Detecting Approximate Incomplete Symmetries in Discrete Point Sets

M. Li F. C. Langbein R. R. Martin
School of Computer Science, Cardiff University, UK
{M.Li,F.C.Langbein,R.R.Martin}@cs.cf.ac.uk

Abstract

Motivated by the need to detect design intent in approximate boundary representation models, we give an algorithm to detect *incomplete symmetries* of discrete points, giving the models' potential local symmetries at various automatically detected tolerances. Here, incomplete symmetry is defined as a set of incomplete cycles which are constructed by, e.g., a set of consecutive vertices of an approximately regular polygon, induced by a single isometry. All seven 3D elementary isometries are considered for symmetry detection. Incomplete cycles are first found using a tolerance-controlled point expansion approach. Subsequently, these cycles are clustered for incomplete symmetry detection. The resulting clusters have well-defined, unambiguous approximate symmetries suitable for design intent detection, as demonstrated experimentally.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling—Geometric Algorithms, Languages, and Systems; J.6 [Computer-Aided Engineering]—Computer-Aided Design.

Keywords: approximate incomplete symmetry, design intent, reverse engineering.

1 Introduction

Many manufactured objects exhibit global and local symmetries as a feature of their design or function, or for ease of manufacturing or analysis [Mills et al. 2001b]. Also, designers prefer symmetric shapes for reasons of aesthetics and simplicity [Barratt 1989]. Explicit detection of such symmetries in geometric models has uses in speeding up analysis, enforcing constraints in editing, and so on. We are thus interested in detecting the design intent of an approximate boundary representation (B-rep) model as represented by its symmetries. We assume that the model is *approximate* as it may have been transferred from a CAD system with large tolerances, or it may have been created by a reverse engineering system [Varady et al. 1997] in which errors may arise due to measurement, approximation, numerical algorithms, and even a worn source object.

Current methods of geometric design intent detection can detect global approximate symmetries [Mills et al. 2001a], approximate congruences between sub-parts [Gao et al. 2003], and other local regularities, e.g. parallel and orthogonal planes [Langbein et al. 2004]. However, local symmetries still need to be considered: e.g. the model in Figure 6(b) has cylindrical holes with rotational symmetries and slots with translational symmetries. Symmetries are often incomplete in the sense that repetitions are missing: e.g. one or two of the cylindrical holes in Figure 6(b) might be missing. Translational symmetry is always incomplete due to its infinite nature. Symmetries may need to be merged if they are induced by the same isometry, e.g. the translational symmetries of the slots on each side in Figure 6(b).

To detect local symmetries, we use similar ideas to those used for global symmetry detection in [Mills et al. 2001a]. We extract *representative points* from a solid model: special points which uniquely characterise each model entity (including vertices, edges and faces). Symmetries of the representative points are sufficient to indicate symmetries of the underlying shapes. Using these representative points, we describe an algorithm to detect *approximate incomplete symmetries* as subsets of discrete point sets. By *symmetry* we mean an isometry which maps a subset of the point set onto itself, represented as a point permutation. As we assume that the input model is *approximate*, the subset may only be mapped approximately onto itself under the symmetry. By *incomplete* we mean that not all points needed for the symmetry are present in the subset, e.g. six vertices of a regular octagon.

Although exact symmetry detection has been widely studied these methods cannot be directly extended to *approximate* symmetries by simply replacing tests for equality by tests for approximate equality. Algorithms for exact symmetry detection rely on making *local* decisions about which elements match under symmetry. For approximate symmetries, such decision must be based on *global* properties. Point matching is no longer a Boolean property—points match to a certain degree, and in general, multiple potential matches have to be considered, increasing algorithmic complexity.

We detect approximate incomplete symmetry using two steps. First we detect incomplete cycles induced by some isometry; a complete cycle represents a special case of an incomplete cycle. Given a single point, its orbit under repetitive application of an isometry gives a complete cycle. If certain points are absent in the input point set, an incomplete cycle occurs. As we consider approximate symmetries, the points are not mapped exactly onto each other, but only within a certain tolerance. To detect approximate incomplete cycles, we build on our earlier approach [Li et al. 2006] for complete approximate cycles using a point expansion technique, starting from approximate isosceles triangles as seed sets. The expansion process is strictly based on our definition of approximate symmetry to control possible accumulated errors during the expansion process.

In the second step we merge incomplete cycles induced by the same isometry. Due to their approximate nature, we have to carefully decide whether different cycles are induced by the same isometry. This isometry has to map the points of each cycle approximately onto each other such that the points in the original set and its image match uniquely. Such matching is used to provide a similarity measure between every pair of distinct incomplete cycles. Based on this, a cycle clustering algorithm is used to generate incomplete symmetries as incomplete cycle clusters. As demonstrated experimentally, the detected approximate symmetries do seem suited for design intent detection.

We consider all elementary isometries in 2D and 3D: reflection, inversion, translation, rotation, glide-reflection, rotation-reflection, and screw translation: see Section 3. A uniform algorithm is provided to detect all of these as sets of incomplete cycles mainly

based on point distance computations. A minimum number l of consecutive points is prescribed for each cycle $(P_1,\ldots,P_l)$ such that an isometry inducing the cycle can be deduced by mapping $(P_1,\ldots,P_{l-1})$ to $(P_2,\ldots,P_l)$. E.g. for an n-fold rotation, $n>3$, four points are needed to determine the isometry *and* ensure we have a rotation.

This approach allows us to automatically determine tolerances at which incomplete symmetries are present, and does not require predefined tolerance bounds as input. Such tolerances are generally difficult to estimate, and in the same model, different local symmetries often have different tolerances, so fixing a unique tolerance is not appropriate. Later selection amongst the detected symmetries can be performed by analysing geometric constraint systems [Langbein et al. 2004; Gao et al. 2006].

Our method for symmetry detection using discrete points is quite different from symmetry approaches used in image processing, e.g. [Sun and Sherrah 1997], and mesh processing [Podolak et al. 2006; Mitra et al. 2006], which work with dense point data, where the point distributions are far more important than locations of individual points. Such work is mainly concerned with detecting *dominant* symmetries by partial matching of images or meshes under user selected tolerances. Instead, we generate *all* possible subset symmetries of a much smaller discrete point set. Each point is a characteristic feature point of a B-rep model whose boundary surfaces are represented analytically. Nevertheless, it might be possible to extend our method to detecting symmetries of meshes by extracting certain key feature points from meshes.

Section 2 considers related work on symmetry detection. Section 3 defines approximate incomplete symmetries. Section 4 gives an overview of our algorithm, and some details are discussed in Section 5. We demonstrate practical examples in Section 6.

2 Related work

Detection of exact global symmetries of shapes and point sets has been widely studied. Symmetries of points, lines and polyhedra in 3D can be found in $O(n\log n)$ time [Sugihara 1984]. The same time order also holds for more general 3D objects [Brass and Knauer 2004]. Relatively few results concern detection of *exact* symmetric subsets. Brass [2003] detects rotational symmetries by finding isosceles triangles and combining them into symmetric subsets efficiently using a tree. Tate and Jared [2003] find reflection symmetries present in a B-rep model by finding face loops and grouping them according to similarity.

Most previous work on approximate symmetry detection has considered *global* approximate symmetries, using various definitions. Iwanowski [1991] points out that testing approximate symmetry in the plane is NP-hard if approximate symmetry is defined in terms of the existence of an exactly symmetric object similar to the approximate object. Alternatively, approximate symmetries may be defined by checking whether an isometry exists which maps the point set approximately onto itself within some tolerance, which yields high-order polynomial time algorithms [Alt et al. 1988]. Mills et al [2001a] give a low-order polynomial time algorithm for approximate global symmetry detection that combines the combinatorial and geometric nature of symmetries. Based on this, Li et al [2006] show how to detect complete approximate symmetry cycles in a similar way to exact cycle detection given by Brass [2003], using a strict error-controlled point expansi1on idea. This paper further extends this method to detecting incomplete cycles of all symmetry types, and to merging cycles induced by the same isometry.

There is little further work on detecting approximate incomplete symmetries. We are only aware of [Robins et al. 1999], which is confined to subsets of points regularly arranged on a line. A completely different approach to detecting approximate symmetries is to define an asymmetry measure [Zabrodsky et al. 1995].

3 Approximate incomplete symmetry

In this section we first define complete approximate symmetry and then incomplete cycles. Merging such cycles leads to approximate incomplete symmetries.

Throughout this paper we use $\mathbb{E}^d$ to represent d-dimensional Euclidean space (here $d=2$ or 3) and $\|P-Q\|$ for the Euclidean distance between $P,Q\in\mathbb{E}^d$; $\mathscr{D}(\mathscr{S})=\{\|P-Q\| : P,Q\in\mathscr{S}\}$ is the distance set for a point set $\mathscr{S}$. We abuse the definition of $l \bmod N$ to mean l modulo N except when the answer is 0 whereupon we set it to N, as we use indices starting from 1. We define approximate equality of two real numbers a,b as $a=_\varepsilon b$, meaning $|a-b|\le\varepsilon$.

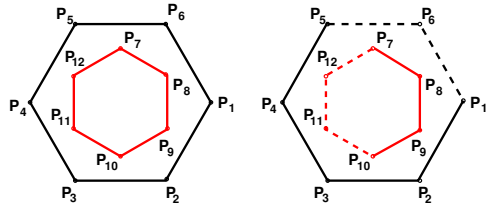
We use the following notation for elementary 3D symmetry groups: **M**: reflection; **I**: inversion; **T**: translation; $\mathbf{C_n}$: n-fold rotation; **Z**: glide, i.e. reflection at a line followed by translation parallel to the line, giving a 'zig-zag' symmetry in a plane; $\mathbf{S_n}$: rotation-reflection, i.e. reflection at a plane followed by rotation about an axis perpendicular to that plane, giving, e.g., an anti-prism; **W**: screw, i.e. rotation about an axis followed by translation along the axis. These comprise all elementary isometries in 3D [Weyl 1952].

We define an *approximate symmetry* of a point set in terms of a permutation of the points which maps distances between the points approximately onto each other [Mills et al. 2001a]. Let $\mathscr{S}\subset\mathbb{E}^d$ be a point set. We call a pair (ε,σ) with $\varepsilon\ge 0$ and a permutation σ on $\mathscr{S}$ an *approximate symmetry* of $\mathscr{S}$, if $=_\varepsilon$ is an equivalence relation on $\mathscr{D}(\mathscr{S})$, and $\|P-Q\|=_\varepsilon\|\sigma(P)-\sigma(Q)\|$ for all $P,Q\in\mathscr{S}$.

Above, for the points in $\mathscr{S}$ to be mapped *uniquely* onto each other by the permutation σ we require that $=_\varepsilon$ forms an equivalence relation, i.e. it groups the distances in $\mathscr{D}(\mathscr{S})$ into distinct sets of approximately equal distances (equivalence classes). For this to be satisfied, a tolerance interval $[E_{\min}(\mathscr{S}),E_{\max}(\mathscr{S}))$ exists for ε at which the symmetry is present. These are called the *minimal* and *maximal tolerances* of $\mathscr{S}$. $E_{\min}(\mathscr{S})$ ensures equality of all the distances in the same distance group and $E_{\max}(\mathscr{S})$ separates different groups. See Section 5.1 for details of how they are found. Clearly we need $E_{\min}(\mathscr{S})<E_{\max}(\mathscr{S})$ for $\mathscr{S}$ to be symmetric. A similar condition also has to be satisfied when we consider incomplete cycles and incomplete symmetries. This condition plays a key role in deciding whether a point set is symmetric or not at an automatically detected tolerance, as we will see in Sections 4 and 5.

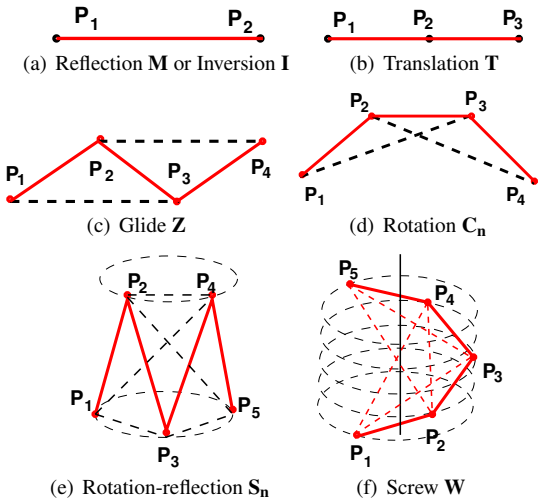
For example, the 2D points $\mathscr{S}=\{P_1,\ldots,P_{12}\}$ in Figure 1(a) have a six-fold rotational symmetry $\sigma:(1,\ldots,6),\ (7,\ldots,12)$ at some tolerance ε. It consists of two cycles $(1,\ldots,6)$ and $(7,\ldots,12)$. The distance sets $G_r=\{\|P_l-P_{(l+r) \bmod 6}\|,\ 1\le l\le 6\}$ for $r=1,2,3$ are distance equivalence classes of $=_\varepsilon$ determined by (ε,σ).

In general, some points in an otherwise symmetric point set may not be present. E.g. omitting points P_6,P_{11},P_{12} from Figure 1(a) gives the set $\mathscr{S}=\{P_1,\ldots,P_5,P_7,\ldots,P_{10}\}$ in Figure 1(b). However, completing such incomplete symmetries is not uniquely determined—adding different points may produce different symmetries. For example, we could complete $\mathscr{S}$ to have a twelve-fold rotational symmetry. To avoid such problems, we define an *incomplete symmetry* as the combination of *incomplete cycles* constructed from lists of *consecutive* points, e.g. points $\mathscr{C}_1=(P_1,\ldots,P_5)$ and $\mathscr{C}_2=$



(a) Complete symmetry (b) Incomplete symmetry

Figure 1: Complete and incomplete approximate symmetries



(a) Reflection **M** or Inversion **I** (b) Translation **T**

(c) Glide **Z** (d) Rotation $\mathbf{C_n}$

(e) Rotation-reflection $\mathbf{S_n}$ (f) Screw **W**

Figure 2: Incomplete cycle types

$(P_7,\ldots,P_{10})$ in Figure 1(b). Further requirements are prescribed below to uniquely define an incomplete cycle $\mathscr{C} \subset \mathscr{P}$, specifically (C1) to ensure membership in a potentially symmetric set, (C2) to ensure points in $\mathscr{C}$ are sufficiently distinct from other points in $\mathscr{P}$, and (C3) to ensure maximal cardinality under the previous two conditions.

Let $\mathscr{C} = \{P_l,\ldots,P_c\}$ be a sequence of $c \geq 2$ points from $\mathscr{P}$, $\varepsilon \geq 0$ and let σ be an injection mapping P_l to P_{l+1} for $l = 1,\ldots,c-1$. We say that $\mathscr{C}$ has a *(maximal approximate) incomplete cycle* (ε,σ) if

(C1) $=_\varepsilon$ is an equivalence relation on the distance set $\mathscr{D}(\mathscr{C})$, and $\|P-Q\| =_\varepsilon \|\sigma(P)-\sigma(Q)\|$ for all $P,Q \in \mathscr{C}$;

(C2) no point in $\mathscr{P} \setminus \mathscr{C}$ can replace a point in $\mathscr{C}$ such that (C1) is still true;

(C3) no single point in $\mathscr{P} \setminus \mathscr{C}$ can be added to $\mathscr{C}$ for any tolerance ε such that (C1) and (C2) are still true.

Complete cycles are special cases of incomplete cycles in the sense that σ further maps P_c onto P_1. We refer to both incomplete and complete cycles as cycles for simplicity. It is not difficult to see that any sequence of consecutive points forming a symmetry of any type satisfies the incomplete cycle conditions.

Incomplete symmetries come from merging cycles sharing a *single* isometry which approximately maps points within each cycle onto each other: see condition (I1) below. To sufficiently specify this isometry, a minimum number of points $T_{\min}$ is required according to symmetry type T: 2 for **M**, **I** and $\mathbf{C_2}$; 3 for **T** and for a regular triangle; 4 for $\mathbf{C_n}$ with $n \geq 4$, **Z** and for a regular tetrahedra; 5 for $\mathbf{S_n}$ and **W**. Various incomplete cycles of different symmetry types are shown in Figure 2. Furthermore, as in the incomplete

Algorithm: $\mathscr{I} \leftarrow$ APPROXINCOMPLETESYMMETRIES($\mathscr{P}$)
Input: Set of distinct points $\mathscr{P} \subset \mathbb{E}^d, d = 2,3$
Output: Incomplete symmetries $(\varepsilon_l,\mathscr{S}_l)$ of $\mathscr{P}$, $\mathscr{I} = \{(\varepsilon_l,\mathscr{S}_l) : l = 1,\ldots,N\}$

01 $\mathscr{C} \leftarrow$ INCOMPLETECYCLES ($\mathscr{P}$)
02 $\mathscr{M} \leftarrow$ empty *// mapping of symmetry types to cycle lists*
03 **for each** $C \in \mathscr{C}$
04 $\quad\mathscr{T} \leftarrow$ SYMMETRYTYPES (C)
05 $\quad$**for each** $T \in \mathscr{T}$, append $(\mathscr{M}[T], C)$
06 **end for**
07 $\mathscr{M}[\mathbf{M}] \leftarrow \mathscr{M}[\mathbf{C_2}] \leftarrow$ POINTPAIRS ($\mathscr{P}$)
08 $\mathscr{I} \leftarrow$ empty *// output symmetries*
09 **for each** cycle list L in $\mathscr{M}$ for a symmetry type T
10 $\quad\mathscr{D}(L) \leftarrow$ matrix of minimal tolerances for each consistent incomplete cycle pair in L for T
11 $\quad\mathscr{G} \leftarrow$ CLUSTERCYCLES (L, $\mathscr{D}(L)$)
12 $\quad$**for each** complete cluster $\mathscr{S}$ in $\mathscr{G}$
13 $\quad\quad$**if** $E_{\min}(\mathscr{S}) < E_{\max}(\mathscr{S})$, $\mathscr{I} \leftarrow \mathscr{I} \cup \{(E_{\min}(\mathscr{S}),\mathscr{S})\}$
14 $\quad$**end for**
15 **end for**
16 **return** $\mathscr{I}$

Figure 3: Incomplete symmetry detection algorithm

cycle definition, for an incomplete symmetry $\mathscr{S}$ we need a distance condition (I2) and an unambiguity condition (I3) to distinguish $\mathscr{S}$ from other cycles of the same type in $\mathscr{P}$.

Thus, a point subset $\mathscr{S} \subset \mathscr{P}$ has an *(approximate) incomplete symmetry* (ε,σ) of symmetry type T if

(I1) $\mathscr{S} = \bigcup_{1\leq l\leq n} \mathscr{C}_l$, where $\mathscr{C}_l \cap \mathscr{C}_k = \emptyset$ for all $l \neq k$, and each $\mathscr{C}_l$ is determined by a cycle (ε,σ_l) of type T with at least $T_{\min}$ points, and for each $\mathscr{C}_l$, σ restricted to C_l is σ_l;

(I2) $\|P-Q\| =_\varepsilon \|\sigma(P)-\sigma(Q)\|$ for all $P,Q \in \mathscr{S}$ and $=_\varepsilon$ is an equivalence relation on each distance set $\mathscr{D}(\mathscr{C}_l)$;

(I3) no cycle $\mathscr{C}^*$ of type T in $\mathscr{P} \setminus \mathscr{S}$ exists such that (I2) is true for $\mathscr{C}^* \cup \mathscr{C}_l$ at tolerance less than ε for any $1 \leq l \leq n$.

In particular, two cycles $\mathscr{C}_1, \mathscr{C}_2$ are said to be *consistent* if they satisfy (I1) and (I2): for example, the points in Figure 1(b) have an incomplete six-fold rotational symmetry formed by two consistent cycles $\mathscr{C}_1 = (P_1,\ldots,P_5)$ and $\mathscr{C}_2 = (P_7,\ldots,P_{10})$.

4 Algorithm overview

We now give an overview of our algorithm for detecting incomplete symmetries of a point set $\mathscr{P}$ based on the above concepts. See Figure 3. The algorithm first detects all cycles of $\mathscr{P}$, and then clusters these cycles to find incomplete symmetries. It takes as input a set of points $\mathscr{P} \in \mathbb{E}^d$; no two input points may have the same position. The algorithm outputs all incomplete symmetries as pairs $(\varepsilon_l,\mathscr{S}_l)$ where $\mathscr{S}_l$ is a set of cycles, and ε_l is the minimum tolerance of the incomplete symmetry.

The first step (Lines 01–07) builds on our previous work for detecting complete permutation cycles which induce a symmetry on a point subset [Li et al. 2006]. We extend our earlier algorithm to detect all incomplete cycles (Line 01). For each cycle, we detect its symmetry types and then add it to lists of cycles having that symmetry type (Lines 02–06). Due to their approximate nature, we have to consider more than one symmetry type per cycle. Furthermore, all point pairs in the input set trivially induce **M** and $\mathbf{C_2}$ symmetries, so we add all pairs to the corresponding lists (Line 07).

In the second step we cluster cycles of the same symmetry type so that all cycle permutations in a cluster are induced by the same isometry (Lines 08–15). We have to handle each symmetry type separately (Line 09); the same cycle may have to be considered for multiple symmetry types. Only merging cycles with the same symmetry type greatly reduces the number of possibilities to consider, and it ensures that only proper incomplete symmetries are detected.

To cluster cycles, a similarity measure between cycles of the same symmetry type is required. It is set as $E_{\min}(\mathscr{C}_1 \cup \mathscr{C}_2)$ (Line 10), as this gives the actual maximum matching error when mapping distances between the points in $\mathscr{C}_1 \cup \mathscr{C}_2$ onto each other as determined by the combined permutation involving $\mathscr{C}_1, \mathscr{C}_2$. Details are given in Section 5.2.

We then cluster the cycles by combining them pairwise in order of similarity (Line 11). This process involves constructing a graph whose nodes are the cycles and whose edges are introduced in order during clustering of consistent cycles. The graph's completely connected components are constructed using a similar clustering algorithm to that in [Mills et al. 2001a]. To decide if a component $\mathscr{S}$ has an incomplete symmetry, we must check that $E_{\min}(\mathscr{S}) < E_{\max}(\mathscr{S})$ (Line 13). Otherwise, condition (I2) may not be fulfilled and hence no symmetry can be found for $\mathscr{S}$ or sets containing it, so we omit $\mathscr{S}$ from further consideration. Condition (I3) is automatically satisfied for $\mathscr{S}$ from the clustering order. Specifically, if a cycle $\mathscr{C}^* \in \mathscr{P} \setminus \mathscr{S}$ exists such that $\mathscr{C}^* \cup \mathscr{C}_l$ for a cycle $\mathscr{C}_l$ of $\mathscr{S}$ has an incomplete symmetry at tolerance less than $E_{\min}(\mathscr{S})$, then $E_{\min}(\mathscr{C}^* \cup \mathscr{C}_l) < E_{\min}(\mathscr{S}) \leq E_{\min}(\mathscr{C}_l \cup \mathscr{C}_k)$ for any other cycle C_k in $\mathscr{S}$. This means that $\mathscr{C}^*, \mathscr{C}_l$ have been merged before pair $\mathscr{C}_l, \mathscr{C}_k$. Thus, $\mathscr{C}^*$ must be contained in $\mathscr{S}$. Hence, $\mathscr{S}$ has an incomplete symmetry at tolerance $E_{\min}(\mathscr{S})$. The symmetries detected do not only include the maximal complete graph components, but also its valid complete subsets, which generally have smaller symmetry tolerances.

5 Algorithm details

We now give further algorithmic details concerning cycle detection (Section 5.1), and cycle clustering (Section 5.2).

5.1 Cycle detection based on point expansion

First we discuss how to detect cycles in an input point set $\mathscr{P}$. Starting from an approximate isosceles triangle, we add points until a complete cycle has been found, or no further expansion point exists. To avoid accumulating errors during the expansion process, selection of expansion points is based on our cycle definition.

We first illustrate this idea in 2D for $\mathbf{C_n}$, $n \geq 3$. Let $\mathscr{C} = \{P_l,\ 1 \leq l \leq c\}$ be a point sequence for a cycle (ε, σ) of type $\mathbf{C_n}$. As $=_\varepsilon$ is an equivalence relation on the set $\mathscr{D}(\mathscr{C})$ of all distances between the points in $\mathscr{C}$, its distance equivalence classes are

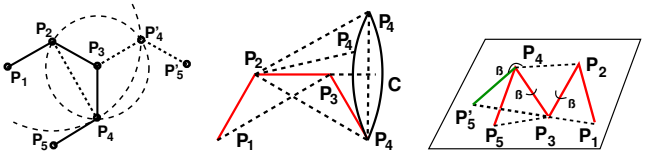
$$\mathscr{G}_r = \{\|P_l - P_{(l+r) \bmod n}\| : P_l, P_{(l+r) \bmod n} \in \mathscr{C}\} \text{ for } 1 \leq r \leq R,$$

where $R = \min(c, \lfloor n/2 \rfloor) - 1$. Correspondingly we have

$$E_{\min}(\mathscr{C}) = \max_{1 \leq r \leq R}(D_r - d_r), \quad E_{\max}(\mathscr{C}) = \min_{1 \leq r \leq R-1}(d_{r+1} - D_r)$$

with $d_r = \min(\mathscr{G}_r(\mathscr{C}))$, $D_r = \max(\mathscr{G}_r(\mathscr{C}))$. Therefore

$$E_{\min}(\mathscr{C}) < E_{\max}(\mathscr{C}), \tag{1}$$



(a) Expansion from three points in 2D (b) Expansion from three points in 3D (c) Expansion from four points in 3D

Figure 4: Cycle symmetry type is determined by the expansion point selection from a seed set

if $=_\varepsilon$ is to be an equivalence relation on $\mathscr{D}(\mathscr{C})$. This forms the core constraint that a cycle must fulfil for our point expansion algorithm. Efficient computation of $E_{\min}(\mathscr{C})$ and $E_{\max}(\mathscr{C})$ is discussed in [Li et al. 2006].

Thus, we select an initial seed set $\mathscr{C} = \{P_1, P_2, P_3\}$ satisfying this condition, generating a cycle. Moreover, it is required that for any possible expansion point P in $\mathscr{C}$, $\mathscr{C} \cup \{P\}$ also satisfies Eq. (1). However, more than one such point in $\mathscr{P}$ may exist. In order for the resulting cycle to be unambiguous (see (C2)), we choose the one among the candidate points minimising the corresponding $E_{\min}(\mathscr{C} \cup \{P\})$. This adaptive tolerance setting strategy avoids adding any points that may violate the unambiguity condition while also ensuring that we always find any expansion point which might exist. Note that a fixed tolerance, either chosen globally for all cycles, or locally based on the initial seed set, would not allow proper cycle generation as different cycles may have different tolerances, and tolerances must be based on all points in a cycle rather than any three consecutive points within it.

Expansion continues until either no further expansion point exists or a complete cycle is found. A cycle $\mathscr{C}$ with c points describes a complete cycle if $\|P_c - P_1\| =_\varepsilon \|P_l - P_{l+1}\|$ for $1 \leq l \leq c-1$. For each resulting cycle, complete or incomplete, its unambiguity condition is then verified to ensure that no single point in $\mathscr{C}$ can be replaced by a point in $\mathscr{P} \setminus \mathscr{C}$ based on the corresponding minimal tolerance. An efficient algorithm for doing so for complete cycles is given in [Li et al. 2006]; the incomplete case can be handled similarly.

We discuss how the point expansion process given for $\mathbf{C_n}$ can also be used for symmetries of other types, by analysing possible locations of expansion points in the exact case. First consider the 2D case. Expansion from the seed set of three points is done slightly different from later expansions. Satisfaction of Eq. (1) for $\mathscr{C} \cup \{P\}$ requires the fourth point P_4 to satisfy $\|P_4 - P_3\| = \|P_2 - P_3\| = \|P_1 - P_2\|, \|P_4 - P_2\| = \|P_1 - P_3\|$: see Figure 4(a). However, two points, P_4 and P_4' in Figure 4(a), may satisfy this equation. Further expanding P_1, P_2, P_3, P_4 produces an isometry of type $\mathbf{C_n}$ while expanding P_1, P_2, P_3, P_4' gives $\mathbf{Z}$. We have to consider both possibilities. Note also the special case $\mathbf{T}$ arises if P_1, P_2, P_3 are collinear.

Turning now to 3D, and considering $\varepsilon = 0$, the fourth point P_4 can lie anywhere on a circle C (rather than in two locations): see Figure 4(b). Different locations of P_4 induce cycles of different types. If P_1, P_2, P_3, and P_4 are all coplanar, all other expansion points in this cycle must be coplanar. In this case, a cycle of type $\mathbf{C_n}$ or $\mathbf{Z}$ or $\mathbf{T}$ can be found. Other locations of P_4 on the circle C give a regular tetrahedron, when all distances involved are equal, or $\mathbf{S_n}$ or $\mathbf{W}$, depending on the next expansion point P_5. As illustrated in Figure 4(c), P_5 can lie on either side of the plane defined by P_2, P_3, P_4. If P_5 lies on the same side as P_1, the corresponding cycle is $\mathbf{S_n}$; otherwise it is $\mathbf{W}$. In the approximate case, all fourth potential expansion points and the two locations for the fifth points must be considered similarly.

Having detected each cycle, we still need to decide its symmetry types: one reason is that cycles of the same type are merged to generate incomplete symmetries. The type is uniquely determined for a complete cycle for $\mathbf{C_n}$ and $\mathbf{S_n}$ from the number of points. Other cycles may have more than one symmetry type in the approximate case. It is important not to rule out any possible symmetry type too soon, and lose information. The cycle merging process later helps to rule out inappropriate symmetries. Therefore, instead of trying to find the optimal isometry determined by the point mapping [Eggert et al. 1997], we determine symmetry types by considering the relative locations of the centres of the circles formed by each consecutive triple of points in $\mathscr{C}$. For example, in 2D, for $\mathbf{C_n}$ these centres lie on the same side of all edges joining consecutive points, whereas they lie on opposite sides for successive triples for $\mathbf{Z}$.

5.2 Finding incomplete symmetries using cycle clustering

To find incomplete symmetries, cycles of the same symmetry type are clustered. In this section, we discuss the computation of the minimal and maximal tolerances for a set of cycles.

First let $\mathscr{C}_1 = \{P_l : 1 \le l \le c_1\}, \mathscr{C}_2 = \{Q_l : 1 \le l \le c_2\}$ describe two cycles with symmetries $(E_{\min}(\mathscr{C}_1), \sigma_1)$ and $(E_{\min}(\mathscr{C}_2), \sigma_2)$ of the same type and extremal tolerances $E_{\min}(\mathscr{C}_1)$, $E_{\min}(\mathscr{C}_2)$, $E_{\max}(\mathscr{C}_1)$ and $E_{\max}(\mathscr{C}_2)$. Let $\mathscr{C} = \mathscr{C}_1 \cup \mathscr{C}_2$. To take a concrete case, we assume that the symmetry type is $\mathbf{C_n}$ in 2D and both cycles have the same ordering. Other symmetry types can be handled similarly.

Cycles $\mathscr{C}_1, \mathscr{C}_2$ are consistent if $E_{\min}(\mathscr{C}) < E_{\max}(\mathscr{C})$. Condition (I2) imposes conditions on values of $E_{\min}(\mathscr{C}), E_{\max}(\mathscr{C})$. By definition, $E_{\min}(\mathscr{C})$ is the minimal tolerance such that $\|P-Q\| =_\varepsilon \|\sigma(P) - \sigma(Q)\|$ for all P, Q in $\mathscr{C}$. This is already satisfied for $P, Q \in \mathscr{C}_1$ at $E_{\min}(\mathscr{C}_1)$ and $P, Q \in \mathscr{C}_2$ at $E_{\min}(\mathscr{C}_2)$. Verifying it for points $P \in \mathscr{C}_1$ and $Q \in \mathscr{C}_2$ can be achieved by finding the maximal difference e^* between suitable matching distances given by the combined permutation for $\mathscr{C}_1$ and $\mathscr{C}_2$, using a similar method to that described for a cycle in Section 5.1, by first setting up equivalent distance groups determined by the cycle permutations. Consequently, we have $E_{\min}(\mathscr{C}) = \max(e^*, E_{\min}(\mathscr{C}_1), E_{\min}(\mathscr{C}_2))$. For the maximal tolerance, as $\mathscr{C}_1$ and $\mathscr{C}_2$ are cycles, $=_\varepsilon$ remains an equivalence relation on $\mathscr{D}(\mathscr{C}_1)$ and $\mathscr{D}(\mathscr{C}_2)$ simultaneously for tolerances smaller than $E_{\max}(\mathscr{C}_1 \cup \mathscr{C}_2) = \min(E_{\max}(\mathscr{C}_1), E_{\max}(\mathscr{C}_2))$.

Computation of the minimal and maximal tolerances for a point set containing more than two cycles can then be derived. Let $\mathscr{C} = \cup_{1 \le l \le n} \mathscr{C}_l$, $n \ge 3$. Then $E_{\min}(\mathscr{C}) = \max_{1 \le l \ne k \le n}(E_{\min}(\mathscr{C}_l \cup \mathscr{C}_k))$, $E_{\max}(\mathscr{C}) = \min_{1 \le l \le n}(E_{\max}(\mathscr{C}_l))$.

6 Experiments and conclusions

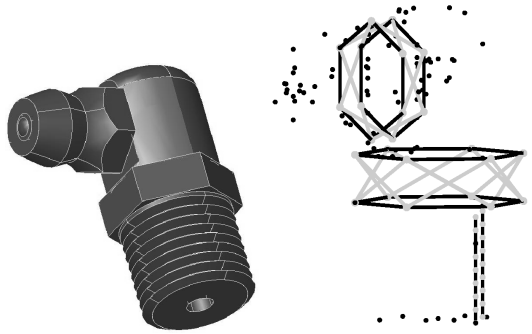
This section demonstrates results obtained by our algorithm from 3D points derived from approximate CAD models, showing its utility for design intent detection. The algorithm was implemented in Matlab using a 3.4GHz Pentium 4 computer with 1GB RAM.

For simplicity, we used the vertices instead of full representative point sets. We do not show mirror symmetries as they can easily be seen in the models and are very simple for our algorithm to detect. While our algorithm does not require any user-defined tolerances, for efficiency, we do not consider tolerances greater than five percent of the longest distance between points in the input point set. The results are summarised in Table 1. $\|\mathscr{P}\|$ is the number of points, S_n the number of seed sets for cycle detection, C_n the number of cycles detected, taking time T_C, and $\|\mathscr{I}\|$ the number of incomplete symmetries detected, taking time $T_{\mathscr{I}}$. The MISUFA model in Figure 5 came from Cadalog, Inc., `http://www.ohyeahcad.com/`, and the other two from the National Design Repository, `http://www.designrepository.org/`.

Model	$\|\mathscr{P}\|$	S_n	$\mathscr{C}_n$	T_C	$\|\mathscr{I}\|$	$T_{\mathscr{I}}$
Figure 5	135	4915	204	698s	12	14s
Figure 6(a)	76	3200	62	251s	12	13s
Figure 6(b)	438	6220	1045	476s	63	24s

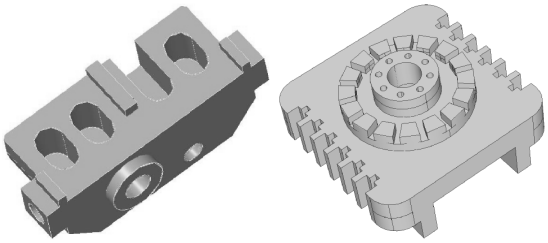
cycle detection for Figure 6(b) done by decomposing it into eight parts

Table 1: Summary of results for the example models



(a) Model (b) Symmetries

Figure 5: Symmetries of the MISUFA model



(a) Spinner Model (b) Monster model

Figure 6: Monster and Spinner models

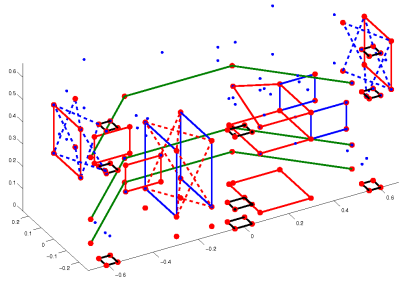
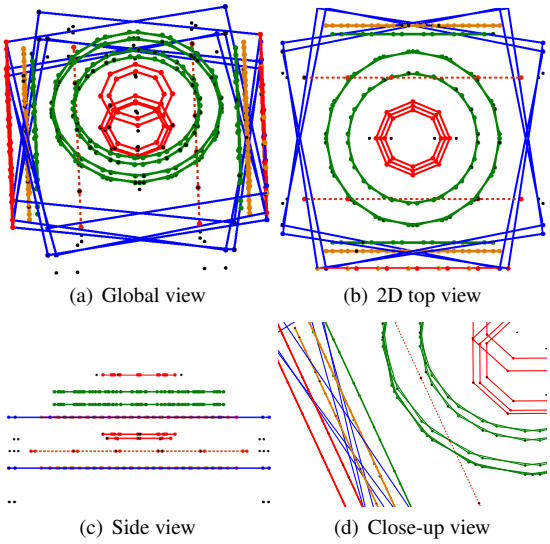


Figure 7: Symmetries of the Spinner model in Figure 6(a)

Five of the symmetries detected in the MISUFA model are shown in Figure 5(b). The model has no global symmetries. The local incomplete symmetries detected by our algorithm include two $\mathbf{C_6}$ symmetries (dark lines), two $\mathbf{S_6}$ (light lines) and one $\mathbf{T}$ (dark lines); each consists of two cycles. The rotational symmetries are obvious parts of the model; the (incomplete) translational symmetry demonstrates evenly spaced circular arcs on the thread. Two rotation-reflection symmetries are also present.

Various local incomplete symmetries are detected for the Spinner model in Figure 6(a). Note the the incomplete symmetry arising from merging three incomplete cycles (drawn in green, running from left to right) each consisting of four points. Other detected



(a) Global view (b) 2D top view

(c) Side view (d) Close-up view

Figure 8: Symmetries of the Monster model in Figure 6(b)

symmetries reveal the structure of the three slots (drawn in black, each consisting of three four-fold cycles), and various holes (drawn in red and blue, generally each consisting of two four-fold cycles).

The final test concerned the Monster model in Figure 6(b). To improve the speed of cycle detection, we first divided the extracted points into eight regions using horizontal planes. (This is straightforward for a user to do and is a reasonable approach as planar symmetries dominate in engineering. Furthermore, a C++ implementation would be likely to be much quicker than our Matlab prototype). Note that symmetry clustering still considered *all* cycles at the same time. Several of the detected symmetries are shown in Figures 8(a), (b), (c) and (d). Cycles having the same symmetry are drawn in the same colour. These include a $\mathbf{C_8}$ symmetry of 5 cycles (from the small cylindrical holes), a $\mathbf{C_{16}}$ symmetry of 5 cycles (from slots in the big cylinder), a $\mathbf{C_4}$ symmetry of 8 cycles (from the four blends in the corners) and $\mathbf{T}$ symmetries (from the side slots).

Our experiments on these and other models show that almost all expected symmetries are detected by our algorithm. Some spurious symmetries are also detected, which is unavoidable for approximate models. The time taken depends on the number of initial seed sets found. Incomplete cycles can be determined in $O(ps)$ time for a set with p points and s initial seed sets. The cycle clustering algorithm takes $O(c^2 \log c)$ time for c cycles. Setting a coarse user-predefined maximal tolerance greatly reduces the initial number of seed sets. Note also precomputing all point distances and storing them also improves efficiency. In future we intend to consider further methods to distinguish between intended and unintended symmetries and other regularities. We are also considering model decomposition to reduce the computational requirements.

References

ALT, H., MEHLHORN, K., WAGENER, H., AND WELZL, E. 1988. Congruence, similarity and symmetries of geometric objects. *Discrete Computational Geometry 3*, 237–256.

BARRATT, K. 1989. *Logic and Design in Art, Science and Mathematics*. Herbet Press, London.

BRASS, P., AND KNAUER, C. 2004. Testing congruence and symmetry for general 3-dimensional objects. *Computational Geometry 27*, 3–11.

BRASS, P. 2003. On finding maximum-cardinality symmetric subsets. *Computational Geometry 24*, 19–25.

EGGERT, D., LORUSSO, A., AND FISHER, R. 1997. Estimating 3-D rigid body transformations: a comparison of four major algorithm. *Machine Vision and App. 9*, 5-6, 272–290.

GAO, C. H., LANGBEIN, F. C., MARSHALL, A. D., AND MARTIN, R. R. 2003. Approximate congruence detection of model features for reverse engineering. In *Proc. Int. Conf. Shape Modelling and Applications*, 69–77.

GAO, X., LIN, Q., AND ZHANG, G. 2006. A C-tree decomposition algorithm for 2D and 3D geometric constraint solving. *Computer-Aided Design 38*, 1, 1–13.

IWANOWSKI, S. 1991. Testing approximate symmetry in the plane is NP-hard. *Theoretical Computer Science 80*, 227–262.

LANGBEIN, F., MARSHALL, A., AND MARTIN, R. 2004. Choosing consistent constraints for beautification of reverse engineered geometric models. *Computer-Aided Design 36*, 3, 261–278.

LI, M., LANGBEIN, F., AND MARTIN, R. 2006. Detecting approximate symmetries of discrete point subsets. submitted.

MILLS, B., LANGBEIN, F., MARSHALL, A., AND MARTIN, R. 2001. Approximate symmetry detection for reverse engineering. In *Proc. 6th ACM Symp. Solid and Physical Modeling*, 241–248.

MILLS, B., LANGBEIN, F., MARSHALL, A., AND MARTIN, R. 2001. Estimate of frequencies of geometric regularities for use in reverse engineering of simple mechanical components. Tech. Rep. GVG 2001–1, Cardiff University. `http://ralph.cs.cf.ac.uk/papers/Geometry/survey.pdf`.

MITRA, N., GUIBAS, L., AND PAULY, M. 2006. Partial and approximate symmetry detection for 3D geometry. *Proc. SIGGRAPH 2006, ACM Trans. Graph. 25*, 3, 560–568.

PODOLAK, J., SHILANE, P., GOLOVINSKIY, A., RUSINKIEWICZ, S., AND FUNKHOUSER, T. 2006. A planar-reflective symmetry transform for 3D shapes. *Proc. SIGGRAPH 2006, ACM Trans. Graph. 25*, 3, 549–559.

ROBINS, G., ROBINSON, B., AND SETHI, B. 1999. On detecting spatial regularity in noisy images. *Information Processing Letters 69*, 189–195.

SUGIHARA, K. 1984. An $n \log n$ algorithm for determining the congruity of polyhedra. *Journal of Computer and System Sciences 29*, 11, 36–47.

SUN, C., AND SHERRAH, J. 1997. 3D symmetry detection using the extended Gaussian image. *IEEE Trans. Pattern Analysis and Machine Intelligence 19*, 2, 164–168.

TATE, S., AND JARED, G. 2003. Recognising symmetry in solid models. *Computer-Aided Design 35*, 7, 673–692.

VARADY, T., MARTIN, R., AND COX, J. 1997. Reverse engineering of geometric models - an introduction. *Computer-Aided Design 29*, 4, 255–268.

WEYL, H. 1952. *Symmetry*. Princeton University Press.

ZABRODSKY, H., PELOG, S., AND AVNIR, D. 1995. Symmetry as a continuous feature. *IEEE Trans. Pattern Analysis and Machine Intelligence 17*, 12, 1154–1166.