensive study, surveys by Bustos et al. [11] and Tangelder and Veltkamp [12] are an excellent resource.

Classic methods for 3D shape retrieval can be classified into three groups: view-based, histogram-based, and transform-based. This classification is based on how a feature is extracted from the shape. View-based methods transform a 3D shape into a set of 2D views and subsequently we can apply image techniques to describe the obtained views. For example, the Depth Buffer method [13] computes six views corresponding to the six faces of the bounding cube of an object. Each view stores the projected distances from the object to the projection plane. Then, each view is represented by Fourier coefficients and the final vector is the concatenation of the six obtained views. Another example is the PANORAMA descriptor [14], which computes three views taken from the lateral faces of cylinders oriented according to the coordinate axes. Similar to the Depth Buffer, each lateral face encodes the distance from the object to the face. Then, Fourier and Wavelets coefficients are extracted from each view, which form the final descriptor.

Histogram-based methods summarize shape properties in order to use them as features. For instance, Shape Distribution [15] is a method that computes several geometric properties (distances between pairs of surface points, angles between three random surface points, etc). The method consists of sampling a large amount of points on the shape surface and subsequently measuring some property. Each value obtained for the chosen property is accumulated in a histogram. Thus, the histogram represents an approximation of the distribution of the property and it is expected to be distinctive for each object.

Transform-based methods consist of converting the geometric information by using some mathematical transformation prior to the feature extraction. The goal of applying a transformation is to enhance some information which is not evident in the Euclidean space. In particular, in 3D model retrieval, there is an interest for spherical harmonics to extract features from shapes. Vranic proposed the ray-based descriptor [13] by using a spherical function which is able to capture the behavior of the rays starting in the origin and the intersections with the shape. Similarly, Kazhdan et al. [16] used spherical harmonics frequencies along with the Gaussian Euclidean Distance Transform in a volume representation of a shape.

An interesting and new approach is the combination of different descriptors to improve the performance of individual descriptors. The basic idea is that different descriptors could extract complementary features and their combination could lead to improvements. Bustos et al. [5] proposed to dynamically

combine several descriptors using a weighting scheme dependent on the query. Similarly, Vranic [6] proposed to combine three descriptors: Silhouette, Ray-based and Depth-Buffer. This combination (which was called DESIRE) improved the performance of the individual descriptors. On the other hand, Papadakis et al. [7] suggested combining 2D and 3D features to improve the performance of retrieval. As a 2D feature, the authors proposed to use the Depth Buffer method and as 3D feature, they proposed to use spherical harmonics transform for spherical functions obtained from the shape.

The aforementioned combination methods consist of somehow combining two or more description methods. However, these techniques still rely on the global shape for the calculation of each descriptor. Recent approaches have considered the combination of global information and part-based information. Li and Johan [8] used global and local radial distances to describe a shape. First, the method computes a radial distance descriptor by uniformly dividing the surface of a sphere containing the object. The division considers bins at different angle intervals and the average distance of each bin to the object is stored on it. Second, the local component of the method consists of uniformly dividing the bounding cube of the object into $N \times N \times N$ cells. For each vertex on the shape, the method computes the minimum distance to the cell centers and the distance is assigned to the vertex. Finally, thirteen views are extracted using the assigned distances as RGB values. The distance between two shapes with this representation is measured pair-wise between global and local descriptions.

Also, Bustos et al. [9] proposed a simple partitioning scheme in order to combine it with global descriptors. Given a shape, the method computes a global descriptor for it. Next, the shape is divided into eight parts according to the eight octants obtained with the coordinate axes in the 3D Euclidean space. Finally, the method computes a descriptor for each part. To measure the distance of global-partial representations, the authors evaluated several weighting schemata where adaptive weighting showed the best performances. Similarly, Schreck et al. [10] also took the octant partition as a basis. Nevertheless, this new technique considered the matching of the parts as a bipartite graph matching problem. In addition, the authors tested the use of different numbers of parts. Interestingly, it was shown that not using all parts (6 or 7 depending on the dataset) outperformed the retrieval performance.

Regarding the use of local features to decompose a 3D shape, several approaches have been proposed for non-rigid and partial shape retrieval. Toldo et al. [17] proposed to apply a spectral clustering to decompose a mesh into regions. Each region was further described with information such as the shape index, radial geodesic distances and normal directions. The final representation was obtained using a multi-level bag-of-features approach. On the other hand, Shapira et al. [18] presented a technique for describing mesh segments. The segmentation is hierarchically performed using SDF histograms [19]. Next, contextual information is used in order to improve the matching between parts. A bipartite graph is used to measure the context-aware distance between two objects. In addition, mesh decomposition is recently being used as an alternative to 3D shape matching and retrieval. Litman et al. [20] defined maximally stable components on meshes using geometry diffusion. Similarly, Sipiran and Bustos [21] used a clustering in the geodesic space to define key-components on meshes.

Our method can be considered as a combination of local and global approaches for the problem of generic shape retrieval. Specifically, we aim at evaluating if the performance of global descriptors can be improved by using a mesh decomposition approach.

3. Data-aware 3D partitions

Previous approaches have tried to use 3D mesh partitions as input in retrieval tasks. In this section, we present a partition algorithm based on finding groups of discriminative local features. Our method does not guarantee disjoint or complete partitions. However, as it uses interest points detected on the mesh for partitioning, we believe that the resulting fragments are representative enough. Therefore, the partitions can be useful for improving the matching between two 3D models.

Our method consists of three steps:

- *Interest point detection*: We aim at selecting a small set of points on the mesh surface. We consider that a vertex is interesting if it has an outstanding geometric structure in comparison with its neighborhood.
- *Clustering of interest points*: We perform a clustering in order to find groups of interest points under some constraints.
- *Cluster-based partition*: We use the resulting clusters for defining representative partitions for matching.

3.1. Interest point detection

There is no agreement about what an interest point is and how it can be formally defined. We will define an interest point as a mesh's vertex whose geometric structure is different from its neighborhood. Note that the robustness of the method will depend on how the geometric structure is measured and how it can support variations such as noise or missing data. In particular, for a general-purpose 3D object retrieval system, we require a robust and fast method. This is because shapes can come from several sources, so there are no guarantees with respect to manifoldness, noise, resolution, and so on.

We use the Harris 3D method [22] to select the set of interest points in a mesh. This technique has proven to be effective and robust against several transformations. In particular, it has been shown to deliver good repeatability (localization) in light of different scales, noise, and other model transformations [23]. This is a desirable property in that the Harris detector can be expected to find comparable interest points for inclusion in the similarity function. Also, in [24] it was shown that the Harris detector provides points on the surface which are comparable to keypoints annotated by humans. Furthermore, in the same report, it was noted that the Harris 3D method offers a good number of features compared to other algorithms. For example, it delivers less interest points than Mesh saliency and SD-corners; and it delivers more features than Heat Kernel Signatures [24]. Therefore, we consider the Harris detectors to not only provide repeatable, but also meaningful interest points which are expected useful for the similarity function. In addition, the detector is fast enough, and it can be easily used without detriment to the retrieval time. Briefly, the Harris 3D method can be summarized with the following steps:

- It determines a neighborhood around each vertex. The neighborhood can be spatial (all points lying inside a 3D sphere centered in a vertex), adaptive (all points forming rings around a vertex and a certain geodesic distance from the vertex), and rings (all points around a vertex regarding the number of rings).
- Then the method finds a canonical local system by applying PCA to the neighborhood.
- After that, the algorithm fits a quadratic surface on the normalized neighborhood.
- Subsequently, it computes derivatives on the fitted surface. Gaussian functions are used for smoothing derivatives. By using

integration between the derivatives and the Gaussians, the method is robust to local geometric changes.
- Then the method constructs (using the derivatives) the auto-correlation function needed to evaluate the Harris operator. Subsequently, a response is computed for each vertex.
- Finally, it selects a set of vertices as interest points by applying some criteria on the vertex's response. The original method proposed two criteria: a number of vertices with the highest response and an spatial criterion for well distributed interest points.

In this paper, we use the Harris 3D method for computing the vertex's response, and subsequently we select the vertices with the highest response. In Section 5, we evaluate the effect of the selection of keypoints in the performance of our retrieval method.

3.1.1. Control of mesh resolution

Note that the Harris 3D method depends on local neighborhoods around a vertex. Nevertheless, generic 3D shapes could come from different sources where their primary goal was not the analysis or processing. It is therefore common to find objects with bad triangulations. Moreover, many meshes are optimized for rendering, so regular portions of them are represented by large triangles. It poses a problem for 3D analysis, where meshes with regular triangulations are preferably needed. Therefore, it is necessary to control the size of the neighborhoods prior to the interest point detection. In addition, our goal is to ensure a consistent neighborhood computation along the entire mesh.

We implement the algorithm for control of mesh resolution proposed by Johnson [25]. This algorithm assumes the spacing between vertices as the resolution to be improved. More specifically, the mesh resolution is the median of the edge length histogram. The goal is to decrease the edge length spread (or variance) of the histogram around a desired resolution. To accomplish this goal, the algorithm performs local operations over the edges which are too large (split operation) or too small (collapse operation). Each edge is associated with a weight which combines its length difference with the desired mesh resolution and the geometric shape change if any operation is performed. Finally, a greedy strategy performs local operations guided by a priority queue defined over the weights. An example of our implementation is shown in Fig. 2.

3.2. Clusters of interest points

Once we have computed the interest points for a mesh, our goal is to use them for extracting representative partitions. The main idea is to find clusters of interest points in the 3D space, so each cluster would define a portion of the mesh which is interesting and distinctive. We propose an adaptive clustering algorithm taking into account the intra-cluster and the inter-cluster structure of the clusters which generate the mesh partitions. In addition, we add a constraint regarding the number of elements per cluster. It allows us to discard interest points which do not belong to any cluster, and therefore they should not be part of the partition representation. Our algorithm is derived from Leow and Li [26], and is presented in Algorithm 1.

Algorithm 1. Adaptive clustering.

Require: Set of points P
Require: Inter-cluster distance R
Require: Intra-cluster distance S
Require: Minimum number of elements per cluster Nm
Require: Number of iterations $Iter$
Ensure: Set of clusters $C = \{C_1, \ldots, C_m\}$

```
Let C a set of clusters
C←∅
for j←1 to Iter do
    for each p∈P do
        Let C_i be the closest cluster to p with distance d.
        If C = ∅ then d = 2R.
        if d > R then
            Create a new cluster C_new with p as element.
            Insert C_new into C.
        else if d≤S then
            Insert point p into C_i.
        end if
        Remove point p from P.
    end for
    for each cluster C_i in C do
        if |C_i|≥Nm then
            Update centroid for C_i
        else
            Insert each point in C_i into P.
            Remove C_i from C.
        end if
    end for
end for
Return C
```

The adaptive clustering algorithm uses two distance thresholds R, S, and the minimum number of points in a cluster N_m as parameters. The algorithm scans $Iter$ times the set of points trying to find clusters which hold two criteria: the distance between each point within a cluster to its centroid is not larger than S (intra-cluster constraint), and the distance between cluster's centroids is not smaller than R (inter-cluster constraint). In addition, if after a scan, the number of points inside a cluster is less than N_m, the

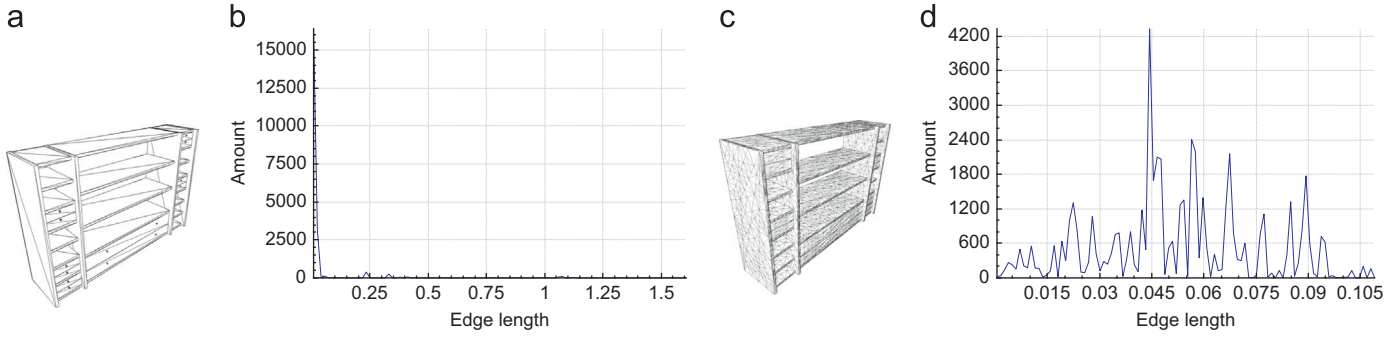

Fig. 2. Effect of mesh resolution: shape with bad triangulation (a) and its poorly distributed edge length histogram (b). Shape processed with the mesh resolution algorithm (c) and its improved edge length histogram.

cluster is discarded. Also, the points of the small cluster are pushed back in the point collection for the next iteration. If a cluster has more than N_m points, its centroid is updated.

Note that there may exist points which never hold with the cluster constraints, and those points are simply discarded. We are interested in groups of interest points, because these could be in a representative part of the mesh. Hence the behavior of discarding isolated points is important for our purposes, because those points could be noise and therefore would not represent an important feature of the mesh.

Another important aspect is the flexibility of the adaptive clustering algorithm with respect to the obtained number of clusters. The number of clusters depends on the point distribution and the cluster parameters. This is an advantage because each object would have a different number of clusters depending on their interest points. In this way, each object would have a data-aware flexible representation.

The presented clustering algorithm determines a spatial partitioning of the keypoints where clusters are always circle-shaped. Hence, we also tested the DBSCAN [27] algorithm for clustering which has a different functionality. DBSCAN is a density-based algorithm which is able to detect clusters of arbitrary shapes and it is based on proximity and density concepts. However, after experimentation, we found that it was difficult to correctly define the density thresholds for this algorithm. In addition, in many of our experiments, DBSCAN computed very few clusters per shape, underestimating the representational power of the interest points.

3.3. Partitioning and description

Our partitioning algorithm is quite simple. For each cluster of interest points, we proceed as follows:

- The algorithm computes the smallest 3D sphere containing all points within the cluster. We used a linear programming algorithm for this purpose [28]. The outputs of this step are the center of the sphere and the radius.
- Next, we extract the portion of the mesh lying inside of a 3D sphere formed by the previously computed center and the radius scaled by a factor of δ (we study the effect of δ in Section 5). It can be done by scanning the complete set of vertices and verifying which vertex lies inside the sphere. However, this can be computationally expensive for large meshes. Here we use an improved method. We build a kd-tree with all vertices of the mesh, and thus a range search is performed using as query the center of the sphere and the radius. Note that the kd-tree needs to be built only once, and it can be used for each partition by changing the query. Finally, the method builds a new mesh using the set of points inside the sphere and their associated faces.

For description, we compute a global descriptor for the entire model and subsequently compute a global descriptor for each partition. Formally, given an object O, its representation is defined as

$$S_O = \{(s_O, P_O) | s_O \in R^n \text{ and } P_O = \{p_O^1, p_O^2, \ldots, p_O^m\}, p_O^i \in R^n\},$$

where s_O is an n-dimensional descriptor representing the complete object, P_O is a set of m n-dimensional descriptors representing each partition. Fig. 3 depicts an example of some partitions obtained with our algorithm.

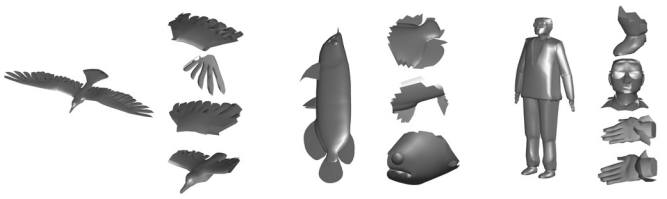

Fig. 3. Three examples of partitions obtained with our method using the parameter configuration used in Section 5.2.

4. Matching

At this point, we need to define a distance between two representations as shown previously. Given two 3D objects O and Q, each with their representations:

$$S_O = \{(s_O, P_O) | s_O \in R^n \text{ and } P_O = \{p_O^1, p_O^2, \ldots, p_O^m\}, p_O^i \in R^n\}$$

and

$$S_Q = \{(s_Q, P_Q) | s_Q \in R^n \text{ and } P_Q = \{p_Q^1, p_Q^2, \ldots, p_Q^k\}, p_Q^i \in R^n\},$$

where O has m partitions, and Q has k partitions. Our goal in this section is to define an appropriate distance $d(S_O, S_Q)$, which measures the dissimilarity between two objects using their representations. The main problem is how to define a dissimilarity between two sets of descriptors with different lengths.

In this paper, we only consider a linear combination between the global-to-global distance and the partition-based distance. That is

$$d(S_O, S_Q) = \mu \|s_O - s_Q\|_2 + (1-\mu) d(P_O, P_Q), \quad (1)$$

where $0 \le \mu \le 1$ weights the contribution of the involved terms.

4.1. Integer linear programming

The matching problem is how to find a correspondence set between two collections of descriptors. Clearly, this problem is difficult because it is not possible (at least not within a reasonable time) to evaluate all possible combinations of correspondences. We will state the problem using a linear programming formulation for searching a feasible solution.

We define an indicator variable as follows:

$$x(i,j) = \begin{cases} 1 & \text{if } p_O^i \text{ matches } p_Q^j \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Note that $|x| = m \times k$, that is each element in P_O could be matched with each element in P_Q. Let us think of x as a binary string of length $m \times k$. The number of configurations for x is $2^{m \times k}$, which allows us to figure out the complexity of the problem. Obviously, if m and k are large enough, the number of possible matches increases exponentially. Nevertheless, we can add some constraints to the problem. For instance, if ${p_O}^i$ is already matched to ${p_Q}^j$, then ${p_O}^i$ should not be matched to any other element in P_Q. Formally, if $x(i,j) = 1$, then $\sum_j x(i,j) = 1$, and therefore also $\sum_i x(i,j) = 1$.

We use the indicator variable x to formulate an objective function as follows:

$$f(x) = \sum_{i,j} \|p_O^i - p_Q^j\|_2 \cdot x(i,j), \quad (3)$$

where the goal is to find the optimum x^* which minimizes $f(x)$. Formally,

$$x^* = \underset{x}{\operatorname{argmin}}\, f(x), \quad (4)$$

subject to

$$\sum_i x(i,j') = 1 \quad \text{and} \quad \sum_j x(i',j) = 1 \ \forall i,j$$

Moreover, we can consider the optimum $f(x^*)$ as the dissimilarity function $d(P_O, P_Q)$. However, the optimum $f(x^*)$ depends on the number of matches, reaching lower values when P_O and P_Q have a few elements. In order to overcome this problem, we normalize the value of the optimum, and we obtain the final dissimilarity measure:

$$d(P_O, P_Q) = \frac{f(x^*)}{\min(|P_O|, |P_Q|)}. \tag{5}$$

Note that the normalization in Eq. (5) also contributes to maintain the symmetry of the distance. This is an important aspect if one considers indexing the distance for fast searching.

4.1.1. Numerical aspects

To numerically solve Eq. (4), we define a matrix of distances

$$C(i,j) = \|p_O^i - p_Q^j\|_2, \tag{6}$$

where each element of this matrix stores the L_2 distance between descriptors from P_O and P_Q. Thus, the problem of finding x^* in Eq. (4) can be stated as a binary linear programming problem

$$\min_x C^T x \quad \text{such that} \begin{cases} Ax \le b \\ A_{eq}x = b_{eq} \\ x \text{ is binary} \end{cases} \tag{7}$$

where C and x are linearized versions of themselves, A and b represent linear inequality constraints, and A_{eq} and b_{eq} represent linear equality constraints. In fact, the constraints $\sum_i x(i,j) = 1$ and $\sum_j x(i,j) = 1$ need to be placed in the linear constraints.

The solution for the problem in Eq. (7) is given by a branch-and-bound algorithm which tries to solve it using LP-relaxation approaches [29].

4.2. Integer quadratic programming

The linear programming formulation finds the best set of correspondences only regarding the dissimilarity between descriptors in P_O and P_Q. The problem with this formulation is that it discards the spatial information of the partitions from which the descriptors come. Obviously, our algorithm does not ensure consistency in the spatial sense. In this section, we enrich our previous formulation by adding spatial consistency between descriptors.

Recalling the indicator variable x. If we have two correspondences $x(i,j) = 1$ and $x(i',j') = 1$, one can expect that the spatial relationship between fragments i and i' from shape O is quite similar to the spatial relationship between fragments j and j' from shape Q. Of course, the idea is to minimize the difference between spatial distances of partitions, while maintaining the dissimilarity between descriptors. Therefore our new formulation for Eq. (3) is

$$f(x) = \alpha \sum_{i,j,i',j'} |d_S^O(i,i') - d_S^Q(j,j')| x(i,j) x(i',j') + \beta \sum_{i,j} \|p_O^i - p_Q^j\|_2 \cdot x(i,j) \tag{8}$$

where $d_S^O(i,i')$ is the spatial distance between fragments i and i' from the object O, α and β are weights to set the contribution of the spatial consistency and the descriptor dissimilarity, respectively. In addition, the new formulation is subject to the same constraints as Eq. (4). Finally, we can use the new formulation to find an optimum x^* and therefore we will use the same distance as shown in Eq. (5).

Regarding the spatial distances, during the process of finding the partitions, we compute distances between the centers of the spheres which generate the partitions. In this way, the algorithm makes available the spatial information in the matching.

4.2.1. Numerical aspects

To numerically solve Eq. (4) using the objective function in Eq. (8), we define a matrix with the distance differences as follows:

$$D(\{i,j\}, \{i',j'\}) = |d_S^O(i,i') - d_S^Q(j,j')|, \tag{9}$$

where $\{i,j\}$ denotes the linear index of the pair (i,j). Clearly, we need to consider the complete set of spatial relationships between pairs of partitions. The dimension of the matrix D is $mk \times mk$. Thus, the problem of finding x^* in Eq. (8) can be stated as a binary quadratic programming problem

$$\min_x \frac{1}{2} x^T D x + C^T x \quad \text{such that} \begin{cases} Ax \le b \\ A_{eq}x = b_{eq} \\ x \text{ is binary} \end{cases} \tag{10}$$

where C and the constraints were defined in Eq. (7).

The solution for the problem in Eq. (10) is also given by a branch-and-bound algorithm with *LP*-relaxation, but in this case using a quadratic objective function [30].

5. Experiments

In this section, we present our experiments and results. The section is organized as follows. Section 5.1 presents the experimental setup, in addition to the dataset and evaluation measures. Section 5.2 presents a study of the contribution of partition matching in the overall method. Section 5.3 presents a sensitivity analysis of parameters. Section 5.4 discusses the effectiveness of our method in a class-by-class analysis. Section 5.4.1 investigates the correlation between effectiveness and important aspects such as number of vertices and number of parts. Finally, Section 5.4.2 presents results using the PANORAMA descriptor.

5.1. Experimental setup

For our experiments, we use the SHREC'2009 generic benchmark [31]. This benchmark contains 720 shapes organized in 40 classes with 18 shapes per class. To evaluate the retrieval effectiveness of our method, we use common measures in the retrieval community such as mean average precision (MAP), nearest neighbor (NN), First Tier (FT) and Second Tier (ST). Briefly, we describe each measure as follows:

- Mean average precision (MAP): Given a query, its average precision is the average of all precision values computed in each relevant object in the retrieved list. Given several queries, the mean average precision is the mean of average precision of each query.
- Nearest neighbor (NN): Given a query, it is the precision at the first object of the retrieved list.
- First Tier (FT): Given a query, it is the precision when C objects have been retrieved, where C is the number of relevant objects to the query.
- Second Tier (ST): Given a query, it is the precision when $2*C$ objects have been retrieved, where C is the number of relevant objects to the query.

In the retrieval experiments, each object in the collection is used as query, and subsequently we average the measures for each object to obtain the effectiveness for the entire dataset.

Regarding the descriptors, in this paper, we tested the DSR and PANORAMA descriptors. Each time we describe a mesh using these

descriptors, the input mesh is normalized in pose (rotation, translation and scale) prior to the description. As DSR is faster to compute than PANORAMA, we preferred to use DSR for presenting a detailed study of our approach. Subsequently, we use PANORAMA to validate our results.

5.2. The role of partition matching

The goal of this section is to show the contribution of the partition matching in the distance computation. Recall the definition of our distance in Eq. (1). Our distance is a linear combination between global distance (using the DSR descriptor) and partition distance. The contribution of the partition distance in the final distance depends on the parameter μ. So we conducted an experiment to measure the effect of μ in the effectiveness of the proposed distance.

We test different values for μ in the interval [0, 1] and investigate the best value according to the obtained MAP. As our objective is to evaluate only the effect of μ, we fixed the values for any other parameter (see Section 5.3 for a sensitivity analysis about parameters). Next, we show a summarized description of the parameters used in this experiment:

- For the Harris 3D algorithm, we select 200 keypoints for each object.
- For the clustering algorithm, we consider the length of the diagonal of the minimum bounding box of an object (*diag*) to define the spatial parameters: R and S. Thus, $R = 0.1 \times diag$, $S = 0.2 \times diag$, $Nm = 10$, and $Iter = 10$. Note that the R and S parameters vary for each object. The Nm and $Iter$ parameters were set empirically.
- The scale factor of the sphere radius in the patch extraction step was set to 1.
- In addition, α and β in Eq. (8) are 1.

We compare our two proposals, linear programming matching (LPM) and quadratic programming matching (QPM) with a baseline algorithm (GM), which only uses the global descriptors for retrieval (note that GM is a special case of our proposed distance when $\mu = 1$).

Table 1 shows the MAP for several values of μ, using both techniques LPM and QPM. The best result for LPM is obtained in $\mu = 0.9$ with 49.52. This value shows an improvement with respect to using only global descriptors. Note that $\mu = 1.0$ represents the GM baseline approach, as it considers a total contribution of the global descriptor distance. It is worth noting that the best MAP value for LPM is obtained through a large contribution of the global distance. In contrast, the incorporation of geometric consistency in the QPM approach does not seem to contribute to the effectiveness. Nevertheless, the shown MAP values are an average

Table 1
MAP values for different values of μ (values are in [0,100] scale).

μ	LPM	QPM
0	9.39	5.14
0.1	15.06	6.51
0.2	21.93	8.42
0.3	29.20	11.26
0.4	35.90	14.99
0.5	41.14	19.97
0.6	44.90	26.20
0.7	47.49	33.63
0.8	48.93	41.47
0.9	**49.52**	47.79
1.0	49.10	49.10

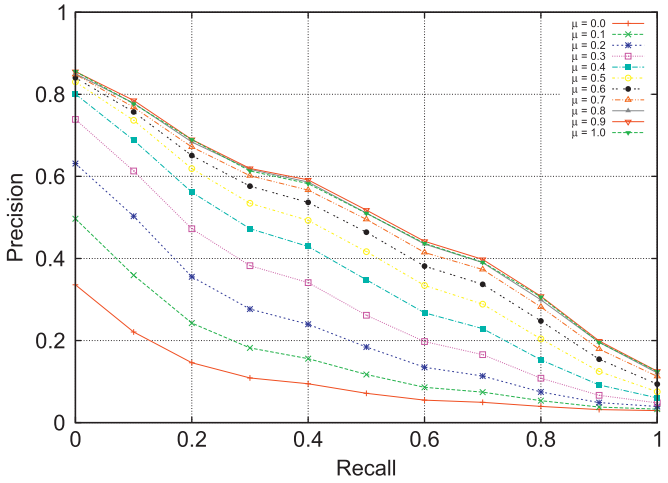

Fig. 4. Recall–precision.

of the entire dataset. This can bring up the fact that it is possible that certain classes exploit the geometric consistency. We dedicate Section 5.4 to study this situation.

We also show a recall–precision plot for the different configurations of μ (see Fig. 4). Note the improvement of our method when $\mu = 0.9$ in contrast to other values, even when global matching is used ($\mu = 1$). Moreover, the precision improvement is visible in every recall value, so it confirms the results in Table 1.

5.3. Sensitivity analysis

In this section, we evaluate several parameters of our method in order to find the best configuration. We take the finding of the previous section as a starting point. That is, all results presented in this section were computed for $\mu = 0.9$. The parameters to be evaluated are:

- *Keypoint selection*: We evaluate six configurations taken into account a fixed number of keypoints (200, 300, and 400) or a number of keypoints as a ratio of the number of vertices on the mesh (at ratios 1%, 2% and 4%). In the presented results, we use the labels *IP*-200, *IP*-300, *IP*-400, *IP*-0.01, *IP*-0.02, and *IP*-0.04, respectively.
- *Clustering parameters*: We evaluate three configurations regarding the spatial constraints of the clustering algorithm. In our experiments, we use the diagonal of the bounding box of an object (*diag*) as reference to the clustering parameters. Thus, it is possible to associate the clustering parameters with the size of an object. We define three configurations corresponding to small, medium, and large clusters. The configurations are defined as follows (in the presented results, the *diag* factor is discarded to facilitate the reading):
  - Small clusters: $R = 0.1 \times diag$, $S = 0.2 \times diag$.
  - Medium clusters: $R = 0.15 \times diag$, $S = 0.3 \times diag$.
  - Large clusters: $R = 0.2 \times diag$, $S = 0.4 \times diag$.
- *Scaling factor for partition*: We evaluate three different scaling factors for the radius of the partitioning sphere. We use 1.0, 1.25, and 1.5.

As we are interested in studying the impact of the parameters in our approach, we present results using all the aforementioned evaluation measures. In addition, we discuss that effect in the Linear Programming Matching (LPM) and Quadratic Programming Matching (QPM) separately.

Fig. 5 shows the mean average precision for LPM using all possible combinations of parameter configurations. Note that

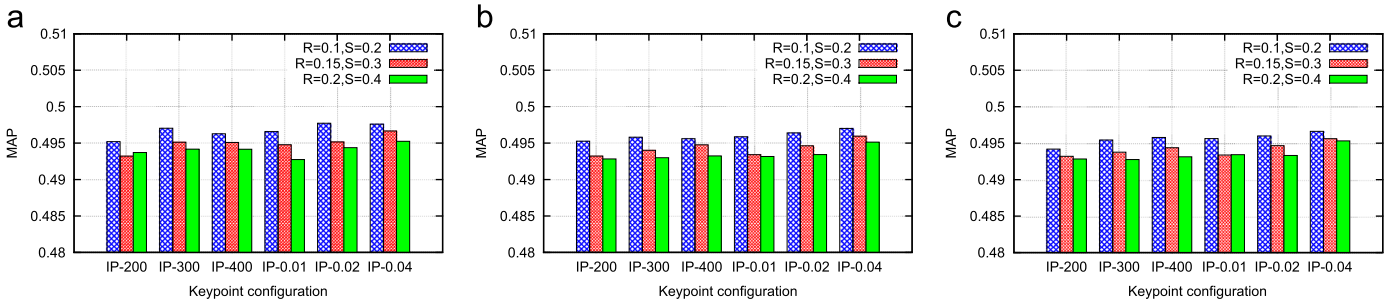

Fig. 5. Mean average precision (MAP) and sensitivity analysis on our Linear Programming Matching approach. (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

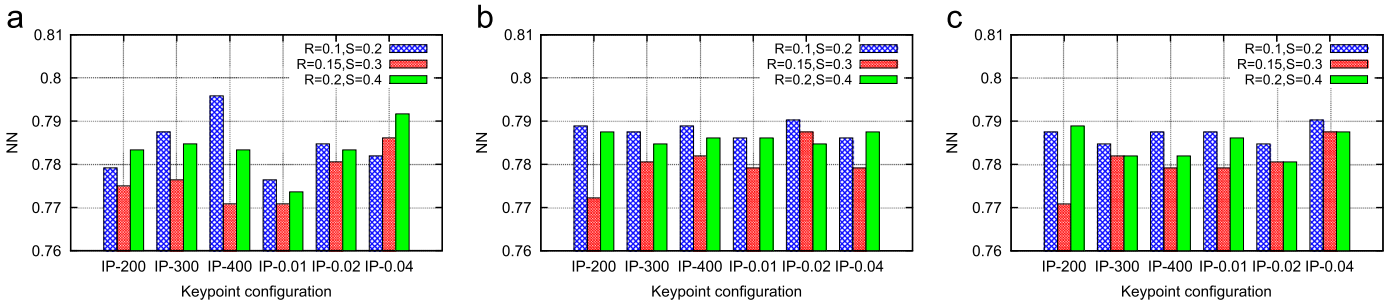

Fig. 6. Nearest neighbor (NN) and sensitivity analysis on our Linear Programming Matching (LPM) approach. (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

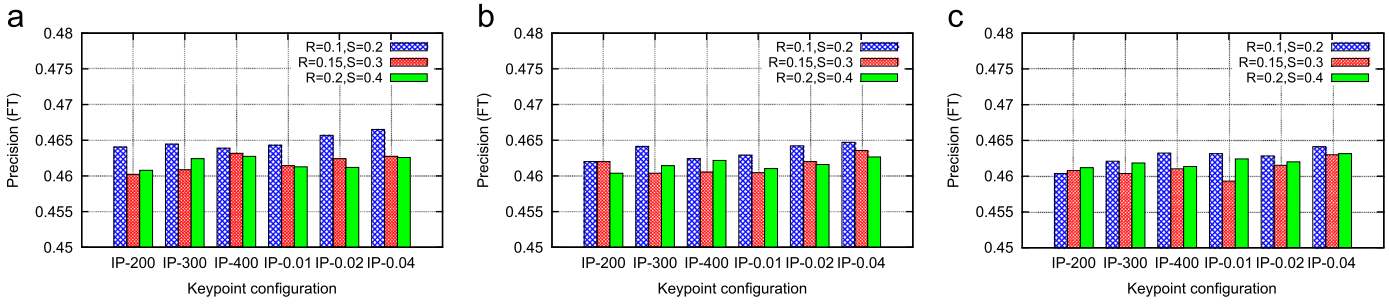

Fig. 7. First tier (FT) and sensitivity analysis on our Linear Programming Matching (LPM) approach. (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

regardless of the used keypoint selection, LPM gives better results when small clusters are used. In addition, the mean average precision decreases when the size of clusters is increased. It means that large partitions are more unstable compared to small partitions. It sounds logical, since small clusters are expected to be compact agglomerations of keypoints, and therefore the region containing them can be considered as a representative partition for an object. In contrast, large clusters allow distant keypoints to belong to the same cluster. As a result, the probability of a keypoint to belong to any cluster is high. Therefore, isolated keypoints (possibly due to noise) could be influencing the generation of poor partitions.

Interestingly, the scaling factor for partition is very related to the previous finding. The scaling factor is used to extract the partitions after the clustering algorithm. So if the scaling factor is large, the partition will be large too. Again, by looking at Fig. 5, small values of δ give the best results. Therefore, we can confirm that there is an inverse relation between partition size and effectiveness.

Another important point is that, for each plot, the best mean average precision was obtained when we selected a number of keypoints in accordance with the number of vertices. In fact, the highest MAP was 0.4977, obtained with IP-0.02, $R=0.1$, $S=0.2$, and $\delta = 1.0$. The reason to choose this in contrast to a fixed number of keypoints is evident. With a fixed number of keypoints, it is not possible to guarantee a good representation for a shape. This fact conditions the representativeness power of the keypoints, because it is likely that when we took a fixed number of keypoints per model, this amount can be large for some models and small for others. On the other hand, the adaptive alternative seems to be a good choice taking into account that an object can have an arbitrary number of vertices.

Also, we present results for nearest neighbor (NN), First Tier (FT), and Second Tier (ST) in Figs. 6, 7, and 8, respectively. The NN measure evaluates the capacity of an algorithm to retrieve a relevant object in the first position of the retrieved list. The highest value 0.7958 was obtained using a fixed number of keypoints (400), small clusters ($R=0.1$, $S=0.2$) and the smallest scaling factor (1.0). On the other hand, our results about FT and ST show a similar behavior as MAP. That is, higher values are obtained when clusters are small, and the scaling factor for a partition is small. The highest value 0.4665 for FT was obtained with IP-0.04, $R=0.1$, $S=0.2$ and $\delta = 1.0$. Similarly, the highest value 0.320507 was obtained with IP-0.02, $R=0.1$, $S=0.2$, and $\delta = 1.0$.

Regarding QPM, Figs. 9, 10, 11 and 12 show our results for MAP, NN, FT and ST, respectively. Surprisingly, the best MAP scores were obtained with medium size clusters. Moreover, unlike the mean average precision of LPM, large clusters give better results when

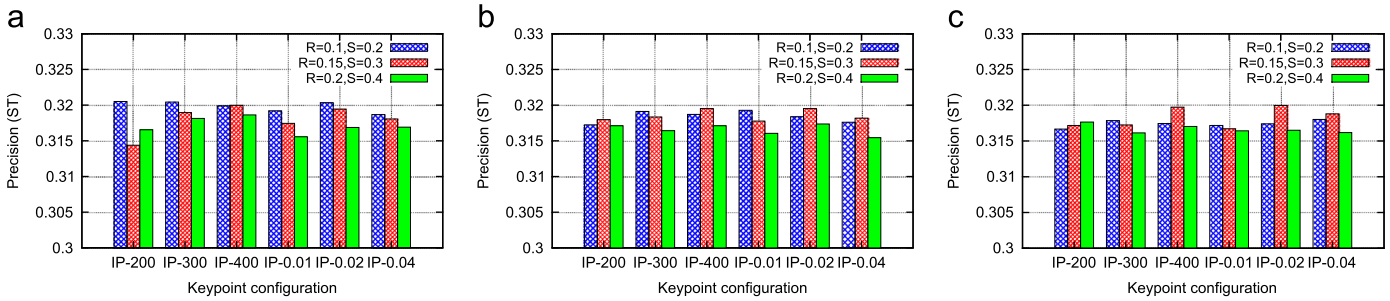

Fig. 8. Second tier (ST) and sensitivity analysis on our Linear Programming Matching (LPM) approach. (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

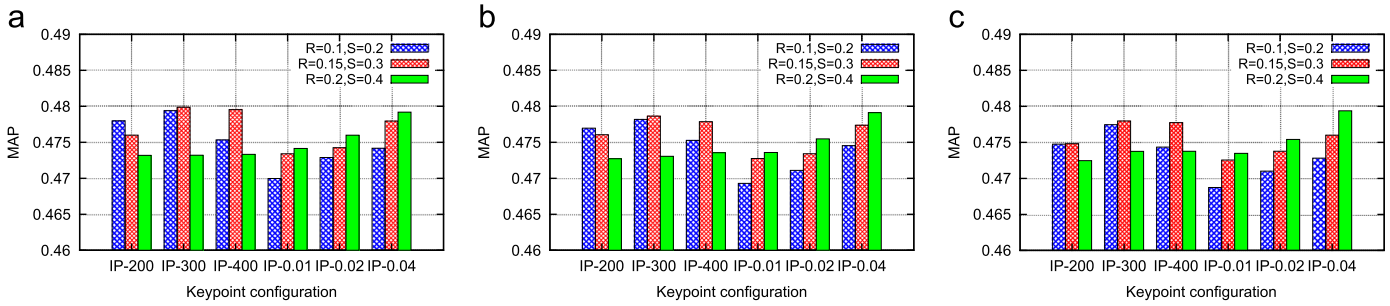

Fig. 9. Mean average precision (MAP) and sensitivity analysis on our Quadratic Programming Matching approach (QPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

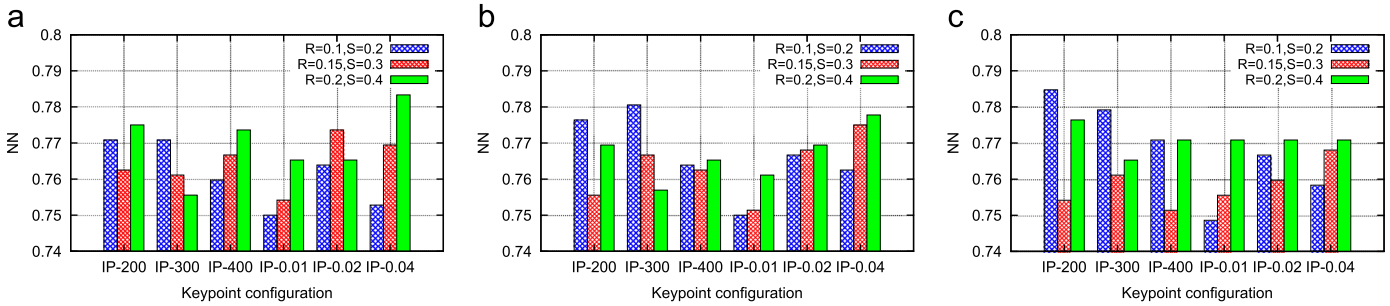

Fig. 10. Nearest neighbor (NN) and sensitivity analysis on our Quadratic Programming Matching approach (QPM). (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

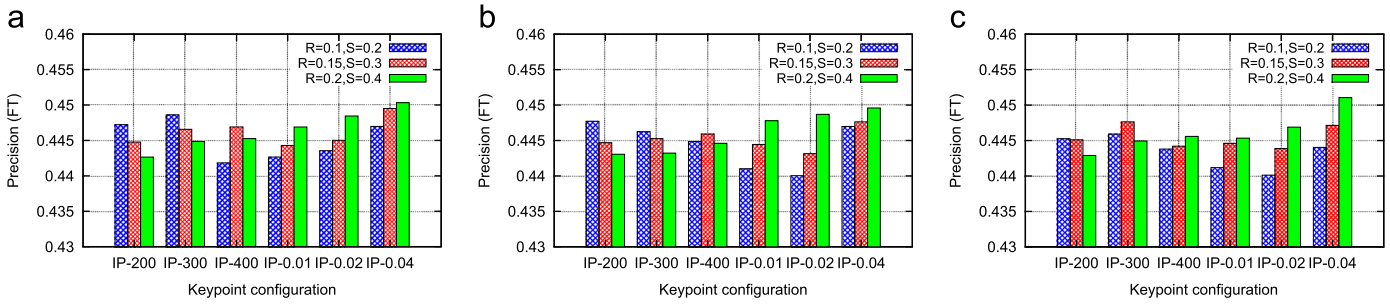

Fig. 11. First tier (FT) and sensitivity analysis on our Quadratic Programming Matching (QPM) approach. (a) $\delta = 1.0$. (b) $\delta = 1.25$. (c) $\delta = 1.5$. Plot were scaled to best visualization.

the number of keypoints depends on the number of vertices. In this connection, the configuration that delivers the largest partitions (R=0.2, S=0.4 and $\delta = 1.0$) has one of the highest MAP score. Although this situation contrasts too much with the results obtained for LPM, there is a reason for this behavior. The QPM approach depends not only on the similarity between part descriptors, but also on their geometric disposition. In fact, QPM gives the same importance to the similarity between descriptors and their consistency. In our opinion, the geometric consistency is causing this phenomenon. Our reasoning is that larger parts are more consistent in a geometrical sense than smaller parts. For instance, consider two human shapes: one with arms close to body, and other with open arms forming a T. Small partitions could characterize hands, legs, and head. Differently, large partitions could characterize upper body and lower body. So obviously the geometric consistency of upper body and lower body remains

more similar than hands, legs and head in this scenario. This situation is not uncommon in 3D datasets, and the SHREC'2009 dataset is not an exception. Therefore, in our opinion, the results we obtained exhibit the importance of considering the size of partitions as a key aspect to accomplish good effectiveness.

Similar to LPM, the First Tier (see Fig. 11) and Second Tier (see Fig. 12) in QPM exhibit an analogous behavior to its mean average precision. For the First Tier, large clusters give better results against their counterpart. Moreover, the highest FT score 0.4511 is obtained with the largest possible partition (IP-0.04, $R=0.2$, $S=0.4$, $\delta=1.5$). Likewise, using the Second Tier, the predominant scores are present either when using large clusters or when using a scaling factor greater than 1.0.

5.4. Class-by-class analysis

In this section, we show a more detailed evaluation of our approaches from the point of view of the effectiveness in each class of the dataset. The motivation to perform this evaluation is two-fold. First, all the retrieval measures used in previous experiments are a result of averaging. Average is a good way to condense a series of values. However, it can also hide valuable information in finer levels of analysis. Second, after seeing the results obtained in previous sections, our approach can be suitable depending on shape classes. So it is necessary to study the effect of our approach in each class of the dataset. Therefore, this can reveal useful information to decide when to effectively use our proposal. The results of this section were computed using the best combination found in the sensitivity analysis of Section 5.3, namely IP-0.02, $R=0.1$, $S=0.2$ and $\delta=1.0$.

Fig. 13 shows the mean average precision for each class in the SHREC'09 dataset. We divide the classes into two figures to best visualization. Each figure shows the comparison of Global Matching, Linear Programming Matching and Quadratic Programming Matching for each class as clustered bars. Our method was able to improve the effectiveness in 30 out of 40 classes. Moreover, when the objects within the same class have similar local structures and geometric consistency (such as in bookshelf, bird, apartment and skyscrape), the QPM approach outperforms the global matching and LPM.

Also, note the existence of 10 classes (single house, chair, round table, quadruped, mug, floor lamp, desk lamp, sword, biplane and bicycle) where it was not possible to improve the effectiveness with any of our approaches. However, it is also worth noting that in general, all of these 10 classes share a characteristic: the high variability of objects within the same class not only in a global sense, but also in a local sense. To illustrate this point, let us take as example the class *Chair*, on which our approaches did not improve. In our opinion, it is due to the high variability in the global sense. Moreover, shape parts also have a high variability (see Fig. 14). As a result, the keypoints can be concentrated in different parts of the models, as each object can contain distinctive local features not repeatable in its class. Therefore, LPM and QPM cannot take advantage of the partitioning technique proposed in this paper. Consequently, this can be cause for the moderate improvement of our method with respect to global matching. However, we believe that similar situations could influence the effectiveness of any algorithm.

The found evidence allows us to state the strengths and limitations of our approach. On one hand, our method can improve the effectiveness in classes that share local information. That is, when models from the same class have common and similar parts, an improvement is expected. On the other hand, our approaches cannot deal with extreme variability of parts between objects within the same class.

5.4.1. Correlation analysis

In this section, we investigate the possible relationships between several factors that affect the effectiveness of our proposed methods. To do so, we use a correlation analysis and a statistical significance study among eight variables defined in the following, also introducing the abbreviations for each variable to be used in the analysis:

- Number of partitions (NP).
- Number of vertices (NV).
- MAP for global matching (GM).
- MAP for LPM (LPM).
- MAP for QPM (QPM).
- MAP gain of LPM over GM (G1).
- MAP gain of QPM over GM (G2).
- MAP gain of QPM over LPM (G3).

The last three variables were obtained by computing the difference of MAP scores of the involved methods. To obtain the data in this experiment, we computed the eight values using each model in the collection as a query. Therefore, we obtained eight values for each model, and subsequently we used all that information to compute the correlation matrix shown in Table 2. In addition, we computed the p-values for testing the hypothesis of no correlation. So for each correlation value, we have a p-value indicating the statistical significance of that correlation. We assume p-values <0.05 as significant. The matrix of p-values is shown in Table 3.

The information provided by the correlation matrix and the p-values allow us to verify some aspects observed in the previous experiments. For instance, there is a high correlation between the number of partitions and the three gain measures, namely G1, G2 and G3. First, the correlation between the number of partitions and G1 (MAP gain of LPM over GM) is positive. So the greater the number of partitions, the higher the improvement of LPM over GM. Second, the correlation between the number of partitions and G2 and G3 (MAP gain of QPM over GM and LPM, respectively) is negative. So it

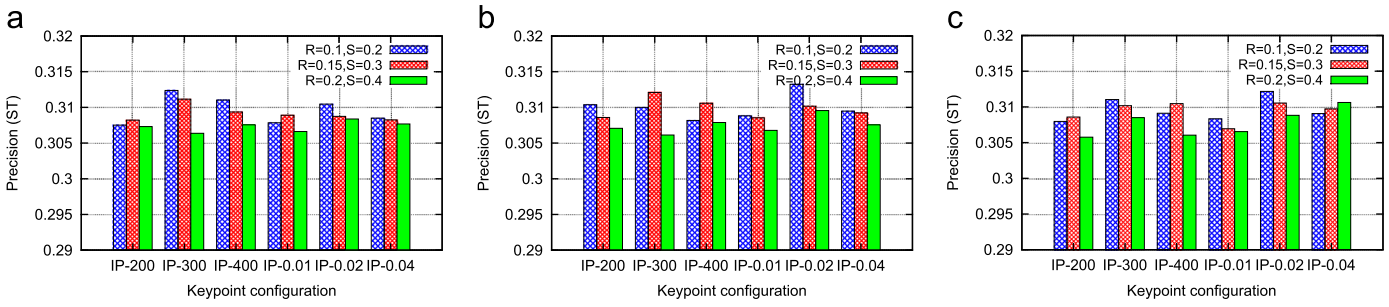

Fig. 12. Second tier (ST) and sensitivity analysis on our Quadratic Programming Matching (QPM) approach. (a) $\delta=1.0$. (b) $\delta=1.25$. (c) $\delta=1.5$. Plot were scaled to best visualization.

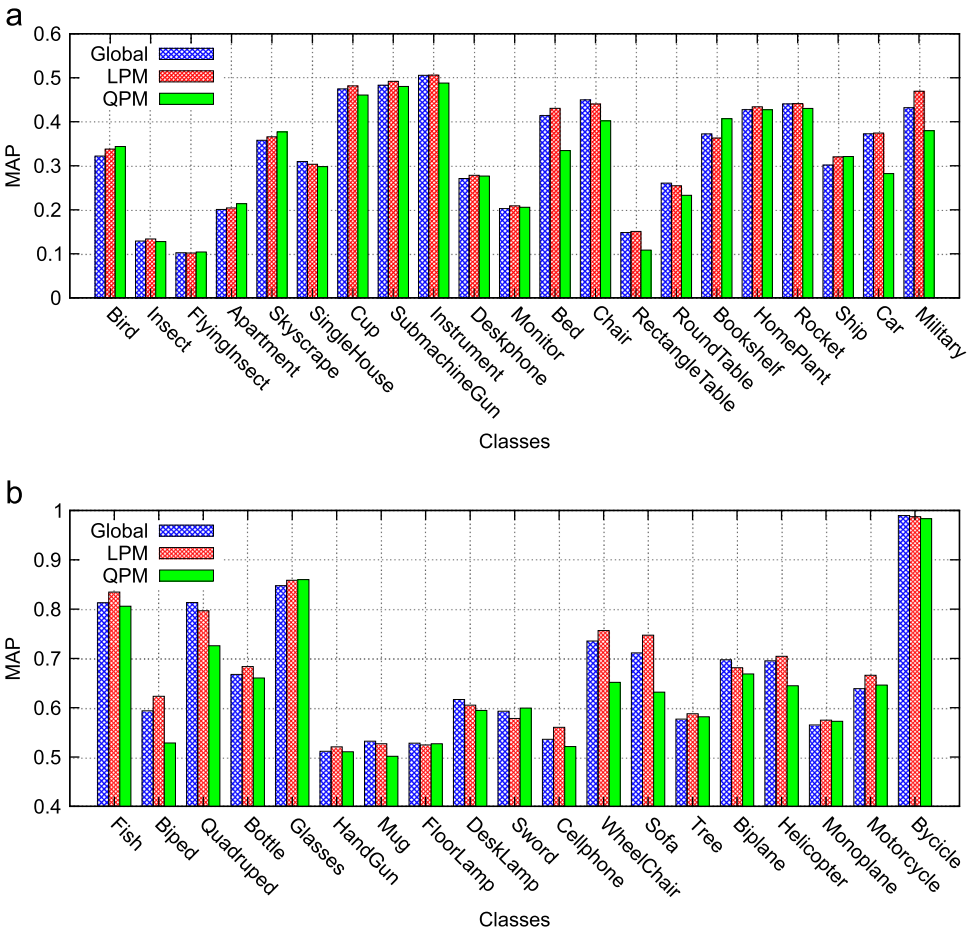

Fig. 13. Mean average precision for each class in the SHREC'09 dataset. Plots were scaled for better visualization.

Fig. 14. Samples of class *Chair*. Note the high variability of parts amongst shapes.

Table 2
Correlation matrix between eight variables: Number of partitions (NP), number of vertices (NV), MAP for GM (GM), MAP for LPM (LPM), MAP for QPM (QPM), MAP gain for LPM over GM (G1), MAP gain for QPM over GM (G2), and MAP gain for QPM over LPM (G3).

Variables	NP	NV	GM	LPM	QPM	G1	G2	G3
NP	1.0000	0.0323	−0.0131	−0.0015	−0.0696	0.1318	−0.2658	−0.3145
NV	0.0323	1.0000	0.0361	0.0431	0.0361	0.0815	−0.0039	−0.0377
GM	−0.0131	0.0361	1.0000	0.9962	0.9784	0.0149	−0.2119	−0.2132
LPM	−0.0015	0.0431	0.9962	1.0000	0.9776	0.1022	−0.1972	−0.2351
QPM	−0.0696	0.0361	0.9784	0.9776	1.0000	0.0484	−0.0055	−0.0254
G1	0.1318	0.0815	0.0149	0.1022	0.0484	1.0000	0.1566	−0.2625
G2	−0.2658	−0.0039	−0.2119	−0.1972	−0.0055	0.1566	1.0000	0.9119
G3	−0.3145	−0.0377	−0.2132	−0.2351	−0.0254	−0.2625	0.9119	1.0000

means that QPM benefits from fewer partitions. These two situations are consistent with the findings of Section 5.3 when we showed that QPM presents better results with large partitions. So now we can conclude that QPM is suitable for matching few large partitions. In contrast, LPM can perform a good matching regardless of the geometric consistency, just by considering more small partitions. It is expected that the small partitions belong to distinctive features of the objects. In summary, we have shown the importance of the number of partitions and their size in the improvement of the retrieval effectiveness of our technique.

Regarding the number of vertices of the meshes, there is a useful correlation which deserves attention. The number of

Table 3
Matrix of p-values for the correlation between eight variables: number of partitions (NP), number of vertices (NV), MAP for GM (GM), MAP for LPM (LPM), MAP for QPM (QPM), MAP gain for LPM over GM (G1), MAP gain for QPM over GM (G2), and MAP gain for QPM over LPM (G3).

Variables	NP	NV	GM	LPM	QPM	G1	G2	G3
NP	1.0000	0.3875	0.7262	0.9682	0.0621	0.0004	0.0000	0.0000
NV	0.3875	1.0000	0.3329	0.2484	0.3329	0.0287	0.9160	0.3121
GM	0.7262	0.3329	1.0000	0.0000	0.0000	0.6895	0.0000	0.0000
LPM	0.9682	0.2484	0.0000	1.0000	0.0000	0.0060	0.0000	0.0000
QPM	0.0621	0.3329	0.0000	0.0000	1.0000	0.1949	0.8836	0.4957
G1	0.0004	0.0287	0.6895	0.0060	0.1949	1.0000	0.0000	0.0000
G2	0.0000	0.9160	0.0000	0.0000	0.8836	0.0000	1.0000	0.0000
G3	0.0000	0.3121	0.0000	0.0000	0.4957	0.0000	0.0000	1.0000

Table 4
Results for different values of μ in LPM using PANORAMA (values are in [0,100] scale).

μ	NN	FT	ST	MAP
0	42.0833	21.2337	15.3064	20.4128
0.1	60.6944	31.3889	21.8709	31.3174
0.2	74.5833	40.6536	28.6029	42.286
0.3	81.8056	48.268	33.7908	50.9521
0.4	85.4167	54.1912	38.701	57.3621
0.5	88.3333	57.8758	41.393	61.7762
0.6	88.8889	60.3513	43.1822	64.4322
0.7	88.4722	61.585	44.375	65.9167
0.8	88.8889	62.165	44.951	66.6028
0.9	**89.0278**	**62.3366**	**45.3023**	**66.813**
1.0	89.0278	61.9853	44.7917	66.7291

vertices is highly correlated with the MAP gain of LPM over GM (G1). In other words, as the correlation is positive, we can say that LPM benefits from meshes with a large number of vertices. It reveals a remarkable connection with the previous analysis. With large number of vertices, one can expect meshes with more detail, and hence they can contain rich features. Moreover, if we prefer to select small clusters, the resulting partitions will be distinctive and small. In addition, we can obtain many partitions since the number of keypoints could depend on the number of vertices. Finally, following our previous analysis, LPM obtains better effectiveness when the input is a set of many distinctive partitions as a product of meshes with many vertices.

It is also worth noting the dependency of our approaches (LPM and QPM) on the global matching. This can be evidenced in the high correlation between GM and both LPM and QPM. Obviously, this fact is in accordance with the use of $\mu = 0.9$ which is associated to a large contribution of the global matching in the final distance computation.

5.4.2. Results with PANORAMA

In this section, we present the results of our method using the PANORAMA descriptor [14]. For this experiment, we used the best parameter configuration as shown in Section 5.3. Table 4 shows the results by varying the contribution of the part matching (parameter μ) in the LPM technique. Similar to a previous experiment (see Section 5.2), we obtained the best results when $\mu = 0.9$. This result validates our argument about the contribution of the partition matching in the effectiveness of generic shape retrieval.

6. Conclusions

In this paper, we presented a shape retrieval method that combines global descriptors and part-based descriptors. We proposed a method for determining data-adaptive partition from meshes. Partitions were derived from agglomerations of distinctive keypoints on shapes. Finally, matching between partitions was stated as an integer program in order to compute correspondences.

From our experiments, it is possible to say that partition matching contributes to improving the retrieval effectiveness. Our method was able to achieve significant improvements in classes with objects containing common distinctive parts. In contrast, there is a limitation when objects within a class do not share common distinctive parts. Therefore, the partition matching degrades the effectiveness of global descriptors instead of improving it. Nevertheless, we believe that our approach partially attenuated this limitation with its ability to determine characteristic partitions. In our opinion, our method offers new representational capabilities for 3D shapes which have proven to be effective in conjunction with global descriptors. In addition, we found a high correlation between the achieved effectiveness and the partitions provided by our method. Specifically, the number and size of the partitions play an important role for defining an effective similarity measure. This is because these two factors are well related to the quality of partitions (and their distinctiveness) and therefore, they influence the overall performance.

In our opinion, the use of high-level local structures in 3D shape retrieval is a promising research direction. Moreover, the use of information in higher levels of abstraction (for instance, functionality) should benefit the definition of more effective similarity models.

Acknowledgments

This research has been partially funded by CONICYT (Chile) through the Doctoral Scholarship, and FONDECYT (Chile) Project 1110111. The work of Tobias Schreck was supported by EC FP7 STREP Project PRESIOUS, Grant no. 600533.

References

[1] Keim DA. Efficient geometry-based similarity search of 3D spatial databases. In: Delis A, Faloutsos C, Ghandeharizadeh S, editors. Proceedings of the ACM international conference on management of data (SIGMOD). ACM Press; 1999. p. 419–30 ISBN 1-58113-084-8.
[2] Atmosukarto I, Wilamowska K, Heike C, Shapiro LG. 3D object classification using salient point patterns with application to craniofacial research. Pattern Recognition 2010;43(4):1502–17.
[3] You CF, Tsai YL. 3D solid model retrieval for engineering reuse based on local feature correspondence. Int J Adv Manuf Technol 2009;46(5–8):649–61.
[4] Bustos B, Keim D, Saupe D, Schreck T, Vranić D. An experimental effectiveness comparison of methods for 3D similarity search. Int J Digital Libr 2006;6 (1):39–54 (Special issue on Multimedia Contents and Management in Digital Libraries).
[5] Bustos B, Keim DA, Saupe D, Schreck T, Vranic D. Automatic selection and combination of descriptors for effective 3D similarity search. In: Proceedings of the international symposium on multimedia software engineering, 2004.
[6] Vranic DV. DESIRE: a composite 3D-shape descriptor. In: Proceedings of the IEEE international conference on multimedia and expo, 2005.
[7] Papadakis P, Pratikakis I, Theoharis T, Passalis G, Perantonis SJ. 3D object retrieval using an efficient and compact hybrid shape descriptor. In: Perantonis SJ, Sapidis N, Spagnuolo M, Thalmann D, editors. Proceedings of the workshop on 3D object retrieval (3DOR),. Eurographics Association; 2008. p. 9–16 ISBN 978-3-905674-05-7.
[8] Li B, Johan H. 3D model retrieval using global and local radial distances. In: Proceedings of the international workshop on advanced image technology, 2010.
[9] Bustos B, Schreck T, Walter M, Barrios JM, Schaefer M, Keim DA. Improving 3D similarity search by enhancing and combining 3D descriptors. Multimedia Tools Appl 2011;58(1):81–108.
[10] Schreck T, Scherer M, Walter M, Bustos B, Yoon SM, Kuijper A. Graph-based combinations of fragment descriptors for improved 3D object retrieval. In: Proceedings of the ACM multimedia systems, 2012.
[11] Bustos B, Keim DA, Saupe D, Schreck T, Vranic DV. Feature-based similarity search in 3D object databases. ACM Comput Surv 2005;37(4):345–87.
[12] Tangelder JWH, Veltkamp RC. A survey of content based 3D shape retrieval methods. Multimedia Tools Appl 2008;39(3):441–71.

[13] Vranic D. 3D model retrieval. PhD thesis. University of Leipzig; 2004.
[14] Papadakis P, Pratikakis I, Theoharis T, Perantonis S. PANORAMA: a 3D shape descriptor based on panoramic views for unsupervised 3D object retrieval. Int J Comput Vision 2009;89(2–3):177–92.
[15] Osada R, Funkhouser TA, Chazelle B, Dobkin DP. Shape distributions. ACM Trans Graph 2002;21(4):807–32.
[16] Kazhdan M, Funkhouser T, Rusinkiewicz S. Rotation invariant spherical harmonic representation of 3D shape descriptors. Proceedings of the Eurographics Symposium on Geometry Processing (SGP'03). Eurographics Association. 2003.
[17] Toldo R, Castellani U, Fusiello A. Visual vocabulary signature for 3D object retrieval and partial matching. In: Spagnuolo M, Pratikakis I, Veltkamp RC, Theoharis T, editors. 3DOR. Eurographics Association; 2009. p. 21–8, ISBN 978-3-905674-16-3.
[18] Shapira L, Shalom S, Shamir A, Cohen-Or D, Zhang H. Contextual part analogies in 3D objects. Int J Comput Vision 2010;89(2–3):309–26, http://dx.doi.org/10.1007/s11263-009-0279-0.
[19] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis Comput 2008;24(4):249–59, http://dx.doi.org/10.1007/s00371-007-0197-5.
[20] Litman R, Bronstein AM, Bronstein MM. Diffusion-geometric maximally stable component detection in deformable shapes. Comput Graph 2011;35(3):549–60.
[21] Sipiran I, Bustos B. Key-component detection on 3D meshes using local features. In: Spagnuolo M, Bronstein MM, Bronstein AM, Ferreira A, editors. 3DOR. Eurographics Association; 2012. p. 25–32 ISBN 978-3-905674-36-1.
[22] Sipiran I, Bustos B. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. Vis Comput 2011;27:963–76.
[23] Bronstein AM, Bronstein MM, Bustos B, Castellani U, Crisani M, Falcidieno B, et al., SHREC 2010: Robust feature detection and description benchmark. In: Proceedings of the workshop on 3D object retrieval (3DOR'10). Eurographics Association; 2010.
[24] Dutagaci H, Cheung C, Godil A. Evaluation of 3D interest point detection techniques via human-generated ground truth. Visual Comput 2012;28:901–17, http://dx.doi.org/10.1007/s00371-012-0746-4.
[25] Johnson AE, Hebert M. Control of polygonal mesh resolution for 3-d computer vision. Graphical Models Image Process 1998;60(4):261–85, http://dx.doi.org/10.1006/gmip.1998.0474.
[26] Leow WK, Li R. The analysis and applications of adaptive-binning color histograms. Comput Vis Image Underst 2004;94:67–91, http://dx.doi.org/10.1016/j.cviu.2003.10.010.
[27] Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: International conference on knowledge discovery and data mining, 1996. p. 226–31.
[28] Matoušek J, Sharir M, Welzl E. A subexponential bound for linear programming. In: Proceedings of the eighth annual symposium on computational geometry. SCG '92. New York, NY, USA: ACM; 1992. p. 1–8. http://dx.doi.org/10.1145/142675.142678, ISBN 0-89791-517-8.
[29] Wolsey L. Integer programming. Wiley-interscience series in discrete mathematics and optimization. Wiley; 1998 ISBN 9780471283669.
[30] Bemporad A, Mignone D, Morari M. An efficient branch and bound algorithm for state estimation and control of hybrid systems. In: European control conference, Karlsruhe, Germany, 1999.
[31] Godil A, Dutagaci H, Akgül CB, Axenopoulos A, Bustos B, Chaouch M, et al. SHREC'09 track: generic shape retrieval. In: Spagnuolo M, Pratikakis I, Veltkamp RC, Theoharis T, editors. Proceedings of the workshop on 3D object retrieval (3DOR). Eurographics Association; 2009. p. 61–8 ISBN 978-3-905674-16-3.