Computer Aided Geometric Design 80 (2020) 101876

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Reverse engineering of CAD models via clustering and approximate implicitization ☆

Andrea Raffo [a,b,*], Oliver J.D. Barrowclough [a], Georg Muntingh [a]

[a] Department of Mathematics and Cybernetics, SINTEF, Forskningsveien 1, 0373 Oslo, Norway
[b] Department of Mathematics, University of Oslo, Moltke Moes vei 35, 0851 Oslo, Norway

ARTICLE INFO

Article history:
Received 17 October 2018
Received in revised form 17 April 2020
Accepted 1 May 2020
Available online 15 May 2020

Keywords:
Reverse engineering
Clustering
Statistical learning
Approximate implicitization

ABSTRACT

In applications like computer aided design, geometric models are often represented numerically as polynomial splines or NURBS, even when they originate from primitive geometry. For purposes such as redesign and isogeometric analysis, it is of interest to extract information about the underlying geometry through reverse engineering. In this work we develop a novel method to determine these primitive shapes by combining clustering analysis with approximate implicitization. The proposed method is automatic and can recover algebraic hypersurfaces of any degree in any dimension. In exact arithmetic, the algorithm returns exact results. All the required parameters, such as the implicit degree of the patches and the number of clusters of the model, are inferred using numerical approaches in order to obtain an algorithm that requires as little manual input as possible. The effectiveness, efficiency and robustness of the method are shown both in a theoretical analysis and in numerical examples implemented in Python.

1. Introduction

Reverse engineering, also known as back engineering, denotes a family of techniques moving from the physical instantiation of a model to its abstraction, by extracting knowledge that can be used for the reconstruction or the enhancement of the model itself (Chikofsky and Cross, 1990). Such methods are widely used in applications to obtain CAD models, for instance from point clouds acquired by scanning an existing physical model. Among the various reasons behind this interest are the speed-up of manufacturing and analysis processes, together with the description of parts no longer manufactured or for which only real-scale prototypes are available. In mechanical part design, high accuracy models of manufactured parts are needed to model parts of larger objects that should be assembled together precisely. Industrial design and jewelry reproduction often use reverse engineered CAD models, which are usually easier to obtain than by directly designing complex free-form shapes with CAD systems. Medicine applies reverse engineering directly to the human body, such as in the creation of bone pieces for orthopedic surgery and prosthetic parts. Finally, reverse engineering is often exploited in animation to create animated sequences of pre-existing models.

In the field of isogeometric analysis (IGA), volumetric (trivariate) CAD models based on B-splines are used both for design and analysis. One major step in supporting IGA in industry is to provide backwards compatibility with today's boundary

☆ Editor: Michela Spagnuolo.
* Corresponding author.
E-mail address: andrea.raffo@sintef.no (A. Raffo).

https://doi.org/10.1016/j.cagd.2020.101876

represented (B-rep) CAD models. Conversion from bivariate surface models to trivariate volume models is a challenging, and as yet unsolved problem. Methods for constructing such models include block structuring (Skytt and Dokken, 2016) and volumetric trimming (Dokken et al., 2019), or a hybrid of the two. In all cases, information about the underlying surfaces in the model is of key importance. This problem was the original motivation behind the work in this paper. However, the method can also be utilized for many of the applications outlined above in the description of reverse engineering. In particular, redesign of models is an interesting application. As CAD models are typically represented as discrete patches, modifying them individually becomes a cumbersome job when the number of patches is large. Our method clusters all patches belonging to the same primitives together. This could be a key tool in better supporting parametric design in CAD systems that are based on direct modelling.

The first step of a reverse engineering process is digitalization, i.e., the acquisition of 2-D or 3-D data point clouds. Once the acquisition is completed, model features are identified from the point clouds using segmentation techniques. Edge-based approaches to segmentation aim to find boundaries in the point data representing edges between surfaces. Faced-based segmentation proceeds in the opposite order, trying to partition a given point cloud according to the underlying primitive shapes (Várady et al., 1997). We refer the reader to Niu et al. (2015a,b), Song et al. (2015), Shu et al. (2016), Xia et al. (2016), Ghadai et al. (2018) for further strategies of feature detection in CA(G)D. Finally, surface modelling techniques are applied to represent points in each of the detected regions. Some of the most used representations are point clouds, meshes (e.g. polygons or triangles), boundary representations (e.g. NURBS or B-spline patches), constructive solid geometry (CSG) models, spatial partitioning models, and feature-based and constraint-based models (Fudos, 2006). An example of relevant problem related to the just-obtained representations is part-in-whole retrieval (PWR), where a part is given as a query model and all models containing such a query part are retrieved (see, for example, Xiao et al., 2011 and Muraleedharan et al., 2019).

In this paper the geometry is reconstructed using implicit algebraic surfaces. According to a survey reported in Rossignac (1987), "99 percent of mechanical parts can be modelled exactly if one combines natural quadrics with the possibility of representing fillets and blends." Hence algebraic surfaces of low degree naturally provide global representations and are thus well suited to our application of detecting which patches belong to the same underlying geometry. The formal problem statement that we solve is as follows: given a set of non-overlapping rational parametrized patches in $\mathbb{R}^n$, partition the patches into subsets corresponding to the underlying primitive shape they originate from.

The main contributions of this paper are:

- The introduction of a novel algorithm to group patches of a given CAD model with respect to the underlying primitive geometry.
- A theoretical study of the stability and the computational complexity of the method.
- The validation of the method on both synthetic and real data.

The paper is organized as follows. In Section 2 we introduce the main tools of our method, presenting basic definitions and existing results that will be used later on. In Section 3, the core of our paper, we present the building blocks and mathematical results forming the foundation of the detection method. In Section 4 we formalize these results into an explicit algorithm. In Section 5 we present a theoretical analysis of the robustness of the method. In Section 6 we present several examples of the application of our method, based on an implementation of the algorithm in Python that is available online (Raffo et al., 2019). Section 7 concludes the main part of the paper, discussing encountered challenges and directions for future research. Finally, in Appendix A we report some of the proofs regarding the mathematical theory behind the algorithm.

2. Background

2.1. Clustering methods

Clustering is a well-established unsupervised statistical learning technique, gathering a group of objects into a certain number of classes (or clusters) based on a flexible and non-parametric approach (see Hastie et al., 2001 and its references). The grouping is performed so as to ensure that objects in the same class are more similar to each other than to elements of other classes. The problem can be traced back to ancient Greece and has extensively been studied over the centuries for its various applications in medicine, natural sciences, psychology, economics and other fields. As a consequence, the literature on the topic is vast and heterogeneous with the yearly *Classification Literature Automated Search Service* listing dozens of books and hundreds of papers published on the topic.

Mathematically, clustering refers to a family of methods aiming to partition a set $X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ such that all elements of a cluster share the same or closely related properties. In the context of statistics, any coordinate of a (data) point $\mathbf{x}_i = (x_{i1}, \dots, x_{in})^T \in \mathbb{R}^n$ is the realization of a *feature* or *attribute*.

The grouping is performed by defining a notion of dissimilarity between elements of X.

Definition 1 *(Dissimilarity).* A *dissimilarity* on a set X is a function $d : X \times X \to \mathbb{R}$ such that for all $\mathbf{x}, \mathbf{y} \in X$:

1. $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$ (symmetry);
2. $d(\mathbf{x},\mathbf{y}) \geq 0$ (positive definiteness);
3. $d(\mathbf{x},\mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$ (identity of indiscernibles).

The concept of dissimilarity is more general than the notion of distance, where in addition to the three properties in Definition 1 we take in account also the triangular inequality:

Definition 2 *(Distance).* A *distance* on a set X is a dissimilarity $d : X \times X \to \mathbb{R}$ such that:

4. $d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{z}) + d(\mathbf{z},\mathbf{y}) \qquad \forall\, \mathbf{x},\mathbf{y},\mathbf{z} \in X$ (triangular inequality).

The choice of the dissimilarity (or distance) strongly depends on the problem of interest, since it allows one to determine which elements are closer to each other by giving a greater importance to certain properties compared to other unfavorable ones. From a geometrical viewpoint, the use of a dissimilarity rather than a distance leads to a generalized metric space (Khamsi, 2015).

Example 1. A traditional way to measure distances in a Euclidean space $\mathbb{R}^n$ is a Minkowski distance, i.e., a member of the family of metrics:

$$d_P(\mathbf{x},\mathbf{y}) := \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}, \quad p \geq 1.$$

Common choices in this family are the Manhattan, Euclidean and Chebyshev distances (respectively: $p = 1, 2$ and ∞).

Once the dissimilarity d between elements has been chosen, it is necessary to define a generalization of d to compare subsets of X. In this context generalization means that if two clusters are singletons, then their dissimilarity corresponds to the original dissimilarity between the respective elements. Clearly the generalization of d is not uniquely determined and must be chosen on a case-by-case basis. Finally, an algorithm for grouping the data is designed via the defined d so as to reach a faithful clustering after a finite number of steps.

Conventional clustering algorithms can be classified into two categories: hierarchical and partitional algorithms. There are two families of hierarchical algorithms:

- Agglomerative hierarchical algorithms start with singletons as clusters and proceed by merging the clusters that are closest step by step.
- Divisive hierarchical algorithms work the other way around, starting with a single cluster containing all the elements and proceeding with a sequence of partitioning steps.

In this work we follow the agglomerative hierarchical approach in order to minimize the a priori knowledge that is required, consisting in general of only the number of clusters. Partitional algorithms require further information. In our method we propose an approach to infer this parameter, in order to keep the algorithm automatic. We emphasize that the result of different clustering methods highly depends on the considered dissimilarity, which is both an advantage and a disadvantage since it requires an additional choice to be made. For further details we refer the reader to Gan et al. (2007) and references therein.

2.2. Approximate implicitization

Implicitization is the process of computing an implicit representation of a parametric hypersurface. In algebraic geometry, traditional approaches to implicitization are based on Gröbner bases, resultants and moving curves and surfaces (Kotsireas, 2004). These methods present different computational challenges, such as the presence of additional solutions and a low numerical stability. Over the last decades several alternatives have been introduced in CAGD to reach an acceptable trade-off between accuracy of representation and numerical stability.

Approximate implicitization (see Barrowclough and Dokken, 2012; Dokken, 1997) defines a family of algorithms for "accurate" single polynomial approximations. Approximate implicitization can be performed piecewise by dividing the model into smooth components. This approach is of interest in applications from computer graphics, where the models are rarely described by a single polynomial. To simplify our presentation, we will restrict our attention to hypersurfaces in $\mathbb{R}^n$ (varieties of codimension 1), even though the classical theory can be applied to varieties of any codimension. In order to avoid degenerate parametric hypersurfaces, we assume their domains $\Omega \subset \mathbb{R}^{n-1}$ to be (Cartesian products of) closed intervals.

Definition 3 *(Exact implicitization of a parametric hypersurface).* Let $\mathbf{p} : \Omega \subset \mathbb{R}^{n-1} \to \mathbb{R}^n$ be a hypersurface in $\mathbb{R}^n$. An *exact implicitization* of $\mathbf{p}$ is any nonzero n-variate polynomial q such that:

$$q(\mathbf{p}(\mathbf{s})) = 0, \quad \mathbf{s} \in \Omega$$

Definition 4 *(Approximate implicitization of a parametric hypersurface).* Let $\mathbf{p} : \Omega \subset \mathbb{R}^{n-1} \to \mathbb{R}^n$ be a parametric hypersurface. An approximate implicitization of $\mathbf{p}$ within the tolerance $\varepsilon \geq 0$ is any nonzero n-variate polynomial q, for which there exists a continuous direction function $\mathbf{g} : \Omega \to \mathbb{S}^n$, with $\mathbb{S}^n \subset \mathbb{R}^n$ the unit sphere, and a continuous error function $\eta : \Omega \to (-\varepsilon, \varepsilon)$, such that:

$$q(\mathbf{p}(\mathbf{s}) + \eta(\mathbf{s})g(\mathbf{s})) = 0, \quad \mathbf{s} \in \Omega$$

Lemma 1 *(Dokken (1997)). Let $q(\mathbf{x}) = 0$ define an algebraic hypersurface of degree m and $\mathbf{p} = \mathbf{p}(\mathbf{s})$ be a (polynomial or) rational parametrization of (multi)degree $\mathbf{n}$ expressed in a given basis. Then it follows that the composition $q\big(\mathbf{p}(\mathbf{s})\big)$ can be expressed in a basis $\boldsymbol{\alpha}(\mathbf{s})$ for the polynomials of (multi)degree at most $m\mathbf{n}$. Explicitly,*

$$q\big(\mathbf{p}(\mathbf{s})\big) = (\mathbf{D}\mathbf{b})^T \boldsymbol{\alpha}(\mathbf{s}),$$

where

- *$\mathbf{D}$ is a matrix built from products of the coordinate functions of $\mathbf{p}(\mathbf{s})$;*
- *$\mathbf{b}$ is a column vector containing the unknown coefficients of q with respect to a chosen basis for the polynomials of total degree at most m.*

Note that, if $\mathbf{b} \neq 0$ is in the nullspace of $\mathbf{D}$, then $q\big(\mathbf{p}(\mathbf{s})\big) = 0$ and $\mathbf{b}$ contains the coefficients of an exact implicitization of $\mathbf{p}(\mathbf{s})$. If the kernel of $\mathbf{D}$ is trivial we look for an approximate implicit representation of $\mathbf{p}$ by minimizing the *algebraic error* $||q \circ \mathbf{p}||_\infty$.

Proposition 2 *(Dokken (1997)). Let $q(\mathbf{x}) = 0$ be an algebraic hypersurface of degree m and $\mathbf{p} = \mathbf{p}(\mathbf{s})$ be a polynomial or rational parametrization of (multi)degree $\mathbf{n}$ expressed in a given basis. Then*

$$\min_{\|\boldsymbol{b}\|_2=1} \max_{\mathbf{s}\in\Omega} |q\big(\mathbf{p}(\mathbf{s})\big)| \leq \max_{\mathbf{s}\in\Omega} ||\boldsymbol{\alpha}(\mathbf{s})||_2 \sigma_{\min},$$

where $\sigma_{\min}$ is the smallest singular value of the matrix $\mathbf{D}$ defined in Lemma 1.

Proof. Applying the Cauchy-Schwarz inequality,

$$\begin{aligned}\min_{\|\boldsymbol{b}\|_2=1} \max_{\mathbf{s}\in\Omega} |q\big(\mathbf{p}(\mathbf{s})\big)| &= \min_{\|\boldsymbol{b}\|_2=1} \max_{\mathbf{s}\in\Omega} |(\mathbf{D}\mathbf{b})^T \boldsymbol{\alpha}(\mathbf{s})| \\ &\leq \max_{\mathbf{s}\in\Omega} ||\boldsymbol{\alpha}(\mathbf{s})||_2 \min_{\|\boldsymbol{b}\|_2=1} ||\mathbf{D}\mathbf{b}||_2 \\ &= \max_{\mathbf{s}\in\Omega} ||\boldsymbol{\alpha}(\mathbf{s})||_2 \sigma_{\min},\end{aligned}$$

where we used that the smallest singular value $\sigma_{\min}$ of $\mathbf{D}$ takes the form

$$\sigma_{\min} = \min_{\|\boldsymbol{b}\|_2=1} ||\mathbf{D}\mathbf{b}||_2. \quad \square$$

Notice that the upper bound on the maximal algebraic error depends on the choice of the basis $\boldsymbol{\alpha}$. If the basis forms a non-negative partition of unity, then

$$\min_{\|\boldsymbol{b}\|_2=1} \max_{\mathbf{s}\in\Omega} |q\big(\mathbf{p}(\mathbf{s})\big)| \leq \sigma_{\min}.$$

Denoting by $\mathbf{b}_i$ the singular vector corresponding to the singular value σ_i of $\mathbf{D}$ and with q_i the algebraic hypersurface identified by $\mathbf{b}_i$, we have

$$|q_i\big(\mathbf{p}(\mathbf{s})\big)| \leq \max_{\mathbf{s}\in\Omega} ||\boldsymbol{\alpha}(\mathbf{s})||_2 \sigma_i. \tag{1}$$

Thus the singular value σ_i is a measure of how accurately $q_i(\mathbf{x}) = 0$ approximates $\mathbf{p}(\mathbf{s})$.

2.2.1. *Discrete approximate implicitization*

One of the fastest and simplest numerical implementations of approximate implicitization is based on a discrete least squares approximation of a point cloud $\mathscr{P} = \{\mathbf{p}(\mathbf{s}_i)\}_{i=1}^N$ sampled from the parametric manifold. The choice of using a point cloud sampled in parameter space is equivalent to choosing $\boldsymbol{\alpha}$ to be a Lagrange basis. This approach considers a *collocation matrix* $\mathbf{D}$, expressed elementwise in terms of a basis $\{\pi_j\}_{j=1}^M$ of the set of n-variate polynomials of total degree at most m as

$$D_{i,j} = \pi_j\big(\mathbf{p}(\mathbf{s}_i)\big), \qquad i = 1, \dots, N, \qquad j = 1, \dots, M. \tag{2}$$

Analogously to Proposition 2, we can bound the algebraic error as

$$\min_{\|\boldsymbol{b}\|_2=1} \max_{\mathbf{s}\in\Omega} |q(\mathbf{p}(\mathbf{s}))| \le \max_{\mathbf{s}\in\Omega} \|\boldsymbol{\alpha}(\mathbf{s})\|_2 \sigma_{\min} \le \Lambda(\boldsymbol{\alpha})\sigma_{\min},$$

where

- $\Lambda(\boldsymbol{\alpha})$ is the Lebesgue constant from interpolation theory defined by $\Lambda(\boldsymbol{\alpha}) := \max_{\mathbf{s}\in\Omega} \|\boldsymbol{\alpha}(\mathbf{s})\|_1$.
- $\sigma_{\min}$ is the smallest singular value of $\mathbf{D}$. This singular value depends on the point cloud $\mathscr{P}$, the total degree m and the basis $\{\pi_j\}_{j=1}^M$. Thus, a more correct notation is $\sigma_{\min}^{(m)}(\mathscr{P}, \{\pi_j\}_{j=1}^M)$. For the sake of simplicity, the dependence on the point cloud $\mathscr{P}$, the total degree m and the basis $\{\pi_j\}_{j=1}^M$ are omitted when not risking a misunderstanding.

We summarize the procedure to compute a discrete approximate implicitization as follows:

Algorithm 1. Given a point cloud $\mathscr{P} := \{\mathbf{p}(\mathbf{s}_1), \dots, \mathbf{p}(\mathbf{s}_N)\}$ sampled from a parametric hypersurface $\mathbf{p}$ and a degree m for the implicit polynomial.

1. Construct the collocation matrix $\mathbf{D}$ by evaluating the polynomial basis $\{\pi_j\}_j$ at each point of $\mathscr{P}$ as in (2).
2. Compute the singular value decomposition $\mathbf{D} = \mathbf{U\Sigma V}^T$.
3. Select $\mathbf{b} = \mathbf{v}_{\min}^T$ the right singular vector corresponding to the smallest singular value $\sigma_{\min}^{(m)}\left(\mathscr{P}, \{\pi_j\}_{j=1}^M\right)$.

3. The algorithm

CAD models can nowadays include millions of patches and billions of control points, making it necessary to develop algorithms that can process a huge amount of data. The novel approach we present combines flexibility, a controlled computational complexity and a high robustness to work in floating-point arithmetic. In exact arithmetic, the algorithm returns exact results. Our method does not require knowledge of the degree of the patches or the number of primitive shapes from which a certain model originates. Our idea is to infer these parameters using numerical approaches, to keep the algorithm automatic when they are not a priori available.

In this section the individual parts of the proposed reverse engineering algorithm are described in detail. We start with a pre-processing step aimed at dividing the set of all components in subsets according to their degree. Next, we describe a strategy for grouping the patches according to the primitive shapes they originate from, followed by description of how the free parameters are set. We end the section with a sketch of the complete algorithm.

3.1. *Calibration and pre-processing*

Let X be the set of all the patches composing the model in $\mathbb{R}^n$ (here: $n = 2, 3$). The pre-processing step consists of the partitioning $X = \cup_i X_i$ of the patches according to the suspected degree of their implicit form. The following lemma shows that we can use approximate implicitization to achieve our purpose.

Lemma 3. *Let τ be a non-trivial polynomial or rational patch in $\mathbb{R}^n$. Then the implicit degree of τ can be written as*

$$m := \min\left\{\bar{m} \in \mathbb{N}^* \text{ s.t. } \sigma_{\min}^{(\bar{m})} = 0\right\}, \tag{3}$$

where $\sigma_{\min}^{(\bar{m})}$ is the shorthand notation for the smallest singular value of Algorithm 1. Here, $\mathscr{P}$ is a point cloud sampled on τ and $\{\pi_j\}_{j=1}^M$ is any basis of total degree $\bar{m}$.

Proof. From Proposition 2 it follows that

- Approximate implicitization of degree $\bar{m} < m$ provides the coefficients of an approximate implicit representation, together with strictly positive smallest singular value that measures the accuracy of the approximation.

- Approximate implicitization of degree $\bar{m} = m$ provides the coefficients of the exact implicit representation, and the smallest singular value is zero. □

Remark 4. Let $N_{\min}$ be the minimum number of samples guaranteeing a unique exact implicitization (e.g. $N_{\min} = m^2 + 1$ for a non-degenerate rational parametric planar curve of implicit degree m). Lemma 3 can be extended to consider discrete approximate implicitization if at least $N_{\min}$ unique samples are considered (see Barrowclough and Dokken, 2012 for details).

Remark 5. Trimmed surfaces introduce a few additional complexities, but can be dealt with in a similar way. Given that trimming curves are often irregular, it is in general not possible to achieve a regular sampling that conforms to the boundaries of the model. In general we may say that if the underlying surface belongs to a certain primitive, then so will any trimmed region of that surface. Conversely, although it is possible that the trimmed region of a surface belongs to a certain primitive while the underlying surface does not, such cases are pathological. We can thus deal with trimmed surfaces in two ways. The first approach is to simply sample the underlying surface regularly, disregarding the trimming region. Another approach is to perform random oversampling of points within the trimmed region to ensure that enough samples are taken. The latter approach is also robust to pathological examples.

Equation (3) is useful to determine the implicit degree of a patch. This characterization for the implicit degree fails when we work in floating-point arithmetic, due to the approximations adopted in the computation. A possible modification is the relaxation of (3) by the weaker criterion

$$m := \min\left\{\bar{m} \in \mathbb{N}^* \text{ s.t. } \sigma_{\min}^{(\bar{m})} < \xi^{(\bar{m})}\right\}, \tag{4}$$

where the threshold $\xi^{(\bar{m})}$ is introduced to take into account the rounding-off error of floating point arithmetic and other possible sources of uncertainty. Note that this parameter depends on the implicit degree chosen in the computation of discrete approximate implicitization.

Suppose that we want to distinguish the patches of degree m from those of greater degree. A statistical approach to infer such thresholds is the following.

Algorithm 2. Given the implicit degree of interest m.

1. Generate:
   - $Q_1 \gg 1$ random patches of implicit degree m;
   - $Q_2 \gg 1$ random patches of implicit degree $m + 1$.
2. For each of these two sets, average the smallest singular values computed by applying discrete approximate implicitization of degree m on each patch. Denote these two values with $\overline{\xi}_1$ and $\overline{\xi}_2$.
3. Finally, compute $\xi^{(m)}$ as the geometric mean of $\overline{\xi}_1$ and $\overline{\xi}_2$, i.e.,

$$\xi^{(m)} = \sqrt{\overline{\xi}_1 \overline{\xi}_2}.$$

The use of a geometric mean allows computation of a value whose exponent in the scientific form is intermediate to the ones of $\overline{\xi}_1$ and $\overline{\xi}_2$. Notice that in applications such as CAD, curves and surfaces have often low implicit degree (e.g. degree 1 and 2 for natural quadrics). As we will see in Section 5, the estimation of the implicit degree can fail in floating-point arithmetic.

3.2. An agglomerative approach

As previously mentioned, hierarchical algorithms are subdivided into agglomerative and divisive hierarchical algorithms. The agglomerative approach starts with each object belonging to a separate cluster. Then a sequence of irreversible steps is taken to construct a hierarchy of clusters.

In our case, X is the set of patches to cluster. We want to reach a partition where patches in the same cluster are the only ones originating from the same primitive shape. We propose to derive the dissimilarity measure from discrete approximate implicitization. Since we have already partitioned $X = \cup_i X_i$ according to the suspected implicit degree of the patches, we can assume that all the patches in X have same degree m. Starting from each patch in a single cluster, at each step the two clusters with smallest dissimilarity are merged.

As a first step to define a dissimilarity on X, we identify each patch with a point cloud of N points sampled on its parametrization, where $N \geq N_{\min}$ and $N_{\min}$ is the constant of Remark 4. Typically a uniform sampling scheme is chosen. This assumption will guarantee the minimum number of samples for a unique exact implicitization when considering a pair of patches lying on the same primitive shape. Let $\ddot{X}$ be the set of such point clouds, following the indexing and partitioning of X. Notice that the points can be chosen such that the patches in X are in 1-1 correspondence to the point clouds of $\ddot{X}$, i.e.,

$$X \ni \tau \overset{1-1}{\longleftrightarrow} \ddot{\tau} := \{P_1^{\tau}, \dots, P_N^{\tau}\} \in \ddot{X}.$$

Definition 5 *(Family of candidate dissimilarities d_λ).* Let $\lambda \geq 0$. For each pair of patches τ_1 and τ_2 in X, let $\ddot{\tau}_1$ and $\ddot{\tau}_2$ be their respective point clouds in $\ddot{X}$. We define

$$d_\lambda(\tau_1, \tau_2) := \sigma_{\min}^{(m)}(\ddot{\tau}_1 \cup \ddot{\tau}_2) + \lambda ||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_2)||_2,$$

where

- λ is a regularization parameter.
- $\sigma_{\min}^{(m)}(\ddot{\tau}_1 \cup \ddot{\tau}_2)$ is the smallest singular value computed by applying discrete approximate implicitization of degree m to the point cloud $\ddot{\tau}_1 \cup \ddot{\tau}_2$.
- $||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_2)||_2$ is the Euclidean distance between the center of masses of the two point clouds.

Lemma 5. *Let X be a set of patches such that the centers of masses of the elements of X are distinct. Let d_λ be the map defined in Definition 5. Then:*

i) The map d_λ is a dissimilarity $\iff \lambda > 0$.
ii) There exists $\lambda^* > 0$ such that d_λ is a distance $\iff \lambda \geq \lambda^*$.

Proof. i) It is obvious that d_λ is non-negative and symmetric for any $\lambda \geq 0$. Let's then prove the identity of indiscernibles. Let τ_1, τ_2 be a pair of patches in X. Then

$$\begin{aligned} d_\lambda(\tau_1, \tau_2) = 0 &\iff \sigma_{\min}^{(m)}(\ddot{\tau}_1 \cup \ddot{\tau}_2) + \lambda ||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_2)||_2 = 0 \\ &\iff \begin{cases} \sigma_{\min}^{(m)}(\ddot{\tau}_1 \cup \ddot{\tau}_2) = 0 \\ \lambda ||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_2)||_2 = 0 \end{cases}, \end{aligned}$$

where the last equivalence arises from the non-negativity of the two terms. Notice that:

- The smallest singular value σ is zero iff τ_1 and τ_2 lie on the same hypersurface of degree m.
- $\lambda ||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_2)||_2 = 0$ iff the patches have the same center of mass or $\lambda = 0$. Since the first case is excluded by hypothesis, the lemma is proved.

ii) Let τ_1, τ_2, τ_3 be three patches in X. Then $d(\tau_1, \tau_2) \leq d(\tau_1, \tau_3) + d(\tau_1, \tau_2)$ iff

$$\lambda \geq \frac{\sigma_{\min}^{(m)}(\ddot{\tau}_1 \cup \ddot{\tau}_2) - \sigma_{\min}^{(m)}(\ddot{\tau}_1 \cup \ddot{\tau}_3) - \sigma_{\min}^{(m)}(\ddot{\tau}_2 \cup \ddot{\tau}_3)}{\sum_{i=1}^{2} ||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_i) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_3)||_2 - ||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_2)||_2} =: \lambda_{\tau_1, \tau_2, \tau_3},$$

which is well-defined by the triangle inequality for the Euclidean distance and since distinct patches have distinct center of masses by assumption. Hence, it follows that the triangular inequality holds (only) for d_λ with

$$\lambda \geq \lambda^* := \max_{\tau_1, \tau_2, \tau_3 \in X} \lambda_{\tau_1, \tau_2, \tau_3}. \quad \square$$

Remark 6. Notice that:

i) Any pair of patches τ_1 and τ_2 belongs to the same primitive shape iff $\sigma_{\min}^{(m)}(\ddot{\tau}_1 \cup \ddot{\tau}_2) = 0$ or, equivalently, iff $d_0(\tau_1, \tau_2) = 0$.
ii) A parameter $\lambda > 0$ is required to exploit the theory of clustering analysis. On the other hand, λ should penalize the term $||\mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\mathrm{CM}}(\ddot{\tau}_2)||_2$ in order to preserve the behavior of d_0 described i). We can choose $\lambda \approx 0$, e.g. $\lambda = 10^{-10}$, to approximate d_0 by a dissimilarity. We remark that this constant is typically much smaller than λ^*.
iii) The assumption in Lemma 5 is reasonable for applications such as CAD, where patches do not overlap.

The dissimilarities between pairs of patches are stored in the $|X| \times |X|$ matrix $\mathbf{D}_X$, known as the *dissimilarity matrix*.

Now that we have defined a dissimilarity between elements, we extend it to clusters by defining the map $D_\lambda : 2^X \times 2^X \to [0, +\infty)$ using a complete-linkage approach:

$$D_\lambda(C_i, C_j) := \max_{\tau_k \in C_i, \tau_l \in C_j} d_\lambda(\tau_k, \tau_l).$$

The main reason for choosing complete-linkage is its relatively low computational cost, since the dissimilarity matrix at step k is just a submatrix of $\mathbf{D}_X$. In addition numerical results show that it works well in practice.

Finally, starting from each patch in a separate cluster, we merge at each step the pair of clusters with smallest dissimilarity D_λ. The merging continues until the correct number of primitive shapes is detected, using the stopping criterion described in the next section.

3.3. Stopping criterion for the agglomerative approach

How should one estimate the final number of clusters? We propose to define a stopping criterion considering the map d_0 as follows:

- At each iteration k we compute, for each cluster, the maximum value of d_0 for pairs of patches. From a numerical viewpoint, it corresponds to an empirical estimation of the maximum error in the approximate implicitization of the patches.
- We consider then the maximum of the maxima and denote it by $e^{(k)}$. We will refer to $e^{(k)}$ as the *representation error at iteration k*.

Suppose L is the number of underlying primitives and let $P := |X|$ be the number of patches. Then:

- In exact arithmetic, $e^{(k)} = 0$ for $k = 1, \dots, P - L$ and $e^{(k)} > 0$ for $k = P - L + 1, \dots, N - 1$. Therefore the number of iterations can be defined as the maximum index k such that $e^{(k)} = 0$, i.e.,
$$\bar{k} := \max\{k | e^{(k)} = 0\}. \tag{5}$$
- In floating-point arithmetic, Equation (5) does not, in general, return any integer. We thus aim at designing a stopping criterion that can handle round-off error. We propose to estimate the number of iterations $\bar{k}$ as the index where the representation error jumps significantly for the first time, i.e.,
$$\bar{k} := \min\{k | e^{(k)} > \eta\}.$$

One way to choose the stopping tolerance η is to proceed similarly to Algorithm 2:

Algorithm 3. Given the maximum implicit degree of interest $m_{\max}$:

1. Generate $P_3 \gg 1$ sets $\{\mathscr{D}_i\}_{i=1}^{M}$ of random patches of implicit degree between 1 and $m_{\max}$.
2. Compute the number of iterations $\bar{k}_i$ for an exact clustering of each $\mathscr{D}_i$, considering that the number of clusters for training sets is known.
3. Compute, for each $\mathscr{D}_i$, the respective representation error e_i at iteration $\bar{k}_i$.
4. Define $\eta := (\min e_i)^2$.

Such an empirical threshold η lies between 0 and the smallest representation error of an incorrect representation. Whenever $e^{(k)}$ is smaller than η, the algorithm will proceed by joining the clusters having this smallest dissimilarity. Otherwise, the algorithm will stop.

As an alternative to this stopping criterion based on absolutes, one may prefer the following relative stopping criterion.

Algorithm 4. Given a sequence $e^{(k)}$, $k = 1, \dots, P - 1$, of representation errors:

1. Set $e^{(0)} := e^{(1)}$.
2. Define $\tilde{e}^{(k)} := e^{(k)}/e^{(k-1)}$, for $k = 1, \dots, N - 1$.
3. Define $\bar{k} := \left(\arg\max_k \tilde{e}^{(k)}\right) - 1$.

Additional details on the use of these two criteria are provided in Section 6.

4. Sketch of the algorithm for the detection of primitive shapes

We now present a sketch of the complete detection algorithm described earlier in this section. The input consists of a finite set of patches X lying on different manifolds and having different centers of mass. As output, the algorithm returns the partition of X corresponding to the manifolds the patches come from.

First, we partition X according to the implicit degree of the patches as described in Section 3.1. The tuning parameters $\{\xi^{(m)}\}$ are computed using Algorithm 2. The maximum implicit degree of the patches $m_{\max}$ is returned as a result of this first step.

Finally, we apply the agglomerative clustering approach described in Section 3.2 to each of the subsets X_i. We will denote with $X_i = \cup_j X_{i,j}^{(k)}$ the partition of the cluster X_i into the clusters $X_{i,j}^{(k)}$ at step k, and with $X_i = \cup_j X_{i,j}$ the final partition corresponding to the primitive shapes.

The tolerance η is assumed to be previously computed as in Section 3.3. We recall that in the agglomerative approach each patch constitutes a single cluster at the initial step.

```
Input: The set of polynomial patches $X$.
Output: The partition $X_1, \dots, X_{m_{\max}}$ according to the implicit degree.
Set $m := 0$ (degree for discrete approximate implicitization);
while $|X| > 0$ do
    Set $m := m + 1$ (increment the degree for approximate implicitization);
    Compute $\xi^{(m)}$ as defined in Algorithm 2;
    Set $X_m := \emptyset$;
    for $\tau \in X$ do
        if $\sigma_{\min}^{(m)}(\tau) < \xi^{(m)}$ then
            Set $X_m := X_m \cup \{\tau\}$;
            Set $X := X \backslash \{\tau\}$;
        end
    end
end
Set $m_{\max} := m$;
return $X_1, \dots, X_{m_{\max}}$
```

Algorithm 5: Partition of the set of patches X according to their implicit degrees.

```
Input: | $X = \cup_i X_i$   the initial partition according to the implicit degree.
       | $\eta$             the threshold for the stopping criterion.
Output: The final partition $X = \cup_{i,j} X_{i,j}$ according to the underlying primitive shapes.
for $i = 1 : m_{\max}$ do
    Set $P_i := |X_i|$ (number of clusters at step 0);
    if $P_i > 1$ then
        Initial step:
        Set $\mathbf{D}_X := \mathbf{0}_{P_i \times P_i}$ (dissimilarity matrix at step 0);
        for $j_1 = 1 : P_i$ do
            for $j_2 = j_1 + 1 : P_i$ do
                Set $\mathbf{D}_X(j_1, j_2) := d_\lambda\left(\tau_{j_1}, \tau_{j_2}\right)$.
            end
        end
        $\mathbf{D}_X := \mathbf{D}_X + \mathbf{D}_X^T$;
        Define the partition $X_i = \cup_{j=1}^{P_i} X_{i,j}^{(0)}$, where each cluster $X_{i,j}^{(0)}$ contains one and only one patch (agglomerative approach);
        Agglomerative process:
        Set $k := 0$ and $e^{(0)} := 0$;
        while $e^{(k)} < \eta$ do
            Set $P_i^k := |X_i|$ (number of clusters at step $k$);
            if $k = 0$ then
                Set $\mathbf{D}_X^0 := \mathbf{D}_X$ (dissimilarity matrix at step 0);
            else
                Set $\mathbf{D}_X^k := \mathbf{0}_{P_i^k \times P_i^k}$ (dissimilarity matrix at step $k$);
                for $j_1 = 1 : P_i^k$ do
                    for $j_2 = j_1 + 1 : P_i^k$ do
                        Set $\mathbf{D}_X^k(j_1, j_2) := D_\lambda(X_{i,j_1}^{(k)}, X_{i,j_2}^{(k)})$;
                    end
                end
                $\mathbf{D}_X^k := \mathbf{D}_X^k + \left(\mathbf{D}_X^k\right)^T$;
            Find the pair of clusters $X_{i,\bar{j}_1}^{(k)}$ and $X_{i,\bar{j}_2}^{(k)}$ minimizing $D_\lambda$;
            Set the partition for step $k + 1$ by merging $X_{i,\bar{j}_1}^{(k)}$ and $X_{i,\bar{j}_2}^{(k)}$;
            Compute $e^{(k+1)}$ as described in Section 3.3;
            Set $k := k + 1$;
        end
    end
end
return $X = \cup X_{i,j}^{(k-1)}$
```

Algorithm 6: Partition of the set of patches X corresponding to the underlying primitive shapes.

4.1. Computational complexity

In this section we analyze the computational complexity of the proposed algorithm in the case of curves and using approximate implicitization with total degree at most m.

4.1.1. Complexity of approximate implicitization

We start by determining the computational complexity for approximate implicitization. For curves in $\mathbb{R}^2$, the collocation matrix $\mathbf{D}$ (2) has the form $(\pi_j(\mathbf{p}_i))_{i=1,j=1}^{N,M}$, involving:

- N points $\mathbf{p}_1, \ldots, \mathbf{p}_N$ sampled from one or more segments.
- M polynomials $\pi_1, \ldots, \pi_M$ spanning the space $\mathbb{R}_m[x, y]$ of bivariate polynomials of total degree at most m. In particular,

$$M := \dim \mathbb{R}_m[x, y] = \frac{(m+1)(m+2)}{2}. \tag{6}$$

The computational complexity of approximate implicitization is the result of two contributions:

- For the monomial basis considered in this paper, using a triangular scheme to assemble the entries of $\mathbf{D}$ reduces the computational complexity to $\mathcal{O}(Nm^2)$.
- The singular value decomposition of an $N \times M$ matrix has computational complexity $\mathcal{O}(\min(NM^2, N^2M))$ (Beneš and Kruis, 2018). In our setting, this yields

$$\mathcal{O}(\min(NM^2, N^2M)) = \mathcal{O}(\min(Nm^4, N^2m^2))$$

In case of points lying exactly on parametric curves, we can (without loss of generality) set $N := N_{\min}$, where $N_{\min}$ is the minimum number of samples that guarantees a unique exact implicitization. By assuming the parametric planar curve to be rational non-degenerate, $N_{\min} = m^2 + 1$ (see Remark 4), which leads to a total computational cost of $\mathcal{O}(m^6)$. Although approximate implicitization exhibits high complexity with respect to patch degree, we are mainly concerned with low degree patches in this work, as motivated in the introduction. Due to this, the total computational time is dominated by the number of patches $P = \sum_{m=1}^{m_{\max}} P_m$, to be clustered.

4.1.2. Estimating the degree of the patches

Algorithm 5 runs approximate implicitization of degree m a total of $P - \sum_{k=0}^{m-1} P_k$ times, where we set $P_0 := 0$ and where $m = 1, \ldots, m_{\max}$. Thus the routine for estimation of degree has $\mathcal{O}(P)$ complexity in the number of patches. This result is independent of the dimension of the hypersurfaces to be clustered.

4.1.3. Assembling the dissimilarity matrix

Considering the degree of the patches to be constant (e.g. $m_{\max}$ in the worst case), the complexity of assembling the dissimilarity matrix is proportional to the number of elements in the matrix, that is $\mathcal{O}(P^2)$, regardless of the dimension of the hypersurfaces to be clustered. It may be noted, moreover, that each element of the dissimilarity matrix can be computed independently, implying that the matrix assembly is highly parallelizable (*embarrassingly parallel*).

4.1.4. Clustering procedure

The naive implementation of agglomerative hierarchical clustering requires $\mathcal{O}(P^3)$ operations (Day and Edelsbrunner, 1984). However, even with this implementation, for all cases we have encountered so far, the assembly of the dissimilarity matrix requires most of the computational time. The dominance of the assembly procedure is even more prominent for surfaces in $\mathbb{R}^3$. Thus, in practice, the number of patches required before the clustering step becomes dominant is excessively high. If the clustering of such a large number of patches is required, it is also possible to exploit the complete-linkage approach to implement the clustering procedure with $\mathcal{O}(P^2)$ complexity (Defays, 1977).

5. Theoretical analysis

In this section we state three propositions on the stability and robustness of discrete approximate implicitization under scaling, translation and rotation when the monomial basis is considered. The proofs are given in the appendix. These results of the study are generalizable to hypersurfaces of $\mathbb{R}^n$.

Proposition 7 *(Scaling and smallest singular value). Let*

$$\mathbf{p}_{\mathbf{a}}(t) = \big(a_1 x(t), a_2 y(t)\big), \qquad t \in [a, b] \subset \mathbb{R},$$

be a family of scaled polynomial or rational parametric segments, where $\mathbf{a} = (a_1, a_2) \in \mathbb{R}^2$ *with* $a_1, a_2 > 0$. *Let* $\mathscr{P}_{\mathbf{a}} := \{\mathbf{p}_{\mathbf{a}}(t_j)\}_j$ *be a family of uniformly sampled point clouds. Then*

$$\lim_{(a_1,a_2)\to(0,0)} \sigma_{\min}^{(m)}(\mathscr{P}_{\mathbf{a}}) = 0.$$

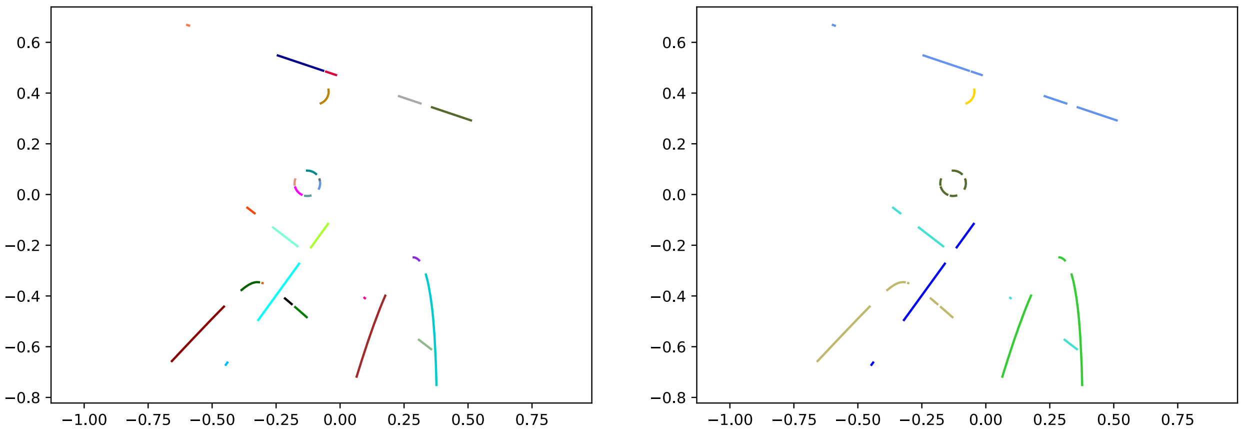


Fig. 1. Initial (left) and corresponding classified dataset (right), with segments colored by cluster. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Proposition 8 *(Translations and smallest singular value). Let*

$$\mathbf{p}_{\mathbf{a}}(t) = \big(x(t) + a_1, y(t) + a_2\big), \qquad t \in [a, b] \subset \mathbb{R},$$

be a family of translated polynomial or rational parametric segments, where $\mathbf{a} := (a_1, a_2) \in \mathbb{R}^2$. *Let* $\mathscr{P}_{\mathbf{a}} := \{\mathbf{p}_{\mathbf{a}}(t_j)\}_j$ *be a family of uniformly sampled point clouds. Then*

$$\lim_{a_k \to \infty} \sigma^{(m)}_{\min}(\mathscr{P}_{\mathbf{a}}) = 0, \qquad k = 1, 2.$$

Proposition 9 *(Rotation and smallest singular value). Let*

$$\mathbf{p}_\theta(t) = \big(x(t) \quad y(t)\big) \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \qquad t \in [a, b] \subset \mathbb{R},$$

be a family of polynomial or rational parametric segments rotated about the origin, where $\theta \in [0, 2\pi]$. *Let* $\mathscr{P}_\theta := \{\mathbf{p}_\theta(t_j)\}_j$ *be a family of uniformly sampled point clouds. Then there exists* $\alpha, \beta \in \mathbb{R}_{>0}$ *such that*

$$\alpha \le \sigma^{(m)}_{\min}(\mathscr{P}_\theta) \le \beta.$$

These propositions imply that the algorithm is not guaranteed correct in floating-point arithmetic, since the smallest singular value can be small enough that the weak condition (4) or the stopping criterion of Section 3.3 fail. However, these issues can be mitigated by performing further preprocessing on the data (rescaling to a fixed size, calibrating, etc.). The next section shows that the algorithm works well in practice.

6. Experimental results

Our algorithm has been tested on two- and three-dimensional examples. We first describe its behavior for segments originating from lines and conics in a plane, then we study its correctness, robustness and efficiency when applied to line segments and Bézier approximations of circular segments, and finally we test its correctness on a real-world industrial example of a 3D CAD model.

6.1. Straight lines and conics

Let $\mathscr{F}_c = \{\mathscr{C}_i\}_{i=1}^L$ be a family of L curves, where the type of each curve (e.g. straight line, parabola, ellipse, hyperbola) is randomly chosen with equal probability. Each curve $\mathscr{C}_i$ is obtained as follows: random parameters are sampled to define a rotation about the origin, a dilation and a translation of the canonical curve chosen as representative of the type of $\mathscr{C}_i$. A family of P_i continuous subsegments is sampled. The extracted segments are gathered in the set X. The method is tested on the set X, which, after a suitable rescaling, is contained in the region $[-1, 1] \times [-1, 1]$. An example is shown in Fig. 1.

We consider the stopping tolerance presented in Algorithm 3, as it has experimentally shown to work well with data affected by the only round-off error. The algorithm is run 10^5 times to compute the average misclassification rate of the approach. This index is given by the ratio between the number of misclassified segments and the total number of segments, averaged over the number of times the algorithm is run. The average misclassification rate is $2.14 \cdot 10^{-2}$. Equivalently, the 97.86% of the segments are on average correctly classified. Notice that the inaccuracy can be explained by the randomness

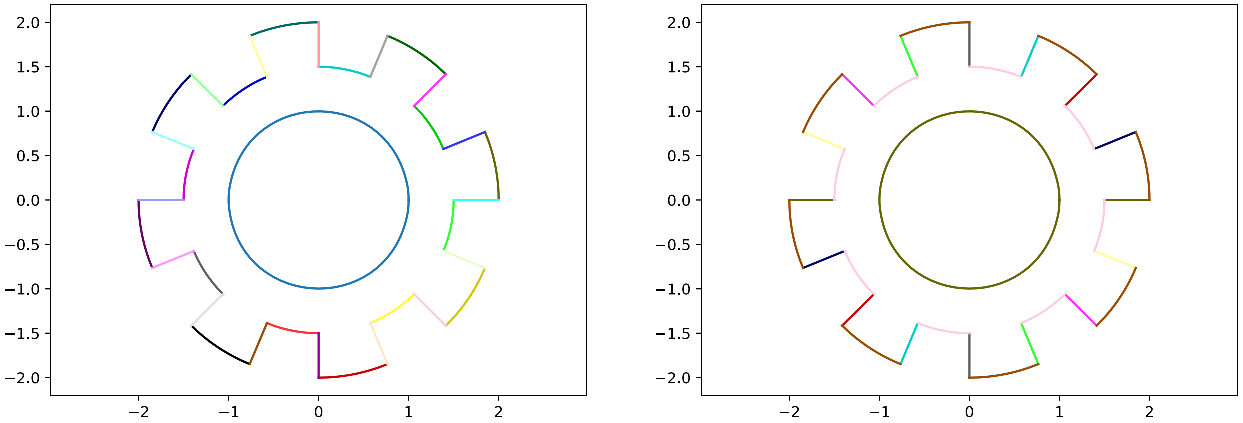
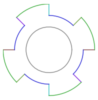
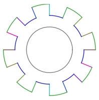
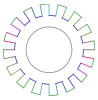
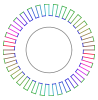
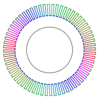
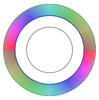


Fig. 2. Initial gear (left) and corresponding classified gear (right), with segments colored by cluster.

Table 1
A breakdown of the CPU times t_i (seconds) and complexity order $o_i := \log_2(t_{i+1}/t_i)$ when applied to gears with exponentially increasing number of teeth/segments.

						
# teeth	4	8	16	32	64	128
# segments	17	33	65	129	257	513
t^{assembly}	0.006	0.021	0.071	0.256	1.095	4.025
o^{assembly}	-	1.708	1.778	1.843	2.096	1.879
$t^{\text{clustering}}$	0.002	0.003	0.008	0.042	0.115	0.405
$o^{\text{clustering}}$	-	0.980	1.339	2.352	1.436	1.823
t^{total}	0.009	0.025	0.081	0.300	1.211	4.433
o^{total}	-	1.483	1.681	1.888	2.015	1.872

of the dataset, where some of the dilations can compromise the degree estimation or the stopping criteria (see Section 5 for details about the robustness of discrete approximate implicitization).

6.2. *Cubic Bézier curve approximations to circular segments*

Let X be the set of segments of the two-dimensional gear in Fig. 2. The gear is built as follows:

- Three concentric circles with radii $1, 1.5$ and 2 are approximated by means of cubic Bézier curves. Notice that the segments do not lie exactly on the underlying primitives.
- Concentric line segments are used to connect inner and outer arcs to form the teeth.

Again, we apply the algorithm in order to detect the primitive shapes underlying the model. We here use the stopping criteria introduced in Algorithm 4, because experimentally more robust when working with polynomial Bézier approximations. The resulting output shows the correct detection of the clusters.

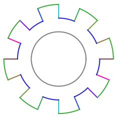
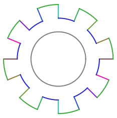
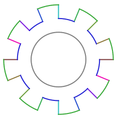
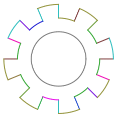
6.2.1. *Increasing number of teeth*

The algorithm is applied to gears with increasing numbers of teeth and run on a 2019 MacBook pro with 2.4 GHz 8-cores Intel®Core$^{\text{TM}}$ i9-processor, resulting in the CPU times shown in Table 1. Although these times suggest that both the dissimilarity matrix assembly and the clustering procedure in the implementation (Raffo et al., 2019) have $\mathcal{O}(P^2)$ complexity, a further breakdown of the CPU times (Raffo et al., 2019) shows that, in the naive implementation, repeatedly locating the smallest singular value in the clustering procedure yields $\mathcal{O}(P^3)$ complexity. However, for $P \leq 16384$ patches, this term is dominated by the $\mathcal{O}(P^2)$ complexity of the remainder of the algorithm.

6.2.2. *Increasing Gaussian noise*

Table 2 displays the results of running the algorithm on an 8-tooth gear with addition of synthetic Gaussian noise. Here, the mean is set to 0 while the standard deviation is increased until the algorithm fails. This experiment suggests that, although well-suited for parametrically defined patches affected by round-off error, this method may require additional information when the input is perturbed by a significant amount of noise.

Table 2
Sensitivity of the method when an 8-tooth gear is perturbed by Gaussian noise of fixed mean $\mu = 0$ and increasing standard deviation σ in each direction. Here, ✓ and ✗ mean, respectively, a correct and an incorrect clustering of the patches. The asterisk signifies that the stopping criterion of Algorithm 4 fails, but the method yields the correct result via a user-defined threshold (or when the number of clusters is known).

				
σ	10^{-5}	10^{-4}	10^{-3}	10^{-2}
result	✓	✓*	✓*	✗

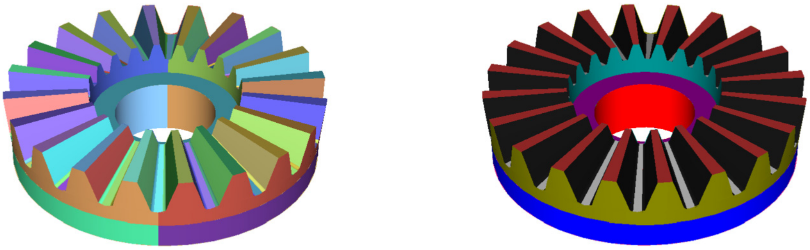


Fig. 3. Initial unclassified (left) and corresponding classified (right) Nugear model provided by STAM, with patches colored by cluster.

6.3. Surface patches from a real-world industrial example

In this example we utilize industrial data provided by the high-tech engineering firm STAM, based in Genoa, Italy. The data, from STAM's Nugear model, consists of 133 trimmed B-spline patches where the underlying geometry comes from planes, cylinders and cones. The individual patches are colored randomly in Fig. 3 (left). In order to deal with the trimmed patches in this example, we opt for a random oversampling approach, where we extract a random subsample of 64 points from oversampled surface data, cf. Remark 5. This approach is robust as we can always increase the density of the oversampling until we obtain 64 points within the trimmed region. Moreover, 64 points is always sufficient for dealing with patches of algebraic degree less than or equal to two, which are the targeted primitives of this example. The samples are generated using the same tessellation code that is used for visualization purposes.

The algorithm for surfaces in $\mathbb{R}^3$ proceeds in exactly the same way as the version for curves in $\mathbb{R}^2$, with the exception that we must apply approximate implicitization to surfaces in $\mathbb{R}^3$ rather than curves in $\mathbb{R}^2$. The algorithm correctly classifies all surfaces that should be classified together. The rest of the patches, dark-colored in Fig. 3 (right), should all be clustered individually. However, an empirical user-defined threshold was required in order to avoid some of these surfaces being classified together incorrectly. The issue arises in that some of the bicubic spline surfaces are very close to planar, but not exactly. These surfaces are each classified as quadrics in the first step of the algorithm, but due to the almost linear nature of the surfaces, pairs of them also fit well to quadric surfaces. Thus, it is difficult to choose a threshold that stops the algorithm at the appropriate point.

7. Conclusion

This paper presents a novel method for clustering patches of a CAD model with respect to the underlying surface they originate from, based on approximate implicitization. The method has been tested on both synthetic and real world industrial data, and some theoretical results are presented to show the properties of the method. The results show that the approach is computationally feasible and can handle thousands of patches in a matter of seconds.

Our core idea is conceptually quite simple: reverse engineering of CAD patches into their underlying primitives, by using their low-degree implicit representation consistently throughout the algorithm as a basis for establishing similarity in a clustering procedure. However, several modifications were necessary to make this work in practice. Clustering proceeds on local patches, and the goodness of fit of approximate implicitization depends on the scale and position of the coordinate system (e.g., locally a circle looks like a line). This was addressed by introducing an initial tuning step to adapt tolerances to the provided data, and by adding a term to the dissimilarity metric measuring the distance between patches. The latter change also makes our dissimilarity satisfy the formal definition of a dissimilarity for the types of patches considered in this paper.

For future work, we intend to extend the method to other contexts. For example, reverse engineering of physical models could also be done in this way, although work would be required on stabilizing the algorithm in the case of noisy data. The algorithm could also be applied to tessellated models, both for the purposes of redesign and upsampling of the tessellation resolution. In that case, the challenge would be segmenting which parts of the tessellation belong to different ‘patches’ of the model. Finally, we aim at designing a more robust version that can be used to treat data affected by noise and outliers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 675789 and No. 680448.

Appendix A

In this appendix we provide the proofs left out of Section 5, for discrete approximate implicitization of degree $d = 1, 2$ to point clouds with sufficiently many points sampled from curves of at least this degree. In this case the collocation matrix $\mathbf{D}$ has full rank.

Proof of Proposition 7. Equipping the basis with the lexicographic order, we can express the collocation matrix of the scaled curve $\mathbf{p}_{a_1,a_2}$ as

$$\mathbf{D}_{a_1,a_2} = \mathbf{D}_{1,1} \cdot \mathbf{S}_{a_1,a_2}, \qquad \mathbf{S}_{a_1,a_2} := \operatorname{diag}([1, a_1, a_2, a_1^2, a_1 a_2, a_2^2, \dots]),$$

with $\mathbf{D}_{1,1} := \mathbf{D}$. Notice that:

- The smallest singular value $\sigma_{\min}^{(m)}(\mathscr{P}_{\mathbf{a}})$ of $\mathbf{D}_{a_1,a_2}$ is the reciprocal of the largest singular value of

  $$\mathbf{D}^{\dagger}_{a_1,a_2} = \mathbf{S}^{-1}_{a_1,a_2} \cdot \mathbf{D}^{\dagger}_{1,1},$$

  where † denotes the Moore-Penrose inverse.
- The largest singular value of $\mathbf{D}^{\dagger}_{a_1,a_2}$ is the square root of the largest eigenvalue of $\mathbf{D}^{\dagger}_{a_1,a_2}(\mathbf{D}^{\dagger}_{a_1,a_2})^T$.

Therefore

$$\sigma_{\min}^{(m)}(\mathscr{P}_{\mathbf{a}}) = \left[\lambda_{\max}(\mathbf{B})\right]^{-1/2},$$

where

$$\mathbf{B} = \mathbf{S}^{-1}_{a_1,a_2} \cdot \mathbf{C} \cdot \mathbf{S}^{-1}_{a_1,a_2}, \qquad \mathbf{C} := \mathbf{D}^{\dagger}_{1,1} \cdot \mathbf{D}^{\dagger}_{1,1}{}^{T}.$$

Suppose $\mathbf{B}$ has eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$. Then

$$tr(\mathbf{B}) = \sum_{i=1}^{n} \lambda_i = c_{11} + c_{22}\frac{1}{a_1^2} + c_{33}\frac{1}{a_2^2} + \dots$$

where $c_{i,i}$ is the square of the Euclidean norm of the i-th row of $\mathbf{D}^{\dagger}_{a_1,a_2}$. Since $\mathbf{D}^{\dagger}_{a_1,a_2}$ has full rank whenever $a_1, a_2 \neq 0$, then $tr(\mathbf{B})$ (and consequently λ_1) approaches $+\infty$ when a_1 or a_2 approaches zero. It follows that $\sigma_{\min}^{(m)}(\mathscr{P}_{\mathbf{a}})$ approaches zero when a_1 or a_2 approaches zero. □

Proof of Proposition 8. The collocation matrix $\mathbf{D}_{a_1,a_2}$ has the form

$$\mathbf{D}_{a_1,a_2} = \mathbf{D}_{0,0}\mathbf{T}_{a_1,a_2}, \qquad \mathbf{T}_{a_1,a_2} := \left(\begin{array}{c|cc|ccc} 1 & a_1 & a_2 & a_1^2 & a_1 a_2 & a_2^2 \\ 0 & 1 & 0 & 2a_1 & a_2 & 0 \\ 0 & 0 & 1 & 0 & a_1 & 2a_2 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right),$$

and $\mathbf{D}_{0,0} := \mathbf{D}$. Notice that, since discrete approximate implicitization of degree 2 is applied to a curve of implicit degree greater than 2, we can assume that the collocation matrix has full rank. Notice also that:

- The smallest singular value of $\mathbf{D}_{a_1,a_2}$ is the reciprocal of the largest singular value of

$$\mathbf{D}^{\dagger}_{a_1,a_2} = \mathbf{T}_{-a_1,-a_2} \cdot \mathbf{D}^{\dagger}_{0,0}.$$

- The largest singular value of $\mathbf{D}^{\dagger}_{a_1,a_2}$ is the square root of the largest eigenvalue of $\mathbf{B} := \mathbf{D}^{\dagger}_{a_1,a_2}(\mathbf{D}^{\dagger}_{a_1,a_2})^T$.

Therefore

$$\sigma^{(m)}_{\min}(\mathscr{P}_{\mathbf{a}}) = \left[\lambda_{\max}(\mathbf{B})\right]^{-1/2},$$

where

$$\mathbf{B} := \mathbf{T}_{-a_1,-a_2}\mathbf{C}\mathbf{T}^T_{-a_1,-a_2}, \qquad \mathbf{C} := \mathbf{D}^{\dagger}_{0,0}\left(\mathbf{D}^{\dagger}_{0,0}\right)^T.$$

Suppose $\mathbf{B}$ has eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Then

$$tr(\mathbf{B}) = \sum_{i=1}^{n}\lambda_i = \sum_{0\leq i+j\leq 4} \alpha_{i,j} a_1^i a_2^j, \tag{7}$$

for some $\alpha_{i,j} \in \mathbb{R}$. Notice that $\alpha_{0,4} > 0$, since it is the square of the Euclidean norm of the 4-th row of the full-rank matrix $\mathbf{D}^{\dagger}_{0,0}$. A similar argument holds for $\alpha_{4,0}$. It follows that $tr(\mathbf{B})$ (and consequently λ_1) approaches $+\infty$ when either a_1 or a_2 approaches ∞, and consequently $\sigma^{(m)}_{\min}(\mathscr{P}_{\mathbf{a}})$ approaches zero when either a_1 or a_2 approaches ∞. □

Proof of Proposition 9. We treat separately the following cases:

Discrete approximate implicitization of degree 1. The collocation matrix $\mathbf{D}_\theta$ can be expressed as

$$\mathbf{D}_\theta = \mathbf{D}_0 \cdot \mathbf{R}_\theta, \qquad \mathbf{R}_\theta := \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix},$$

with $\mathbf{D}_0 := \mathbf{D}$. It follows that the SVD of $\mathbf{D}_\theta$ is

$$\mathbf{D}_\theta = \mathbf{D}_0\mathbf{R}_\theta = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{R}_\theta = \mathbf{U}\boldsymbol{\Sigma}\widetilde{\mathbf{V}}^T,$$

where $\widetilde{\mathbf{V}} := \mathbf{R}^T_\theta\mathbf{V}$ is unitary since $\mathbf{R}_\theta$ and $\mathbf{V}$ are unitary. Thus, the smallest singular value of discrete approximate implicitization of degree 1 is not influenced by rotations, and it suffices to choose $\alpha = \beta = \sigma^{(m)}_{\min}(\mathscr{P}_0)$.

Discrete approximate implicitization of degree greater than 1. The collocation matrix $\mathbf{D}_\theta$ can be expressed as

$$\mathbf{D}_\theta = \mathbf{D}_0 \cdot \mathbf{R}_\theta,$$

where $\mathbf{R}_\theta$ is the block diagonal matrix

$$\mathbf{R}_\theta = \left(\begin{array}{c|cc|ccc|c} 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & \cos\theta & \sin\theta & 0 & 0 & 0 & \cdots \\ 0 & -\sin\theta & \cos\theta & 0 & 0 & 0 & \cdots \\ \hline 0 & 0 & 0 & \cos^2\theta & \sin\theta\cos\theta & \sin^2\theta & \cdots \\ 0 & 0 & 0 & -2\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta & 2\sin\theta\cos\theta & \cdots \\ 0 & 0 & 0 & \sin^2\theta & -\sin\theta\cos\theta & \cos^2\theta & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right)$$

The case of degree greater than 1 differs since the matrix $\mathbf{R}_\theta$ is not unitary. The smallest singular value of $\mathbf{D}_\theta$ is the square root of the smallest eigenvalue of $\mathbf{D}^T_\theta\mathbf{D}_\theta$. The eigenvalues of $\mathbf{D}^T_\theta\mathbf{D}_\theta$ are continuous functions in θ. Since $\theta \in [0, 2\pi]$ lies in a compact space, we conclude by the extreme value theorem that each of these eigenvalues has a compact and connected range. Since $\mathbf{D}_\theta$ has full rank, we conclude that the range does not contain 0. □

References

Barrowclough, O.J.D., Dokken, T., 2012. Approximate implicitization using linear algebra. J. Appl. Math.

Beneš, Š., Kruis, J., 2018. Singular value decomposition used for compression of results from the finite element method. In: Special Section - CIVIL-COMP 2017. Adv. Eng. Softw. 117, 8–17.

Chikofsky, E.J., Cross, J.H., 1990. Reverse engineering and design recovery: a taxonomy. IEEE Softw. 7 (1), 13–17.

Day, W.H.E., Edelsbrunner, H., 1984. Efficient algorithms for agglomerative hierarchical clustering methods. J. Classif. 1 (1), 7–24.

Defays, D., 1977. An efficient algorithm for a complete link method. Comput. J. 20 (4), 364–366.

Dokken, T., 1997. Aspects of intersection algorithms and approximation. PhD thesis. University of Oslo.

Dokken, T., Skytt, V., Barrowclough, O.J.D., 2019. Trivariate spline representations for computer aided design and additive manufacturing. Comput. Math. Appl. 78, 2168–2182.

Fudos, I., 2006. CAD/CAM methods for reverse engineering: a case study of reengineering jewelry. Comput-Aided Des. Appl., 683–700.

Gan, G., Ma, C., Wu, J., 2007. Data Clustering Theory. Algorithms and Applications. ASA-SIAM Series on Statistics and Applied Probability.

Ghadai, S., Balu, A., Sarkar, S., Krishnamurthy, A., 2018. Learning localized features in 3D CAD models for manufacturability analysis of drilled holes. Comput. Aided Geom. Des. 62, 263–275.

Hastie, T., Tibshirani, R., Friedman, J., 2001. The Elements of Statistical Learning. Springer Series in Statistics.

Khamsi, M.A., 2015. Generalized metric spaces: a survey. J. Fixed Point Theory Appl. 17 (3), 455–475.

Kotsireas, I.S., 2004. Panorama of methods for exact implicitization of algebraic curves and surfaces. Geom. Comput. 11, 126–155.

Muraleedharan, Lakshmi Priya, Kannan, Shyam Sundar, Muthuganapathy, Ramanathan, 2019. Autoencoder-based part clustering for part-in-whole retrieval of cad models. Comput. Graph. 81, 41–51.

Niu, Z., Martin, R.R., Langbein, F., Sabin, M.A., 2015a. Rapidly finding CAD features using database optimization. Comput. Aided Des.

Niu, Z., Martin, R.R., Sabin, M.A., Langbein, F., Bucklow, H., 2015b. Applying database optimization technologies to feature recognition in CAD. Comput-Aided Des. Appl.

Raffo, A., Barrowclough, O.J.D., Muntingh, G., 2019. georgmuntingh/ImplicitClustering. https://github.com/georgmuntingh/ImplicitClustering.

Rossignac, J.R., 1987. Constraints in constructive solid geometry. In: Proceedings of the 1986 Workshop on Interactive 3D Graphics. I3D '86. ACM, New York, NY, USA, pp. 93–110.

Shu, Z., Qi, C., Xin, S., Hu, C., Wang, L., Liu, L., 2016. Unsupervised 3D shape segmentation and co-segmentation via deep learning. Comput. Aided Geom. Des. 43, 39–52.

Skytt, V., Dokken, T., 2016. Models for isogeometric analysis from CAD. In: IsoGeometric Analysis: A New Paradigm in the Numerical Approximation of PDEs. Springer, pp. 71–86.

Song, M., Sun, Z., Liu, K., Lang, X., 2015. Iterative 3D shape classification by online metric learning. Comput. Aided Geom. Des. 35–36, 192–205.

Várady, T., Martin, R.R., Cox, J., 1997. Reverse engineering of geometric models—an introduction. Comput. Aided Des. 29 (4), 255–268.

Xia, Q., Li, S., Qin, H., Hao, A., 2016. Automatic extraction of generic focal features on 3D shapes via random forest regression analysis of geodesics-in-heat. Comput. Aided Geom. Des. 49, 31–43.

Xiao, Dong, Lin, Hongwei, Xian, Chuhua, Gao, Shuming, 2011. CAD mesh model segmentation by clustering. In: Shape Modeling International (SMI) Conference 2011. Comput. Graph. 35 (3), 685–691.