A Retrieval Method for 3D CAD Assembly Models Using 3D Radon Transform and Spherical Harmonic Transform

Kaoru KATAYAMA\(^{a)\}, Member and Takashi HIRASHIMA\(^{b)\}, Nonmember

SUMMARY We present a retrieval method for 3D CAD assemblies consisted of multiple components. The proposed method distinguishes not only shapes of 3D CAD assemblies but also layouts of their components. Similarity between two assemblies is computed from feature quantities of the components constituting the assemblies. In order to make the similarity robust to translation and rotation of an assembly in 3D space, we use the 3D Radon transform and the spherical harmonic transform. We show that this method has better retrieval precision and efficiency than targets for comparison by experimental evaluation.

key words: 3D CAD model, assembly, retrieval, layout of component, 3D Radon transform, spherical harmonic transform

1. Introduction

Constructing 3D CAD assembly models in designing products becomes popular as inexpensive computational resources can be accessed easily. A 3D CAD assembly model consists of multiple components which have various shapes and are made by various materials. We call simply such 3D CAD assembly model an assembly. In this paper, we present a retrieval method using the 3D Radon transform and the spherical harmonic transform for assemblies with different layouts of the components. The layout of components constituting an assembly is an important factor since it includes technical knowledge obtained from previous product design. Designers can avoid constructing assemblies with the same defects as existing models by comparing newly created assemblies with them. Although there are a lot of studies on retrieval for 3D CAD models, they are mainly focused on distinguishing the shapes of the whole models or their components [1]–[4]. The proposed method can distinguish both the layouts and the shapes of the components constituting assemblies in addition to the shapes of the whole assemblies. The components with the same shape and different properties such as materials are also distinguished. Figure 1 shows the examples of assemblies called Pump and Gear. Pump 1 and Pump 2 of Fig. 1 (a) and Fig. 1 (b) have the same shape but different layouts of the components. The different colored components are made of different materials. The shape of Gear of Fig. 1 (c) is different from the shapes of Pump 1 and Pump 2. We want to distinguish these three assemblies.

We call a set of components which have the same shape and are made by the same material a subassembly.

Given an assembly as a query, we compute similarity between the query and each of assemblies in a database and rank the assemblies in the database according to their similarity. We call an assembly given as a query and one in a database as a query model and a database model, respectively. We define a similarity between a query model and a database model by the sum of similarities between their subassemblies. Since a query model and a database model do not have the same positions and orientations generally in 3D space, we need a robust similarity between their subassemblies to such variations. We compute such similarity by two methods. One is a numerical method using the spherical harmonic transform of feature quantities for each of subassemblies. Another is a combinational method by matching the feature quantities of two subassemblies. The feature quantity for a subassembly is computed from its projections, that is, the 3D Radon transform or the projection to 2D planes. Each of the projections is constructed by combining the projection of the subassembly itself and the projection of the other subassemblies constituting the assembly to reflect both the shapes and layouts of the components and their positions in the whole assembly to the feature quantity. The projections of the subassembly and the other subassemblies are normalized by dividing their respective volumes.

We evaluate these methods experimentally using the assemblies which we prepare as a benchmark data. Each of the assemblies is represented as a set of polygons. The proposed methods can be applied to assemblies represented in the other formats such as volumetric models if we can compute the projections of their subassemblies. The results of the experiments show that the method using the 3D Radon transform and the spherical harmonic transform has better retrieval precision and efficiency than targets for compari-
2. Related Work

There are a lot of works on retrieval for general 3D models including 3D CAD models. Bustos et al. [1], Iyer et al. [2], Tangelder et al. [3] and Li et al. [4] give overviews of the existing retrieval methods for general 3D models. Most of the retrieval methods for general 3D models focus on distinguishing their shapes. They do not consider assembly structures of 3D CAD assemblies. Although there are not many works on retrieval for 3D CAD assemblies, Lupinetti et al. [5] provide a comprehensive review on recent works. They analyze and categorize the existing retrieval techniques and their background such as user requirements from several criteria. In Chen et al. [6], Deshmukh et al. [7] and Miura et al. [8], an assembly is represented as a hierarchical graph or a graph and the similarity between two assemblies is computed using graph matching algorithms. It is difficult to represent geometric layouts of components by graphs. Hu et al. [9] represent an assembly as a vector in the vector space. The vector consists of the feature quantities of their components. These existing methods for retrieving 3D CAD assemblies do not also satisfy our requirement for detection of differences in geometric layouts of components constituting 3D models with the same shapes.

The spherical harmonic transform is often used to represent a shape of a 3D model as a vector in some high dimensional vector space [10]–[14]. Such vector is called a shape descriptor. López et al. [15] give a comprehensive overview of studies on shape descriptors for 3D models. Our requirement for retrieving assemblies is to distinguish assemblies with the same shape by layouts and properties of their components. It cannot be achieved by using the existing shape descriptors for 3D models as they are. A 3D model is defined as a set of functions on concentric spheres in Saupe et al. [10], Vranic et al. [11], Kazhdan et al. [12] and Papadakis et al. [14]. These are normalized for translation by moving the center of its mass to the origin in advance for example. Then the shape descriptor of a 3D model is computed by the spherical harmonic transform of each of the functions. On the other hand, we define the absolute values of the Fourier coefficients as a set of functions on a sphere. It is not necessary to normalize a 3D model for translation. Although Papadakis et al. [14] also propose the shape descriptor using the 3D Radon transform, it is different from ours since we normalize the result of the 3D Radon transform to reflect not only the shape and the layout of the components but also the location of a subassembly in the assembly containing it.

The existing benchmark data of 3D models are summarized by Savelonas et al. [16]. Since there are no suitable benchmark data in them for evaluating our retrieval method for assemblies with different layouts of the components, we prepare 3D assemblies for evaluation.

We previously developed another retrieval method for assemblies to distinguish the different layouts of the components [17]. The projection onto planes and the combinational method to compute the distance between subassemblies, which are described in Sect. 3.1.2 and Sect. 3.2.2, respectively, are close to our previous method. Therefore we evaluate retrieval precision and efficiency of the method using the 3D Radon transform and the spherical harmonic transform, which are described in Sect. 3.1.1 and Sect. 3.2.1, respectively, by comparing with them experimentally.

3. Proposed Method

We retrieve the database model with the same shapes and layouts of the components as a given query model by ranking the database models according to their dissimilarities to the query model. The dissimilarity between a query model and a database model is computed in the following procedure.

1. Compute normalized projections for each of the subassemblies constituting the query model and the database model
2. Compute feature quantities for each of the subassemblies from their projections
3. Compute dissimilarities among the subassemblies of the query model and those of the database models from their feature quantities, which are robust to translations and rotations of the both models in 3D space
4. Compute dissimilarities between the query model and the database model from the dissimilarities between their subassemblies

The database model with the minimum dissimilarity to the query model is returned as the answer to the query. The projections and the feature quantities for subassemblies of database models can be computed in advance. We call feature quantity and dissimilarity feature and distance, respectively, below. In order to distinguish differences in shapes and layouts between components of a query model and those of a database model, we compute the distance between a
query model and a database model from the distances between their subassemblies. We denote a set of subassemblies constituting an assembly model \( m \) by \( S(m) \). The superscripts such as \( X^1, X^2 \) and \( Y^2 \) in the input parameters to the following algorithms are suffixes and do not mean exponents.

3.1 Normalized Projection of Assembly for Distinguishing Layouts of Components

We propose two kinds of normalized projections for a subassembly \( s \in S(m) \). Each of the projections is constructed as sum of the projection of \( s \) itself and the projection of the other subassemblies \( S(m) \setminus s \) to reflect not only the shape and the layout of the components constituting \( s \) but also the location of \( s \) in assembly \( m \) to the projection. In order to incorporate the feature of \( s \) into the feature of the whole assembly \( m \), the projection of \( s \) itself and the projection of \( S(m) \setminus s \) are normalized by dividing by the volumes of \( s \) and \( S(m) \setminus s \), respectively. The most memory space in the proposed methods is temporarily necessary for computing the normalized projection of \( s \) since the projections of all subassemblies \( S(m) \) constituting \( m \) are needed. In order to obtain a robust feature of \( s \) to translation and rotation of \( m \), we compute the Radon transform or the Fourier transform of normalized projection of \( s \) onto a line.

3.1.1 Projection onto Lines

The projection of a subassembly \( s \) onto a line is computed from normalized cross-sectional areas of \( s \) and \( S(m) \setminus s \) by a set of planes perpendicular to the x-axis in 3D space. Figure 2 (a) illustrates the projection of a subassembly onto a line. For each of the degrees specified by longitudes \( \Lambda \) and latitudes \( \Phi \), we rotate the assembly consisted of \( s \) and \( S(m) \setminus s \), and compute their cross-sectional areas by the set of planes. The planes are specified by the x-coordinates \( x^1 \) of the intersection points with the x-axis. This means to compute the 3D Radon transform of \( s \) and \( S(m) \setminus s \). For the plane whose x-coordinate of the intersection point is \( x \in X^1 \), a projection \( np(s, x) \) of \( s \) is computed from the normalized cross-sectional areas of \( s \) and \( S(m) \setminus s \) as follows.

![Fig. 2 Two kinds of projection of subassembly](image)

Algorithm 2 ProjectOntoLine\((m, x^1, N^1)\)

**Input:** an assembly model \( m \), a set \( X^1 \) of x-coordinates, a positive integer \( N^1 \) to specify the highest frequency of the stored Fourier coefficients.  
**Output:** projection \( afp \) of \( m \)

1: for all \( s \in S(m) \) do  
2: for all \( x \in X^1 \) do  
3: \( a(s, x) \leftarrow \) the cross-sectional area of \( s \) by the plane perpendicular to the x-axis at the point whose x-coordinate is \( x \)
4: end for  
5: end for  
6: for all \( s \in S(m) \) do  
7: for all \( x \in X^1 \) do  
8: \( np(s, x) \leftarrow \frac{a(s, x)}{\sum_{s' \in S(m) \setminus s} a(s', x)} + \frac{\sum_{s' \in S(m) \setminus s} a(s', x)}{\sum_{s' \in S(m) \setminus s} a(s', x)} \)  
9: end for  
10: \( afp \leftarrow \) the absolute values of the Fourier coefficients from the lowest to the \( N^1 \)th frequency in computing the Fourier transform of \( np \) for \( s \) along \( X^1 \)  
11: end for  
12: return \( S(m) \times N^1 \) array \( afp \)

![Fig. 3 Normalization in projection onto plane](image)

\[
np(s, x) = \frac{a(s, x)}{\sum_{s' \in X^1} a(s', x)} + \frac{\sum_{s' \in S(m) \setminus s} a(s', x)}{\sum_{s' \in S(m) \setminus s} a(s', x)}
\]

where \( a(s, x) \) is the cross-sectional area of \( s \) by the plane whose intersection point with the x-axis is \( x \). In order to compute \( np \) for a subassembly, we need the cross-sectional areas of all subassemblies, which are stored in the array \( a \). The spacial efficiency of this algorithm depends on the size \( |S(m)| \times |X^1| \) of \( a \).

When an assembly \( m \) is translated in 3D space, the projections of its subassembly \( s \) onto lines are also translated accordingly. In order to obtain a robust feature to such translation, we compute the Fourier transform of the projection \( p(s, x) \) along \( x \) and the absolute values of the Fourier coefficients. The absolute values from the lowest frequency to the \( N^4 \)th frequency are stored in \( |S(m)| \times N^1 \) array \( afp \). Algorithm 2 describes these processes formally.

3.1.2 Projection onto Planes

A projection of subassembly \( s \) onto a plane is computed from a set of normalized lengths of intersection of \( s \) and \( S(m) \setminus s \) with each of lines perpendicular to the x-y plane in 3D space. Figure 2 (b) illustrates the projection of a sub-
assembly onto a plane. For each of the degrees specified by
longitudes \( \Lambda \) and latitudes \( \Phi \), we rotate the assembly con-
sisted of \( s \) and \( S(m) \setminus s \), and compute the lengths of inter-
section of \( s \) and \( S(m) \setminus s \) with the lines. Each of the lines is
specified by the x-y coordinate \((x, y)\) in \( X^2 \times Y^2 \) which is its
intersection point with the x-y plane. The length of the inter-
section of \( s \) with the line specified by \((x, y)\) is denoted by
\( l(s, x, y) \). A projection \( np(x, y) \) of \( s \) in \((x, y)\) is computed
from the normalized lengths of intersection of \( s \) and \( S(m) \setminus s \) as follows.

\[
np(x, y) = \frac{\sum_{(x', y') \in X^2 \times Y^2} l(x', y') \times s \in \{x = x', y = y\}}{\sum_{(x', y') \in X^2 \times Y^2} l(x', y') \times s \in \{x = x', y = y\}}.
\]

In order to compute \( np \) for a subassembly, we need the
lengths of the intersections at all points in \( X^2 \times Y^2 \) of all
subassemblies, which are sorted in the array \( l \). The spacial
efficiency of this algorithm depends on the size \(|X^2| \times |Y^2| \times
|S(m)| \times l |l| .

According to the position and the orientation of assembly \( m \) in 3D space, projection \( np \) translates and rotates on the
projection plane. In order to obtain a robust feature to such
rotation and translation, we use the 2D Radon transform and
the Fourier transform. The discrete 2D Radon transform of
\( np \) is computed according to specified radii \( R^2 \) and angles
\( \Theta^2 \). We store the result of the discrete 2D Radon transform
as 2D array \( rp \) with size \( 2|R^2| \times |\Theta^2| \) by combining the re-
results in \( \theta \) and \( \theta + \pi \) for each \( \theta \in \Theta^2 \). When \( np \) is translated
and rotated in the projection plane, \( rp \) is shifted toward the
radial direction and the angular direction, respectively. We

### Algorithm 3: ProjectOntoPlane \( (m, X^2, Y^2, \Theta^2, R^2, N_1^2, N_2^2) \)

**Input:** an assembly model \( m \), sets \( X^2 \) and \( Y^2 \) of x-coordinates and y-
coordinates, sets \( \Theta^2 \) and \( R^2 \) of angles and radii for the 2D Radon transform,
positive integers \( N_1^2 \) and \( N_2^2 \) to specify the highest frequencies of
the stored Fourier coefficients

**Output:** projection \( afp \) of \( m \)

1. \( \text{for all } s \in S(m) \) do
2. \( \text{for all } x \in X^2 \) and \( y \in Y^2 \) do
3. \( l(x, x', y) \leftarrow \text{the length of intersection of } s \text{ with the line perpendicular to the x-y plane at point } (x, y, 0) \)
4. \( \text{end for} \)
5. \( \text{end for} \)
6. \( \text{for all } s \in S(m) \) do
7. \( \text{for all } x \in X^2 \) and \( y \in Y^2 \) do
8. \( np(x, x', y) = \frac{\sum_{(x', y') \in X^2 \times Y^2} l(x', y') \times s \in \{x = x', y = y\}}{\sum_{(x', y') \in X^2 \times Y^2} l(x', y') \times s \in \{x = x', y = y\}} \)
9. \( \text{end for} \)
10. \( rp \leftarrow \text{the 2D Radon transform of } np \) \( s \) according to \( R^2 \) and \( \Theta^2 \)
11. \( ffrp \leftarrow \text{the absolute values of the Fourier coefficients from the lowest to the } N_1^2 \text{th frequency by computing the 1D Fourier transform of } rp \) \( \text{along } \Theta^2 \)
12. \( ffrp \leftarrow \text{the absolute values of the Fourier coefficients from the lowest to the } N_2^2 \text{th frequency by computing the 1D Fourier transform of } ffrp \) \( \text{along } \Theta^2 \)
13. \( \text{end for} \)
14. \( \text{return } |S(m)| \times N_1^2 N_2^2 \) array \( afp \) constructed from \( \text{from 3D array } ffrp \)

### Algorithm 4: ComputeSubDistance \( (p^d, p^r, \Lambda, \Phi, L, N) \)

**Input:** outputs \( p^d, p^r \) of Algorithm 2 or Algorithm 3 for a database model and a query model, a set \( \Lambda \) of longitudes, a set \( \Phi \) of latitudes, a positive integer \( L \) to specify the highest degree of the spherical harmonic function, a positive integer \( N \) which is \( N_1^2 \) or \( N_2^2 \) specified in Algorithm 2 or Algorithm 3

**Output:** distances \( sd \) between subassemblies of \( s \) and \( q \)

1. \( \text{for all } (s', p') \in S(m^d) \times S(m^r) \) do
2. \( sp^d \leftarrow \text{ExpandBySphericalHarmonics}(s', p^d, \Lambda, \Phi, L, N) \)
3. \( sp^r \leftarrow \text{ExpandBySphericalHarmonics}(s', p^r, \Lambda, \Phi, L, N) \)
4. \( sd(s', p') \leftarrow \sum_{l=1}^{N} \sum_{m=1}^{L} \left| sp^d(l, m) - sp^r(l, m) \right|^2 \)
5. \( \text{end for} \)
6. \( \text{return } sd \)

### Algorithm 5: ExpandBySphericalHarmonics \( (s, p, \Lambda, \Phi, L, N) \)

**Input:** the output \( p \) of Algorithm 2 or Algorithm 3, a set \( \Lambda \) of longitudes, a set \( \Phi \) of latitudes, a positive integer \( L \) to specify the highest degree of the spherical harmonic function, a positive integer \( N \) which is \( N_1^2 \) or \( N_2^2 \) specified in Algorithm 2 or Algorithm 3

**Output:** result \( sp \) of expanding by the spherical harmonic functions

1. \( \text{for } n = 1 \) to \( N \) do
2. \( \text{for } l = 0 \) to \( L \) do
3. \( \text{Expand function } g_n^\ell \text{ whose value in } (\Lambda, \Phi) \) \( p(\Lambda, \Phi, s, n) \) by the spherical harmonic functions of degree \( l \)
4. \( sp^d(l, m) \leftarrow \sum_{n=1}^{N} \left| g_n^\ell(l, m) \right|^2 \) where \( g_n^\ell(l, m) \) is the coefficient for the spherical harmonic function of degree \( l \) and order \( m \)
5. \( \text{end for} \)
6. \( \text{end for} \)
7. \( \text{return } sp \)

3.2 Distances between Subassemblies

We define two kinds of the distances between two sub-
assemblies which are robust to rotation of assemblies in 3D
space. The distance between two assemblies is computed
from the distances between their subassemblies as described in
Sect. 3.3.

3.2.1 Distance Using Spherical Harmonic Transform

One way to compute a robust distance between subas-
assemblies to rotation in 3D space is to use the spherical
harmonic transform. We regard the absolute values of the Fourier co-
efficients computed from the normalized projections of a
subassembly in Algorithm 2 or Algorithm 3 for each of lon-
gitudes \( \Lambda \) and latitudes \( \Phi \) as values of scalar functions on a
3.2.2 Distance Based on Matching Features Computed from Normalized Projections

We define another distance between subassembly $s^d \in S(m^d)$ and subassembly $s^q \in S(m^q)$ by sum of squared differences between their corresponding features. The feature corresponding to the absolute value of the Fourier coefficient of frequency $n$, which is computed from the normalized projection of a subassembly $s^d \in S(m^d)$ at $(\lambda, \varphi) \in \Lambda \times \Phi$ is denoted by $p^d(\lambda, \varphi, s^d, n)$. The feature of a subassembly $s^q \in S(m^q)$, which is computed from the projection at $(\lambda', \varphi') \in \Lambda \times \Phi$ is denoted by $p^q(\lambda', \varphi', s^q, n)$ in the same manner. The difference $pd(\lambda, \varphi, \lambda', \varphi')$ between the features of $s^d$ and $s^q$ is as follows.

$$pd(\lambda, \varphi, \lambda', \varphi') = \sum_{n \in N} \left( p^d(\lambda, \varphi, s^d, n) - p^q(\lambda', \varphi', s^q, n) \right)^2$$

Here $N$ is $N^1$ or $N_2^1 \times N_2^2$ in the input parameters of Algorithm 2 or Algorithm 3. We compute this distance for all bijections between the projection angles $\Lambda \times \Phi$ for $s^d$ and $s^q$. The distance $sd(s^d, s^q)$ between $s^d$ and $s^q$ is the minimum in the sums of $pd(\lambda, \varphi, \lambda', \varphi')$ for each of the bijections.

$$sd(s^d, s^q) = \min_{\text{bijection } B \text{ between } \Lambda \times \Phi} \sum_{(\lambda, \varphi), (\lambda', \varphi') \in B} pd(\lambda, \varphi, \lambda', \varphi')$$

We use $sd(s^d, s^q)$ by formalizing it as the combinatorial problem called the assignment problem and solving the problem. Algorithm 6 shows these processes.
the assignment problem and solving the problem in the same way as computing the distance between subassemblies by the combinatorial method described in Sect. 3.2.2. Figure 4 illustrates computation of the distance between a database model and a query model. Algorithm 1 shows the whole process.

4. Experimental Evaluation

We propose two kind of projections for a subassembly, ProjectionOntoLines (Algorithm 2) and ProjectionOntoPlanes (Algorithm 3), and two kinds of distances between subassemblies, ComputeSubDistance1 (Algorithm 4) and ComputeSubDistance2 (Algorithm 6), to compute the distance between two assemblies. We compare these methods by retrieval precision and processing time experimentally. As described in Sect. 2, there is not suitable existing methods for comparison except our previous work [17]. Since ProjectionOntoPlane and ComputeSubDistance2 is close to the methods in our previous work, we evaluate retrieval precision and efficiency of the method using ProjectionOntoLines and ComputeSubDistance2 by comparing them with the other methods.

We prepare the assemblies shown in Fig. 5 as a benchmark data set in this experiment since there are no suitable existing benchmark data sets including assemblies with different layouts of the components as described in Sect. 2. This benchmark data set includes the nine kinds of assemblies with different shapes. For each kind of the assemblies, there are five different types of layouts of the components constituting its subassemblies. The layouts of the components for the five kinds of the assemblies are indicated by colors as examples in Fig. 6. Pump 1 of Fig. 6 (p) means type 1 of layout of the components constituting the assembly called a pump, for example. Figure 6 shows the five kinds of assemblies out of the nine kinds of assemblies due to the limited space. Every assembly consists of five subassemblies and is randomly translated and rotated in 3D space. Each component constituting an assembly is represented as a set of polygons in the STL format which is one of standard formats for CAD models. The number of polygons of which an assembly is composed is 1000 unless it is specified otherwise. These assemblies are used as both database models and query models. We construct them by modifying assemblies downloaded from GrabCAD [18]. Table 1 shows the input parameters of Algorithms 1, 2, 3, 4, 5 and 6 in the experiments. We develop all of the programs using MATLAB R2019a on 64bit Windows 10 and use a PC with a Intel Core i7 CPU and 32GB RAM.

4.1 Distances between Different Kinds of Assemblies

Different kinds of assemblies can be distinguished easier than different types of the same kind of assemblies since they have different shapes. We confirm it in this experiment in order to focus on distinguishing the different types of the same kind of assemblies in the following experiments. Fig-
Figure 7 shows average distances between different kinds of assemblies in each of the proposed methods. We compute the average by computing the distance between the same two assemblies 9 times. We use type 1 of each kind of the prepared assemblies as database models and query models in this experiment. Positions and orientations of the assemblies are different every time in computing the distance. The number of the projections and the size of the projection for a subassembly are 72 and 100, respectively. The size of the feature is 36. The other input parameters to the proposed algorithms according to these parameters are as shown in Table 1. The results show that the distance between the query model and the same database model as the query is always shorter than the distance between the query model and a different database model from the query. Different kinds of assemblies can be distinguished by any of the proposed methods.

4.2 Retrieval Precision for Same Kind of Assemblies

We evaluate the proposed methods based on average retrieval precision for the same kind of assemblies with the same shape. Given an assembly as a query, we retrieve the same type of the model as the query from the five types of the database models which are the same kind as the query. For each kind of the assemblies in the benchmark data set, every type of them is given as a query. We perform this retrieval four times for a query model. Positions and orientations of the database models and the query models are different every time. The average retrieval precision for one of the proposed methods is the rate at which the distance between a query model and the same type of the database model is shorter than the distances from the different types of the database model models.

Figure 8 shows the average retrieval precision in varying the number of the projections for each of subassemblies constituting an assembly. Figures 8 (a), 8 (b), 8 (c) and 8 (d) show the results for four sizes of the features, which are specified by input parameters to the proposed algorithms. In their captions, size $N$ of the feature means $N_1$ for ProjectionOntoLine or $N_1N_2$ for ProjectionOntoPlane. The size of the projection for a subassembly is fixed to 100. The input parameters to the proposed algorithms are as shown in Table 1. The combination of ProjectionOntoLine and ComputeSubDistance1 using the 3D Radon transform and the spherical harmonic transform has the highest retrieval precision except when the number of the projections is 8. When the number of the projections is more than 32 or the size of the feature is more than 16, they do not affect the retrieval precision significantly. The reason may be that they are enough for capturing the features of the assemblies prepared as the benchmark data set.

Figure 9 shows the average retrieval precision in varying the size of the projection for a subassembly. Figures 9 (a), 9 (b), 9 (c) and 9 (d) show the results for four sizes of the features as Fig. 8. The number of the projections for a subassembly is fixed to 72. The combination of ProjectionOntoLine and ComputeSubDistance1 has the highest retrieval precision in any size of the projection and any size of the feature for a subassembly. When the size of the feature is more than 16, the size of the feature and the size of the projection do not affect the retrieval precision significantly. It may be because they are enough for capturing the features of the prepared assemblies.

---

### Table 1: Input parameters for proposed algorithms

| (a) Number of Projections |
|--------------------------|
| number of projections     |
|                          |
| 8 | 18 | 32 | 50 | 72 |
| $|L| |Φ| |Λ| |
| 2 | 3 | 4 | 5 | 6 |
| 4 | 6 | 8 | 10 | 12 |
| L | 2 | 3 | 4 | 5 | 6 |

| (b) Size of Projection |
|------------------------|
| size of projection     |
|                        |
| X1 | X2 | Y1 | Y2 |
| 30 | 36 | 44 | 57 |
| 30 | 36 | 44 | 57 |
| 30 | 36 | 44 | 57 |
| 256 | 256 | 256 | 256 |
| 15 | 18 | 22 | 29 |

| (c) Size of Feature   |
|-----------------------|
| size of feature       |
|                       |
| N1 | N2 | N3 | N4 |
| 4  | 16 | 36 | 64 |
| 4  | 16 | 36 | 64 |
| 2  | 4  | 6  | 8  |
| 2  | 4  | 6  | 8  |
4.3 Processing Time to Compute Distance between Assemblies

We compare average processing time of the proposed methods, which is required for retrieving the database model which is the answer to a query model. Figures 10, 11 and 12 show the average processing time in varying the number of the projections for a subassembly, the number of the polygons constituting an assembly and the size of the projection for a subassembly, respectively. The number of the projections, the size of the projection and the size of the feature are fixed to 72, 100 and 64, respectively when they are not varied in the experiments. The input parameters to the proposed algorithms are as described in Table 1.

Figures 10 (a), 11 (a) and 12 (a) show the average time to compute the projections for all subassemblies constituting an assembly by ProjectionOntoLine and ProjectionOntoPlane. For a projection of a subassembly by ProjectionOntoLine, the cross-sectional areas of the subassembly with
planes are computed $|X^1|$ times. On the other hand, for a projection of an subassembly by ProjectionOntoPlane, the lengths of intersection of the subassembly with lines are computed $|X^1| \times |Y^2|$ times. Therefore ProjectionOntoLine is more efficient than ProjectionOntoPlane as their processing time shows.

Figures 10 (b), 11 (b) and 12 (b) show the average time to compute the distance between two assemblies from the outputs of ProjectionOntoLine and ProjectionOntoPlane. The results shows that ComputeSubDistance1 using the spherical harmonic transform is more efficient than ComputeSubDistance2 using the combinatorial method. As the number of the projections increases, ComputeSubDistance2 needs more processing time than ComputeSubDistance1 as shown in Fig. 10 (b). Since the inputs to ComputeSubDistance1 and ComputeSubDistance2 do not depend on the number of polygons constituting an assembly and the size of the projection for a subassembly, they have little effect on the processing time to compute the distance between two assemblies.

The whole processing time to retrieve the database model for a query model is shown in Fig. 10 (c), Fig. 11 (c) and Fig. 12 (c). It depends on the processing time of ProjectionOntoLine and ProjectionOntoPlane since the processing time of ComputeSubDistance1 and ComputeSubDistance2 is much shorter than their processing time. The results show that the methods using ProjectionOntoLine and ComputeSubDistance1, that is, using the 3D Radon transform and the spherical harmonic transform is more efficient than the other methods.

5. Conclusion

We propose the retrieval methods for assemblies, which distinguish not only shapes of 3D CAD assemblies but also layouts of their components. The proposed method contains the two kinds of projections for a subassembly and the two kinds of distances between subassemblies. The results of our experiments show that the method using the 3D Radon transform and the spherical harmonic transform has better retrieval precision and efficiency than the the targets for comparison. Improving the implementation of the proposed method is a future work.

Acknowledgments

This work is supported by JSPS KAKENHI Grant Number 15K12028 and 18K11317.
References

[1] B. Bustos, D.A. Keim, D. Saupe, T. Schreck, and D.V. Vranić, “Feature-based similarity search in 3D object databases,” ACM Comput. Surv., vol.37, no.4, pp.345–387, Dec. 2005.

[2] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani, “Three-dimensional shape searching: state-of-the-art review and future trends,” Comput. Aided. Design., vol.37, no.5, pp.509–530, 2005.

[3] J.W.H. Tangelder and R.C. Veltkamp, “A survey of content based 3D shape retrieval methods,” Multimed. Tools. Appl., vol.39, no.3, pp.441–471, 2008.

[4] B. Li, Y. Lu, C. Li, A. Godil, T. Schreck, M. Aono, M. Burtscher, Q. Chen, N.K. Chowdhury, B. Fang, H. Fu, T. Furuya, H. Li, J. Liu, H. Johan, R. Kosaka, H. Koyanagi, R. Ohbuchi, A. Tatsuma, Y. Wan, C. Zhang, and C. Zou, “A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries,” Comput. Vis. Image. Und., vol.131, pp.1–27, 2015.

[5] K. Lupinetti, J.-P. Pernot, M. Monti, and F. Gianninni, “Content-based cad assembly model retrieval: Survey and future challenges,” Comput. Aided. Design., vol.113, pp.62–81, 2019.

[6] X. Chen, S. Gao, S. Guo, and J. Bai, “A flexible assembly retrieval approach for model reuse,” Comput. Aided. Design., vol.44, no.6, pp.554–574, 2012.

[7] A.S. Deshmukh, A.G. Banerjee, S.K. Gupta, and R.D. Sriram, “Content-based assembly search: A step towards assembly reuse,” Comput. Aided. Design., vol.40, no.2, pp.244–261, 2008.

[8] T. Miura and S. Kanai, “3D Shape Retrieval considering Assembly Structure,” Asian Symposium for Precision Engineering and Nanotechnology, 2009.

[9] K.-M. Hu, B. Wang, J.-H. Yong, and J.-C. Paul, “Relaxed lightweight assembly retrieval using vector space model,” Comput. Aided. Design., vol.45, no.3, pp.739–750, 2013.

[10] D. Saupe and D.V. Vranić, “3d model retrieval with spherical harmonics and moments,” Pattern Recognition, Lecture Notes in Computer Science, vol.2191, pp.392–397, Springer Berlin Heidelberg, 2001.

[11] D.V. Vranić, “An improvement of rotation invariant 3d-shape based on functions on concentric spheres,” Proc. 2003 International Conference on Image Processing, pp.III–757-60, Sept. 2003.

[12] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, “Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors,” Proc. 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp.156–164, 2003.

[13] D. Zarpalas, P. Daras, A. Axenopoulos, D. Tzovaras, and M.G. Strintzis, “3D Model Search and Retrieval Using the Spherical Trace Transform,” EURASIP Journal on Advances in Signal Processing, vol.2007, no.1, p.023912, Oct. 2006.

[14] P. Papadakis, I. Pratikakis, T. Theoharis, G. Passalis, and S. Perantonis, “3D object retrieval using an efficient and compact hybrid shape descriptor,” Proc. 1st Eurographics Conference on 3D Object Retrieval, 3DOR ’08, pp.9–16, Eurographics Association, 2008.

[15] G.L. López, A.P.P. Negrón, A.D. Jiménez, J.R. Rodríguez, and R.I. Paredes, “Comparative analysis of shape descriptors for 3D objects,” Multimed. Tools. Appl., vol.76, no.5, pp.6993–7040, March 2017.

[16] M.A. Savelonas, I. Pratikakis, and K. Sifakis, “An overview of partial 3D object retrieval methodologies,” Multimed. Tools. Appl., vol.74, no.24, pp.11783–11808, Dec. 2015.

[17] K. Katayama and T. Sato, “A Matching Method for 3D CAD Models with Different Assembly Structures Using Projections of Weighted Components,” Journal of Information Processing, vol.25, pp.376–385, 2017.

[18] GrabCAD. http://grabcad.com/library.

Kaoru Katayama received Ph.D. degree in informatics from Kyoto University in 2000. He is currently an associate professor in the Department of Computer Science, Tokyo Metropolitan University. His research interests are in the areas of data engineering and data mining.

Takashi Hirashima is a student of Tokyo Metropolitan University. His research interests are in the area of data engineering.