applied sciences

MDPI

Article

Cube of Space Sampling for 3D Model Retrieval

Zong-Yao Chen [1], Chih-Fong Tsai [1] and Wei-Chao Lin [2,3,*]

[1] Department of Information Management, National Central University, Taoyuan 320317, Taiwan; gp01pc1@gmail.com (Z.-Y.C.); cftsai@mgt.ncu.edu.tw (C.-F.T.)
[2] Department of Information Management, Chang Gung University, Taoyuan 33302, Taiwan
[3] Department of Thoracic Surgery, Chang Gung Memorial Hospital, Linkou, Taoyuan 333423, Taiwan
* Correspondence: viclin@gap.cgu.edu.tw

Abstract: Since the number of 3D models is rapidly increasing, extracting better feature descriptors to represent 3D models is very challenging for effective 3D model retrieval. There are some problems in existing 3D model representation approaches. For example, many of them focus on the direct extraction of features or transforming 3D models into 2D images for feature extraction, which cannot effectively represent 3D models. In this paper, we propose a novel 3D model feature representation method that is a kind of voxelization method. It is based on the space-based concept, namely CSS (Cube of Space Sampling). The CSS method uses cube space 3D model sampling to extract global and local features of 3D models. The experiments using the ESB dataset show that the proposed method to extract the voxel-based features can provide better classification accuracy than SVM and comparable retrieval results using the state-of-the-art 3D model feature representation method.

Keywords: 3D model; 3D object; content-based retrieval; re-sampling; collision detection; similarity match

Citation: Chen, Z.-Y.; Tsai, C.-F.; Lin, W.-C. Cube of Space Sampling for 3D Model Retrieval. *Appl. Sci.* **2021**, *11*, 11142. https://doi.org/10.3390/app112311142

Academic Editor: Kuo-Ching Ying

Received: 10 October 2021
Accepted: 17 November 2021
Published: 24 November 2021

1. Introduction

In the last few years, 3D images and models have gradually extended to different applications and become more popular, such as in Computer-Aided Design (CAD) [1], molecular biology, virtual reality, video and computer games, movies etc. These applications usually require managing a large number of 3D models. In addition, since the number of 3D models is rapidly increasing, there is an urgent need to effectively search from 3D model databases. As a result, 3D model retrieval has become a very important research issue.

In fact, in the field of Computer-Aided Design, a productive approach is to reuse and modify similar existing models instead of always creating new ones [2]. Therefore, for CAD datasets, accurate identification and retrieval of all components is necessary in order to efficiently reuse existing designs [2]. According to Zhu et al. [2], there are two basic types of approaches for the matching and retrieval of 3D CAD data: manufacturing feature-based techniques and shape content-based techniques.

In image retrieval, images usually have to be processed in several steps before they can be queried, including: segmentation; feature extraction; representation; and query processing. Similarly, 3D models or objects also require several steps of processing before retrieval. Figure 1 shows the process of 3D model retrieval [3].

However, although various approaches have been developed for content-based 3D model classification, recognition and retrieval, none of them have achieved high performance on all shape classes [4]. Moreover, some of the existing feature extraction methods are only based on 2D images, which use many different views per image to represent a 3D model. Compared with some of these methods, image-based approaches have shown to produce some promising results, but they are not effective over various types of 3D models. Therefore, we believe that using the image-based approach to represent 3D models

is likely to omit some important information, which has limited discriminative power for 3D model classification.

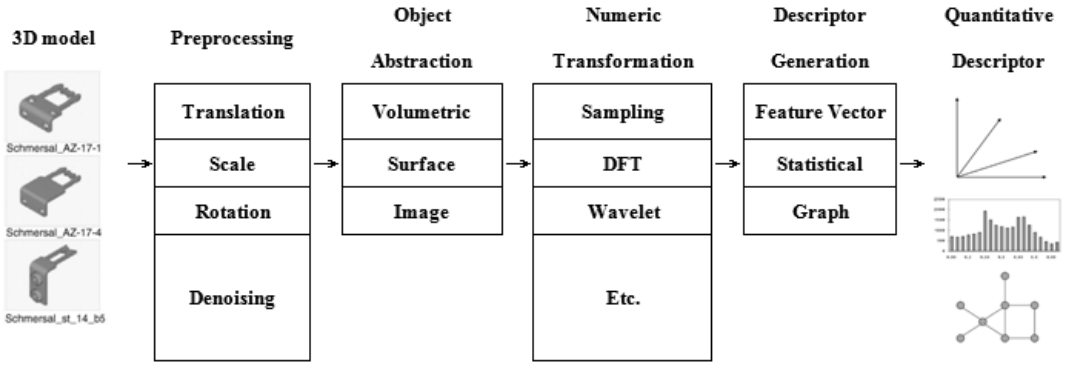


Figure 1. The common 3D model retrieval process.

In image-based 3D model retrieval, using a large number of 2D images to represent a 3D model is impractical, since it is time consuming and requires a very large storage space. Moreover, some important information would not be fully extracted when 3D models are described by images only, such as: fine features; spatial features; global features; etc. In other words, the method of using limited pictures to represent a 3D model has a limited potential to extract better representative feature descriptors for 3D models. Therefore, we propose a method to extract both global and localized features to represent 3D models.

The main objective of this paper is to construct a space-based concept 3D model feature representation, namely CSS (Cube of Space Sampling). The CSS method is a branch of the voxelization method. Although voxel data has been widely used in some previous studies [5–7], most of the voxelization methods or voxel data are used in the visualization field. In contrast to these studies, this paper focuses on voxel-based features, i.e., based on the voxel data rather than the visual display. It is worth mentioning that we believe that the voxel-based feature approach has great potential and robustness for 3D model retrieval, and that our experiments confirm this argument.

In CSS, cube space 3D model sampling is used to extract whole, new features from 3D models. In addition, the extracted features not only provide the global distribution of information, but also provide information on local features.

The rest of this paper is organized as follows: Section 2 overviews related work of 3D model retrieval and feature representation; Section 3 introduces the proposed feature extraction approach; and experimental results and the conclusion are provided in Sections 4 and 5, respectively.

2. Literature Review

2.1. 3D Model Retrieval

Past 3D model retrieval research focuses on the following issues: correctness (good discrimination); automation; robustness; and calculation speed, etc. (1) Correctness: to correctly identifying the degree of similarity between 3D models, so that users can find similar models. (2) Automation: using more sophisticated methods to complete certain processes automatically (such as comparison, alignment, etc.) (3) Robustness: this can be divided into the following issues: translation; rotation and scaling; mesh changes; noise; and shape deformation. (4) Calculation speed: efficiently retrieving 3D models.

While the above four issues are very important, the main advantage of 3D models is the idea of "build once, reuse often" (i.e., modify them to become a new model or reuse). Therefore, in order to fully demonstrate this advantage of 3D models, i.e., to achieve re-use, it is very important to accurately classify 3D models for the system in order to easily find the most similar model to a query.

2.2. Different Types of Models

From the data point of view, 3D model data can be divided into dynamic and static 3D models. A dynamic 3D model can be likened to a toy doll with movable joints, which can be carried out in accordance with the need of the designer's action; thus, a 3D model can have a variety of actions. As a result, the system must be able to distinguish between these actions, which are not the same but have the same model form, as these models are classified as the same model.

On the other hand, most static 3D models are "Engineering modeling", such as those used in construction, furniture, cars, and various spare parts. The number of such models is usually large, but the level of difference between them is small. In order to reduce development time, these models are normally re-used or modified, so the retrieval system should be able to accurately distinguish between the parts in order to improve the ease of their re-use or re-development.

2.3. Model-Matching Methods

Previous research in model-matching methods can be classified broadly into: feature-based; graph-based; and other methods [1].

- Feature-based: previous feature-based research can be divided into: global features; global feature distribution; spatial maps; and local features. Feature-based model matching represents features of a model using a single descriptor consisting of a d-dimensional vector of values, where the dimension (d) is fixed for each model.
    1. Global features: Zhang and Chen [8] extracted global features such as volume, area, statistical moments, and Fourier transform coefficients efficiently. Paquet et al. [9] applied bounding boxes, cord-based, moment-based and wavelet-based descriptors for 3D shape matching. Kazhdan et al. [10] described a reflective symmetry descriptor as a 2D function associating a measure of reflective symmetry to every plane (specified by two parameters) through the model's centroid. Since these methods only used global features to characterize the overall shape of the objects, they could not describe objects in detail, but their implementation was straightforward [1].
    2. Global feature distribution: Osada et al. [11] introduced and compared shape distributions, which measured properties based on distance, angle, area and volume measurements between random surface points. Ohbuchi et al. [12] investigated shape histograms that were discretely parameterized along the principal axes of the inertia of the model. Ip et al. [13] investigated the application of shape distributions in the context of CAD and solid modeling. However, they often performed poorly when they needed to distinguish between shapes having similar gross shape properties but significantly different detailed shape properties [1].
    3. Local features: Zaharia and Prêteux [14] described the 3D Shape Spectrum Descriptor, which was defined as the histogram of shape index values, and calculated over an entire mesh. The shape index, first introduced by Koenderink [15], was defined as a function of the two principal curvatures on continuous surfaces.
    4. Graph-based: graph-based methods can be roughly divided into three categories according to the type of graph used, which are: model graphs; Reeb graphs; and skeletons [1,16].

In skeletons, the focus is on skeletal graph matching with node-to-node correspondence based upon the topology and radial distance about the edge [14].

For Reeb graphs, mathematically, they are defined as the quotient space of a shape (S) and a quotient function (f). Biasotti et al. [17] compared Reeb graphs obtained by using different quotient functions and highlighted how the choice of f determines the final matching result.

The advantages of graph-based methods include the fact that that they are good at capturing shape features, which are relatively high-level and natural, meaning graph-based methods can support local and multi-resolution retrieval. In addition, they are robust when the model has a detailed level of structure. However, the drawbacks include the fact that they are computationally expensive, sensitive to noise and difficult to index and match [16].

- Other methods: other methods, such as view-based similarity, assumes that two 3D models are similar. In particular, the level of similarity between 3D models is judged by different viewing results [17,18]. For example, LightField Descriptor is one of many outstanding 3D model feature descriptors.

On the other hand, Novotni and Klein [19] introduce a geometry similarity approach to 3D shape matching, which is based on calculating the volumetric error between one object and a sequence of offset hulls of the other objects. Another approach is weighted-point set-based similarity, where shape descriptors consist of weighted 3D points [20].

2.4. Normalization

- Translation: in general, 3D modeling software (such as 3ds Max, Maya, etc.) uses different settings or different coordinate systems [21]. However, different coordinate systems are equal to different "spaces". Sometimes, a creator modeling 3D models does not place them in the center of the coordinate systems; that is, the locations of 3D models are not aligned in the coordinate systems. In addition, when these 3D models are created from different coordinate systems, the center of each model is not the same. This will cause more difficulties and biases in the process of re-sampling.
- Scale: similarly, 3D modeling software may use different settings or different scales to construct the 3D modeling space [22]. As a result, different modeling software may produce the models with different scales. In practice, it is common that in a 3D model dataset each 3D model always has different sizes or scales. This usually causes some problems in re-sampling and errors of comparison.
- Rotation: since each model may not have the same angle when placed (e.g., upright, flat, etc.), several difficulties in comparisons can arise (e.g., reversed, tilted, etc.). To this end, 3D models must be appropriately adjusted. In the literature, some methods, such as PCA and CPCA, have been applied to solve such problems [21].

2.5. Summary

In summary, no related studies applied the collision detection method along with the voxelization method for 3D model feature extraction, and no studies show how robust the voxel-based feature is for problems caused by rotation. In addition to 3D model normalization, it is necessary to pre-process 3D models before extracting their features or descriptors. The pre-processing steps include: the transformation between different 3D data representations (e.g., to transform polygon meshes into voxel grids); the partition of model units; and vertex clustering, etc. [22–24]. According to the literature review, we can observe that no single most credible method of 3D model retrieval has yet arisen between standard sampling methods or most discerning feature methods. Therefore, we propose a new approach of 3D model re-sampling, in which various types of features are extracted in the interest of improving classification accuracy.

3. 3D Model Feature Representation

3.1. Preprocessing

Before feature extraction, pre-processing for the raw data of a 3D model is usually required, such as translation, scale, and rotation, etc. [1,18]. After these pre-processing tasks are completed, some specific features can be extracted, which are used as the descriptors or feature representations of a 3D model.

- Translation: in our approach, we focused on the calculation of each model by the X, Y and Z axes to obtain the maximum length, which was then used to find the center of a

3D model. Next, a unified coordinate system was established. Finally, the 3D model was moved to the center of the coordinate system, so that all models will be aligned.

- Scale: in our approach, we applied the maximum length of each axis to perform normalization for each 3D model. In addition, we set the maximum axial length to one, which made all of the normalized 3D models fit into a cube space with a length, width and height of one.
- Rotation: when the abovementioned preprocessing tasks were completed, the 3D models were ready for re-sampling. Through the CSS method, the 3D model was be changed into CSS data (a type of voxel), and then the CSS data were transformed into voxel-based features. Since the features contain the space and the global information (because CSS data is very information-rich and powerful) of the model, there was no need to perform any additional processing to solve the rotation problem.

3.2. Cube of Space Sampling (CSS)

After pre-processing, feature extraction from 3D models can be performed. In this paper, a spatial scale of sampling for re-sampling 3D models was extracted. In other words, we focused on extracting the features of the spatial concept of a 3D model. This concept is similar to a set of building blocks, which is based on a number of small building blocks to construct and fit to each 3D model. The re-sampling process is described as follows:

- Step one: normalization. All 3D models were normalized and shifted to the center of the cube space, in which they have the same views.
- Step two: selection of cutting number. The cutting number of CSS, i.e., N (e.g., N = 10), was used to determine the density of sampling. In particular, the average cut into N blocks of the cube space in each dimension was considered. As a result, we obtained the total number N^3 of sub-spaces. Next, these sub-spaces were used to perform re-sampling for each 3D model.
- Step three: re-sampling. Our method applied a collision detection method to perform re-sampling, which entailed testing each subspace Sijk as a collision with a 3D model (where i, j, and k mean the x, y, and z axes respectively). If it was a collision, then the subspace was labeled as 1; if not, 0. Consequently, we could obtain a sampling matrix, which was composed of 0s and 1s. However, this sampling matrix acted a basis for our classification features. The labeling method is by:

$$S_{ijk} \in [1, 0], I = 1, 2 \ldots N, j = 1, 2 \ldots N, k = 1, 2 \ldots N \quad (1)$$

Figure 2 shows the pseudo code of the CSS process.

```
Define the Re-sampling number N
Produce the sub-space matrix S (N* N* N) with N
For each 3D model M in a chosen dataset
  Normalize M
  All subspace Sijk = 0
  For each triangle (T) in the 3D model
    Find the AABB (Axis-Aligned Bunding Box) space (A) of triangle (T)
    Divide space A into some subspaces (Uqyp)
    For each subspace Uqyp in space A
      If triangle (Ty) and subspace Uqyp is collision, then
        Sqyp = 1
      End
    End
  End
End
```

Figure 2. The pseudo code of CSS.

Note that there are two key techniques which were used in our method: "AABB collision detection" and "Fast Collision detection". "AABB collision detection" was used to reduce the detection area, and "Fast Collision detection" was for determining whether the subspace was a collision. "AABB collision detection" and "Fast Collision detection" are introduced as follows.

1. 3D collision detection: 3D collision detection in virtual reality and 3D games are the common technique used in the literature [25], which is based on collision detection to realize the physical phenomena. However, in our approach, a subspace was regarded as another 3D model or object, and a collision between two objects was measured. Then, each subspace with or without a collision was labeled to obtain a sampling matrix. Figure 3 shows an explanation for why we required collision detection between a 3D model and its sampling subspaces.
2. The candidate subspace for collision detection: as 3D models are usually presented by irregular shapes, high computational costs are usually required for precise collision detection between objects. In the literature, the most efficient method is AABB collision detection [26]. It uses the most closely placed cuboids to replace an irregular 3D model to perform complex and cumbersome collision detection. However, directly applying this method to our approach proved difficult. This is because if the re-sampling scope is too large, it will cause the loss of many features in detail. On the other hand, although the AABB method could not be used to fully detect a collision between objects, it could effectively reduce the number of spaces that we needed for the detection. Therefore, we used the AABB method to find out the most similar rectangular space with the smallest size for the surface of each 3D model, which was then used to map into the subspace to find candidate subspaces. In short, the rectangular space is a set of subspaces, which could show possible collision in the surface. Figure 4 shows an example of AABB collision detection.
3. Fast collision detection: most collision detection algorithms try to minimize the number of primitive–primitive intersections that need to be computed. A fast and reliable method for computing the primitive–primitive intersection is desired. Since rendering hardware is often targeted for triangles, the primitives in collision detection algorithms are often triangles as well [16].

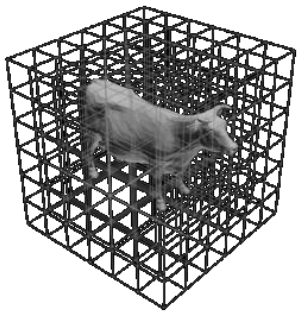


Figure 3. The concept of the CSS method. Each greys cube is the subspace of a 3D model, and each has an associated label to show whether they contain a collision in the 3D model.

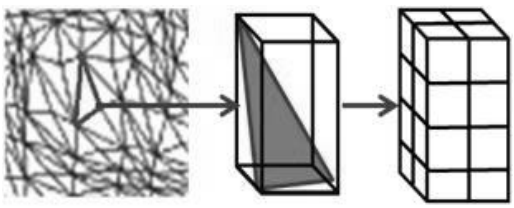


Figure 4. The AABB Collision detection method. Each small black cube is the unit of sampling (cube space).

For this problem, "A Fast Triangle-Triangle Intersection Test" method was applied [27] to execute the task of collision detection for accelerating the efficiency of sampling.

3.3. Feature Rperesentation by CSS Data Transformation

In order to retain the information of CSS, we did not use a complex feature extraction method to produce feature vectors of 3D models. On the contrary, we performed some simple statistical calculations on each 'CSS data'. After these calculations were completed, we obtained a set of data, which could be directly used for 3D model classification. In other words, CSS data could be converted into a group of features as 3D model feature descriptors for classification. In this paper, we propose six different calculation methods as follows. Note that in our experiments present in Section 4, we set our CSS sampling to 50.

- Hollow feature

$$Null\ Feature_{ry} = \sum_{k=1}^{n} \left(S_{ijk} \right) \tag{2}$$

$$Null\ Feature_{ry} = \sum_{k=1}^{n} \left(S_{ikj} \right) \tag{3}$$

$$Null\ Feature_{ry} = \sum_{k=1}^{n} \left(S_{kij} \right) \tag{4}$$

where n is the number of sampling, M is the total number of 3D models, and S_{ijk} is the subspace ($i = 1, 2 \ldots n$, $j = 1, 2 \ldots n$, $y = i + n^{(j-1)}$, $r = 1, 2 \ldots M$)

Hollow Feature: the typical CSS feature. This is the same as general 3D objects, which only have an empty shell or the skin. So, we call them "Hollow". We calculated the sum of the cross-section of each dimension to form a simple feature.

- Hollow + TF-IDF

Transform the "Hollow Feature" matrix into the TFIDF matrix.

Hollow + TFIDF: We applied the term frequency–inverse document frequency (TF-IDF) methods on the typical "Hollow" feature.

- Hollow + ^2

Square of each "Hollow Feature" vector.

Hollow + ^2: In order to increase the difference between each column, we squared each "Hollow Feature" vector.

- MMF

n is the number of sampling.
For each dimensions
  For $i = 1, 2 \ldots n$
    For $j = 1, 2 \ldots n$
      $y = j + n^{(i-1)}$
      □MD_y = max distance between two points in row S_{ij1}~S_{ijn}
      □ND_y = min distance between two points in row S_{ij1}~S_{ijn}
      □OD_y = the distance between the origin and the max distance.
    End
  End
End
Combine the features MD, ND and OD.
Obtain a set of MMF features from a 3D model.

MMF: this is also a typical CSS feature. Different from the "Hollow", we applied the other method to calculate the sum of each cross section.

- MMF + ˆ2

Square of each "MMF Feature" vector.

MMF + ˆ2: similarly, to increase the difference between each column, we squared each "MMF Feature" vector.

- Full

Based on "Hollow Feature" data to fill the vacancies within the model.

Full: due to the "Hollow" feature internal being empty, we attempted to fill the hollow part to form a new type of feature, namely "Full".

3.4. An Example of CSS

Here we show an example for the CSS process. Figure 5 shows a 3D model which has not yet been processed for re-sampling.

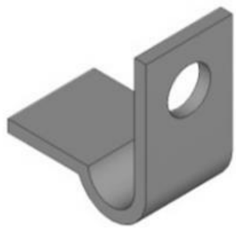
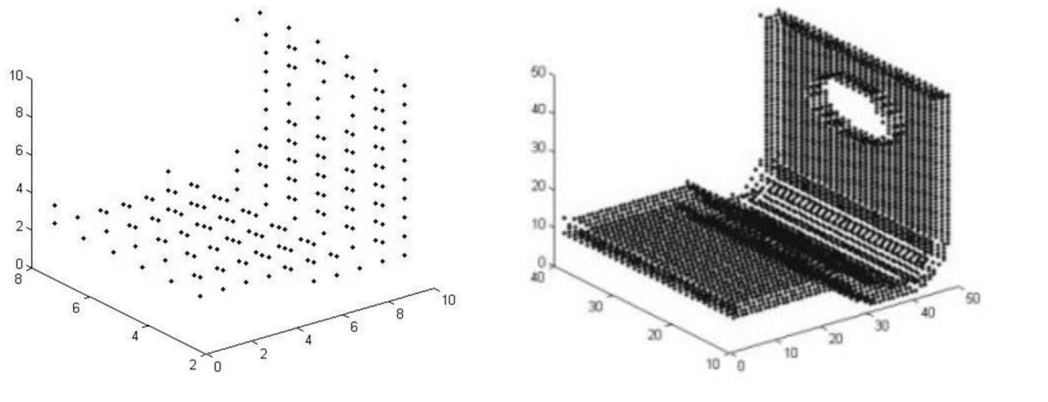


Figure 5. An original 3D model.

In order to examine the effect of sampling, we set the sampling number N to 10 and 50 individually, and then performed re-sampling for this object. The sampling results are shown in Figure 6a,b. Finally, a feature vector was produced through a simple sum for this object as shown in Figure 7.



(**a**) (**b**)

Figure 6. The results of using different sampling numbers. (**a**) N is 10; (**b**) N is 50.

0	0	0	5	2	1	5	2	1	5	2	1	0	0	0	0	0	0	5	2	1	4	2	2	5	3	1	0	0	0	0	0	0	1	1	5	1	1	5	1	1	5	0	0	0	0	0	0	1	1
0	0	0	2	2	1	1	1	1	2	2	1	0	0	0	0	0	0	2	2	1	2	2	1	1	1	1	0	0	0	0	0	0	2	2	1	2	2	1	2	2	1	0	0	0	0	0	0	1	1
3	2	2	3	3	2	0	0	0	0	0	0	0	0	0	3	2	2	2	2	2	0	0	0	0	0	0	0	0	0	3	2	2	3	2	2	0	0	0	0	0	0	0	0	0	3	2	2	0	0

Figure 7. Feature representation of the 3D model (based on N = 50 and Hollow feature calculation).

4. Experiments

4.1. The Dataset

In this paper, we used the ESB (Engineering Shape Benchmark) [20] dataset (https://engineering.purdue.edu/cdesign/wp/downloads/, accessed on 23 November 2021) for

the experiments. The dataset is used for evaluating shape-based search methods relevant to the mechanical engineering domain. In 3D model retrieval, the ESB dataset is the commonly used benchmark. This dataset is based on the 3D component models of computer-aided design, and it contains 867 3D CAD models. In addition, there are 45 different classes in ESB.

4.2. The Baseline

Since we wanted to prove that the voxel-based feature is robust against the rotation problem, we compared it to the most well-known method, LFD (LightField Descriptor) [20,21]. the LFD method does not need the additional process or methods for rotation. Therefore, we compared CSS with LFD in terms of classification and retrieval.

LFD is a moment- type of features, its concept being to use many different angles to obtain different 2D images. Then, some features are extracted from these images for 3D model retrieval.

In the LFD approach, each 3D model had a set of LightField Descriptors, and each set of LightField Descriptors was composed of ten different camera systems. Each LightField Descriptor had 20 viewpoints (i.e., 20 images). Then, the ten most important pictures were chosen to represent the LightField Descriptor (each image was based on $256 * 256$ pixels). Therefore, each 3D model must render 100 images from 10 different camera systems, in total.

After feature extraction, we obtained some different features, which were: ART feature (ART); circularity feature (CIR); color descriptor (CCD); eccentricity feature (ECC); and Fourier descriptor feature (FD). For detailed information, please refer to [4,28].

4.3. The Classification Model

In order to assess the performance of CSS, the support vector machine (SVM) classifier was constructed for 3D model classification. SVM is the widely used classification technique for many pattern recognition problems [29]. In this paper, LibSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/, accessed on 23 November 2021) was used. Specifically, we used the polynomial kernel function to construct the SVM classifier. In addition, the parameter setting for Cost was set to 128 and the Gamma value was set to 0.65.

Moreover, since the number of each class is significantly different, to make a more reliable conclusion, we used k-fold cross-validation by k = 2, 5, 10, 15, and 20 (The experimental environments are as follows: CPU: Intel(R) Core(TM) i7-3770 @ 3.40GHz, RAN: 32GB, OS: Windows 7-64bit, Code: Matlab R2012a).

4.4. Experimental Results

In Table 1, we observed that among all of the features, the CSS based feature, "Hollow + TFIDF", had the best classification accuracy. In comparison with the baseline, using "Hollow" and "Hollow + TFIDF" features performed better, with a significant level of performance difference based on the Wilcoxon rank-sum test ($p < 0.05$). Although using the "Full" feature provides similar accuracy to the baseline, it still performed slightly better.

We used two sets of different values (5, 20, 60, 140, 220, 300, 380, 460, 540, 620, 700, and 859) and (1, 3, 5, 7, 9, 11, 13, and 15) to calculate the precision and recall rates of each feature, respectively. In this experiment, for simplicity, the Euclidean distance was used as a similarity measure to distinguish between 3D models. The results of precision and recall are shown in Figures 8 and 9, respectively.

Table 1. Classification accuracy of each feature representation method.

	2-Fold	5-Fold	10-Fold	15-Fold	20-Fold
Baseline	33.53%	42.61%	44.82%	46.33%	47.03%
Hollow	*37.60%*	*47.38%*	*50.76%*	*52.15%*	*52.27%*
Hollow + TFIDF	*37.49%*	*49.48%*	*52.85%*	*54.25%*	*54.60%*
Hollow + ^2	25.73%	34.46%	36.32%	37.72%	37.72%
Full	*33.53%*	*43.77%*	*46.92%*	*48.54%*	*48.31%*
MMF	28.17%	37.95%	40.16%	41.68%	41.33%
MMF + ^2	24.91%	33.41%	35.27%	36.44%	36.44%

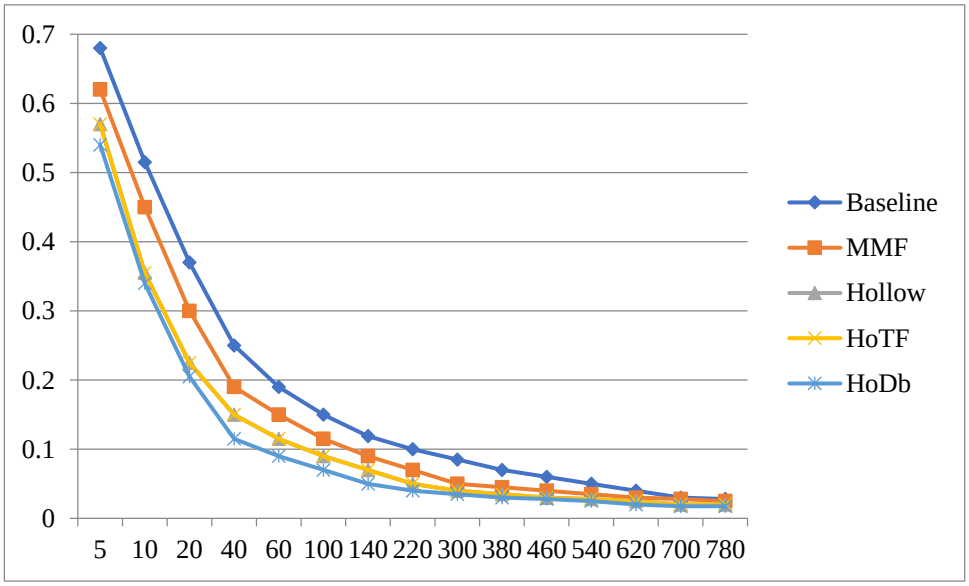


Figure 8. The precision rate of each feature representation method.

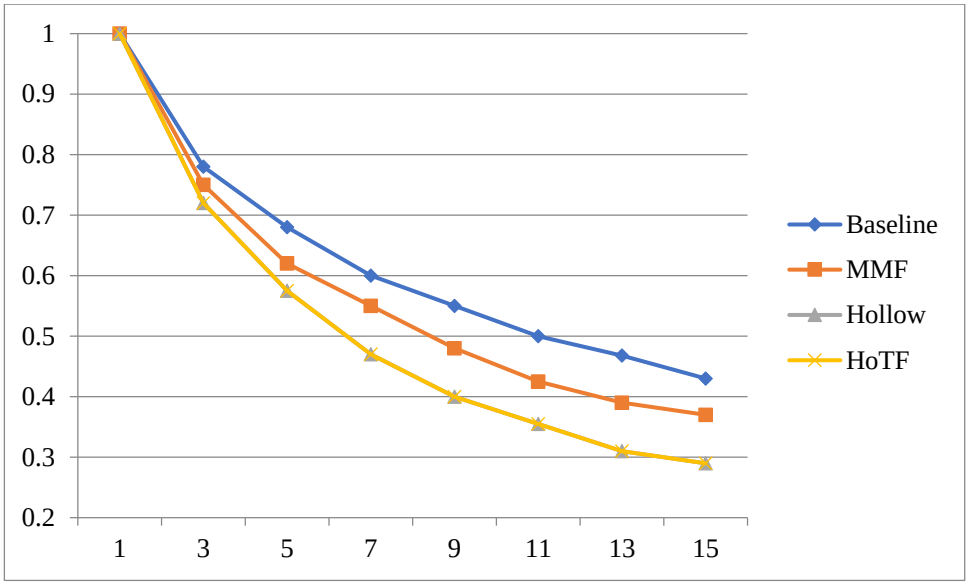


Figure 9. The recall rate of each feature representation method.

These results show that the baseline performed slightly better than CSS. However, since the proposed method was only based on simple statistics, the CSS features still

contained a lot of noise and non-critical data, which was likely to be affected by the Euclidean distance leading to poorer precision and recall.

Nevertheless, considering the classification accuracy, using the SVM classifier can effectively highlight the discriminative power of CSS features; thus, better results can be obtained.

5. Conclusions

In this paper, we propose a novel approach for 3D model re-sampling, namely CSS (Cube of Space Sampling). It is a kind of voxelization method and can be transformed into the voxel-based feature for 3D model classification and retrieval. Particularly, the voxel-based feature via the CSS method can represent 3D models well, and it can also solve the 3D model normalization problem (e.g., rotation problem). The proposed CSS method can work on the non-completely closed 3D models without failure. Therefore, the CSS method can be fully applied to all types of 3D models.

The experimental results based on the ESB dataset show that the proposed method is indeed able to effectively extract useful features for 3D model retrieval. More specifically, the CSS features can improve classification accuracy by SVM when compared with a state-of-the-art technique. However, our proposed approach does not aim to replace the existing features, but to provide new data to extract important features. Therefore, the CSS features provide an opportunity for future research, which could use a different form of data for feature extraction and increase the accuracy of classification. On the other hand, this proposed method still has some shortcomings. For example, the computational cost is high during feature extraction.

There are several issues to be addressed in future work. In this study, since the CSS data were only based on some simple statistics to create a set of features, other methods of dealing with our CSS data to highlight the characteristics of each model (such as FFT, etc.) could be applied. In addition, other existing features (e.g., Skeleton) could be combined with the CSS features to further increase the classification accuracy and retrieval effectiveness. On the other hand, as the processing time of the proposed approach is longer than the baseline, some more efficient strategies for space sampling need to be proposed with regards to computational cost during feature extraction. Similarly, for the index issue, the 3D spatial index could be considered to speed up the calculation. Moreover, further performance comparisons could be conducted based on some deep-learning techniques, such as convolutional neural networks [30–32] and other feature representations, such as view-based 3D model feature representation [33,34]. Finally, it would be interesting to examine the performance of CSS for different types of 3D objects, such as objects composed of multiple parts that are not watertight, and the sensitivity of CSS when applied to arbitrarily rotated models, or swapping the XYZ axes.

Author Contributions: Conceptualization, Z.-Y.C.; methodology, Z.-Y.C., C.-F.T. and W.-C.L.; software, W.-C.L.; validation, Z.-Y.C. and W.-C.L.; formal analysis, Z.-Y.C., C.-F.T. and W.-C.L.; resources, Z.-Y.C.; data curation, Z.-Y.C.; writing—original draft preparation, Z.-Y.C., C.-F.T. and W.-C.L.; writing—review and editing, Z.-Y.C., C.-F.T. and W.-C.L.; supervision, C.-F.T.; project administration, C.-F.T.; funding acquisition, W.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology of Taiwan, grant MOST 110-2410-H-182-002 and Chang Gung Memorial Hospital, Linkou, grant BMRPH13 and CMRPG3J0732.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 110-2410-H-182-002 and in part by Chang Gung Memorial Hospital, Linkou, under Grant BMRPH13 and CMRPG3J0732.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tangelder, J.W.H.; Veltkamp, R.C. A Survey of Content Based 3D Shape Retrieval Methods. In Proceedings of the Shape Modeling International, Genova, Italy, 7–9 June 2004; pp. 145–156.
2. Zhu, K.; Wong, Y.; Lu, W.; Loh, H. 3D CAD model matching from 2D local invariant features. *Comput. Ind.* **2010**, *61*, 432–439. [CrossRef]
3. Bustos, B.; Keim, D.; Saupe, D.; Schreck, T. Content-based 3D object retrieval. *IEEE Eng. Med. Boil. Mag.* **2007**, *27*, 22–27. [CrossRef] [PubMed]
4. Laga, H.; Nakajima, M. A boosting approach to content-based 3D model retrieval. In Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia, GRAPHITE '07, Perth, Australia, 1–4 December 2007; ACM Press: New York, NY, USA, 2007; pp. 227–234.
5. Eisemann, E.; D'ecoret, X. Fast Scene Voxelization and Applications. In Proceedings of the ACM Symposium on Interactive 3D Graphics and Games, Redwood City, CA, USA, 14–17 March 2006.
6. Huang, J.; Yagel, R.; Filippov, V.; Kurzion, Y. An Accurate Method for Voxelizing Polygon Meshes. In Proceedings of the 1998 IEEE Symposium on Volume Visualization, VVS-'98, Research Triangle Park, NC, USA, 19–20 October 1998; ACM Press: New York, NY, USA, 1998; pp. 119–126.
7. Schwarz, M.; Seidel, H. Fast parallel surface and solid voxelization on GPUs. *ACM Trans. Graph.* **2010**, *29*, 1–10. [CrossRef]
8. Zhang, C.; Chen, T. Indexing and retrieval of 3D models aided by active learning. In *ACM Multimedia*; ACM Press: New York, NY, USA, 2001; pp. 615–616.
9. Paquet, E.; Murching, A.; Naveen, T.; Tabatabai, A.; Rioux, M. 2000. Description of shape information for 2-D and 3-D objects. *Signal. Process. Image Commun.* **2000**, *16*, 103–122. [CrossRef]
10. Kazhdan, M.; Chazelle, B.; Dobkin, D.; Funkhouser, T.; Rusinkiewicz, S. A reflective symmetry descriptor for 3D models. *Algorithmica* **2004**, *38*, 201–225. [CrossRef]
11. Osada, R.; Funkhouser, T.; Chazells, B.; Dobkin, D. Matching 3D models with shape distributions. In Proceedings of the Shape Modeling International, Genoa, Italy, 7–11 May 2001.
12. Ohbuchi, R.; Takei, T. Shape-similarity comparison of 3D models using alpha shapes. In Proceedings of the 11th Pacific Conference on Computer Graphics and Applications, Canmore, AB, Canada, 8–10 October 2003.
13. Ip, C.Y.; Lapadat, D.; Sieger, L.; Regli, W.C. Using shape distributions to compare solid models. In Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, Saarbrücken, Germany, 17–21 June 2002; ACM Press: New York, NY, USA, 2002; pp. 273–280.
14. Zaharia, T.; Preteux, F.J. 3D shape-based retrieval within theMPEG-7 framework. In *Nonlinear Image Processing and Pattern Analysis XII*; International Society for Optics and Photonics: Bellingham, WA, USA, 2001; Volume 4034, pp. 133–145.
15. Koendering, J. *Solid Shape*; MIT Press: Cambridge, MA, USA, 1990.
16. Vranic, D.V.; Saupe, D.; Richter, J. Tools for 3D-object retrieval: Karhunen-loeve transform and spherical harmonics. In Proceedings of the IEEE 2001 Workshop Multimedia Signal Processing, Cannes, France, 3–5 October 2001; pp. 293–298.
17. Biasotti, S.; Marini, S.; Mortara, M.; Patane, G.; Spagnuolo, M.; Falcidieno, B. 3D shape matching through topological structures. In Proceedings of the International Conference on Discrete Geometry for Computer Imagery, Naples, Italy, 19–21 November 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 194–203.
18. Sundar, H.; Silver, D.; Gagvani, N.; Dickenson, S. Skeleton based shape matching and retrieval. In Proceedings of the 2003 International Conference on Shape Modelling International, Seoul, Korea, 12–16 May 2003; pp. 130–139.
19. Novotni, M.; Klein, R. A geometric approach to 3D object comparison. In Proceedings International Conference on Shape Modeling and Applications, Cambridge, MA, USA, 13–17 June 2001; pp. 154–166.
20. Yang, Y.; Lin, H.; Zhang, Y. Content-Based 3-D Model Retrieval: A Survey. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2007**, *37*, 1081–1098. [CrossRef]
21. Vranic, D.V.; Saupe, D. Description of 3D-shape using a complex function on the sphere. In Proceedings of the IEEE International Conference Multimedia and Expo, Lausanne, Switzerland, 26–29 August 2002; pp. 177–180.
22. Min, P.; Halderman, A.; Kazhdan, M.; Funkhouser, A. Early experiences with a 3D model search engine. In Proceedings of the Web3D Symposium, Saint Malo, France, 9–12 March 2003; ACM Press: New York, NY, USA, 2003; pp. 7–18.
23. Funkhouser, T.; Min, P.; Kazhdan, M.; Chen, J.; Halderman, A.; Dobkin, D.; Jacobs, D. A search engine for 3D models. *ACM Trans. Graph.* **2003**, *22*, 83–105. [CrossRef]
24. Ankerst, M.; Kastenmuller, G.; Kriegel, H.; Seidl, T. Nearest neighbor classification in 3D protein databases. In Proceedings of the ISMB, Heidelberg, Germany, 6–10 August 1999; pp. 34–43.
25. Jiménez, P.; Thomas, F.; Torras, C. 3D collision detection: A survey. In Proceedings of the 9th Pacific Conference on Computers & Graphics, Tokyo, Japan, 16–18 October 2001; pp. 269–285.
26. Klosowski, J.T.; Held, M.; Mitchell, J.S.B.; Sowizral, H.; Zikan, K. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. Vis. Comput. Graph.* **1998**, *4*, 21–36. [CrossRef]
27. Möller, T. 1997. A fast triangle-triangle intersection test. *J. Graph. Tools* **1997**, *2*, 25–30. [CrossRef]

28. Chen, D.-Y.; Tian, X.-P.; Shen, Y.-T.; Ouhyoung, M. On visual similarity based 3D model. *Comput. Graph. Forum* **2003**, *22*, 223–232. [CrossRef]
29. Fan, R.-E.; Chen, P.-H.; Lin, C.-J. Working set selection using second order information for training SVM. *J. Mach. Learn. Res.* **2005**, *6*, 1889–1918.
30. Ding, B.; Tang, L.; He, Y.-J. An Efficient 3D Model Retrieval Method Based on Convolutional Neural Network. *Complexity* **2020**, *2020*, 1–14. [CrossRef]
31. Gao, Z.; Li, Y.; Wan, S. Exploring Deep Learning for View-Based 3D Model Retrieval. *ACM Trans. Multimedia Comput. Commun. Appl.* **2020**, *16*, 1–21. [CrossRef]
32. Hoang, L.; Lee, S.-H.; Kwon, K.-R. A Deep Learning Method for 3D Object Classification and Retrieval Using the Global Point Signature Plus and Deep Wide Residual Network. *Sensors* **2021**, *21*, 2644. [CrossRef] [PubMed]
33. Wang, D.; Wang, B.; Yao, H.; Liu, H. Center-push loss for joint view-based 3D model classification and retrieval feature learning. *Signal Image Video Process.* **2021**. [CrossRef]
34. Li, W.; Su, Y.; Zhao, Z.; Hao, T.; Li, Y. Exploring contextual information for view-wised 3D model retrieval. *Multimedia Tools Appl.* **2021**, *80*, 16397–16412. [CrossRef]