Multi-scale Symmetry Detection of CAD models

Chaoren Li[1] , Ming Li[2] , Shuming Gao[3]

[1]State Key Laboratory of CAD&CG, Zhejiang University, P.R. China,

Abstract. Multi-scale symmetries are important to improve shape understanding of CAD models and facilitate many engineering applications such as direct modeling and reverse engineering. In order to extract multi-scale symmetries for various engineering applications, a FSM (Frequent Sub-graph Mining)-based symmetry detection approach is proposed. Firstly, a B-rep model is converted into congruence-labelled graph whose labels imply congruence information among faces and edges. Secondly, complete multi-scale congruence features are detected effectively using the method of frequent sub-graph mining. Thirdly, a set of heuristic filter rules are used to obtain important multi-scale features that fit local features of model well from complete congruence features. Finally, complex symmetry structure of every congruence feature is analyzed using a novel multi-scale congruence features based method. The algorithmic performance is tested on various CAD models exhibiting rich symmetries at multiple scales, and its applications on smart direct modeling and multi-scale model simplification are also shown.

Keywords: Multi-scale symmetry detection, Symmetric structure, Direct modeling, Frequent sub-graph mining
DOI: https://doi.org/10.14733/cadaps.2019.50-66

1 INTRODUCTION

CAD models are often constructed from features and sub-features recursively through different forms of repetition, and contain symmetries at different geometric scales, see Figure 1. Multi-scale symmetry analysis provides multi-level and coarse to fine understanding of the models under study, which helps semantic understanding of the models and facilitates many downstream applications such as reverse engineering, intelligent direct modeling, model simplification and model retrieval etc. In this paper, we are interested in extracting multi-scale symmetries from CAD models in boundary representations (B-rep); the CAD models are always referred to B-rep CAD models in the paper for brevity, unless stated otherwise.

Extracting multi-scale symmetries from CAD models remains a challenging task. The main difficulties come from the challenges that neither the hierarchies of the symmetry structures nor the symmetry relations between different model parts are known in a B-Rep model. Consequently, the symmetries can be detected in two possible ways, either firstly extracting the symmetry scales or building the symmetry relations. However, first detecting symmetry relations is much more difficult, considering the complex compound structure of symmetries among small-scale geometrical parts, see Figure 1 for example, and the difficulty of detecting non-commutative structural regularity [17]. Thus, we choose in this paper to first extract multi-scale symmetric regions and then to extract symmetry relations among these regions.

Obtaining meaningful shared geometrical patterns effectively and efficiently is key to extract multi-scale symmetric regions, as no feature information exists in B-Rep model and there are a large number of different geometrical patterns. In order to resolve the issue, frequent subgraph mining (FSM) method is introduced in our method by mining frequent geometrical patterns in a given model. FSM is essentially a graph mining approach, aiming to extract all frequent subgraphs whose occurrences are above a prescribed threshold. FSM is well developed and lots of approaches have been proposed [6]. Noticing that CAD models are naturally suited to graph representations, detecting congruent geometrical parts of a model sharing the same pattern, or forming potential symmetric regions, can be achieved via applying FSM.

After detecting congruent geometric parts, we next detect of symmetric relations among these congruent geometrical parts. The basic symmetric relations studied in this paper include reflection, translation, rotation relations, which are common and very important symmetry types exhibiting in CAD models. Additionally, compound symmetry relations, or called *symmetry structure*, is also detected, as they represent important design intent of the construction process of model.

In summary, the main contribution of the study is

- The multiscale symmetries are detected from a B-Rep model, which is seldom studied in previous work.
- The sub-graph mining technique is introduced to filter unwanted symmetries and greatly reduces the manual interaction, and can preserve various symmetries at different scales.
- Various numerical examples on realistic CAD models are tested to demonstrate the approach's performance.

1.1 Related work

Multi-scale symmetries from CAD models were seldom studied in previous studies, and the paper is the first on this topic to our best knowledge. Most existing related work on detecting symmetries from CAD models generally focus on extracting global [19, 7] or isolated local [11, 3, 12, 13] symmetries or symmetries at the same scales. They seldom study important compound relations of different symmetries types in the models. Related work on detecting multi-scale symmetries and symmetric structure in the communities of computer graphic and CAD is further explained below.

Pauly et al. [17] use the method of transformation space voting and commutative group theory to discover structural regularity from mesh models. In this study, a variety of important structures are detected using a unified computational framework. However the approach did not consider reflection symmetry and the structural regularity including it.

Wang et al. [20] propose an approach to find a symmetry-induced meaningful hierarchical organization of an input 3D mesh model. They use a set of precedence rules to obtain a novel structural representation of 3D objects. The presentation is built based on certain heuristic rules and provides a *unique* symmetric structure for an input model. Different from the study, multiple reasonable symmetric structures can be obtained through our methods. This novelty of multiplicities is important as different multiple symmetry explanations can be generated from a given model.

Xu et al. [21] extracted multi-scale intrinsic symmetries through two levels of clustering from 3D mesh models. In this approach, prominent and overlapping intrinsic symmetries at multiple scales can be extracted robustly. However the notion of scale here is based on intrinsic distances instead of area of symmetry region, so the approach cannot be applied in our scenario.

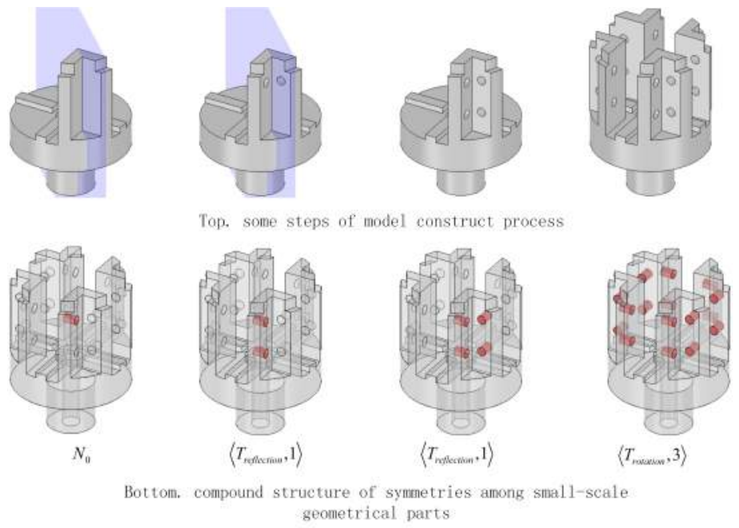

Top. some steps of model construct process
N_0
$\langle T_{reflection}, 1\rangle$
$\langle T_{reflection}, 1\rangle$
$\langle T_{rotation}, 3\rangle$
Bottom. compound structure of symmetries among small-scale geometrical parts

Figure 1: Different forms of repetition of features and sub-features recursively is shown in construct process of model(top) and compound symmetry structure in small-scale geometrical parts brought by recursive repetition (bottom).

Li et al. [13] introduced a concept of regularity feature tree to detect design intent from B-Rep CAD model created by reverse engineering. Broken symmetries are also recovered in these researches [14], and symmetric relations between different parts are also related later on [15]. However, the symmetry relations are essentially built on the same scale. Different from this, the paper studies the challenging problem of detecting hierarchical symmetries are different scales.

2 BASIC CONCEPTS

Basic concepts of our algorithm are introduced in this chapter.

Scale: The scale of a geometrical region is defined by the area enclose by it. Multi-scale geometrical parts represent geometrical parts whose areas vary.

Symmetry: Concept of symmetry is well defined [14, 16]. In the context of geometry, a symmetric structure is a set of entities that are invariant under geometric transformations such as reflections, translations, rotations. In another way, we say that a geometric part M is symmetric with respect to another part M' in a model if and only if a geometrical transformation T exists such that $M = T(M')$. In this case, we say M and M' form a reflectional symmetry, rotational symmetry, or translational symmetry if T is a reflectional, rotational, translational transformation.

In this paper, we focus on reflectional, rotational, translational symmetries and the compound symmetry (*symmetry structure*) formed by their combination; see figure 2 for example.

Congruence: two geometric entities are congruent if they can be transformed into each other under an isometric transformation [3]. If two geometrical entities form reflectional, rotational or translational symmetries with each other, they must be congruent.

Congruence features: Given a model M and its geometrical entity P, if another different congruent geometrical entity P' in M exists, we call P a congruence feature of M.

Congruence group: A congruence group G of a model M is a set of congruence features in M that are congruent with each other; see the examples are shown in Figure 3.

Congruence-labelled Adjacency Graph (CLAG): A concept extended from AAG introduced by Joshi et al. [8]. CLAG of a CAD model is a labelled graph represented as $G(V, E, L_V, L_E)$, in which V is a set of

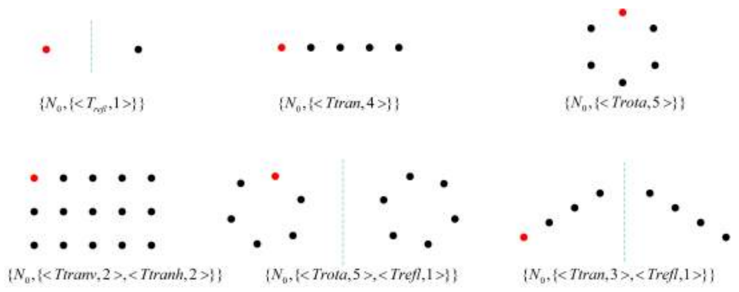

$\{N_0,\{<T_{refl},1>\}\}$
$\{N_0,\{<Ttran,4>\}\}$
$\{N_0,\{<Trota,5>\}\}$
$\{N_0,\{<Ttranv,2>,<Ttranh,2>\}\}$
$\{N_0,\{<Trota,5>,<Trefl,1>\}\}$
$\{N_0,\{<Ttran,3>,<Trefl,1>\}\}$

Figure 2: Symmetries discussed in this paper: top are illustrations of three elementary symmetry type, bottom are examples compound structures formed by the three elementary types.

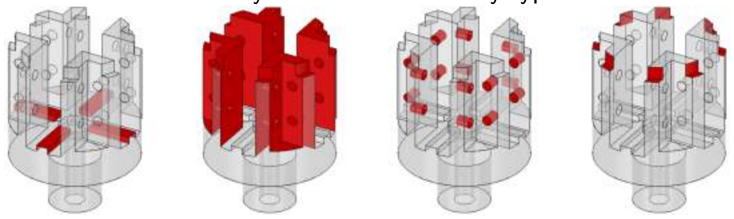
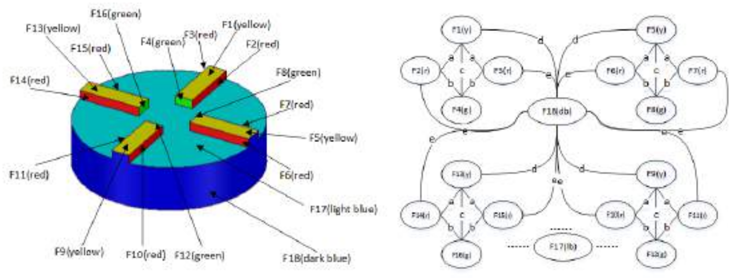


Figure 3: Illustrations of congruence groups whose scales are varied, congruence features in congruence group are marked red in figure.

vertices representing faces of the CAD model, $E \subseteq V \times V$ is a set of edges representing two adjacent faces; L_V and L_E are sets of vertices and edge labels. In addition, we use the same labels for vertices corresponding to congruent faces, and same labels for edges sharing the same underly geometry (See Figure 4 for example).

Frequent sub-graph mining: Frequent sub-graph mining (FSM) is a frequent graph pattern mining technique in data mining field [6]. FSM is used to extract all frequent sub-graphs whose occurrence frequency is larger than a specified threshold in a given graph database or a large graph.

Symmetry structure: Symmetry structure is used to model compound symmetry relations amongst congruence features of a congruence group; see Figure 2 for example, and is formalized as a tuple $(N_0, \{\langle T_i, n_i\rangle\})$. Here, N_0 is a representative congruence feature of G, $\langle T_i, n_i\rangle$ represents a kind of generation operation that maps symmetry transformation T_i onto congruence features that have been obtained n_i times to generate new congruence features. Accordingly, $\{\langle T_i, n_i\rangle\}$ represents all the congruence features in G that can be obtained, and $(N_0, \{\langle T_i, n_i\rangle\})$ represents all congruence features of G and symmetry relations amongst G through a series of generate operations.



vertex
Figure 4: Illustration of CLAG, left is CAD model, right is corresponding CLAG, vertex labels of CLAG are a series of color name, edge labels are marked using alphabet, edges connected to vertex F_{17} are not shown in right figure for clearness

3 APPROACH OVERVIEW

Our multi-scale symmetry detection algorithm takes a B-Rep CAD model as input and returns multi-scale congruence features and their corresponding symmetry structures or generative model. The main idea is to first extract complete multi-scale congruence features of the model, and then to use heuristic rules to filter meaningless congruence features, finally to analyze high-level symmetry structure of each set of congruence features.

Congruent geometrical parts of model may be repeating design features. These features in various domains have different geometrical characteristics, and it is very challenging to obtain them from a B-Rep model that only contains geometrical and topological information. In order to extract complete multi-scale congruence features to meet designer requirements, the method of frequent sub-graph mining is introduced to mine complete multi-scale congruence features from input model.

There are a large number of congruence features, and redundant congruence relations exist among them. Thus, a set of geometric heuristic rules are further used to filter congruence features that are meaningless in most situations so that rich and non-redundant congruence relations can be kept after filtering.

At last four main types of important symmetry relations and symmetry structure are detected for every congruence feature: reflection symmetry, rotate pattern, linear pattern and rectangular pattern.

The overall steps of the approach are explained below.

1. Step 1: Convert input model to a graph formulation CLAG (congruence-labelled adjacency graph) for multi-scale congruence feature extraction.

2. Step 2: Use Frequent sub-graph mining(FSM) [6] method to mine in CLAG, and to extract complete multi-scale congruence features. FSM is a frequent graph pattern mining technique in data mining field. It is used to extract all frequent sub-graphs whose occurrence frequency is larger than a specified threshold in a given data set, given data set can be a graph database or a large graph.

3. Step 3: Filter meaningless congruence features from features obtained in step 2;

4. Step 4: Symmetry structure are detected for every congruence feature that is kept in step 3.

Pseudo code of the whole approach is showed in Algorithm 1.

Algorithm 1: Multi-scale symmetries detection

```
Input :
    $A$: input B-Rep CAD model
Output :
    $N$: a set of multi-scale congruence feature groups
    $G$: symmetry structure of corresponding congruence feature group
1: $CLAG$ = CLAGConstruction($A$)
2: $C$ = FSMOfConGrpExtract($CLAG$,$A$)
3: $N$ = Filter($C$)
4: Set $G$ to empty set
5: for congruence feature group $N_i$ in $N$ do
6:     $G_i$ = SymStruDetect($N_i$,$N$), add $G_i$ to set $G$
7: end for
8: return $N$,$G$
```

4 TECHNICAL DETAILS

In this section, technical details on each step of the overall algorithm are further explained.

4.1 Construction of CLAG

CLAG is a kind of graph structure that contains congruence information of a CAD model. CLAG is used to extract shared geometrical patterns via applying FSM, and the patterns are to be used for obtaining important symmetric relations later on. Storing congruence information in CLAG is key issue here, and we use the same label to represent the congruence relation amongst various geometric entities.

One key issue to solve in this step is to determine whether two faces or edges are congruent. It is solved here via converting it into a problem of discrete point alighnment problem using the approach proposed in [3]. Specifically, a set of representative feature points are first extracted from the geometrical entities, which is sufficinet to unqiuely represent the geometry of the entities. These points includes those asosciated to all vertices of an entity, together with some key points sampled following a certain sampling way. These points are determined based on the geometry of each entity and does not depend on specific boundary representation of the model. After this, we then convert the problem of congruence detection into a problem of determining whether an isometry exists between the two sets of feature points. Further algorithmic details on generating the sample points, and setting an appropriate mapping tolerance are referred to [3].

Built on this, the CLAG construction process is explained below:

1. Step 1: Cluster congruent faces of model A into the same set, and give every set a different face label.
2. Step 2: Based on the results obtained in step 1 for each face, add a corresponding vertex label to each vertex of CLAG based on its associated face label.
3. Step 3: Determine the edge label for each edge in CLAG based on its adjacent faces in model A. If the two edges share the same adjacent faces, add the same edge label in CLAG; otherwise, generate a new edge label for it.

4.2 Extract multi-scale congruence groups

In order to extract multi-scale congruence group, the FSM method is first used to mine frequent subgraph patterns and to get the corresponding embeddings from CLAG. As one-one mapping exists between vertex set of CLAG and face set of model, every pattern and its embeddings have a corresponding candidate congruence group whose members are potentially congruent.

After this, congruence-based clustering is carried out for each candidate congruence group to get congruence groups whose members are indeed congruent through geometrical verification. As FSM method can mine all the frequent patterns in CLAG, all the congruence groups in a model can be obtained via the proposed approach. It also direclty works for different sizes of region areas, or across multiple scale.

The pseudo code of this stage is depicted in Algorithm 2. We first introduce the important concept of embedding in FSM. If a sub-graph G_i of input graph G is isomorphic to a given graph pattern P, G_i is called one embedding of P in G, as illustrated in Figure 5. In fact, FSM is used in our method to extract all frequent sub-graph patterns whose frequency of occurrence, namely count of embeddings, is above a given threshold from CLAG. The whole process includes three steps: generating candidate sub-graph pattern; detecting all embeddings of the patterns generated from CLAG; computing frequency of occurrence of the current candidate patterns, pruning patterns whose frequencies are below a given threshold. The above three steps repeat until all processed. The main issue in this process is how to generate candidate sub-graph pattern and to compute

Algorithm 2: Congruence Groups Extraction Using FSM (FSMOfConGrpExtract)

```
Input :
    $A$: input B-Rep CAD model
    $CLAG$: Congruence-labelled adjacency graph of $A$
Output :
    $\{C_i\}$: Complete Multi-Scale Congruence Groups
1:  $PatSet \leftarrow$ empty sub-graph pattern set of $CLAG$
2:  $EmbeddingsSet \leftarrow$ empty set of Embeddings Set
3:  $P_i$ = FFSM-Generate($PatSet$,$CLAG$), generate new sub-graph pattern
4:  while $P_i$ is not empty pattern do
5:      $\{E_i\} \leftarrow$ FindEmbeddings($P_i$,$CLAG$), find all embeddings of $P_i$ in $CLAG$
6:      if cardinality of $\{E_i\}$ is greater than 1 then
7:          add $P_i$ to $PatSet$, add $\{E_i\}$ to $EmbeddingsSet$
8:      end if
9:      $P_i$ = FFSM-Generate($PatSet$)
10: end while
11: $\{C_i\} \leftarrow$ empty set of congruence group
12: for each embedding set $\{E_i\}$ in $EmbeddingsSet$ do
13:     $\{C_j\} \leftarrow$ Congruence-Cluster($\{E_i\}$,$A$)
14:     add $\{C_j\}$ to $\{C_i\}$
15: end for
16: return $\{C_i\}$
```

frequency of occurrence of candidate patterns efficiently. Note that find all embeddings of candidate sub-graph pattern involves a complex process of detecting isomorphic subgraphs within an input graph, which is essentially a NP-hard problem and time-consuming.

We use the CAM (Canonical Adjacency Matrix), proposed in [5], to improve efficiency of the time-consuming process of frequent subgraph mining. CAM is a canonical labelling strategy, and it takes maximal lexicographic order in all sequences obtained from adjacency matrices of a given graph. It is to identify one graph pattern uniquely and facilitates isomorphism checking. The main behind it is that the isomorphic graphs have the same canonical labels and the canonical labels of non-isomorphic graphs are non-isomorphic. CAM has suboptimal property, exploring CAM tree construct of suboptimal CAMs can generate candidate sub-graph pattern efficiently. Based on these consdierations, we choose CAM as our labelling strategy and FFSM (fast frequent sub-graph mining) [5] to implement function FFSM-Generate, which is used to generate candidate sub-graph patterns for efficiency improvement.

The anchor edge technology used in HSIGRAM method in [9] is also introduced in the proposed approach to find all embeddings of a given pattern to speed up computing efficiency. The anchor edge is used as a constraint of subgraph isomorphism to reduce search space as we only need to search subgraph around anchor edge. The technology can help to keep balance between space and time expenses, based on which we choose to implement the function FindEmbeddings.

Corresponding geometrical entities of embeddings in a set are extracted after using FSM. These geometrical entities are candidate congruence features that form candidate congruence group. After this, geometrical verification is applied on these geometrical entities through extracting feature point set from geometrical parts and verifying whether two feature point set are congruent according to the method proposed [3]. At last congruent geometrical parts are clustered to one same congruence group. These are what function Congruence-Cluster does.

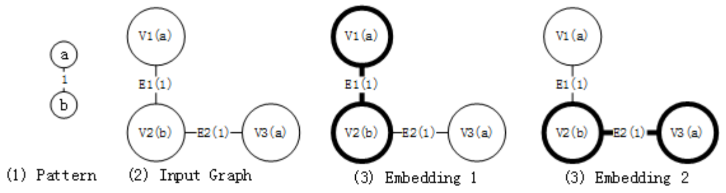


Figure 5: Illustration of overlapped embeddings

Finally we introduce two important details in this stage. First is that threshold used in our FSM method is 1 because reflection symmetry is one common and important type, if one pattern occurs more than 1 times in model, it's candidate reflection symmetry. In order to explore all possible situations, costs of FSM is huge as threshold is low, lots of mature parallel computing methods can be used to speed the whole process, such as GPU [18], map-reduce [1], and this is also one advantage of introducing FSM algorithm. Second is that the frequency count of pattern is using maximum independent set (MIS) proposed in [9] when there are situations that embeddings of pattern overlap with each other. As is shown in figure 5, the frequency of pattern is 1 instead of 2 in our paper.

4.3 Congruence group filter

After multi-scale congruent groups are extracted from a CAD model in the last stage, some congruence groups are still redudant in most engineering applications. Actually, there are too many duplicate congruence information among them because some congruence groups are covered by other congruence groups. Obtaining fewer but better congruence information is very important for practical applications. The following three heuristic rules are thus proposed to obtain congruence features that are thought most important.

1. Rule 1. Congruence groups whose members overlap in some areas should be filtered;

2. Rule 2. Maximal congruence group should be kept;

3. Rule 3. Congruence group whose members are consistent with local feature should be kept.

The three rules are also illustrated in Fig. 6.

In general situation, a design that two repeated features share the common parts is very rare. But there are a large number of congruence groups got in last stage that their members overlap. Rule 1 is used in our filter process as the advantages far outweight the disadvantages. If any two members of one congruence group do not contain overlapped part, then this congruence group is non-overlapped congruence group. Rule 1 is used to obtain non-overlapped congruence groups. Illustrations of non-overlapped congruence group are in Figure 6 left.

As mentioned before, there are too many duplicate congruence information among congruence groups obtained in 4.2. The reason is that members of some congruence groups are sub-region of other congruence groups. We say a non-overlapped congruence group $G = \{G_i\}\,(i = 0, 1, \dots)$ congruence-cover another non-overlapped congruence group $N = \{N_i\}\,(i = 0, 1, \dots)$ if G and N have same member count and there is one to one mapping between members of G and N so that every member of N is sub-region of corresponding member of G. If no congruence group can congruence-cover one congruence group, we say it has local-maximum property and call it maximal congruence group. Maximal congruence group is special kind of congruence group that contains maximum congruence information because any non-maximal congruence feature has one corresponding maximum congruence feature that covers all congruence relations of it. Illustrations of maximal congruence feature are in Figure 6 middle. In order to obtain rich and non-redundant congruence information, rule 2 is proposed to extract all maximal congruence features.

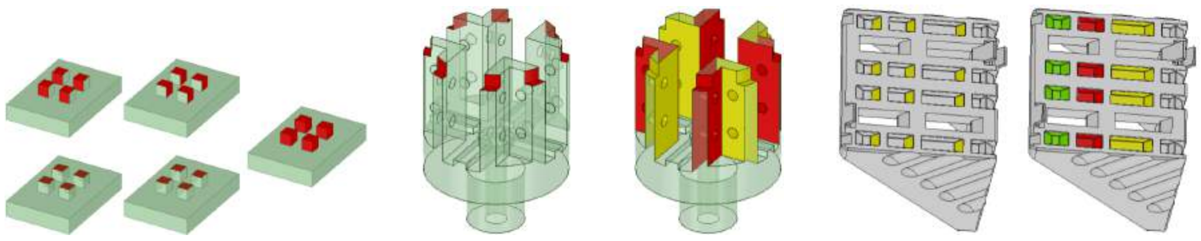


Figure 6: Illustration of three congruence group filter rules. Left: the left is a congruence group that contains 16 members that overlap, while the right is non-overlapping congruence group that contains 4 features made up of the 16 entities in the left. Middle: the left is a non-overlapping congruence group that contains 8 members, while the right is the corresponding maximal congruence group that contains the congruence group on the left. Right: the left is a congruence group whose member contains only a single face, while the right is the corresponding local features.

Congruence groups whose members contain only one single face are meaningless in most situations as feature-level congruence information is more important. In the view of humans, model often is decomposed into multiple parts through concavity and convexity analysis and locality of faces. Every face has one corresponding local geometrical parts. One face may belong to members of multiple maximal congruence groups, but generally a unique one exists whose member best matches the local shape of the face. We think this maximal congruence feature is most important, so rule 3 is proposed to obtain such maximal congruence feature, Illustrations are in Figure 6 right.

We next introduce the congruence feature filter process.

1. Step 1. Check whether members of every congruence group overlap. If they do overlap, filter the corresponding group; otherwise, keep the congruence group. After step 1, the set of non-overlapped congruence group is obtained.

2. Step 2. Check whether every non-overlapped congruence group is a maximal congruence group. Filter those congruence groups that are not maximal. As mentioned above, exploring CAM tree method is used in our FSM method, one given congruence group can be judged whether is maximal effectively through utilizing parent-child relation in CAM tree. After step 2, the set of maximal congruence groups is obtained.

3. Step 3. Perform local feature analysis for every face. Then get local shape of every face, and find maximal congruence feature whose member wrap the corresponding local shape most compactly. After this, add the congruence group to final result set. After step 3, the final set of congruence groups is obtained.

One important part of our filter process is the local feature analysis for a given face as involved in step 3. Convexity and concavity are used here because they are good clues to segment models. PCAAG and PCVAG are proposed to utilize convexity and concavity. PCAAG is a kind of special attributed adjacency graphs (AAG) of model proposed in [4], and there is one path for any two vertexes of PCAAG and every edge of the path is composed of concave arc. Partly convex adjacency graph (PCVAG) can be define in the same way of PCAAG, just replace concave to convex. PCAAG and PCVAG can be used to recognize concave and convex geometrical parts of model, but sometimes convex (concave) parts obtained using PCVAG (PCAAG) is too large. Thus we only want to obtain local feature around given face. The concept of large-scale face is proposed to solve this problem.

1. Step 1. Initialize faces set of result local feature **FaSet** to be input face F_i.

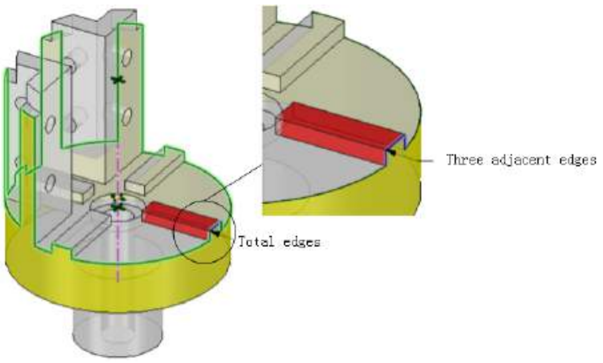


Figure 7: Illustration of large-scale face evaluating, Fi of AdFaSet is yellow, faces of FaSet is red, ratio of edge length is small in the example, so Fi is a large-scale face of FaSet

2. Step 2. Find adjacent faces set **AdFaSet** of **FaSet**, whose members has at least one convex adjacent edge with the faces in **FaSet**.
3. Step 3. For every face F_j in **AdFaSet**, evaluate whether F_j is a large-scale face for current **FaSet**. If we can find a face F_k that is not a large-scale face, add F_k to **FaSet** and then jump to step 2. If we cannot find such face or **AdFaSet** is empty set, jump to step 4.
4. Step 4. The face set **FaSet** is a local feature that contains input face F_i.

Large-scale face is used in step 3, and the objective is to obtain a good local feature in case the convex or concave feature does not fit local well. How to determine whether a face F_i of **AdFaSet** is a large-scale face for **FaSet** is a key issue. Heuristic method is based on observations that, large-scale face F_i is usually adjacent with other large-scale faces that does not belong to **FaSet**, and the adjacent edges E_a of F_i with **FaSet** are a small portion of its total edges E_t. So the ratio of edge length of E_a with E_t is used to evaluate whether F_i is a large-scale face with **FaSet**, as illustrated in Figure 7.

In the end, we comment here that the case of "virtual edge" may exist in a CAD models. For example, two adjacent cylindrical or planar faces may be separated by "virtual" edges, although they actually have the same underlying surface, and should be merged into a single one. The case can be overcome with the concept of maximal decomposition primitives in [12].

4.4 Symmetry structure detection

We solve the problem of detecting multi-scale symmetries, through extracting multi-scale firstly, then detecting symmetries. As introduced in previous sections, symmetry structure formed by reflection, rotation and translation symmetry are symmetries that we are interested in as it cover most important symmetry relations in CAD models. The symmetry structure detection process includes two aspects, compound manner and base symmetry types, that is, the base symmetry types and how do these base symmetries form together symmetry structures. The difficulty in detecting symmetry structure is how to decide the compound manner of symmetries among members of congruence group. Exploring compound manner is very challenging as the number of arbitrary combinations of the three base symmetries is very large. Benefits from multi-scale congruence groups obtained in last stage, the compound manner can be extracted easily through multi-scale analysis. Then classic symmetry detection algorithm is used to detect base symmetry types after compound manner is got, so that complete symmetry structure can be achieved finally. How to get compound manner and base symmetry types detection algorithm are explained in next sections.

Using multi-scale analysis to extract compound manner is based on one observation: CAD model are formed from repeated features and sub-features recursively. Because there are symmetries among sub-features belonging to one parent feature, there are symmetries among the parent features, which results in complex symmetry structures among sub-features of all parent features. Therefore, if we know symmetry structure of parent features and symmetric relations among sub-features that belong to one parent features, symmetry structure of all sub-features can be obtained through combing the two symmetries. Sub-features form small scale congruence groups and parent features form large scale congruence groups in our context.

Scale-over is introduced to model this kind of relation among multi-scale congruence groups. If one member of congruence group N contains multiple members of congruence group G, we call N **scale-over** G. If there is no congruence group K so that N **scale-over** K and K **scale-over** G, then we call G is child of N, N is parent of G. If N is parent of G, then symmetries among members of N is applicable to members of G because members of G are sub-regions of members of N. This kind of parent-child relations exist among multi-scale congruence groups are used to analyze compound manner of symmetries among members of small scale congruence group. Assume that symmetry structure of N is $\left(N_0, \{\langle T_i^N, n_i\rangle\}\right)$, N_0 is one congruence feature of N, $(G_0, G_1, \cdots, G_t)$ are members of G and sub-regions of N_0 as N is parent of G. After symmetry detection is carried out on $(G_0, G_1, \cdots, G_t)$, one generate step $\left(G_0, \{\langle T^G, t\rangle\}\right)$ can be obtained, then symmetry structure of G is $\left(G_0, \{\langle T^G, t\rangle, \{\langle T_i^N, n_i\rangle\}\}\right)$. In order to utilize parent-child relations to analyze symmetry structure of congruence groups efficiently, global scale-over trees whose vertex is congruence group are set up firstly. Then a top-down symmetry structure analysis is carried out for these trees using depth-first traversal and method above.

Base symmetry type detection is the next work after solving the problem of compound manner. It is resolved based on the following assumption: compound relations of symmetry structures in CAD models can be solved using multi-scale analysis. We focus on reflection, rotation, translation symmetries in this step. Reflection symmetry is obtained through mirror plane vote method similar to [19]. Classic point-based method proposed by Peter Brass [2] is used to extract rotate pattern and linear pattern from congruence features through using center of gravity of feature point set of them. Rotation symmetry and translation symmetry can be extracted from rotate pattern and linear pattern easily.

Compound relations of symmetry structure rely on multi-scale parent-child relations analysis totally in our method, our base symmetry detection algorithm just detect base symmetry types and choose one best type. Key issue here is to determine the most important symmetry relations of congruence features if there are multiple choices. A set of heuristic rules are proposed to determine this. Patterns are prior to reflection symmetry because there are a large number of reflection symmetry relations among patterns, for example, any two member of rotate pattern is reflection symmetric with each other. Patterns whose count of members is larger is more important because regularity is stronger if count of members is larger. Reflection symmetry relation whose pairs is larger is more important because we think one mirror plane that is shared by more symmetry pair is more important.

5 RESULTS AND APPLICATIONS

In this section, a range of complex CAD models are used to test two important aspects of our algorithm including multi-scale congruence groups extraction and symmetry structure detection. We also show how the results can be used to two important applications of smart direct modeling and multi-scale model simplification.

Multi-scale Congruence Group Extraction. Figure 8 shows results of multi-scale congruence group extraction for 6 different CAD models. For each model, the source model is marked red, and the same color shows scale-cover parent-child relations of the corresponding congruence groups. Figure whose borders are dashed show one member of congruence group for clearness as members are hard to distinguish sometimes.

Anc101_a. Four congruence groups are extracted from the model. Members of the congruence group are good features including hole and concave features, especially the congruence groups marked by yellow border.

Its members are two-level concave features, any single level concave feature does not appear in result, which claims that our filter rules are effective.

Kim is also a classic model in feature recognition domain. Note the congruence groups marked by blue border, the scale-over parent-child relations are rich among these congruence groups, which can be used to analyze complex symmetry structures of the smallest scale congruence group. Especially both four elements and eight elements congruence groups marked by blue border are extracted, which shows the comprehensiveness of our multi-scale extraction algorithm.

Drivetrain is a practical CAD model whose scales are rich and the compound manner of symmetries is rich. The multiple branches among congruence groups marked by yellow border show that our method can got various symmetry explanation of model. Because 4-element and 6-element congruence groups sharing the same parent are both extracted in our result, and members of them overlap with each other, it ultimately leads to different results in [20]. The method proposed in [20] only chooses one symmetry type of them to get its symmetry hierarchy, which may lead to unexpected case in losing some symmetry information.

Monster, **Engine**, **Focus** are all practical CAD models. 10 elements and 12 elements congruence groups marked by yellow border in **Monster** show two interpretations of the linear pattern, including concave feature interpretation and convex feature interpretation as both of them can fit local feature well. The fact shows that our algorithm does not lose reasonable congruence groups arbitrarily. Scale of members of 6 elements congruence groups marked by blue border in **Engine** are very small features, which reflects that our algorithm does not ignore rich symmetries among small features.

Focus is another example to show that our filter rules are effective. There are a large number of congruence groups extracted from **Focus** before filter, because there are various repeated geometrical patterns in the model due to the translation and rotation patterns, but only four reasonable congruence groups are left after the filter process.

Symmetry Structure Detection. Figures 9 and 10 show results of symmetry structure detection for congruence groups. Six congruence groups that are small-scaled and have complex symmetry structure of different forms are used here to show the effectiveness of our algorithm. Geometrical parts marked red in the figure with red border show the one member of congruence group, and we use manner of generation to demonstrate the symmetry structure. Different compound manners of base symmetries are showed in our examples, where one example showing compound of reflection symmetries, two examples showing compound of reflection symmetry and translation symmetry, two examples showing compound of reflection symmetry and rotation symmetry, one example showing compound of translation symmetry and rotation symmetry. As showed in the results, reflection symmetry type can be handled well using the proposed approach, different from previous approach proposed in [17] that is difficult to handle such symmetry types.

Smart Direct Modeling. Figure 11 shows two examples of smart direct modeling. Important symmetric semantics is obtained through our multi-scale symmetries analysis. User can keep important symmetry semantics of model unchanged during direct modeling process to make editing process more intelligent as it's shown in figure 11.

Multi-Scale Model Simplification. Figure 12 shows an example of multi-scale model simplification. Multi-scale congruence groups are extracted and scale-cover relations among them are analyzed in our algorithm, so multi-scale model simplification is performed through filling all congruence features whose scale is smaller than specified scale utilizing scale-cover relations obtained.

6 CONCLUSION

The paper presents a novel approach to detect multi-scale symmetries from CAD models. These multi-scale congruence groups are believed an important aspect for various engineering applications. One important contribution of the study is that the introduced FSM method provides a customizable framework to extract different kinds of multi-scale congruence groups from CAD models through different pruning strategies. In

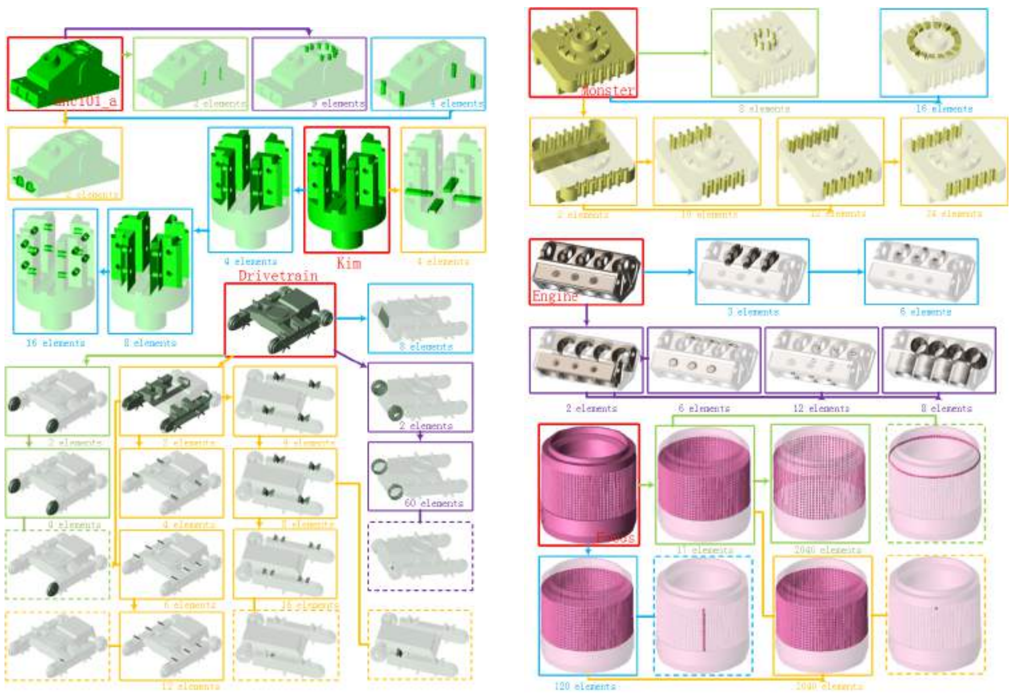


Figure 8: Multi-scale Congruence Group Extraction Result

addition, a novel approach to analyze symmetric structure of congruence groups is also proposed. Performance of the proposed approach was also applied to two important applications, the intelligent direct modeling and multi-scale model simplification.

Limitations and future work. Although important multi-scale symmetries are detected from CAD models using the proposed approach, it mainly works for regular models, that is, models composed of planes and quadratic surfaces. Free-form CAD models are not taken into account in the study because multi-scale congruence detection of free-form surfaces is a challenging issue and requires further research efforts. Further taking into account of concpets like CAD feature-tree may help to resolve the issue [10].

The proposed symmetry detection algorithm at present mainly works for structures formed through mirror symmetry, translational symmetry, rotatation symmetry in different geometric scales. Although such symmetries cover most popular symmetries in CAD models, future research efforts are still needed to detect more complex symmetric structures formed through composition of these symmetries. In particular, the approach is only valid when there is no interactions between features. Other techniques have to be further developed to handle cases of interaction features.

In the end, we also point out that the applied FSM technique, although very effective in detecting various congruence relations, is on the other hand also very time consuming as exhaustive congruence patterns have to be explored. Due to these reasons, directly applying the approach to highly complex models containing a variety of frequent patterns may be too time consuming for practical applications. Developping more advanced pruning techniques or including geometric heuristics in the FSM approach is still required for efficiency improvement.

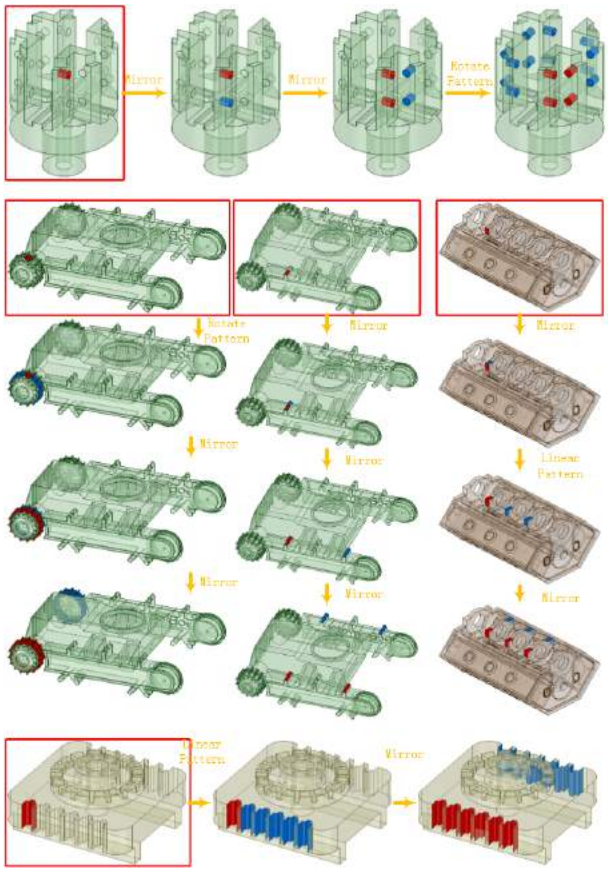


Figure 9: Symmetry Structure Detection Result 1

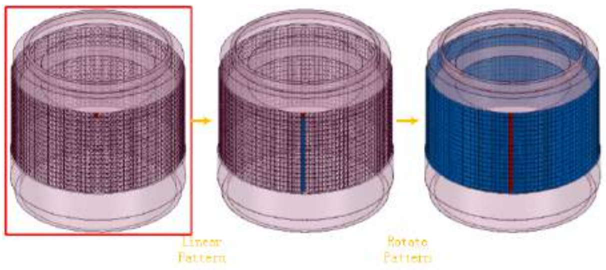


Figure 10: Symmetry Structure Detection Result 2

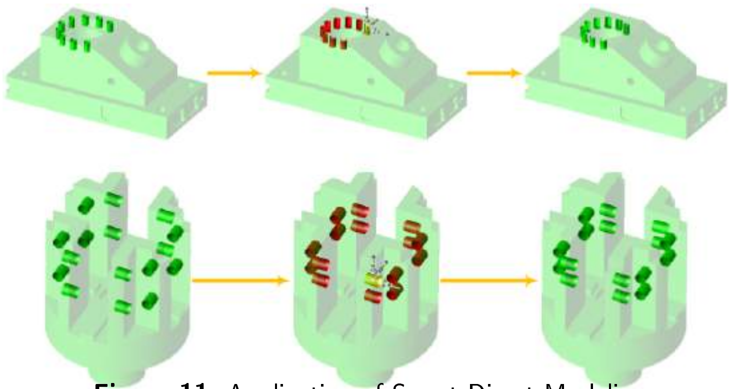


Figure 11: Application of Smart Direct Modeling

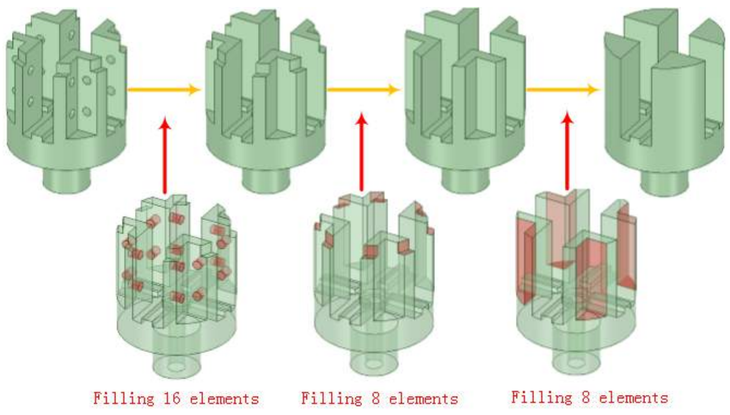


Figure 12: Application of Multi-scale Model Simplification

ACKNOWLEDGEMENTS

The work described in this paper is partially supported by the NSF of China (No. 61472356), and the National Key Research and Development Program (2016YFC1101302) from the MIST of China.

ORCID

Ming Li, http://orcid.org/0000-0002-9711-0745
Shuming Gao, http://orcid.org/0000-0002-7061-9912

REFERENCES

[1] Alkan, S.: A distributed graph mining framework based on mapreduce. Master's thesis, Computer Engineering Department, Middle East Technical University, 2010.

[2] Brass, P.: On finding maximum-cardinality symmetric subsets. Computational Geometry, 24(1), 19–25, 2003. http://doi.org/http://doi.org/10.1016/S0925-7721(02)00046-9.

[3] Gao, C.; Langbein, F.C.; Marshall, A.D.; Martin, R.R.: Approximate congruence detection of model features for reverse engineering. In Shape Modeling International, 2003, 69–77. IEEE, 2003. http://doi.org/https://doi.org/10.1109/SMI.2003.1199603.

[4] Gao, S.; Shah, J.J.: Automatic recognition of interacting machining features based on minimal condition subgraph. Computer-Aided Design, 30(9), 727–739, 1998. http://doi.org/http://doi.org/10.1016/S0010-4485(98)00033-5.

[5] Huan, J.; Wang, W.; Prins, J.: Efficient mining of frequent subgraphs in the presence of isomorphism. In Third IEEE International Conference on Data Mining, 2003., 549–552. IEEE, 2003. http://doi.org/http://doi.org/10.1109/ICDM.2003.1250974.

[6] Jiang, C.; Coenen, F.; Zito, M.: A survey of frequent subgraph mining algorithms. Knowledge Engineering Review, 28(1), 75–105, 2013. http://doi.org/https://doi.org/10.1017/S0269888912000331.

[7] Jiang, J.; Chen, Z.; He, K.: A feature-based method of rapidly detecting global exact symmetries in CAD models. Computer-Aided Design, 45(8), 1081–1094, 2013. http://doi.org/https://doi.org/10.1016/j.cad.2013.04.005.

[8] Joshi, S.; Chang, T.C.: Graph-based heuristics for recognition of machined features from a 3D solid model. Computer-Aided Design, 20(2), 58–66, 1988. http://doi.org/https://doi.org/10.1016/0010-4485(88)90050-4.

[9] Kuramochi, M.; Karypis, G.: Finding frequent patterns in a large sparse graph. Data Mining and Knowledge Discovery, 11(3), 243–271, 2005. http://doi.org/http://doi.org/10.1007/s10618-005-0003-9.

[10] Li, K.; Foucault, G.; Leon, J.; Trlin, M.: Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical components. Computer-Aided Design, 53(5), 70–89, 2014. http://doi.org/http://doi.org/10.1016/j.cad.2014.03.005.

[11] Li, K.; Foucault, G.; Leon, J.C.: Symmetry plane detection for 3D CAD volumes. In ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 465–477. American Society of Mechanical Engineers, 2010. http://doi.org/https://doi.org/10.1115/DETC2010-28338.

[12] Li, K.; Foucault, G.; Leon, J.C.; Trlin, M.: Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical components. Computer-Aided Design, 2014. http://doi.org/https://doi.org/10.1016/j.cad.2014.03.005.

[13] Li, M.; Langbein, F.C.; Martin, R.R.: Constructing regularity feature trees for solid models. In Geometric Modeling and Processing-GMP 2006, 267–286. Springer, 2006. http://doi.org/https://doi.org/10.1007/11802914_19.

[14] Li, M.; Langbein, F.C.; Martin, R.R.: Detecting approximate incomplete symmetries in discrete point sets. Computer-Aided Design, 40(1), 76–93, 2008. http://doi.org/https://doi.org/10.1145/1236246.1236294.

[15] Li, M.; Langbein, F.C.; Martin, R.R.: Detecting design intent in approximate CAD models using symmetry. Computer-Aided Design, 42(3), 183–201, 2010. http://doi.org/https://doi.org/10.1016/j.cad.2009.10.001.

[16] Mitra, N.J.; Pauly, M.; Wand, M.; Ceylan, D.: Symmetry in 3D geometry: Extraction and applications. In Computer Graphics Forum, vol. 32, 1–23. Wiley Online Library, 2013. http://doi.org/https://doi.org/10.1111/cgf.12010.

[17] Pauly, M.; Mitra, N.J.; Wallner, J.; Pottmann, H.; Guibas, L.J.: Discovering structural regularity in 3D geometry. In ACM Transactions on Graphics, vol. 27, 43, 2008. http://doi.org/https://doi.org/10.1145/1399504.1360642.

[18] Ray, A.; Holder, L.B.: Efficiency improvements for parallel subgraph miners. In FLAIRS Conference, 2012.

[19] Tate, S.J.; Jared, G.E.: Recognising symmetry in solid models. Computer-Aided Design, 35(7), 673–692, 2003. http://doi.org/https://doi.org/10.1016/S0010-4485(02)00093-3.

[20] Wang, Y.; Xu, K.; Li, J.; Zhang, H.; Shamir, A.; Liu, L.; Cheng, Z.; Xiong, Y.: Symmetry hierarchy of man-made objects. In Computer Graphics Forum, vol. 30, 287–296, 2011. http://doi.org/https://doi.org/10.1111/j.1467-8659.2011.01885.x.

[21] Xu, K.; Zhang, H.; Jiang, W.; Dyer, R.; Cheng, Z.; Liu, L.; Chen, B.: Multi-scale partial intrinsic symmetry detection. ACM Transactions on Graphics, 31(6), 181, 2012. http://doi.org/https://doi.org/10.1145/2366145.2366200.