

Representation and System-Agnostic Automated CAD Interoperability Testing based on Model Interchangeability

Duygu Sap¹ and Daniel P. Szabo²

¹CAMaCS, University of Warwick, Coventry, CV4 7AL, Warwickshire, United Kingdom

²Department of Operations Research, Eötvös Loránd University, Budapest, Hungary.

*Corresponding author: duygu.sap@warwick.ac.uk

Abstract

In this paper, we present two theoretically supported algorithmic frameworks to test the interoperability of distinct CAD systems by determining the interchangeability of their models with respect to a specified shape comparison criterion. We also introduce a command-line interface that implements these tests.

Both algorithmic frameworks provide indirect and representation-independent comparisons of CAD models through their abstract proxies and the interpretation of CAD models via queries. They allow local model comparisons via differential properties and provide a wide set of global model comparisons. The first algorithmic framework follows a semi-automated approach and requires the user to provide template files containing the proxy information. The second algorithmic framework follows a fully-automated approach and generates the proxies either directly from the CAD models or via the automatically-generated template files. Template files enable us to execute the interoperability testing in a system-independent manner by decoupling the executable component of our testing from the CAD systems. The semi-automated framework may be utilized in determining the interchangeable usability of CAD models in a set of applications for which the relevant template file provides sufficient information or when we do not have access to the CAD systems. On the other hand, the fully-automated framework provides a more practical interoperability testing based on CAD model interchangeability with wider applicability.

Our frameworks and testing tools enable the investigation of the interoperability of generic CAD software that may be developed for different operating systems or employ different scripting languages. We allow the user to state a tolerance level for the testing and provide accuracies for our abstract model-based testing results.

Keywords: geometric interoperability, shape comparison, model interchangeability, shape proxies, query-based analysis.

1. Introduction

The interoperability of a CAD system C_1 with another CAD system C_2 is its ability to interpret correctly and work with C_2 's models and/or algorithms (Hoffmann et al., 2011, 2014; Sap & Shapiro, 2019). This kind of interoperability is also known as geometric interoperability. To evaluate geometric interoperability, we focus on testing model equivalence independently of proprietary representations, algorithms, or APIs used by different CAD systems. Thus, model interchangeability is the core challenge in testing interoperability in our context.

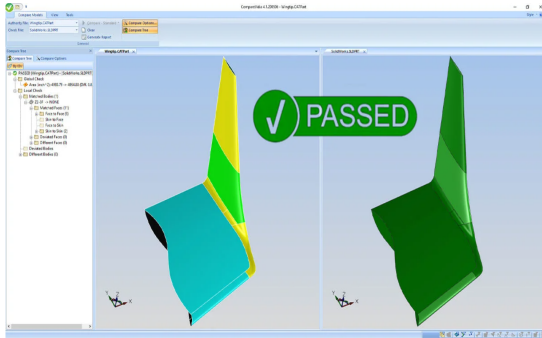
CAD systems provide a wide range of geometric representations and file formats, and may employ different algorithms and APIs (*OpenCASCADE*, 2020; *OpenSCAD*, 2019; *Parasolid*, 2019; *Rhinoceros 5*, 2012). The inherent differences between systems, such as the differences in semantics, assumptions, constraints, representation accuracy, and algorithmic precision, limit the interoperability of CAD systems (Sap & Shapiro, 2019). CAD models may also contain errors or unnoticeable anomalies that can impede interoperability and/or their reusability (Gonzalez-Lluch et al., 2017).

1.1 Motivation

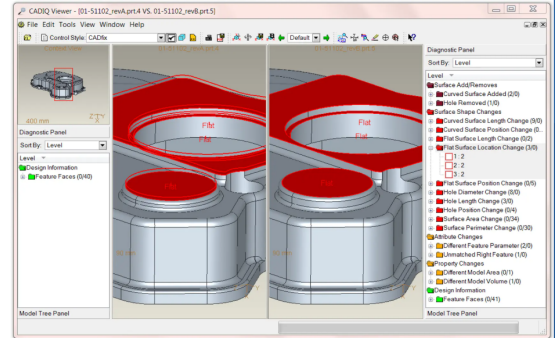
CAD system interoperability has a wide range of applicability since CAD models may serve as domains for many scientific and engineering problems such as numerically-controlled manufacturing, finite element analysis, and coordinate-measuring machining (*Technical Data Package 31000A*, 2013). It is important to ensure that transferring a CAD model from one system to another would not result in an unpredictable CAD data loss since low quality CAD data often negatively influence the applications that use the CAD data (Danjou & Koehler, 2007; Horwood & Kulkarni, 2005; McKenney, 1998).

The interoperability of CAD systems is supported by semi-automated heuristic tools, requiring expertise and significant manual labor (Wang et al., 2011). There are many software companies that offer interoperability, translation verification, and validation solutions. (*CADdoctor: Healing, Optimization and Simplification*, 2019; *CADfix*, 2019; *CADIQ*, 2025; CompareVidia, 2025; Simon, 2019) are some of the software tools provided by these companies. These software tools run checks on the models transferred and then utilize some automated or manual healing to compensate for the data and model quality loss. Thus, the models are modified to establish interoperability without any quantifiable measure. Models may also be simplified for use in certain applications such as manufacturing or analysis. For example, (Mounir et al., 2012) introduce a CAD geometry simplification technique that combines detail elimination and face merging. Their method can be used in finite element method (FEM) simulations. Figure 1 gives a glimpse into the methodologies employed by some of the existing interoperability testing tools. It illustrates that these tools perform model comparison and validation with redundant testing, a lack of a generalizable and systematic approach, and a reliance on heuristics.

Department of Defense (DoD) requires approved validation processes to verify that 3D models are applicable as reference data (*Technical Data Package 31000A*, 2013). A thorough analysis of some of these CAD model validation processes is provided in (Sap, 2019). Since 1998, the CAx Implementor Forum has been carrying out geometry validation testing based on a standard format, namely, STEP (Boy et al., 2012). Their testing procedure is based on the comparison of the results they derive from experimenting over various sets of models that do not seem to form bases for any geometric model space (Sap & Shapiro, 2019).



Digital Product Definition (DPD) Validation (CompareVidia, 2025)



Comprehensive mode analysis (CADIQ, 2025)

Figure 1: Some existing CAD interoperability and model validation procedures

Many commercial tools compare CAD models or validate translated models with respect to their basic shape properties such as volume and surface area (*CADdoctor: Healing, Optimization and Simplification*, 2019; *CADfix*, 2019; *CADIQ*, 2025). These tools focus on providing practical and short-term solutions for a narrow set of problems and/or offer unpredictably accurate and system-specific interoperability solutions.

Standard CAD model formats such as STEP (*ISO 10303, Industrial Automation Systems and integration*, 1998) and IGES (Smith et al., 1983) neither provide any measures over the data loss nor offer any guarantees on the shape validity after format translations or model transfers. Exchanging models in a standard format such as STEP does not guarantee the preservation of the shape properties (Sap & Szabo, 2020) (See Figure 2 and Tables 1 and 2). Also, despite the existence of such standard formats, distinct CAD systems still have proprietary representations.

Table 1: Some Shape Property Computations

| Model | Volume | Surface Area | Centroid |
|-------|---------|--------------|--------------------------|
| .3dm | 4943.22 | 5804.82 | (2e-06, -4.3e-06, -2.78) |
| STL | 4924.15 | 5782.56 | (-5.9e-05, -0.01, -2.74) |
| STEP | 4943.14 | 5804.77 | (1e-06, 2e-06, -2.78) |

(Sap & Shapiro, 2019) introduced a theoretical framework for geometric interoperability based on invariant shape properties using abstract mathematical models as proxies for

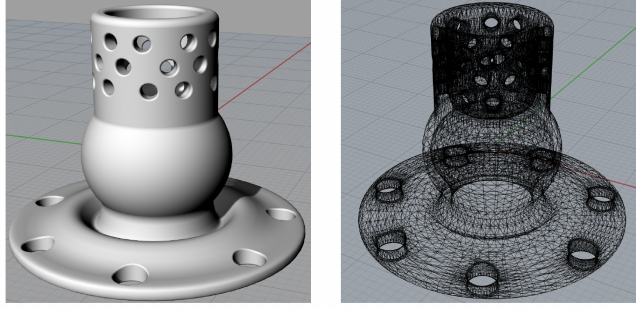


Figure 2: Left: Original CAD model in Rhino .3dm format, Right: Translated version in a standard CAD format, STL

Table 2: Computation Precisions

| Model | Volume | Surface Area | Centroid |
|-------|--------|--------------|---------------------------|
| .3dm | 0.002 | 0.0011 | (3.9e-06,3.7e-06,8.1e-06) |
| STL | 1e-06 | 1e-06 | (5.3e-14,1e-13,1e-09) |
| STEP | 0.0032 | 0.0011 | (7.6e-06,7.4e-06,1.2e-05) |

CAD models. However, (Sap & Shapiro, 2019) mainly focuses on the derivation of fundamental theoretical results and global shape comparisons. With this work, we improve and extend the framework presented in (Sap & Shapiro, 2019) by introducing local shape comparisons and local shape proxies, and expanding the set of global shape comparisons and global shape proxies. Moreover, we place greater emphasis on quantitative testing and the application of theory by introducing algorithmic frameworks and automated testing tools.

1.2 Overview

In this paper, we test the interoperability of two generic CAD systems by determining the interchangeability of their models that represent the same 3D shape in the physical world. Our theoretical results extend the proxy model and query-based approach described in (Sap & Shapiro, 2019) via additional proxy models and multi-level shape comparisons. More precisely, we present local shape comparisons and local shape proxies, while also extending the set of global shape comparisons and global shape proxies presented in (Sap & Shapiro, 2019). Thus, our theoretical results provide a more comprehensive CAD model comparison than the approach presented in (Sap & Shapiro, 2019). In addition, we mostly focus on the application of the theory and obtaining quantitative results. For example, we compute and compare the number of shared topological features rather than checking if two models are homotopic as done in (Sap & Shapiro, 2019).

The proxy model approach restricts the domain of comparison necessary for verifying the interoperability of CAD systems based on model comparisons. It also enables us to visualize and analyze the underlying 3D shape represented by a CAD model from different mathematical perspectives independently from their CAD representations, and therefore,

the CAD systems. The query-based approach allows avoiding global representation conversions and format translations that are sensitive to error accumulations (Hoffmann et al., 2014). In query-based interoperability scenarios, systems can employ different representations and algorithms, and they retain their copies of the models separately (Hoffmann et al., 2014).

We introduce theoretically supported algorithmic frameworks for systematically evaluating the interoperability of CAD systems, and provide a command-line interface (CLI) for conducting these evaluations in either a semi-automated or a fully automated manner. Unlike the commercial tools mentioned in Section 1.1, we explicitly include the accuracy and tolerance levels involved in testing to enable tracking of potential data loss during model exchange or translation.

We construct the proxies of the CAD models using the data obtained from the CAD systems through queries. The type of proxies are determined by the type of comparisons we need to do. For example, if we need to do a topological comparison, we utilise topological proxies such as covers or algebraic complexes. The proxies serve as abstract substitutes for the CAD models in our testing. We extract the necessary information for constructing these proxies and, if needed, store these information in template files via our CLI. The proxy model approach reflects the representation-agnostic aspect of our interoperability testing scheme.

Our testing procedure supports two distinct levels of automation. The semi-automated testing requires template files as inputs. Users can generate these template files manually or via our additional command-line interface. The automated testing uses CAD models as inputs and directly generates proxies from these models. Thus, the automated testing is executed in fewer steps with less user intervention. On the other hand, the semi-automated system may be preferable in the case where one is interested in the interchangeable usability of CAD models in specific computational design and engineering applications since template files allow for selective data storage. It can also be used when we do not have access to the CAD systems and are provided the CAD information in a template file by the user. Template files emphasize on the system-agnostic aspect of our CAD interoperability testing as it liberates the models from proprietary file formats.

Our computations involve point-membership queries (PMQ), distance queries, and differential queries (where available). We evaluate various geometric, topological, algebraic, integral, and differential properties of the models using proxy models such as point clouds, unions of balls, algebraic complexes, and shape operators. We let the user set a tolerance value for the comparison test they run, and provide accuracies for our testing results.

Remark 1. *In practice, the type of shape comparison would be determined by the application of interest since different applications may utilize different underlying structures of a shape. With the proper choice of a shape proxy, we can verify the equivalence of CAD models by comparing the underlying structures represented by these models. One can use such an equivalence to determine a set of applications in which the CAD models*

can be used interchangeably, thus, define the boundaries for the application domain of
CAD interoperability.

Our proxy model and shape property computations depend on a parameter, ϵ , which
is primarily determined by the accuracy of PMQ tolerances, algorithm precisions within
the involved systems, and the minimum feature sizes of the CAD models, as discussed in
(Sap & Shapiro, 2019). Here, the feature size refers to the distance between a point on
the manifold and its medial axis.

We emphasize that our interoperability testing extends beyond model comparison, ad-
dressing systems built for distinct operating systems and employing different algorithms
and APIs. Through our automated testing framework, we facilitate interoperability across
these heterogeneous CAD environments, enabling broader integration and practical ap-
plication.

2. Methodology

The shape comparisons described in this section primarily focus on the topology and
local geometry of the 3D shapes represented by the given CAD models. The topological
and geometrical vectorization of CAD models allows us to eliminate the dependencies
on different geometric representations used to encode and visualize the underlying 3D
shapes from the model comparison. Topology and geometry comparisons complement
each other in CAD model analysis. For example, two 3D shapes that are homotopic,
like a coffee mug and a donut, can be vastly different. Therefore, it is insufficient to
assume equivalence based solely on topology when comparing CAD models. We must also
consider their geometry, which provides critical information about their spatial properties
at the global scale, as well as their local deformations and curvature. Since the global
geometry comparison is already discussed in (Sap & Shapiro, 2019), we describe the
methodology for local geometry comparisons in Section 2.1. Then, in Section 2.2, we
describe the methodology for topological comparisons that yield quantitative comparisons
of the homological groups.

2.1 Local Shape Comparison via Differential Properties

Differential properties can be derived easily if the CAD software is capable of providing
differential information through queries. Differential information can also be approxim-
ated via the lower level queries (Hoffmann et al., 2014; Sap & Shapiro, 2019).

We assume that the compared surface parts of the CAD models are manifolds, meaning
that every point on each surface part has a neighborhood homeomorphic to an open
Euclidean disc, and are locally C^2 . Thus, we can use tangent planes, shape operators,
and normal vector fields as proxies to check the local geometric similarity, compare the
Gaussian curvature, mean curvature, and the second fundamental forms of the surfaces.

Lemma 1 shows how we can determine the local geometric similarity of CAD models in the neighborhood of a mutual surface point by using tangent planes as proxies.

Lemma 1. Suppose M_1 and M_2 are 3D models representing a fixed 3D shape in two distinct CAD systems, and let M_i^t be the tangent plane of the surface of M_i at a generic point p labelled as 'on' via the PMQ for $i = 1, 2$. Let $\mathcal{N}_p^\epsilon = B_\epsilon(p)$ define the disc consisting of the points that are geodesically ϵ -distant from p , where $\epsilon \leq \delta$ for δ denoting the minimum feature size of the shape. Then, the models M_1 and M_2 are locally congruent with an accuracy of 4ϵ about the point p , i.e.

$$d_H(M_1(\mathcal{N}_p^\epsilon), M_2(\mathcal{N}_p^\epsilon)) \leq 4\epsilon, \quad (1)$$

where $d_H(\cdot, \cdot)$ denotes the Hausdorff metric.

Proof. Let

$$\begin{aligned} M_i^t(\mathcal{N}_p^\epsilon) &= \{p_j^i \in M_i^t : d(p_j^i, p) = \epsilon \\ &\text{and } \angle(\overrightarrow{pp_j^i}, \overrightarrow{pp_{j+1}^i}) = \frac{2\pi}{n}, j = 1, 2, \dots, n\}, \end{aligned} \quad (2)$$

where $d(\cdot, \cdot)$ denotes the usual Euclidean metric and $\angle(\overrightarrow{pp_j^i}, \overrightarrow{pp_{j+1}^i})$ denotes the angle between the vectors $\overrightarrow{pp_j^i}$ and $\overrightarrow{pp_{j+1}^i}$. Since $\epsilon \leq \delta$ by definition, we have

$$d_H(M_i(\mathcal{N}_p^\epsilon), M_i^t(\mathcal{N}_p^\epsilon)) \leq \epsilon.$$

Then,

$$\begin{aligned} d_H(M_1(\mathcal{N}_p^\epsilon), M_2(\mathcal{N}_p^\epsilon)) &\leq d_H(M_1(\mathcal{N}_p^\epsilon), M_1^t(\mathcal{N}_p^\epsilon)) \\ &\quad + d_H(M_1^t(\mathcal{N}_p^\epsilon), M_2^t(\mathcal{N}_p^\epsilon)) \\ &\quad + d_H(M_2^t(\mathcal{N}_p^\epsilon), M_2(\mathcal{N}_p^\epsilon)) \leq 4\epsilon \end{aligned} \quad (3)$$

since $d(p_j^1, p_m^2) \leq d(p_j^1, p) + d(p, p_m^2) \leq 2\epsilon$ for any $j, m = 1, 2, \dots, n$.

□

Another approach for comparing shapes locally based on their differential properties is through the use of shape operators as proxies. Shape operator is a way of viewing the second fundamental form, which is a local and an extrinsic property of the surface that describes how a surface bends in R^3 (Carmo, 2016). The matrix representation of the shape operator, S_p , with respect to the first and second fundamental forms of the surface is given by the following (O'Neill, 2006):

$$S_p = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} = \begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1} \begin{pmatrix} L & M \\ M & N \end{pmatrix} \quad (4)$$

where E , F and G denote the coefficients of the first fundamental form, and L, M, N denote the coefficients of the second fundamental form (Crane et al., 2013). The first fundamental form captures the intrinsic geometry of a surface, that is, it can also be used to calculate distances and angles over the surface. We can derive the coefficients of the fundamental forms, thus, the shape operators by using the differential queries in the following:

$$\begin{aligned} E &= r_u \cdot r_u, & F &= r_u \cdot r_v, & G &= r_v \cdot r_v, \\ L &= r_{uu} \cdot n, & M &= r_{uv} \cdot n, & N &= r_{vv} \cdot n, \end{aligned} \quad (5)$$

where $r = r(u, v)$ denotes the surface parametrization and subscripts indicate differentiation with respect to the parameters u and v . For example, r_u denotes the derivative of r with respect to u , and r_{uv} denotes the derivative of r_u with respect to v . Thus, if the CAD systems provide the first and second-order derivative information and the normal vectors, then we can derive the shape operators $S_p : T_p M_i \rightarrow T_p M_i$, where $T_p M_i$ denotes the tangent plane to the surface of M_i at a generic point $p \in \partial M_i$. Using S_p , we can derive the principal curvatures, mean and Gaussian curvatures at p .

Lemma 2. *Let S_p^i be the shape operator associated with the model M_i at a point p with an 'on' label. Suppose that there exists a subspace $T_p M$ of $T_p M_i$ such that $T_p M = \{v \in R^3 : v \in T_p M_1 \cap T_p M_2\}$. Then, over $T_p M$*

$$\|S_p^1 - S_p^2\|_o \leq |\kappa_{max}^1| + |\kappa_{max}^2|, \quad (6)$$

where $\|\cdot\|_o$ denotes the operator-norm, and κ_{max}^i denotes the maximum principal curvature of the model M_i at point p , respectively.

Proof. It follows from the fact that the principal curvatures of M_i at p are the eigenvalues of S_p (Hatcher, 2001). Let the restrictions of S_p^i to $T_p M$ be denoted by \tilde{S}_p^i , then we have

$$\begin{aligned} \|\tilde{S}_p^1 - \tilde{S}_p^2\|_o &= \sup\{\|(\tilde{S}_p^1 - \tilde{S}_p^2)v\| : v \in T_p M, \|v\| = 1\} \\ &\leq \sup\{\|\tilde{S}_p^1 v\| : v \in T_p M, \|v\| = 1\} \\ &\quad + \sup\{\|\tilde{S}_p^2 v\| : v \in T_p M, \|v\| = 1\} \\ &\leq |\kappa_{max}^1| + |\kappa_{max}^2| \end{aligned}$$

□

Lemma 2 implies that at a mutual surface point p , the amount of bending of the surfaces in a tangential direction valid for both shapes differs at most by the sum of their maximum principal curvatures at p . Using this result, we can deduce that two models that are created by different CAD systems would be deemed interchangeable in applications where the bending direction at a fixed point is well defined for both models, if the upper bound in (6) is less than or equal to the the maximum tolerable bending difference at that

point. We note that this curvature-based upper bound can be large when the principal curvatures at that point are high; however, this does not imply that the difference in the shapes would be unacceptable for all applications.

Remark 2. S_p can be considered an implicit proxy in local model comparisons even if the CAD systems provide principal curvatures directly via queries and we do not need to compute the shape operator via (4).

Note that proxy models that rely on parametric information can only be used if the CAD models are represented by B-reps in the CAD systems. If curvatures and parametric derivatives cannot be obtained directly via queries, we use the normal vector fields of the surfaces associated with the local regions of interest as proxies and compare the discrete Gaussian curvatures. For a point p labeled as *on* by both of the systems during the PMQ checks, we define the normal vector field associated with the point cloud around p as follows:

- Let $\{p_i\}_{i=1}^5$ be *on* points scattered uniformly in a convex ϵ^* -neighborhood of p with $\epsilon^* \geq \epsilon$ on M_i , where ϵ denotes the proxy parameter. Suppose that $\{p_i\}$ s are ordered counter-clockwise in a circle around p .
- Use the normal vectors $\{N_i\}_{i=1}^5$ at the barycenters of the triangles formed by these points (See Figure 3)

The discrete Gaussian curvature at the point p could then be defined as the area on the unit sphere bounded by the spherical polygon in Figure 3 formed via a discrete version of the Gauss map, and the normal vectors (Crane et al., 2013). Using this definition, we compute and compare the discrete Gaussian curvatures of the surfaces at p . We remark that the Gaussian curvature is an intrinsic property, unlike the shape operator. Thus, it allows us to compare the CAD models based on their inherent surface geometry, regardless of their position or orientation within their ambient domains, offering a different form of comparison.

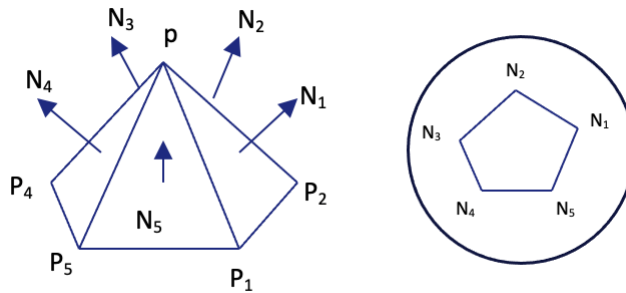


Figure 3: Image of the neighborhood of p under the Gauss map

2.2 Global Shape Comparison via Topological Properties 265

In this section, we describe the global shape comparisons that yield quantitative comparisons of the homological groups. Such comparisons can be used for detecting a mismatch in the number of 1D (circular) holes or 2D holes (cavities) in a model comparison, or the loss of such features after a model transfer. It is possible to assess and compare the geometry of models locally without assuming any changes in the shapes they represent, whereas one would need to explicitly decompose a model, that is, impose changes in the shapes that the models represent, to assess and compare the topological properties of the models in a local manner. Thus, global comparisons provide a first-level assessment for determining the interoperability of CAD systems based on model interchangeability, while local comparisons offer a second-level, more detailed model comparison.

We define algebraic complexes, including the Vietoris-Rips (Hausmann, 1995) and Witness complexes (Edelsbrunner & Harer, 2010) as proxies. Then, we compare the number of k -dimensional holes in the models by using these proxies and the evolutions of such holes by using the filtrations of these complexes.

First, we state some key topological concepts that are used or referred to in this section below.

Definition 1 (Simplicial Complex (Hatcher, 2001)). *A simplicial complex \mathcal{K} is a set of simplices (e.g. triangles in 2D, tetrahedrals in 3D) that satisfies the following:*

- (a) *Every face of a simplex in \mathcal{K} is also in \mathcal{K} .*
- (b) *The non-empty intersection of any two simplices $\sigma_1, \sigma_2 \in \mathcal{K}$ is a face of both σ_1 and σ_2 .*

Definition 2 (Cech Complex (Ghrist, 2015)). *A Cech complex denoted by \mathcal{C}_ϵ is a simplicial complex that corresponds to the nerve (Hatcher, 2001) of the union of ϵ -balls around the points in a point cloud that resides in a metric space (Chazal et al., 2014).*

Definition 3 (Vietoris-Rips Complex (Hausmann, 1995)). *Given a point set $\{x_i\} \in X$, the Vietoris-Rips complex is defined as an abstract simplicial complex $R_\epsilon(X)$ over X by the following:*

$$[x_0, x_1, \dots, x_k] \in R_\epsilon(X) \iff d(x_i, x_j) \leq \epsilon, \forall i, j \in [0, k],$$

where $[x_0, x_1, \dots, x_k]$ is a k -simplex.

Definition 4 (Filtration (Hausmann, 1995)). *A filtration is the evolution of k -dimensional holes along a sequence of simplicial complexes. The following nested sequence of subcomplexes of a complex \mathcal{R}^i is an example of a filtration we denote by \mathcal{F}^i :*

$$\emptyset =: \mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \dots \subseteq \mathcal{R}_n := \mathcal{R}^i \tag{7}$$

By the Nerve Theorem (Ghrist, 2015), Cech complex is homotopy equivalent to the union of balls. This implies that the Cech complex \mathcal{C}_ϵ^i we construct by using the union of balls M^i derived from the CAD model M_i is homotopy equivalent to M_i once the necessary conditions for the homotopy equivalence between M^i and M_i are met. Since the computing time of a Cech complex is exponential in point cloud size, in practice, Vietoris-Rips complexes are used (Edelsbrunner & Harer, 2008; Zomorodian, 2010). Although the *Nerve Theorem* does not apply to Vietoris-Rips complexes, these complexes still provide a considerably good approximation of the model due to the following relation between the Cech and Vietoris-Rips complexes (Edelsbrunner & Harer, 2010; Hatcher, 2001):

$$\mathcal{C}_\epsilon \subset \mathcal{R}_\epsilon \subset \mathcal{C}_{\sqrt{2}\epsilon} \quad (8)$$

From (8), it follows that both complexes are equal for sufficiently small ϵ . Therefore, while constructing the Vietoris-Rips complex, a ball size in the range $(\epsilon, \sqrt{2}\epsilon)$ should be considered as the allowable distance between points. We refer the interested reader to the survey by Edelsbrunner and Harer (Edelsbrunner & Harer, 2008) for more details on these concepts.

Let \mathcal{R}_ϵ^i be a Vietoris-Rips complex obtained from the point cloud, denoted by \tilde{M}^i , consisting of the points that are labeled as *on* during the PMQ analysis of the CAD model M_i . Let \mathcal{F}^i and \mathcal{F}^j be the filtrations of the complexes \mathcal{R}_ϵ^i and \mathcal{R}_ϵ^j , respectively. Via \mathcal{F}^i (resp. \mathcal{F}^j), we build a sequence of \mathcal{R}_ϵ^i (resp. \mathcal{R}_ϵ^j) to identify the homological groups of the shape represented by \tilde{M}^i (resp. \tilde{M}^j). We assume that \tilde{M}^i and \tilde{M}^j are totally bounded spaces, i.e., they have finite ϵ -covers for every $\epsilon > 0$ so that the persistence diagrams of \mathcal{R}_ϵ^i and \mathcal{R}_ϵ^j are well-defined (Chazal et al., 2014). Then, we use the bottleneck distance (Edelsbrunner & Harer, 2010) to compare the persistence diagrams.

Lemma 3. *Let $D(\mathcal{R}_\epsilon^i)$ be the persistence diagram of the filtration \mathcal{F}^i of the complex \mathcal{R}_ϵ^i , which serves as a proxy for the CAD model M_i . Then,*

$$d_b(D(\mathcal{R}_\epsilon^i), D(\mathcal{R}_\epsilon^j)) \leq 4\epsilon, \quad (9)$$

where $d_b(\cdot, \cdot)$ denotes the bottleneck distance (Edelsbrunner & Harer, 2010).

Proof. By Lemma 1 in (Sap & Shapiro, 2019) and Theorem 5.2 in (Chazal et al., 2014), we have

$$d_b(D(\mathcal{R}_\epsilon^i), D(\mathcal{R}_\epsilon^j)) \leq 2d_{GH}(\tilde{M}^i, \tilde{M}^j) \leq 4\epsilon, \quad (10)$$

where $d_{GH}(\cdot, \cdot)$ denotes the Gromov-Hausdorff metric. Thus, the models may have the same topological features, which are defined as the homological groups of dimension less than or equal to 2, with at most 4ϵ accuracy. \square

Remark 3. *Note that the impact of the removal or unintentional disappearance of a hole of a certain size during model transfer or format translation can vary depending on the*

application of interest. Quantifying the sizes of features that may disappear during model transfer or that do not belong to the shared feature set of the models being compared may be crucial for some applications.

Another algebraic complex that we use is the *Witness complex* (Edelsbrunner & Harer, 2010). A Witness complex is constructed from a subset of a given point set which is called the set of landmark points. Landmark points serve as the vertices in the algebraic complex and the rest of the points in the given set help determine which simplices occur in the complex. The computation of topological invariants via Witness complexes was presented in (Silva & Carlsson, 2004). We note that Witness complexes can be used instead of Vietoris-Rips complexes and may be preferable while using certain data sets, since they do not require the user to define the maximum filtration value and offer reduced computational cost. However, since they provide a coarser approximation of the topology of the data set and are sensitive to the choice of landmark points, we use them only as a complementary tool to support our analysis based on Rips complexes.

3. Algorithmic Frameworks

The first framework yields semi-automated testing since it requires template files which would substitute for the CAD model files in the interoperability testing. The second framework offers fully-automated testing as it lets the user execute the interoperability testing using only the CAD model files.

Our implementations of these frameworks can be executed over CAD models with non-standard formats such as the ones provided by Solidworks (namely, .sldprt) and Autodesk (namely, .ipt), as well as the models in STEP format (See Section 4).

3.1 Testing components

The key executable component of the testing systems based on these frameworks is named *DTest*. In the semi-automated framework, *DTest* runs over template files, and in the automated framework it runs over the CAD models.

Another important component of our testing is the *template file*. We introduce an additional command-line interface called *PTemplate*, for generating template files. *PTemplate* can be used independently by the user to generate the template files required by the semi-automated framework or utilized within the automated framework to decouple *DTest* from the CAD systems (See Figure 5). *PTemplate* is among the distinguishing features of our testing approach, setting it apart from the conventional CAD interoperability, model comparison, and validation testing practices.

In the following subsections, we briefly describe *DTest* and the template file.

3.1.1 DTest

DTest determines the proxy parameter, constructs and compares the proxy models, and evaluates the interchangeability of the CAD models based on the accuracy of the test and the tolerance value entered by the user. It has two main functions: *configure* and *evaluate*.

- *Configure*
 - Constructs the proxy models that can substitute for the CAD models in the model comparison,
 - Computes the shape properties of the proxy models (if a property comparison is prompted).
- *Evaluate*
 - Runs the comparison test over the models via their proxies,
 - Derives and outputs an interoperability report with respect to the standards that *DTest* sets for the model interchangeability based on the information provided by the user.

We note that the comparison test may refer to the comparison of shape properties or the comparison of the geometry of the CAD models. Thus, *configure* may not need to compute shape properties in some cases. For example, if we are investigating the geometric similarity of models, then *evaluate* directly compares the CAD models with respect to the relevant similarity measure using the proxy models, which would be point clouds in this scenario.

DTest can connect to and query multiple CAD systems independent of platform. This centralized structure of *DTest* makes it fully scalable to implement more queries, or connect to more CAD systems. *DTest* can also delegate the communication with CAD systems to *PTemplate*. Another highly scalable feature of *DTest* is that it can compute the proxy information in parallel.

3.1.2 Template files

Template files hold all the information required for the proxy model constructions. They can be generated by the user manually or via *PTemplate*. A template file is created via *PTemplate* by the following command:

```
./PTemplate <CAD_Model_file> <Density> <CAD_System>  
<Units> [-o Optional Parameters]
```

The optional parameters are information that can either be inferred or computed automatically, but can also be specified. They include:

<template_name> <absolute_tolerance> <parametric_tolerance> <dimension> 397
 <boundary_dimension> <convexity>. 398

The template file comprises the following four sections with their listed contents: 399

- OS: Operating system and its version 400
- CAD System and Version 401
- Queries 402
- Parameter units 403
- Tolerances: Different types of tolerances a CAD system may utilize 404
- Model information: ProxyFile, Topology type, Model Dimension, Convexity 405
- Bounding box dimensions 406

For example, when we run *PTemplate* over the rocket shaft model, it creates a template 407
 file of the following form. 408

```
<?xml version= "1.0"?> 409
<template> 410
  <OS>Ubuntu 22.04</OS> 411
  <CAD_System_and_Version> 412
    <CAD_System>4</CAD_System> 413
    <Version>6</Version> 414
  </CAD_System_and_Version> 415
  <Queries_to_use> 416
    <PMQ>1</PMQ> 417
    <distance>1</distance> 418
    <volume>1</volume> 419
    <area>1</area> 420
    <surface_properties>1</surface_properties> 421
    <hausdorff>1</hausdorff> 422
  </Queries_to_use> 423
  <Unit_parameters>mm</Unit_parameters> 424
  <Tolerances> 425
    <Abs_tol>0.0001</Abs_tol> 426
    <Par_tol>0.00001</Par_tol> 427
    <ang_tol>0.00000000001</ang_tol> 428
    <alg_tol>1</alg_tol> 429
  </Tolerance> 430
  <Model_Information> 431
    <FileName>ProxyFile</FileName> 432
```

```

    <Topology>Polysurface </Topology> 433
    <Model_dim>3</Model_dim> 434
    <Boundary_dim>2</Boundary_dim> 435
    <Convexity>1</Convexity> 436
  </Model_information> 437
  <Bounding_box_coords> 438
    <xmin>325</xmin> 439
    <ymin>-55</ymin> 440
    <zmin>-55</zmin> 441
    <xmax>604</xmax> 442
    <ymax>55</ymax> 443
    <zmax>55</zmax> 444
  </Bounding_box_coords> 445
</template> 446

```

Query responses are stored in a .txt file denoted by the *ProxyFile* above. For different 447
lists of available and necessary queries, this proxy file may need to be updated. Here, 448
we include convexity as a geometric property since it is an important component of the 449
theoretical results associated with the derivation of the integral property comparisons 450
(Sap & Shapiro, 2019). 451

Remark 4. *Template files would ideally accommodate the system information that was 452
active in the design environment where the model was created. However, it may not be 453
possible to retrieve the system information active during the generation of a CAD model 454
after the model is saved in a standard format. Therefore, we use the system information 455
active in the CAD environment where the model was read in STEP format without any 456
issues when we execute tests on models in STEP format using the semi-automated testing. 457*

Template file generation is an intermediary step that enables us to filter the wide 458
range of data available in CAD model files and extract the model data we need for proxy 459
constructions independent of any proprietary CAD file format and model representation. 460
This yields multiple desirable results. First, the unpredictable data loss that may occur 461
due to representation conversions or format translations is avoided. Secondly, the user 462
is granted the freedom to choose whether they would like to provide *DTest* with the 463
CAD models or not. Thirdly, the testing procedure can be carried over representation- 464
independent information of the CAD models. Lastly, *DTest* can avoid communication 465
with the CAD systems. The last two results emphasize on how we establish the repres- 466
entation and system-independence in the interoperability testing of CAD systems. 467

We note that the automation offered by *PTemplate* is limited to the global compar- 468
isons of the models. To automatically generate local proxies via *PTemplate*, we need to 469
be able to identify the local regions that should be compared or let the user specify these 470
regions. For example, one may use methods for determining the minimum feature, and 471
then identifying its neighborhood as the base space for the local proxy assuming that 472

the neighborhoods of minimum features are the local parts of the models that should be compared. Such investigations are beyond the scope of this paper, though.

3.2 Semi-Automated Testing Framework

Within this framework, we build the proxy models by using the information provided in the template files and compute the properties of these proxy models (Sap & Szabo, 2020). During the testing based on this framework, *DTest* collects the necessary information from the template files to determine the parameters such as ϵ (ball radius or parameter for the Vietoris-Rips filtration, etc). Figure 4 illustrates the structure of this framework.

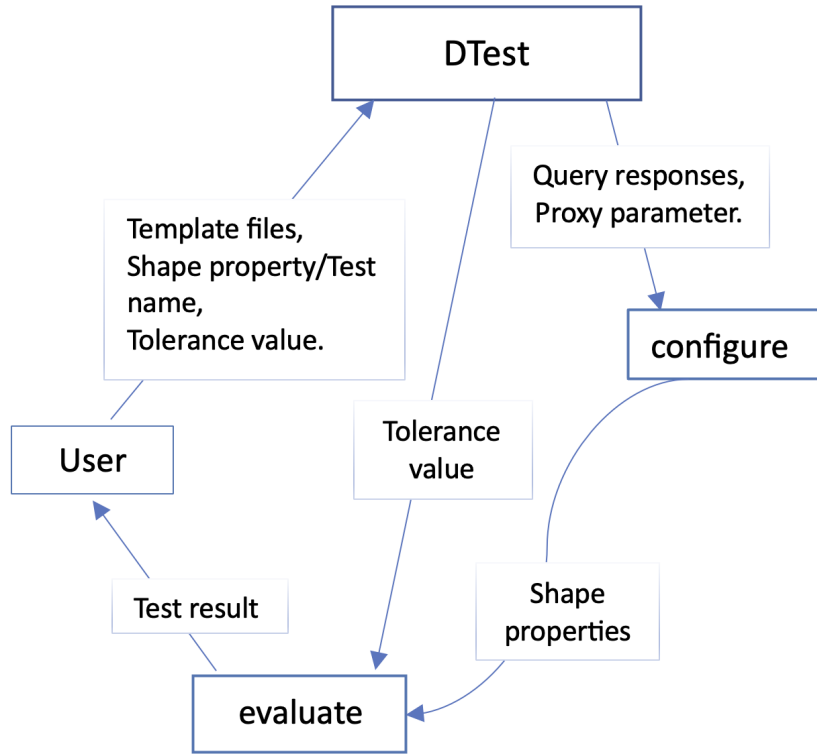


Figure 4: Semi-automated testing and discovery for interoperability diagram

3.3 Automated Testing Framework

The automated testing framework allows for building the proxy models and computing the properties of these proxy models by using the CAD models that are saved in different formats in distinct CAD systems as well as the CAD models given in standard formats such as STEP. This testing procedure also does not require format translations and is representation-independent, and derives the property information via queries. Figure 5 illustrates the structure of this framework.

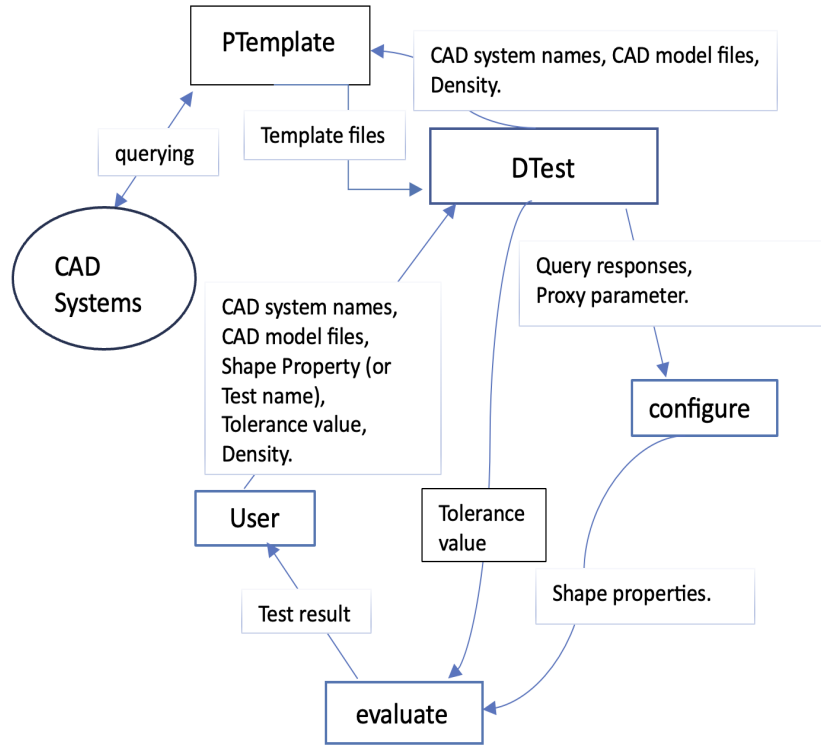


Figure 5: Automated testing and discovery of interoperability diagram

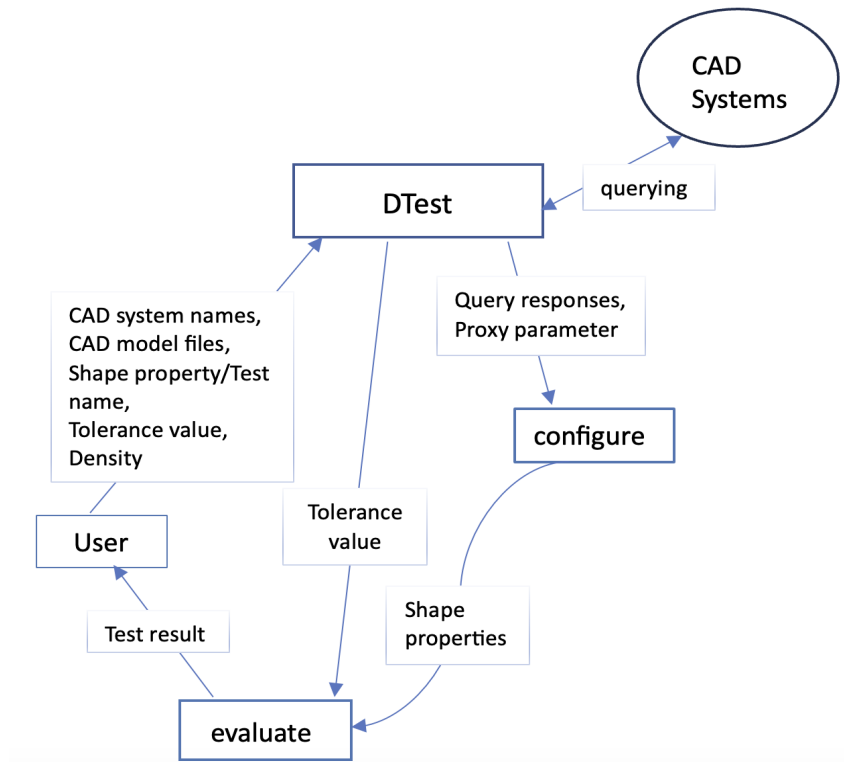


Figure 6: Reduced automated testing and discovery of interoperability diagram

For practical reasons, we use a reduced version of the system in Figure 5 during local 488 comparisons. This reduced system is illustrated in Figure 6 and can also be used for global 489 comparisons, if preferred. The only difference between the original automated *DTest* and 490

its reduced copy is that the original version delegates the querying of CAD systems to *PTemplate*, thus, offers a system-independent interoperability testing. This results in obtaining a higher-level automation of the semi-automated approach we described in Section 3.2. The reduced version on the other hand offers a more practical automated interoperability testing where *DTest* directly queries the CAD systems and uses these query responses in the interoperability testing.

4. Applications

In this section, we focus on model interchangeability as an application of the testing based on the automated algorithmic framework since it forms the basis of our interoperability analysis as described in the previous sections.

4.1 Generic interoperability testing via abstraction

Suppose M_1 and M_2 are two CAD models created in the respective CAD systems C_1 and C_2 . M_1 and M_2 are deemed interchangeable if the result of a comparison test states that M_1 and M_2 differ by a scalar value θ which includes the accuracy of the method and is smaller than the tolerance value specified by the user. This result would imply that the CAD systems C_1 and C_2 can interoperate with an accuracy of θ in applications where their models M_1 and M_2 are used to represent the same physical shape in 3D.

To determine if M_1 and M_2 are interchangeable based on the stated test with the stated tolerance value, we first construct the proxies M^i of M_i for $i = 1, 2$ with the query responses. Then, we compare the geometry and topology of M^1 and M^2 . Finally, we return a quantitative comparison of the models M_1 and M_2 by using the accuracy of M^i in approximating M_i and the discrepancy between the proxies.

We execute the testing by running the following command:

```
./DTest <System1> <System2> <Model1> <Model2> <Output File> <Density>
<Tolerance>
```

where *Output File* denotes the file the test results are written to, and *Density* refers to the density of the point grid over the models' bounding boxes that we use for the point-membership queries (PMQ).

In many cases, the use of multiple tests independently or in a sequential order may be necessary. For example, one may need to know the location of a hole that disappeared after a format translation or a model transfer, or one may need to identify a hole that may exist only in one of the CAD models. In these scenarios, we would first compare the topology of the models to detect the absence of a hole with a size above the tolerated value, and then compare the geometry of the models to determine the location of a dense mismatch in the PMQ responses suggesting the presence of a region of the same size as the hole consisting of points that are labelled as *out* for one of the models.

In Sections 4.2 and 4.3, we present the experiments that we conducted on CAD models representing the shaft and nose of a 3D rocket model in SolidWorks (28.0) and AutoDesk Inventor (2021).

4.2 The Local Comparison of CAD Models

Both SolidWorks and AutoDesk Inventor can provide differential information directly via queries and allow access to the surface parametrization. On the rocket nose where the surface is locally C^2 , we manually pick a point classified as ‘on’ and in each system find the closest point on the surface, along with its parameters, and query both systems independently for the normal vectors, curvatures, and first order derivative information at this point. See Figure 7 for the location of the chosen point. The results of the differential property checks over the models at this point are shown in Table 3.

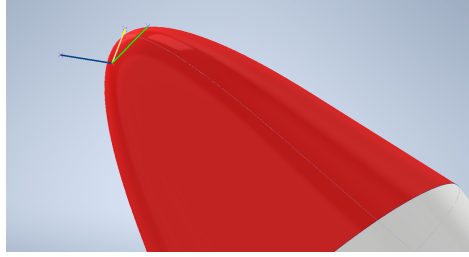


Figure 7: The normal vector along with the principal eigenvectors at a specified point on the rocket nose

Table 3: Local Differential Property Check Results

| SolidWorks | AutoDesk Inventor |
|------------------------------|-----------------------------|
| Normal Vectors | |
| $(-0.71, -0.71, 1.14e-5)$ | $(-0.71, -0.71, 1.14e-5)$ |
| Max Curvature | |
| 1.04016573063088 | 1.04016573063086 |
| Min Curvature | |
| 0.530330085889595 | 0.530330085889594 |
| r_u Derivative | |
| $(-1.04e-14, 7.11e-5, 4.41)$ | $(2.96e-15, 7.11e-5, 4.41)$ |
| r_v Derivative | |
| $(-4.00, 3.00, -4.93e-5)$ | $(-4.00, 3.00, -4.93e-5)$ |

We conclude from the data that there are minimal differences in the differential properties of the SolidWorks and AutoDesk Inventor models despite the complexity of the local queries, which include distance queries to find a point on the boundary. The curvatures matched up to machine precision.

Using our proxy model approach, we also compare the surfaces locally through their tangent planes at this chosen point, p , as described in the Section 2.1. In a local surface

region with an (8×8) array of points, we obtain the following results when we set $\epsilon = 0.0533274494$.

$$\begin{aligned} d_H(M_i(\mathcal{N}_{p_0}^\epsilon), M_i^t(\mathcal{N}_{p_0}^\epsilon)) &= 2.96163366 \times 10^{-3} \leq \epsilon, \\ d_H(M_2^t(\mathcal{N}_{p_0}^\epsilon), M_1^t(\mathcal{N}_{p_0}^\epsilon)) &= 7.606238603 \times 10^{-5} \\ &\leq 0.106654899 = 2\epsilon, \end{aligned}$$

where $i = 1, 2$.

Moreover, we have

$$\begin{aligned} d_H(M_1(\mathcal{N}_{p_0}^\epsilon), M_2(\mathcal{N}_{p_0}^\epsilon)) &\leq d_H(M_1(\mathcal{N}_{p_0}^\epsilon), M_1^t(\mathcal{N}_{p_0}^\epsilon)) \\ &+ d_H(M_2^t(\mathcal{N}_{p_0}^\epsilon), M_1^t(\mathcal{N}_{p_0}^\epsilon)) + d_H(M_2(\mathcal{N}_{p_0}^\epsilon), M_2^t(\mathcal{N}_{p_0}^\epsilon)) \\ &\leq 2 \times 2.96163366 \times 10^{-3} + 7.599055739 \times 10^{-5} \\ &= 0.00599932970244 \leq 0.213309797 = 4\epsilon \end{aligned}$$

Note that these results are supportive of the theory (See Lemma 1).

For completeness, we also derive the shape operators of the models represented by the following diagonal matrices in the base of their eigenvectors:

$$\begin{aligned} S_{p_0}^1 &= \begin{bmatrix} 0.530330085889595 & 0 \\ 0 & 1.04016573063088 \end{bmatrix} \\ S_{p_0}^2 &= \begin{bmatrix} 0.530330085889594 & 0 \\ 0 & 1.04016573063086 \end{bmatrix} \end{aligned}$$

This also yields $\|S_{p_0}^1 - S_{p_0}^2\| = 1.9985 \times 10^{-14} \leq 2.08033146126174 = |\kappa_{max}^1| + |\kappa_{max}^2|$ as expected by the theory (See Lemma 2).

Remark 5. We included different numbers of significant digits for different quantities while listing the experimental results to highlight the discrepancies in a precise manner.

4.3 The Global Comparison of CAD Models

In these experiments, the shaft or the nose of the rocket model shown with a grid of points in Figure 8 is given as a SolidWorks model. We run *DTest* over this SolidWorks model and its counterpart by the Autodesk Inventor. For the comparison of integral properties, we used ball covers as the proxies, thus, defined the proxy parameter as the ball radius. We utilized the Structural Bioinformatics Library (SBL) (Cazals & Dreyfus, 2017) in generating these proxies and computing their surface areas and volumes within our testing. In particular, we used their implementation of (Cazals et al., 2011) to compute the volume and surface area of a union of balls.

Then, we constructed Vietoris-Rips and Witness complexes from the point clouds generated by ϵ -samplings of the above-mentioned rocket model parts. We used JavaPlex

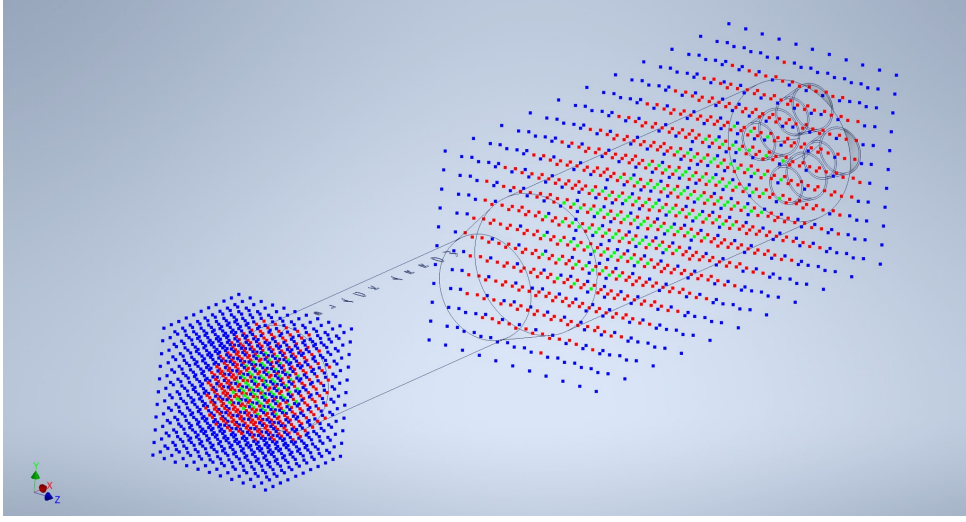


Figure 8: The black arrow rocket model with point clouds over the rocket shaft and nose. PMQ results: *green*, *red*, and *blue* points indicate *in*, *on*, and *out* points, respectively.

(Tausz et al., 2014) to derive the complexes and persistence barcodes, and the Gudhi library (Gudhi: Geometric understanding in higher dimensions, 2023) to obtain the persistence diagrams and compute the bottleneck distances. We used persistence barcodes (H. Edelsbrunner & Zomorodian, 2002) to illustrate the existence intervals of the topological features for different parameter values.

For each CAD model, the query points to construct the proxy reside in a fixed grid with dimensions that depend on the bounding box of the model. We use the user input density value, which is denoted by d , along with the bounding box values $\{b_x^f, b_y^f, b_z^f, b_x^b, b_y^b, b_z^b\}$ to compute a grid of points to query. We do so by choosing $(d + 3)^3$ points on the model with a proxy parameter ϵ , which in the experiments below is defined as the radius of circumscribed sphere of a grid cell, i.e.,

$$\epsilon = \frac{1}{2} \sqrt{\left(\frac{b_x^b - b_x^f}{d}\right)^2 + \left(\frac{b_y^b - b_y^f}{d}\right)^2 + \left(\frac{b_z^b - b_z^f}{d}\right)^2}.$$

Then, we define the point set as follows:

$$\{(b_x^f - \epsilon + \alpha a_x, b_y^f - \epsilon + \beta a_y, b_z^f - \epsilon + \gamma a_z) : 0 \leq \alpha, \beta, \gamma \leq d + 2; \alpha, \beta, \gamma \in \mathbb{Z}\},$$

where $a_x = \frac{b_x^b - b_x^f + 2\epsilon}{d+2}$, $a_y = \frac{b_y^b - b_y^f + 2\epsilon}{d+2}$, $a_z = \frac{b_z^b - b_z^f + 2\epsilon}{d+2}$. This bounding box is computed by *PTemplate* using the given system, and then saved in the template file.

4.3.1 Experiment 1

In this experiment, we compare the shafts of the rocket models. The proxy parameter, ϵ , is taken as 15 mm. While this is relatively large compared to the model dimensions specified in the sample template, we recognize this as a limitation and discuss potential

mitigation strategies for future work in Section 5.1.

*Running test on model 1 black_arrow_shaft.sldprt and on model 2
black_arrow_shaft.ipt with a tolerance of 14 mm.*

The compatibility of the models is as follows:

Volume:

SolidWorks and AutoDesk Inventor proxy models have compatible volumes with a spatially-scaled accuracy of 9.3016 mm.

Surface Area:

SolidWorks and AutoDesk Inventor proxy models have compatible surface areas with a spatially-scaled accuracy of 3.4641 mm.

Hausdorff Distance:

SolidWorks and AutoDesk Inventor proxy models have geometric dissimilarity measured by the Hausdorff distance as 0.1144658 mm.

The observed differences in volumetric and surface area computations highlight the discrepancies between model proxies, providing accuracy levels for the comparison of representation independent abstract models. In the existence of a mutual proxy, that is, when the accuracy levels above equals zero, the volumetric and surface area differences of the CAD models would be bounded above by the apriori error estimates associated with the inaccuracies of the proxy model approximations of the CAD models. The existence of a mutual proxy would yield homotopy equivalence and all of its consequences presented in (Sap & Shapiro, 2019).

Table 4: Witness Filtration of the Shaft Model

| Attributes | SolidWorks | AutoDesk Inventor |
|---|---|---|
| <i>Max dimension</i> | 3 | 3 |
| <i>Max filtration value (mm)</i> | 4.45 | 4.45 |
| <i>Number of divisions</i> | 1000 | 1000 |
| <i>Size of the point cloud</i> | 573×3 | 672×3 |
| <i>Number of landmark points</i> | 50 | 50 |
| <i>Number of simplices</i> | 935 | 928 |
| <i>Betti numbers array</i> | [1, 0, 1] | [1, 0, 1] |
| <i>Euler characteristic</i> | 2 | 2 |
| <i>Persistent holes with their persistence intervals (mm)</i> | Dim=0: [0.0, ∞), Dim=2: [2.5142, ∞). | Dim=0: [0.0, ∞), Dim=2: [2.4831, ∞). |

In Table 4, we present the details of the filtration of the Witness complexes for the shaft part of the rocket model. In Figures 9 and 10, we illustrate the persistence barcodes that we obtained from the Witness filtrations of the shaft model in SolidWorks and AutoDesk Inventor, respectively. The number of points labelled as *in* or *on* during the PMQ analysis of the CAD models differ as can be seen in Table 4. This exemplifies the inherent unpredictability involved in comparisons based solely on query-based approaches.

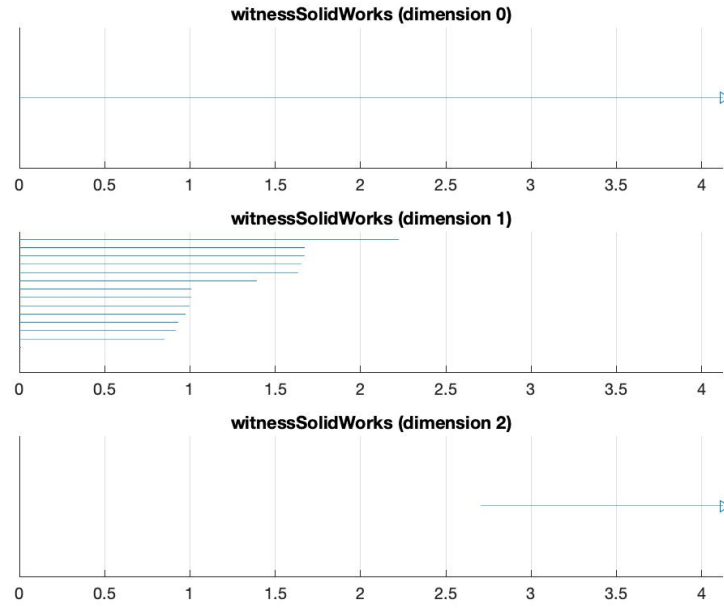


Figure 9: The persistence barcodes obtained from the Witness filtration for the SolidWorks shaft model

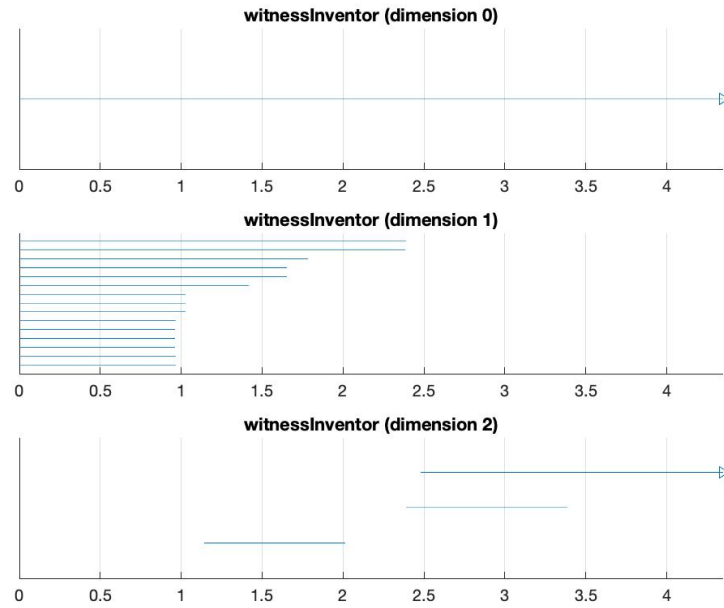


Figure 10: The persistence barcodes obtained from the Witness filtration for the AutoDesk Inventor shaft model

We remark that the results obtained in this experiment are supportive of Lemma 3 609 in Section 2.2 and the general topological theory considering the shapes represented by 610 these CAD models. 611

4.3.2 Experiment 2

The following shows an output of the same experiment carried over the rocket nose with a proxy paramater, $\epsilon = 9$ mm.

Running test on model 1 black_arrow_nose.sldprt and model 2 black_arrow_nose.ipt with a tolerance of 6 mm.

The compatibility of the models is as follows:

Volume:

SolidWorks and AutoDesk Inventor proxy models have compatible volumes with a spatially-scaled accuracy of 5.2422 mm.

Surface Area:

SolidWorks and AutoDesk Inventor proxy models have compatible surface areas with a spatially-scaled accuracy of 4.8305 mm.

Hausdorff Distance:

SolidWorks and AutoDesk Inventor proxy models have geometric dissimilarity measured by the Hausdorff distance as 0.06466714 mm.

In Tables 5 and 6, we present the details of the filtration of the respective Vietoris-Rips and Witness complexes for the nose part of the rocket model.

Table 5: Vietoris-Rips Filtration of the Nose Model

| Attributes | SolidWorks | AutoDesk Inventor |
|---|---|---|
| <i>Max dimension</i> | 3 | 3 |
| <i>Max filtration value (mm)</i> | 16 | 16 |
| <i>Number of divisions</i> | 100 | 100 |
| <i>Size of the point cloud</i> | 314×3 | 315×3 |
| <i>Number of simplices</i> | 24928 | 24994 |
| <i>Betti numbers array</i> | [1, 0, 1] | [1, 0, 1] |
| <i>Euler characteristic</i> | 2 | 2 |
| <i>Persistent holes with their persistence intervals (mm)</i> | Dim=0: [0.0, ∞), Dim=2: [10.4, ∞). | Dim=0: [0.0, ∞), Dim=2: [10.4, ∞). |

Table 6: Witness Filtration of the Nose Model

| Attributes | SolidWorks | AutoDesk Inventor |
|---|----------------------------|----------------------------|
| <i>Max dimension</i> | 3 | 3 |
| <i>Max filtration value (mm)</i> | 2.0684 | 2.0712 |
| <i>Number of divisions</i> | 1000 | 1000 |
| <i>Size of the point cloud</i> | 314×3 | 315×3 |
| <i>Number of landmark points</i> | 30 | 30 |
| <i>Number of simplices</i> | 530 | 605 |
| <i>Betti numbers array</i> | [1, 0, 1] | [1, 0, 1] |
| <i>Euler characteristic</i> | 2 | 2 |
| <i>Persistent holes with their persistence (mm) intervals</i> | Dim=0: [0.0, ∞) | Dim=0: [0.0, ∞) |
| | Dim=2: [1.5781, ∞) | Dim=2: [1.5740, ∞) |

In Tables 7 and 8, we illustrate the bottleneck distances between the persistence diagrams for the homology groups of dimension 0, 1, and 2, of both the shaft and nose models provided by Solidworks and AutoDesk Inventor, yielding comparisons between the persistence of the topological features of the models.

Table 7: Bottleneck Distances between the Persistence Diagrams of the Rips Complexes

| Hole Dimensions | Shaft | Nose |
|-----------------|-------|------|
| 0 | 13.99 | 3.68 |
| 1 | 2.28 | 0.00 |
| 2 | 0.00 | 0.00 |

Table 8: Bottleneck Distances between the Persistence Diagrams of the Witness Complexes

| Hole Dimensions | Shaft | Nose |
|-----------------|-------|------|
| 0 | 0.00 | 0.00 |
| 1 | 0.82 | 0.79 |
| 2 | 0.36 | 0.00 |

We note that the Witness complex implementation uses $\frac{R}{8}$, where R denotes the maximum distance from the landmark points to the other points in the set, as the maximum filtration value, whereas we define the maximum filtration values for the Rips complexes. Thus, we observe a noticeable scaling difference between the maximum filtration values resulting from the use of these complexes. Table 7 suggests the presence of mismatches between the models' 0 and 1-dimensional homological groups within our tolerance threshold and provides a comparison yielding a positive interoperability result based on the expected accuracy. We note that these mismatches were not as emphasized by the Witness complex (See Table 8). Thus, by using these two different types of algebraic complexes simultaneously, our testing yields complementary results, increasing the confidence levels of the test outcomes.

4.3.3 Comments on Experimental Results

The results highlight the impact of minor differences between the systems. The systems calculate bounding boxes differently (as can be seen from the Hausdorff distance), and the noise generated from this difference is then amplified by particular choices of the PMQ tolerance. For example, a particular choice of the PMQ tolerance could remove an entire row of balls from a proxy model, which can significantly impact the properties. This issue can be alleviated and more accurate property approximations can be obtained by using a much more dense randomized array of query points. However, this would require considerably more computational power than we had in this research. The experiments were conducted on a laptop running Windows 10 with an Intel CoreTM i7-7500U (2.7GHz) processor, 16 GB of DDR4 RAM, and Intel HD Graphics 620 graphics card, and took up to 30 minutes to complete. They took up to 30 minutes, with proxy computation being the main factor influencing the runtime. Although this may not be practically ideal, the goal of this research is to offer theoretically supported algorithms that can be used for verifying the interoperability of CAD systems based on the interchangeability of their models in a systematized manner and introduce command-line interfaces that can be utilized in carrying out these checks in an automated manner. Our testing is applicable to verification checks involving larger sets of query points. Numerical results support our theoretical derivations and serve as proofs of concept.

We note that using a sequence of processing systems comes with the risk of introducing supplementary differences in the models. However, we commit to keeping track of discrepancies in the accuracy of representations and algorithms. One of our primary goals is to eliminate the unpredictability of data loss that arises from using distinct systems or tools by quantifying errors and providing test results with confidence measures.

5. Conclusion

In this paper, we presented two theoretically-supported algorithmic frameworks that yield semi-automated and fully-automated testing for the verification of interoperability of two distinct CAD systems based on the interchangeability of their models. Both algorithmic frameworks provide indirect and representation-independent comparisons of CAD models through their abstract proxies and the interpretation of CAD models via queries. They allow local model comparisons via differential properties and provide a wide set of global model comparisons. Our theoretical results extend the proxy model and query-based approach described in (Sap & Shapiro, 2019) via additional proxy models and shape comparison tests.

We established independence from CAD representations and file formats by restricting the domain of comparison required to verify the interchangeability of CAD models to shape proxies. We developed testing tools that enable establishing independence from proprietary scripting languages and algorithms used in creating and analyzing CAD mod-

els. Precisely, we introduced a command-line interface, *DTest*, that constructs proxies for the CAD models and executes the interoperability testing by focusing on a set of shape comparison tests with specified tolerance values. With its centralized nature, *DTest* can create the system components in any scripting language used by the CAD systems and can operate with CAD software written for different operating system versions over a network connection, and it could even be hosted on the web. We presented alternative versions of *DTest* that would operate at different levels of automation. The level of automation is mainly determined by the involvement of template file generation, which can be used as an intermediary step to filter the wide range of data available in CAD model files and extract the model data we need for proxy constructions, in the framework. We provided test results for two distinct parts of a rocket model to demonstrate our testing procedure and illustrate the use of our testing tools. These results support our theoretical results and serve as proofs of concept. Our approach is suitable for verification checks involving larger sets of query points and complex shapes. However, we note that such verifications may require considerably more computational power. In Section 5.1, we discuss how we addressed the various technical challenges that we encountered during implementation and also provide insights into potential strategies for mitigating challenges that fall outside the scope of this work.

5.1 Technical Results

A complete testing of model interchangeability with respect to an external proxy model may become intractable in the presence of small features or high precision. This challenge could be alleviated by using selective testing, localization, and statistical measures. For example, an effective strategy for scaling could involve the use of an octree structure, combined with adaptive sampling techniques that increase point density near boundaries or small features. To handle larger and more complex models, another viable approach is to increase the number of query points. This can be done efficiently through parallelization, as point membership queries (PMQs) are inherently independent. Many CAD systems support PMQs on GPUs, enabling massive parallel execution and significantly accelerating the verification process. We leave the scaling to larger and more complex models outside the scope of this work, focusing instead on global comparisons between model parts and local comparisons at specified locations.

During testing, we faced an operating system and CAD software incompatibility. Most CAD systems are built for Windows, however, some geometric analysis tools such as SBL (Cazals & Dreyfus, 2017) are uniquely built for UNIX-based systems. Integrating all of these systems on a single machine presents a significant challenge, as virtual machines tend to be unable to handle these systems without frequent crashes. Our solution was to integrate between different machines via a network connection.

Our testing system currently supports SolidWorks (*SolidWorks*, 2020), Autodesk Inventor (*Autodesk Inventor*, 2020), and OpenCASCADE (*OpenCASCADE*, 2020). Integ-

ration of each of these systems into our system required the use of different scripting languages and implementations of the queries. We addressed this challenge via the centralized nature of *DTest*, which is written in C. For Solidworks and AutoDesk Inventor, we used pre-compiled executables for this connection, while we were able to integrate OpenCASCADE directly into *DTest*.

5.2 Future Direction

Our tests can be used to track the property changes under model transfers via standard formats, which may yield the ability to predict and maintain the model quality for long-term archival and retrieval (LOTAR) (‘Long Term Archiving and Retrieval - LOTAR’, 2019; Sap & Szabo, 2020).

Our approach focuses on model interchangeability as the primary challenge in evaluating the interoperability of CAD systems. However, our theory is fully applicable to model conversion between different representations, once the necessary adjustments are made to the algorithmic frameworks and additional software tools are implemented. Moreover, we note that translation validation can be framed as a model interchangeability problem and, therefore, can be addressed by our methodology and testing scheme.

Additionally, our automated template file generation could be used to generate a repository of template files that can be used for constructing different kinds of proxies and computing different types of properties in the future. For example, we may generate one template file for a topological comparison of 3D models and another one for the comparison of their integral properties.

Another extension of this research may focus on increasing the level of automation by enabling the investigation of tolerance levels that would allow systems to interoperate, when a negative interoperability result is reached for the specified tolerance value.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Duygu Sap: Conceptualization, Methodology, Software, Writing—original draft & reviewing. **Daniel P. Szabo:** Data curation, Software, Writing— editing.

Funding

DS was funded by the EPSRC grant EP/R029423/1. DPS was funded by the Ministry of Innovation and Technology of Hungary from the National Research, Development and

Innovation Fund, financed under the ELTE TKP 2021-NKTA-62. This research was initially funded by the DARPA contracts HR00111620042 and HR0011623402.

Data Availability

The main code required for running DTest is available in our GitHub repository: <https://github.com/17szabod/DTestFull>. For the most recent updates and edits, please contact the authors via email.

Acknowledgments

The authors thank Vadim Shapiro for his valuable support and helpful comments at the initial stages of this research.

References

- Autodesk Inventor* (Computer-aided design (CAD) and engineering software). (2020) (Version 2020, developed by Autodesk Inc.). Autodesk Inc. <https://www.autodesk.com/products/inventor>
- Boy, J., Rosché, P., & Cheney, D. (2012). *Recommended practices for geometric and assembly validation properties* (tech. rep.). CAx Implementor Forum.
- CADdoctor: Healing, Optimization and Simplification* (tech. rep.). (2019). Elysium. <https://www.elysium-global.com/en/product/caddoctor>
- CADfix* (CAD translation, repair and simplification software). (2019). ITI: International TechneGroup. <https://www.iti-global.com/cadfix>
- CADIQ* (CAD validation software). (2025). ITI: International TechneGroup. <https://www.iti-global.com/cadiq>
- Carmo, M. P. D. (2016). *Differential geometry of curves and surfaces*. Web.
- Cazals, F., & Dreyfus, T. (2017). *The structural bioinformatics library: modeling in bio-molecular science and beyond* (tech. rep.). <https://sbl.inria.fr>
- Cazals, F., Harshad, K., & Lorient, S. (2011). Computing the Volume of Union of Balls: a Certified Algorithm. *ACM Transactions on Mathematical Software*, 38(1). <https://doi.org/10.1145/2049662.2049665>
- Chazal, F., de Silva, V., & Oudot, S. (2014). Persistence stability for geometric complexes. *Geom. Dedicata*, 173, 193–214.
- CompareVidia. (2025). *CapVidia: CAD translation and validation software* (tech. rep.). <https://www.capvidia.com>
- Crane, K., De-Goes, F., Desbrun, M., & Schröder, P. (2013). Digital geometry processing with discrete exterior calculus. *ACM SIGGRAPH 2013 Courses*.

- Danjou, S., & Koehler, P. (2007). Challenges for design management. *Computer-Aided Design*, 4(1-6), 109–16. 786
- Edelsbrunner, H., & Harer, J. L. (2008). Persistent homology A Survey. *Discrete & Computational Geometry (DCG)*, 453. 787
- Edelsbrunner, H., & Harer, J. L. (2010). *Computational topology: An introduction*. American Mathematical Society. 788
- Ghrist, R. (2015). Barcodes: The persistent topology of data. *Bull. Amer. Math. Soc.*, 45(1), 61–75. 789
- Gonzalez-Lluch, C., Company, P., Contero, M., Camba, J. D., & Plumed, R. (2017). A survey on 3D CAD model quality assurance and testing tools. *Computer-Aided Design*, 83, 64–79. 790
- Gudhi: Geometric understanding in higher dimensions* (tech. rep.). (2023) (Version 3.7.0). INRIA. <https://gudhi.inria.fr> 791
- H. Edelsbrunner, D. L., & Zomorodian, A. (2002). Topological persistence and simplification. *Discrete Comput. Geom.*, 28(2), 511–533. 792
- Hatcher, A. (2001). *Algebraic topology*. Cambridge University Press. 793
- Hausmann, J. C. (1995). On the Vietoris-Rips complexes and a cohomology theory for metric spaces. *Annals of Mathematics Studies*, 138, 175–188. 794
- Hoffmann, C., Shapiro, V., & Srinivasan, V. (2011). *Geometric interoperability for resilient manufacturing* (tech. rep.). Purdue e-Pubs, Department of Computer Science. 795
- Hoffmann, C., Shapiro, V., & Srinivasan, V. (2014). Geometric interoperability via queries. *Computer-Aided Design*, 46, 148–159. 796
- Horwood, M., & Kulkarni, S. (2005). CAD data quality. *Eng. Des.*, 31(3), 14–6. 797
- ISO 10303, Industrial automation systems and integration* (Technical Report). (1998). International Organization for Standardization (ISO). 798
- Long term archiving and retrieval - LOTAR. (2019). <http://lotar-international.org/home.html> 799
- McKenney, D. (1998). Model quality: The key to CAD/CAM/CAE interoperability. *MSC software Americas users conference. Universal City, CA*. 800
- Mounir, H., Nizar, A., & Abdelmajid, B. (2012). Cad model simplification using a removing details and merging faces technique for a fem simulation. *Journal of Mechanical Science and Technology*, 26(11), 3539–3548. <https://doi.org/10.1007/s12206-012-0869-6> 801
- O’Neill, B. (2006). *Elementary differential geometry (revised 2nd ed.)* Elsevier/Academic Press. 802
- OpenCASCADE* (Open-source software development platform). (2020) (Version 7.4.0). ITI: International TechneGroup. <https://www.opencascade.com> 803
- OpenSCAD* (CAD validation software). (2019) (Version 2019.05). ITI: International TechneGroup. <https://www.openscad.org> 804

- Parasolid* (CAD software). (2019). Siemens. <https://www.plm.automation.siemens.com/global/en/products/plm-components/parasolid.html>
- Rhinoceros 5* (CAD Software). (2012) (Version 5). Robert McNeel and Associates. <https://www.rhino3d.com>
- Sap, D. (2019). *A review of geometric integrity criteria for military standards - 31000A* (tech. rep.). International Computer Science Institute (ICSI).
- Sap, D., & Shapiro, V. (2019). On verification of interoperability of CAD systems with a focus on invariant properties. *Computer-Aided Design*, 115, 256–266.
- Sap, D., & Szabo, D. P. (2020). An automated approach for the discovery of interoperability. *arXiv[cs.GR]*, 2001.10585.
- Silva, V. d., & Carlsson, G. (2004). Topological estimation using witness complexes. *SPBG'04 Symposium on Point - Based Graphics 2004*. <https://doi.org/10.2312/SPBG/SPBG04/157-166>
- Simon, V. (2019). *CADinterop* (tech. rep.). CAD Interop. <https://www.cadinterop.com/en/>
- Smith, B., Brauner, K., Kennicott, P., Liewald, M., & Wellington, J. (1983). *Initial graphics exchange specification(IGES) Version 2.0* (tech. rep.). NTIS, Springfield, VA, USA.
- SolidWorks* (Computer-aided design (CAD) software). (2020) (Version 2020, developed by Dassault Systèmes). Dassault Systèmes. <https://www.solidworks.com>
- Tausz, A., Vejdemo-Johansson, M., & Adams, H. (2014). JavaPlex: A research software package for persistent (co)homology. *Proceedings of ICMS 2014*, 8592, 129–136.
- Technical Data Package 31000A* (Military Standard Practice). (2013). ASSIST online database. http://everyspec.com/MIL-STD/MIL-STD-10000-and-Up/MIL-STD-31000A_45888/
- Wang, S., Morin, B., Roman, D., & Berre, A. J. (2011). A semi-automatic transformation approach for semantic interoperability in MDE.
- Zomorodian, A. (2010). Fast construction of the Vietoris-Rips complex. *Computers & Graphics*, 34(3), 263–271.