Relaxed lightweight assembly retrieval using vector space model

Kai-Mo Hu, Bin Wang, Jun-Hai Yong, Jean-Claude Paul

Relaxed lightweight assembly retrieval using vector space model

Kai-Mo Hu [a,b,c,d,*], Bin Wang [a,c,d], Jun-Hai Yong [a,c,d], Jean-Claude Paul [a,e]

[a] School of Software, Tsinghua University, Beijing 100084, PR China
[b] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
[c] Key Laboratory for Information System Security, Ministry of Education, Beijing 100084, PR China
[d] Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, PR China
[e] INRIA, France

ARTICLE INFO

Article history:
Received 7 March 2012
Accepted 7 October 2012

Keywords:
Lightweight
Typical parts
Assembly retrieval
Vector space model
Relaxed retrieval

ABSTRACT

Assembly searching technologies are important for the improvement of design reusability. However, existing methods require that assemblies possess high-level information, and thus cannot be applied in lightweight assemblies. In this paper, we propose a novel relaxed lightweight assembly retrieval approach based on a vector space model (VSM). By decomposing the assemblies represented in a watertight polygon mesh into bags of parts, and considering the queries as a vague specification of a set of parts, the resilient ranking strategy in VSM is successfully applied in the assembly retrieval. Furthermore, we take the scale-sensitive similarities between parts into the evaluation of matching values, and extend the original VSM to a relaxed matching framework. This framework allows users to input any fuzzy queries, is capable of measuring the results quantitatively, and performs well in retrieving assemblies with specified characteristics. To accelerate the online matching procedure, a typical parts based matching process, as well as a greedy strategy based matching algorithm is presented and integrated in the framework, which makes our system achieve interactive performance. We demonstrate the efficiency and effectiveness of our approach through various experiments on the prototype system.

1. Introduction

Over the last twenty years, Computer Aided Design (CAD) systems have become popular in product development organizations, and this results in the creation of large databases of assemblies. For such organizations, the efficient management and effective reuse of these assemblies become a key factor to stay ahead in the completion.

Searching technologies of mechanical components are the key to getting the required assemblies. However, to the best of our knowledge, existing search methods mainly focus on part retrieval [1–5]. Though part retrieval methods can be applied to find such assemblies whose overall shapes are similar, they all fail to take into consideration relationships, compositions and structures that exist in the assemblies. Recently, some assembly retrieval methods have been proposed [6,7]. They assume that the assemblies possess full information, such as the topological structures, the orientation relationships and the joint information, and thus provide some ingenious mechanisms to retrieve assemblies with characterized topologies or features. However, if the product data in PDM or PLM systems is in lightweight file format (e.g. watertight polygon mesh), the existing methods cannot be applied.

To provide a practical way for the retrieval tasks of assemblies represented in lightweight file format, we propose a new framework, in which only the geometrical information of parts and composition information in assemblies is used. The VSM, which has been used successfully in the information retrieval (IR) field, is applied to get the partial matching and quantitative ranking of assemblies. To overcome the limitation of VSM that fuzzy queries are not supported, we extend the traditional VSM to a relaxed matching framework by integrating the similarities between parts into the evaluation of matching values, and thus provide more flexible means for query specifications and result rankings.

Since the efficiency is essential for online retrieval, we design a typical parts based matching process as well as a greedy strategy based matching algorithm, and apply them in our relaxed retrieval framework. Experiment results demonstrate the efficiency and effectiveness of our work in a variety of manners, and verify its utility in retrieving the lightweight assemblies with desired characteristics.

Additionally, we propose a normal compatibility based assembly decomposition algorithm and a scale-sensitive similarity based parts merging algorithm. Using these two algorithms, the assemblies represented in lightweight formats can be segmented and

* Corresponding author at: School of Software, Tsinghua University, Beijing 100084, PR China. Tel.: +86 10 62795459; fax: +86 10 62795460.
E-mail address: hukaimo02@gmail.com (K.-M. Hu).

merged as a set of representative single parts accurately, such that the extended VSM can be applied directly.

In summary, we make the following contributions:

- Propose a normal compatibility based assembly decomposition algorithm, as well as a scale-sensitive similarity based parts merging algorithm, so as to disassemble the assemblies into sets of representative parts.
- Apply the classical VSM into the assembly retrieval, and extend the traditional VSM to a relaxed matching framework to support fuzzy retrieval.
- Present a typical parts based matching process and a greedy strategy based matching algorithm for the system acceleration. By integrating them into the matching algorithm, our system could achieve interactive performance.

Compared to the state-of-the-art work, we provide a well defined mechanism for the retrieval of assemblies represented in polygon mesh format. By introducing and extending the classical VSM in IR into assembly retrieval, our proposed framework allows users to retrieve assemblies through inputting parts sets, and is able to measure the retrieval results quantitatively.

The remainder of this paper is organized as follows. Section 2 reviews the relevant research on assembly retrieval, followed by an overview of our framework in Section 3. In Sections 4 and 5, we describe the preprocessing algorithms and online retrieval algorithms respectively, and then the experiment results and some discussions are presented in Section 6. Finally, we conclude our work and give some future work in Section 7.

2. Pre knowledge and related work

2.1. Related concepts in information retrieval

Since we try to apply the VSM in the assembly retrieval task, some related concepts in IR are presented. Generally, the classical models consider that each document is described by a set of representative keywords called *index terms*. An index term is simply a word whose semantics helps in remembering the documents' main themes. Since the distinct index terms have varying relevance, this effect is captured through the assignment of numerical *weights* to each index term. Based on different promises, three classic models, namely the Boolean model [8], the probabilistic model [9] and the VSM [10], are regarded as the basic models in IR systems.

The hypothesis of VSM is that two documents are similar if they contain some of the same terms. By assigning non-binary weights to index terms in queries and in documents, the degree of similarity between each document is stored in the system and the user query can be computed to support partial matching. Compared with the Boolean model and the probabilistic model, the main benefits of the VSM are: (1) its term-weighting scheme improves retrieval performance; (2) its partial matching strategy allows retrieval of documents that approximate the query conditions; and (3) its cosine ranking formula sorts the documents according to their degrees of similarity to the query. Theoretically, the VSM has the disadvantage that index terms are assumed to be mutually independent, which is not true in practice. However, modeling the dependencies between index terms is a very difficult task. Since most of the term dependencies are local, and only appear in a few special cases, if we apply them in applications indiscriminately, the overall performance does not improve significantly [11].

In assembly retrieval, if we decompose the assemblies into single parts, and treat an assembly as a set of representative parts, then each assembly can be regarded as a document, and the parts in the assembly can be considered as index terms. By this transformation, the classic VSM can be applied to perform the retrieval task of assemblies. In addition, the geometrical information of single parts can be easily embedded into the matching evaluation. This framework is promising to provide an easy way to retrieve the required assemblies, and thus to access the associated data for the support of downstream manufacturing and service operations.

2.2. Retrieval methods for models and assemblies

Due to its wide applications in a variety of areas, 3D model retrieval has drawn much attention in recent years. Some early research was aimed at proposing powerful descriptors, such as Shape Distribution [12], Shape Histogram [13], Extended Gaussian Image [14], Spherical Harmonic Descriptor [15] and Light Field Descriptor (LFD) [16]. Shilane et al. [17] tested some of the existing descriptors on the Princeton Shape Benchmark, and found that LFD provides the best retrieval precision. With the development of mesh segmentation technologies, some part-in-whole shape retrieval algorithms have been proposed [18,19]. These methods first extract the descriptors of local regions, and then match them using some special techniques. Recently, some deformation-invariant shape descriptors have been introduced [20–22]. Since the pair-to-pair matching of local features is time consuming, the bag-of-words method proposed in IR has been employed in recent 3D model retrieval tasks. For instance, Bronstein et al. [23] used multi-scale diffusion heat kernels as "geometric words", and constructed shape descriptors by means of "bag of features". This method is efficient and can deal with the deformable objects.

In CAD systems, the retrieval tasks are performed to improve the reusability of existing designs, and the work in literature can be classified into two categories. The first category tries to discover and extract common design patterns from databases. For example, Ma et al. [24,25] developed some approaches to extract common local structures as design patterns from boundary representation (B-rep) models. They first translated the models into some graphs, and then extracted the frequent subgraphs as common design structures. The second category targets partial retrieval of CAD models, such that new parts do not have to be designed from scratch. Hong et al. [1] presented a two-step similarity comparison method for B-rep, the overall appearances are compared first, and then the detailed features are matched. To solve the partial matching problem of CAD models, Bespalov et al. [2] proposed a scale-space feature extraction technique based on the recursive decomposition of polyhedral surfaces into surface patches. Their idea of decomposition and retrieval is in some way similar to our framework. Another partial matching work was reported by Bai et al. [4]. They first defined the criteria for determining whether a subpart of CAD models is reusable, and then extracted and stored the reusable subparts in the library for later retrieval. El-Mehalawi and Miller [26,27] provided the representation scheme of CAD models, including representation, indexing, retrieval, matching and similarity assessment.

In recent years, with the wide usage of PDM, more and more assemblies have been accumulated. To reuse the design knowledge, some researchers propose to retrieve similar cases when designing new assemblies. For instance, Kim et al. [28] described the case-based reasoning (CBR) method. By searching through the case library, the previous designs whose identified features are similar to the current case can be found. However, the knowledge in the case library only contains product features and process features. Based on CBR, a recent work on producing interesting redesigns through general design knowledge was reported in [29]. Chao et al. [30] also proposed a case retrieval method, in which the welding and assembly processing information are employed. Provided with assembly drawings, Liu et al. [31,32] proposed to extract component parts, and then perform the assembly drawing

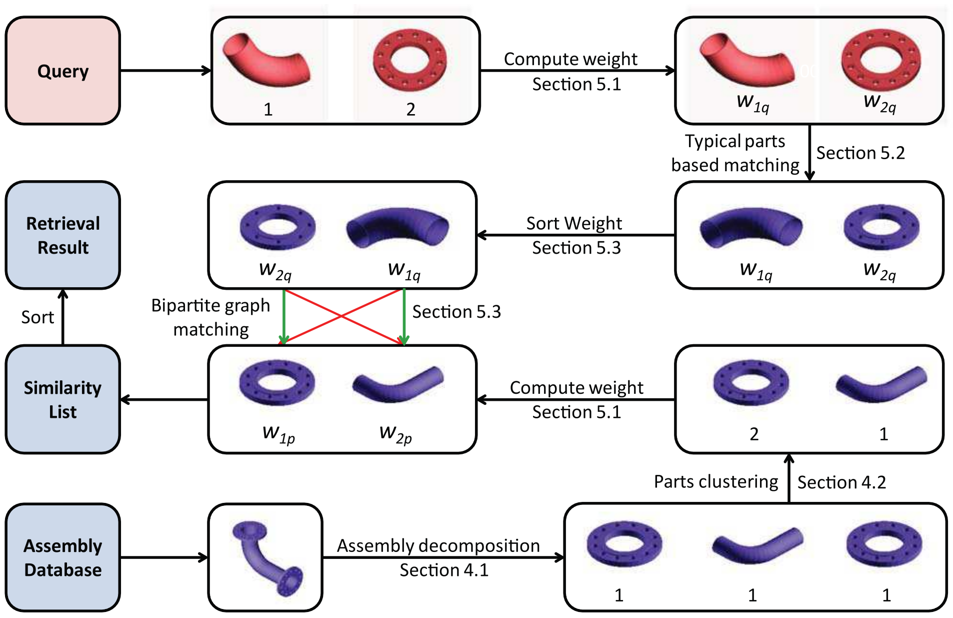


Fig. 1. The overall workflow and sub-procedures of our relaxed retrieval framework. The first row shows the preprocessing phase, while the 2–4 rows depict the online retrieval phase. For each sub-procedure, we present the action and its corresponding description section in our paper around the arrow.

retrieval by matching the extracted component parts. The support vector machine based relevance feedback was adopted to improve the performance. The idea of extracting component parts and performing partial retrieval resembles ours in some way, but they can only deal with 2D assembly drawings, and fuzzy retrieval is not supported.

The approaches that are directly related to assembly retrieval are introduced by Deshmukh et al. [6] and Chen et al. [7]. In [6], the search criteria, search algorithm and search strategies for content-based assembly search are stated in detail, and a simple system is developed to support content-based searches. Though the five assembly search examples have illustrated the promising power of assembly retrieval, their current implementation mainly focuses on the global information of the assemblies, such as the names of the constituent parts, the mating conditions between different parts, the material and mass properties of the parts and so on, and the quantitative rankings are not provided. To overcome the limitations of existing methods, Chen et al. [7] proposed a flexible and effective approach, in which the multilevel descriptor, the hierarchical assembly structure and the semantic assembly interface are presented to preserve the implicit high-level design semantics. Additionally, they provided an efficient indexing mechanism to accelerate the retrieval process.

While these two methods provide goodish solutions for assembly retrieval, they cannot be applied directly to the assemblies represented in lightweight file format, since in such format, only the low-level geometrical information is preserved, and the high-level feature information is lost. Unfortunately, to the best of our knowledge, most of the product data in PDM and PLM systems are stored in lightweight file format, so as to save storage space.

3. Overview

To solve the assembly retrieval problems existing in literature, we propose a relaxed assembly retrieval framework, targeting at solving the retrieval tasks of assemblies represented in a watertight polygon mesh. In this framework, we treat the assemblies as "bags of parts", and apply the well-designed ranking strategies of VSM in our retrieval algorithm. To support fuzzy queries, we take the similarities between parts into the evaluation of matching values, and thus extend the original VSM to a relaxed retrieval framework. A typical parts based matching process and a greedy strategy based matching algorithm are also integrated, so as to make our system achieve interactive performance. The overall workflow is illustrated in Fig. 1.

In the following sections, we first introduce the preprocessing algorithms of assemblies in Section 4, including the assembly decomposition algorithm and the parts merging algorithm, and then describe our relaxed matching framework in detail in Section 5, including the application of VSM, the typical parts based matching process and the greedy matching algorithm.

4. The preprocessing of assemblies

4.1. Normal compatibility based assembly decomposition

Suppose the parts are exported from CAD modeling systems in regular form, in which the dangling edges and dangling faces are not allowed, and one edge can only be adjacent to exactly two faces. Furthermore, we require that the mesh models must be watertight, and the normals of adjacent faces are compatible.

Based on these two constraints, we have designed an algorithm that can accurately decompose an assembly into a set of partial meshes, each of which corresponds to a single part. In this algorithm, a breadth-first search (BFS) strategy is applied to construct the decomposed mesh progressively. Generally, the faces in the assembly $\mathcal{A}$ can be regarded as the nodes in a graph, and the adjacent relationships between faces can be regarded as edges. In the decomposing process, we first construct the set of adjacent faces to u, and then for each adjacent face v, we check whether it is compatible to u. If so, it is placed into the mesh with u. After a complete traversal of BFS started from u, all the faces that belong to the same single part are marked as "visited". Hence, the newly constructed mesh M is added to the set $\mathcal{S}$, and we continue to search the "not visited" faces in $\mathcal{A}$ until all the faces have been marked as "visited". The implementation details are shown in Algorithm 1.

The most time consuming step in Algorithm 1 is the construction of adjacent relationships between faces in $\mathcal{A}$, which requires

Algorithm 1: Assembly decomposition algorithm

```
Input: $\mathcal{A}$: the assembly represented by mesh;
Output: $\mathcal{S}$: the set that stores the decomposed meshes.
1  foreach face $f$ in $\mathcal{A}$ do
2      Mark $f$ as "not visited";
3  end
4  while Exists a "not visited" face $f$ in $\mathcal{A}$ do
5      Initialize an new mesh $M$;
6      $Q \leftarrow \emptyset$;                  ▷ $Q$ is a queue.
7      ENQUEUE($Q, f$);          ▷ Place $f$ at the tail of $Q$.
8      while $Q \neq \emptyset$ do
9          $u \leftarrow$ DEQUEUE($Q$);  ▷ Extract the head of $Q$.
10         foreach $v$ that is adjacent to $u$ in $\mathcal{A}$ do
11             if $v$ is marked as "not visited" and is compatible with
               $u$ then
12                 ENQUEUE($Q$, $v$);
13             end
14         end
15         Add $u$ to $M$;
16         Mark $u$ as "visited";
17     end
18     Add $M$ to $\mathcal{S}$;             ▷ A mesh has been extracted.
19 end
```

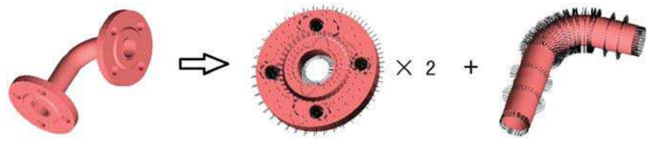


Fig. 2. The illustration of the decomposed parts from an assembly. The pipe fitting is composed of 2 flanges and 1 adapter. The black line segments indicate the normals of their corresponding faces. Note that the normals in each watertight mesh are compatible.

$O(V^2)$ time complexity, and V is the number of faces in $\mathcal{A}$. However, this decomposition algorithm can be performed offline, so it does not hurt the online retrieval performance at all.

4.2. *Scale-sensitive similarity based parts merging*

Fig. 2 gives an example showing how the assemblies are decomposed into single parts using Algorithm 1. However, a single part may appear several times in the same assembly, such as the flanges appearing in Fig. 2. To recognize and merge the identical parts that have been separately decomposed by Algorithm 1, we propose to use robust 3D shape descriptors.

Since LFD [16] performs best according to Shilane's test [17], we employ their program to characterize the parts' global geometries. However, LFD returns small values if two models are similar. To make it fit for our algorithm, we normalize the similarity s_{LFD} returned by Chen's program as follows:

$$s_{\text{geometry}}(u, v) = \begin{cases} 1 - \dfrac{s_{LFD}(u, v) - s_{\min}}{s_{\max} - s_{\min}} & \text{if } s_{\max} > s_{\min} \\ s_{LFD} & \text{if } s_{\max} = s_{\min} \end{cases} \quad (1)$$

where $s_{\min}$ and $s_{\max}$ are the minimum and maximum values returned from all the models by LFD.

Using s_{geometry} alone is not enough, since the relative scales between single parts are also important characteristics in assemblies. However, LFD is a scale-invariant descriptor. To solve this problem, we introduce another invariable s_{scale} to measure the scale differences between two models:

$$s_{\text{scale}}(u, v) = \begin{cases} 1 - \dfrac{|r_u - r_v|}{r_{\max} - r_{\min}} & \text{if } r_{\max} > r_{\min} \\ 1 & \text{if } r_{\max} = r_{\min} \end{cases} \quad (2)$$

where r_u and r_v are the radii of models u and v, while $r_{\min}$ and $r_{\max}$ are the minimum and maximum values of radii in the part database, respectively.

Using the definitions of Eqs. (1) and (2), we define the scale-sensitive similarity measurement as follows:

$$s(u, v) = s_{\text{geometry}}(u, v) * s_{\text{scale}}(u, v), \quad (3)$$

which characterizes both the geometry and scale properties.

Based on this similarity measurement, a parts merging algorithm is designed in Algorithm 2. In this algorithm, we first initialize each part as an individual set, and then traverse every pair (u, v) in the set of decomposed parts. If the similarity between u and v measured by Eq. (3) is larger than a threshold, they are regarded as identical, and the two sets are merged in the case that they are previously in different sets. In the LFD program, the geometry(u, v) always returns the same value if two parts are the same. In our implementation, we consider two parts have the same scale if the difference of their scale(u, v) is less than 1%. Hence, the threshold ϵ was set to 0.99 in our experiment. The merging experiments also verified that this configuration could get good results.

Algorithm 2: Parts merging algorithm

```
Input: $\epsilon$: threshold that identifies the sameness of parts;
       $\mathcal{S}$: the set that stores the decomposed parts.
Output: $\mathcal{C}$: the set that stores the merged parts.
1  foreach part $u \in \mathcal{S}$ do
2      $\mathcal{C} \leftarrow$ MAKE-SET($u$);
3  end
4  for $i \leftarrow 1$ to size($\mathcal{S}$) $- 1$ do
5      $u \leftarrow$ the $i$th part in $\mathcal{S}$;
6      for $j \leftarrow i + 1$ to size($\mathcal{S}$) do
7          $v \leftarrow$ the $j$th part in $\mathcal{S}$;
8          Calculate $s(u, v)$ using Eq. (3);
9          if $s(u,v) \geq \epsilon$ then
10             if FIND-SET($u$) $\neq$ FIND-SET($v$) then
11                 UNION($u$, $v$);     ▷ Union the two sets.
12             end
13         end
14     end
15 end
```

The time complexity of Algorithm 2 is analyzed as follows. Suppose the element number in $\mathcal{S}$ is V, the double loops in lines 4–15 take $O(V^2)$ times. In our prototype system, the FIND-SET and UNION operations have been implemented on the disjoint-set forest, which take $O(1)$ and $O(m\alpha(n))$ respectively. Here, $\alpha(n)$ is a very slowly growing function, and can be regarded as $\alpha(n) \leq 4$ in any conceivable applications [33]. So the upper bound of running time is $O(V^3)$.

Since this algorithm can be executed offline, the high time complexity does not affect the online retrieval performance. In industrial practice, we first run Algorithm 2 online, and then store the merged results for later utilization. When a new part is added, we do not have to run Algorithm 2 again, but only need to scan in the merged parts. If there exists a part whose similarity with the new part is larger than the threshold ϵ, we merge the new part with that part; otherwise, the new part is added into the merged parts set directly.

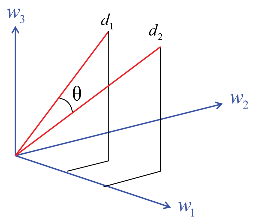


Fig. 3. Two assemblies that are represented in 3-dimensional part vector space (w_1, w_2, w_3). The similarity is measured by the angle between their vectors in the same part vector space.

5. Relaxed assembly retrieval framework

By considering assemblies as"bags of parts", and regarding the query parts as a special type of assembly, the classic IR model, such as VSM can be applied directly to the assembly retrieval task. In addition, the geometrical information of single parts can be easily embedded into this retrieval framework. In this section, we will show how to apply the VSM in assembly retrieval, and how the application can be extended to a relaxed matching framework.

5.1. Vector space model in assembly retrieval

Similar to the document retrieval, we treat the assemblies as a collection $\mathcal{C}$ of parts and regard the user query as a vague specification of a set A of parts. In this scenario, the assembly retrieval problem can be reduced to the problem of determining which assemblies are in the set A and which ones are not. This can be partially viewed as a clustering problem, and two main issues have to be resolved. First, we need to determine what are the features that better describe the assemblies in the set A. Second, we need to determine what are the features that better distinguish the objects in the set A from the remaining parts in the collection $\mathcal{C}$. The first set of features provides for quantification of *intra-cluster* similarity, while the second set of features provides for quantification of *inter-cluster* dissimilarity. We try to balance these two effects.

Basically, we have the hypothesis that two assemblies are similar if they contain some of the same parts. The possible measures of similarity that we might take into consideration are: (a) the number of parts in the assemblies; (b) the number of parts in common; (c) whether the parts are common or unusual and (d) how many times each part appears. To encode this idea, we transform the assemblies into an n-dimensional space, where n is the number of different parts used to index the set of assemblies. An assembly j is represented by a vector $d_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$, and its magnitude in dimension i is defined as w_{ij}, where

$$\begin{cases} w_{ij} > 0, & \text{if part } i \text{ occurs in assembly } j; \\ w_{ij} = 0, & \text{otherwise.} \end{cases}$$

Here, w_{ij} denotes the weight of part i in assembly j.

By transforming the assemblies into part vector space, the similarity between two assemblies is defined as a function of the angle between their vectors in the part vector space. We take two assemblies represented in 3-dimensional part vector space as an example, as shown in Fig. 3. Generally, the correlation between vectors can be quantified by the cosine of the angle between these two vectors. That is,

$$\mathrm{sim}(\mathbf{x}_1, \mathbf{x}_2) = \cos(\theta) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{|\mathbf{x}_1|\,|\mathbf{x}_2|} \tag{4}$$

where $\mathbf{x}_1 \cdot \mathbf{x}_2$ is the inner product of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$.

Using Eq. (4), the similarity between two assemblies can be calculated easily. However, different assignments of part weights may result in different rankings. We will show how to achieve the best performance by balancing the effects of *intra-cluster* similarity and *inter-cluster* dissimilarity.

The *intra-clustering* similarity is quantified by measuring the raw frequency of a term k_i inside an assembly d_j, since a part that appears many times within an assembly is likely to be more important than a part that appears only once. Suppose the part i appears f_{ij} times in assembly j, a simple method is to use f_{ij} as the weight of part i. However, parts are likely to appear more often in complex assemblies. Therefore, f_{ij} should be formalized by some variable related to the complexity of the assembly. A standard method in IR is to formalize f_{ij} so that it is relative to the frequencies of other parts in the assembly. Let $m_j = \max(f_{ij})$ in assembly j, then the *Term Frequency (tf)* is formalized as

$$tf_{ij} = f_{ij}/m_j, \quad \text{when } f_{ij} > 0. \tag{5}$$

Since m_j presents a simple way to reflect the complexity of the assembly, and thus normalizes the tf_{ij} approximately, Eq. (5) works well in our experiments.

The *inter-cluster* dissimilarity is quantified by measuring the inverse of the frequency of a part i among all the assemblies in the collection. This factor is usually referred to as the *inverse document frequency (idf)* in IR. In our intuition, a part that occurs in a few assemblies is likely to be a better discriminator than a part that appears in most or all of the assemblies. To model this concept, we suppose there are n assemblies in database and the number of assemblies in which part i occurs is n_i, then a possible method is to use n/n_i as idf. However, this over-emphasizes small differences. Therefore, the logarithm is applied, and the idf can be revised as

$$idf_i = \log\left(\frac{n}{n_i}\right) + 1, \quad \text{when } n_i > 0. \tag{6}$$

We add 1 to the logarithm such that the idf_i is always positive.

Generally, the *intra-clustering* similarity and the *inter-cluster* dissimilarity depict the importance of a specified part from two different aspects. For an assembly, the higher value a part's tf and idf values are, the more important the part is. Thus, we can model the weighting scheme of a part by the product of tf and idf. Practical experience in IR has also demonstrated that this weighting scheme performs well in a wide variety of circumstances [11]. By applying this observation, and using Eqs. (5) and (6), the weight of the i-th part in the j-th assembly can be formulized as

$$w_{ij} = tf_{ij} * idf_i = \frac{f_{ij}}{m_j} * \left(\log\left(\frac{n}{n_i}\right) + 1\right), \quad \text{when } n_i > 0. \tag{7}$$

In our VSM framework, a query is treated as a special type of assembly. By substituting Eq. (7) to Eq. (4), the similarity between query q and assembly d_j is given by

$$\mathrm{sim}(q, d_j) = \cos(\mathbf{d}_q, \mathbf{d}_j) = \frac{\sum_{k=1}^{n} w_{kq} w_{kj}}{|\mathbf{d}_q|\,|\mathbf{d}_j|}, \tag{8}$$

where $\mathbf{d}_q$ and $\mathbf{d}_j$ are the corresponding weighted part vectors. For the query part weight w_{kq}, we use the suggestion of Salton and Buckley [34]:

$$w_{kq} = \left(0.5 + 0.5 * \frac{f_{kq}}{m_q}\right) * \left(\log\left(\frac{n}{n_k}\right) + 1\right), \quad \text{when } f_{kq} > 0, \tag{9}$$

and for the assembly part, the weight w_{kj} is given by Eq. (7).

5.2. Typical parts based matching process

Suppose there are m parts in the query and n parts in the assembly to be matched. For each online matching process, we need to extract the descriptors for all the m query parts, and then get their pair-to-pair similarities to the n assembly parts, which requires at least $m * n$ similarity calculations. This is insupportable in practice, especially when the scale of the assembly database becomes large. To accelerate the online matching process, we propose a typical parts based matching process, as shown in the following two steps.

In the offline step, we select a number of typical parts, all of which are representative in CAD model databases, both in geometries and scales. Then, all the unique parts that participate in the assemblies are decomposed, and their pair-to-pair scale-sensitive similarities to the typical parts are pre-computed and stored in the similarity table.

In the online step, given a part q provided by the user, we first search the most similar part t from the list of typical parts, and then get the similarity between t and the assembly part p by simply looking up in the similarity table. The final similarity between q and p can be approximately given by:

$$s(q, p) = s(q, t) * s(t, p), \tag{10}$$

where $s(q, t)$ is the scale-sensitive similarity between q and t, and $s(t, p)$ is the scale-sensitive similarity between t and p, which has been calculated in the offline step. Since the count of query parts and typical parts are limited, $s(q, t)$ can be calculated fast in the online step.

5.3. The relaxed matching algorithm

In most cases, we do not know what exact parts the assemblies contain, and still would like to find the assemblies with the desired characteristics by providing some fuzzy query parts. In this situation, simply applying the VSM into the assembly retrieval may result in poor performance, since for most of the fuzzy query parts, we may fail to find their corresponding parts in the assembly database, such that the participation relationships between the query parts and the assemblies can be constructed.

To provide a more universal framework for assembly retrieval, we propose to use the relaxed retrieval mode, which was introduced in [4]. In this mode, the query parts do not necessarily match the assembly parts exactly. Instead of using the Boolean values to indicate whether the query parts match the assembly parts exactly, we take the similarity measurement between query parts and assembly parts into consideration, and present a bipartite graph matching based framework to solve the relaxed retrieval problem.

We consider the m query parts and n assembly parts as the nodes in graph G, and the pair-to-pair similarities between query parts and assembly parts as edges. By constructing such a graph, the similarity measurement between the query and the assembly can be rendered as a bipartite graph matching problem. In this bipartite graph G, the weights of parts are assigned as the associated values of the nodes, and the scale-sensitive similarities between query parts and assembly parts are assigned as the weights of edges, as illustrated in Fig. 4. Suppose that the i-th part in the query is matched to the j-th part in the assembly, then the *matching value* is defined as:

$$S_{ij} = w_{iq} * s(i, j) * w_{jp}, \tag{11}$$

where $s(i, j)$ is the approximate scale-sensitive similarity defined in Eq. (10), and w_{iq} and w_{jp} are the weights of the query part and assembly part, respectively.

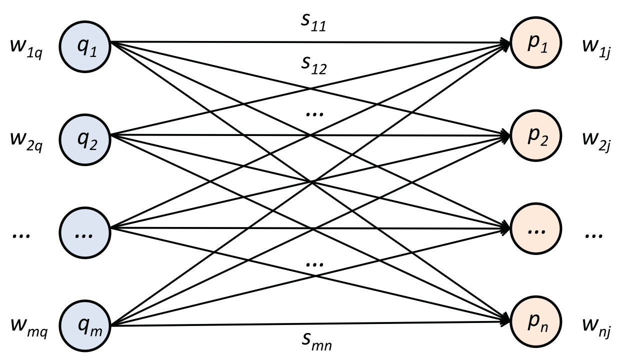


Fig. 4. The illustration of the relaxed matching framework. $w_{1q}, w_{2q}, \ldots, w_{mq}$ represent the weights of the m query parts, while $w_{1p}, w_{2p}, \ldots, w_{np}$ represent the weights of the n assembly parts. The values $s_{11}, s_{12}, \ldots, s_{mn}$ represent the scale-sensitive similarities between query parts and assembly parts.

The final similarity can be computed according to a match M, in which the sum of *matching values* is maximized:

$$M = \arg\max_{(i,j)} \Sigma S_{ij}, \quad 1 \le i \le m, 1 \le j \le n. \tag{12}$$

Given M, the similarity between query q and assembly d_j can be defined similar to Eq. (8):

$$\mathrm{sim}(q, d_j) = \cos(\mathbf{d}_q, \mathbf{d}_j) = \frac{S}{|\mathbf{d}_q|\,|\mathbf{d}_j|} = \frac{\Sigma S_{ij}}{|\mathbf{d}_q|\,|\mathbf{d}_j|}, \tag{13}$$

where S is the summation of *matching values* according to M.

The bipartite graph matching problem can be solved using the Kuhn–Munkras (KM) algorithm. However, the time complexity of its naive implementation is $O(n^4)$, and even the improved implementation still requires $O(n^3)$. This is impractical for large assembly databases. To make the application achieve interactive performance for users, we propose and implement an approximate matching algorithm using greedy strategy. Suppose the number of query parts is smaller than that of the assembly parts, then the matching procedure can be designed as Algorithm 3.

Algorithm 3: Relaxed matching algorithm

```
Input: $Q$: the list of query parts;
       $P$: the list of assembly parts in assembly $\mathcal{A}$.
Output: $S$: the similarity between $Q$ and $\mathcal{A}$.
Calculate the weights of parts in $Q$ using Eq. (9);
Sort the parts in $Q$ by weights in descending order;
Mark all the parts in $P$ as "not matched";
$S \leftarrow 0$;
for $i \leftarrow 1$ to $size(Q)$ do
    $maxMatchValue \leftarrow 0$, $maxMatchPart \leftarrow 0$;
    for $j \leftarrow 1$ to $size(P)$ do
        if $P[j]$ is marked as "not matched" then
            if $S_{ij} > maxMatchValue$ then
                $maxMatchValue \leftarrow S_{ij}$;
                $maxMatchPart \leftarrow j$;
            end
        end
    end
    $S \leftarrow S + maxMatchValue$;
    Mark the $maxMatchPart$-th part as matched;
end
```

In this algorithm, we first sort the parts in Q by their weights in descending order, so as to ensure the parts with larger weights are matched first. Then for each part in the sorted Q, we find such a part in the assembly that its *matching value* is maximized, as

Table 1
The snapshot of our benchmark. The "Total parts in assemblies" are all the parts that participate in the assemblies (with duplication), and the "Shared parts" are such parts that occur in at least two different assemblies.

Name	Count
Total assemblies	614
Total parts in assemblies	5100
Unique parts in assemblies	2814
Shared parts	1164
Typical parts	1755
Parts' average participate count	1.812
Assemblies' average parts count	8.306

shown in lines 5–17. In our implementation, the *matching value* S_{ij} is evaluated by Eq. (11).

If the number of query parts is larger than that of the assembly parts, we first sort the parts in P, and then seek the best match for each part in the sorted P, which is similar to the procedure presented in Algorithm 3. Suppose the number of parts in Q and P are m and n, respectively, then the time complexity of Algorithm 3 is $O(mn)$. In practice, the number of query parts is usually small and can be regarded as a constant in most cases, so the matching process can be done in approximate $O(n)$ time complexity, which is linear to the complexity of the assembly to be matched.

6. Experiment results and discussions

We implemented a prototype system, and the snapshot of the user interface is shown in Fig. 5. To make users easy to refine the fuzzy queries, the prototype system provides a function of retrieving similar parts. By providing an initial query part, users can get its scale-sensitive similar parts from a typical parts list by clicking the button "Query Similar Parts". If the required part is returned, users can add it to the query list, modify its frequency, and perform fuzzy retrieval by clicking "Query Fuzzy Assemblies". In the following subsections, we will first introduce the benchmark constructed in our experiment, and then present and discuss the relaxed retrieval results in several manners.

6.1. Experiment benchmark

There is no publicly available assembly database currently in the literature. To test the effectiveness of our approach, we gathered a large number of assemblies that had been modeled in industry, including household appliances, transportation, buildings, mechanical products and so on. 614 assemblies containing 5100 parts were finally selected as the test benchmark. By removing the duplications from the 5100 parts, we obtained 2814 unique parts for descriptor extraction. Then, the typical parts were selected from those 2814 unique parts in the following manner:

Initially, the scale-sensitive similarity defined in Section 3 was employed as similarity measurement, so that the pair-to-pair similarities among those unique parts can be computed. Then we applied the classical k-means clustering algorithm, and selected the mean of points in each cluster as typical parts. To get the parameter k and the initial partition before executing the k-means algorithm, we utilized Algorithm 2 to cluster similar parts, in which the threshold ϵ was set to 0.6. Table 1 gives the summary of our final benchmark after preprocessing.

6.2. Relaxed retrieval results

6.2.1. Comparison with the original VSM

To illustrate the advantages of our relaxed retrieval framework, a comparison with the performance of the original VSM in assembly retrieval is provided, and we call it exact retrieval for convenience. In this example, we take a flange that participates in seven assemblies in our benchmark as input, and then perform the exact retrieval and relaxed retrieval respectively. In the exact retrieval, the system only returns seven assemblies, in which the provided flange participates exactly, as shown in Fig. 6(a). On the contrary, the relaxed retrieval algorithm returns more relevant assemblies when provided with the same query, and Fig. 6(b) gives the corresponding retrieval result. We see that the top assemblies in Fig. 6(b) contain more flanges than that of Fig. 6(a), since the high frequencies of the flanges make their weights high, and some assemblies containing similar flanges in high frequencies are also ranked in the front.

Some other comparisons have been conducted on our prototype system as well. Compared to the exact retrieval, our relaxed retrieval framework allows users to input any fuzzy queries, takes both the part similarities and assembly compositions into the evaluations of matching values, and thus provides more powerful means for assembly retrieval.

6.2.2. Retrieval performance on query variations

Similar to that of document retrieval, the variations of queries in the relaxed retrieval also lead to the changes of retrieval results. For simplicity, we again take the flange in Fig. 6 as an example, and furthermore, a pipe fitting is added into the query list, as shown in the left side of Fig. 7. Different from the result shown in Fig. 6(b), in which the assemblies are ranked by the number of similar flanges, most of the assemblies that contain two flanges and one pipe fitting are ranked in the front in Fig. 7.

We have tested a number of queries by modifying the number and kind of input parts. The conclusion is, the more kinds of parts we provide, the larger possibility that the required assembly is found, since more kinds of parts tend to characterize more features in the required assembly. Moreover, the relative frequencies between query parts also influence the retrieval results. The higher frequency a query part is specified, the more assemblies containing it are ranked in the front.

6.2.3. Ranking characteristics on different assembly levels

Due to the complexity of assemblies, designers often tend to produce simple sub-assemblies first, and then assemble these sub-productions into a more complicated one. All the relationships between assemblies, sub-assemblies and single parts form a tree-like model, as shown in Fig. 9. In this situation, the parts participate in the assemblies on different assembly levels. Using our relaxed retrieval framework, the sub-assemblies on different assembly levels can be retrieved and sorted well.

We take the three sub-assemblies in a sedan as an example, as shown in Fig. 9. In this assembly tree, the tyre assembly is composed of parts a and b in Fig. 8; the wheel assembly contains parts c to g in Fig. 8, and one type assembly. As depicted in the top of Fig. 9, the front axle assembly constitutes two wheel assemblies and some other subsidiary parts.

By taking different parts in Fig. 8 as input, the similarities calculated by our algorithm vary accordingly, as listed in Table 2. In the first step, we only input model a, and the tyre, the wheel and the front axle assemblies are returned in descending similarities. Though all three assemblies contain a, the tyre contains the least parts, so it matches the query best. In the second step, we add b as input. Since the tyre is composed of a and b exactly, it matches the queries perfectly, and thus has similarity 1.00. The similarities of wheel and front axle are also higher than that in the first retrieval, since both of them contain a and b. In the third step, we take a, b and c as input. For the reason that only the wheel and the front axle contain c, their similarities rise, and the similarity of the tyre decreases. In the fourth step, the models a to d are employed in the query. Different from the ranking in the first three rounds, the

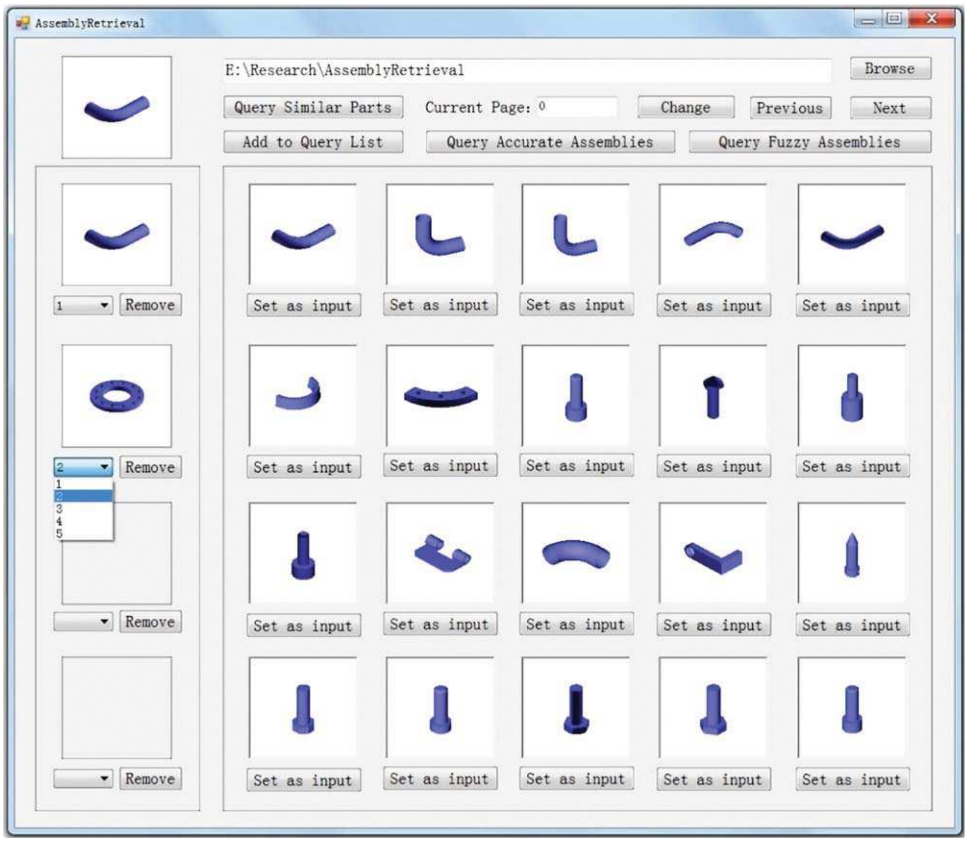
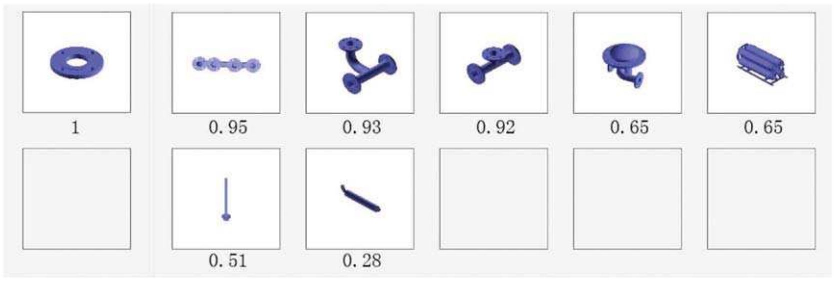
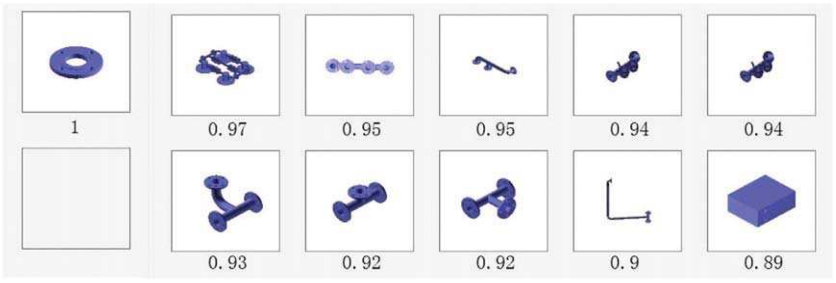


Fig. 5. The user interface of our prototype system. The left side shows the query list, and the right side shows the similar parts of the specified part or the returned assemblies. In this snapshot, the first part is a pipe fitting, and the second one is a flange. The right side shows the list of parts that are similar to the pipe fitting.



(a) The retrieval result using exact retrieval strategy.



(b) The retrieval result using relaxed retrieval strategy.

Fig. 6. The comparison between exact retrieval and relaxed retrieval by providing the same flange. The left side shows the query parts, and the integers below are their frequencies; the retrieved results are depicted in the right side, with similarities listed below the corresponding pictures.

wheel and the front axle are ranked top, because they match the queries better than that of the tyre. The difference between Step five and Step four is that we replace d with g in the query list. This modification does not affect the matchings of the wheel and the front axle, while making the matching of the tyre worse, so its similarity decreases significantly.

Furthermore, we have tested our method on silencer, mounting bracket and other assemblies, all of which are composed of sub-

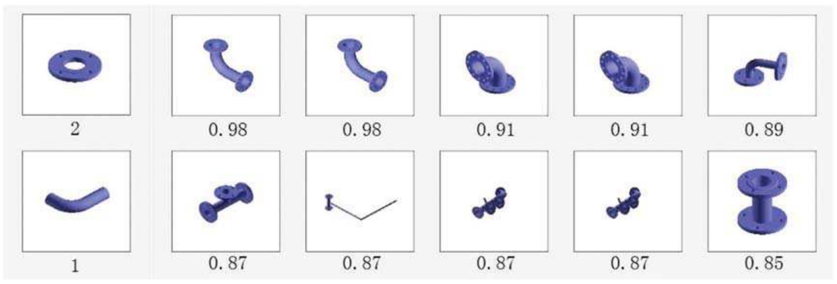


Fig. 7. The relaxed retrieval result of 2 flanges and 1 pipe fitting. The left side is the two inputs, while the right side is the top 10 returned assemblies.

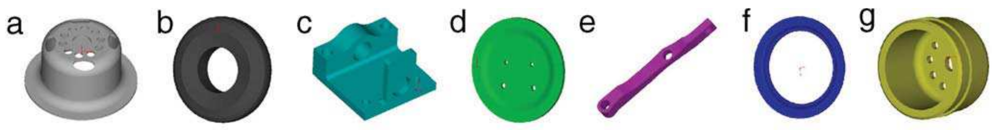


Fig. 8. The parts that participate in the front axle assembly and its sub-assemblies. We denote them in different colors. The assemblies that these parts participate in are shown in Fig. 9.

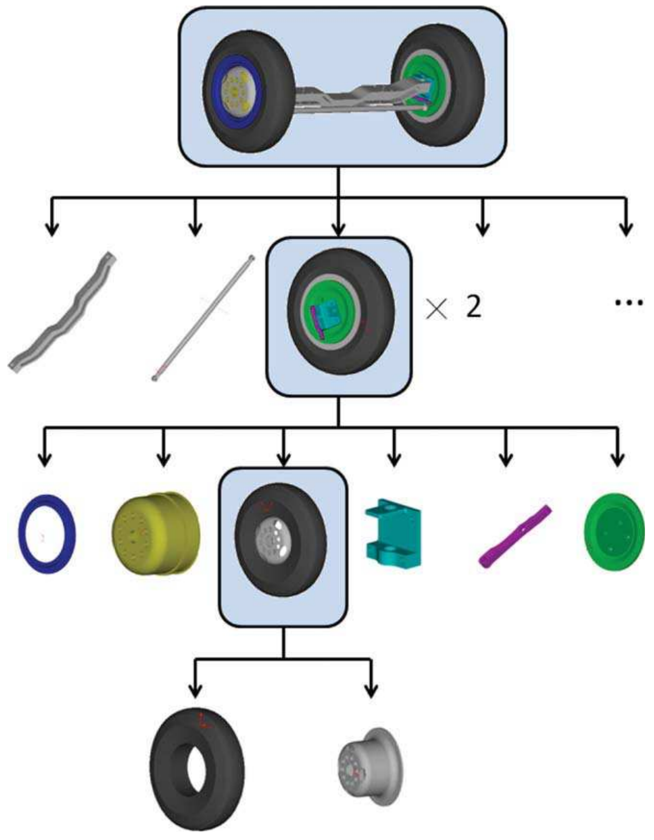


Fig. 9. The three-level assembly tree that is composed of the parts shown in Fig. 8. In this assembly tree, the assemblies from top to bottom are: (a) the front axle assembly, (b) the wheel assembly and (c) the tyre assembly, respectively. All the assemblies and sub-assemblies are in blue boxes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

assemblies and form the multi-level assembly trees. By adjusting the queries, the similarities and rankings of the sub-assemblies change accordingly, and the variation tendencies are similar to Table 2.

Sub-assemblies are higher granularity reused units compared to single parts. Using our relaxed retrieval method, the higher reusable sub-assemblies can be easily retrieved by providing only a few of their participating parts. Our method provides an easy way

Table 2
The rankings of the three assemblies by specifying different combinations of parts in Fig. 8. We use indices a to g to represent the queries. The three associated assemblies are listed in columns 2–4. In each row, the assembly names are followed by their corresponding similarity measurements defined in Eq. (9).

Queries	First	Second	Third
a	Tyre/0.71	Wheel/0.36	Front axle/0.35
a, b	Tyre/1.00	Wheel/0.51	Front axle/0.49
a, b, c	Tyre/0.80	Wheel/0.64	Front axle/0.61
a, b, c, d	Wheel/0.75	Front axle/0.72	Tyre/0.69
a, c, d, g	Wheel/0.75	Front axle/0.72	Tyre/0.34

to organize and reuse those assemblies in PDM databases. This is of great help for assembly analysis and modifications.

6.2.4. *Influence of fuzzy queries on retrieval results*

To illustrate the variations of retrieval results caused by fuzzy queries intuitively, we take a tyre as an experimental case, as shown in Fig. 10. The three most similar parts measured by Eq. (3) are taken as queries in sequence, and their top 5 assemblies ranked by our relaxed retrieval algorithm are presented in rows 2–4, respectively. The simple result indicates that, with a limited modification of similar queries, the retrieval results ranked on the top do not change distinctly.

A statistical experiment was designed, targeting at testing the influence of fuzzy queries. For each of the 1755 typical parts, we queried the top k assemblies, and set them as benchmarks. Then we took the top m similar parts as inputs in sequence, and queried their top k assemblies, respectively. Finally, the assemblies returned by the m similar parts were compared with the benchmark, and the percentages of coincident assemblies were calculated. Fig. 11 illustrates the average percentages of coincident assemblies with the variations of k and m.

We have the conclusion that by setting the most similar parts as input, more than 95% of the assemblies coincide with the benchmark. With the increase of m, the percentages of coincidence assemblies decrease accordingly. However, even if the fifth similar part is set as the query, there are still more than 85% coincident assemblies in the top 20 assemblies. Another observation is that the percentages of coincident assemblies increase quickly when k is smaller than 5, while they grow slowly when k is larger than 6. This indicates that we should pay more attention to all the top 5 assemblies, rather than being confined to the first one when seeking our required results.

Since our relaxed retrieval strategy is not sensitive to similar parts, we can retrieve fuzzy assemblies by inputting fuzzy queries.

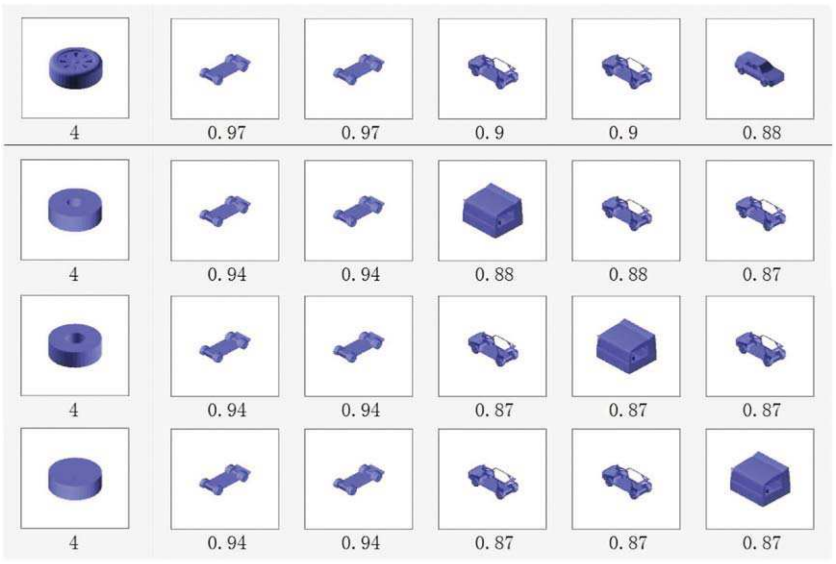


Fig. 10. The test of the relaxed retrieval with similar parts. On the left side, the first row is the original tyre, and 2–4 are its top 3 similar parts. The top 5 assemblies returned by our relaxed retrieval algorithm are presented on the right side of their corresponding rows.

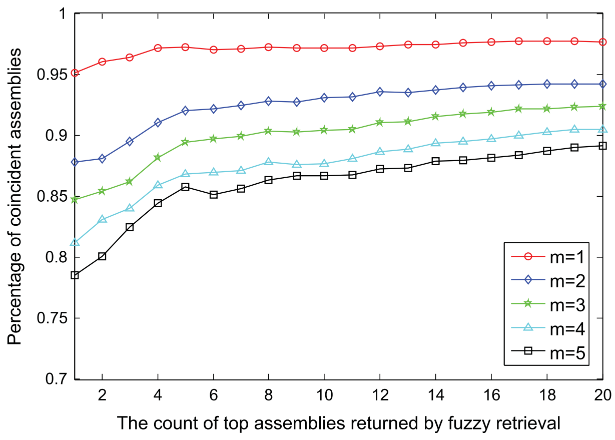


Fig. 11. The percentage of coincident assemblies, according to the similar input variations. The set of assemblies queried by the original part is set as the benchmark. *m* in the legend stands for the rank of the according input measured by the original part.

This is useful if users do not know the exact parts that an assembly contains, or want to retrieve fuzzy assemblies for conceptual design. For instance, if we input a set of parts that participate in the automobiles, the automobiles whose parts are in small variations can also be retrieved.

6.2.5. *Performance statistics of the relaxed retrieval*

Another statistical experiment was designed to test the effectiveness of our relaxed retrieval algorithm. For each assembly, we selected q parts randomly, and then performed the relaxed retrieval by setting them as input. The percentages that the original assemblies were ranked in the top k were evaluated. In our experiment, k varied from 1 to 20, and q varied from 1 to 10. To reduce the errors introduced by random selection, the experiment was performed 10 times, and the average percentages were recorded. In Fig. 12(a), the percentages are presented when q is set to 1, 3, 5, 7 and 9.

The results show that, with the increase of q, the performance gets better and better, since the more parts we provide in the query, the better the characteristics of the required assemblies can be carried. When 9 (which is close to the average count of parts in an assembly, according to Table 1) randomly selected query parts are provided, about 74% of the required assemblies can be found in the top 20 retrieval results. The statistical results also show that it is not wise to input only one part as query, since the percentage is only about 15% by statistics.

To illustrate the power of important parts in assemblies, another contrast experiment was conducted, and the result is depicted in Fig. 12(b). The only difference is that we selected the top q parts with highest weights in the assembly, rather than using the random selection tactic. The results show that the performance improves about 20% on average compared to the randomly selected queries. This demonstrates that high weight parts play more important roles in distinguishing their associated assemblies from others; and in addition, it verifies that the standard $tf * idf$ weighting scheme characterizes the importance of parts well in assembly retrieval.

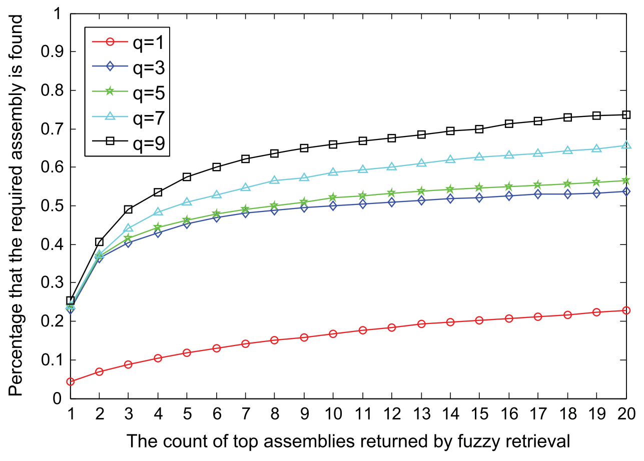
7. Conclusion and future work

In this paper, we have presented a novel framework for lightweight assembly retrieval. The most significant new idea of our approach is that the VSM has been employed in assembly retrieval successfully, and the original VSM in assembly retrieval has been extended into a relaxed matching framework. Since the VSM provides a graceful manner for partial matching, and the relaxed retrieval framework takes both the part similarities and assembly compositions into consideration, the system works pretty well in assembly retrieval, which is demonstrated by a mass of experiments.

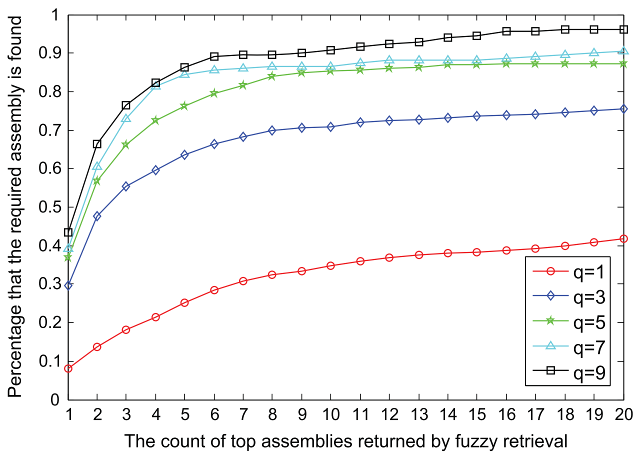
To make the assembly retrieval problem suitable for the application of classical IR models, the face normal compatibility based assembly decomposition algorithm and the scale-sensitive similarity measurement based parts merging algorithm have been presented for the preprocessing of assemblies.

Furthermore, the typical parts based matching process, as well as the greedy strategy based matching algorithm for the solution of bipartite graph matching problems is proposed and integrated in the framework. Though both of them are approximate approaches, they preserve the retrieval performance well, while make our system achieve interactive performance.

Compared to Deshmukh's [6] and Chen's [7] methods, our algorithm only utilizes the geometrical information of parts and the composition information of assemblies, and thus can be applied



(a) Retrieval results with randomly selected queries.



(b) Retrieval results with high weight queries.

Fig. 12. Experiment results showing the effectiveness of the relaxed retrieval algorithm. For each assembly, the relaxed retrieval task is performed by setting its selected *q* parts as inputs. The percentage that the original assembly is ranked in the top *k* results is calculated, where *k* varies from 1 to 20. In the randomly selecting group, the experiment was performed for 10 times, and the results were averaged to reduce the errors introduced by random selection.

to the assemblies represented in lightweight file format in PDM and PLM frameworks. In addition, the extension of VSM enables our algorithm to support fuzzy queries and partial matching. This is especially helpful for the designers if they want to query assemblies by simply providing parts or sub-assemblies.

However, there are still some limitations in our current work. For example, while the LFD descriptor is good at capturing the global geometry, it fails to match local details. On this occasion, two similar parts having different functionalities and assembling requirements may be matched false positively, such as the lock nuts and washers. We will seek more powerful 3D CAD part descriptors in our future work, with which the local details of parts, especially the key features for assembling can be characterized. Moreover, by understanding the functionalities of parts, some semantic based retrieval approaches may also be supposed to improve the assembly retrieval performance.

Another limitation is the lack of considering relative positions between query parts. To some extent, relative positions between important parts also provide useful information for the specification of the required assemblies. We shall provide more powerful user interfaces, such that the relative positions between parts can be specified by users and characterized by our system. It is promising that by integrating the position measurement into the matching evaluation, the performance of the relaxed retrieval would be further improved.

Acknowledgements

The research was supported by National Basic Research Program of China (2010CB328001), National Science Foundation of China (61003096), and National High-tech R&D Program (2012AA040902).

References

[1] Hong T, Lee K, Kim S. Similarity comparison of mechanical parts to reuse existing designs. Computer-Aided Design 2006;38(9):973–84.

[2] Bespalov D, Regli WC, Shokoufandeh A. Local feature extraction and matching partial objects. Computer-Aided Design 2006;38(9):1020–37.

[3] Jayanti S, Kalyanaraman Y, Ramani K. Shape-based clustering for 3D cad objects: a comparative study of effectiveness. Computer-Aided Design 2009; 41(12):999–1007.

[4] Bai J, Gao S, Tang W, Liu Y, Guo S. Design reuse oriented partial retrieval of CAD models. Computer-Aided Design 2010;42(12):1069–84.

[5] Iyer N, Jayanti S, Lou K, Kalyanaraman Y, Ramani K. Three-dimensional shape searching: state-of-the-art review and future trends. Computer-Aided Design 2005;37(5):509–30.

[6] Deshmukh AS, Banerjee AG, Gupta S, Sriram RD. Content-based assembly search: a step towards assembly reuse. Computer-Aided Design 2008;40(2): 244–61.

[7] Chen X, Gao S, Guo S, Bai J. A flexible assembly retrieval approach for model reuse. Computer-Aided Design 2012;44(6):554–74.

[8] Wartick S. Boolean operations. Information Retrieval: Data Structures & Algorithms 1992;264–92.

[9] Robertson SE, Sparck JonesK. Relevance weighting of search items. Journal of the American Society for Information Science 1976;27(3):129–46.

[10] Raghavan VV, Wong SKM. A critical analysis of vector space model for information. Journal of the American Society for Information Sciences 1986; 37(5):279–87.

[11] Ricardo B-Y, Berthier R-N. Modern information retrieval. Cambridge (MA): ACM Press; 1999. p. 27–30.

[12] Osada R, Funkhouser T, Chazelle B, Dobkin D. Shape distributions. ACM Transactions on Graphics 2002;21(4):807–32.

[13] Ankerst M, Kastenm G, Kriegel H-P, Seidl T. Nearest neighbor classification in 3D protein databases. In: Proc. ISMB. 1999.

[14] Horn B. Extended Gaussian images. Proceedings of the IEEE 1984;72(12): 1671–86.

[15] Kazhdan M, Funkhouser T, Rusinkiewicz S. Rotation invariant spherical harmonic representation of 3D shape descriptors. In: Proc. Symposium on geometry processing. 2003.

[16] Chen DY, Tian XP, Shen YT, Ouhyoung M. On visual similarity based 3D model retrieval. Computer Graphics Forum 2003;22(3):223–32.

[17] Shilane P, Min P, Kazhdan M, Funkhouser T. The princeton shape benchmark. In: Proc. Shape modeling international. 2004. p. 167–78.

[18] Gal R, Cohen-Or D. Salient geometric features for partial shape matching and similarity. ACM Transactions on Graphics 2006;25(1):130–50.

[19] Funkhouser T, Shilane P. Partial matching of 3D shapes with priority-driven search. In: Proc. Eurographics symposium on geometry processing. 2006.

[20] Zhang H, Sheffer A, Cohen-Or D, Zhou Q, Van Kaick O, Tagliasacchi A. Deformation-driven shape correspondence. Computer Graphics Forum 2008; 27(5):1431–9.

[21] Rustamov RM. Laplace–Beltrami eigenfunctions for deformation invariant shape representation. In: Proc. Eurographics symposium on geometry processing. 2007.

[22] Gal R. Pose-oblivious shape signature. IEEE Transactions on Visualization and Computer Graphics 2007;13(2):261–70.

[23] Bronstein AM, Bronstein MM, Guibas LJ, Ovsjanikov M. Shape google: geometric words and expressions for invariant shape retrieval. ACM Transactions on Graphics 2011;30(1): Article No. 1.

[24] Ma L, Huang Z, Wu Q. Extracting common design patterns from a set of solid models. Computer-Aided Design 2009;41(12):952–70.

[25] Ma L, Huang Z, Wang Y. Automatic discovery of common design structures in CAD models. Computers and Graphics 2010;34(5):545–55.

[26] El-Mehalawi M, Miller RA. A database system of mechanical components based on geometric and topological similarity. Part I: representation. Computer-Aided Design 2003;35(1):83–94.

[27] El-Mehalawi M, Miller RA. A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment. Computer-Aided Design 2003;35(1): 95–105.

[28] Kim GJ. Case-based design for assembly. Computer-Aided Design 1997;29(7): 497–506.
[29] Chen G, Zhou J, Cai W, Lai X, Lin Z, Menassa R. A framework for an automotive body assembly process design system. Computer-Aided Design 2006;38(5): 531–9.
[30] Chao YS, Liu HJ. Case retrieval in body-in-white parts based on similarities of welding and assembly process. Computer Integrated Manufacturing Systems 2011;17(1):30–6.
[31] Liu R, Baba T, Masumoto D. Component parts extraction from assembly drawings for content based retrieval. In: IEEE international conference on visual information engineering. 2005. p. 45–50.
[32] Liu R, Baba T, Uehara Y, Masumoto D, Nagata S. Device parts retrieval from assembly drawings with svm based active relevance feedback. In: Proc. CIVR 2007. 2007. p. 379–86.
[33] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. 3rd ed. New York: The MIT Press; 2009. p. 561–85.
[34] Salton G, Buckley C. Term-weighting approaches in automatic retrieval. Information Processing & Management 1988;24(5):513–23.