Eurographics Workshop on 3D Object Retrieval (2015)
I. Pratikakis, M. Spagnuolo, T. Theoharis, L. Van Gool, and R. Veltkamp (Editors)

Randomized Sub-Volume Partitioning for Part-Based 3D Model Retrieval

T. Furuya, S. Kurabe, and R. Ohbuchi

University of Yamanashi, Japan

Abstract

Given a query that specifies partial shape, a Part-based 3D Model Retrieval (P3DMR) system would retrieve 3D models whose part(s) matches the query. Computationally, this is quite challenging; the query must be compared against parts of 3D models having unknown position, orientation, and scale. To our knowledge, no algorithm can perform P3DMR on a database having significant size (e.g., 100K 3D models) that includes polygon soup and other not-so-well-defined shape representations. In this paper, we propose a scalable P3DMR algorithm called Part-based 3D model retrieval by Randomized Sub-Volume Partitioning, or P3D-RSVP. To match a partial query with a set of (whole) 3D models in the database, P3D-RSVP iteratively partitions a 3D model into a set of sub-volumes by using 3D grids having randomized intervals and orientations. To quickly compare the query with all the sub-volumes of all the models in the database, P3D-RSVP hashes high dimensional features into compact binary codes. Quantitative evaluation using several benchmarks shows that the P3D-RSVP is able to query a 50K model database in 2 seconds.

Categories and Subject Descriptors (according to ACM CCS): H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Retrieval models.

1. Introduction

Majority of shape-based 3D model retrieval (3DMR) algorithms focuses on Whole-based 3D model retrieval (W3DMR) where a whole, or entire shape is given as a query to the retrieval system. For certain applications and databases, some W3DMR algorithms succeed in producing sufficient retrieval accuracies. Contrary to W3DMR, Part-based 3D model retrieval (P3DMR), in which a system would retrieve 3D models whose part(s) matches a query, has not been studied well in the literature. P3DMR is important for a number of applications, e.g., design and development process for industrial products.

We think that a challenge central to P3DMR is computational efficiency. Out of a very large number of possibilities, we need to find part(s) in 3D models which matches a part-based query. We don't know which model(s) in a database contains a shape specified by the part-based query. In addition, we don't know, a priori, position, scale, and orientation of the desired partial shape of the query in a 3D model.

A straightforward approach to P3DMR is sliding sub-volume search [KHK10][SX14], which is a 3D extension of the sliding-window algorithm used for object detection in 2D images [RBK96][DT05]. In the sliding sub-volume search algorithm, the query is compared against numerous sub-volumes extracted at every position, with diverse scales and orientations, of a 3D model or a 3D scene. However, this approach would become impractical for a database having a large number (e.g., 100K) of 3D models. Assume that locations of sub-volumes are sampled at every intersection of a 3D grid having N_g intervals, and, at each location, sub-volumes are sampled at N_s scales and N_o orientations. The total number of sub-volumes for a database having N_m 3D models is $N_m \times N_g \times N_g \times N_g \times N_s \times N_o$. If we sample sub-volumes using N_g=10, N_s=10, and N_o=10, and if we have N_m=100K 3D models in the database, 10G sub-volumes must be compared against a query. Then, in terms of both storage for features of sub-volumes and time for comparing the sub-volumes with the query, the sliding sub-volume approach would be impractical, at least for an "interactive" retrieval.

To reduce the cost for P3DMR, [FMA*10] and [SYY*05] proposed P3DMR algorithms that employ segmentation of retrieval target 3D models. [FMA*10] and [SYY*05] segment a 3D model in a database into a set of sub-parts. Each sub-part is described by a feature and is compared against a feature of the part-based query. To accelerate retrieval, [FMA*10] employed inverted index. However, this approach requires a large amount of memory for indexing features of a large number of 3D models. Furthermore, improper segmentation of 3D models could lead to low retrieval accuracy.

In this paper, we propose *Part-based 3D model retrieval by Randomized Sub-Volume Partitioning* (*P3D-RSVP*) algorithm for P3DMR applicable to a 3D model database of

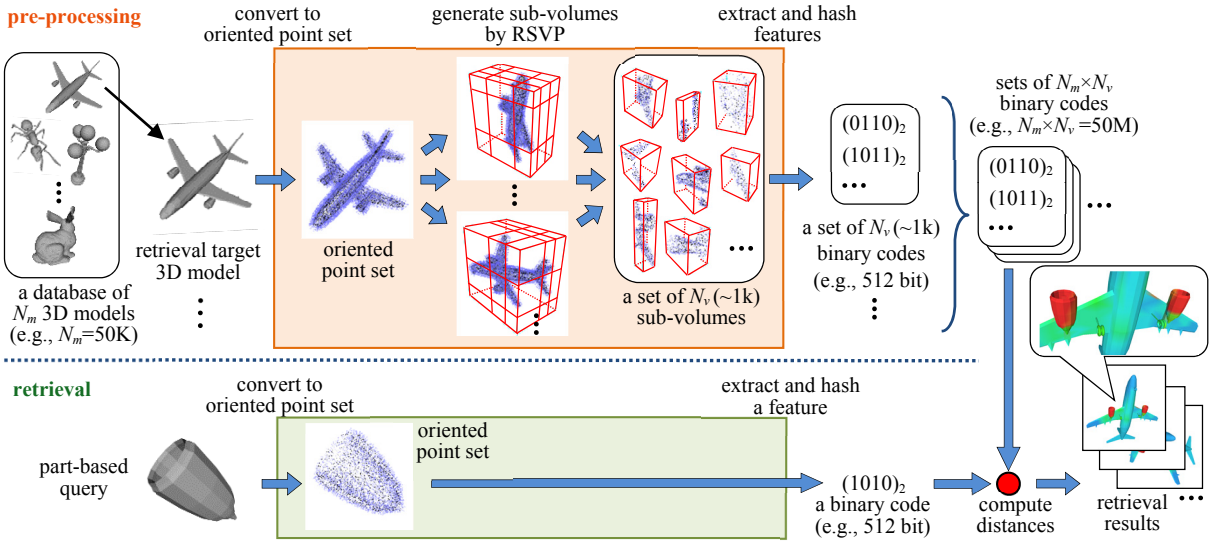


Figure 1: *For pre-processing, each 3D model in a database is partitioned into a set of N_v sub-volumes having diverse positions, scales, and orientations by using 3D grids with random intervals and orientations. Each sub-volume is described by a compact binary code. Retrieval rank is efficiently computed by comparing binary codes of the sub-volumes against a binary code of the part-based query.*

significant size. P3D-RSVP (see Figure 1) randomly and iteratively partitions a retrieval target 3D model into a set of sub-volumes by using 3D grids having random intervals and orientations. By randomizing parameters for the sub-volumes, i.e., their positions, scales, and orientations, the P3D-RSVP generates much smaller number of sub-volumes (e.g., ~1K per model) than the sliding sub-volume approach (e.g., ~100K per model). However, the number of sub-volumes that need to be compared is still quite large if the size of the database is very significant.

To reduce the cost for retrieval, we use compact binary codes to represent features for the queries and the sub-volumes of 3D models. For each sub-volume, 3D geometric features are extracted, encoded, and aggregated. Then, the aggregated high-dimensional feature is dimension-reduced and hashed into a compact binary code by Iterative Quantization (ITQ) [GL11]. Comparison among binary codes can be performed very efficiently in Hamming space.

To quantitatively evaluate efficiency and accuracy of the P3D-RSVP algorithm, we create three new benchmark databases for P3DMR. Experimental results using these benchmark databases show that the P3D-RSVP algorithm produces higher retrieval accuracy than algorithms we have compared against. In addition, it is efficient; it takes less than 2 seconds to query a database having 50K 3D models.

Contribution of this paper can be summarized as follows;

- Proposal of the P3D-RSVP algorithm. It combines randomized sub-volume partitioning of 3D model and compact binary codes to describe the sub-volumes for efficient comparison among the query and 3D models.
- Quantitative evaluation of efficiency and accuracy of the P3D-RSVP algorithm using newly created P3DMR benchmark databases.

The rest of the paper is organized as follows. The next section reviews related work. The P3D-RSVP algorithm is described in Section 3. Experiments and results are presented in Section 4, followed by conclusion and future work in Section 5.

2. Related work

P3DMR algorithms search 3D models whose part(s) matches a part-based query [FMA*10, IG07, AMS*11, SYY*05, SSS*08, LBZ*13]. It is a more difficult problem than W3DMR, primarily due to high computational cost to compare the query against numerous parts of 3D models.

[SPK*13], [SPS14], and [LZQ06] applied W3DMR algorithms using Bag-of-Features (BF) approach [CDF*04] to the task of P3DMR. An algorithm in this class extracts a set of local features from each of the part-based query model and the (whole) 3D model in a database. Then, these sets of features are aggregated by using BF approach and compared against each other. This process works reasonably well for P3DMR if the part-based query is "large enough", e.g., more than 50% of the whole model. However, if the query is "small" relative to the whole model, feature comparison would fail since BF-aggregated vectors for the query and retrieval targets would become significantly different. Therefore, this class of P3DMR algorithms using W3DMR approaches showed limited success.

[FMA*10], [IG07], [SYY*05], and [SSS*08] proposed P3DMR algorithms based on segmentation of retrieval target 3D models. [FMA*10] and [SYY*05] segment the 3D model in the database into a set of sub-parts. Each sub-part is described by a 3D geometric feature and compared against a 3D geometric feature of the query. To accelerate retrieval, [FMA*10] exploited an inverted index. The inverted index works quite well, that is, it took about a second to query the database containing about 800 3D mechanical CAD models. [SSS*08] hierarchically segments a 3D manifold mesh and constructs a graph whose node corresponds to a sub-volume. Then, a bipartite graph matching is performed to compare the query and the

retrieval target 3D model. This class of algorithms using segmentation can perform P3DMR within several seconds if size of a database is small (~1K 3D models). However, they don't scale to a database having significant size (~100K) due to high computational cost for similarity matching among the query and the 3D models. Furthermore, segmentation algorithms often can't handle 3D shapes having topologically not-so-well-defined representations, e.g., polygon soup or point set captured by 3D range scanners.

[MBO10], [KHK10], and [SX14] proposed algorithms to detect object(s) similar to the query from a 3D scene. [KHK10] and [SX14] perform sliding sub-volume search for object detection from a 3D scene. Sub-volumes having diverse scales are generated at every position in the 3D scene, and features extracted from these sub-volumes are compared against a feature of the query. These algorithms effectively identify parts similar in shape to the query from a *single 3D scene*. However, due to their high computational costs, these algorithms are not practical in finding a desired partial shape from a large collection of 3D models or 3D scenes.

Recently, [JMY12] proposed Randomized Sub-Window Partitioning (RSWP) approach for efficient part-based 2D image retrieval. By randomizing position and scale of sub-windows, the algorithm requires smaller number of sub-windows than the sliding window approach. According to [JMY12], the RSWP is faster than the branch-and-bound methods [Lam09] for part-based 2D image retrieval. Our proposed algorithm is an extension of the RSWP to P3DMR.

3. Proposed algorithm

3.1 Overview of the algorithm

We propose Part-based 3D model retrieval by Randomized Sub-Volume Partitioning (P3D-RSVP) algorithm for efficient P3DMR on a large-scale 3D model database. P3D-RSVP (Figure 1) randomly and iteratively partitions a retrieval target 3D model into a set of sub-volumes by using 3D grids with random intervals and orientations. Since total number of sub-volumes per database having significant size becomes very large, comparison between features of the part-based query and features of the sub-volumes must be very efficient. In addition, each sub-volume feature must be compact so that all the features for all the sub-volumes of all the 3D models in the database could be stored on memory.

To reduce the cost of retrieval, high-dimensional real-valued vectors extracted from sub-volumes are hashed into compact (e.g., 512 bit) binary codes. A binary code for a part-based query is efficiently compared against the binary codes for the sub-volumes of 3D models in the database by using Hamming distance. In addition to being fast to compare, binary code is compact enough so that features for all the sub-volumes of all the (e.g., 100K) 3D models in a database could be held on memory for quick retrieval.

3.2 Randomized Sub-Volume Partitioning (RSVP)

3.2.1 Generating oriented point set

The P3D-RSVP algorithm expects 3D model defined either as polygonal mesh or as oriented point set. For a 3D model defined as polygonal mesh, we first convert it into oriented point set by sampling its surfaces. We use the algorithm by Osada et al. [OFC*02] for converting a polygonal model into an oriented point set. The algorithm randomly and uniformly samples points on the surfaces of the 3D polygonal model by using quasi-random number sequence. Each point is associated with the normal vector of the triangle on which the point is sampled. In this paper, we sample N_p=4,000 oriented points on a part-based query model and N_p=16,000 oriented points on a retrieval target 3D model. Oriented point representation of the 3D model is scaled to fit a sphere having diameter 1.

3.2.2 Generating sub-volumes

Given an oriented point set of the retrieval target 3D model, we partition the bounding box of the point set into $N_g \times N_g \times N_g$ (e.g., N_g=3) non-overlapping cuboid sub-volumes by using 3D grid having random intervals. We iterate this partitioning N_i times (e.g., N_i=50). To make the algorithm robust against orientation in 3D space, we randomly rotate the point set prior to each partitioning. We set the lower bound for the 3D grid interval to be 0.05 since a sub-volume that is too small would be useless for comparison between the query and the sub-volumes of 3D models.

As a result of the partitioning, $N_v = N_g \times N_g \times N_g \times N_i$ sub-volumes having diverse positions, sizes and aspect ratios, and orientations are generated per 3D model. By increasing N_g, sub-volumes with smaller scale tend to be generated. Small sub-volumes are necessary if a part-based query is small relative to the target whole 3D model (e.g., querying by a 3D model of a hand to retrieve whole-body human 3D models). By increasing N_i, positions, scales, and orientations for sub-volumes get more diverse. This increases probability to match sub-volumes to the part-based query. We obtain $N_{v\_total} = N_v \times N_m$ subvolumes per database where N_m is the number of 3D models in the database.

3.3 Features for sub-volumes

3.3.1 Extracting sub-volume features

Each sub-volume generated by random grid partitioning is described by a feature vector and is compared against a feature vector of the part-based query. In this paper, we use a feature called *Super Vector of Simplified Point Feature Histograms*, or *SV-SPFH*, to describe the sub-volumes.

A simplistic approach to computing SV-SPFH is as follows. First, a set of oriented points in a sub-volume is first normalized for scales so that it is tightly enclosed by a sphere having diameter 1. For each oriented point, we define a *Sphere-Of-Interest* (*SOI*) whose radius is R within which local 3D geometric feature is computed (R is a parameter). For example, with R=0.5, the SOI is large enough to enclose the sub-volume. SPFH [RBB09], which is rotationally invariant 125-dimensional 3D geometric feature, are densely extracted from each of the oriented points enclosed in the SOI. We then aggregate the set of SPFH features into a feature vector per sub-volume by using Super Vector (SV) encoding [ZYZ*10] to generate a SV-SPFH for the sub-volume. Similarly, we compute a SV-SPFH feature vector for the query.

Using this simplistic approach, temporal cost of computing SV-SPFH for a large number of sub-volumes is quite high. To accelerate extraction of SV-SPFH features from sub-volumes, we actually use an accelerated version of that algorithm that employs a "late-binding" approach. We move the process of extracting and encoding sets of SPFH features to the stage before sub-volume partitioning. After the sub-volume partitioning is done, a SV-SPFH feature for a sub-volume is efficiently computed by simply pooling, or summing, already encoded features that lie inside the sub-volume. Using this late-binding approach, whose detail will be described below, SV-SPFH feature extraction from thousands of sub-volumes of a 3D model became 3 to 4 times faster than the simplistic approach.

Extracting SPFH: Given an oriented point set of a 3D model, which has not been partitioned yet, a set of SPFH is densely extracted with respect to both spatial position and scale. Each SPFH feature is computed at the 3D coordinates of one of N_p oriented points, where N_p=16,000 is used throughout the paper. We don't know the scale of each sub-volume prior to random grid partitioning. Thus, we extract a set of SPFH that covers all the N_s scales at each SPFH center. In this paper, we used number of scales N_s=20. For each oriented point **p** of the 3D model, we define N_s SOIs. Radius R_s for s-th (s=1,…,N_s) scale is computed as $R_s = sR / N_s$. A SPFH is extracted from each of N_s SOI having radius R_s. As we use N_p=16,000 and N_s=20, $N_p \times N_s$=320,000 SPFH features are extracted from a retrieval target 3D model.

Encoding SPFH by SV: The set of densely extracted SPFH from the oriented point set of the 3D model is then encoded by using SV. SV encodes each of the SPFH features by using posterior probabilities and displacements from cluster centers computed in the 125-dimensional SPFH feature space. A codebook is learned by Gaussian Mixture Model clustering on 250,000 SPFH features randomly selected from all the SPFH of 3D models in the database. We use soft-assignment variant of SV coding [FO14], in which a SPFH feature is assigned, with weights, to several neighboring cluster centers. The weights are posterior probabilities of the SPFH feature belonging to the assigned centers. If we use the number of cluster centers, or the number of codewords, N_c=64, then the SV-encoding of a set of SPFH features would result in a set of encoded SPFH features having dimensionality (125+1)×N_c=8,064.

Aggregating encoded SPFH: After sub-volume partitioning, a SV-SPFH feature for the sub-volume is formed by aggregating, or pooling, the SV-encoded SPFH features for the set of oriented points within the sub-volume. A scale index s of the sub-volume is computed as $s = round\left(N_s \cdot l/2\right)$ where l is a diagonal length of the sub-volume. The set of encoded SPFH feautures for scale s within the sub-volume is accumulated into a feature vector and is normalized by power normalization followed by L2 normalization as with [PSM10].

Note that other descriptors for oriented point set which have rotation invariance in 3D space can be used to describe sub-volumes of 3D models. In the preliminary experiments, we compared SPFH against other commonly-used local 3D geometric features, i.e., Spin Image [JH99] and RoPS [GSB*13]. We also compared SV against other feature aggregation methods, i.e., BF, VLAD [JDS*10], Fisher Vector [PSM10], and LLC [WYY*10]. We found that the combination of SPFH and SV, i.e., SV-SPFH, performed the best among the combinations we tried.

3.3.2 Hashing sub-volume features

The SV-SPFH feature extracted from the sub-volume is a high dimensional (e.g., 8,064 dim.), real-valued vector. To reduce temporal cost and storage cost for retrieval, we hash SV-SPFH features into a set of compact binary codes. To do so, we first project the SV-SPFH features onto low-dimensional (e.g., 512 dim.), real-valued subspace by using Kernel PCA (KPCA) with dot kernel. In the low-dimensional space, we learn a hash function by using ITQ algorithm [GL11]. ITQ learns a rotation matrix so that error due to binarization that maps from the original real-valued space to the Hamming space becomes small. We perform 20 iterations to learn the rotation. Then, the SV-SPFH feature **x** for the sub-volume is hashed into a binary code **b** by using the following equation;

$$\mathbf{b} = sign\left(\mathbf{x} \cdot \mathbf{M}_p \cdot \mathbf{M}_r\right) \quad (1)$$

where *sign*() is an element-wise thresholding function by sign, that is, it produces 1 if the element of the vector is > 0 and 0 otherwise. $\mathbf{M}_p$ is a KPCA projection matrix, and $\mathbf{M}_r$ is a rotation matrix for ITQ. $\mathbf{M}_p$ and $\mathbf{M}_r$ are learned by using a set of 5,000 SV-SPFH features randomly selected from all the SV-SPFH features of the sub-volumes in the database.

If we use binary codes having N_b bit each, and if there are N_v sub-volumes per 3D model, SV-SPFH features for a database having N_m 3D models occupies $N_b \times N_v \times N_m$ bit. For example, if N_b=512, N_v=1K, and N_m=100K, footprint of the features is about 6 GByte. Using a modern PC, all the features would fit in its main memory for fast retrieval.

3.4 Ranking 3D models

Ranking of 3D models in the database against a given query is performed efficiently by comparing their binarized SV-SPFH features. The SV-SPFH feature $\mathbf{x}_q$ for the query **q** is hashed into a binary code $\mathbf{b}_q$ by using equation 1. The distance D between the query **q** and the 3D model **t** in the database is computed as;

$$D(\mathbf{q}, \mathbf{t}) = \underset{1 \le i \le Nv}{\operatorname{argmin}}\, d\left(\mathbf{b}_q, \mathbf{b}_{ti}\right) \quad (2)$$

where N_v is the number of sub-volumes for the model **t**, $\mathbf{b}_{ti}$ is a binary code for a sub-volume i of the model **t**. We use Hamming distance as a distance function d between a pair of binary codes. Hamming distance between a pair of binary code can be computed very efficiently by using a combination of (SIMD) XOR instruction and bit count instruction. Finally, 3D models in the database are ranked based on their Hamming distances to the part-based query.

4. Experiments and results

4.1 Experimental setup

To evaluate accuracy and efficiency of the P3D-RSVP algorithm, we created three P3DMR benchmark databases, i.e., P-PSB, P-SH11NR, and P-PSBX. Retrieval target

(whole) 3D models for these benchmark databases are adopted from existing benchmark databases for W3DMR. We created part-based queries from these 3D models by interactively segmenting a part (or parts) from each 3D model. Figure 2 shows examples of part-based queries and retrieval target 3D models as well as their ground-truth categories for the P-PSB and the P-SH11NR. P-PSB and P-PSBX contain polygon soup, open mesh, and other "not-so-well-defined" 3D models. P-SH11NR, on the other hand, contains only closed meshes defining solid 3D models.

The P-PSB contains 304 rigid retrieval target 3D models and 67 part-based queries classified manually into 51 semantic categories. The set of target 3D models for the P-PSB is a subset of the *Princeton Shape Benchmark* (*PSB*) [SMK*04]. The *P-SH11NR* contains 600 non-rigid retrieval target 3D models adopted from SHREC 2011 Non-rigid 3D watertight meshes dataset [LGB*11] and 180 part-based queries classified into 30 semantic categories. For both benchmarks, the sets of part-based query were created by cutting off parts of the retrieval target 3D models.

The P-SPBX was created to evaluate scalability of the proposed algorithm on a benchmark much larger in size than the P-PSB or the P-SH11NR. We artificially created 49,696 "imposter" retrieval target 3D models by randomly scaling and rotating the target 3D models in the P-PSB. The P-PSBX has a total of 50K 3D models. The part-based queries for P-PSBX are the same as in the P-PSB. In computing retrieval accuracy for the P-PSBX, all the imposters are excluded from the retrieval result, and only the 3D models existed in the P-PSB are used.

We wanted to compare retrieval accuracy among of the P3D-RSVP with other P3DMR algorithms. However, to our knowledge, there isn't any P3DMR algorithm for a large-scale database. In the experiments, we thus compare the P3D-RSVP against the following; one "off-the-cuff" P3DMR algorithm and two W3DMR algorithms.

RSWP-SV-DSIFT: This is a naïve extension of RSWP algorithm by Jiang et al. [JMY12]. In a pre-processing step, a retrieval target 3D model is rendered from 20 viewpoints into a set of 20 depth images of size 512×512 pixels. Each rendered image is partitioned into 3×3 non-overlapping 2D rectangular sub-windows by using grids having random intervals. This partitioning is iterated 50 times. For each 3D model, 20×3×3×50 = 9,000 sub-windows (2D images) are generated. As with the P3D-RSVP, the RSWP-SV-DSIFT uses "late-binding" approach for feature aggregation, a set of 4,000 SIFT [Low04] features is densely and randomly extracted per view prior to sub-window partitioning. Each sub-window is described by a SV-DSIFT feature produced by aggregating SV-encoded SIFT features inside the sub-window. By using KPCA, the SV-DSIFT feature is dimension reduced to 512 dimensions.

In the retrieval step, a partial query is also rendered from 20 viewpoints and a set of 1,000 SIFT features is densely extracted per view. The set of SIFT features are aggregated into a feature vector per view and compressed into 512-dimensional vector by using KPCA. Distance between a partial query **q** and a retrieval target model **t** is computed by the following equation where $\mathbf{x}_{qi}$ is a feature vector of i-th view of **q** and $\mathbf{x}_{tj}$ is a feature vector of j-th sub-window of **t**. Distance d is "Cosine distance" in the range [0, 1] computed by subtracting Cosine similarity from 1.0.

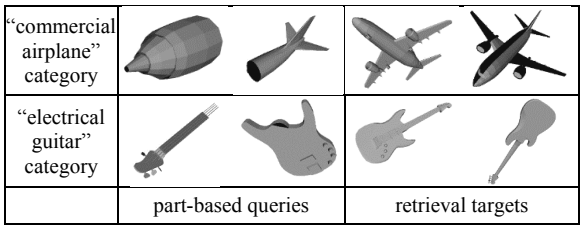
$$D(\mathbf{q},\mathbf{t}) = \sum_{i=1}^{20} \underset{1\le j\le 9000}{\operatorname{argmin}}\, d\left(\mathbf{x}_{qi}, \mathbf{x}_{tj}\right) \tag{3}$$

Per query cost of retrieval for the RSWP-SV-DSIFT is rather high as a large number (i.e., 9,000×20=180,000) of distances among SV encoded floating point feature vector pairs must be computed.

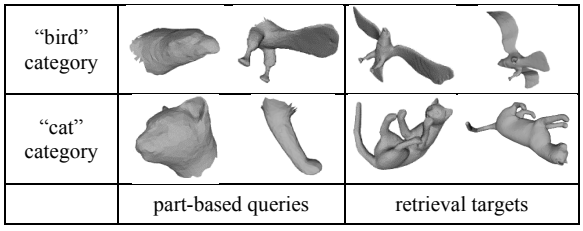
SV-SPFH, SV-DSIFT: These algorithms are essentially for W3DMR in which whole 3D models are compared against each other. Recall that [SPK*13], [SPS14], and [LZQ06] applied this approach to the task of P3DMR. In this approach, both a query model and a retrieval target model are represented by a feature vector per 3D model. That is, the retrieval target model is not partitioned into sub-volumes. For SV-SPFH, a set of SPFH features [RBB09] is densely extracted from the 3D model. The set of SPFH is aggregated by using SV encoding for a feature vector per 3D model. The SV-SPFH feature vectors are compared by using Cosine distance. For SV-DSIFT [FO14], a set of SIFT is densely extracted from depth images rendered from 42 viewpoints uniformly spaced in solid angle around the 3D model. The set of about 13,000 densely sampled SIFT features (about 300 SIFT features per view) is aggregated by SV encoding for a feature vector per 3D model. Distance among SV-DSIFT feature vectors are computed by using Cosine distance. For a W3DMR, SV-DSIFT is one of the most accurate retrieval algorithms [FO14].

As accuracy measures, we use Nearest Neighbor (NN) [%] and Mean Average Precision (MAP) [%]. Since the P3D-RSVP is a randomized algorithm, every experiment is run 5 times and an average value of 5 runs is reported. We do not plot deviation of accuracies in the graphs as deviation of 5 runs was small (within 1 %) for most cases.

Table 1 summarizes combination of parameters for the RSVP determined through preliminary experiments. Parameters for the P-PSBX are the same as those for the P-PSB. Table 2 summarizes the number of sub-volumes generated by the RSVP when parameters in Table 1 are used.



(a) P-PSB



(b) P-SH11NR

Figure 2: *Examples of part-based queries and retrieval target 3D models for the P3DMR benchmark databases.*

We used a PC having two *Intel Xeon E5-2650V2* (8 cores, 16 threads each) CPUs and 256 GByte DRAM. All the programs are parallelized by using OpenMP [OMP]. However, computation time is measured without parallelization for evaluation of efficiency.

Table 1: *Parameters for P3D-RSVP algorithm.*

Parameters	P-PSB	P-SH11NR
# of grid intervals N_g	3	3
# of iterations of partitioning N_i	50	50
radius R of Sphere-Of-Interest	0.30	0.15
# of codewords for codebook	64	64
# of bits for binary code N_b	512	512

Table 2: *Number of sub-volumes generated by RSVP.*

	P-PSB	P-SH11NR	P-PSBX
# of target 3D models N_t	304	600	50,000
# of sub-volumes / model $N_v = N_g \times N_g \times N_g \times N_i$	1,350	1,350	1,350
# of total sub-volumes $N_{v\ total} = N_t \times N_v$	410,400	810,000	67,500,000

4.2 Experimental results

4.2.1 *Parameters for RSVP*

We evaluate influences of parameters for RSVP with regard to retrieval accuracy. For this set of experiments, each SV-SPFH feature is projected onto 512-dimensional real-valued space by KPCA, and resulting real-valued vector is used to find a best performing subset of parameters. Cosine distance is used to compute distance between a pair of SV-SPFH features of a query and a sub-volume of a 3D model.

Figure 3 plots retrieval accuracy against the number of grid intervals N_g. For both the P-PSB and the P-SH11NR, N_g=2 or N_g=3 produces higher retrieval accuracy than other N_g. A special case N_g=1 means no random grid partition is performed on retrieval target 3D models, that is, the P3D-RSVP behaves as an algorithm for W3DMR. We can say that the RSVP helps P3DMR, as the retrieval accuracies for $N_g \geq 2$ are higher than that for N_g=1. For N_g>3, accuracy gradually decreases with N_g for both benchmarks. Presumably, the RSVP with N_g=2 or N_g=3 would tend to produce sub-volumes whose scale is similar to part-based query for the P-PSB and the P-SH11NR.

Figure 4 plots retrieval accuracy against the number of iterations N_i for random grid partitioning. We can observe that, for both the P-PSB and the P-SH11NR, accuracy saturates at N_i =50 and nearly unchanged up to N_i =300.

4.2.2 *Compression of sub-volume feature*

In this section, we evaluate the impact the feature compression algorithm, i.e., KPCA dimension reduction and ITQ hashing, on retrieval accuracy. Efficiency of this step will be discussed in the next section.

Figure 5 compares retrieval accuracies of three feature compression algorithms. In Figure 5, "no reduction" plots accuracy of raw 8,064-dimensional, real-valued, SV-SPFH feature. "KPCA" plots accuracy of SV-SPFH projected onto a lower-dimensional real-valued space by using KPCA. "KPCA+Hashing without ITQ" plots accuracy for hashed SV-SPFH where each element of KPCA-reduced feature vector is binarized by its sign without using ITQ. "KPCA+Hashing with ITQ" is accuracy for hashed SV-SPFH binarized by using ITQ algorithm. For distance metric, *Cosine* distance described before is used for real-valued features ("no reduction" and "KPCA") and Hamming distance is used for binary features ("KPCA+Hashing without ITQ" and "KPCA+Hashing with ITQ").

Figure 5 shows ITQ is effective for generating comact and accurate binary codes. In Figure 5, binary code having more than 512 bit produced by using ITQ is about as accurate as the real-valued SV-SPFH without dimension reduction (i.e., "no reduction"). Binary codes produced without using ITQ ("KPCA+Hashing without ITQ") at 512 bit is about 5 % less accurate than the raw SV-SPFH (i.e., "no reduction").

4.2.3 *Efficiency*

Table 3 compares computation time per query for the three benchmark databases, i.e., the P-PSB, the P-SH11NR, and the P-PSBX. Column "feat." indicates time for extracting SV-SPFH from a part-based query. Column "dist." shows time for computing distances among the feature of the query and the features of *all* the sub-volumes of all the 3D models in the database. For the P-PSBX, we did not experiment with "no reduction" option since uncompressed SV-SPFH features for 68M sub-volumes could not be stored on the HDD used for the experiment.

By using hashing, distance computation among the query and the sub-volumes in the database is significantly accelerated. For example, for P-PSBX, it took about 0.2s for distance computation among a query and about 68M sub-volumes of 50K 3D models. Even for the P-PSBX containing 50K 3D models, total time required for retrieval is less than 2s. Retrieval accuracies of RSVP employing ITQ hashing are nearly identical for the P-PSBX (MAP=35.9%)

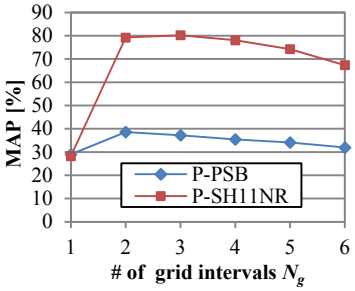


Figure 3: *Number of grid intervals and retrieval accuracy.*

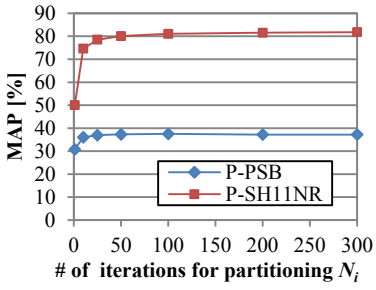


Figure 4: *Number of iterations for RSVP and retrieval accuracy.*

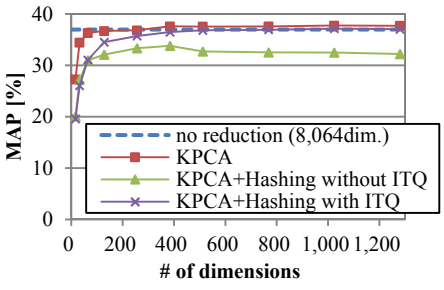


Figure 5: *Reduced dimensionality and retrieval accuracy (P-PSB).*

and P-PSB (MAP=37.0%). Proposed P3D-RSVP algorithm seems to show good scalability both in terms of efficiency and accuracy.

Table 4 shows time for pre-processing. Retrieval phase is accelerated by hashing. However, pre-processing phase, especially feature extraction from retrieval target 3D models in the database, is time-consuming. As mentioned in Section 3, P3D-RSVP algorithm extracts and encodes a set of 3D geometric features prior to random grid partitioning. However, it is still time consuming to process a database containing a large number of 3D models.

Table 5 summarizes spatial complexity for storing all the features for all the sub-volumes of every 3D model in the database. Dominant factor is the features for sub-volumes. Sum of all the other data, which include a codebook for SV encoding, a projection matrix for KPCA, and a rotation matrix for ITQ, occupies meager 16MBytes.

P3D-RSVP using ITQ hashing with 512 bit requires only about 84KBytes per 3D model to store a set of features for the 3D model having 1,350 sub-volumes. For P-PSBX, it takes 4GBytes to store all the binary codes for 68M sub-volumes for 50K 3D models. As shown in Table 3, part-based retrieval can be done in an "interactive" speed (e.g., within a few seconds) since the 4GB footprint of the features is small enough to easily fit in a main memory of contemporary PC. With a larger main memory of 256 Gbyte, for example, we could fit all the features for 3 million 3D models on the main memory. In that case, we estimate that, by exploiting multiple cores, a query into the 3 million model database would take a few seconds.

Table 3: *Computation time* [*s*] *per query.*

algorithms	feat.	dist.	total
P-PSB (304 retrieval target 3D models)			
no reduction (8,064 dim.)	1.518	7.416	8.934
KPCA (512 dim.)	1.653	0.569	2.222
KPCA+ITQ (512 bit)	1.653	0.001	**1.654**
P-SH11NR (600 retrieval target 3D models)			
no reduction (8,064 dim.)	0.700	15.216	15.916
KPCA (512 dim.)	0.839	1.027	1.866
KPCA+ITQ (512 bit)	0.839	0.002	**0.841**
P-PSBX (50,000 retrieval target 3D models)			
no reduction (8,064 dim.)			
KPCA (512 dim.)	1.653	90.442	92.095
KPCA+ITQ (512 bit)	1.653	0.199	**1.852**

Table 4: *Computation time* [*s*] *to pre-process database.*

processes	P-PSB	P-SH11NR
feature extraction	19664.5	32007.0
codebook learning for SV	130.1	120.3
KPCA learning	207.9	202.1
ITQ learning	288.8	270.2
Total	20291.3	32599.6

Table 5: *Storage cost* [*GByte*] *for sub-volume features.*

algorithms	P-PSB	P-SH11NR	P-PSBX
no reduction (8,064 dim.)	12.33	24.33	2027.75
KPCA (512 dim.)	0.78	1.54	128.75
KPCA+ITQ (512 bit)	**0.02**	**0.05**	**4.02**

4.2.4 Comparison of accuracy among algorithms

We compare accuracy of the proposed P3D-RSVP against one "off-the-cuff" P3DMR algorithm, i.e., RSWP-SV-DSIFT and two W3DMR algorithms, i.e., SV-SPFH and SV-DSIFT. Table 6 compares retrieval accuracies using the two P3DMR benchmark databases. As shown in Table 6, for both benchmarks, our proposed P3D-RSVP performs the best among the algorithms we have compared. For example, P3D-RSVP significantly outperforms RSWP-SV-DSIFT. This result shows that randomized sub-volume partitioning of 3D polygonal mesh or 3D point set is more suitable for P3DMR than randomized sub-window partitioning of 2D images. Interestingly, NN for P3D-RSVP is much higher than other algorithms. This result indicates P3D-RSVP is good at searching a 3D model which has a part (nearly) identical to a part-based query. SV-SPFH and SV-DSIFT, which are algorithms for W3DMR, did not perform well for the P3DMR benchmarks. Figure 6 shows examples of retrieval results due to P3D-RSVP and RSWP-SV-DSIFT.

Table 6: *Retrieval accuracy* [%].

algorithms	P-PSB		P-SH11NR	
	NN	MAP	NN	MAP
P3D-RSVP (proposed)	**79.1**	**37.0**	**96.7**	**76.3**
RSWP-SV-DSIFT	32.8	28.8	42.2	40.3
SV-SPFH	40.3	26.4	33.3	24.5
SV-DSIFT	28.4	26.5	28.3	28.5

5. Conclusion and future work

In this paper, we proposed a *Part-based 3D model retrieval by Randomized Sub-Volume Partitioning* (*P3D-RSVP*) algorithm for efficient part-based 3D model retrieval (P3DMR) on a 3D model database having a significant size. The P3D-RSVP randomly and iteratively partitions a retrieval target 3D model into a set of sub-volumes by using random 3D grids. The approach requires comparison with a query and a large number (e.g., 68M for 50K 3D models) of sub-volumes. To accelerate the comparison, a feature describing a sub-volume is hashed into a compact (e.g., 512 bit) binary code. A pair of binary codes can be compared very efficiently by using Hamming distance. Compactness of the binary-coded features is also crucial in fitting all the features for the large number of sub-volumes into a memory for interactive retrieval.

To quantitatively evaluate accuracy and efficiency of the P3D-RSVP algorithm, we created three new P3DMR benchmarks. Experiments using these benchmarks showed that the P3D-RSVP algorithm showed higher accuracy than algorithms we have compared. In addition, it is efficient; it took less than 2 seconds to process a query against a database having 50K 3D models.

As a future work, we need to compare P3D-RSVP with algorithms using sliding sub-volumes or segmentation of 3D models. We also need to improve accuracy of P3DMR, e.g., via more discriminative yet efficient 3D geometric features.

References

[AMS*11] ATTENE M., MARINI S., SPAGNUOLO M. et al.: Part-in-whole 3D shape matching and docking. *The Visual Computer* **27**(11), 991–1004, 2011.

[CDF*04] CSURKA G., DANCE C. R., FAN L., WILLAMOWSKI J., et al.: Visual Categorization with Bags of Keypoints. *ECCV 2004 workshop on Statistical Learning in Computer Vision*, 59–74, 2004.

[DT05] DALAL N., TRIGGS B.: Histograms of Oriented Gradients for Human Detection. *CVPR 2005*, 886–893, 2005.

[FMA*10] FERREIRA A., MARINI S., ATTENE M., et al.: Thesaurus-based 3D Object Retrieval with Part-in-Whole Matching. *IJCV*, **89**(2-3), 327–347, 2010.

[FO14] FURUYA T., OHBUCHI R.: Fusing Multiple Features for Shape-based 3D Model Retrieval. *BMVC 2014*, 2014.

[GL11] GONG Y., LAZEBNIK S.: Iterative quantization: A procrustean approach to learning binary codes. *CVPR 2011*, 817–824, 2011.

[GSB*13] GUO Y., SOHEL F., BENNAMOUN, M., et al.: RoPS: A local feature descriptor for 3D rigid objects based on rotational projection statistics. *ICCSPA 2013*, 1–6, 2013.

[IG07] IP C. Y., GUPTA S. K.: Retrieving Matching CAD Models by Using Partial 3D Point Clouds. *Computer-Aided Design and Applications,* **4**(5), 629–638, 2007.

[JDS*10] JEGOU H., DOUZE M., SCHMID C., et al.: Aggregating local descriptors into a compact image representation, *CVPR 2010*, 3304–3311, 2010.

[JH99] JOHNSON A.E., HEBERT M.: Using spin images for efficient object recognition in cluttered 3D scenes. *PAMI*, **21**(5), 433–449, 1999.

[JMY12] JIANG Y., MENG J., YUAN J.: Randomized visual phrases for object search. *CVPR 2012*, 3100–3107, 2012.

[KHK10] KANEZAKI A., HARADA T., KUNIYOSHI Y.: Partial matching of real textured 3D objects using color cubic higher-order local auto-correlation features. *The Visual Computer*, **26**(10), 1269–1281, 2010.

[Lam09] LAMPERT C. H.: Detecting objects in large image collections and videos by efficient subimage retrieval. *ICCV 2009*, 987–994, 2009.

[LBZ*13] LIU Z.-B., BU S.-H., ZHOU K., et al.: A Survey on Partial Retrieval of 3D Shapes. *Journal of Computer Science and Technology*, **28**(5), 836–851, 2013.

[LGB*11] LIAN Z., GODIL A., BUSTOS B., et al.: SHREC'11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes, *EG 3DOR'11*, 79–88, 2011.

[Low04] LOWE D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, **60**(2), 91–110, 2004.

[LZQ06] LIU Y., ZHA H., QIN H.: Shape Topics: A Compact Representation and New Algorithms for 3D Partial Shape Retrieval. *CVPR 2006*, 2025–2032, 2006.

[MBO10] MIAN A., BENNAMOUN M., OWENS R.: On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes. *IJCV*, **89**(2–3), 348–361, 2010.

[OFC*02] OSADA R., FUNKHOUSER T., CHAZELLE B., et al.: Shape Distributions. *ACM Trans. on Graphics*, **21**(4), 807–832, 2002.

[OMP] OpenMP.org, http://openmp.org

[PSM10] PERRONNIN F., SÁNCHEZ J., MENSINK T.: Improving the fisher kernel for large-scale image classification. *ECCV 2010*, pp.143–156, 2010.

[RBB09] RUSU R.B., BLODOW N., BEETZ M.: Fast Point Feature Histograms for 3D registration. *ICRA 2009*, 3212–3217, 2009.

[RBK96] ROWLEY H. A., BALUJA S., KANADE T.: Human Face Detection in Visual Scenes. *NIPS* 8, 875–881, 1996.

[SMK*04] SHILANE P., MIN P., KAZHDAN M., et al.: The Princeton Shape Benchmark, *SMI 2004*, 167–178, 2004.

[SPK*13] SFIKAS K., PRATIKAKIS I., KOUTSOUDIS A., et al.: 3D Object Partial Matching Using Panoramic Views. *ICIAP 2013*, *LNCS*, **8158**, 169–178, 2013.

[SPS14] SAVELONAS M. A., PRATIKAKIS I., SFIKAS K.: Fisher encoding of adaptive fast persistent feature histograms for partial retrieval of 3D pottery objects. *EG 3DOR'14*), 61–68, 2014.

[SSS*08] SHALOM S., SHAPIRA L., SHAMIR A., et al.: Part analogies in sets of objects. *EG 3DOR'08*, 33–40, 2008.

[SX14] SONG S., XIAO J.: Sliding Shapes for 3D Object Detection in Depth Images. *ECCV 2014*, 634–651, 2014.

[SYY*05] SUZUKI M. T., YAGINUMA Y., YAMADA T., et al.: A partial shape matching method for 3d model databases. *SEA2005*, 389–394, 2005.

[WYY*10] WANG J., YANG J., YU K., et al.: Locality-constrained Linear Coding for Image Classification, *CVPR 2010*, 3360–3367, 2010.

[ZYZ*10] ZHOU X., YU K., ZHANG T., et al.: Image Classification using Super-Vector Coding of Local Image Descriptors, *ECCV 2010*, 141–154, 2010.

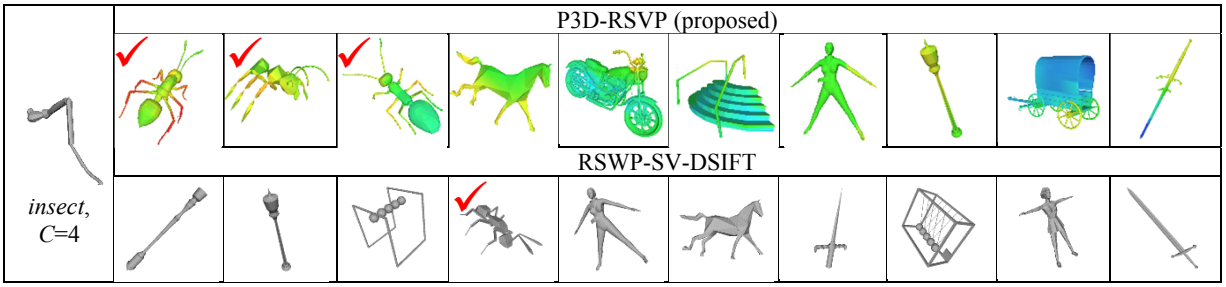


Figure 6: *Examples of retrieval results. For P3D-RSVP, we visualized the part in (whole) 3D models that matched the given part-based query. C is the number of 3D models in the database which belong to the same semantic category with the query.*