Shape Retrieval with Spherical Bounding Volume Hierarchy

Sang Hyun Son
dept. Computer Science and Engineering
Seoul National University
Seoul, Korea
shh1295@snu.ac.kr

Chang Woon Choi
dept. Electrical and Computer Engineering
Seoul National University
Seoul, Korea
zzzmaster@snu.ac.kr

***Abstract*—Thanks to the current development of various techniques to produce 3D models, the number of 3D models to store is growing exponentially these days. Even though text based search method offers some good results, content based search is essential to fully utilize the whole database. However, even though prior search algorithms show efficiency in comparing two models, they settled on conducting such comparisons linearly. Since these approaches have inborn limit of O(N) time complexity, here we propose a novel approach to overcome this limitation. Our approach is based on Bounding Volume Hierarchy(BVH), which offers theoretical guarantee of O($logN$) time complexity for shape retrieval with maintaining efficiency and robustness of previous works.**

***Index Terms*—Shape retrieval, Bounding volume hierarchy, Spherical bounding volume, Residual distance, Hausdorff distance, Chamfer's distance**

I. Introduction

With the resurgence of many fields using 3D shapes such as virtual reality, robotics and with the advancement of modeling techniques for 3D shapes, the amount of 3D models has been increased enormously. Hence, this has led to the development of the search engines for 3D shapes such as the 3D model search engine at Princeton university [1], and the 3D model retrieval system at National Taiwan University [2]. In many cases, content based 3D shape retrieval methods, that use shape properties of the 3D models to search for similar models, work better and useful than text based methods [20]. But unfortunately, the majority of 2D retrieval methods do not generalize directly to content based 3D model retrieval [26]. Thus, various attempts have been made to do content based shape retrieval so far. And there are some criteria to compare those approaches like *efficiency* and *robustness* [26]. For instance, [5] tried feature-based methods to utilize geometric or topological properties of 3D shapes in defining their similarity measure. Also there were graph based methods such as [25] which used a shape descriptor with skeletal graph that encodes geometric and topological information, and [19] which applied model signature graphs, that both model the topology of a shape model by a graph structure. But such methods need a lot of computation power to retrieve 3D shape with respect to query model. In general, the feature-based models use KNN algorithm or a similar algorithm, which takes more than O(N-N^2) times and much more time when using graph structure in 3D model [26].

Here, we present an efficient and robust 3D object search engine that takes 3D mesh data as input and shows the most similar 3D models in our database. We will show that *Spherical Bounding Volume Hierarchy*(BVH) could give precise and robust results as other methods, with much more efficiency. A Bounding Volume Hierarchy is a tree structure of a set of geometric objects that are wrapped in bounding volumes which form the nodes of the tree. We selected sphere for bounding volume in this project, since various distance measure operations run very fast with spheres, and also it is easy to implement. The first half of our work is, building Spherical BVH for both single object(SBVH) and multiple objects or database(MBVH)(See Fig.1 and Fig.2.) . Then with some various distance metrics(Residual Distance, Hausdorff Distance, and Chamfer's Distance), taking advantage of the tree structure, we designed an efficient algorithm that searches the closest SBVH from MBVH for the given query model in O($logN$) time.

Our contributions can be summarized as follows:

- We present a novel approach for content based 3D shape retrieval by building BVH for both single object and multiple objects.
- We designed fast algorithm for 3D shape retrieval that could reduce the time complexity to O($logN$) thanks to the structure of sphere tree.
- On the task of measuring difference between sphere sets, we apply our own distance metric(RD) which shows robust results since it contains the volume information of sphere sets.

II. Related Works

3D Shape Retrieval

In recent years, there have been more attempts to solve 3D shape retrieval using deep learning rather than other classical methods. Liu et al. [18] made deep learning architectures to learn more informative high-level features of the 3D objects descriptors and used it for classification and retrieval tasks. Also Fang Wang et al. [28] tried to retrieve 3D models from 2D human sketches with CNNs. And there were tasks that

used panoramic views with 3D data projection methods, which extract panoramic views to combine and feed to CNN for the retrieval tasks such as [23]. Then, a novel Multi-View CNN(MVCNN) was proposed by Su et al. [24] for 3D object retrieval and recognition/classification tasks. Feng et al. [10] proposed an ensemble of AEs using only one RGB-D model for 3D object retrieval.

But before using DL into 3D retrieval problem, there were many approaches without DL. Those approaches have advantages that they give interpretable search results. Even if those approaches don't give desired results well, they allow us find out reasons easily, and have room for interpretation. There were shape retrieval methods that used global feature of 3D shape such as [6], and C. Y. Ip et al. [16] used global feature distribution of 3D shape to retrieve 3D shape. . Funkhouser et al. [12] utilized spatial map to their 3D search engine. As mentioned before, there were approaches with graph model of 3D shape such as [19], [25].

Bounding Volume Hierarchy

In this article, we use the hierarchy of 3D shapes. There are two fundamentally different types of hierarchies: Space hierarchies and object hierarchies [3]. Space hierarchies are used for partitioning space into subregions. For example, there are Octrees and BSP trees [11]. Object hierarchies are used to partition objects into smaller parts. Typical examples are Bounding Volume Hierarchies [29] and OBB trees [15]. But in our work, we use only object hierarchy(especially, BVH) since it is better than space hierarchies for the purpose we consider : It contains the topology of original model more than space hierarchies.

BVH was commonly used in graphics for the purpose of real-time collision detection [7], and ray tracing [27]. There were some works for constructing BVH for 3D model. Goldsmith et al. [14] implemented an incremental insertion algorithm to build BVH. Then Müller et al. [21] presented a recursive splitting algorithm for constructing BVH. Erleben et al. [8] described a merging algorithm that builds BVH in bottom-up methods. In terms of using the merging method, our approach to building BVH is similar to [8].

III. Spherical Bounding Volume Hierarchy

A. Concept

Bounding volume hierarchy(BVH) is a hierarchical data structure whose nodes represent bounding volumes. The inner nodes of the hierarchy have bounding volumes that enclose every bounding volume of their own descendants. The leaf nodes of the hierarchy store information of original geometric entities, usually polygons. Although it was originally devised as a means to accelerate rendering process [22], due to its simplicity and efficiency, it has been widely used in computer graphics field for various purposes. It is typically used to accelerate collision detection and minimum distance query between distinct geometric entities in the given scene [17].

Inspired by the success of BVH in distance queries, we propose to use it for shape retrieval. To be specific, we decided to use spherical BVH, which uses spheres as bounding volumes. Even though there are many geometric objects that could be used for bounding volumes, such as Axis Aligned Bounding Box(AABB) or Oriented Bounding Box(OBB), we chose sphere because it offers simple distance measures. For instance, it is computationally cheap to compute minimum distance or Hausdorff distance between two different spheres. Also, they are easy to implement compared to other bounding volumes [4]. However, there are also some trade offs for using sphere, which will be discussed at the end of this article.

B. Algorithm

BVH can be constructed in two different contexts : for single object, and for multiple objects. For clarity, we will mark BVH for single object as SBVH and BVH for multiple objects as MBVH.

We have to build SBVH for the given query model to use in search process(Fig.1). The SBVH could be built in bottom-up fashion. To begin with, we can extract sample points on original mesh model. Since triangles in each model differ in size, we have to select more points from bigger triangle than smaller ones. This uniform extraction process is important because it determines the quality of SBVH. If size of bounding volumes has high variance from root nodes, the error will accumulate in the building process and result in inefficient structure. After selecting sample points, we can create spheres centered at those sample points by enlarging each of them until they cover all of the original triangle mesh. These spheres become the leaf nodes of SBVH and we can use them to build inner nodes. To make inner nodes, we have to find appropriate pairs of nodes to merge in the lower level that would result in small amount of volume increase. We used KD tree to find those pairs.

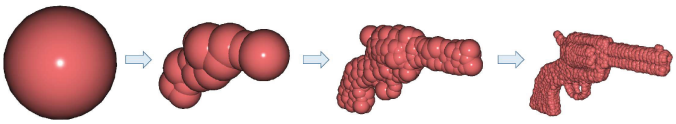

Fig. 1. Example of SBVH for gun model

We also need MBVH for storing multiple objects and to use in search process(Fig.2). For entire 3D models in our database, we build our MBVH by inserting models one by one. Each time the model is inserted, we build SBVH for the model. When the first model is inserted, we can use SBVH of the model itself as our MBVH. From the second model, each time a new model is added, we can consider it as a query model and get search result in our database. When we get the result, we can get the route from root node of our MBVH to the leaf node containing the result. Our algorithm finds the most suitable place to insert our new model in the route. To be specific, our new model creates a new branch and the most suitable place is the node to make the new branch. To minimize error accumulation, we defined the best node to insert new branch as the very node where minimum volume increase occurs when the insertion is made.

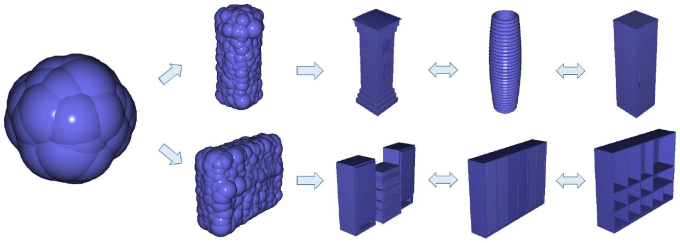

Fig. 2. Example of MBVH for database

IV. Error Metrics

A. Residual Distance(RD)

We coined our method *Residual Distance*(RD), which is a kind of method that measure the difference between two different sphere sets. For given two sphere sets X and Y, we can find subsets of each sphere sets that are not included in the other sphere sets. The volume sum of spheres in such subsets is defined as Residual Distance. Formally, RD for sphere sets X and Y is defined as :

$$RD(X,Y) = volume(X \cap Y) - volume(X \cup Y)$$

The RD would be calculated between sphere sets which are located in same level at each sphere trees. Since exact value of RD is hard to compute, we approximate it by using collision detection algorithm. If we find collision between spheres form each set, we use the collision information to exclude those parts that are included in both sphere sets. After that, we can use only residual parts for our RD measure.

B. Hausdorff Distance

One of the simplest approaches in order to provide a measurement for the difference between two different sphere sets is to use the *Hausdorff distance*(HD), which is a very generic technique to define a distance between two nonempty sets. In this paper, we implemented Hausdorff Distance between two sphere sets, similar to [13] which implemented HD between two octrees whose each cubic is the bounding box of the set of points. Generally, HD between two geometric entities X and Y represents the maximum value among minimum distances between the points on those entities. We can apply this concept to sphere sets place same level in each sphere trees without any modification. Formally, HD for sphere sets X and Y is defined as :

$$HD(X,Y)$$

$$= \max\left\{\max_{x\in X}\left\{\min_{y\in Y} d(x,y)\right\}, \max_{y\in Y}\left\{\min_{x\in X} d(x,y)\right\}\right\}$$

(x, y are centers of spheres in X and Y) We can easily observe that the distances between spheres that intersect with each others are never calculated in HD. From this observation, we can use residual parts from RD computation for HD computation.

C. Chamfer's Distance

Another simple approaches in order to provide a measurement for the difference between two different sphere sets is to use the *Chamfer's Distance*(CD). Similar to the CD defined in [9], We define the *Chamfer's distance* between two sphere sets X, Y as:

$$CD(X,Y) = \sum_{x\in X}\min_{y\in Y} d(x,y) + \sum_{y\in Y}\min_{x\in X} d(x,y)$$

(x, y are centers of Spheres in X, Y)

As we implemented HD, CD would be calculated between two sphere sets that are located at same level of each sphere trees. For each spheres, the CD finds the nearest neighbor in the other set and sums the distances up. Since this error metric does not take account radius of spheres, it may not be a good approach to use in our context. However, we tested it in order to compare our results against other approach using only random sampling of points on models.

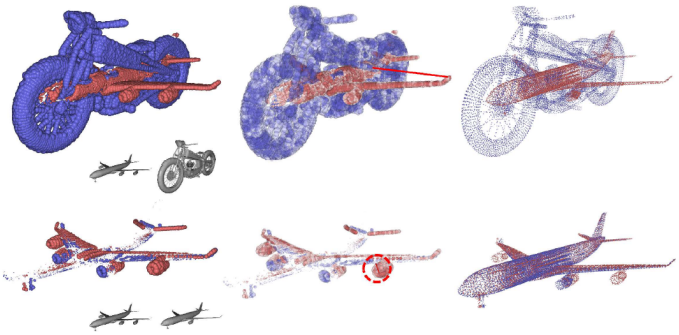

Fig. 3. Visualization of error metrics : Residual Distance, Hausdorff Distance, Chamfer's Distance from left to right

V. Search

A. Algorithm

In our implementation, both SBVH and MBVH obey only one simple principle.

Remark. Let N be a certain node in the hierarchy. Then the sphere sets stored in the child nodes of N must be subsets of that stored in N.

From this principle, we can infer following properties.

Definition V.1. Unilateral Error For given two sphere sets A and B, RD or HD computed from A to B or B to A are called unilateral errors. It is unilateral since real RD or HD should consider both errors from A to B and B to A.

Lemma V.1. For given query model Q, unilateral error from Q to certain node N in the MBVH must be same or smaller than that from Q to any descendent node of N.

Corollary V.1.1. For given query model Q, we can use unilateral error from Q to certain node N in the MBVH for the lower bound of real errors we would get by comparing Q with any models in the subtree rooted at N.

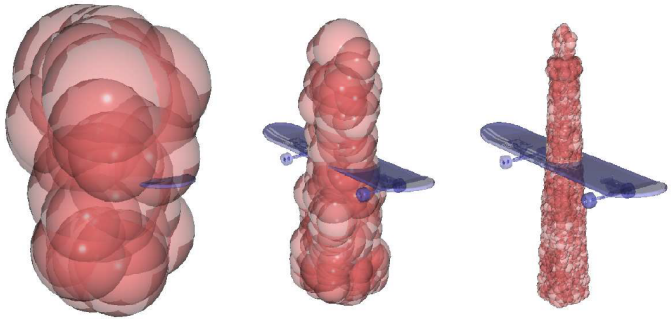

Fig. 4. Unilateral error from the query model(blue) monotonically increases as the compared nodes(red) go to their child nodes

As we discussed lower bound of real errors, upper bound of them can be found.

Lemma V.2. For given query model Q, we can use real error between Q and any model in our database for the upper bound of error between Q and the final result model.

Proof. By definition, the final result model should exhibit least error E with the query model Q. Therefore, it is trivial fact that the error between Q and a random model in our database is same or larger than E. □

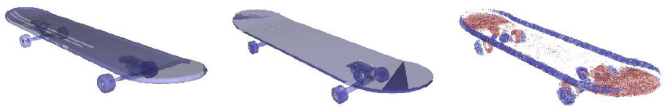

Fig. 5. Error(right) between the query model(left) and a random model(middle) can be used as an upper bound for the minimum error

From these lemmas, we can state our theorem.

Theorem V.3. During search process, we keep upper bound E of minimum error. If the unilateral error between the query model Q and a certain node N is larger than E, we cull out the descendants of N from our search process.

This is the essence of our search algorithm that allows us to reduce time complexity from O(N) to O($logN$). Entire search algorithm runs as stated in $Algorithm1$.

B. Example Results

To test our search algorithm, we used $ShapeNetCore$ data set. It is composed of 50,000 3D models from 55 different categories. However, due to the lack of resources, we used only 10 models from each category and built our search hierarchy. We found that our algorithm gives models from same category as the query model in high priorities for many cases in average 2-3 seconds. Example search results are shown at the end of this article(Fig. 7).

Algorithm 1: Search Algorithm

```
Result: Search Result
Insert root node to priority queue;
Prepare upper bound of error E;
while Priority queue is not empty do
    Pop node N;
    if N is inner node then
        Compute unilateral error E_tmp;
        if E_tmp < E then
            Insert [E_tmp, N] into priority queue;
    else
        Compute error E_tmp;
        if E_tmp < E then
            E = E_tmp;
            SearchResult = N;
    end
end
```

VI. Conclusion

A. Pros

From the above results, we could find out that our algorithm gives pretty accurate results as other shape retrieval algorithms for many cases. Also, there are some advantages that our algorithm has over other algorithms.

First, our algorithm is robust to local perturbations. Even if there are some local perturbations, as long as there are not so many differences exist in BVH structure, our algorithm would give same results.

Second, our algorithm gives intuitive search results. As stated above, our algorithm is based on very intuitive error metrics. Therefore, even if it gives unexpected results, we can easily debug why such outcomes are made both visually and computationally.

Finally, our algorithm guarantees scalability. As we pointed out, hierarchical structure gives theoretical guarantee of O($logN$) running time. In our example, we experienced 4-5 times of acceleration compared to linearly searching every model in the database. This advantage becomes more and more important in current situation where the size of database grows exponentially.

B. Cons

However, there are still many problems that our algorithm has to overcome.

First, we have to solve global alignment problem first. As intuitive our approach is, it is very weak to transformations. Even though two models are same, if they are in different configurations, our algorithm would fail. Therefore, we need a way to align query model into standard configuration.

Second, our current MBVH building algorithm must be improved. For simplicity, we did not implement dynamic update of our MBVH. This ended up in severe error accumulation in our structure. We found that our search algorithm would run much faster if we can solve this error accumulation

issue by dynamic hierarchy update and fully utilize power of hierarchical structure.

Finally, our algorithm does not consider semantic meanings at all. In our data set, categories are defined by semantic meaning. Since our algorithm does not consider such meanings at all, it sometimes gives search result in other category if it shows smaller error than those from same category.

C. Future Work

To address above issues, we are planning to do following works.

First, we are going to solve global alignment issue with BVH. Since BVH offers global topological information in upper levels in the hierarchy, we expect that it would be useful for global alignment. To be specific, we can do alignment step by step by considering more and more geometric entities by going down the hierarchy(Fig. 6).

Second, we will implement tree balancing algorithm to improve our search hierarchy. Every time we insert a new model into our database, we can detect if there is abnormal error increase in some node in the hierarchy. If there is, we have to alleviate it by inserting intermediate nodes in between them or making whole new branch.

Finally, we have to test other bounding volumes. As we pointed out above, we selected sphere as it offers simple distance measures and easy to implement. However, it has severe drawback that error is enormous compared to other bounding volumes. It is well known that OBBs show far superior results than spheres as they are much more efficient in representing original geometries. We expect that those bounding volumes give much better results than current ones.

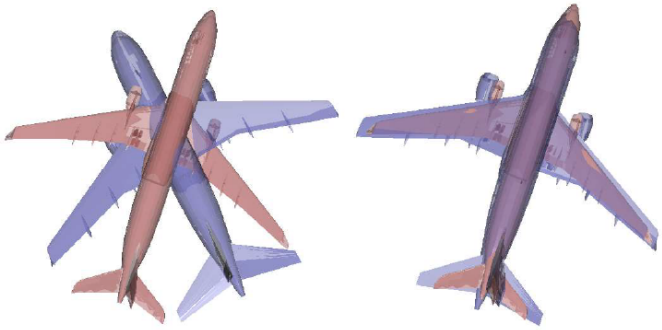

Fig. 6. Global alignment of two models

REFERENCES

[1] 3D model search engine, http://shape.cs.princeton.edu.

[2] 3D model retrieval system, http://3d.csie.ntu.edu.tw/ dynamic/

[3] Kasper Amstrup Andersen, Christian Bay, A survey of algorithms for construction of optimal Heterogeneous Bounding Volume Hierarchies, Technical Report. Copenhague, Denmark: Department of Computer Science, University of Copenhagen, 2006.

[4] Bradshaw, Gareth, and Carol O'Sullivan. "Adaptive medial-axis approximation for sphere-tree construction." ACM Transactions on Graphics (TOG) 23.1 (2004): 1-26.

[5] Bronstein, A. M., Bronstein, M. M., Guibas, L. J., and Ovsjanikov, M. 20011. Shape Google: Geometric words and expressions for invariant shape retrieval. ACM Trans. Graph. 30, 1, Article 1(January 2011).

[6] J. Corney, H. Rea, D. Clark, J. Pritchard, M. Breaks, and R. Macleod. Coarse filters for shape matching. IEEE Computer Graphics and Applications, 22(3):65–74, 2002.

[7] K. Erleben. An Introduction to Ap- proximating Heterogeneous Bounding Volume Hi- erarchies. Technical Report DIKU-TR-02/04, DIKU, 2002.

[8] K. Erleben, J. Sporring, K. Henriksen, and H. Dohlmann. Physics-Based An- imation. Charles River Media, 2005.

[9] Haoqiang Fan, Hao Su, Leonidas Guibas, A Point Set Generation Network for 3D Object Reconstruction from a Single Image, CVPR 2017, pp. 608.

[10] Jie Feng, Yan Wang, and Shih-Fu Chang. 2016. 3D shape retrieval using a single depth image from low-cost sensors. In Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on. IEEE, 1–9. 22(1):83–105, 2003.

[11] J. D. Foley, A. Dam, S. K. Feiner, and J. F. Hughes. Computer Graphics Principles and Practice - Second Edition in C

[12] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. ACM Transactions on Graphics, . Corrected version from 1997, 18th Printing, Addison Wesley, 1997.

[13] D. Girardeau-Montaut, Michael Rouw, Raphael Marc, Guillaume Thibault, Change Detection on Points Cloud Data Acquired with a Ground Laser Scanner, ISPRS, 2015,

[14] J. Goldsmith and J. Salmon, Automatic Creation of Object Hierarchies for Ray Tracing. IEEE CGA, 1987.

[15] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. Com- puter Graphics (SIGGRAPH'96 Proceedings), 1996.

[16] C. Y. Ip, D. Lapadat, L. Sieger, and W. C. Regli. Using shape distributions to compare solid models. In Solid Modeling '02, pages 273–280, 2002.

[17] Larsen, Eric, et al. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[18] Zhenbao Liu, Shaoguang Chen, Shuhui Bu, and Ke Li. 2014. High-level semantic feature for 3D shape based on deep belief networks. In Multimedia and Expo (ICME), 2014 IEEE International Conference on. IEEE, 1–6.

[19] D. McWherter, M. Peabody, W. C. Regli, and A. Shokoufandeh. An approach to indexing databases of graphs. Technical report, Drexel University, sep 2001.

[20] P. Min, M. Kazhdan, and T. Funkhouser. A comparison of text and shape matching for retrieval of online 3D models. 2004.

[21] G. Müller, S. Schafer, and D. W. Fellner. Automatic Creation of Object Hierar- chies for Radiosity Clustering. Technical Report TUBS-CG-1999-06, TU Braunschweig, 1999.

[22] Rubin, Steven M., and Turner Whitted. "A 3-dimensional representation for fast rendering of complex scenes." ACM SIGGRAPH Computer Graphics. Vol. 14. No. 3. ACM, 1980.

[23] Konstantinos Sfikas, Ioannis Pratikakis, and Theoharis Theoharis. 2017. Ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval. Computers Graphics (2017)

[24] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In Proceedings of the IEEE international conference on computer vision. 945–953

[25] H. Sundar, D. Silver, N. Gagvani, and S. Dickenson, Skeleton based shape matching and retrieval. In SMI 2003, pp. 130-139.

[26] Johan W.H. Tangelder, Remco C. Veltkamp, A Survey of Content Based 3D Shape Retrieval Methods, SMI 2004.

[27] Ingo Wald, Solomon Boulos, Peter Shirley, Ray tracing deformable scenes using dynamic bounding volume hierarchies, ACM Transactions on Graphics 2007.

[28] Fang Wang, Le Kang, Yi Li, Sketch-based 3D Shape Retrieval using Convolutional Neural Networks, CVPR 2015.

[29] G. Zachmann. Geometric Data Structures for Computer Graphics. University of Bonn, 2003.

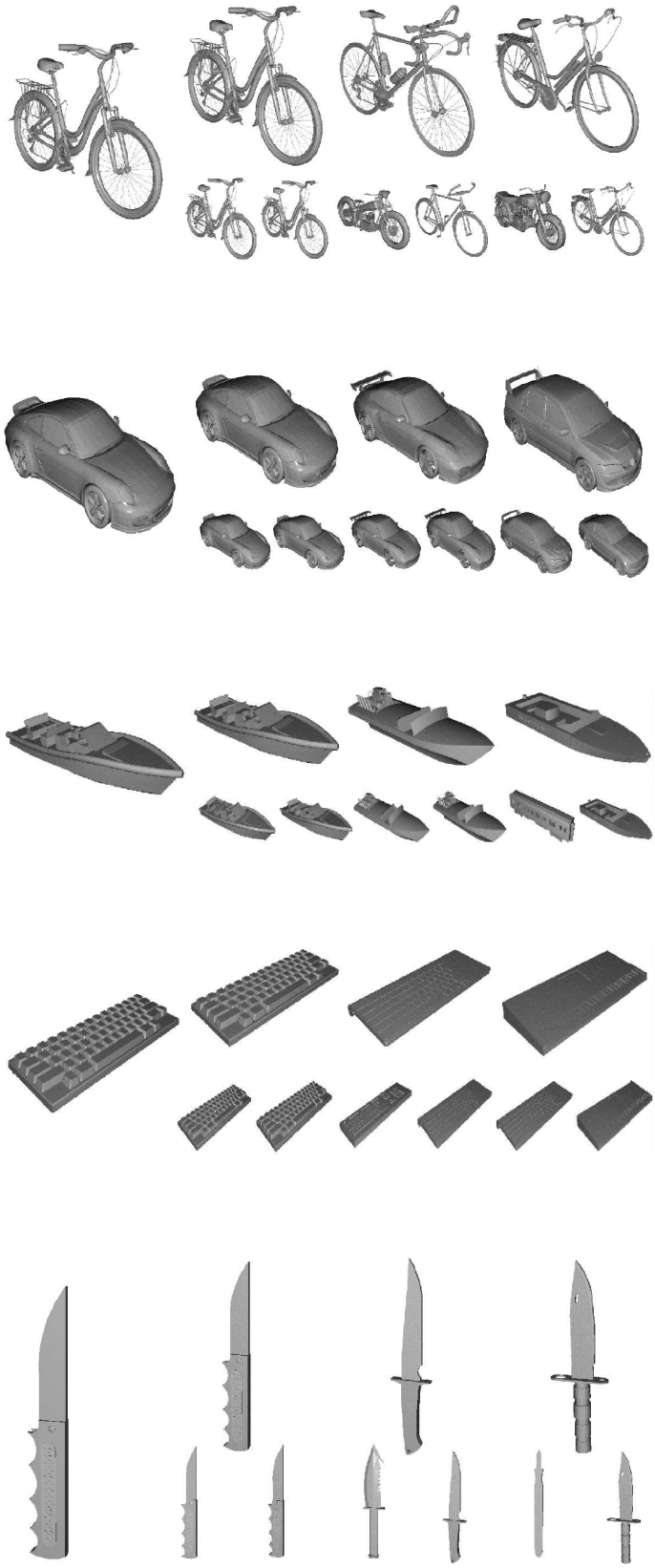

Fig. 7. Search results for sample models : Aligned from left to right, with RD(top), HD(lower left), CD(lower right) result for each model