UiO : **University of Oslo**

Andrea Raffo

Mathematical methods for geometry reconstruction and shape analysis

Thesis submitted for the degree of Philosophiae Doctor

Department of Mathematics
Faculty of Mathematics and Natural Sciences

Department of Mathematics and Cybernetics
Stiftelsen SINTEF

2022

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 2497

ISSN 1501-7710

Cover: Hanne Baadsgaard Utigard.
Print production: Reprosentralen, University of Oslo.

To mom and dad

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of *Philosophiae Doctor* at the University of Oslo.

The work that comprises this thesis has been performed as part of the EU-project Algebraic Representations in Computer-Aided Design for complEx Shapes (ARCADES) from the European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska-Curie grant agreement No 675789. The ARCADES Network consists of 14 partners including industrial companies, research institutes and universities, based throughout Europe. ARCADES aims at inverting the trend of CAD industry lagging behind mathematical breakthroughs, by exploiting cutting-edge research in applied mathematics and algorithm design. Examples of applications ARCADES looks at include animation and computer graphics, big data, fabrication and manufacturing and simulation.

The thesis is a collection of six papers, presented in chronological order. The papers are preceded by an introductory chapter that relates them together and provides background information and motivation for the work. The first paper has been written during my stay at the geometry group at SINTEF, under the supervision of Dr. Oliver J. D. Barrowclough and Dr. Georg Muntingh. Two papers have been written together with Dr. Silvia Biasotti, during a three-month research stay at the Department of Applied Mathematics and Information Technologies of the National Research Council of Italy (CNR-IMATI), located in Genoa, Italy. The remaining three papers have been finalized while working as a research fellow at CNR-IMATI, and include authors from different academic areas.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to the main source of funding for this project — the EU Initial Training Network ARCADES. I would like to thank SINTEF for the role played in my personal growth: both professionally and personally speaking. I am grateful to all the other institutions I had the pleasure to spend time at: ATHENA Research and Innovation Center (Athens, Greece), RISC-Software GmbH (Hagenberg im Mühlkreis, Austria) and National Research Council of Italy (Genoa, Italy).

My appreciation goes to my main supervisor Prof. Michael Floater, who has been influential in shaping this thesis. I would like to express my gratitude to my supervisors at SINTEF, Dr. Oliver J. D. Barrowclough, Dr. Heidi E. I. Dahl, Dr. Tor Dokken and Dr. Georg Muntingh, for our instructive conversations, as well as for always encouraging me to pursue my own academic interests

rather than to stick to theirs. Thanks also go to Prof. Ioannis Emiris and Dr. Alexander Leutgeb, who guided me well during my secondments in Greece and Austria. I will be forever thankful to Dr. Silvia Biasotti and Dr. Bianca Falcidieno: our stimulating discussions, together with your constructive criticism, have enhanced my passion for applied mathematics and motivated me to do my best. A big thanks to my Master's thesis supervisor at the University of Bergen, Prof. Antonella Zanna Munthe-Kaas: your support — both academic and human — will never stop to influence my present.

Many other people deserve my gratitude as they contributed — directly and indirectly — to this thesis. The professors at the University of Genoa (Italy), especially Prof. Mauro Beltrametti and Prof. Claudio Estatico. At the University of Oslo, Prof. Riccardo De Bin, Prof. Stine Camilla Johansen, Prof. Martin Reimers and Prof. Arne B. Sletsjøe, whose excellent teaching provided much inspiration for my work. My friends in Austria, Greece, Italy and Norway, for these amazing years throughout Europe. Mrs. Eline Blom Hoen for all the support and inspiration.

Last but not least, a special and heartfelt THANKS goes to my parents Antonio and Daniela, and to my grandparents Alfredo, Franca, Gigio and Ilda.

Andrea Raffo
Oslo, March 2022

Marie Skłodowska-Curie Actions

This thesis was funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 675789.

List of Papers

Paper I

Andrea Raffo, Oliver J. D. Barrowclough and Georg Muntingh. “Reverse engineering of CAD models via clustering and approximate implicitization”. In: *Computer Aided Geometric Design.* Vol. 80, (2020), DOI: 10.1016/j.cagd.2020.101876.

Paper II

Andrea Raffo and Silvia Biasotti. “Weighted quasi-interpolant spline approximations: Properties and applications”. In: *Numerical Algorithms.* Vol. 87, (2021), pp. 819–847. DOI: 10.1007/s11075-020-00989-4.

Paper III

Andrea Raffo and Silvia Biasotti. “Data-driven quasi-interpolant spline surfaces for point cloud approximation”. In: *Computers & Graphics.* Vol. 89, (2020), pp. 144-155. DOI: 10.1016/j.cag.2020.05.004.

Paper IV

Chiara Romanengo, Andrea Raffo, Silvia Biasotti and Bianca Falcidieno. “Extraction of geometric primitives from 3D point clouds for CAD applications”. *Manuscript.*

Paper V

Chiara Romanengo, Andrea Raffo, Yifan Qie, Nabil Answer and Bianca Falcidieno. “Fit4CAD: A point cloud benchmark for fitting simple geometric primitives in CAD objects”. In: *Computers & Graphics.* Vol. 102, (2022), pp. 133–143. DOI: 10.1016/j.cag.2021.09.013.

Paper VI

Andrea Raffo, Ulderico Fugacci, Silvia Biasotti, Walter Rocchia, Yonghuai Liu, Ekpo Otu, Reyer Zwiggelaar, David Hunter, Evangelia I. Zacharaki, Eleftheria Psatha, Dimitrios Laskos, Gerasimos Arvanitis, Konstantinos Moustakas, Tunde Aderinwale, Charles Christoffer, Woong-Hee Shin, Daisuke Kihara, Andrea Giachetti, Huu-Nghia Nguyen, Tuan-Duy Nguyen, Vinh-Thuyen Nguyen-Truong, Danh Le-Thanh, Hai-Dang Nguyen and Minh-Triet Tran “SHREC 2021: Retrieval and classification of protein surfaces equipped with physical and chemical properties”. In: *Computers & Graphics*. Vol. 99, (2021), pp. 1–21. DOI: 110.1016/j.cag.2021.06.010.

Contents

Chapter 1

Introduction

It has often been said we are living in the data age. Every single day, staggering amounts of information are collected, stored, and shared worldwide. A data deluge, as it has been called by the weekly newspaper "The Economist". Yet, much of it remains, totally or partially, unexploited: ironically, "we are drowning in information, but we starved from knowledge" [37]. In this sea of uncertainty, geometry represents a very natural tool for representing and analysing massive amounts of data, in countless applications (e.g., artificial intelligence, machine learning, big data and knowledge discovery) and for a number of fields (e.g., medicine, chemistry, computer science, physics and geoscience).

In many practical situations, information consists of a set of sample points describing some phenomena. A natural question deals with *geometry reconstruction*, i.e., the recovering of the geometry and the topology of such data points. A lot of research was done in this direction, under the umbrella term of approximation theory. However, most of these efforts were traditionally devoted to the case of sampled smooth curves and surfaces, thus curbing the usage of such approaches to the case of sufficiently clean points. Unfortunately, in most real-world scenarios, data is indeed affected by noise and potentially other imperfections (e.g., in the case of geospatial data). Another critical point is the need to generalize well on unseen data, that is, to capture the underlying trend of the input points to be able to make accurate predictions (e.g., when dealing with rainfall fields measurements).

Another issue concerns the automatic analysis and comparison of data. An example of application, particularly relevant in these days, is the identification of potential binding sites on molecules. The family of techniques exploiting geometry for this purpose is often referred to as *shape analysis*. Compared to the search of accurate representations that is typical of geometry reconstruction, the main focus of shape analysis is to *describe* shapes; as pointed out in [32]:

> "*an object representation contains enough information to reconstruct (an approximation to) the object, while a description only contains enough information to identify an object as a member of some class*".

While the representation of an object is usually more complete than a description, it does not necessarily disclose any high-level information useful to discriminate it from other shapes.

This thesis addresses a range of problems in geometry reconstruction and shape analysis. All the proposed methods are motivated in the light of real-world applications, ranging from the need to identify geometric relationships in mechanical engineering to the recognition of proteins from an ensemble of geometries in structural biology. Despite geometry being its undisputed star, this thesis combines data, concepts, perspectives and techniques from different scientific disciplines, making it inherently interdisciplinary. All the papers here presented include simulation on real data and, in most cases, consider benchmarking to unveil strengths and weakness of different approaches.

The thesis is organized as follows. The remainder of this chapter provides some relevant background information to the problems of geometry reconstruction (Section 1.1), shape analysis (Section 1.2) and benchmarking (Section 1.3). Chapter 2 provides a summary of the papers included in this thesis, in the light of such problems. In these two chapters, things will be often kept conceptual, to allow readers from different backgrounds to grasp the type of problems this thesis deals with; the technical details, as well as the complete references, are left to the papers.

1.1 Geometry reconstruction

Reconstructing geometric shapes from sets of sampled points has blossomed during the last decades, partly because of the massive proliferations of commodity real-time scanners such as the Microsoft Kinect. Despite its apparent simplicity, geometry reconstruction is a multifaceted problem, because of the many variables involved and the difficulty of inferring connectivity information from a set of disconnected points.

Traditionally, the attention was mostly turned towards the approximation of sufficiently regular functions with simpler functions (e.g., polynomials or piecewise polynomials). However, acquisition methods tend to produce data affected by a variety of properties and imperfections, such as nonuniform or low sampling density, noise, outliers, misalignment and missing data. The presence of point cloud artefacts poses significant challenges in approximation theory. One example of a situation where data imperfection cannot simply be ignored is the patient-specific, computer-aided estimation of physiological parameters based on biomedical images, such as in [51].

Another challenge arises when the purpose is not limited to the minimization of some loss functions over the input points but, instead, the shape has to provide good predictions on independent data. To put it differently, the inferred shape must be descriptive of the whole phenomenon under study and not only of the samples that have been acquired. Modelling rainfall fields in environmental applications [40] is just an example of areas where geometry and inference go hand in hand.

Besides their quality, input points can be structured or unstructured, that is, can be accompanied by connectivity information or not. The use of dense datasets helps recovering such information that, when missing, can lead to unwanted holes or other topological errors; however, this comes at the cost of much higher computational times and a larger memory usage, which is detrimental in most applications. Moreover, it is not always feasible to increase the sampling density at will (e.g., when modelling biological macromolecules and cells [55]).

This section is devoted to the introduction of three families of methods, each of which is a mainstay to address some specific problems dealing with geometry reconstruction:

- Approximate implicitization (see Section 1.1.1) was originally introduced in the Computer Aided Geometric Design (CAGD) community in 1997 as a way to pass from rational parametric representations to implicit ones [19]. The method was later generalized to approximate implicit representations starting from an input point cloud: experiments using industrial data were proposed in 2006 [47], while the theory behind the cases where the point cloud is sampled from a rational curve/surface was proposed in 2012 [2]. Paper I applies this technique to extract geometric information out of CAD models.

- Quasi-interpolation (see Section 1.1.2) is a popular tool in approximation theory and its applications, because of its inherent computational efficiency: function approximation is performed without requiring the solution of any large-scale system of equations but, instead, makes use of direct formulas using discrete function values based on the local properties of the basis functions. Papers II and III introduce a quasi-interpolation scheme to compute local approximations, in the form of explicit representations, of perturbed point clouds.

- Regression analysis (see Section 1.1.3) combines different branches of mathematics, such as geometry and measure theory, to infer the relationships between a set of given variables and predict one or more output variables. It is popular among different communities, including statistics, probability theory, data science and artificial intelligence. Paper II interprets the quasi-interpolation scheme there introduced in light of nonparametric regression.

1.1.1 Approximate implicitization

Despite the variety of alternative shape representations, implicit and parametric representations still play a fundamental role in modelling geometric objects. The reasons are not only historic, but rather rely on the complementarity of their properties. For example, it is easy to determine whether a point is on (or inside or outside) an implicit curve or surface, but this determination is generally not as painless when dealing with parametric representations. On the other hand, parametric representations are best suited for point generation. Having both representations available makes it possible to address a wide range of problems (e.g., intersection problems).

Several methods to pass from one representation to the other have been introduced over the years, with a history that dates back at least as far as the early part of the nineteenth century. In this section we focus on the computation of implicit representations from parametric ones, a process known under the name of *implicitization*.

In elimination theory (see, for example, [28]), the problem of implicitization is solved by the elimination of the parameter variables, by considering a systematic method based on linear algebra. The result is a (hyper)surface represented, in general, by a single polynomial. One approach to solving the elimination problem makes use of resultants [12, 52], which can be thought of as a mathematical tool to verify whether a set of polynomials share any common roots; resultants can be computed, for example, by exploiting Sylvester and Bezóut matrices [16]. A more recent method is related to Gröbner bases, introduced by Bruno Buchberger in 1965 [10]: a Gröbner basis is a set of multivariate polynomials that has desirable algorithmic properties, which can be used in many computational tasks; example of computations dealing with this concept include elimination theory and computing cohomology [22]. Despite their mathematical elegance, the practical use of these algorithms is limited by several computational challenges, such as:

- *Additional solutions.* Implicit polynomials obtained by resultant computation happen to contain additional factors that are not part of the implicit equation of the (hyper)surface itself. Numerically, this undesired effect may be unsolvable: small perturbations in the coefficients of a reducible polynomial can indeed render it irreducible.

- *High polynomial degrees.* Exact implicit representations of rational parametric forms can have undesirably high algebraic degrees, making computations expensive and contributing to numerical instability.

- *Self-intersections and unwanted branches.* This problem is not directly connected to the use of a specific technique of implicitization, but to the use of exact implicit forms themselves.

To address these problems, numerical methods were introduced in the literature. Approximate implicitization [18, 19] opened up the possibility to approximate a parametrically represented manifold by an algebraic hypersurface of chosen total degree $m \geq 1$. A feature of the method is that, when the total degree is high enough and exact precision arithmetic is used, the method provides exact implicitizations. Let:

- $\mathbf{p} : \Omega \subset \mathbb{R}^{n-1} \to \mathbb{R}^n$ be a rational parametric hypersurface of (multi)degree $\mathbf{n}$, where Ω is the Cartesian product of closed intervals.

- $q_{\mathbf{b}}(\mathbf{x}) = \mathbf{q}(\mathbf{x})^{\mathrm{T}}\mathbf{b}$ be a family of n-variate implicit polynomials of total degree m, where
$$\mathbf{q}(\mathbf{x}) := (q_1(\mathbf{x}), \cdots, q_N(\mathbf{x}))^{\mathrm{T}}$$
is a vector containing a set of basis functions for $q_{\mathbf{b}}(\mathbf{x})$, being
$$N := \binom{m+n}{n}$$
their number, and where $\mathbf{b}$ is the corresponding vector of coefficients.

The composition $q_{\mathbf{b}}(\mathbf{p}(\mathbf{s}))$ can be factorized as

$$q_{\mathbf{b}}(\mathbf{p}(\mathbf{s})) = (\mathbf{D}\mathbf{b})^{\mathrm{T}}\boldsymbol{\alpha}(\mathbf{s}), \tag{1.1}$$

where:

- $\boldsymbol{\alpha}(\mathbf{s})$ is a basis for the polynomials of (multi)degree at most $m\mathbf{n}$.
- $\mathbf{D}$ is a matrix that expresses such a composition in term of the basis $\boldsymbol{\alpha}(\mathbf{s})$ and the coefficients in $\mathbf{b}$.

Approximate implicitization methods aim at minimizing the *algebraic error* to the parametric hypersurface while keeping the total degree low; in other words, the purpose is to find a polynomial $q_{\bar{\mathbf{b}}}$ which minimizes (in some sense) $|q_{\mathbf{b}}(\mathbf{p}(\mathbf{s}))|$ for $\mathbf{s} \in \Omega$. The choice of considering the algebraic error rather that the geometric error lies in the fact that minimizing the latter is a computationally intractable problem, while the former can provide a good approximation away from singularities [19]. From the factorization given by Equation 1.1, it follows that

$$\begin{aligned}\max_{\mathbf{s}\in\Omega} |q_{\mathbf{b}}(\mathbf{p}(\mathbf{s}))| = \max_{\mathbf{s}\in\Omega} |(\mathbf{D}\mathbf{b})^{\mathrm{T}}\boldsymbol{\alpha}(\mathbf{s})| \leq \\ \leq ||\mathbf{D}\mathbf{b}||_2 \max_{\mathbf{s}\in\Omega} ||\boldsymbol{\alpha}(\mathbf{s})||_2 \leq ||\mathbf{D}\mathbf{b}||_2,\end{aligned} \tag{1.2}$$

where the last inequality holds when considering a basis $\boldsymbol{\alpha}(\mathbf{s})$ that is a non-negative partition of unity within the domain of interest. This inequality tells us that the choice of coefficients $\mathbf{b}$ controls the quality of the final approximation. Notice that if we can find a coefficient vector $\mathbf{b} \neq \mathbf{0}$ that satisfies $\mathbf{D}\mathbf{b} = 0$, then we have found an exact implicitization of $\mathbf{p}(\mathbf{s})$. We apply singular value decomposition on $\mathbf{D}$ and select the vector with unit normal $\mathbf{b}_{\min}$ corresponding to the smallest singular value $\sigma_{\min}$; from Equation 1.2, it follows that

$$|q_{\mathbf{b}_{\min}}(\mathbf{p}(\mathbf{s}))| \leq \sigma_{\min}, \tag{1.3}$$

i.e., the smallest singular value bounds the pointwise error of approximation.

One of the simplest and fastest numerical implementation of approximate implementation is based on a least squares approximation of a point cloud sampled from the (parametric) hypersurface, as introduced in [2]. Given a basis $\Pi := \{\pi_1, \cdots, \pi_M\}$ for the space of n-variate implicit polynomials of total degree at most m and a finite sequence of points $\{P_1, \cdots, P_N\} \subset \mathbb{R}^n$:

- Form the collocation matrix
$$\mathbf{D} := (\pi_i(P_j))_{\substack{i=1,\cdots,M \\ j=1,\cdots,N}}.$$
- Compute the singular value decomposition $\mathbf{D} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$.

- The vector $\mathbf{b}_{\min} = (b_1^{\min}, \cdots, b_M^{\min})^{\mathrm{T}}$ corresponding to the smallest singular value $\sigma_{\min}$ of $\mathbf{D}$ identifies the approximate implicit representation

$$q := \sum_i b_i^{\min} \cdot \pi_i.$$

By further assuming that the points are sampled from a parametric representation, i.e., $P_i = \mathbf{p}(\mathbf{s}_i)$ for any i, this is equivalent to considering the $\boldsymbol{\alpha}$-basis to be the Lagrange basis at the given nodes; note that the Lagrange basis provides a partition of unity, but it is not non-negative on the region of interest. Analogously to Equation 1.3, we can bound the algebraic error as

$$|q_{\mathbf{b}_{\min}}(\mathbf{p}(\mathbf{s}))| \leq \Lambda(\boldsymbol{\alpha})\sigma_{\min},$$

where $\Lambda(\boldsymbol{\alpha})$ is the Lebesgue constant from interpolation theory defined as $\Lambda(\boldsymbol{\alpha}) := \max_{\mathbf{s}\in\Omega} ||\boldsymbol{\alpha}(\mathbf{s})||_1$.

1.1.2 Quasi-interpolation

Quasi-interpolation is very useful in approximation theory and its applications, as it allows to compute approximations without the need to solve any linear system of equations [13, 45]. Quasi-interpolants (QIs) can be defined as linear operators of the form

$$Qf := \sum_{j\in J} \mu_j(f) g_j, \tag{1.4}$$

where $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ is a function being approximated, $\mu_j(f)$ are linear functionals, $g_j : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ are functions at our disposal. The coefficients $\mu_j(f)$ are, in general, one of the following types: linear combinations of given values of the function f to be approximated (*discrete type*); linear combinations of values of derivatives of f, of order at most d (*differential type*); linear combinations of weighted mean values of f (*integral type*). Equation 1.4 can be interpreted as a “reconstruction” formula: given some input data sampled from the true function f, it creates a tentative reconstruction Qf. In this thesis, we will assume the functions g_j to be (polynomial) spline functions, thus restricting our attention to spline quasi-interpolants; however, for the sake of conciseness, we will omit the word “spline” when not risking a misunderstanding.

The earliest case of quasi-interpolation is often traced back to Bernstein’s approximation, which builds a quasi-interpolation of a univariate continuous function $f : [0, 1] \to \mathbb{R}$ as

$$Q_n f := \sum_{j=0}^{n} f(x_j) g_j,$$

where $x_j = j/n$ and where

$$g_j(x) = \binom{n}{i} x^i (1-x)^{n-i}$$

are the $n+1$ Bernstein polynomials of degree n. The sequence Q_nf is proved to converge uniformly to f [8].

A rather straightforward generalization of this quasi-interpolant, which considers B-splines rather than Bernstein polynomials, is known as the *Variation Diminishing Spline Approximation* (VDSA). The reader who is not familiar with B-splines is referred to Appendix 1.A. Given a $(p+1)$-regular knot vector $\boldsymbol{\tau} = [\tau_1, \cdots, \tau_{n+p+1}]$ with $\tau_1 = a$ and $\tau_{n+p+1} = b$, VDSA approximates any continuous function $f : [a,b] \subset \mathbb{R} \to \mathbb{R}$ by the expression

$$Vf := \sum_{j=1}^{n} f(\tau_j^*) g_j,$$

where $\tau_j^* := (\tau_{j+1} + \cdots, \tau_{j+p})/p$ are the *knot averages* and where $g_j := B[\boldsymbol{\tau}^{(j)}]$ is the B-spline of degree p over the (local) knot vector $\boldsymbol{\tau}^j := [\tau_j, \cdots, \tau_{j+p+1}]$. When all knots occur $p+1$ times, then Q_nf interpolates the n points sampled on f. Besides its simplicity, the VDSA owes its popularity to its shape preserving properties:

- *Preservation of bounds on a function.* The VDSA is bounded by the minimum and maximum values of f, i.e.,

$$\min_{[a,b]} f(x) \leq Vf \leq \max_{[a,b]} f(x).$$

- *Preservation of monotonicity.* If f is increasing (resp. decreasing) then Vf is also increasing (resp. decreasing).

- *Preservation of convexity.* If f is convex (resp. concave) then Vf is also convex (resp. concave).

It should be noted that these shape preserving properties are a direct consequence of B-spline properties and of the relationship between a spline and its control polygon (see, for example, [34]). Another point to note is that VDSA is a local method, i.e., the approximation at a point x only depends on the function samples near x.

Several quasi-interpolants have been introduced in the last decades, for example to handle local refinements for spline spaces. However, most of the theory has developed under the assumption that the data being modelled are sampled from a sufficiently regular function, thus in the absence of any point cloud artefacts.

1.1.3 Regression analysis

The term *regression analysis* refers to a set of mathematical methods for quantifying the dependencies between a set of *explanatory variables*, also known as *predictors*, and one or more *response variables*. For the sake of simplicity, we will restrict to the case of *simple regression*, i.e., regression problems where there is only one response variable. Most (simple) regression methods are written in the additive form

$$Y = f(X_1, \cdots, X_p; \beta_1, \cdots, \beta_q) + \varepsilon, \tag{1.5}$$

where:

- Y names the response variable.
- $X_1, \cdots, X_p$ denote the explanatory variables, which are commonly assumed to be deterministic.
- $\beta_1, \cdots, \beta_q$ are the *unknown parameters* to be estimated.
- ε is the *random error*, which is not directly observed in data.

As a further restriction, we will assume that the unknown parameters have some hidden true values, thus adopting the frequentist approach; the Bayesian description of probability would have assumed the unknown parameters to be random variables themselves, turning the search for an accurate approximation of the true parameters into the search for an accurate approximation of the posterior probability.

1.1.3.1 Parametric vs nonparametric regression

Regression models are often dychotomized into *parametric regression methods* versus *nonparametric regression methods*. Parametric regression assumes that any observation $Y = y$ is the realization of an unknown but predetermined probability distribution (e.g., the normal distribution with unknown expectation and variance); the goal is to estimate such a distribution by computing an estimation of its unknown parameters. Conversely, nonparametric regression only assumes that such a density exists, without choosing any specific family in advance.

A classical example of parametric regression is given by linear models, which remain among the mainstays of modern scientific research despite their simplicity. Univariate linear regression models assume that the response variable can be written as

$$Y = \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \varepsilon.$$

For n independent observations $\mathbf{Y} = (Y_1, \cdots, Y_n)^{\mathrm{T}}$, an alternative matrix formulation is

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where:

- $\mathbf{X} = (x_{i,j})$ denotes the $n \times (p+1)$ *model matrix*, with $x_{i,j}$ the value of explanatory variable X_j for observation i and with X_0 set to 1.
- $\boldsymbol{\beta}$ denotes the $(p+1) \times 1$ *parameter vector*.

- $\mathbf{X}\boldsymbol{\beta}$ is the *linear predictor*, geometrically consisting of a (hyper)plane.
- $\boldsymbol{\varepsilon}$ denotes the $n \times 1$ *error vector.*

Having chosen the model, the standard approach to obtain parameter estimates $\hat{\boldsymbol{\beta}}$ and fitted values $\hat{\mathbf{y}} := \mathbf{X}\hat{\boldsymbol{\beta}}$ uses the least squares loss; such $\hat{\boldsymbol{\beta}}$ is known as the *ordinary least squares estimator* (OLS). While estimating the coefficients merely requires choosing a loss function, drawing inferences is commonly handled by assuming that predictors are deterministic and that Y follows a normal distribution. The reasons behind the popularity of linear regression models are multifaceted: in addition to their easy interpretation and computability, they have proven to work well in many situations. Moreover, the OLS estimator is shown to be the Best (i.e., minimum-variance) Linear Unbiased Estimator (BLUE, also known as Gauss-Markov theorem; see, for example, [27]). Several extensions to (parametric) linear regression have been proposed since its introduction. One of the most famous is that of *generalized linear models* (GLMs, [39]). GLMs allow the response variable to have a distribution within the exponential family, a large class of probability distributions that includes normal, binomial, Poisson and gamma distributions, among others; moreover, they allow the response to be a nonlinear function of $\mathbf{X}\boldsymbol{\beta}$.

Parametric models usually give more information but, at the same time, can also lead to significantly biased conclusions if the wrong distribution is chosen (e.g., in hypothesis testing). On the contrary, nonparametric statistics require fewer assumptions about the data and, consequently, can better face those situations where the true distribution is unknown or cannot easily be approximated by a probability distribution. Nonparametric methods include kernel smoothing and local polynomials [23], regression and smoothing splines [15, 54], reproducing kernel Hilbert spaces [6] and wavelets [38]. We conclude this section by briefly focusing on kernel methods, as they are the most relevant class of approaches for this thesis.

Kernel methods can be seen as a generalization of the k-*nearest neighbour average* which, in the case of one predictor, is defined as

$$\hat{f}(x) = \frac{1}{k} \sum_{(x_i, y_i) \in N_k(x)} y_i,$$

where $N_k(x)$ denotes the set of the k points whose abscissa is the closest to x in the L_2 distance, among a set of input points. Being the k-nearest neighbor average discontinuous (and piecewise constant), it is not well-suited for applications where the geometry of the approximation is relevant; on the other hand, what makes it special is it being an estimate of the conditional expectation $\mathbb{E}(Y|X = x)$ which require little training. Instead of giving all the points in the neighbourhood equal weight, the *Nadaraya-Watson kernel-weighted average* assigns weights that die off smoothly with distance from the target point; its expression is given by

$$\hat{f}(x) := \frac{\sum_{i=1}^{N} K_\lambda(x, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x, x_i)},$$

where

$$K_\lambda(x, x_i) = D\left(\frac{|x - x_i|}{h_\lambda(x)}\right) \tag{1.6}$$

is a predefined kernel with window width $h_\lambda(x)$, indexed by some unknown parameter λ. In practice, the width of the local neighborhood has to be determined by fixing λ; this is performed by trading-off bias and variance, as will be detailed in Section 1.1.3.2. Examples of popular kernels, determined by the functions in Table 1.1, are shown in Figure 1.1.

Table 1.1: Examples of common kernels. Here, $\mathbb{1}$ denotes the indicator function.

Kernel	**Function D**
Rectangular	$D(u) = \frac{1}{2}\mathbb{1}_{[-1,1]}(u)$
Epanechnikov	$D(u) = \frac{3}{4}(1-u^2)\mathbb{1}_{[-1,1]}(u)$
Tri-cube	$D(u) = (1-\lvert u\rvert^3)^3\mathbb{1}_{[-1,1]}(u)$
Gaussian	$D(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right)$

1.1.3.2 Model assessment and selection

As regression analysis is interested in maximizing the prediction capability on independent test data rather than just minimizing a functional over some input points, the assessment of the generalization performance is extremely important. Indeed, it guides the choice of the regression method or model, in addition to quantifying the quality of the model that is ultimately chosen.

Let us consider the general additive formulation given in Equation 1.5, again considering only one predictor for readibility and visual appeal. We further assume that $\mathbb{E}[\varepsilon] = 0$ and $\mathrm{Var}(\varepsilon) = \sigma_\varepsilon^2$. Given some input (training) data $\mathcal{D}$, the (squared) expected prediction error (EPE) of the regression fit $\hat{f}$ at an input point $X = x$, i.e.,

$$\mathrm{EPE}\left(Y, \hat{f}(x)\right) := \mathbb{E}_{Y|X,\mathcal{D}}\left[(Y - \hat{f}(X))^2 \middle| X = x\right],$$

can be decomposed as the sum of three quantities

$$\mathrm{EPE}\left(Y, \hat{f}(x)\right) = \mathrm{Bias}^2[\hat{f}(x)] + \mathrm{Var}[\hat{f}(x)] + \sigma_\varepsilon^2,$$

where:

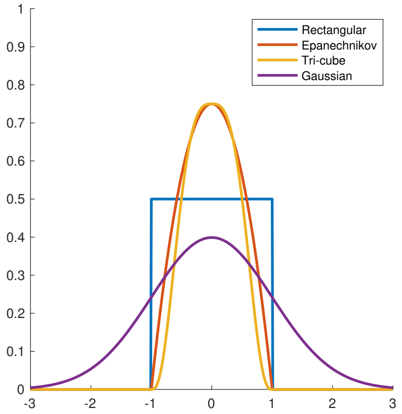


Figure 1.1: Examples of common kernels, determined by the functions given in Table 1.1. Each kernel is here centered around the origin, and calibrated to integrate to 1.

- The *bias*

  $$\mathrm{Bias}^2[\hat{f}(x)] := \left(\mathbb{E}_{\mathcal{D}}\left[\hat{f}(x)\right] - f(x)\right)^2$$

  measures the error coming from erroneous assumptions made by the regression method to make the target function f easier to learn (e.g., when the model does not incorporate all the necessary variables, or when the form of the relationship is too simple). Models with high bias tend to oversimplify f, commonly leading to high error on both training and test data.

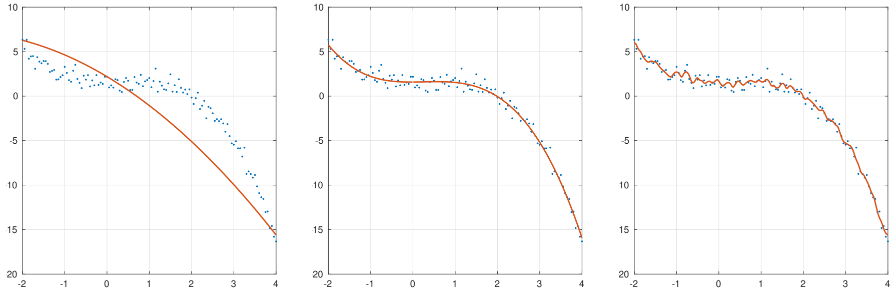
- The *variance*

  $$\mathrm{Var}[\hat{f}(x)] := \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(x) - \mathbb{E}_{\mathcal{D}}\left[\hat{f}(x)\right]\right)^2\right]$$

  quantifies the model sensitivity to small fluctuation in the input data. A regression method with high variance has paid a lot of attention to the input but does not generalize well on data which has not been seen before.

- The *irreducible error* σ_ε^2, also called *Bayes error*, is the variance of the target around its true mean, and cannot be avoided no matter how well we make the estimation.

In statistical learning, the concepts of bias and variance are often accompanied by those of *underfitting* and *overfitting*. Underfitting refers to the scenario where the fitted model is unable to capture the underlying pattern of the data; underfitted models have usually high bias and low variance; underfitting occurs, for example, when trying to model a very complex pattern with a linear model. Conversely, overfitting happens when the fitted model relies too much on the input, thus capturing the noise along with the underlying trend of the training data; overfitted model usually have low bias but high variance; overfitting can originate from very complex models, such as decision trees. Figure 1.2 illustrates these issues.



underfitting

good balance

overfitting

Figure 1.2: Generalization performance. In case of underfitting, the model is unable to capture the underlying pattern of the data, thus resulting in low performance on input data as well as on unseen data. A model suffering from overfitting does not succeed in capturing the trend of the input while avoiding point cloud artefacts such as noise or outliers; this results in a poor generalization capability. An optimal balance between bias and variance allows to alleviate these problems.

1.2 Shape analysis

The description and analysis of the geometrical properties of a given object is of tremendous importance in applied fields. Examples of applications include the processing and comparison of mechanical parts in Computer-Aided Design and Manufacturing [33], the identification of protein binding regions in biochemistry [57], the retrieval and classification of archeological finds in cultural heritage [44], and the identification of anomalies for medical purposes [24].

Unlike for geometry reconstruction, a key point to note when dealing with shape analysis is that the analysis of shapes in 2D is radically different from that of shapes in 3D. For example, 2D images can be easily thought as a set of pixels, although the perspective projection might cause gaps due to occlusions or to lighting conditions. On the other hand, 3D geometric models are much more complex to handle, but provide a complete representation of the object shape: as a result, they do not contain projections, reflections, shadows or occlusions.

The core problems in 3D shape analysis may be classified in different ways, according to various criteria. We here adopt the categorization proposed in [9, 30].

- **Segmentation**. In shape analysis, the term refers to the process of partitioning an input object (e.g., a point cloud, a triangle mesh) into multiple and meaningful segments (e.g., subsets of points, sub-meshes). Segmentation often assists other tasks, such as parametrization, texture mapping, compression and shape matching.

- **Semantic labeling**. Also known as *annotation*, *semantic labeling* is the task of assigning labels to a model or parts of it. For example, by studying the geometry and topology of carpal bones in the human body, one can assign tags such as "scaphoid","lunate", "triquetrum", "pisiform", "trapezium","trapezoid", "capitate" and "hamate"; in this case, annotation can support diagnosis and follow-up analysis of musculoskeletal pathologies.

- **Feature detection and shape description**. A *feature* is, broadly speaking, any piece of measurable information about a phenomenon. Among the desirable properties of a feature there are: conciseness, identifiability, invariance to rigid transformations, noise resistance, occlusion invariance. Moreover, when considering more than a single feature, statistical independence is usually required. Feature detection can be considered a fundamental step in the definition of concise yet informative signatures of shape models, also known as *shape descriptors* or *descriptions*, thus enabling subsequent shape analysis problems such as matching, retrieval and classification.

- **Registration**. The identification of meaningful correspondences between discrete sets of points on different models is a problem that arises in many domains of science. Examples of applications include manufacturing (e.g., finding possible defects on a product given a model) and medicine (e.g., finding correspondences between 3D MRI scans of the same person or of different people).

- **Matching**. Pure mathematicians typically define shape in term of class equivalence under a group of transformations. Although theoretically sound, this view turns out being incomplete when dealing with applications. Many contexts, such as computational chemistry, require in fact more than just knowing when two shapes are exactly the same. *Matching* deals with the definition and evaluation of shape similarity; thus, similarity measures are at the core of every shape matching algorithm.

- **Retrieval, classification and clustering**. In shape analysis, *retrieval* is the problem of identifying all models in a set that are similar to a query object, on the basis of some properties. By *classification* we instead mean the problem of partitioning an input set into a number of preset classes, possibly to determine to which of these classes a new object belongs. When classes are not predefined, the process of grouping input instances takes the name of *clustering.*

The remainder of this section deepens the discussion on shape analysis, by focusing on two of the just-introduced applications:

- Shape description via the Hough transform technique, see Section 1.2.1. Papers IV and V apply the algebraic Hough transform to a twofold problem: the segmentation of CAD point clouds, and the definition of shape descriptors for further aggregating such segments on the basis of different correlation queries.

- Retrieval, classification and clustering, see Section 1.2.2. Paper VI deals with the retrieval of classification of protein surfaces on the basis of their geometry and of some physicochemical properties. Papers IV and V focus on the problem of point classification when segmenting CAD point clouds. Finally, papers I and IV apply hierarchical clustering to infer geometric information from CAD models and point clouds.

1.2.1 Shape description via the Hough transform technique

Introduced in 1962 by P. V. C. Hough in the form of a patent [26], the Hough transform (or transformation) is one of the most known feature extraction techniques in image analysis, computer vision, and digital image processing.

The Hough transform (HT) was originally presented for detecting and plotting straight lines, corresponding to the tracks of subatomic particles in bubble chamber photographs. The idea is to turn the problem of line detection in the image space into a voting procedure in the parameter space. More precisely, given a set of points $(x_1, y_1), \cdots, (x_N, y_N)$ sampled from a straight line $y = \bar{a}x + \bar{b}$, the lines $y_i = ax_i + b$ in the parameter space (a, b) all intersect exactly in one point, which uniquely identifies the original straight line in the image space. The discretization of the parameter space into small cells allows to count the number of lines passing through a specific cell; the slope-intercept pair characterizing the input point cloud corresponds to the pair identified by the most voted cell.

Since its inception, the definition of the Hough transform has been constantly modified to be able to recognize other shapes and, just as importantly, to do it in more computationally affordable ways [36]. To eliminate the issue of the unboundedness of the parameter space (a, b), Duda and Hart [20] proposed the use of the *Hesse normal form* of straight lines. In the same paper, they also extended the method to handle circles and ellipses in a 3D or 4D parameter space, respectively. In the generalized Hough transform [1], pre-computed look-up tables are used to recognize arbitrary profiles in images. Several authors have developed probabilistic approaches to speed up the HT by choosing a subset of data points, e.g., with fuzzy and Bayesian versions of the method [41, 56].

Very recently, formal results on algebraic geometry have contributed to lay the foundations of a general theory which extends the Hough transform to algebraic objects of codimension greater than (or equal to) one [5, 43], as for the case of space curves.

1.2.1.1 A gentle introduction to the algebraic Hough transform

For the sake of brevity and clarity, we will introduce the Hough transform for the simplest case of hypersurfaces in their implicit form. However, the formulation can be generalized to algebraic objects of codimension greater than one, possibly in their parametric form.

Most of the results in this section hold over an infinite integral ring K. For applications, however, it is sufficient to assume $K := \mathbb{R}$.

Let $F(\mathbf{x}, \mathbf{\Lambda}) \in \mathbb{R}[\mathbf{x}, \mathbf{\Lambda}]$ be a real polynomial in the two sets of variables $\mathbf{x} := (x_1, \cdots, x_n)$ and $\mathbf{\Lambda} := (\Lambda_1, \cdots, \Lambda_t)$. Let $\mathbb{A}^n_{\mathbf{x}}$ and $\mathbb{A}^t_{\mathbf{\Lambda}}$ be real affine spaces of coordinates $\mathbf{x}$ and $\mathbf{\Lambda}$, which will be referred to as *image space* and *parameter space*, respectively. For every t-tuple of parameters $\boldsymbol{\lambda} := (\lambda_1, \dots, \lambda_t) \in \mathbb{R}^t$, we consider the polynomial $f_{\boldsymbol{\lambda}}(\mathbf{x}) := F(\mathbf{x}, \boldsymbol{\lambda}) \in \mathbb{R}[\mathbf{x}]$ and its locus

$$\Gamma_{\boldsymbol{\lambda}}(F) := \{\mathbf{x} : f_{\boldsymbol{\lambda}}(\mathbf{x}) = 0\} \subset \mathbb{A}^n_{\mathbf{x}},$$

which are commonly assumed to be irreducible hypersurfaces. Similarly, every $\mathbf{x}_P := (x_{1,P}, \cdots, x_{n,P})$ gives rise to the polynomial $f_{\mathbf{x}_P}(\mathbf{\Lambda}) := f(\mathbf{x}_p, \mathbf{\Lambda}) \in \mathbb{R}[\mathbf{\Lambda}]$, where its locus

$$\Gamma_{\mathbf{x}_P}(F) := \{\mathbf{\Lambda} : f_{\mathbf{x}_P}(\mathbf{\Lambda}) = 0\} \subset \mathbb{A}^t_{\mathbf{\Lambda}}$$

is called the *Hough transform of* $\mathbf{x}_P$ *with respect to* F.

Under the assumption that the polynomials $f_{\boldsymbol{\lambda}}(\mathbf{x})$ are irreducible in $\mathbb{R}[\mathbf{x}]$, the following general statements hold true:

1. Let $\boldsymbol{\lambda} \in \mathbb{R}^t$. For any $\mathbf{x}_P \in \Gamma_{\boldsymbol{\lambda}}(F)$, the Hough transform $\Gamma_{\mathbf{x}_P}(F)$ passes through $\boldsymbol{\lambda}$.

2. Let $\boldsymbol{\lambda}$, $\boldsymbol{\lambda}' \in \mathbb{R}^t$ such that $\boldsymbol{\lambda} \neq \boldsymbol{\lambda}'$. Assume that, for any $\mathbf{x}_P \in \Gamma_{\boldsymbol{\lambda}}(F)$, the Hough transform $\Gamma_{\mathbf{x}_P}(F)$ passes through $\boldsymbol{\lambda}'$. Then the two hypersurfaces $\Gamma_{\boldsymbol{\lambda}}(F)$ and $\Gamma_{\boldsymbol{\lambda}'}(F)$ coincide.

3. The following conditions are equivalent:

   - For any hypersurfaces $\Gamma_{\boldsymbol{\lambda}}(F) = \Gamma_{\boldsymbol{\lambda}'}(F)$, it follows that $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$.
   - For any $\boldsymbol{\lambda}$, one has

   $$\bigcap_{\mathbf{x}_P \in \Gamma_{\boldsymbol{\lambda}}(F)} \Gamma_{\mathbf{x}_P}(F) = \{\boldsymbol{\lambda}\}.$$

A family of hypersurfaces $\Gamma_{\mathbf{\Lambda}}(F)$ in $\mathbb{A}^n_{\mathbf{x}}$ that meets the above equivalent conditions is said to be *Hough regular*. The hypersurfaces of a Hough-regular family are uniquely identified by their parameters; in some sense, they behave similarly to straight lines as in the original definition of the Hough transform (see Section 1.2.1), making the identified coefficients characteristic of the shape. The parameters of a Hough regular family can thus be used as shape descriptors: while not necessarily providing the best approximation of the whole input point cloud, the Hough transform can indeed recognize patterns and further allow comparisons on the base of the identified parameters. For example, [35] applies the Hough transform to two problems: (1) the identification, in clinical X-ray tomography (CT) slices, of bone profiles in different skeleton districts such as the lumbar vertebrae and the spinal canal, with the goal to explore bone and spinal marrow pathophysiology; (2) the detection of solar eruption fronts from high resolution images, to monitor the solar activity and study its connection with space weather and invasive phenomena in the geosphere.

Figure 1.3 reports four examples of algebraic plane curves, whose families are particularly popular because of their characteristic shapes:

- *Kepler egg*:
$$(x^2+y^2)^2 - a_1x^3 = 0$$
- *Lemniscate of Bernoulli curve*:
$$(x^2+y^2)^2 - 2a_1(x^2-y^2) = 0$$
- *Lamet curve*:
$$a_2x^m + a_1y^m - a_1^m a_2 = 0$$
- *Citrus curve*:
$$a_1^4a_2^2y^2 + \left(x - \frac{a_1}{2}\right)^3 \left(x + \frac{a_1}{2}\right)^3 = 0$$

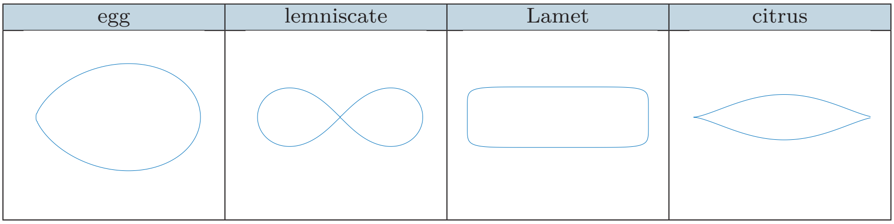


Figure 1.3: A set of algebraic plane curves.

1.2.1.2 The Hough transform in practice

Given an input point cloud $\mathscr{D}$ in the image space $\mathbb{A}^n_{\mathbf{x}}$, representing some profiles of interest, the aim is to detect a hypersurface from a family $\Gamma_{\mathbf{\Lambda}}(F)$ in $\mathbb{A}^n_{\mathbf{x}}$ that best fits the input data.

While in theory it is sufficient to compute the intersection of at least two Hough transforms, such an intersection is generally empty in practice: this occurs, for example, because of numerical reasons or for the presence of point cloud artefacts in the input data (e.g., noise, outliers). To overcome this issue, the following voting strategy is commonly adopted:

1. *Region detection and discretization.* Find a suitable bounded region in the parameter space for the input profile, then discretize it into cells.

2. *Accumulator function and voting procedure.* Initialize an *accumulator matrix*[1], i.e., a matrix such that its entries are in a 1-1 correspondence with the cells of the discretized region. The value of an entry in the accumulator matrix corresponds to the number of Hough transforms, computed over the input data, that intersect the corresponding cell in the discretized region.

3. *Parameter identification.* Once the accumulator matrix has been computed, the parameters corresponding to its maximum entry are taken as representative of the best fit of the input data.

The detection of a region of interest in an unbounded parameter space can be performed in several ways. When the parameters have geometric meaning (e.g., the number of convexities in a m-convexity curve, the radius of a circular cylinder), estimates can be more easily computed (e.g., by considering the minimal bounding box of the input point cloud). Recently, the use of the Moore-Penrose pseudoinverse has been proposed to compute an estimate of the parameters when the polynomial $F(\mathbf{x}, \mathbf{\Lambda})$ is linear with respect to the set of variables $\mathbf{\Lambda}$ [4].

1.2.1.3 Strengths and limitations

The Hough transform has been widely used as a pattern recognition technique, mostly because of two considerable advantages. Firstly, the adoption of a voting strategy makes it fairly robust to most point clouds artefacts (i.e., low or uneven sampling density, noise, outliers, misalignment and missing data). Secondly, when the Hough-regularity property holds, it naturally provides geometric descriptors that can be used for shape comparison.

On the other hand, the HT paradigm suffers from the following drawbacks:

- *Lack of dictionaries.* The use of mathematical representations for shapes is a guarantee of theoretical strength but, at the same time, it represents one of the Achille's heels. While the literature is relatively rich in examples of plane curves (see, for example, [48]), the case space curves and surfaces is limited. In the case of digital images, a solution to this problem considers a piecewise-defined Hough transform, which computes a sequence of polynomials connected C^0 at cusps, G^1 otherwise [14].

- *Large memory requirement.* The accumulator matrix used in the HT can be interpreted as an unnormalized histogram; the problem of finding a/the peak can thus be seen as the search for a/the maximum likelihood estimate in the density distribution corresponding to such an histogram. As the number of parameters increases, the adoption of tensor-product grids causes

[1] The term *tensor* would be mathematically preferable than that of "matrix". However, the latter is used for historical reasons.

the number of bins and the memory cost to scale exponentially; moreover, the number of Hough transforms required to define a peak will surge, due to the curse of dimensionality [3].

- *Computational complexity.* The increase in the memory requirement is accompanied by that of the computational complexity. Indeed, if we denote by M the number of cells in the discretized parameter space and by S the number of points at which the Hough transform must be computed, then $\mathcal{O}(MS)$ checks are expected.

1.2.2 Retrieval, classification and clustering

Similarly to other types of multimedia information such as text documents or audio files, the increasing availability of acquisition devices has led to an exponential growth of data repositories. To name a few: the Protein Data Bank [7], which contains over 175,000 3D shapes of proteins, nucleic acids, and complex assemblies; the Princeton Shape Database [21], which stores about 36,000 polygonal surface models of everyday objects collected from the World Wide Web; and the National Design Repository [42], which counts tens of thousands of mechanical parts from 3D CAD models. Users, whether experts or beginners, can considerably benefit from search engines that are able to retrieve, classify or cluster the members of an unstructured repository.

To allow such tasks, one should first extract a signature (or shape decription) of each member, for example, by applying the Hough transform technique seen in Section 1.2.1. The aim is to discard all unnecessary information while preserving salient properties, thus enabling a more efficient and effective computational handling of the repository. Obviously, different problems can potentially require different shape descriptions. Then, it is necessary to define some distances or dissimilarities, to compare the descriptions instead of the original members. This is sufficient, in theory, to deal with the retrieval and classification of a given repository. To perform clustering, one should also specify an additional ingredient: how the repository is supposed to be partitioned, starting from the just-computed distances (or dissimilarities).

However, commonly to other types of multimedia data, shape variability can complicate things: when some classes have a larger intra-class shape variability than others, or have very heterogeneous sizes, predictive performance of a retrieval or a classification method is usually negatively affected. Despite the apparent similarity with 2D images, the analysis of 3D models presents additional challenges:

- *Invariance by rigid and nonrigid transformations.* Depending on the final purpose, a method could be required to identify similarities up to rigid (i.e., translations, rotations, reflections and their combinations) and non-rigid (e.g., obtained by bending and stretching) transformations.

- *Geometric and topological noise.* 3D models, especially when obtained by using laser scanning or approximations based on ray casting, are often affected by geometric and topological noise. This makes subsequent processing such as remeshing and parametrization a lot harder. In the example of Figure 1.4, the model exhibits artificial holes which result in a change of the overall topology.

- *Partial data.* In applications such as cultural heritage, it is common to handle models prominently affected by missing data. In these cases, one must be able to gather together 3D models even when consisting of fragments of a larger object.

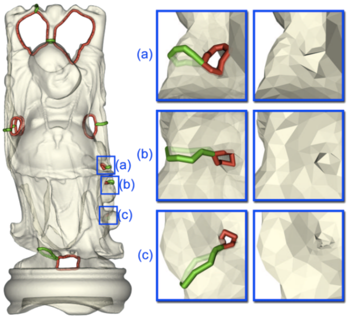


Figure 1.4: Three artificial holes (a-c, left) in a triangle mesh, as a result of topological noise in the scanned point cloud. In (a-c, right), the three holes are filled by making use of topological persistence, see [17]. Pictures courtesy of Professor Tamal Krishna Dey.

1.3 Benchmarking, aka unveiling a method's perks (and flaws)

Benchmarking allows developers and researchers to compare the strengths of different algorithms, as well as to expose their limitations, by considering a set of standard tools and procedures in a standard way. Benchmarks are usually required to be available online, to be free of charge and without copyright issues, so that anyone can use them for scientific publications. A benchmark comes in the form of three ingredients:

- *A dataset.* Generally speaking, it consists of a collection of data referring to some specific task (e.g., classification of digital surfaces with similar geometric reliefs). A number of issues need to be addressed to create a benchmark dataset. As minimum requirements, the collection must be representative of the problem of interest, have a reasonable number of models, and have certain generalization ability to evaluate new methods. In the absence of these characteristics, any final outcome is suspect of biases and wrong conclusions.

- *A ground truth.* Once the data has been acquired, a collection of information which is assumed to be correct and which is relevant for that specific context has to be collected (e.g., the true classifications of the digital surfaces from the previous point). Unfortunately, many practical situations have no automatic way to determine the ground truth. For example, in the segmentation of medical images, the ground truth has to be created manually by domain experts, without the support of automatic software segmentation; indeed, clinical experts usually interpret anatomical structures more precisely than a standardized algorithm or software program.

- *A set of evaluation measures.* Depending on the dataset, the context and the ground truth, the way to evaluate the performance of a method can significantly change. Examples of popular measures in shape retrieval are: Nearest Neighbour (NN), first and second tiers; e-Measures (eM); Discounted Cumulated Gain (DCG) and Normalized Discounted Cumulated Gain (NDCG); precision-recall curves; Average Precision (AP) and mean Average Precision (mAP) [49]. In shape classification, several evaluation measures are derived from confusion matrices, a special kind of contingency table: for example, true positive and negative rates, false positive and negative rates, positive and negative predictive values, accuracy and the Fisher scores [29]. To evaluate the efficiency of an algorithm, it is common to exhibit both the execution times over the whole dataset (or some statistics of it) and its computational complexity; while the former is more intuitive and easily obtainable, the latter can be easily used for comparisons when the algorithms to be compared have not been run on the same machine.

Compared to repositories, such as those mentioned in Section 1.2.2, benchmarks are not simple datasets, but contain all the information to assess the performance of an algorithm targeting some specific problem.

In shape analysis, one of the most established venues for benchmarking new and existing methods is the 3D SHape REtrieval Contest (SHREC, [53]), launched in 2006 in collaboration with the Eurographics Workshop on 3D Object Retrieval (3DOR) and now at its sixteenth edition. Over the years, SHREC has provided benchmarks to evaluate and compare methods in several domains of science, including language technology [11], cultural heritage [50], medicine [25], biology and chemistry [31].

References

[1] Ballard, D. “Generalizing the Hough transform to detect arbitrary shapes”. In: *Pattern Recognition* vol. 13, no. 2 (1981), pp. 111–122.

[2] Barrowclough, O. and Dokken, T. “Approximate implicitization using linear algebra”. In: *Journal of Applied Mathematics* (2012).

[3] Bellman, R. E. *Dynamic programming.* Princeton Landmarks in Mathematics and Physics. Dover Publications, 2003.

[4] Beltrametti, M. et al. “Moore–Penrose approach in the Hough transform framework”. In: *Applied Mathematics and Computation* vol. 375 (2020).

[5] Beltrametti, M. C. and Robbiano, L. “An algebraic approach to Hough transforms”. In: *Journal of Algebra* vol. 371 (2012), pp. 669–681.

[6] Berlinet, A. and Thomas-Agnan, C. *Reproducing Kernel Hilbert Spaces in Probability and Statistics.* Springer, 2004.

[7] Berman, H. M. et al. “The Protein Data Bank”. In: *Nucleic Acids Research* vol. 28, no. 1 (Jan. 2000), pp. 235–242.

[8] Bernstein, S. N. “Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités”. In: *Communications de la Société mathématique de Kharkow* vol. 13, no. 1 (1912), pp. 1–2.

[9] Biasotti, S. et al. *Mathematical Tools for Shape Analysis and Description.* Vol. 6. Williston, VT, USA: Morgan & Claypool, 2014, pp. 1–138.

[10] Buchberger, B. “An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal”. PhD thesis. University of Innsbruck, 1965.

[11] Caputo, A. et al. “SHREC 2021: Skeleton-based hand gesture recognition in the wild”. In: *Computers & Graphics* vol. 99 (2021), pp. 201–211.

[12] Cayley, A. “Note sur la méthode d’élimination de Bézout”. In: *Journal für die reine und angewandte Mathematik* vol. 53 (1857), pp. 366–367.

[13] Cheney, E. W. “Approximation Theory, Wavelets and Applications”. In: ed. by Singh, S. P. Vol. 454. Springer. NATO Science Series (Series C: Mathematical and Physical Sciences), 1995. Chap. Quasi-interpolation, pp. 37–45.

[14] Conti, C., Romani, L., and Schenone, D. “Semi-automatic spline fitting of planar curvilinear profiles in digital images using the Hough transform”. In: *Pattern Recognition* vol. 74 (2018), pp. 64–76.

[15] Craven, P. and Wahba, G. “Smoothing noisy data with spline functions”. In: *Numerische Mathematik* vol. 31 (1978), pp. 377–403.

[16] David A. Cox, J. L. and O’Shea, D. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics. Springer, 2010.

[17] Dey, T. K. et al. “Computing geometry-aware handle and tunnel loops in 3D models”. In: *ACM Transactions On Graphics* vol. 27, no. 3 (2008), pp. 1–9.

[18] Dokken, T. “Approximate implicitization”. In: *Mathematical Methods for Curves and Surfaces.* Ed. by Lyche, T. and Schumaker, L. L. Vanderbilt University Press, 2001, pp. 81–102.

[19] Dokken, T. “Aspects of Intersection Algorithms and Approximation”. PhD thesis. University of Oslo, 1997.

[20] Duda, R. O. and Hart, P. E. “Use of the Hough Transformation to Detect Lines and Curves in Pictures”. In: *Commununications of the ACM* vol. 15, no. 1 (Jan. 1972), pp. 11–15.

[21] Funkhouser, T. et al. “A Search Engine for 3D Models”. In: *ACM Transactions On Graphics* vol. 22, no. 1 (2003), pp. 83–105.

[22] Gao, X.-S. and Chou, S.-C. “Implicitization of rational parametric equations”. In: *Journal of Symbolic Computation* vol. 14, no. 5 (1992), pp. 1–15.

[23] Ghosh, S. *Kernel Smoothing: Principles, Methods and Applications.* John Wiley & Sons, 2017.

[24] Grothausmann, R. et al. “Shape and Facet Analyses of Alveolar Airspaces of the Lung”. In: *Shape in Medical Imaging.* Ed. by Reuter, M. et al. Springer International Publishing, 2018, pp. 49–64.

[25] Gubins, I. et al. “SHREC 2020: Classification in cryo-electron tomograms”. In: *Computers & Graphics* vol. 91 (2020), pp. 279–289.

[26] Hough, P. V. C. *Method and means for recognizing complex patterns.* US Patent 3069654. Dec. 1962.

[27] Johnson, R. A. and Wichern, D. V. *Applied multivariate statistical analysis.* 6th ed. PEARSON Prentice Hall, 2007.

[28] Kreuzer, M. and Robbiano, L. *Computational Commutative Algebra 1.* Springer, 2000.

[29] Kuhn, M. and Johnson, K. *Applied Predictive Modeling.* Springer New York, 2018.

[30] Laga, H. et al. *3D Shape Analysis: Fundamentals, Theory, and Applications.* Wiley, 2018.

[31] Langenfeld, F. et al. “SHREC 2020: Multi-domain protein shape retrieval challenge”. In: *Computers & Graphics* vol. 91 (2020), pp. 189–198.

[32] Lee, R. N. “Two-Dimensional Critical Point Configuration Graphs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 6, no. 4 (1984), pp. 442–450.

[33] Li, W. et al. “Computer Aided Design (CAD) model search and retrieval using frequency domain file conversion”. In: *Additive Manufacturing* vol. 36 (2020).

[34] Lyche, T. and Mørken, K. *Spline Methods Draft.* 2011.

[35] Massone, A. M. et al. “Profile Detection in Medical and Astronomical Images by Means of the Hough Transform of Special Classes of Curves”. In: *Journal of Mathematical Imaging and Vision* vol. 51 (2015), pp. 296–310.

[36] Mukhopadhyay, P. and Chaudhuri, B. B. “A survey of Hough Transform”. In: *Pattern Recognition* vol. 48, no. 3 (2015), pp. 993–1010.

[37] Naisbitt, J. *Megatrends: Ten New Directions Transforming Our Lives.* Warner Books, 1982.

[38] Nason, G. *Wavelet Methods in Statistics with R.* Springer, 2008.

[39] Nelder, J. A. and Wedderburn, R. W. M. "Generalized Linear Models". In: *Journal of the Royal Statistical Society. Series A (General)* vol. 135, no. 3 (1972), pp. 370–384.

[40] Patané, G. et al. "Comparing methods for the approximation of rainfall fields in environmental applications". In: *ISPRS Journal of Photogrammetry and Remote Sensing* vol. 127 (2017), pp. 57–72.

[41] Philip, K. et al. "The fuzzy Hough transform-feature extraction in medical images". In: *IEEE Transactions on Medical Imaging* vol. 13, no. 2 (1994), pp. 235–240.

[42] Regli, W. C. and Gaines, D. M. "A repository for design, process planning and assembly". In: *Computer-Aided Design* vol. 29, no. 12 (1997), pp. 895–905.

[43] Robbiano, L. "Hyperplane sections, Gröbner bases, and Hough transforms". In: *Journal of Pure and Applied Algebra* vol. 219, no. 6 (2015), pp. 2434–2448.

[44] Romanengo, C., Biasotti, S., and Falcidieno, B. "Recognising decorations in archaeological finds through the analysis of characteristic curves on 3D models". In: *Pattern Recognition Letters* vol. 131 (2020), pp. 405–412.

[45] Sablonnière, P. "Univariate spline quasi-interpolants and applications to numerical analysis". In: *Rendiconti del Seminario Matematico.* Vol. 63. 2. Università degli studi di Torino / Politecnico di Torino, 2005, pp. 211–222.

[46] Schumaker, L. L. *Spline Functions: Basic Theory.* Cambridge University Press, 2007.

[47] Shalaby, M. F. et al. "Piecewise approximate implicitization: experiments using industrial data". In: *Algebraic Geometry and Geometric Modeling.* Ed. by Elkadi, M., Mourrain, B., and Piene, R. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 37–51.

[48] Shikin, E. V. *Handbook and atlas of curves.* Taylor & Francis, 1995.

[49] Shilane, P. et al. "The Princeton Shape Benchmark". In: *Shape Modeling International.* June 2004.

[50] Sipiran, I. et al. "SHREC 2021: Retrieval of cultural heritage objects". In: *Computers & Graphics* vol. 100 (2021), pp. 1–20.

[51] Sourbron, S. and Buckley, D. L. "Tracer kinetic modelling in MRI: estimating perfusion and capillary permeability". In: *Physics in medicine and biology* vol. 57, no. 2 (2012), R1–R33.

[52] Sylvester, J. J. “On a theory of the syzygetic relations of two rational integral functions, comprising an application to the theory of Sturm’s functions, and that of the greatest algebraical common measure”. In: *Philosophical Transactions of the Royal Society of London* vol. 143 (1853), pp. 407–548.

[53] Veltkamp, R. C. et al. *SHREC2006: 3D Shape Retrieval Contest*. Tech. rep. UU-CS-2006-030. Utrecht University, 2006.

[54] Wegman, E. J. and Wright, I. W. “Splines in Statistics”. In: *Journal of the American Statistical Association* vol. 78, no. 382 (1983), pp. 351–365.

[55] Yu, Y. et al. “Polyhedral 3D structure of human plasma very low density lipoproteins by individual particle cryo-electron tomography”. In: *Journal of Lipid Research* vol. 57, no. 10 (2016), pp. 1879–1888.

[56] Zana, F. and Klein, J. “A multimodal registration algorithm of eye fundus images using vessels detection and Hough transform”. In: *IEEE Transactions on Medical Imaging* vol. 18, no. 5 (1999), pp. 419–428.

[57] Zhao, R. et al. “Protein pocket detection via convex hull surface evolution and associated Reeb graph”. In: *Bioinformatics* vol. 34, no. 17 (Sept. 2018), pp. i830–i837.

Appendix 1.A B-splines

This appendix introduces univariate B-splines. For a more complete and comprehensive exposition, the reader is referred to [46].

Definition 1.1 (Univariate B-splines)**.** Let $\boldsymbol{\tau} = [\tau_1, \dots, \tau_{p+2}]$ be any non-decreasing sequence, commonly referred to as *local knot vector.* A *(univariate) B-spline* $B[\boldsymbol{\tau}] : \mathbb{R} \to \mathbb{R}$ of degree $p \geq 0$ is the function recursively defined by

$$B[\boldsymbol{\tau}](x) := \frac{x - \tau_1}{\tau_{p+1} - \tau_1} B[\tau_1, \dots, \tau_{p+1}](x) + \frac{\tau_{p+2} - x}{\tau_{p+2} - \tau_2} B[\tau_2, \dots, \tau_{p+2}](x), \quad (1.7)$$

where

$$B[\tau_i, \tau_{i+1}](x) := \begin{cases} 1, & \text{if } x \in [\tau_i, \tau_{i+1}) \\ 0, & \text{elsewhere} \end{cases}, \quad i = 1, \dots, p+1.$$

Here, the convention is assumed that "$0/0 = 0$".

By denoting

$$\omega_{i,p}(x) := \frac{x - \tau_i}{\tau_{p+i} - \tau_i},$$

we can rewrite the recursive equation as

$$B[\boldsymbol{\tau}](x) = \omega_{1,p}(x) B[\tau_1, \dots, \tau_{p+1}](x) + (1 - \omega_{2,p}(x)) B[\tau_2, \dots, \tau_{p+2}](x),$$

and by repeating the recurrence relation we obtain

$$\begin{aligned} B[\boldsymbol{\tau}](x) &= \omega_{1,p}(x)\omega_{1,p-1}(x) B[\tau_1, \dots, \tau_p](x) + \\ &+ \omega_{1,p}(x)(1 - \omega_{2,p-1}(x)) B[\tau_2, \dots, \tau_{p+1}](x) + \\ &+ (1 - \omega_{2,p}(x))\omega_{2,p-1}(x) B[\tau_2, \dots, \tau_{p+1}](x) + \\ &+ (1 - \omega_{2,p}(x))(1 - \omega_{3,p-1}(x)) B[\tau_3, \dots, \tau_{p+2}](x) = \\ &= \cdots = \\ &= \prod_{i=0}^{p-1} \omega_{1,p-i}(x) B[\tau_1, \tau_2](x) + \cdots + \prod_{i=0}^{p-1} (1 - \omega_{i+2,p-i}(x)) B[\tau_{p+1}, \tau_{p+2}](x) \end{aligned}$$

The above expansion allows to derive the following fundamental properties:

- *Piecewise structure.* $B[\boldsymbol{\tau}]$ is a polynomial of degree p in any non-empty interval $[\tau_i, \tau_{i+1})$.
- *Positivity.* $B[\boldsymbol{\tau}] \geq 0$, and $B[\boldsymbol{\tau}](x) > 0$ for any $x \in (\tau_1, \tau_{p+2})$.
- *Compact support.* The support of $B[\boldsymbol{\tau}]$, i.e., the closure of the subset of the domain where $B[\boldsymbol{\tau}]$ is non-zero, is the compact interval $[\tau_1, \tau_{p+2}]$.

Definition 1.2 (Univariate spline space)**.** Given a global knot vector $\boldsymbol{\tau} = [\tau_1, \dots, \tau_{n+p+1}]$, the *spline space* $\mathbb{S}_{p,\boldsymbol{\tau}}$ is the linear space defined by

$$\mathbb{S}_{p,\boldsymbol{\tau}} := \operatorname{span}\left\{ B[\boldsymbol{\tau}^{(1)}], \dots, B[\boldsymbol{\tau}^{(n)}] \right\},$$

where $\boldsymbol{\tau}^{(j)} := [\tau_j, \dots, \tau_{j+p+1}]$ for any $j = 1, \dots, n$. An element $f \in \mathbb{S}_{p,\boldsymbol{\tau}}$ is called a *spline function*, or just a *spline*, of degree p with knots $\boldsymbol{\tau}$.

Lemma 1.1 (Marsden's identity)**.** *Let the global knot vector $\boldsymbol{\tau} = [\tau_1, \dots, \tau_{n+p+1}]$ be given. Then we have*

$$\frac{(x-t)^{p-m}}{(p-m)!} = \sum_{i=1}^{n} \frac{(-1)^m}{p!} \frac{d^m \psi_{i,p}(t)}{dt^m} B[\tau_i, \cdots, \tau_{i+p+1}](x),$$

for any $x \in [\tau_1, \tau_{n+p+1}]$ and $0 \leq m \leq p$, and where

$$\psi_{i,0}(t) := 1, \quad \psi_{i,p}(t) = (\tau_{i+1} - t) \cdots (\tau_{i+p} - t).$$

The result in Lemma 1.1 leads to the following properties:

- *(Local) representation of polynomials.* By using the Taylor expansion, we have that any polynomial q of degree p can be written in terms of B-splines of the same degree. That is to say,

$$q(x) = \sum_{i=1}^{n} \Lambda_{i,p}(q) B[\tau_i, \cdots, \tau_{i+p+1}](x), \tag{1.8}$$

  for any $x \in [\tau_1, \tau_{n+p+1}]$ and where

$$\Lambda_{i,p}(q) := \sum_{m=0}^{p} \frac{(-1)^m}{p!} \frac{\mathrm{d}^m \psi_{i,p}(t)}{\mathrm{d}t^m} \frac{\mathrm{d}^{p-m} q(t)}{\mathrm{d}t^{p-m}}, \quad t \in \mathbb{R}.$$

- *(Local) partition of unity.* When setting $q(x) = 1$, Equation 1.8 gives

$$1 = \sum_{i=1}^{n} B[\tau_i, \cdots, \tau_{i+p+1}](x), \quad x \in [\tau_1, \tau_{n+p+1}].$$

Theorem 1.A.1 (Curry-Schoenberg)**.** *Let the global knot vector $\boldsymbol{\tau} = [\tau_1, \dots, \tau_{n+p+1}]$ be given, so that no knot occurs more than $p+1$ times. Then, $\left\{B[\boldsymbol{\tau}^{(j)}]\right\}_{j=1}^{n}$ is a basis for $\mathbb{S}_{p,\boldsymbol{\tau}}$.*

Lastly, we define a class of knot vectors, relevant for the definition of Variation Diminishing Spline Approximations (see Section 1.1.2).

Definition 1.3. A knot vector $\boldsymbol{\tau} = [\tau_1, \dots, \tau_{n+p+1}]$ is said to be $(p+1)$-*regular* if

1. $n \geq p+1$,
2. $\tau_1 = \tau_{p+1}$ and $\tau_{n+1} = \tau_{n+p+1}$,
3. $\tau_j < \tau_{j+p+1}$ for $j = 1, \dots, n$.

Chapter 2

Summary of Papers

In this section we give a summary of the papers included in the thesis. For each paper, we outline the main contribution, specify what kind of numerical simulation has been performed and which programming languages were employed. Table 2.1 further contextualizes the papers with respect to the three main topics outlined in the introduction (see Sections 1.1—1.3).

Paper I introduces an algorithm to extract geometric information from CAD models, mainly for reverse engineering purposes. More specifically, given a set of non-overlapping rational parametrized patches, the aim is to partition such patches into subsets corresponding to the underlying primitive shape they originate from. Since the algorithm is intended for CAD applications, the most relevant cases are those of two- and three-dimensional patches (i.e., curve and surface patches); nevertheless, it can be trivially extended to any dimension. Performance in terms of robustness and computational efficiency is evaluated both theoretically and numerically. A Python implementation of the algorithm for curves has been made available online. Since no similar method was present in the CAD domain, the paper fills a gap in the literature.

Paper II defines a family of quasi-interpolation methods to handle various types of data imperfections. The paper proves the family's numerical properties and gives an interpretation in terms of nonparametric regression, by studying bias and variance of the corresponding parameter estimators. The simulation on real data shows how this quasi-interpolant can successfully deal with applications such as curve fitting, surface reconstruction, and rainfall approximation and forecasting. A Python implementation of the method for the approximation of rainfall fields has been set online. The paper fills a gap in the literature, as it firstly extend a quasi-interpolation method to address perturbed point clouds.

Paper III provides a data-driven implementation of the quasi-interpolant scheme from Paper II, based on the supervised learning paradigm. The numerical study is here further deepened, by comparing with a number of well-known approximation methods and a set of performance indicators. In contrast to the naive implementation presented in Paper II, the use of a data-driven paradigm allows a full shift from the problem of approximation to that of prediction. All testing has been performed in Python and C. The paper shows that our method has a prediction capability that is comparable with that of other well-known methods on clean data, while exhibits a better performance on noisy data.

Paper IV focuses on the problem of point cloud segmentation for CAD applications. The paper combines the Hough transform technique with hierarchical clustering. The purpose of applying the Hough transform is twofold: to break the input point cloud into meaningful segments, either simple (planes, spheres, circular cylinders and cones) or more complex ones (e.g., general cylinders and cones, convex combination of curves), and to aggregate further such segments (e.g., to find all segments lying on the same cylinder, possibly up to some rigid transformation). Numerical simulation on synthetic and industrial data has been performed in Matlab. The advantages over previous segmentation (and fitting) methods are many: the use of the Hough transform allows to work with data affected by various point cloud artifacts (e.g., noise, outliers, uneven sampling), and is able to provide mathematical representations – at least theoretically – of segments sampled by any family of parametric or implicit representations. Compared to other implementations of the Hough transform, the use of initial estimates makes it possible to speed up the search for optimal solutions, by localizing its search in a dimensionally-reduced parameter space.

Paper V introduces Fit4CAD, a benchmark for fitting simple geometric primitives in CAD objects. The dataset is composed of 225 point clouds, obtained by sampling CAD objects and possibly perturbed by simulating missing data. The dataset is already split into a training set and a test set, to allow the testing of learning-based methods. To evaluate the performance of a candidate segmentation algorithm, a set of classification measures and quality indicators is introduced. Finally, the benchmark is used to compare two methods: the Hough-based approach from Paper IV and a clustering method based on a primitive growing framework. The benchmark has been made available online, with the aim of helping other researchers in evaluating and comparing their methods. The main contributions of the paper include the benchmark creation and its use to study the Hough-based segmentation method.

Paper VI applies shape analysis to structural biology. The aim is to evaluate the performance of different algorithms to classify and retrieve protein surfaces. A dataset of approximately $5,000$ protein surfaces and some corresponding physicochemical properties was created. Each model is given in the form of an OFF file and a TXT file: the former contains the triangle mesh representing the protein surface; the latter gives the physicochemical properties at the mesh vertices. The performance of the algorithms is evaluated by considering a set of classification and retrieval measures. The benchmark has been made available online. Its creation, as well as the methods' evaluation, has been carried out on C++, R and Matlab. Similarly to Paper V, the benchmark itself is one of the main contributions; in addition, the paper tests how physicochemical properties affect methods in computational geometry, which is of interest in application domains outside (applied) mathematics and computer science (e.g., in biochemistry).

Papers	Geometry reconstruction	Shape analysis	Benchmarking	Coding
Paper I	approximate implicitization	clustering	✗*	Python
Paper II	quasi-interpolation, nonparametric regression	-	✗*	Python
Paper III	quasi-interpolation, nonparametric regression	-	✓	Python, C
Paper IV	-	shape descriptors, segmentation	✓	Matlab
Paper V	-	shape descriptors, segmentation, classification	✓	Matlab
Paper VI	-	shape descriptors, retrieval, classification	✓	C++, R, Matlab

Table 2.1: Classification table for the six papers included in this thesis. Asterisks indicate that the paper has not used a benchmark, but has performed numerical simulation on real data.

Independent contributions to the papers

As independent contributions, the Ph.D. candidate has:

Paper I

- Investigated the combinability of approximate implicitization and cluster analysis.
- Designed a semi-automatic algorithm that, through the estimation of relevant parameters, can cluster patches w.r.t. the underlying primitive shape.
- Addressed the theoretical proofs behind the method.
- Proposed and studied the numerical stability of approximate implicitization, as well as identified the method shortcomings.
- Tested the algorithm on synthetic data and on a 3D model.

Paper II

- Proposed the use of the quasi-interpolation paradigm for efficient approximaton of point clouds affected by noise and outliers.
- Generalized a simple quasi-interpolation method and studied its geometric and probabilist properties.
- Taken care of the numerical simulation on the data provided by the co-author.

Paper III

- Proposed and implemented a data-driven method, based on the supervised learning paradigm.
- Identified a number of methods, among the popular ones with an implementation freely-available, for comparison purposes.
- Performed a structured comparison w.r.t. data from different sources and on the basis of different metrics.

Paper IV

- Investigated the use of the Hough Transform technique in CAD applications, to address shortcomings of traditional CAGD methods (e.g., approximate implicitazion).
- Tested adaptive methods for the speeding up of the search of optimal parameters in voting-based approaches.
- Selected CAD models for testing purposes and pre-processed them.
- Defined the equations of helical surfaces and helical strips, as generalization of simpler surfaces/curves.
- Analysed the output of the Hough Transform via cluster analysis.
- Taken care of data visualization for Figures IV.3-IV.11 and the tables.

Paper V

- Pre-processed part of the ABC dataset, to filter some of the files through bash scripts and recover missing information (e.g., implicit representations).
- Used Onshape to create CAD models and, from those, extract point clouds and ground truth representations.
- Simulated missing parts through CloudCompare.
- Split the dataset into a training set and a test set.
- Selected the classification measures and interpreted the results of the two studied methods w.r.t. them.
- Interpreted the results of the approximation measures.
- Taken care of data visualization for Figures V.2, V.3, V.9 and V.10, and Tables V.2 and V.3.

Paper VI

- Got acquainted with NanoShaper and Delphi for the approximation of molecular surfaces and of the electrostatic potential at the mesh vertices.
- Downloaded $5,854$ PDB files from the Protein Data bank repository, starting from a list of PDB codes, and preprocessed them.

- Generated the triangle meshes approximating the corresponding SES surfaces; studied and approximated hydrophobicity and presence of electron donors and acceptors at the mesh vertices.
- Identified a set of classification and retrieval measures that could provide a rich description of the performance of computational methods, and used it to evaluate all methods submitted to the SHREC track; discussed pros and cons w.r.t. the classification measures.
- Taken care of data visualization for all figures and tables.

Papers

Paper I

Reverse engineering of CAD models via clustering and approximate implicitization

Andrea Raffo, Oliver Barrowclough, Georg Muntingh

Published in *Computer Aided Geometric Design*, June 2020, volume 80, DOI: 10.1016/j.cagd.2020.101876.

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.

Keywords: reverse engineering, clustering, statistical learning, approximate implicitization.

Contents

I.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 [3]. 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 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 [19] and volumetric trimming [7], 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 [21]. We refer the reader to [10, 15, 16, 18, 20, 22] 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 [8]. 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, [23] and [14]).

In this paper the geometry is reconstructed using implicit algebraic surfaces. According to a survey reported in [17], "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 I.2 we introduce the main tools of our method, presenting basic definitions and existing results that will be used later on. In Section I.3, the core of our paper, we present the building blocks and mathematical results forming the foundation of the detection method. In Section I.4 we formalize these results into an explicit algorithm. In Section I.5 we present a theoretical analysis of the robustness of the method. In Section I.6 we present several examples of the application of our method, based on an implementation of the algorithm in Python that is available online[1]. Section I.7 concludes the main part of the paper, discussing encountered challenges and directions for future research. Finally, in Appendix I.A we report some of the proofs regarding the mathematical theory behind the algorithm.

I.2 Background

I.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 [11] 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

[1] https://github.com/georgmuntingh/ImplicitClustering

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 $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ 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 $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})^T \in \mathbb{R}^n$ is the realization of a *feature* or *attribute*.

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

Definition I.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 I.1 we take in account also the triangular inequality:

Definition I.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 unfavourable ones. From a geometrical viewpoint, the use of a dissimilarity rather than a distance leads to a generalized metric space [12].

Example I.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 [9] and references therein.

I.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 [13]. 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 [1, 6]) 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 I.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 I.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 I.1 (Dokken [6])**.** *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 I.1 (Dokken [6])**.** *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 I.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

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

■

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

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

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

I.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

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

Analogously to Proposition I.1, we can bound the algebraic error as

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

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 I.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* (I.2)*.*

2. *Compute the singular value decomposition* $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

3. *Select* $\mathbf{b} = \mathbf{v}_{\min}^T$ *the right singular vector corresponding to the smallest singular value* $\sigma_{\min}^{(m)}\left(\mathscr{P}, \{\pi_j\}_{j=1}^M\right)$.

I.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.

I.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 I.2. *Let* τ *be a non-trivial polynomial or rational patch in* $\mathbb{R}^n$. *Then the implicit degree of* τ *can be written as*

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

where $\sigma_{\min}^{(\bar{m})}$ *is the shorthand notation for the smallest singular value of Algorithm I.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 I.1 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 I.1. 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 I.2 can be extended to consider discrete approximate implicitization if at least $N_{\min}$ unique samples are considered (see [1] for details).

Remark I.2. 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 (I.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 (I.3) by the weaker criterion

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

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 I.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 I.5, the estimation of the implicit degree can fail in floating-point arithmetic.

I.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 I.1. 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.,

$$X \ni \tau \xleftrightarrow{1-1} \ddot{\tau} := \{P_1^\tau, \ldots, P_N^\tau\} \in \ddot{X}.$$

Definition I.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 I.3. *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 I.5. Then:*

1. *The map* d_λ *is a dissimilarity* $\iff \lambda > 0$.

2. *There exists* $\lambda^* > 0$ *such that* d_λ *is a distance* $\iff \lambda \geq \lambda^*$.

Proof.

1. 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

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

   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.
   - $\lambda ||\mathbf{r}_{\text{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\text{CM}}(\ddot{\tau}_2)||_2 = 0$ iff the patches have the same center of mass or $\lambda = 0$. Since the first case is excluded by hypothesis, the lemma is proved.

2. 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

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

   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

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

■

Remark I.3. Notice that:

1. Any pair of patches τ_1 and τ_2 belongs to the same primitive shape iff $\sigma^{(m)}_{\min}(\ddot{\tau}_1 \cup \ddot{\tau}_2) = 0$ or, equivalently, iff $d_0(\tau_1, \tau_2) = 0$.

2. A parameter $\lambda > 0$ is required to exploit the theory of clustering analysis. On the other hand, λ should penalize the term $||\mathbf{r}_{\text{CM}}(\ddot{\tau}_1) - \mathbf{r}_{\text{CM}}(\ddot{\tau}_2)||_2$ 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^*.

3. The assumption in Lemma I.3 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.

I.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 $k = 1, \ldots, P - L$ and $e^{(k)} > 0$ for $k = P - L + 1, \ldots, N - 1$. Therefore the number of iterations can be defined as the maximum index k such that $e^{(k)} = 0$, i.e.,

$$\bar{k} := \max\{k | e^{(k)} = 0\}. \tag{I.5}$$

- In floating-point arithmetic, Equation I.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 I.2:

Algorithm I.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 I.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 I.6.

I.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 I.3.1. The tuning parameters $\{\xi^{(m)}\}$ are computed using Algorithm I.2. The maximum implicit degree of the patches $m_{\max}$ is returned as a result of this first 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 I.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 I.5: Partition of the set of patches X according to their implicit degrees.

Finally, we apply the agglomerative clustering approach described in Section I.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 I.3.3. We recall that in the agglomerative approach each patch constitutes a single cluster at the initial step.

```
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 $\mathbf{D}_X(j_1, j_2) := d_\lambda(\tau_{j_1}, \tau_{j_2})$.
            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 I.3.3;
            Set $k := k + 1$;
        end
    end
end
return $X = \cup X_{i,j}^{(k-1)}$
```

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

I.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.

I.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 $\mathbf{D}$ (I.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,

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

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 $\mathcal{O}(\min(NM^2, N^2M))$ [2]. 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 I.1), 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.

I.4.1.2 Estimating the degree of the patches

Algorithm I.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.

I.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*).

I.4.1.4 Clustering procedure

The naive implementation of agglomerative hierarchical clustering requires $\mathcal{O}(P^3)$ operations [4]. 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 [5].

I.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 I.2 (Scaling and smallest singular value)**.** *Let*

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

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* $\mathscr{P}_\mathbf{a} := \{\mathbf{p_a}(t_j)\}_j$ *be a family of uniformly sampled point clouds. Then*

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

Proposition I.3 (Translations and smallest singular value)**.** *Let*

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

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

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

Proposition I.4 (Rotation and smallest singular value)**.** *Let*

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

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 α, $\beta \in \mathbb{R}_{>0}$ such that

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

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 (I.4) or the stopping criterion of Section I.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.

I.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.

I.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 Figure I.1.

We consider the stopping tolerance presented in Algorithm I.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 of the dataset, where some of the dilations can compromise the degree estimation or the stopping criteria (see Section I.5 for details about the robustness of discrete approximate implicitization).

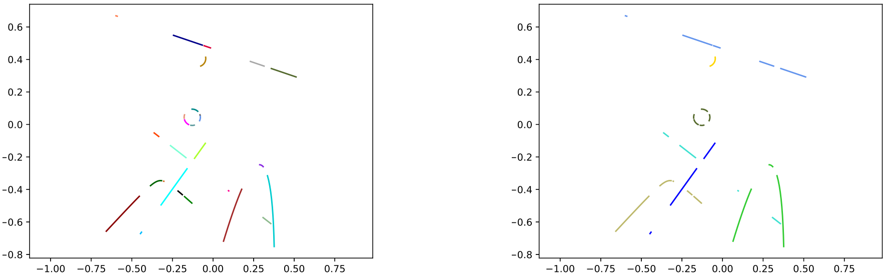


Figure I.1: Initial (left) and corresponding classified dataset (right), with segments colored by cluster.

I.6.2 Cubic Bézier curve approximations to circular segments

Let X be the set of segments of the two-dimensional gear in Figure I.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 I.4, because experimentally more robust when working with polynomial Bézier approximations. The resulting output shows the correct detection of the clusters.

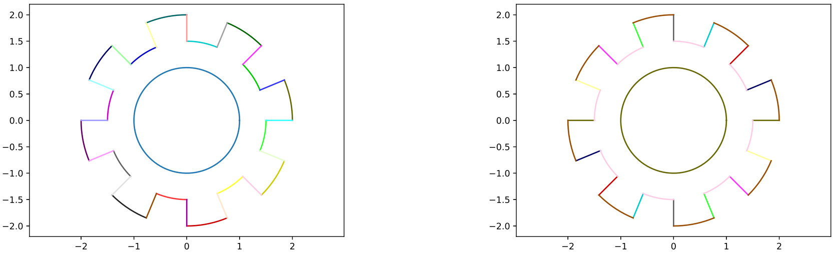


Figure I.2: Initial gear (left) and corresponding classified gear (right), with segments coloured by cluster.

I.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 I.1. Although these times suggest that both the dissimilarity matrix assembly and the clustering procedure in the implementation[2] have $\mathcal{O}(P^2)$ complexity, a further breakdown of the CPU times 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.

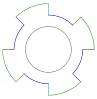
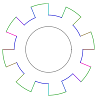
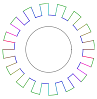
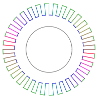
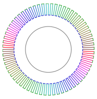
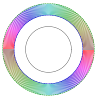
						
# teeth	4	8	16	32	64	128
# segments	17	33	65	129	257	513
t^{assembly}	0.006	0.021	0.071	0.256	1.095	4.025
o^{assembly}	-	1.708	1.778	1.843	2.096	1.879
$t^{\text{clustering}}$	0.002	0.003	0.008	0.042	0.115	0.405
$o^{\text{clustering}}$	-	0.980	1.339	2.352	1.436	1.823
t^{total}	0.009	0.025	0.081	0.300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

Table I.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.

I.6.2.2 Increasing Gaussian noise

Table I.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.

I.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 coloured randomly in Figure I.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, c.f. Remark I.2. 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

[2] https://github.com/georgmuntingh/ImplicitClustering

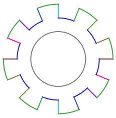
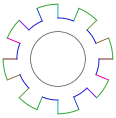
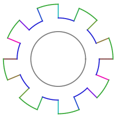
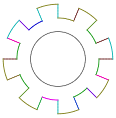
				
σ	10^{-5}	10^{-4}	10^{-3}	10^{-2}
result	✓	✓*	✓*	✗

Table I.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 I.4 fails, but the method yields the correct result via a user-defined threshold (or when the number of clusters is known).

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 Figure I.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.

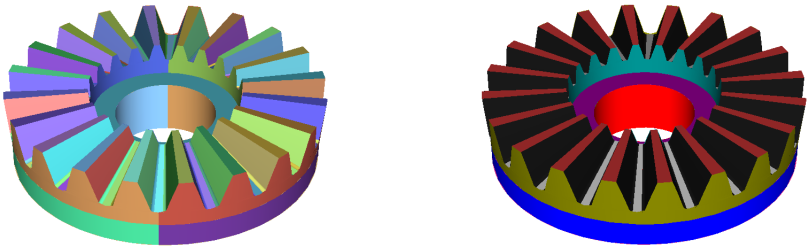


Figure I.3: Initial unclassified (left) and corresponding classified (right) Nugear model provided by STAM, with patches coloured by cluster.

I.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.

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.

References

[1] Barrowclough, O. and Dokken, T. “Approximate implicitization using linear algebra”. In: *Journal of Applied Mathematics* (2012).

[2] Beneš, Š. and Kruis, J. “Singular Value Decomposition used for compression of results from the Finite Element Method”. In: *Advances in Engineering Software* vol. 117 (2018). Special Section - CIVIL-COMP 2017, pp. 8–17.

[3] Chikofsky, E. and Cross, J. “Reverse engineering and design recovery: a taxonomy”. In: *IEEE Software* vol. 7, no. 1 (1990), pp. 13–17.

[4] Day, W. and Edelsbrunner, H. “Efficient algorithms for agglomerative hierarchical clustering methods”. In: *Journal of Classification* vol. 1, no. 1 (1984), pp. 7–24.

[5] Defays, D. "An efficient algorithm for a complete link method". In: *The Computer Journal* vol. 20, no. 4 (1977), pp. 364–366.

[6] Dokken, T. "Aspects of intersection algorithms and approximation". PhD thesis. University of Oslo, 1997.

[7] Dokken, T., Skytt, V., and Barrowclough, O. "Trivariate spline representations for computer aided design and additive manufacturing". In: *Computers & Mathematics with Applications* vol. 78 (2019), pp. 2168–2182.

[8] Fudos, I. "CAD/CAM Methods for Reverse Engineering: A Case study of Reengineering Jewellery". In: *Computer-Aided Design and Applications* (2006), pp. 683–700.

[9] Gan, G., Ma, C., and Wu, J. *Data Clustering Theory. Algorithms and Applications.* ASA-SIAM series on statistics and applied probability, 2007.

[10] Ghadai, S. et al. "Learning localized features in 3D CAD models for manufacturability analysis of drilled holes". In: *Computer Aided Geometric Design* vol. 62 (May 2018), pp. 263–275.

[11] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning.* Springer Series in Statistics, 2001.

[12] Khamsi, M. A. "Generalized metric spaces: A survey". In: *Journal of Fixed Point Theory and Applications* vol. 17, no. 3 (2015), pp. 455–475.

[13] Kotsireas, I. "Panorama of methods for exact implicitization of algebraic curves and surfaces". In: *Geometric Computation* vol. 11 (2004), pp. 126–155.

[14] Muraleedharan, L. P., Kannan, S. S., and Muthuganapathy, R. "Autoencoder-based part clustering for part-in-whole retrieval of CAD models". In: *Computers & Graphics* vol. 81 (2019), pp. 41–51.

[15] Niu, Z. et al. "Applying Database Optimization Technologies to Feature Recognition in CAD". In: *Computer-Aided Design and Applications* (2015).

[16] Niu, Z. et al. "Rapidly finding CAD features using database optimisation". In: *Computer-Aided Design* (2015).

[17] Rossignac, J. "Constraints in Constructive Solid Geometry". In: *Proceedings of the 1986 Workshop on Interactive 3D Graphics.* I3D '86. Chapel Hill, North Carolina, USA: ACM, 1987, pp. 93–110.

[18] Shu, Z. et al. "Unsupervised 3D shape segmentation and co-segmentation via deep learning". In: *Computer Aided Geometric Design* vol. 43 (2016), pp. 39–52.

[19] Skytt, V. and Dokken, T. "Models for Isogeometric Analysis from CAD". In: *IsoGeometric Analysis: A New Paradigm in the Numerical Approximation of PDEs.* Springer, 2016, pp. 71–86.

[20] Song, M. et al. "Iterative 3D shape classification by online metric learning". In: *Computer Aided Geometric Design* vol. 35–36 (May 2015), pp. 192–205.

[21] Várady, T., Martin, R., and Cox, J. “Reverse engineering of geometric models—an introduction”. In: *Computer-Aided Design* vol. 29, no. 4 (1997), pp. 255–268.

[22] Xia, Q. et al. “Automatic extraction of generic focal features on 3D shapes via random forest regression analysis of geodesics-in-heat”. In: *Computer Aided Geometric Design* vol. 49 (Dec. 2016), pp. 31–43.

[23] Xiao, D. et al. “CAD mesh model segmentation by clustering”. In: *Computers & Graphics* vol. 35, no. 3 (2011). Shape Modeling International (SMI) Conference 2011, pp. 685–691.

Appendix I.A On the numerical stability of discrete approximate implicitization

In this appendix we provide the proofs left out of Section I.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 I.2. 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

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

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 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. 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 $tr\left(\mathbf{B}\right)$ (and consequently λ_1) approaches $+\infty$ when a_1 or a_2 approaches zero. It follows that $\sigma^{(m)}_{\min}\left(\mathscr{P}_{\mathbf{a}}\right)$ approaches zero when a_1 or a_2 approaches zero. $\blacksquare$

Proof of Proposition I.3. The collocation matrix $\mathbf{D}_{a_1,a_2}$ has the form

$$\mathbf{D}_{a_1,a_2} = \mathbf{D}_{0,0}\mathbf{T}_{a_1,a_2}, \qquad \mathbf{T}_{a_1,a_2} := \left(\begin{array}{c|cc|ccc} 1 & a_1 & a_2 & a_1^2 & a_1a_2 & a_2^2 \\ 0 & 1 & 0 & 2a_1 & a_2 & 0 \\ 0 & 0 & 1 & 0 & a_1 & 2a_2 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right),$$

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
$$\sigma^{(m)}_{\min}\left(\mathscr{P}_{\mathbf{a}}\right) = \Big[\lambda_{\max}\left(\mathbf{B}\right)\Big]^{-1/2},$$
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
$$tr\left(\mathbf{B}\right) = \sum_{i=1}^{n} \lambda_i = \sum_{0 \leq i+j \leq 4} \alpha_{i,j} a_1^i a_2^j, \tag{I.7}$$
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 follow that $tr\left(\mathbf{B}\right)$ (and consequently λ_1) approaches $+\infty$ when either a_1 or a_2 approaches ∞, and consequently $\sigma^{(m)}_{\min}\left(\mathscr{P}_{\mathbf{a}}\right)$ approaches zero when either a_1 or a_2 approaches ∞. $\blacksquare$

Proof of Proposition I.4. 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

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

where $\widetilde{\mathbf{V}} := \mathbf{R}_\theta^T\mathbf{V}$ 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 $\alpha = \beta = \sigma_{\min}^{(m)}(\mathscr{P}_0)$.

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 $\mathbf{D}_\theta^T\mathbf{D}_\theta$. The eigenvalues of $\mathbf{D}_\theta^T\mathbf{D}_\theta$ 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. ■

Authors' addresses

Andrea Raffo Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, andrea.raffo@ge.imati.cnr.it

Oliver J. D. Barrowclough Department of Mathematics and Cybernetics, SINTEF, PO Box 124 Blindern, 0314 Oslo, Norway, oliver.barrowclough@sintef.no

Georg Muntingh Department of Mathematics and Cybernetics, SINTEF, PO Box 124 Blindern, 0314 Oslo, Norway, georg.muntingh@sintef.no

Paper II

Weighted quasi-interpolant spline approximations: Properties and applications

Andrea Raffo, Silvia Biasotti

Published in *Numerical Algorithms*, August 2021, volume 87, pp. 819–847. DOI: 10.1007/s11075-020-00989-4.

II

Abstract

Continuous representations are fundamental for modeling sampled data and performing computations and numerical simulations directly on the model or its elements. To effectively and efficiently address the approximation of point clouds we propose the Weighted Quasi Interpolant Spline Approximation method (wQISA). We provide global and local bounds of the method and discuss how it still preserves the shape properties of the classical quasi-interpolation scheme. This approach is particularly useful when the data noise can be represented as a probabilistic distribution: from the point of view of nonparametric regression, the wQISA estimator is robust to random perturbations, such as noise and outliers. Finally, we show the effectiveness of the method with several numerical simulations on real data, including curve fitting on images, surface approximation and simulation of rainfall precipitations.

Keywords: spline methods, quasi-interpolation nonparametric regression, point clouds, noise.

Contents

II.1 Introduction

Modelling sampled data with a continuous representation is essential in many applications such as, for instance, image resampling [8], geometric modelling [19], isogeometric analysis (IgA) [30] and the numerical solution of PDE boundary problems [4].

Spline interpolation is largely adopted to approximate data from a function or a physical object because of the simplicity of its construction, its ease and accuracy of evaluation, and its capacity to approximate complex shapes through mathematical element fitting and interactive design [48]. It is often preferred to polynomial interpolation because it yields visually effective results even when using low degree polynomials, while avoiding the Runge's phenomenon for higher degrees [28]. B-splines represent a popular way for dealing with spline interpolation and are nowadays the most powerful tool in CAGD [9]. Several generalizations to non-polynomial splines are possible, such as generalized splines [7], which admit also trigonometric or exponential bases, or non-uniform rational B-splines (NURBS) [45]. The B-spline extension to higher dimensions consists of multivariate spline functions based on a tensor product approach. Unfortunately, classical tensor product splines lack local refinement, which is often fundamental in those applications dealing with large amounts of data. For this reason several alternative structures that support local refinement have been introduced in the last decades; for instance, in the context of a tensor-product paradigm, T-splines [49], hierarchical B-splines [20], locally refined (LR) B-splines [18] and (truncated) Hierarchical B-splines (THB) [26].

When dealing with real data – for instance, acquired by laser scanners, photogrammetry and diagnostic devices – there are many source of uncertainty, such as resolution, precision, occlusions and reflections. Furthermore, digital models often undergo post-processing stages after acquisition, and these may introduce additional geometric and/or numerical artefacts [13]. Most of the existing inverse approximation techniques are executed as a deterministic problem and the parameters involved in the model are treated as unambiguous values. Despite the recent introduction of uncertainty-based inverse analysis tools such as evidence-theory, fuzzy and interval uncertainties [38], at the best of our knowledge, only few modelling approaches identify uncertainty theories as a good solution for explicitly modelling data uncertainty, adopting, for instance, interproximation [15] or fuzzy numbers [2]. Unfortunately, these efforts were quite isolated and their computational complexity prevented their massive adoption.

In this scenario, we aim at preserving the use of B-spline bases because of their simplicity, their approximation capability and accuracy. To effectively and efficiently approximate raw data and point clouds possibly affected by noise and outliers, we propose the adoption of a novel quasi-interpolation scheme. Quasi-interpolation is a well known technique [5, 47] that does not require to solve any linear system, unlike the traditional spline approaches, and therefore it allows to define more efficient algorithms. Whilst there are works on the use of quasi-interpolant methods for function approximation [5, 10, 11, 32, 44, 51], to the best of our knowledge, less efforts have been devoted to define quasi-interpolant schemes for point clouds [1, 3, 6, 25].

As working assumptions, we assume the point cloud to be embedded in an Euclidean space $\mathbb{R}^{d+1}$ and locally represented as a height field $y = f(x_1, \dots, x_d)$. We obtain a method which is not only robust, but also has a reduced computational complexity thanks to the adopted quasi-interpolation scheme. The method properties, presented in detail for the uni- and bivariate cases for simplicity of notation, can be easily extended to consider data of arbitrary dimension. We also discuss how the shape properties of monotonicity and convexity derive from classical spline theory. Since we aim at addressing data affected by noise, we provide a probabilistic interpretation of the method. We illustrate its properties over a number of examples, ranging from curve fitting to the approximation of scalar fields defined on surfaces. In summary, the main contributions of this work are:

- The introduction of a novel quasi-interpolation scheme to approximate point clouds, possibly affected by noise and outliers, together with a theoretical study of its numerical properties (Section II.2).
- The interpretation of our approach in terms of the nonparametric regression scheme, together with the theoretical study of bias and variance of the wQISA estimator (Section II.3).
- The validation of the method on real data from different applications, including curve fitting, surface reconstruction and rainfall approximation and forecasting (Section II.4).

Finally, concluding remarks are provided in Section II.5.

II.2 Weighted quasi-interpolant spline approximation for point clouds

In this Section we first summarise some basic notation and definitions on B-splines. We then formally introduce the weighted quasi-interpolant spline approximations, provide their global and local bounds and discuss in what sense they preserve the shape properties.

II.2.1 Basic concepts on spline spaces

From B-splines theory, it is well known that a non-decreasing sequence $\mathbf{t} = [t_1, \dots, t_{n+p+1}]$, which is commonly referred to as *global knot vector*, generates n B-splines of degree p over $\mathbf{t}$. In practice, the construction of each of these B-splines requires only a subsequence of $p+2$ consecutive knots, collected in a *local knot vector*.

Definition II.1 (Univariate B-spline)**.** Let $\mathbf{t} := [t_1, \dots, t_{p+2}]$ be a (local) knot vector. A *B-spline* $B[\mathbf{t}] : \mathbb{R} \to \mathbb{R}$ of degree p is the function recursively defined by

$$B[\mathbf{t}](x) := \frac{x - t_1}{t_{p+1} - t_1} B[t_1, \dots, t_{p+1}](x) + \frac{t_{p+2} - x}{t_{p+2} - t_2} B[t_2, \dots, t_{p+2}](x), \quad \text{(II.1)}$$

where

$$B[t_i, t_{i+1}](x) := \begin{cases} 1, & \text{if } x \in [t_i, t_{i+1}) \\ 0, & \text{elsewhere} \end{cases}, \quad i = 1, \dots, p+1.$$

Here, the convention is assumed that “0/0 = 0”.

By assuming $t_1 < t_{p+2}$, it follows that $B[\mathbf{t}]$ is a piecewise polynomial of degree p. The continuity at each unique knot is $p - m$, where m is the number of times the knot is repeated. $B[\mathbf{t}]$ is smooth in each open subinterval (t_i, t_{i+1}), where $i = 1, \dots, p+1$, and is non-negative over $\mathbb{R}$. The *support* of $B[\mathbf{t}]$, i.e., the closure of the subset of the domain where $B[\mathbf{t}]$ is non-zero, is the compact interval $\operatorname{supp}(B[\mathbf{t}]) = [t_1, t_{p+2}]$.

Definition II.2 (Univariate spline space)**.** Given a global knot vector $\mathbf{t} = [t_1, \dots, t_{n+p+1}]$, the *spline space* $\mathbb{S}_{p,\mathbf{t}}$ is the linear space defined by

$$\mathbb{S}_{p,\mathbf{t}} := \operatorname{span}\left\{ B[\mathbf{t}^{(1)}], \dots, B[\mathbf{t}^{(n)}] \right\},$$

where $\mathbf{t}^{(i)} := [t_i, \dots, t_{i+p+1}]$ for any $i = 1, \dots, n$. An element $f \in \mathbb{S}_{p,\mathbf{t}}$ is called a *spline function*, or just a *spline*, of degree p with knots $\mathbf{t}$.

By assuming that no knot occurs more than $p+1$ times, it follows that $\left\{ B[\mathbf{t}^{(i)}] \right\}_{i=1}^{n}$ is a basis for $\mathbb{S}_{p,\mathbf{t}}$. A B-spline basis forms a partition of unity over $[t_1, t_{n+p+1}]$. We can refine a spline curve $f = \sum_{i=1}^{n} b_i B[\mathbf{t}^{(i)}]$ by inserting new knots in $\mathbf{t}$ and then computing the coefficients of f in the augmented spline space. An efficient way to perform this process is the *Oslo algorithm* [16].

Lastly, we specify the type of knot vectors we will consider in the next sections, as they allow to define B-spline bases that interpolate the boundaries.

Definition II.3. A knot vector $\mathbf{t} = [t_1, \dots, t_{n+p+1}]$ is said to be $(p+1)$-*regular* if

1. $n \geq p+1$,
2. $t_1 = t_{p+1}$ and $t_{n+1} = t_{n+p+1}$,
3. $t_j < t_{j+p+1}$ for $j = 1, \dots, n$.

Definition II.4 (Tensor product B-spline)**.** A *tensor product B-spline* of multi-degree $\mathbf{p} := (p_1, \dots, p_d) \in \mathbb{N}^d$ is a separable function $B : \mathbb{R}^d \to \mathbb{R}$ defined as

$$B[\mathbf{t}_1, \dots, \mathbf{t}_d](\mathbf{x}) := \prod_{k=1}^{d} B[\mathbf{t}_k](x_k), \quad \text{(II.2)}$$

where $\mathbf{x} = (x_1, \dots, x_d)$ and $\mathbf{t}_k = [t_{k,1}, \dots, t_{k,p_k+2}] \in \mathbb{R}^{p_k+2}$ is the local knot vector along x_k, for any $k = 1, \dots, d$.

By assuming that $t_{k,1} < t_{k,p_k+2}$ for any $k = 1, \ldots, d$, it follows that $B[\mathbf{t}_1, \ldots, \mathbf{t}_d]$ is a piecewise polynomial of multi-degree $\mathbf{p}$.

Definition II.5 (Tensor product spline space). A *tensor product spline space* $\mathbb{S}_{\mathbf{p},[\mathbf{t}_1,\ldots,\mathbf{t}_d]}$ is the linear space defined by

$$\mathbb{S}_{\mathbf{p},[\mathbf{t}_1,\ldots,\mathbf{t}_d]} := \bigotimes_{k=1}^{d} \mathbb{S}_{p_k,\mathbf{t}_k} = \operatorname{span}\left\{ \prod_{k=1}^{d} B[\mathbf{t}_k^{(i_k)}] \text{ s.t. } i_k = 1, \ldots, n_k \right\},$$

where $\mathbf{t}_k \in \mathbb{R}^{n_k+p_k+1}$ is a global knot vector for any $k = 1, \ldots, d$. An element $f \in \mathbb{S}_{\mathbf{p},[\mathbf{t}_1,\ldots,\mathbf{t}_d]}$ is called a *tensor product spline* function, or just a *spline*, of multi-degree $\mathbf{p}$ with knot vectors $\mathbf{t}_1, \ldots, \mathbf{t}_d$.

The tensor product spline representation inherits all the properties (local support, non-negativity, local smoothness, partition of unity) of the univariate case. We refer the reader to [48] for a more exhaustive introduction to B-splines.

II.2.2 Weighted Quasi Interpolation Spline Approximation

We introduce our method for the general case of a point cloud $\mathcal{P} \subset \mathbb{R}^{d+1}$. Again, we assume that the point cloud can be locally represented by means of a function

$$y = f(x_1, \ldots, x_d).$$

Definition II.6. Let $\mathcal{P} \subset \mathbb{R}^{d+1}$ be a point cloud and $\mathbf{p} \in \mathbb{N}^d$ a (multi)-degree with all nonzero components. Let $\mathbf{t}_k \in \mathbb{R}^{n_k+p_k+1}$ be a $(p_k + 1)$-regular knot vector with boundary knots $t_{p_k} = a_k$ and $t_{n_k+1} = b_k$, for $k = 1, \ldots, d$. The *Weighted Quasi Interpolant Spline Approximation* (wQISA) of degree $\mathbf{p}$ to the point cloud $\mathcal{P}$ over the knot vectors $\mathbf{t}_k$ is defined by

$$f_w := \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \hat{y}_w\left(\xi_1^{(i_1)}, \ldots, \xi_d^{(i_d)}\right) \cdot B[\mathbf{t}_1^{(i_1)}, \ldots, \mathbf{t}_d^{(i_d)}], \tag{II.3}$$

where $\xi_k^{(i_k)} := (t_{k,i_k+1} + \ldots + t_{k,i_k+p_k})/p_k$ are the *knot averages* and

$$\hat{y}_w(\mathbf{u}) := \frac{\sum\limits_{(x_1,\ldots,x_d,y)\in\mathcal{P}} y \cdot w_{\mathbf{u}}(x_1, \ldots, x_d)}{\sum\limits_{(x_1,\ldots,x_d,y)\in\mathcal{P}} w_{\mathbf{u}}(x_1, \ldots, x_d)} \tag{II.4}$$

are the *control points estimators* of *weight functions* $w_{\mathbf{u}} : \mathbb{R}^d \to [0, +\infty)$.

The function $w_{\mathbf{u}} : \mathbb{R}^d \to [0, +\infty)$ of Definition II.6 defines a *window* around each point $\mathbf{u} \in \mathbb{R}^d$ and is also called a *Parzen window*. An example is the weight function:

$$w_{\mathbf{u}}(\mathbf{x}) := \begin{cases} 1/k, & \text{if } \mathbf{x} \in N_k(\mathbf{u}) \\ 0, & \text{otherwise} \end{cases}, \tag{II.5}$$

where $k \in \mathbb{N}^*$ and $N_k(\mathbf{u})$ denotes the neighborhood of $\mathbf{u}$ defined by the k closest points of the point cloud. In this case, $\hat{y}_w$ defines the k-nearest neighbor (k-NN) regressor (see figure II.1). Commonly, the function $w_{\mathbf{u}}$ depends on a distance, for examples:

$$w_{\mathbf{u}}(\mathbf{x}) = \mathbb{1}_{||\mathbf{x}-\mathbf{u}||_2 \leq r} \qquad \text{(Characteristic)} \qquad \text{(II.6a)}$$

$$w_{\mathbf{u}}(\mathbf{x}) = e^{-||\mathbf{x}-\mathbf{u}||_2/2\sigma^2} \qquad \text{(Gaussian)} \qquad \text{(II.6b)}$$

$$w_{\mathbf{u}}(\mathbf{x}) = e^{-||\mathbf{x}-\mathbf{u}||_2/\sqrt{2}\sigma} \qquad \text{(Exponential)} \qquad \text{(II.6c)}$$

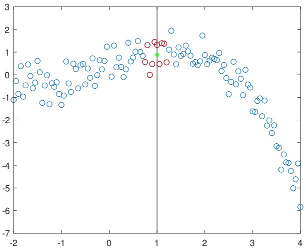


Figure II.1: Parzen windows and control points estimators. Given a 2D point cloud (in blue), we compute $\hat{y}_w$ at $u = 1$ (in green) by using the 10 nearest points (in red).

Note that:

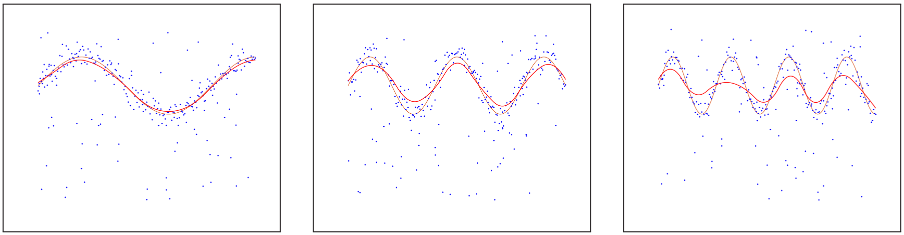
- $w_{\mathbf{u}}$ depends on the point $\mathbf{u} \in \mathbb{R}^d$ of interest, and can thus be adapted to local information (e.g., variable level and/or nature of noise).
- The quality of an approximation strongly depends on the spline space and the weight functions that are chosen in Definition II.6. As shown in Figure II.2, a given spline space and weight function is not always able to capture the relevant trends of a point set.

II.2.3 Properties

We first introduce bounds for the wQISA approximation. We then explain in what sense shape properties (monotonicity and convexity) are preserved in case of raw data. While we refer the reader to Appendix II.A for a detailed introduction of the univariate case, here, we focus our attention on the bivariate setting, i.e., on representations of the form $z = f(x, y)$. The extension of these results to higher dimensions is straightforward and just requires a more involved notation.

For the sake of simplicity, we re-write Equation II.3 as

$$f_w(x, y) := \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \hat{z}_w\left(\xi_x^{(i)}, \xi_y^{(j)}\right) \cdot B[\mathbf{x}^{(i)}, \mathbf{y}^{(j)}](x, y),$$



(a) (b) (c)

Figure II.2: wQISA curve approximation of three point clouds. The point sets (in blue) are sampled from $y = sin(\pi x)$ in (a), $y = sin(2\pi x)$ in (b) and $y = sin(3\pi x)$ in (c) and then perturbed with Gaussian noise and outliers. Here, we consider a spline space of dimension 10 over a uniform knot vector and a Gaussian weight function (see Equation II.6b) of fixed variance, combined with quartiles to filter the outliers. The figures shows the original functions (in orange) and the approximations (in red).

where we customize the notation by denoting with $\xi_x^{(i)}$ (resp. $\xi_y^{(j)}$) the i-th (resp. j-th) knot average with respect to the global knot vector $\mathbf{x}$ (resp. $\mathbf{y}$) along x (resp. y).

II.2.3.1 Global and local bounds

Proposition II.1 (Global bounds)**.** *Let $\mathcal{P} \subset \mathbb{R}^3$ be a point cloud. Given z_{min}, $z_{max} \in \mathbb{R}$ that satisfy*

$$z_{min} \leq z \leq z_{max}, \qquad \textit{for all } (x, y, z) \in \mathcal{P},$$

then the weighted quasi interpolant spline approximation to $\mathcal{P}$ from some spline space $\mathbb{S}_{p,[\mathbf{x},\mathbf{y}]}$ and some family of weight functions $w_u : \mathbb{R}^2 \to [0, +\infty)$ has the same bounds

$$z_{min} \leq f_w(x, y) \leq z_{max}, \qquad \textit{for all } (x, y) \in \mathbb{R}^2.$$

Proof. From the partition of unity of a B-spline basis, it follows that

$$\underset{\substack{\rotatebox{90}{$\leq$}\,①\\ z_{min}}}{\min_i \hat{z}_w(\xi_x^{(i)}, \xi_y^{(j)})} \leq \underset{\substack{\rotatebox{-90}{$=:$}\\ f_w}}{\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \hat{z}_w(\xi_x^{(i)}, \xi_y^{(j)}) \cdot B[\mathbf{x}^{(i)}, \mathbf{y}^{(j)}]} \leq \underset{\substack{\rotatebox{90}{$\geq$}\,②\\ z_{max}}}{\max_i \hat{z}_w(\xi_x^{(i)}, \xi_y^{(j)})} \tag{II.7}$$

where the inequalities ① and ② are a direct consequence of defining $\hat{z}_w$ by means of a convex combination. ■

The bounds of Proposition II.1 can potentially lead to local bounds, for example when the weight functions have bounded support. We discuss this possibility in Corollary II.1.

Corollary II.1 (Local bounds)**.** *Let $\mathcal{P} \subset \mathbb{R}^3$ be a point cloud. Let $x \in [x_\mu, x_{\mu+1})$ for some μ in the range $p_x + 1 \leq \mu \leq n_x$ and $y \in [y_\nu, y_{\nu+1})$ for some ν in the range $p_y + 1 \leq \nu \leq n_y$. Then*

$$\alpha(\mu,\nu) \leq f_w(x,y) \leq \beta(\mu,\nu)$$

for some $\alpha(\mu,\nu), \beta(\mu,\nu)$ which belong to $[z_{min}, z_{max}]$.

Proof. By using the property of local support for B-splines, it follows that

$$f_w(x,y) = \sum_{i=\mu-p_x}^{\mu} \sum_{j=\nu-p_y}^{\nu} \hat{z}_w(\xi_x^{(i)}, \xi_y^{(j)}) B[\mathbf{x}^{(i)}, \mathbf{y}^{(j)}](x,y).$$

Hence

$$\begin{array}{ccc}
\displaystyle\min_{\substack{i=\mu-p_x,\dots,\mu \\ j=\nu-p_y,\dots,\nu}} \hat{z}_w(\xi_x^{(i)}, \xi_y^{(j)}) & \leq f_w(x,y) \leq & \displaystyle\max_{\substack{i=\mu-p_x,\dots,\mu \\ j=\nu-p_y,\dots,\nu}} \hat{z}_w(\xi_x^{(i)}, \xi_y^{(j)}) \\
\rotatebox{-90}{$\geq$}\textcircled{3} & & \rotatebox{-90}{$\leq$}\textcircled{4} \\
\min\left\{z \text{ s.t. } (x,y,z) \in \mathcal{P}_{\mu,\nu}\right\} & & \max\left\{z \text{ s.t. } (x,y,z) \in \mathcal{P}_{\mu,\nu}\right\} \\
\rotatebox{-90}{$=:$} & & \rotatebox{-90}{$=:$} \\
\alpha(\mu,\nu) & & \beta(\mu,\nu)
\end{array} \tag{II.8}$$

where

$$\mathcal{P}_{\mu,\nu} := \bigcup_{\substack{i=\mu-p_x,\dots,\mu \\ j=\nu-p_y,\dots,\nu}} \operatorname{supp}\left(w_{(\xi_x^{(i)}, \xi_y^{(j)})}\right) \cap \mathcal{P}$$

and where supp denotes the support of a function. The inequalities ③ and ④ are a direct consequence of defining $\hat{z}_w$ by means of a convex combination. Note that the set $\mathcal{P}^*$ of points that are effectively used to compute the approximation, i.e.,

$$\mathcal{P}^* := \bigcup_{\substack{\mu=p_x+1,\dots,n_x \\ \nu=p_y+1,\dots,n_y}} \mathcal{P}_{\mu,\nu},$$

may be a proper subset of $\mathcal{P}$. ■

Note also that the results of Proposition II.1 and Corollary II.1 are independent from the type of mesh but rather rely on the partition of unity property. Therefore, a possibility is to consider local refinement strategies in order to further reduce the computational complexity and gain more flexibility only where truly needed.

II.2.3.2 Shape preservation

Shape preserving representations are crucial in geometric modeling (e.g., in CAD and CAM). Many classical quasi-interpolant strategies for function approximation preserve shape properties, such as the Bernstein approximants, the B-spline or multiquadratic (MQ) quasi-interpolants, the Variation Diminishing Spline Approximation (VDSA) and so on [22, 32, 39, 55].

In case of points clouds with defects, the average dataset trend is more important than the position of a single point with respect to the others. We thus introduce a notion of monotonicity and convexity for point clouds that take this consideration into account. Given a family of weight functions, we say that a point cloud is *w-monotone* (resp. *w-convex*) if the control point estimator $\hat{z}_w$ is monotone (resp. convex) (see Appendix II.A for a formal definition).

Monotonicity and convexity of the individual coordinates are preserved from w-monotonicity and w-convexity as a direct consequence of the univariate case, which is detailed in Appendix II.A. More precisely:

- *(Monotonicity)* Let us suppose that $\hat{z}_w(\cdot, y_0) : \mathbb{R} \to \mathbb{R}$ is monotonic for all $y_0 \in [a_2, b_2]$ (or at least it is its restriction to the nodes $\{\xi_x^{(i)}\}_{i=1}^{n_x}$). Then, f_w is an increasing function of x for each y. This statement is formally proved in Proposition II.3.

- *(Convexity)* Let us suppose that $\hat{z}_w(\cdot, y_0) : \mathbb{R} \to \mathbb{R}$ is convex for all $y_0 \in [a_2, b_2]$ (or at least it is its restriction to the nodes $\{\xi_x^{(i)}\}_{i=1}^{n_x}$). Then, f_w is a convex function of x for each y. This statement is formally proved in Proposition II.4.

In the multivariate setting, joint monotonicity and convexity straightforwardly derive from the control net shape [27, 39], here defined by $\hat{z}_w$. More precisely, a w-monotone (resp. w-convex) point cloud has a monotone (resp. convex) wQISA approximation.

II.2.3.3 Computational complexity

The wQISA method takes as input the point cloud $\mathcal{P} \subset \mathbb{R}^{d+1}$, the tensor product spline space $\mathbb{S}_{\mathbf{p},[\mathbf{t}_1,\ldots,\mathbf{t}_d]}$ defined by a multi-degree and a set of regular knot vectors, and the Parzen window function w. The approximation defined by Equation II.3 is computed by evaluating Equation II.4 as many times as the dimension of the tensor product spline space, i.e., $\dim(\mathbb{S}_{\mathbf{p},[\mathbf{t}_1,\ldots,\mathbf{t}_d]}) = \prod_{i=1}^{d} n_i$.

The single control point estimation depends on the function w chosen and, in particular, on its support (if global or local). In the numerical simulations proposed in Section II.4, we mainly focus on k-NN and Inverse Distance Weight (IDW) functions (see Equations II.5 and II.20) and, therefore, we here exhibit the computational complexity of wQISA for these choices of w. A deepen study of the computational complexity can be found in [46].

The k-nearest neighbor can be efficiently computed using the k-d tree algorithm in $O(N \log(N))$ operations [21], where N is the number of points of the cloud. The k-d tree then spatially stores the data in a structure such that, at runtime, the evaluation of w costs $O(k)$. Thus, the computation cost of the wQISA algorithm is given by the maximum of $O(N \log(N))$ and $O(k \cdot \dim(\mathbb{S}_{\mathbf{p},[\mathbf{t}_1,\ldots,\mathbf{t}_d]}))$.

The IDW weight is global and thus computes, for a single control point estimation, the linear combination of N terms. Since computing the weight of any point is at most as expensive as the inverse of an Euclidean norm, the computational complexity is $O(Nd)$. The computational cost of the wQISA algorithm is then $O(N \cdot \dim(\mathbb{S}_{\mathbf{p},[\mathbf{t}_1,\ldots,\mathbf{t}_d]}))$.

II.3 The wQISA method from a probabilistic perspective

Regression analysis techniques are widely used for prediction and forecasting. In regression problems, the conditional expectation of a response variable Y with respect to its predictor variables $X_1, \ldots, X_p$ is often approximated by its first-order Taylor expansion. Linearity in the predictors leads to a much easier interpretability of the model and is very efficient with sparse and small data. Global and local least square approaches are among the most popular linear regression methods. Nevertheless, these models need to solve linear systems of equations, which thus unnecessarily increases computational complexity as the data size increases. Moreover, linear models often depend on the normal distribution of the residuals, making them unreliable when the actual distribution is asymmetric or prone to outliers.

As the assumption of linearity might be too restrictive for real-world phenomena, various methods for moving beyond it have been introduced. A popular approach, known as *linear basis expansion*, considers multiple transformations of the predictors and then applies linear models in this richer space. Compared to traditional linear models, polynomial transformations of the predictors offer a more flexible data representation as they lead to higher-order Taylor expansions. On the other hand, they suffer a lack of local shape control due to their global nature. Compared to polynomial bases, piecewise polynomials allow to combine an increased flexibility with a reduced number of coefficients to compute. Furthermore, nonparametric regression may be used for a variety of purposes, such as scatterplot smoothing for pure exploration and interval estimates for uncertainty examination [29].

As we theoretically and numerically show in Sections II.3 and II.4, the wQISA method offers a competitive alternative to handle strongly perturbed large point clouds at a reduced computational cost, even when prone to outliers. In this Section we interpret the WQISA method as a non parametric regression problem. Independent ongoing studies on quasi-interpolation from a stochastic perspective are in [23, 24].

II.3.1 Formulation of the regression problem

Let Y be a univariate response variable. For the sake of simplicity, we restrict here to two predictor variables X_1 and X_2. As for the previous sections, the generalization to the multivariate case is trivial and just requires only a more involved notation. From now on, we assume that the relationship between the predictors and the dependent variable can be expressed as the conditional expectation:

$$\mathbb{E}(Y|X_1 = x_1, X_2 = x_2) = f_w(x_1, x_2).$$

The approximation f_w is here restricted to belong to a subspace of $\mathbb{S}_{\mathbf{p},[\mathbf{x}_1,\mathbf{x}_2]}$, where $\mathbf{p} \in \mathbb{N}^* \times \mathbb{N}^*$ is the (bi)-degree of the spline space over the (global) knot vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. More precisely, the relation between the observations Y_i and the independent variables $X_{i,1}$ and $X_{i,2}$ is formulated as

$$Y_i = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} c_{j_1,j_2} B[\mathbf{x}_1^{(j_1)}, \mathbf{x}_2^{(j_2)}](X_{i,1}, X_{i,2}) + \varepsilon_i, \quad i = 1, \dots, N, \tag{II.9}$$

where

- $B[\mathbf{x}_1^{(j_1)}, \mathbf{x}_2^{(j_2)}] : \mathbb{R}^2 \to [0, 1]$ is the (j_1, j_2)-th tensor product B-spline function with respect to the global knot vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively along X_1 and X_2.
- ε_i is the residual or disturbance term – an unobserved random variable that perturbs the linear relationship between the dependent variable and regressors.

Relation II.9 can be expressed, up to a reordering of the indexes (j_1, j_2), in the matrix form

$$\mathbf{Y} = \mathbf{B} \cdot \mathbf{c} + \boldsymbol{\varepsilon}, \tag{II.10}$$

where $\mathbf{Y} \in \mathbb{R}^{N\times 1} \ni \boldsymbol{\varepsilon}$, $\mathbf{B} \in \mathbb{R}^{N\times(n_1\cdot n_2)}$ and $\mathbf{c} \in \mathbb{R}^{(n_1\cdot n_2)\times 1}$.

II.3.2 Definition of the coefficient estimators

There are different methods to fit a linear model to a given dataset. In the following, we introduce our new estimators for the B-spline coefficients. The (j_1, j_2)-th component of $\hat{\mathbf{c}}$ is defined by

$$\hat{c}_{j_1,j_2} := \frac{\sum_{i=1}^{N} Y_i \cdot w_{(\xi_1^{(j_1)}, \xi_2^{(j_2)})}(X_{i,1}, X_{i,2})}{\sum_{i=1}^{N} w_{(\xi_1^{(j_1)}, \xi_2^{(j_2)})}(X_{i,1}, X_{i,2})}, \tag{II.11}$$

where $\xi_1^{(j_1)}$ and $\xi_2^{(j_2)}$ are the *knot averages* with respect to the B-spline B_{j_1,j_2} along the two directions. Notice that the weight functions

$$w_{(\xi_1^{(j_1)}, \xi_2^{(j_2)})} : \mathbb{R}^2 \to [0, +\infty)$$

act both as a penalty term and as a smoother on the given data.

II.3.3 Inference for regression purposes: the bias-variance decomposition

Suppose the data arise from a model $Y = f(X_1, X_2) + \varepsilon$. For the sake of simplicity, we assume here that the values of the predictors are fixed in advance, hence nonrandom. Further, we assume the error terms ε_i to be *independent identically distributed* (i.i.d) with mean $\mu_\varepsilon = 0$ and variance σ_ε^2.

The generalization performances of a method relies on the simultaneously minimization of two sources of error:

- The *bias* measures the difference between the model's expected predictions and the true values. High bias means an oversimplification of the model, i.e., the model does not produce accurate predictions (underfitting). The bias of a model is formally defined by

$$\text{Bias}^2\left[\hat{f}_w(X_1, X_2)\right] := \left(\mathbb{E}\left[\hat{f}_w(X_1, X_2)\right] - f(X_1, X_2)\right)^2. \tag{II.12}$$

- The *variance* measures the model's sensitivity to small fluctuations in the training set. High variance can result in a model that interpolates the given data but does not generalize on data which hasn't seen before (overfitting). The variance of a model is defined by

$$\text{Var}\left[\hat{f}_w(X_1, X_2)\right] := \mathbb{E}\left[\left(\hat{f}_w(X_1, X_2) - \mathbb{E}\left[\hat{f}_w(X_1, X_2)\right]\right)^2\right]. \tag{II.13}$$

II.3.3.1 Bias of a wQISA model

Let $(X_1, X_2) \in [x_{1,\mu}, x_{1,\mu+1}) \times [x_{2,\nu}, x_{2,\nu+1})$ for some $\mu = p_1 + 1, \ldots, n_1$ and for some $\nu = p_2 + 1, \ldots, n_2$. By using the property of local support of B-splines, we can then express $\mathbb{E}[\hat{f}_w(X_1, X_2)]$ as

$$\mathbb{E}[\hat{f}_w(X_1, X_2)] = \sum_{j_1=\mu-p_1}^{\mu} \sum_{j_2=\nu-p_2}^{\nu} \mathbb{E}[\hat{c}_{j_1,j_2}] \cdot B[\mathbf{x}_1^{(j_1)}, \mathbf{x}_2^{(j_2)}](X_1, X_2), \tag{II.14}$$

where

$$\mathbb{E}\left[\hat{c}_{j_1,j_2}\right] = \frac{\sum\limits_{i=1}^{N} f(X_{i,1}, X_{i,2}) \cdot w_{(\xi_1^{(j_1)}, \xi_2^{(j_2)})}(X_{i,1}, X_{i,2})}{\sum\limits_{i=1}^{N} w_{(\xi_1^{(j_1)}, \xi_2^{(j_2)})}(X_{i,1}, X_{i,2})} \tag{II.15}$$

is a convex combination. Once a spline space and weight functions have been chosen, Equations II.14 and II.15 can be combined to write down the exact formula of bias. However, in some cases bounds can simplify its study. Analogously to Proposition II.1, we can compute the following bounds for $\mathbb{E}\left[\hat{c}_{j_1,j_2}\right]$

$$\min_{i \in \mathcal{I}_{\hat{c}_{j_1,j_2}}} f(X_{i,1}, X_{i,2}) \le \mathbb{E}\left[\hat{c}_{j_1,j_2}\right] \le \max_{i \in \mathcal{I}_{\hat{c}_{j_1,j_2}}} f(X_{i,1}, X_{i,2}), \tag{II.16}$$

where

$$\mathcal{I}_{\hat{c}_{j_1,j_2}} := \Big\{ i = 1, \dots, N \text{ s.t. } w_{(\xi_1^{(j_1)},\xi_2^{(j_2)})}(X_{i,1}, X_{i,2}) \neq 0 \Big\}.$$

By combining Equations II.14 and II.16, it follows that

$$\begin{array}{ccccc} \min\limits_{i\in\mathcal{I}_{\mu,\nu}} f(X_{i,1},X_{i,2}) & \le & \sum\limits_{j_1=\mu-p_1}^{\mu}\sum\limits_{j_2=\nu-p_2}^{\nu}\mathbb{E}\big[\hat{c}_{j_1,j_2}\big]B[\mathbf{x}_1^{(j_1)},\mathbf{x}_2^{(j_2)}](X_1,X_2) & \le & \max\limits_{i\in\mathcal{I}_{\mu,\nu}} f(X_{i,1},X_{i,2}) \\ \shortparallel & & \shortparallel & & \shortparallel \\ \alpha(\mu,\nu) & & \mathbb{E}\big[\hat{f}_w(X_1,X_2)\big] & & \beta(\mu,\nu) \end{array} \tag{II.17}$$

where

$$\mathcal{I}_{\mu,\nu} := \bigcup_{\substack{j_1=\mu-p_1,\dots,\mu \\ j_2=\nu-p_2,\dots,\nu}} \mathcal{I}_{\hat{c}_{j_1,j_2}}$$

and where α and β denote the lower and upper bounds. We conclude that

$$\text{Bias}^2\Big[\hat{f}_w(X_1,X_2)\Big] \begin{cases} \le \big(\alpha(\mu,\nu) - f(X_1,X_2)\big)^2, & \text{if } \mathbb{E}\big[\hat{f}_w(X_1,X_2)\big] \le f(X_1,X_2) \\ \le \big(\beta(\mu,\nu) - f(X_1,X_2)\big)^2, & \text{if } \mathbb{E}\big[\hat{f}_w(X_1,X_2)\big] \ge f(X_1,X_2) \end{cases}, \tag{II.18}$$

where $\alpha(\mu,\nu)$ and $\beta(\mu,\nu)$ denote the minimum of maximum in Equation II.17.

II.3.3.2 Variance of a wQISA model

In the following Lemma we provide an exact formula for the variance while proving that, in the worst possible case, the variance will still be upper bounded by σ_ε^2. For the sake of simplicity, we consider a reordering of B-splines as in Equation II.10. This choice allows to substitute the indexes (j_1, j_2) with a single index j.

Lemma II.1. *The variance of $\hat{f}_w$ is upper-bounded by the variance of the error, i.e.,*

$$Var\Big[\hat{f}_w(X_1, X_2)\Big] \le \sigma_\varepsilon^2.$$

Proof.

$$\begin{aligned} \text{Var}\Big[\hat{f}_w(X_1,X_2)\Big] &= \mathbb{E}\left[\Big(\hat{f}_w(X_1,X_2) - \mathbb{E}\big(\hat{f}_w(X_1,X_2)\big)\Big)^2\right] = \\ &= \mathbb{E}\left[\left(\sum_i \hat{c}_i B_i(X_1,X_2) - \sum_i \mathbb{E}\left[\hat{c}_i\right] B_i(X_1,X_2)\right)^2\right] = \\ &= \mathbb{E}\left[\left(\sum_i \left(\hat{c}_i - \mathbb{E}\left[\hat{c}_i\right]\right) B_i(X_1,X_2)\right)^2\right] = \\ &= \sum_i\sum_j \mathbb{E}\left[\left(\hat{c}_i - \mathbb{E}\left[\hat{c}_i\right]\right)\left(\hat{c}_j - \mathbb{E}\left[\hat{c}_j\right]\right)\right] B_i(X_1,X_2)B_j(X_1,X_2) = \\ &= \sum_i\sum_j \text{Cov}\left(\hat{c}_i, \hat{c}_j\right) B_i(X_1,X_2)B_j(X_1,X_2), \end{aligned}$$

where

$$\begin{aligned}\operatorname{Cov}\left(\hat{c_i}, \hat{c_j}\right) &= \operatorname{Cov}\left(\frac{\sum_{k_1} Y_{k_1} \cdot w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right)}{\sum_{k_1} w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right)}, \frac{\sum_{k_2} Y_{k_2} \cdot w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_2,1}, X_{k_2,2}\right)}{\sum_{k_2} w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_2,1}, X_{k_2,2}\right)}\right) = \\ &= \frac{\sum_{k_1}\sum_{k_2} w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right) \cdot w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_2,1}, X_{k_2,2}\right) \operatorname{Cov}\left(Y_{k_1}, Y_{k_2}\right)}{\sum_{k_1}\sum_{k_2} w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right) \cdot w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_2,1}, X_{k_2,2}\right)} = \\ &= \sigma_\varepsilon^2 \frac{\sum_{k_1} w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right) \cdot w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_1,1}, X_{k_1,2}\right)}{\sum_{k_1}\sum_{k_2} w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right) \cdot w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_2,1}, X_{k_2,2}\right)}.\end{aligned}$$

Thus

$$\begin{aligned}\operatorname{Var}\Big[\hat{f}_w(X_1, X_2)\Big] &= \sum_i \sum_j \operatorname{Cov}\left(\hat{c_i}, \hat{c_j}\right) B_i(X_1, X_2) B_j(X_1, X_2) = \\ &= \sigma_\varepsilon^2 \sum_i \sum_j \frac{\sum_{k_1} w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right) \cdot w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_1,1}, X_{k_1,2}\right)}{\sum_{k_1}\sum_{k_2} w_{(\xi_1^{(i)},\xi_2^{(i)})}\left(X_{k_1,1}, X_{k_1,2}\right) \cdot w_{(\xi_1^{(j)},\xi_2^{(j)})}\left(X_{k_2,1}, X_{k_2,2}\right)} \cdot \\ &\cdot B_i(X_1, X_2) B_j(X_1, X_2) \leq \sigma_\varepsilon^2,\end{aligned}$$

where the inequality holds because $B_i B_j$ has the partition of unity property. ■

In Lemma II.1, the exact expression of the variance makes it possible to compute exact and approximated (pointwise) standard error bands (see Equation II.19).

II.3.3.3 Bias-variance decomposition for the k-NN weight

Let's consider an example to show how the results of the section work in practice. Let w be a k-NN weight function (see Equation II.5). The exact expression of the bias is found by combining Equation II.12 with the expected value

$$\begin{aligned}\mathbb{E}[\hat{f}_w(X_1, X_2)] = \frac{1}{k} \sum_{j_1=\mu-p_1}^{\mu} \sum_{j_2=\nu-p_2}^{\nu} & B[\mathbf{x}_1^{(j_1)}, \mathbf{x}_2^{(j_2)}](X_1, X_2) \cdot \\ \cdot \sum_{(X_{i,1}, X_{i,2}) \in N_k(\xi_1^{(j_1)}, \xi_2^{(j_2)})} & f(X_{i,1}, X_{i,2}).\end{aligned}$$

The exact expression of variance is given by

$$\operatorname{Var}\Big[\hat{f}_w(X_1, X_2)\Big] = \sigma_\varepsilon^2 \sum_i \sum_j \frac{k_{i,j}}{k^2} \cdot B_i(X_1, X_2) B_j(X_1, X_2) \leq \frac{\sigma_\varepsilon^2}{k},$$

where $k_{i,j}$ is the number of points in common, if any, among the k-closest to $(\xi_1^{(i)}, \xi_2^{(i)})$ and $(\xi_1^{(j)}, \xi_2^{(j)})$.

For small k, the estimate $\hat{f}_w$ can potentially adapt itself better to the underlying f, as it will avoid points further away to the knot averages. Under the assumption of increasing the point cloud size while keeping the sampling uniform, the bias for the 1-NN weight function vanishes entirely as the size of the training set approaches infinity and the mesh is uniformly refined. On the other hand, larger values of k can decrease the variance.

II.3.3.4 Numerical interpretation of the bias-variance decomposition

Figure II.3 shows the effect of spline spaces of different dimensions on the simple example

$$Y = \sin \pi X + \varepsilon,$$

with $X \sim U[-2, 2]$ and $\varepsilon \sim N(0, \sigma^2)$. Our dataset consists of $N = 300$ points (x_i, y_i) sampled on the exact curve and then perturbed.

The weighted quasi interpolant spline approximations for three different uniform knot vectors are shown. For the sake of simplicity, we here considered a 10-NN weight function. The shaded regions in the figures represent the (pointwise) standard error band of $\hat{f}_w$, i.e., the region

$$\hat{f}_w(X) \pm z^{(1-\alpha)} \cdot \sqrt{Var\left[\hat{f}_w(X)\right]}, \qquad \text{(II.19)}$$

where $z^{(1-\alpha)}$ is the $1 - \alpha$ percentile of the normal distribution. The three approximations displayed in Figures II.3(b-d) give a graphical representation of the bias-variance trade-off problem with respect to the dimension of the spline space:

n=5 The spline under-fits the data, with a more dramatic bias in those regions with a higher curvature

n=15 Compared to the previous case, the fitted function is closer to the true function. The variance has not increased appreciably yet.

n=30 The spline over-fits the data, which leads to a locally increased width of the bands.

In practice, the tuning parameters (here: n) can be selected via automatic procedures, for instance by using the K-fold cross-validation, generalized cross-validation and the so-called C_p statistic [29]. In Figure II.3(a) we include the 5-fold cross-validation curve

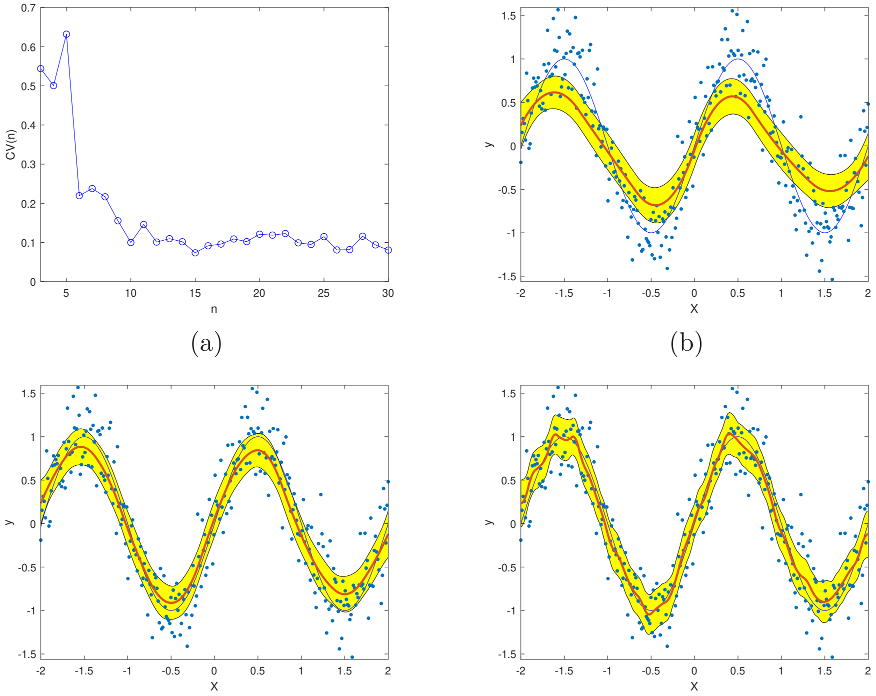
$$CV(n) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \hat{f}_w(x_i)\right)^2,$$

where f_w depends on n via the spline space.

Figure II.4 shows the approximation of a point cloud affected by the non-uniform noise $\varepsilon \sim N(0, s(X))$, defined as follows:

$$s(X) = e^{-\dfrac{1}{4(1 + e^{4X-2})}}.$$

The dataset consists of $N = 400$ points and is approximated by a spline space containing $n = 15$ B-splines over a uniform knot vector.



(a)

(b)

(c)

(d)

Figure II.3: Bias-variance tradeoff. In (a) we show the CV(n) curve for a realization from the chosen nonlinear additive error model. The minimum is reached at $n = 15$. The remaining panels show the data, the true function (in blue), the weighted quasi interpolant spline approximations (in red) and the (yellow shaded) bands of Equation II.19, for spline spaces of dimension $n = 5$ (b), $n = 15$ (c) and $n = 50$ (d). The bands corresponds here to an approximate 95% confidence interval.

II.4 Numerical simulations

We draw the effectiveness of our method in a number of real data coming from different sources and application domains. While [46] focused on the local approximation of 3D point clouds by wQISA surfaces, this section shows how the method is able to address the approximation problem for different dimensions. Indeed, our examples include curve approximation (on images and 3D objects), surface approximation (of 3D point clouds) and simulation of natural phenomena (like rainfall precipitation) over surfaces. Unless otherwise stated, we focus here on (bi-)quadratic spline approximations defined over uniform knot vectors, as they provide a sufficient flexibility for our purposes. Nevertheless, one can consider (bi-)degrees as additional parameters to assess and perform knot insertion to increase the degrees of freedom only where they are actually

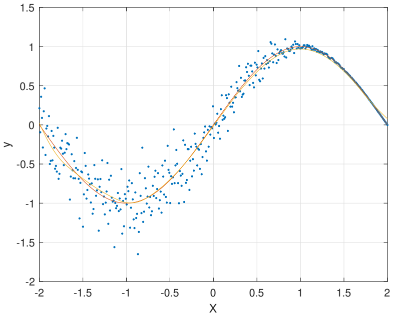


Figure II.4: Variable noise approximation. We show the original curve $f(X) = \sin(\pi/2 \cdot X)$ (in red) and the spline approximation (in yellow).

needed.

II.4.1 Evaluation criteria

The data acquisition devices and the subsequent post-processing operations generally introduce geometric and numerical artefacts. Unfortunately, for most of the data, the information on the quality of the acquisition devices and type of post-processing operations are lost or not available. Therefore, the hypothesis that the data to be approximated are *exact* is often unrealistic. Differently from other model representations, the peculiarity of wQISA is its capability of dealing with data affected by noise and outliers. This fact reflects on the measurements we can adopt to analyse the quality of the data approximation: indeed, it is not important how much the wQISA interpolates the original data rather it remains in a reasonable approximation range. To the best of our knowledge, a single performance measure able to capture such complex information does not exist; therefore, we will analyse the wQISA output with a number of measures, each one able to highlight different approximation aspects.

- When N observations Y_i are approximated by $\hat{Y}_i$, two popular measures of the statistical dispersion are the *Mean Squared Error* (MSE)

  $$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

  and the *Mean Absolute Error* (MAE)

  $$MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|.$$

  Although the MSE and MAE quantities are sample dependent and highly affected by data perturbation, they offer a very intuitive quantification of how close a point cloud and its approximation are.

- The *Hausdorff distance* is a well-known distance between two sets of points and applies for point clouds in all dimensions. In particular, we consider the Directed Hausdorff distance [17] from the points $a \in A \subset \mathbb{R}^t$ to the points $b \in B \subset \mathbb{R}^t$ as follows:

  $$d_{dHaus}(A, B) = \max_{a \in A} \min_{b \in B} d(a, b),$$

  with d the Euclidean distance. In order to have a coherent distance evaluation through models of different size, we normalize d_{dHaus} with respect to the diameter of the point cloud.

- The *Jaccard index* (also known as intersection over union) quantitatively estimates how two sets overlap. It is has been previously adopted to measure the performance of curve recognition methods for images [52] and 3D models [40]. The Jaccard index between two point sets A and B is defined as:

  $$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

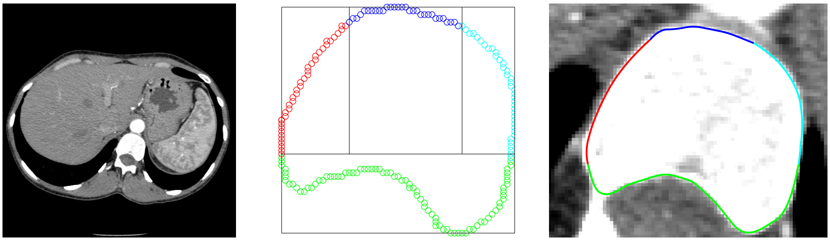
  where $|\cdot|$ denotes here the number of elements. The Jaccard index varies from 0 to 1, the higher the better. In our context, it can be adapted to the ratio of elements of the original point cloud that lie on the standard error bands of Equation II.19.

II.4.2 Curve approximation

We consider a 512 × 512 axial X-ray CT slice of a human lumbar vertebra (see Figure II.5(a)). First, we apply an edge detection technique, to detect the set of edge points. In this specific example, we adopt the Canny edge detector [12], others methods could be applied too. We select a bounding box for the point cloud, which is then partitioned in smaller sub-regions (see Figure II.5(b)). Lastly, we apply our technique to each sub-region to obtain a global approximation (see Figure II.5, right). Here, a 1-NN weight is set as the number of points is relatively small. Uniform knot vectors are considered as they produce reasonable approximations. The number of B-splines is chosen, in each sub-region, by a Leave-One-Out cross-validation [29]. Interpolating conditions are imposed at the boundaries in order to have a more natural C^1 continuity (see Figure II.5(c)). Notice that the shape of the vertebra is correctly preserved in the passage from the image to the final approximation.

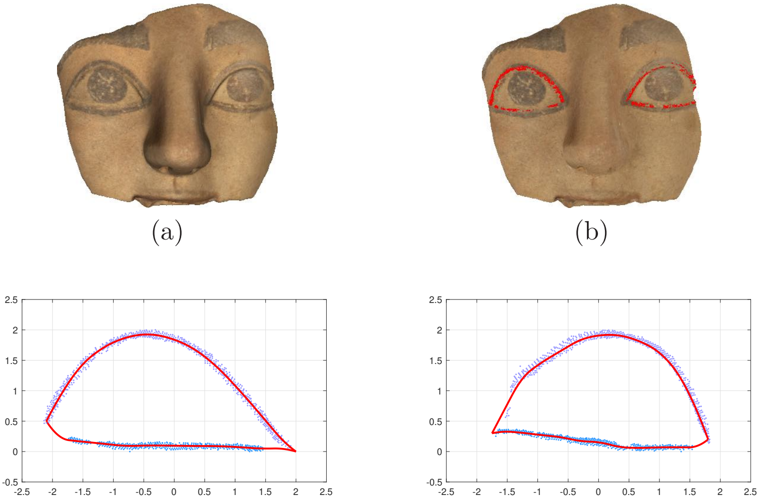
Figure II.6 shows an example of eye contour approximation from 3D models. We consider a fragment of a votive statue [35] stored in STARC repository[1] at The Cyprus Institute and extract the eye contours by filtering the point cloud through the values of the mean curvature values and clustering [54]. Each contour is projected onto its regression plane and then locally approximated. We here test a k-NN weight, with k to be assessed from patch to patch. Knot

[1] http://public.cyi.ac.cy/starcRepo/



(a) (b) (c)

Figure II.5: X-ray CT slice. In (a), the original image is shown. Figure (b) displays the edge points and the chosen partition: V1 in green, V2 in red, V3 in blue and V4 in light blue. In (c), the piecewise defined curve is superimposed to an enlargement of the original image.



(a) (b)

(c) (d)

Figure II.6: Approximation of the eye contour on a fragment of archaeological artifact. The statue (a) is first preprocessed to filter the eye contours points (b). Then, each point cloud is locally approximated. Here the points are clustered into: LE1 (left eye, light blue), LE2 (left eye, light purple), RE1 (right eye, light blue), RE2 (right eye, light purple).

vectors are again assumed to be uniform. For each eye profile, two curves are detected; the extrema knots of the two curves are fixed to be the same and are automatically selected as the leftmost and rightmost points of the whole profile. Notice that with these choices our approach is also able to fill the gaps in a reasonable way.

Table II.1 reports the values of the parameters n and k that best approximate the original curve segments and the corresponding error measures for the wQISA approximations.

Table II.1: Parameters and accuracy measures for the curve fitting examples. For each cluster of points we report: the sample size, the number of B-splines n, the tuning parameter k for the k-NN weight, the Mean Absolute Error (MAE), the Root Mean Squared Error (RMSE), the Jaccard index and the normalized Hausdorff distance. Parameters with asterisks are set by user.

	Lumbar Vertebra				**Left Eye**		**Right Eye**	
	V1	V2	V3	V4	LE1	LE2	RE1	RE2
sample size	82	38	30	38	422	730	428	638
n	20	6	8	7	12	12	8	12
k	1*	1*	1*	1*	5	5	5	5
MAE	0.656	0.3323	0.278	0.292	0.025	0.052	0.025	0.043
RMSE	0.898	0.418	0.378	0.379	0.030	0.074	0.030	0.079
Jaccard	0.988	1.000	1.000	1.000	0.995	1.000	1.000	1.000
Hausdorff	0.014	0.016	0.011	0.012	0.010	0.021	0.017	0.041

II.4.3 Surface approximation

A simulation on terrain data is shown in Figure II.7. The data are part of the Liguria-LAS dataset adopted as testbed in the iQmulus project [31], and come from a LIDAR dataset with spatial resolution of one meter. The area here selected contains 379.831 points. It is located in the Liguria region, in the north-west of Italy. The Liguria morphology, with several small catchments and even small rivers, is very challenging for the approximation methods to capture and preserve the most important and potentially critical characteristics [43]. The data are obtained with multiple swipes by airplane lidar acquisition. Some points come from multiple laser positions and therefore the same point can have multiple elevation values. In addition, since the data were only minimally post-processed to convert them to .las format, they contain also noise and outliers. In this example, we choose a C^1 (bi-)quadratic spline approximation because it is smooth enough to represent smooth terrains in a good way. We consider an Inverse Distance Weight (IDW), defined by:

$$w_{(u,v)}(x,y) := \begin{cases} \dfrac{1}{||(x,y)-(u,v)||_2}, & \text{if } |C_{(u,v)}| = 0 \\ \begin{cases} \dfrac{1}{|C_{(u,v)}|}, & \text{for all } (x,y) = (u,v) \\ 0, & \text{else} \end{cases}, & \text{if } |C_{(u,v)}| \neq 0 \end{cases} \tag{II.20}$$

where $|C_{(u,v)}| := \{(x,y,z) \in \mathcal{P} \text{ s.t. } (x,y) = (u,v)\}$. The IDW assigns greater influence to the points the closest to the knot averages and hence the most significant for the terrain approximation. The uniform knot vectors define in the final approximation 1024 B-splines in both directions and are chosen such that the MSE for the relative punctual error of each element is lower than 0.05 (which correspond to 0.05% of deviation).

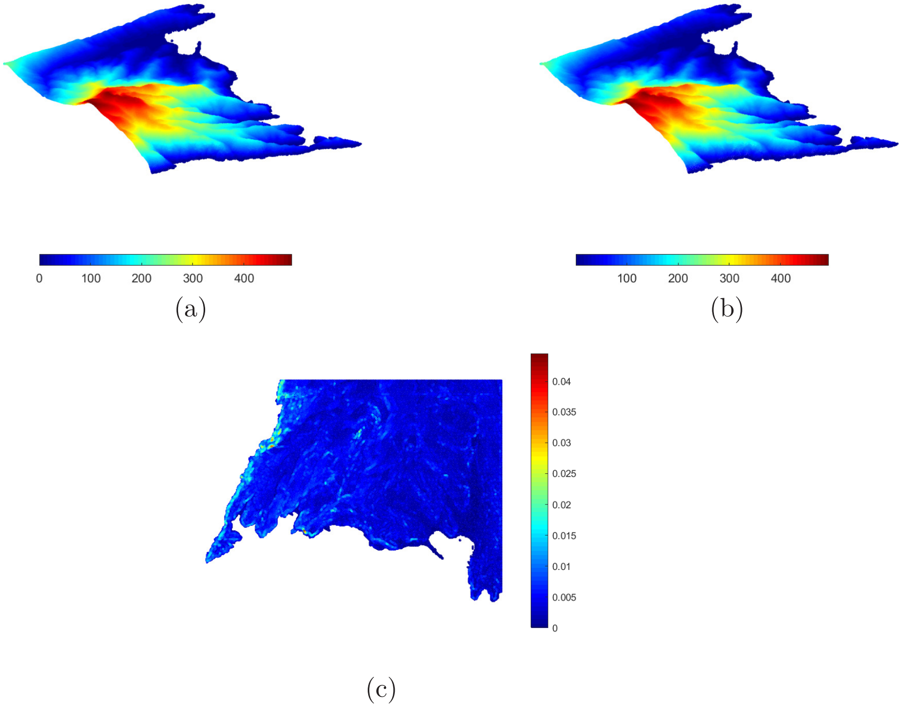


Figure II.7: Portofino, Liguria, Italy. A data point cloud from the given region of interest (a) is approximated via a IDW weight (b). The colors represent the elevation and vary from blue (low elevation) to red (high elevation). A graphical representation of the punctual error, normalized by the maximum elevation, is provided in (c). The statistics for the error are: min=0.0000, max=0.0445, mean=0.0021, median=0.0017, RMSE=0.0029 and std=0.0019.

The method has been also tested for the approximation of the boundary of 3D models. As currently stated, wQISA is suitable to approximate surface portions that can be represented in a local Cartesian coordinate system in the form $z = f(x,y)$. Therefore, the object surface needs a subdivision into charts, for instance following the approaches in [42, 50]. Once the charts have been obtained, we compute the desired representation adopting as z value the height value of the chart with respect to its best fitting regression plane [54]. Figure II.8 visually show some details of two wQISA approximations for 3D points clouds: the surfaces in the boxes approximate the regions pointed by the (light blue) lines. These models come from the Visionair Shape Repository, VSR [53]. Given the low level of noise, a pure 1-NN weight function is here tested. The approximation shows a correct recovery of the main details of the artefact. Nevertheless, the feeble details are lost as an effect of the smoothing effect of this weight function.

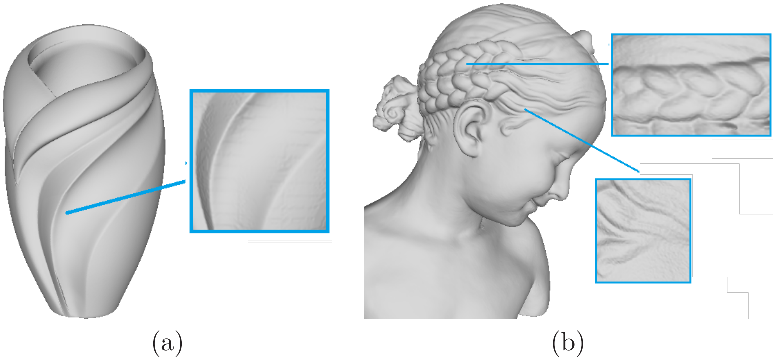


Figure II.8: Examples on two 3D models. For each model we highlight some details of the wQISA approximation computed by a 1-NN weight function. The statistics of the relative punctual error are: for the vase, MSE=$4.1884e-06$ and std=0.0015; for the curl, MSE=$4.1493e-06$ and std=0.0016; for the tress, MSE=$3.7918e-05$ and std=0.0057.

II.4.4 Approximation of surface properties

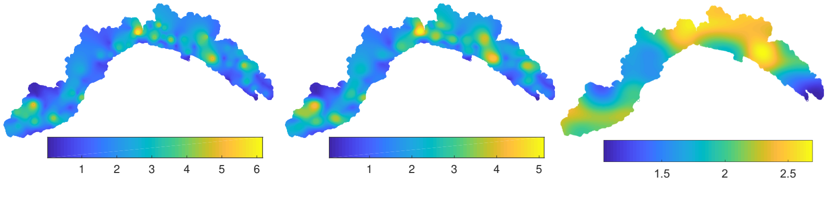
As a further case study, we propose the approximation of a precipitation event over the Liguria region. To this purpose we consider an event occurred between January 16 and 20, 2014, which was responsible of heavy rain for about five days over all the Liguria region. The data we are considering were gathered from rain gauges maintained by Regione Liguria. The network is spread over the whole region, with 143 measure stations. These data come from the use case adopted for the comparison of six rainfall precipitation methods in [43].

Here, we compare wQISA with k-NN weight functions with two other methods: radial basis functions (RBF) with Gaussian kernel, as considered in [43], and the Multilevel B-splines Approximation (MBA) [36]. In the RBF implementation a global support [14] is adopted (all the 143 rain samples are considered) and a direct solver is applied to the linear system, which is symmetric and positive-definite. The MBA approximation is obtained with the default settings of the implementation of the Geometry Group at SINTEF Digital, which is freely available at: https://github.com/orochi663/MBA.

Table II.2: Statistics for the error distribution of the cross validation.

Method	Min [mm]	Max [mm]	Mean [mm]	Median [mm]	Std [mm]	MSE [mm^2]
RBF	0.0317	2.9363	1.0903	1.0070	0.7830	1.7973
MBA	0.0341	3.3489	1.1667	1.0243	0.8767	2.1969
wQISA	0.0471	2.8293	0.9883	0.9013	0.6885	1.4657

A quantitative comparison is provided in Table II.2 and computed by performing 5 times a 5-fold cross-validation on each method. For more details, we refer once again the reader to [29] (chapter 7). The optimal parameters for a k-NN wQISA are chosen by minimizing the average MSE and are: $k = 9$, with 10 inner knots for each direction. In Figure II.9 we sample the precipitation fields approximated with the three methods in a set of points, representing the Liguria region. Although our approximation looks smoother and less detailed, it has in practice a better generalization performance as a learning method – that is a better prediction capability on independent test data. An implementation of the wQISA method for rainfall data with the choice of the optimal values for the k parameter and the cross-validation tests reported in Table II.2 is freely availale at: https://github.com/rea1991/wQISA.



(a) (b) (c)

Figure II.9: Rainfall approximation with RBF (a), MBA (b) and wQISA (c).

II.5 Concluding remarks and future perspectives

We defined a novel quasi-interpolant reconstruction technique (wQISA), specifically designed to handle large and noisy point sets, even when equipped with outliers. The robustness and the versatility of the method are theoretically discussed from the point of view of numerical analysis (Sections II.2.3) and probability theory (Section II.3). Numerical examples are provided in Section II.4.

In this work we presented a quasi-interpolant scheme that applies to point clouds even equipped with noise and outliers. Our definition of the control point estimators combines computational efficiency with the possibility to work with different types of noise, as well as a reduced sensitivity to outliers. The computational complexity is, in fact, comparable to that of a weighted average. We gave evidence of the approximation effectiveness of the method over a wide range of real data and application domains.

As a further development of the method, we think it is possible to extend wQISA to more general refinement schemes, for instance opportunely selecting the point neighbours [37], such as in the case of LR B-splines or THB-splines [18, 34]. This is particularly relevant because these locally refining schemes naturally deal with isogeometric computations and simulation and offers the valuable perspective to practically adopt this work for Computer Aided Design and Manufacturing (CAD/CAM), Finite Element Analysis and IsoGeometric Analysis [26, 33, 41].

Acknowledgements. The authors thank Dr. Bianca Falcidieno and Dr. Michela Spagnuolo for the fruitful discussions; Dr. Oliver J. D. Barrowclough, Dr. Tor Dokken and Dr. Georg Muntingh for their concern as supervisors; the reviewers, for their positive suggestions, which have significantly contributed to extend the references.

Funding: this project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675789. This work has been co-financed by the "POR FSE, Programma Operativo Regione Liguria" 2014-2020, No RLOF18ASSRIC/68/1, and partially developed in the CNR-IMATI activities DIT.AD021.080.001 and DIT.AD009.091.001.

References

[1] Amir, A. and Levin, D. "Quasi-interpolation and outliers removal". In: *Numerical Algorithms* vol. 78, no. 3 (July 2018), pp. 805–825.

[2] Anile, A. M. et al. "Modeling unc ertain data with fuzzy B-splines". In: *Fuzzy Sets and Systems* vol. 113, no. 3 (2000), pp. 397–410.

[3] Beatson, R. and Powell, M. "Univariate multiquadric approximation: quasi-interpolation to scattered data". In: *Constructive Approximation* vol. 8 (1992), pp. 275–288.

[4] Boffi, D., Brezzi, F., and Fortin, M. *Mixed Finite Element Methods and Applications*. Vol. 44. Springer Series in Computational Mathematics. Springer, 2013.

[5] Boor, C. de and Fix, G. J. "Spline approximation by quasi-interpolants". In: *Journal of Approximation Theory* vol. 8, no. 1 (1973), pp. 19–45.

[6] Bracco, C., Giannelli, C., and Sestini, A. “Adaptive scattered data fitting by extension of local approximations to hierarchical splines”. In: *Computer Aided Geometric Design* vol. 52-53 (2017), pp. 90–105.

[7] Bracco, C. et al. “Generalized spline spaces over T-meshes: Dimension formula and locally refined generalized B-splines”. In: *Applied Mathematics and Computation* vol. 272 (2016), pp. 187–198.

[8] Briand, T. and Monasse, P. “Theory and Practice of Image B-Spline Interpolation”. In: *Image Processing On Line* vol. 8 (2018), pp. 99–141.

[9] Buffa, A. and Sangalli, G. *IsoGeometric Analysis: A New Paradigm in the Numerical Approximation of PDEs.* second. New York, NY, USA: Springer, 2012.

[10] Buhmann, M. D. “On Quasi-interpolation with Radial Basis Functions”. In: *Journal of Approximation Theory* vol. 72, no. 1 (1993), pp. 103–130.

[11] Buhmann, M. D. and Dai, F. “Pointwise approximation with quasi-interpolation by radial basis functions”. In: *Journal of Approximation Theory* vol. 192 (2015), pp. 156–192.

[12] Canny, J. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 8, no. 6 (June 1986), pp. 679–698.

[13] Cao, L. “Data Science: A Comprehensive Overview”. In: *ACM Computing Surveys* vol. 50, no. 3 (June 2017), 43:1–43:42.

[14] Carr, J. C. et al. “Reconstruction and Representation of 3D Objects with Radial Basis Functions”. In: *proceedings of SIGGRAPH ’01*. New York, NY, USA: ACM, 2001, pp. 67–76.

[15] Cheng, F. and Barsky, B. A. “Interproximation: interpolation and approximation using cubic spline curves”. In: *Computer-Aided Design* vol. 23, no. 10 (1991), pp. 700–706.

[16] Cohen, E., Lyche, T., and Riesenfeld, R. R. “Discrete B-splines and subdivision techniques in computer-aided design and computer graphics”. In: *Computer Graphics & Image Processing* vol. 14, no. 2 (1980), pp. 87–111.

[17] Deza, M. M. and Deza, E. *Encyclopedia of Distances.* Springer Berlin Heidelberg, 2009.

[18] Dokken, T., Lyche, T., and Pettersen, K. F. “Polynomial splines over locally refined box-partitions”. In: *Computer Aided Geometric Design* vol. 30, no. 3 (2013), pp. 331–356.

[19] Farin, G. *Curves and Surfaces for Computer Aided Geometric Design (3rd Ed.): A Practical Guide.* San Diego, CA, USA: Academic Press Professional, Inc., 1993.

[20] Forsey, D. R. and Bartels, R. H. “Hierarchical B-spline refinement”. In: *ACM SIGGRAPH Computer Graphics* (1988), pp. 205–212.

[21] Friedman, J. H., Bentley, J. L., and Finkel, R. A. “An Algorithm for Finding Best Matches in Logarithmic Expected Time”. In: *ACM Trans. Math. Softw.* vol. 3, no. 3 (Sept. 1977), pp. 209–226.

[22] Gao, W. and Wu, Z. “Approximation orders and shape preserving properties of the multiquadric trigonometric B-spline quasi-interpolant”. In: *Computers & Mathematics with Applications* vol. 69, no. 7 (2015), pp. 696–707.

[23] Gao, W. et al. “Multivariate Monte Carlo approximation based on scattered data”. In: *SIAM Journal on Scientific computing* vol. 32, no. 4 (2020), A2262–A2280.

[24] Gao, W. et al. “Optimality and regularization properties of quasi-interpolation: both deterministic and stochastic perspectives”. In: *SIAM Journal on numerical analysis* vol. 58, no. 4 (2020), pp. 2059–2078.

[25] Gao, W. and Zhang, R. “Multiquadric trigonometric spline quasi-interpolation for numerical differentiation of noisy data: a stochastic perspective”. In: *Numerical Algorithms* vol. 77, no. 1 (Jan. 2018), pp. 243–259.

[26] Giannelli, C. et al. “THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 299 (2016), pp. 337–365.

[27] Goodman, T. N. T. “Shape Preserving Representations”. In: *Mathematical Methods in Computer Aided Geometric Design* (1989), pp. 333–351.

[28] Gregory, J. A. “Shape Preserving Spline Interpolation”. In: *Computer Aided Design* vol. 18, no. 1 (Jan. 1986), pp. 53–57.

[29] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* second. Springer, 2009.

[30] Hughes, T. J. R., Cottrell, J. A., and Bazilevs, Y. “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 194, no. 39 (2005), pp. 4135–4195.

[31] *iQmulus: A High-volume Fusion and Analysis Platform for Geospatial Point Clouds, Coverages and Volumetric Data Sets.* http://iqmulus.eu/. 2012.

[32] Jiang, Z.-W. et al. “High accuracy multiquadric quasi-interpolation”. In: *Applied Mathematical Modelling* vol. 35, no. 5 (2011), pp. 2185–2195.

[33] Johannessen, K. A., Kvamsdal, T., and Dokken, T. “Isogeometric analysis using LR B-splines”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 269 (2014), pp. 471–514.

[34] Johannessen, K. A., Remonato, F., and Kvamsdal, T. “On the similarities and differences between Classical Hierarchical, Truncated Hierarchical and LR B-splines”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 291 (2015), pp. 64–101.

[35] Karageorghis, V. and Karageorghis, J. *The coroplastic art of ancient Cyprus*. English. Nicosia : A.G. Leventis Foundation, 1991.

[36] Lee, S., Wolberg, G., and Shin, S. Y. “Scattered data interpolation with multilevel Bsplines”. In: *IEEE Transactions on Visualization and Computer Graphics* vol. 3, no. 3 (July 1997), pp. 228–244.

[37] Lenarduzzi, L. “Practical selection of neighbourhoods for local regression in the bivariate case”. In: *Numerical Algorithms* vol. 5, no. 4 (1993), pp. 205–213.

[38] Long, X. Y. et al. “Unified uncertainty analysis under probabilistic, evidence, fuzzy and interval uncertainties”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 355 (2019), pp. 1–26.

[39] Lyche, T. and Mørken, K. *Spline Methods Draft*. 2011.

[40] Moscoso Thompson, E. et al. “SHREC’19 track: Feature Curve Extraction on Triangle Meshes”. In: *Eurographics Workshop on 3D Object Retrieval*. Ed. by Biasotti, S. et al. 2019.

[41] Occelli, M. et al. “LR B-Splines implementation in the Altair RadiossTM solver for explicit dynamics IsoGeometric Analysis”. In: *Advances in Engineering Software* (Mar. 2019).

[42] Ohtake, Y. et al. “Multi-level Partition of Unity Implicits”. In: *ACM Transactions on Graphics* vol. 22, no. 3 (July 2003), pp. 463–470.

[43] Patané, G. et al. “Comparing methods for the approximation of rainfall fields in environmental applications”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* vol. 127 (2017), pp. 57–72.

[44] Patrizi, F. et al. “Adaptive refinement with locally linearly independent LR B-splines: Theory and applications”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 369 (2020), p. 113230.

[45] Piegl, L. and Tiller, W. *The NURBS Book*. New York, NY, USA: Springer-Verlag, 1996.

[46] Raffo, A. and Biasotti, S. “Data-driven quasi-interpolant spline surfaces for point cloud approximation”. In: *Computers & Graphics* vol. 89 (2020), pp. 144–155.

[47] Sablonnière, P. “Univariate spline quasi-interpolants and applications to numerical analysis”. In: *Rendiconti del Seminario Matematico*. Vol. 63. 2. Università degli studi di Torino / Politecnico di Torino, 2005, pp. 211–222.

[48] Schumaker, L. L. *Spline Functions: Basic Theory*. Cambridge University Press, 2007.

[49] Sederberg, T. W. et al. “T-Splines and T-NURCCS”. In: *ACM Transactions on Graphics* vol. 22, no. 3 (2003), pp. 477–484.

[50] Sorgente, T. et al. “Topology-driven shape chartification”. In: *Computer Aided Geometric Design* vol. 65 (2018), pp. 13–28.

[51] Speleers, H. and Manni, C. “Effortless quasi-interpolation in hierarchical spaces”. In: *Numerische Mathematik* vol. 132, no. 1 (Jan. 2016), pp. 155–184.

[52] Taha, A. A. and Hanbury, A. “Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool”. In: *BMC Medical Imaging.* 2015.

[53] *The Shape Repository.* http://visionair.ge.imati.cnr.it/ontologies/shapes/. 2011.

[54] Torrente, M.-L., Biasotti, S., and Falcidieno, B. “Recognition of feature curves on 3D shapes using an algebraic approach to Hough transforms”. In: *Pattern Recognition* vol. 73 (2018), pp. 111–130.

[55] Wu, Z. and Schaback, R. “Shape preserving properties and convergence of univariate multiquadric quasi-interpolation”. In: *Acta Mathematicae Applicatae Sinica* vol. 10 (1994), pp. 441–446.

Appendix II.A The univariate case

We will suppose – up to a rotation – that the point cloud $\mathcal{P}$ can be locally represented by a function of the form $f : [a, b] \subset \mathbb{R} \to \mathbb{R}$.

Definition II.7. Let $\mathcal{P} \subset \mathbb{R}^2$ be a point cloud, $p \in \mathbb{N}^*$ and $\mathbf{x} = [x_1, \ldots, x_{n+p+1}]$ a $(p+1)$-regular (global) knot vector with fixed boundary knots $x_{p+1} = a$ and $x_{n+1} = b$. The *Weighted Quasi Interpolant Spline Approximation* of degree p to the point cloud $\mathcal{P}$ over the knot vector $\mathbf{x}$ is defined by

$$f_w(x) := \sum_{i=1}^{n} \hat{y}_w(\xi^{(i)}) B[\mathbf{x}^{(i)}](x), \tag{II.21}$$

where $\xi^{(i)} := (x_i + \ldots + x_{i+p})/p$ are the *knot averages* and

$$\hat{y}_w(t) := \frac{\sum\limits_{(x,y)\in\mathcal{P}} y \cdot w_t(x)}{\sum\limits_{(x,y)\in\mathcal{P}} w_t(x)}$$

are the *control points estimators* of *weight functions* $w_t : \mathbb{R} \to [0, +\infty)$.

II.A.1 Properties

II.A.1.1 Global and local bounds

Proposition II.2 (Global bounds)**.** *Let $\mathcal{P} \subset \mathbb{R}^2$ be a point cloud. Given $y_{min}, y_{max} \in \mathbb{R}$ that satisfy*

$$y_{min} \leq y \leq y_{max}, \qquad \textit{for all } (x,y) \in \mathcal{P},$$

then the weighted quasi interpolant spline approximation to $\mathcal{P}$ from some spline space $\mathbb{S}_{p,\mathbf{x}}$ and some weight function w has the same bounds

$$y_{min} \leq f_w(x) \leq y_{max}, \qquad \textit{for all } x \in \mathbb{R}.$$

Proof. From the partition of unity property of a B-spline basis, it follows that

$$\begin{array}{ccccc} \min_i \hat{y}_w(\xi^{(i)}) & \leq & \sum_{i=1}^{n} \hat{y}_w(\xi^{(i)}) B[\mathbf{x}^{(i)}](x) & \leq & \max_i \hat{y}_w(\xi^{(i)}) \\ \rotatebox{90}{$\geq$}\,\textcircled{1} & & \rotatebox{90}{$=:$} & & \rotatebox{90}{$\leq$}\,\textcircled{2} \\ y_{min} & & f_w(x) & & y_{max} \end{array} \tag{II.22}$$

where the inequalities ① and ② are a direct consequence of defining $\hat{y}_w$ by means of a convex combination. ■

The bounds of Proposition II.2 can potentially lead to local bounds. We discuss this situation in Corollary II.2.

Corollary II.2 (Local bounds)**.** *Let $\mathcal{P} \subset \mathbb{R}^2$ be a point cloud. If $x \in [x_\mu, x_{\mu+1})$ for some μ in the range $p+1 \leq \mu \leq n$, then*

$$\alpha(\mu) \leq f_w(x) \leq \beta(\mu),$$

for some $\alpha(\mu), \beta(\mu)$ which belong to $[y_{min}, y_{max}]$.

Proof. By using the property of local support for B-splines, it follows that

$$f_w(x) = \sum_{i=\mu-p}^{\mu} \hat{y}_w(\xi^{(i)}) B[\mathbf{x}^{(i)}](x)$$

over $[x_\mu, x_{\mu+1})$. Thus, we can re-write the chain of inequalities (**??**) as

$$\begin{array}{ccc} \min\limits_{i=\mu-p,\ldots,\mu} \hat{y}_w(\xi^{(i)}) & \leq f_w(x) \leq & \max\limits_{i=\mu-p,\ldots,\mu} \hat{y}_w(\xi^{(i)}) \\ \rotatebox{90}{$\geq$}\,\textcircled{3} & & \rotatebox{90}{$\leq$}\,\textcircled{4} \\ \min\left\{ y \text{ s.t. } (x,y) \in \bigcup\limits_{i=\mu-p}^{\mu} \mathcal{P}_i \right\} & & \max\left\{ y \text{ s.t. } (x,y) \in \bigcup\limits_{i=\mu-p}^{\mu} \mathcal{P}_i \right\} \\ \rotatebox{90}{$=:$} & & \rotatebox{90}{$=:$} \\ \alpha(\mu) & & \beta(\mu) \end{array} \tag{II.23}$$

where

$$\mathcal{P}_i := \bigcup_{i=\mu-p,\ldots,\mu} \left\{ \operatorname{supp}\left(w_{\xi^{(i)}}(\cdot) \right) \right\} \cap \mathcal{P}.$$

■

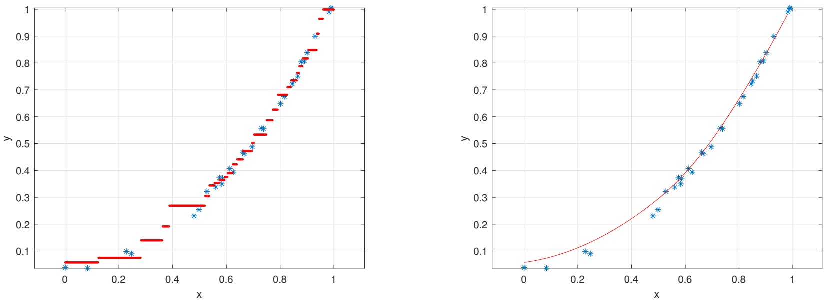
Notice that the set of points which are effectively used to compute the approximation, i.e.,

$$\mathcal{P}^* := \bigcup_{i=p+1,\dots,n} \mathcal{P}_i$$

may be a proper subset of $\mathcal{P}$.

II.A.1.2 Preservation of monotonicity

Definition II.8 (w-monotonicity)**.** Let $w_t : \mathbb{R} \to [0, +\infty)$ be a family of weight functions, where $t \in \mathbb{R}$. A point cloud $\mathcal{P} \subset \mathbb{R}^2$ is said to be *w-increasing* if for all $x_1 \leq x_2$, $\hat{y}_w(x_1) \leq \hat{y}_w(x_2)$. $\mathcal{P}$ is said to be *w-decreasing* if for all $x_1 \leq x_2$, $\hat{y}_w(x_1) \geq \hat{y}_w(x_2)$.



(a) (b)

Figure II.10: w-monotonicity and its preservation. Figure (a) shows an example of an estimator $\hat{y}_w : \mathbb{R} \to \mathbb{R}$ (in red) for a given point cloud (in blue) with respect to a 3-NN weight function. Figure (b) graphically compares the original point cloud (in blue) to its wQISA (in red).

The key ingredient to prove the preservation of monotonicity through our method is the following lemma.

Lemma II.2. *Let $p \in \mathbb{N}^*$ and $\mathbf{x} = [x_1, \dots, x_{n+p+1}]$ be a $(p+1)$-regular (global) knot vector with fixed boundary knots $x_{p+1} = a$ and $x_{n+1} = b$. In addition, let $f = \sum_{i=1}^{n} c_i B[\mathbf{x}^{(i)}] \in \mathbb{S}_{p,\mathbf{x}}$. If the sequence of coefficients $\{c_i\}_{i=1}^{n}$ is increasing (decreasing) then f is increasing (decreasing).*

Proof. The Lemma is proven in [39], pp. 114–115. ■

Proposition II.3. *Let $\mathcal{P} \subset \mathbb{R}^2$ be a point cloud, $p \in \mathbb{N}^*$ and $\mathbf{x} = [x_1, \dots, x_{n+p+1}]$ be a $(p+1)$-regular (global) knot vector with fixed boundary knots $x_{p+1} = a$ and $x_{n+1} = b$. If $\mathcal{P}$ is w-increasing (decreasing) then f_w is also increasing (decreasing).*

Proof. By definition of w-increasing (decreasing) point cloud, the sequence of control points $\{\hat{y}_w(\xi^{(i)})\}_{i=1}^n$ is increasing (decreasing). By Lemma II.2, this is sufficient to conclude that f_w is increasing (decreasing). ■

II.A.1.3 Preservation of convexity

Definition II.9 (w-convexity)**.** Let $w_t : \mathbb{R} \to [0, +\infty)$ be a family of weight functions, where $t \in \mathbb{R}$. A point cloud $\mathcal{P} \subset \mathbb{R}^2$ is said to be *w-convex* if for all $x_1 \leq x_2$ and for any $\lambda \in [0, 1]$,

$$\hat{y}_w((1-\lambda)x_1 + \lambda x_2) \leq (1-\lambda)\hat{y}_w(x_1) + \lambda \hat{y}_w(x_2).$$

$\mathcal{P}$ is said to be *w-concave* if $\mathcal{P}_- := \{(x, -y) | (x, y) \in \mathcal{P}\}$ is w-convex.

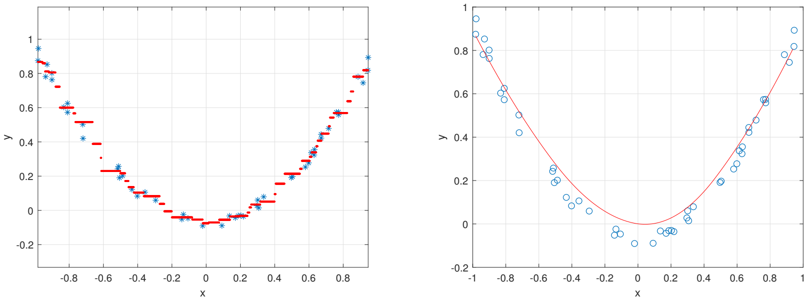


Figure II.11: w-convexity and its preservation. Figure (a) shows an example of an estimator $\hat{y}_w : \mathbb{R} \to \mathbb{R}$ (in red) for a given point cloud (in blue) with respect to a 3-NN weight function. Figure (b) graphically compares the original point cloud (in blue) to its wQISA (in red).

The preservation of convexity is a consequence of the following lemma.

Lemma II.3. *Let $p \in \mathbb{N}^*$ and $\mathbf{x} = [x_1, \ldots, x_{n+p+1}]$ be a $(p+1)$-regular (global) knot vector with fixed boundary knots $x_{p+1} = a$ and $x_{n+1} = b$. Lastly, let $f = \sum_{i=1}^n c_i B[\mathbf{x}^{(i)}] \in \mathbb{S}_{p,\mathbf{x}}$. Define Δc_i by*

$$\Delta c_i := \begin{cases} \dfrac{c_i - c_{i-1}}{x_{i+p} - x_i}, & \text{if } x_i < x_{i+p} \\ \Delta c_{i-1} & \text{if } x_i = x_{i+p} \end{cases}$$

for $i = 2, \ldots, n$. Then f is convex on $[x_{p+1}, x_{n+1}]$ if it is continuous and if the sequence $\{\Delta c_i\}_{i=2}^n$ is increasing.

Proof. See [39], p. 118. ■

Proposition II.4. *Let $\mathcal{P} \subset \mathbb{R}^2$ be a point cloud, $p \in \mathbb{N}^*$ and $\mathbf{x} = [x_1, \ldots, x_{n+p+1}]$ be a $(p+1)$-regular (global) knot vector with fixed boundary knots $x_{p+1} = a$ and $x_{n+1} = b$. If $\mathcal{P}$ is w-convex (concave) then f_w is also convex (concave).*

Proof. Let

$$\Delta c_i := \frac{\hat{y}_w(\xi^{(i)}) - \hat{y}_w(\xi^{(i-1)})}{x_{i+p} - x_i} = \frac{\hat{y}_w(\xi^{(i)}) - \hat{y}_w(\xi^{(i-1)})}{(\xi^{(i)} - \xi^{(i-1)})p}$$

with $x_i < x_{i+p}$. Since $\mathcal{P}$ is w-convex then these differences must be increasing and consequently f_w is convex by Lemma II.3. ■

Authors' addresses

Andrea Raffo Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, andrea.raffo@ge.imati.cnr.it

Silvia M. Biasotti Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, silvia.biasotti@ge.imati.cnr.it

Paper III

Data-driven quasi-interpolant spline surfaces for point cloud approximation

Andrea Raffo, Silvia Biasotti

Published in *Computers & Graphics*, June 2020, volume 87, pp. 144–155. DOI: 10.1016/j.cag.2020.05.004.

Abstract

In this paper we investigate a local surface approximation, the *Weighted Quasi Interpolant Spline Approximation* (wQISA), specifically designed for large and noisy point clouds. We briefly describe the properties of the wQISA representation and introduce a novel data-driven implementation, which combines prediction capability and complexity efficiency. We provide an extended comparative analysis with other continuous approximations on real data, including different types of surfaces and levels of noise, such as 3D models, terrain data and digital environmental data.
Keywords: spline methods, quasi-interpolation, point clouds, noise, data-driven model assessment.

Contents

III.1 Introduction

Due to the recent progress in the acquisition by laser scanners, photogrammetry and diagnostic devices and the popularity of remote sensing technologies, there has been an exponential growth in the availability of data and the need of efficient representations. A good representation model must be, at the same time, *efficient*, that is based on the minimum amount of data, yet *effective*, that is able to support conservative conclusions and keep as much information as possible. To handle large data volumes it is often necessary to adopt approximation strategies so that a single point becomes representative of a region or a set of properties [1, 12]. Given a set of measurements (and their spatial locations), its model approximation has to permit deducing information about the process that generated those data, even at locations different from those at which the measurements were obtained. Surface reconstruction, terrain elevation estimation and the approximation of rainfall and pollution fields over Digital Elevation Models (DEMs) are all examples of spatial data approximation. Moreover, an efficient approximation allows the recovery of the digital representation of a physical shape that commonly contains a variety of properties and defects, such as: geometric features; noise and outliers; perturbations introduced when the acquisition conditions are not optimal (low resolution of instruments, motions, etc.) or different acquisition techniques collect data sets of different resolution (laser scans or photogrammetry); incomplete data (e.g, in the acquisition of broken artefacts or scans of objects partially occluded [21]).

Quasi-interpolation schemes [4, 40] are popular for data approximation because, unlike traditional least squares approximations, they do not require solving a linear system. We use B-splines as basis functions because they are computationally convenient, for instance with respect to the common radial basis functions, as (piecewise) polynomial bases require low-order integration schemes to be exactly computed. The use of piecewise algebraic approximations makes our method suitable also to CAD applications, where B-splines and NURBS are *de facto* the standard tools. Moreover, the treatment of essential boundary conditions is more natural for structured approaches like the spline-based ones than in meshless methods, as it relies on the number of repetition of each knot value.

The wQISA implementation proposed in this paper is specifically designed to define powerful prediction methods from low quality points. Starting from the theoretical wQISA definition given in [38] for a single tensor-product mesh, in this work we derive a multi-level approximation algorithm and provide an extended comparative analysis with other methods in the literature. The main contributions of the paper include: a detailed description of the wQISA method when used for surface approximation; a multi-level approximation algorithm based on a data-driven definition of the weight functions; an extensive comparison of the wQISA outcome with other well-known continuous approximation methods.

The remainder of the paper is organized as follows. Section III.2 overviews the literature on data approximation focusing on methods related to continuous surface approximation. Section III.3 overviews the wQISA method and its properties, together with a multi-level implementation of the method based on a data-driven mesh refinement strategy. Section III.5 extensively compares the wQISA outcome on different real use cases, with respect to a number of well-known approximation methods and a set of performance indicators that are detailed in Section III.4. Discussions and concluding remarks are provided in Section III.6.

III.2 Previous work

The literature on data approximation is vast and we cannot do justice to all contributions. We limit our review to methods whose output satisfies some smoothness requirement, either local or global. For a more complete list of methods related to our use cases, we refer to [3] for a recent survey on surface reconstruction methods, to [20] for an overview on methods for modelling terrain data and to [35] for a comparative analysis of methods for approximating rainfall data.

Meshless methods. Kriging is a geo-statistical approach largely used to approximate remote sensing measurements [34]. In particular, (ordinary) Kriging incorporates correlation information into data approximation through a variogram model. The main limitation of ordinary Kriging is the limited scalability; indeed, Kriging's computation scales quadratically with respect to the number of observations. The Moving Least Square (MLS) approach is largely adopted for surface reconstruction [2, 14, 43]. The basic idea behind MLS is to approximate the surface in the neighbour of a point with the tangent plane in that point. Several variations have been introduced, for instance Feng et al. [13] devised a model that uses multiple curves/surfaces approximation, that allows separating mixed scanning points received from a thin-wall object and consists of a second-order extension of the method in [31]. Other implicit approximation techniques express the data as linear combination of basis elements. For instance, radial basis functions (RBFs) are central for scattered data approximation [17]. The quality of the interpolation depends on the choice of the basis function. Originally introduced as a global method [7, 41], RBFs have been adapted to compact support [15]. A combination of MLS and RBF is presented in [37]. Wavelets and wavelet transforms [9] have been applied for data compression and noise reduction by truncating the wavelet decomposition. Among other techniques, we mention Poisson based methods, which have been applied for example to surface reconstruction from point sets [28].

Mesh-based methods. Tensor-product spline surfaces (piecewise polynomial or NURBS) are a well established representation for modelling smooth shapes. Several local refinement methods have been proposed to overcome the limited mesh adaptivity to the data, such as T-splines [42], locally refined (LR) B-splines [11, 27] and (truncated) hierarchical B-splines [22]. Forsey and Bartels use hierarchical B-splines for interpolation and least square approximation of gridded data [16]. Lee et al. [30] introduce a multi-level quasi-interpolant, the Multilevel B-spline Approximation (MBA), where the coefficients depends on values of tensor product B-splines defined on lattices. Greiner and Horman [24] address approximation and interpolation of data by global least squares over hierarchical tensor-product spline spaces. Kiss et al [29] adapt this approach to handle THB-splines. Davydov et al. [10] introduce a two-stage method based on extended B-splines with focus on curvilinear domains. Skytt et al. propose least square fitting and MBA of LR B-spline surfaces for the approximation of terrain [44] and bathymetry data [45]. A comparison between RBFs and LR B-splines approximations has been performed by Patané et al. [36](Chapter 2). A data reduction scheme for model simplification is introduced in [5].

III.3 Weighted Quasi Interpolant Spline Approximations

Quasi-interpolation is a low-cost and accurate procedure in function approximation theory. The term quasi-interpolation has been interpreted differently according to the context. We follow the Cheney [8] definition of a quasi interpolant as any linear operator L of the form

$$Lf := \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f(x_i, y_j) g_{i,j}, \tag{III.1}$$

where $f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ is a function being approximated, $n_x \in \mathbb{N} \cup \{+\infty\} \ni n_y$, (x_i, y_j) are given *nodes* and $g_{i,j} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ are functions at our disposal. Differently from classic Quasi Interpolation methods that focus to function approximation, the Weighted Quasi Intepolant Spline Approximation (wQISA, [38]) aims at point cloud approximation, where the data are assumed to be affected by noise, outliers or partially missing. In this Section we summarise the wQISA concept and list its theoretical properties. Then, we sketch a data-driven implementation of the wQISA algorithm, which allows to deal with both approximation and prediction problems. Finally, we discuss its computational complexity.

III.3.1 Preliminary concepts

In a general pipeline for approximation, we envisage that a complex surface is decomposed in multi-charts, e.g., [33, 46] and then the approximated patches are stitched together. In this paper we focus on the quality of local approximations, reserving to future investigations the gluing of multiple patches. From the mathematical point of view, every surface can be locally projected onto a plane by using an injective map and, if locally regular, it can be expressed in local coordinates as $(x, y, z(x, y))$. Given a degree p, a knot vector is said to be $(p + 1)$-*regular* if no knot occurs more than $p + 1$ times and each boundary knot occurs exactly $p + 1$ times.

Definition III.1. Let $\mathcal{P} \subset \mathbb{R}^3$ be a point cloud and $\mathbf{p} = (p_x, p_y) \in \mathbb{N}^* \times \mathbb{N}^*$ be a bi-degree. Let $\mathbf{x}$ be a $(p_x + 1)$-regular knot vector with boundary knots $x_{p_x} = a_1$ and $x_{n_x} = b_1$ and let $\mathbf{y}$ be a $(p_y + 1)$-regular knot vector with boundary knots $y_{p_y} = a_2$ and $x_{n_y} = b_2$. The *Weighted Quasi Interpolant Spline Approximation* of bi-degree $\mathbf{p}$ to the point cloud $\mathcal{P}$ over the knot vectors $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$f_w(x, y) := \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \hat{z}_w(x_i^*, y_j^*) \cdot B[\mathbf{x}_i, \mathbf{y}_j](x, y), \tag{III.2}$$

where $x_i^* := (x_i + \ldots + x_{i+p_x})/p_x$ and $y_j^* := (y_j + \ldots + y_{j+p_y})/p_y$ are the *knot averages*, the expression

$$\hat{z}_w(u,v) := \frac{\sum\limits_{(x,y,z)\in\mathcal{P}} z \cdot w(x,y,u,v)}{\sum\limits_{(x,y,z)\in\mathcal{P}} w(x,y,u,v)} \tag{III.3}$$

is the *control points estimator* with weight function $w : \mathbb{R}^2 \times \mathbb{R}^2 \to [0,+\infty)$ and $B[\mathbf{x}_i, \mathbf{y}_j]$ denotes the tensor product B-spline of bi-degree $\mathbf{p}$ which is uniquely determined by the *local knot vectors* $\mathbf{x}_i = [x_i, \ldots, x_{i+p_x+1}]$ and $\mathbf{y}_j = [y_j, \ldots, y_{j+p_y+1}]$.

Remark III.1. In Definition III.1, a wQISA depends on the following inputs: a point cloud, a tensor mesh (uniquely defined by a bi-degree and two knot vectors) and a weight function.

Remark III.2. The weight function w defines a *window* around each point (u,v) and the literature is rich of possible choices, including functions with global and local support. Very popular examples for the function w are:

- Indicator of radius $r > 0$:

$$w(x,y,u,v) := \mathbb{1}_{||(x-u,y-v)||_2 \leq r}, \tag{III.4}$$

- Gaussian of standard deviation σ:

$$w(x,y,u,v) := e^{-||(x-u,y-v)||_2/2\sigma^2}, \tag{III.5}$$

- k-Nearest Neighbors (k-NN):

$$w(x,y,u,v) := \begin{cases} 1/k, & \text{if } (x,y) \in N_k(u,v) \\ 0, & \text{otherwise} \end{cases}, \tag{III.6}$$

  where $k \in \mathbb{N}^*$ and $N_k(u,v)$ denotes the neighborhood of (u,v) defined by the k closest points of the point cloud.
- Inverse Distance Weight (IDW):

$$w(x,y,u,v) := \begin{cases} \dfrac{1}{||(x,y)-(u,v)||_2}, & \text{if } |C_{(u,v)}| = 0 \\ \begin{cases} \dfrac{1}{|C_{(u,v)}|}, & \forall (x,y) = (u,v) \\ 0, & \text{else} \end{cases}, & \text{if } |C_{(u,v)}| \neq 0 \end{cases} \tag{III.7}$$

  where

$$|C_{(u,v)}| := \{(x,y,z) \in \mathcal{P} \text{ s.t. } (x,y) = (u,v)\}.$$

Details on how to fix the free parameters, if any, are provided in Section III.3.3.3.

Figure III.1 shows the stability of wQISA (until failure) against increasing amount of noise and outliers. In these simulations, a k-NN weight is filtered from outliers by using quartiles.

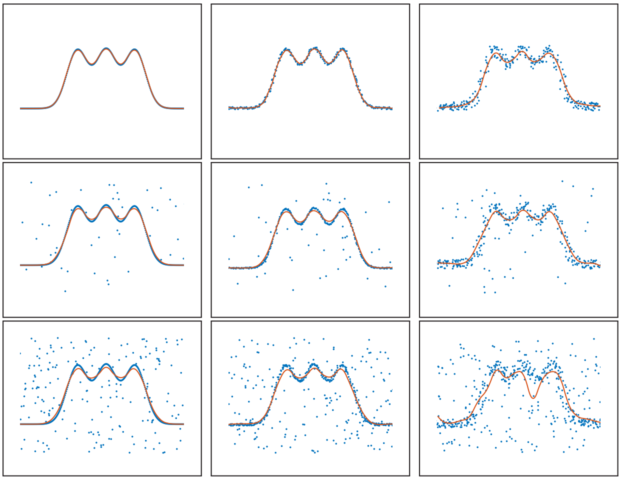


Figure III.1: Noise and outliers robustness. The input point cloud is represented in blue, the output approximation is displayed in red. The noise increases from the left to the right. The outliers increase from top to bottom.

III.3.2 Properties

Similarly to the classical quasi-interpolant schemes [4], the wQISA method satisfies a number of desirable regularity properties, here briefly recalled. For more details and for a probabilistic interpretation of the method, we refer the reader to [38].

Global bounds Let $\mathcal{P} \subset \mathbb{R}^3$ be a point cloud and $z_{\min}, z_{\max} \in \mathbb{R}$ that satisfy

$$z_{\min} \leq z \leq z_{\max}, \qquad \text{for all } (x, y, z) \in \mathcal{P}.$$

Then the weighted quasi interpolant spline approximation to $\mathcal{P}$ from some spline space $\mathbb{S}_{p,[\mathbf{x},\mathbf{y}]}$ and some weight function w has the same (global) bounds

$$z_{min} \leq f_w(x, y) \leq z_{max}, \qquad \text{for all } (x, y) \in \mathbb{R}^2. \tag{III.8}$$

Local bounds Let $x \in [x_\mu, x_{\mu+1})$ for some μ in the range $p_x + 1 \leq \mu \leq n_x$ and $y \in [y_\nu, y_{\nu+1})$ for some ν in the range $p_y + 1 \leq \nu \leq n_y$. The global bounds of Equation III.8 can be refined, by using the B-splines' property of local representation, to

$$\min_{\substack{i=\mu-p_x,\ldots,\mu \\ j=\nu-p_y,\ldots,\nu}} \hat{z}_w(x_i^*, y_j^*) \leq f_w(x, y) \leq \max_{\substack{i=\mu-p_x,\ldots,\mu \\ j=\nu-p_y,\ldots,\nu}} \hat{z}_w(x_i^*, y_j^*).$$

We can simplify these bounds to:

$$\min_{(x,y,z)\in\mathcal{P}_{\mu,\nu}} z \leq f_w(x) \leq \max_{(x,y,z)\in\mathcal{P}_{\mu,\nu}} z, \tag{III.9}$$

where

$$\mathcal{P}_{\mu,\nu} := \bigcup_{\substack{i=\mu-p_x,\ldots,\mu \\ j=\nu-p_y,\ldots,\nu}} \left\{ \operatorname{supp}\left(w(\cdot,\cdot,x_i^*,y_j^*) \right) \right\} \cap \mathcal{P}.$$

Note that the wQISA bounds depends on the functions w. By choosing weight functions having compact support, one can control these bounds while reducing the number of points required for each control point estimation.

Special configurations, shape preservation and rate of convergence From basic spline theory, it is straightforward that if $x_{i+1} = \ldots = x_{i+p_x} < x_{i+p_x+1}$ and $y_{j+1} = \ldots = y_{j+p_y} < y_{j+p_y+1}$ then wQISA interpolates the control point estimate $\hat{z}(x_i^*, y_j^*)$. By assuming (1) any knot in $\mathbf{x}$ and $\mathbf{y}$ to have maximum multiplicity, (2) w to be a 1-NN weight function and (3) the point cloud to consist of samplings of a continuous function $f : [a_1, b_1] \times [a_2, b_2] \to \mathbb{R}$ at the knot averages, we have that f_w corresponds to the *Variation Diminishing Spline Approximation* (VDSA) of f of bi-degree $\mathbf{p}$ to the point cloud $\mathcal{P}$ over the knot vectors $\mathbf{x}$ and $\mathbf{y}$ (see for example [32]). In this perspective, wQISA can be seen as a generalization of VDSA to perturbed data through a wider family of weight functions.

There is also another point of view: wQISA corresponds to applying VDSA to $\hat{z}_w : \mathbb{R}^2 \to \mathbb{R}$. As pointed out in [32], VDSA preserves certain shape properties - such as monotonicity and convexity - of the function being approximated. In our case, wQISA will preserve the monotonicity and convexity of $\hat{z}_w$, i.e., of the average trend of the point cloud, with respect to the chosen weight function. In case of points clouds with defects, the average trend is indeed more important than the position of a point with respect to the others. By considering this analogy between VDSA and wQISA, we conclude that the latter is characterized by a linear convergence (see for example [32] for the rate of convergence of VDSA).

III.3.3 Data-driven implementation

When dealing with approximation, the quality of a method relies on its prediction capability over independent samples, i.e., on data that has not been used to “train" the model. Whilst for function approximation the accuracy of a method can be quantified by directly comparing the approximation with the original function, in case of point clouds approximation the estimation of the approximation error on new samples is not trivial (for instance, it is often impossible to re-sample the data). Moreover, only a few benchmarks are available. One way to overcome this problem comes from statistical learning, and is here adopted in the form of a learning-based implementation of wQISA. Our implementation consists of four main steps:

Step 1 *Data pre-processing.* The input point cloud is used to generate a *training*, a *validation* and a *test* set, each of which has a specific task in the approximation of the input data. Different strategies are proposed, depending on the size of the input (see Section III.3.3.1).

Step 2 *wQISA formulation.* Given a tensor mesh and a weight function, the training set is used to define the control point estimator $\hat{z}_w$. Notice that $\hat{z}_w$ may depend on free parameters to be tuned (see Section III.3.3.2).

Step 3 *Parameter settings.* A prediction error is defined on the validation set, in order to fix the possible free parameters of the weight functions III.3.3.3).

Step 4 *Model assessment.* Having obtained an approximation model on the basis of the previous steps, the test set is used for estimating the generalization error on new data (see Section III.3.3.5).

While the first and the last steps are done only once, the second and the third ones are run in a while-loop, by refining the mesh at each iteration (see Section III.3.3.4).

Remark III.3. The use of different point clouds for Steps 1-3 in our data-driven implementation is necessary to avoid data overfitting. This is particularly relevant when looking for a model with a good prediction capability, rather than just aiming at minimizing the error on a finite set of points (see, for instance, [26, 39]).

We provide a simplified flowchart of the algorithm in Figure III.2 and a pseudo-algorithm in Algorithm III.1, while referring to Sections III.3.3.1-III.3.3.5 for a detailed description of the implemented procedures.

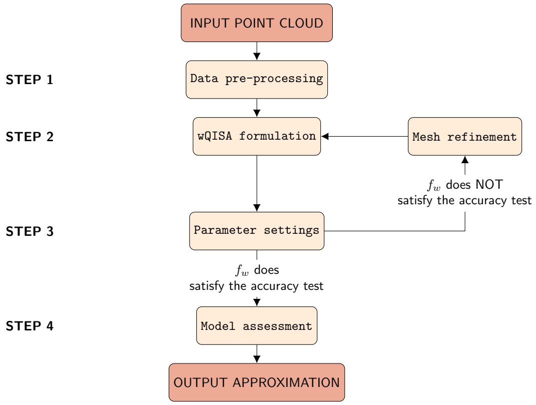


Figure III.2: Flowchart of the data-driven wQISA algorithm.

III.3.3.1 Data pre-processing

The input point cloud $\mathcal{P}$ is used to generate three new sets: a training set $\mathcal{T}$, a validation set $\mathcal{V}$ and a test set $\mathcal{U}$. There is no general rule on how to choose the number of observations in each of the three sets. As suggested in [26], a typical split for large point clouds might be 50% for training set, and 25% each for validation and testing sets. The three sets must have similar data distribution

```
Input: $\mathcal{P}$, $\mathbf{x}$, $\mathbf{y}$, $w$, $\varepsilon$;
Output: $f_w(\mathbf{x}, \mathbf{y})$ (spline approximation), MSE (generalization error)
function DataPreProcess($\mathcal{T}$, $\mathcal{V}$, $\mathcal{U}$)
    return $\mathcal{T}$, $\mathcal{V}$, $\mathcal{U}$;
end function
function wQISA-Definition($\mathcal{T}$, $\mathbf{x}$, $\mathbf{y}$, $w$)
    Apply Equation III.3 and get $f_w(\mathbf{x}, \mathbf{y}, k)$;
    return $f_w(\mathbf{x}, \mathbf{y}, k)$;
end function
function wParameters($\mathcal{V}$, $f_w(x, y, k)$)
    Find the parameter $k$ that minimizes GMSE;
    Get $f_w(x, y)$ by fixing the optimal parameter $k$;
    return [$f_w(x, y)$,GMSE];
end function
function RefineMesh($\mathcal{V}$,$\mathbf{x}$, $\mathbf{y}$,$f_w(\mathbf{x}, \mathbf{y})$,$\varepsilon$)
    Compute LMSE from $\mathcal{V}$ and $f_w(\mathbf{x}, \mathbf{y})$;
    Check elements where LMSE is above $\varepsilon$;
    Split those elements by knot refining both $\mathbf{x}$ and $\mathbf{y}$;
    return $\mathbf{x}$ and $\mathbf{y}$;
end function
function GeneralizationError($\mathcal{U}$,$f_w(\mathbf{x}, \mathbf{y})$)
    return MSE of $f_w(\mathbf{x}, \mathbf{y})$ over $\mathcal{U}$;
end function
function main(($\mathcal{P}$, $\mathbf{x}$, $\mathbf{y}$, $w$, $\varepsilon$))
    /*Split the data*/
[$\mathcal{T}$, $\mathcal{V}$, $\mathcal{U}$]=DataPreprocess($\mathcal{P}$);
/*Define the family of approximations*/
$f_w(x, y, k)$=wQISA-Definition($\mathcal{T}$, $\mathbf{x}$, $\mathbf{y}$, $w$);
/*Fix the parameters (here, $k$)*/
[$f_w(x, y)$,GMSE]=wParameters($\mathcal{V}$, $f_w(x, y, k)$);
/* While stopping criteria not satisfied*/
[$\mathbf{x}, \mathbf{y}$] = RefineMesh($\mathcal{V}$,$\mathbf{x}$, $\mathbf{y}$,$f_w(\mathbf{x}, \mathbf{y})$,$\varepsilon$);
$f_w(x, y, k)$=wQISA-Definition($\mathcal{T}$, $\mathbf{x}$, $\mathbf{y}$, $w$);
[$f_w(x, y)$,GMSE]=wParameters($\mathcal{V}$, $f_w(x, y, k)$);
/*end*/
MSE=GeneralizationError($\mathcal{U}$,$f_w(\mathbf{x}, \mathbf{y})$);
return [$f_w(\mathbf{x}, \mathbf{y}), MSE$];
end function
```

Algorithm III.1: wQISA pseudocode for k-NN weight

properties, i.e., the noise distribution and the point sampling of the validation and test sets must be comparable to the one of the training set. For instance, the training set $\mathcal{T}$ can be easily obtained in Matlab from $\mathcal{P}$ by applying the function `pcdownsample` with 50% of sampling rate and then, $\mathcal{V}$ and $\mathcal{U}$ can be obtained with the sample procedure applied to the complement of $\mathcal{T}$. Figure III.3 visually shows a possible data partitioning. In order to achieve even more robust results, one can consider several splits of the data into training, validation, and test sets.

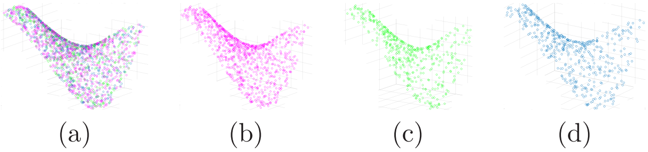


Figure III.3: Point cloud partitioning. (a) The original point cloud $\mathcal{P}$; (b) the training set $\mathcal{T}$; (c) the validation set $\mathcal{V}$; (d) the test set $\mathcal{U}$.

In many real-world applications there might be insufficient data to split the set of measurements into three parts. In these cases, we use, the K-fold cross-validation: the input point cloud is split into K roughly equal-sized parts and $K-1$ parts are used to train the model, while the remaining one is used to test it. This process is repeated by changing the part to be used for accuracy evaluation, and then combining the K estimates of the prediction error. In this case, the validation and test sets are the same.

III.3.3.2 wQISA formulation

The control point estimator is defined by Equation (III.3). As pointed out in Remark III.1, the wQISA input consists of:

1. *A point cloud.* We use the training point cloud computed in Section III.3.3.1.

2. *A tensor product spline space defined by regular knot vectors.* At the first iteration, one can consider a tensor mesh consisting of a single element with maximum knot multiplicities, with this element being the bounding box of the input data. The mesh is then (globally) refined, iteration-by-iteration, where the approximation shows a lower precision (see Section III.3.3.4). Although the spline bi-degree can be considered as an additional parameter, we here choose to focus on C^1 bi-quadratic spline surfaces as they are smooth enough to represent data in a good way [44].

3. *A weight function.* We here decide the weight function to be used on a case-by-case basis. One could nevertheless consider dictionaries of weight functions and then choose the more suitable via an opportune validation error.

Having these inputs fixed, the approximation defined in Equation (III.2) may still depend on additional parameters to be tuned (see Section III.3.3.3).

III.3.3.3 Parameter settings

The control point estimator depends on the parameters defining a weight function (e.g., k in a k-NN weight), if any. Statistical learning fixes the free parameters by minimizing some prediction error on an independent data set, here represented by the validation set. The reason for not using the training set for parameter settings is the need to avoid overfitting, i.e., a model that has good performances on the input set but has a low prediction capability on new data. We here fix the parameters by minimizing the *Global Mean Squared Error* (GMSE), i.e., the Mean Squared Error (MSE) over the whole validation point cloud. The minimization of the chosen loss function can be undertaken via iterative methods (e.g., stochastic optimization), or by directly evaluating the loss function at a finite number of parameters (e.g., in case of k-NN weight).

III.3.3.4 Termination criterion

The steps described in the Sections III.3.3.2 and III.3.3.3 are run in a while loop. The refinement level of a mesh is fixed by considering the last iteration before the GMSE starts increasing (for instance, this occurs when the model starts overfitting the data), with a maximum number of iterations (here we set it to 15). At each iteration of the while loop, we compute a local validation error and use it, together with a user-defined threshold, to decide which elements should be split by knot insertion. We here insert the knot at the mid-value, in each of the two coordinate directions.

We here consider the *Local Mean Squared Error* (LMSE). Given an element of the mesh, the LMSE on that element is the Mean Squared Error over any validation point whose projection onto the plane (x, y) falls into that element. If no projection lies inside that element, the LMSE for that element is set to zero.

III.3.3.5 Model assessment

Once the model has been selected, the test set is used to estimate the generalization error. In a data-rich situation, validation and test sets are distinct, and so are in general the validation and test errors. In case the data set was not large enough to be split into three parts, these validation and test sets are equal. This leads the validation and test errors to be the same, and results, in general, in an underestimation of the generalization error (see once more [26]).

III.3.4 Computational complexity

The computational complexity of a single wQISA iteration depends on the chosen weight function and tensor product spline space. Given the linear space $\mathbb{S}_{\mathbf{p},[\mathbf{x},\mathbf{y}]}$ of tensor product B-splines of bi-degree $\mathbf{p}$ over the regular knot vectors $\mathbf{x} \in \mathbb{R}^{n_x+p_x+1}$ and $\mathbf{y} \in \mathbb{R}^{n_y+p_y+1}$, there are exactly $\dim(\mathbb{S}_{\mathbf{p},[\mathbf{x},\mathbf{y}]}) = n_x \cdot n_y$ control points to be estimated.

The Gaussian weight is global and thus computes, for a single control point estimation, the linear combination of N addends. Since the weight of any point is at most as expensive as the exponential of an Euclidean norm, the computational complexity of a control point estimate is $O(N)$. To considerably reduce the computational complexity, one can recover local support by composing global weight functions with a k-NN tree (see Equation III.6) or with a weight functions with local support, such as the indicator weight function (see Equation III.4). Indeed, in case of a k-NN weight function the time needed to compute all the coefficients through k-d trees is proportional to $O(N \log(N))$, where N is the number of points of the cloud [19]. For its efficiency, k-d trees are already adopted for noise point clouds reconstruction, see for instance [23].

The number of iterations to reach the local minimum of the loss function depends on the input point cloud, the chosen approximation method, and the loss function itself (here: the GMSE over the validation point cloud). In these settings, the use of a method with a high convergence rate can cause an overshooting of the minimum. Experimental results on the number of iterations of our method are reported in Section III.5 and show that wQISA converges in a reasonably low number of iterations.

n the following, we analyze the CPU times to approximate a point cloud with increasing cardinality, randomly sampled from the mathematical function

$$z = \sqrt{64 - 81\big((x-0.5)^2 + (y-0.5)^2\big)}/(9-0.5)$$

over a mesh with an increasing number of knots. We here approximate an IDW weight function by considering, for each knot average, only the K closest points (K is set to 500). We have used, for this test, a prototype implementation developed in Python. These tests run on a 2019 MacBook pro with 2.4 GHz 8-cores Intel®Core$^{\mathrm{TM}}$ i9-processor, resulting in the CPU times shown in the log-log plot of Figure III.4.

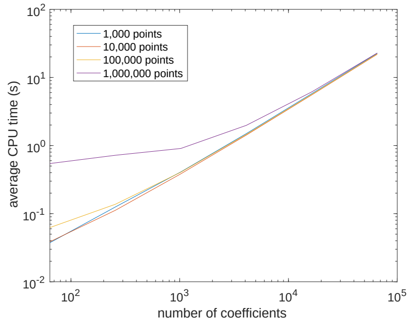


Figure III.4: CPU times. Log-log plot of the CPU times when increasing the point cloud cardinality and the number of coefficients.

III.4 Experimental settings

Despite the popularity of data approximation, we could not find a benchmark able to address all the use cases we are targeting (surface reconstruction, terrain modelling and spatial data measurement approximation). For this reason we propose a new comparative analysis. In this Section we detail the approximation methods and the performance metrics we adopt in our work.

III.4.1 Approximation methods

We here overview the three popular approximation schemes used as wQISA counterparts throughout this paper: namely, the implicit approximation with Radial Basis Functions (RBF), Kriging and the Multilevel B-spline Approximation (MBA).

III.4.1.1 Implicit approximation with radial basis functions

This implicit approximation has the form

$$f(x, y) := \sum_{i=1}^{N} w_i \phi(||(x - x_i, y - y_i)||_2), \tag{III.10}$$

where the approximating function is represented as a linear combination of *radial basis functions*, each associated with a different center (x_i, y_i) and weighted by an unknown coefficient w_i. The weights w_i can be computed by imposing interpolatory constraints of the data, where the trivial null solution is avoided by adding normal constraints. Depending on the properties of ϕ, RBFs can be locally- or globally-supported. The computational complexity of global and local approximations is respectively $O(N^3)$ and $O(N \log N)$. We here consider the following kernels:

$$\phi(r) := e^{(r/\varepsilon)^2}, \qquad \text{(Gaussian)} \tag{III.11a}$$

$$\phi(r) := \sqrt{1 + (r/\varepsilon)^2}, \qquad \text{(Multiquadric)} \tag{III.11b}$$

$$\phi(r) := \frac{1}{\sqrt{1 + (r/\varepsilon)^2}} \qquad \text{(Inverse multiquadric)} \tag{III.11c}$$

$$\phi(r) := e^{-\sqrt{r}}, \qquad \text{(Modified Gaussian)} \tag{III.11d}$$

where ε is a shape parameter that here approximates the average distance between nodes. An additional parameter $\alpha \geq 0$ is introduced by performing an L^2-regularization on the interpolation matrix, in order to increase the smoothness of the approximation ($\alpha = 0$ is for interpolation). We consider the implementation from the Python class `interpolate.Rbf`, contained in the `Scipy` library.

III.4.1.2 Kriging

Kriging owes its importance to its capability to take into account the correlation among input data, which may strongly affect the approximation, e.g., when unevenly distributed. Kriging is defined by the weighted average

$$f(x, y) := \mathbf{w}^t \mathbf{z} = \sum_{i=1}^{N} w_i z_i, \tag{III.12}$$

where the weights $\mathbf{w} \in \mathbb{R}^N$ are the solution to the linear system $\mathbf{Cw} = \mathbf{c}$, where $\mathbf{C}$ is the covariance matrix of the input point cloud and $\mathbf{c}$ is the vector of covariances between the input points and the prediction point $(x, y) \in \mathbb{R}^2$. Despite its popularity, Kriging suffers several issues when applied to large data sets. For each sample, the ordinary kriging estimator needs to solve a linear system and thus the computational complexity scales quadratically with the number of input points. We use here the Python package `PyKrige`.

III.4.1.3 Multilevel B-spline approximation

Multilevel B-spline Approximation (MBA) was originally introduced in [30] for approximating scattered data by tensor product B-spline surfaces. A peculiarity of MBA, that makes it analogue to wQISA, is its explicit formulation, which allows it to compute an approximation without solving any equation system. In this respect, MBA can be considered as an example of quasi-interpolation method. At the first step, the tensor product mesh consists of only one element. At each successive iteration, each element is halved in the two coordinate directions. The coefficient c_k of a B-spline B_k is given by

$$c_k = \frac{\sum_i B_k(x_{i,k}, y_{i,k})^2 \phi_i}{\sum_i B_k(x_{i,k}, y_{i,k})^2},$$

where $(x_{i,k}, y_{i,k})_i$ is the set of points within the support of the B-spline B_k and

$$\phi_i := \frac{B_k(x_{i,k}, y_{i,k}) z_i}{\sum_l B_l(x_{i,k}, y_{i,k})^2},$$

where the sum in the denominator is taken over all B-splines which contains $(x_{i,k}, y_{i,k})$ in their support. The computational complexity scales linearly with the number of input points. We here use the implementation of the Geometry Group at SINTEF ICT, available at https://github.com/orochi663/MBA.

III.4.2 Evaluation measures

To evaluate the quality of an approximation against the input point cloud, we need to define some error measures. We here present a comparison with a number of measures, each one able to highlight different approximation aspects.

III.4.2.1 Punctual error and its statistics

Given an approximation defined on the training point cloud, the *absolute punctual error* is computed on the validation point cloud. The absolute punctual error is studied via measures of central tendency (mean) and of statistical dispersion (standard deviation and mean square error).

III.4.2.2 Hausdorff distance and L^∞ norm

The *Hausdorff distance* (or *two-sided Hausdorff distance*) d_{Haus} between two non-empty subsets A and B in $(\mathbb{R}^d, ||\cdot||_2)$ is given by

$$d_{\text{Haus}}(A,B) := \max\left\{\sup_{a\in A}\inf_{b\in B}||a-b||_2, \sup_{b\in B}\inf_{a\in A}||a-b||_2\right\},$$

where sup represents the supremum and inf the infimum. In our case, A is chosen to be the validation point cloud, while B is the image of an approximation f over its domain. In our settings, the sup and inf can be replaced by max and min respectively. We will use this distance for all sets of points embedded in an Euclidean space. For data measurements that associate a scalar value to a point of a grid we will use instead the L^∞ norm.

III.5 Experimental simulations

In our experiments we consider real data coming from different use cases. We classify the data as affected by different levels of noise: low (e.g., 3D point clouds from high-quality laser scans), average (e.g., terrains from Lidar or sonar acquisition) and high (e.g., air pollution and rainfall measurements from sensors and radars). The data size varies from few dozens to hundreds of thousands of points. All data used in our experiments are provided as additional material. To provide a fair comparative analysis - and avoid any method to overfit - a data-driven implementation is used for all the considered techniques. Hence, the approximation parameters and the model validation are kept distinct and performed on the same training and validation datasets for all methods. In addition, the optimal shaping parameter α for the RBF approximations is found through a constrained trust-region method, while for Kriging we always use the default package settings (linear variogram model).

In all Tables, we use the following naming convention: Gaussian kernel (RBF^1); multiquadric kernel (RBF^2); inverse multiquadric kernel (RBF^3); modified Gaussian kernel (RBF^4).

III.5.1 Surface reconstruction

As examples of scattered 3D point clouds, we consider three models from the Science and Technology in Archaeology Research Center (STARC, [47]) of The Cyprus Institute, which consist of a digitisation of archaeological fragments. The complete models are shown in Figure III.5 (left column): the model labelled as A.1 is part of a vessel; models A.2 and A.3 are parts of votive statues. The models are acquired via high-quality laser scans. For our experiments, we select a region of interest in every model (highlighted in red), which is then up-sampled via a uniform Montecarlo approach; namely, A.1 is sampled with $30,000$ points, A.2 with $47,642$ points and A.3 with $29,843$ points. We here test wQISA with the Gaussian weight functions of Equation III.5; the standard deviation is $\sigma = 10^{-4}$ for the model A.1 and $\sigma = 10^{-3}$ for A.2 and A.3.

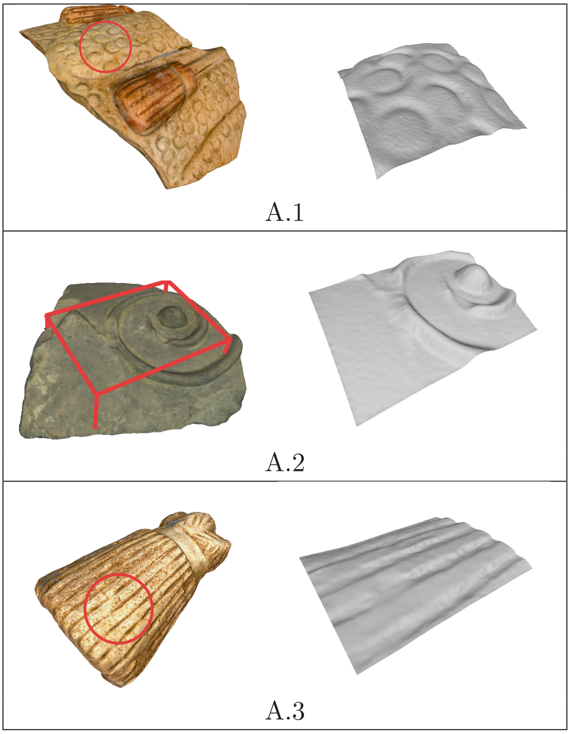
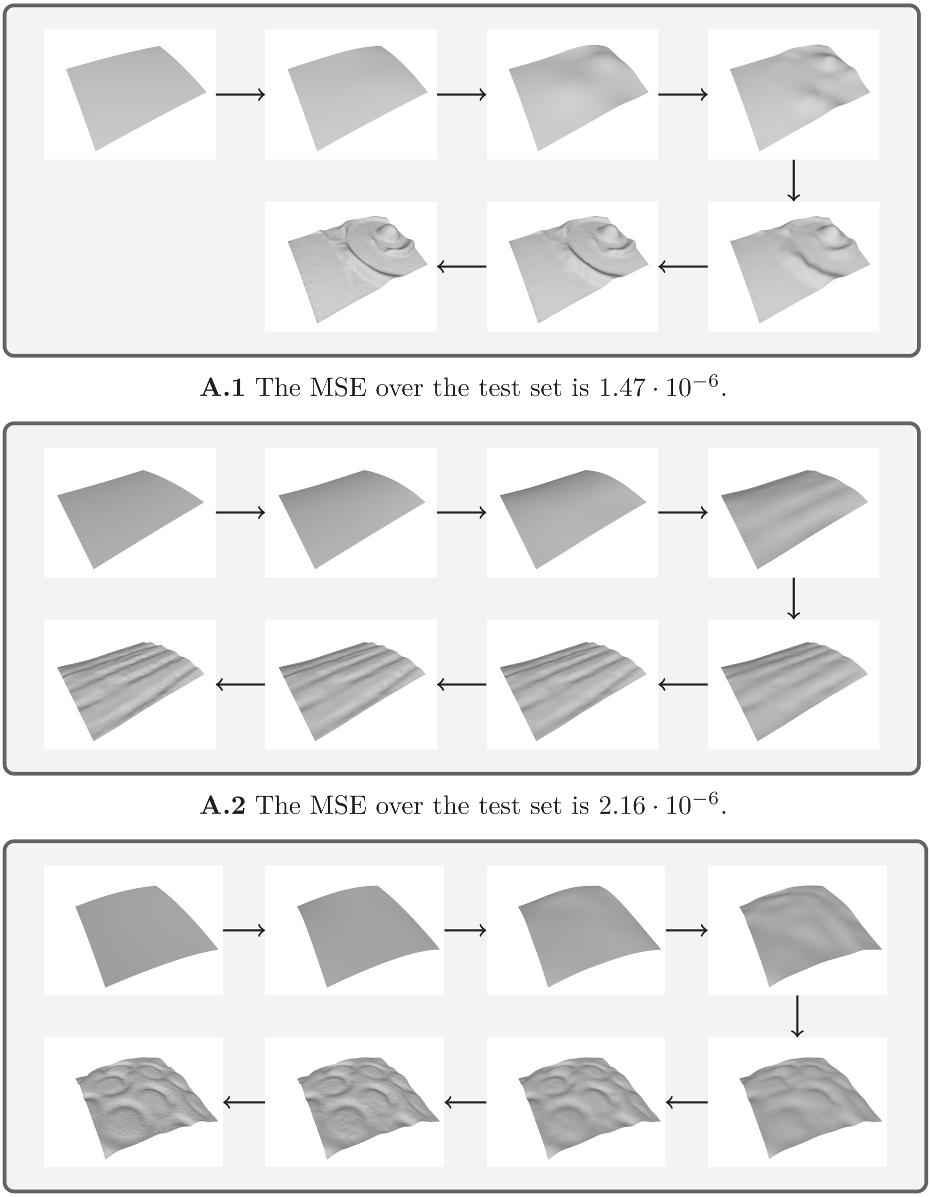


Figure III.5: Left sides: the original models (from STARC repository [47]). Right sides: outcomes of the wQISA method via Gaussian weights for the areas highlighted in red.

Figure III.5 (right column) depicts the local reconstructions obtained via wQISA. The approximations show a correct recovery of the main details of the artefacts, with an accurate shape preservation: in A.1, the symmetry of the bulge at the center of the spiral is maintained; in A.2, we can notice a faithful reconstruction even in presence of small geometric variations, such as chiselings or local reliefs; finally, A.3 shows a reconstruction characterized by circular patterns. Figure III.6 depicts the iterative process that leads to the final wQISA approximations and reports the MSE values computed in Step 4 of Algorithm III.1.

We refer to Table III.1 for a detailed quantitative comparison of the approximation performances of wQISA, RBFs, Kriging and MBA of the models in Figure III.5. We highlight in bold the values with the lowest order of magnitude, which correspond to the best performances. This Table also lists the required number of iterations. From these experiments we can conclude that wQISA performs comparatively well with respect to the other methods when the standard deviation and the Hausdorff distance are considered. This is not surprising because the idea behind wQISA is to fit the overall data trend rather that



A.1 The MSE over the test set is $1.47 \cdot 10^{-6}$.

A.2 The MSE over the test set is $2.16 \cdot 10^{-6}$.

A.3 The MSE over the test set is $6.92 \cdot 10^{-9}$.

Figure III.6: Representation step by step of the approximations of the artifacts in Fig. III.5 via wQISA with Gaussian weight function.

converging to single values. Tables III.2 reports the size of the optimal grid for wQISAs and MBAs. The optimal smoothing parameters for the RBF approximations can be found in Table III.3.

Table III.1: Local approximation of 3D models. The labels A.1, A.2 and A.3 refer to the models in Figure III.5. In bold, we highlight the errors with the lowest order of magnitude. The single asterisk highlights those cases where the maximum number of iterations, here set to 15, is reached. Kriging runs in a single step and no number is displayed.

		Punctual error			Hausdorff	Number of Iterations
		mean	st.d.	MSE		
A.1	wQISA	$4.445\cdot10^{-5}$	$\mathbf{6.881\cdot10^{-5}}$	$\mathbf{6.723\cdot10^{-9}}$	$\mathbf{9.229\cdot10^{-4}}$	7
	RBF^1	$1.194\cdot10^{-5}$	$\mathbf{3.399\cdot10^{-5}}$	$\mathbf{1.305\cdot10^{-9}}$	$\mathbf{9.303\cdot10^{-4}}$	15*
	RBF^2	$3.94\cdot10^{-3}$	$3.052\cdot10^{-3}$	$2.480\cdot10^{-5}$	$1.968\cdot10^{-2}$	13
	RBF^3	$4.496\cdot10^{-3}$	$3.635\cdot10^{-3}$	$3.335\cdot10^{-5}$	$2.436\cdot10^{-2}$	15*
	RBF^4	$2.760\cdot10^{-3}$	$2.241\cdot10^{-3}$	$1.261\cdot10^{-5}$	$1.425\cdot10^{-2}$	15*
	Kriging	$2.801\cdot10^{-5}$	$\mathbf{5.352\cdot10^{-5}}$	$\mathbf{3.650\cdot10^{-9}}$	$\mathbf{9.307\cdot10^{-4}}$	-
	MBA	$\mathbf{9.451\cdot10^{-6}}$	$\mathbf{3.137\cdot10^{-5}}$	$\mathbf{1.071\cdot10^{-9}}$	$\mathbf{9.301\cdot10^{-4}}$	7
A.2	wQISA	$7.046\cdot10^{-4}$	$\mathbf{9.590\cdot10^{-4}}$	$1.420\cdot10^{-6}$	$\mathbf{6.670\cdot10^{-2}}$	8
	RBF^1	9.112	22.504	589.495	$1.351\cdot10^{3}$	15*
	RBF^2	$\mathbf{8.043\cdot10^{-5}}$	$\mathbf{2.044\cdot10^{-4}}$	$\mathbf{4.826\cdot10^{-8}}$	$\mathbf{6.412\cdot10^{-2}}$	12
	RBF^3	$1.097\cdot10^{-2}$	$4.990\cdot10^{-2}$	$2.61\cdot10^{-3}$	2.264	15*
	RBF^4	$2.199\cdot10^{-2}$	$2.973\cdot10^{-2}$	$1.361\cdot10^{-3}$	1.154	15*
	Kriging	$1.153\cdot10^{-3}$	$1.301\cdot10^{-3}$	$3.027\cdot10^{-6}$	$\mathbf{6.695\cdot10^{-2}}$	-
	MBA	$1.214\cdot10^{-4}$	$\mathbf{2.503\cdot10^{-4}}$	$\mathbf{7.738\cdot10^{-8}}$	$\mathbf{6.695\cdot10^{-2}}$	13
A.3	wQISA	$1.054\cdot10^{-3}$	$\mathbf{9.926\cdot10^{-4}}$	$2.081\cdot10^{-6}$	$\mathbf{5.640\cdot10^{-2}}$	8
	RBF^1	23.964	98.608	$1.030\cdot10^{4}$	$4.342\cdot10^{3}$	15*
	RBF^2	$\mathbf{1.440\cdot10^{-4}}$	$\mathbf{2.212\cdot10^{-4}}$	$\mathbf{6.958\cdot10^{-8}}$	$\mathbf{5.552\cdot10^{-2}}$	12
	RBF^3	$3.008\cdot10^{-2}$	$4.139\cdot10^{-2}$	$2.617\cdot10^{-2}$	7.891	15*
	RBF^4	$8.760\cdot10^{-2}$	$1.189\cdot10^{-1}$	$2.179\cdot10^{-2}$	7.066	15*
	Kriging	$\mathbf{1.510\cdot10^{-4}}$	$\mathbf{1.601\cdot10^{-4}}$	$\mathbf{4.845\cdot10^{-8}}$	$\mathbf{5.527\cdot10^{-2}}$	-
	MBA	$\mathbf{1.635\cdot10^{-4}}$	$\mathbf{2.185\cdot10^{-4}}$	$\mathbf{7.446\cdot10^{-8}}$	$\mathbf{5.506\cdot10^{-2}}$	10

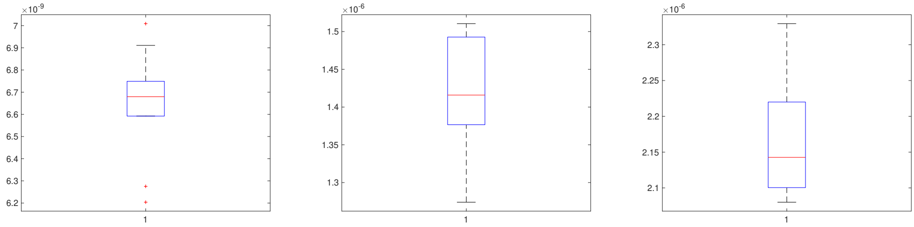
Table III.2: Size of the tensor mesh for wQISA and MBA.

Method	**A.1**	**A.2**	**A.3**
wQISA	50 × 64	128 × 128	128 × 128
MBA	128 × 128	4096 × 4096	128 × 128

Table III.3: Optimal smoothing parameter α for the RBFs implementations.

Method	**A.1**	**A.2**	**A.3**
RBF^1	$1.27\cdot10^{-3}$	$1.49\cdot10^{-1}$	$1.49\cdot10^{-1}$
RBF^2	$2.37\cdot10^{-1}$	$3.49\cdot10^{-2}$	$3.43\cdot10^{-2}$
RBF^3	$2.83\cdot10^{-1}$	$4.54\cdot10^{-2}$	$2.89\cdot10^{-1}$
RBF^4	$2.86\cdot10^{-1}$	$1.42\cdot10^{-1}$	$1.42\cdot10^{-1}$

The robustness of the data-driven implementation to different choices of the training, validation and test sets (see Section III.3.3.1) is here assessed by considering ten different data splits for each 3D model. The resulting MSEs over the respective validation point clouds are graphically represented in the form of boxplots (see Figure III.7).



(A.1) (A.2) (A.3)

Figure III.7: Robustness to different training, validation, and test sets for the models in Figure III.5. We show with box-plots the MSE over 10 validation sets.

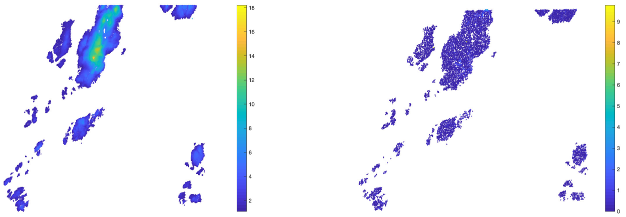
III.5.2 Terrain data modelling

Acquisition methods such as Lidar, photogrammetry and sonar produce huge amounts of data that may be perturbed. Methods to efficiently and effectively handle these point clouds are thus required. Since the Kriging and RBFs implementations we are considering are not tailored for big data and present some scalability limitations, here, we limit our attention to the two local methods: wQISA and MBA.

The first data set comes from the island municipality of Værøy and consists of many islands. The data are provided by the Norwegian Mapping Authority Kartverket, and are freely downloadable at https://hoydedata.no/LaserInnsyn/. The original point cloud is pre-processed to remove the data outside the mainland, and reduced to 44,529 points. The peculiarity of this terrain is that alternate regions with high variability and almost flat regions.

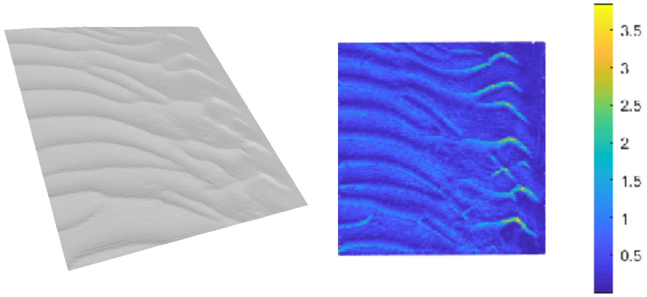
The second data set corresponds to submarine sand dunes off the coast of France. The data were obtained using bathymetric surveys and are detailed in [18]. The selected part of the data set contains 759,952 points. The difference in sea depth is about 33.2 m. This terrain was already adopted in [44], it is relatively large and presents slight ripples on a wide wavefront.

For both models, we consider a wQISA representation with IDW weight (see Equation III.7). Figures III.8 shows the wQISA representation of the first terrain, together with the absolute punctual error. The optimal mesh is reached in 11 iteration (1024 × 1024 elements) for wQISA and in 10 iterations (512 × 512 elements) for MBA. Similarly, the approximation and the absolute punctual error of the second terrain are shown in Figure III.9. For this model, the optimal mesh for wQISA contains 6400 × 6400 elements while for MBA it contains 1024 × 1024 elements.



(a) (b)

Figure III.8: Værøy. Figure (a) shows a wQISA of the islands. Figure (b) shows the absolute punctual error.



(a) (b)

Figure III.9: Sand dunes. In (a): a wQISA of the sand dunes (north is pointing upwards to the left). In (b): the punctual error.

Table III.4: Terrain data: Værøy (B.1) and sand dunes (B.2).

		Punctual error			
		mean	st.d.	MSE	Hausdorff
B.1	wQISA	0.2158	0.2582	0.1132	**9.6266**
	MBA	**0.1513**	**0.2257**	**0.0738**	13.1604
B.2	wQISA	0.0492	0.0641	0.0065	**8.2569**
	MBA	**0.0330**	**0.0543**	**0.0040**	8.2579

A quantitative comparison of the wQISA and the MBA performances can be found in Table III.4. MBA provides a better punctual approximation (highlighted by the MSE measure), but it is again less accurate than wQISA for the overall fit (as highlighted by the Hausdorff distance).

III.5.3 Data measurements

We analyse the approximation of strongly perturbed data sets. We consider two precipitation events, with the aim of deepening the comparisons previously presented in [35]. Although focusing here on a specific environmental application, we highlight that these approximation methods do not limit to rainfall field simulations or to a specific sample size. We propose in the additional material the study of a rather small data set (just 11 samples) measuring air pollution. In this case the B-spline approximations rather reduces to linear polynomial approximations, and thus we can consider it as a limiting case.

The first event occurred on September 29, 2013, and was characterized by light rain over Liguria with two different thunderstorms that caused local flooding and landslides. This data set counts only 143 measurements, all acquired by the rain gauge network maintained by Regione Liguria, which is spread over the whole region. A quantitative comparison is provided in Table III.5 and is computed by performing a LOO cross-validation on each method. The optimal parameters are here listed:

- *wQISA with k-NN weight.* The optimal number of neighbors k is $k = 2$, where we have checked for any $k = 1, \dots, 10$. The optimal mesh has 8×8 elements.
- *RBF.* The optimal smoothing parameter α for the RBF approximations is computed by minimizing the mean squared error: $\alpha = 1.00$ for the Gaussian kernel (RBF^1); $\alpha = 1.22$ for the RBF with multiquadric kernel (RBF^2); $\alpha = 1.00$ for the RBF with inverse kernel (RBF^3); $\alpha = 34.31$ for the RBF with modified Gaussian kernel (RBF^4).
- *MBA.* The mean squared error start increasing after the at iteration number 6, with an optimal tensor mesh containing 16×16 elements.

wQISA has the best generalization performances over the data set with respect to standard deviation, mean squared error and L^∞ norm. This means that wQISA offers more accurate predictions on independent data.

The second event occurred on January, 2014. The data set contains $19,187$ measurements, gathered from different devices: rain gauges and weather radar. Besides the 143 rain gauges by Regione Liguria, another 25 measure stations by Genoa municipality are considered. The remaining points come from raw radar acquisitions, at first as reflectivity measurements with a range of 400 km. The frequency of mountains over the whole Ligurian territory affects the quality of radar acquisitions and a pre-processing step is needed to remove ground clutter effects; processed data are then combined with observations gathered from rain gauges, which are more reliable measurements but do not cover the whole region. The integration of radar data in the interpolation of the precipitation field makes it possible to extend rainfall fields also to areas surrounding Liguria, and therefore to have a clearer picture about the temporal evolution of precipitation events. Since the temporal interval is different for each acquisition device, rainfall measurements have been cumulated. In this study, a 30 minutes cumulative

Table III.5: Rainfall fields, statistics for the error distribution for the LOO cross-validation.

Method	Mean [mm]	Std [mm]	MSE [mm^2]	L^∞ [mm^2]	Numb. of iterations
wQISA	0.405	**0.565**	**0.483**	**4.952**	4
RBF^1	4.058	4.720	38.747	39.957	114
RBF^2	**0.390**	0.587	0.497	5.321	20
RBF^3	5.472	4.326	48.659	42.285	132
RBF^4	0.790	0.936	1.500	6.400	26
Kriging	0.474	0.613	0.600	5.668	-
MBA	0.412	0.629	0.562	4.983	5

step has been used (240 time samples). This procedure of integrating gauge and radar data was made to alleviate the well-known error and uncertainty that characterize radar estimates. Spurious signals may be caused, for example, by radar failure or by shielding of the radar beam by mountain ranges [25]. However, various outliers still perturbs the data. The simulation over five time intervals is given in Table III.6.

Again, wQISA provides the best generalization performances with respect to standard deviations and mean squared error. The difficulty of handling these datasets is reflected by the variability of the L^∞ norm; because of the presence of outliers, the maximum of the absolute differences is largely unstable and, for each dataset, there is a different winner, even if the wQISA is generally one of the best performing methods also in with respect to this measure. The optimal parameters for the RBF approximations are provided in Table III.7. The optimal parameters k for the wQISA with k-NN weight function are: $k = 1$ (00:30), $k = 2$ (01:00), $k = 5$ (01:30), $k = 1$ (00:30), $k = 5$ (02:00) and $k = 7$ (02:30).

III.6 Concluding remarks

The weighted quasi interpolant spline approximation (wQISA) is a simple and robust procedure to obtain a spline approximation of point clouds. This paper presents a data-driven implementation of wQISA [38], inspired to the supervised learning paradigm. The method has been tested on several real-world data from different application domains and different levels of noise. Experiments have shown that wQISA is able to accurately generalize to data different from those used to define it, i.e. it can effectively describe geometric shapes and measurements not only at those points where it was defined. In other words, wQISA can be successfully used also in prediction tasks, as a linear regression model in supervised learning.

When dealing with data affected by low level of noise, the wQISA method offers approximations that are comparable with other well-known methods. However, the experiments show a better adherence of the wQISA surfaces to point

Table III.6: Rainfall fields, statistics for the error distribution.

		Mean	Std	MSE	L^{∞}	Numb. of
	Method	[mm]	[mm]	[mm^2]	[mm]	iterations
00:30	wQISA	**0.321**	**0.597**	**0.460**	14.578	7
	RBF^1	6.193	89.241	$8 \cdot 10^3$	$4.16 \cdot 10^3$	121
	RBF^2	0.625	0.924	1.244	14.356	80
	RBF^3	1.549	30.776	949.591	$1.55 \cdot 10^3$	6
	RBF^4	1.135	1.570	3.753	15.600	23
	Kriging	0.958	1.248	2.476	14.473	-
	MBA	0.675	1.053	1.563	**14.238**	5
01:00	wQISA	**0.270**	**0.589**	**0.419**	11.125	10
	RBF^1	130.886	225.675	$6.80 \cdot 10^4$	$6.94 \cdot 10^3$	122
	RBF^2	0.362	2.03	4.985	91.068	122
	RBF^3	144.857	124.710	$3.65 \cdot 10^4$	903.952	122
	RBF^4	0.790	0.936	1.500	**6.400**	728
	Kriging	0.964	1.233	2.245	12.449	-
	MBA	0.595	1.009	1.373	12.815	5
01:30	wQISA	0.305	0.647	0.512	10.792	8
	RBF^1	1.297	3.062	11.055	64.532	6
	RBF^2	0.302	0.645	**0.509**	10.822	35
	RBF^3	0.503	1.254	1.826	20.700	7
	RBF^4	1.109	1.641	3.363	13.334	24
	Kriging	0.474	**0.613**	0.600	**5.668**	-
	MBA	**0.299**	0.653	0.516	10.810	8
02:00	wQISA	**0.457**	**1.078**	**1.370**	28.484	7
	RBF^1	266.258	246.256	$1.32 \cdot 10^5$	$2.69 \cdot 10^3$	122
	RBF^2	0.749	4.389	19.822	101.409	122
	RBF^3	369.316	326.457	$2.43 \cdot 10^5$	**18.775**	137
	RBF^4	1.283	2.222	6.582	34.145	12
	Kriging	1.197	1.892	5.014	33.140	-
	MBA	0.916	1.560	3.271	31.4356	5
02:30	wQISA	0.423	**0.974**	**1.127**	**16.069**	9
	RBF^1	36.954	42.373	$3.16 \cdot 10^3$	470.072	124
	RBF^2	0.432	0.991	1.169	$3.22 \cdot 10^3$	90
	RBF^3	0.850	3.112	10.408	128.381	11
	RBF^4	1.544	2.933	10.984	23.973	12
	Kriging	1.776	2.334	8.600	22.421	-
	MBA	**0.391**	1.058	1.271	31.9741	8

clouds, as reflected by the smaller Hausdorff distance. On these relatively error-free models, we have experimentally noticed a convergence with less iterations than the other methods considered. On the one hand, the wQISA method has linear convergence. On the other hand, this becomes an advantage when dealing with strongly perturbed data, as it allows to adapt better to the underlying

Table III.7: Optimal smoothing parameter α for the RBFs implementations.

Method	00:30	01:00	01:30	02:00	02:30
RBF^1	0.00	1.00	0.68	1.00	0.00
RBF^2	439.64	1.00	0.19	0.99	0.99
RBF^3	0.00	1.00	0.00	0.59	0.00
RBF^4	104.84	1.00	0.98	0.24	0.23

distribution. Indeed, on very noisy data wQISA shows a very good stability with respect to standard deviation and mean squared error. In the simulation of precipitation events, wQISA outperforms meshless methods (RBF and Kriging), which in [35] showed the best approximation capability. Experiments confirm that wQISA is able to successfully handle hundreds of thousands of points.

To sum up, two conclusions can be drawn from the comparative analysis. First: it is difficult and possibly ill-advised to search for an *absolute* approximation method, because different approximation techniques can provide better results in different contexts and using different evaluation metrics. Second: for the studied point clouds, the experimental results show that data-driven wQISA exhibits a good prediction capability when it comes to strongly perturbed data.

As a further development of the method, we think it is possible to consider adaptive refinement schemes, such as in the case of LR B-splines or THB-splines [6, 11]. This is particularly relevant because these locally refining schemes naturally deal with isogeometric computations and simulation and offers the valuable perspective to practically adopt this work for Computer Aided Design and Manufacturing (CAD/CAM), Finite Element Analysis and IsoGeometric Analysis [22, 27]. Finally, we intend to extend the comparative analysis to include other real-world applications, methods, and high dimensional data.

Acknowledgements. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675789. The work has been partially developed in the CNR-IMATI activities DIT.AD021.080.001 and DIT.AD009.091.001.

The authors also thanks: Dr. Bianca Falcidieno and Dr. Michela Spagnuolo for the fruitful discussions; Dr. Oliver J. D. Barrowclough, Dr. Tor Dokken and Dr. Georg Muntingh for their concern as supervisors; Dr. Vibeke Skytt for her help in collecting the terrain data sets; Laboratoire Domaines Océaniques - CNRS UMR 6538 for the submarine sand dunes data set; Dr. Simone Pittaluga for the rainfall field data sets; Eng. Simone Cammarasana for his help in generating pictures with Delaunay triangulations.

References

[1] Ackermann, J. and Goesele, M. "A Survey of Photometric Stereo Techniques." In: *Foundations and Trends in Computer Graphics and Vision* vol. 9, no. 3–4 (2015), pp. 149–254.

[2] Alexa, M. et al. "Point set surfaces". In: *Proceedings of the conference on visualization 01, VIS 01, IEEE Computer Society.* 2001, pp. 21–28.

[3] Berger, M. et al. "A Survey of Surface Reconstruction from Point Clouds". In: *Computer Graphics Forum* vol. 36, no. 1 (2017), pp. 301–329.

[4] Boor, C. de and Fix, G. J. "Spline approximation by quasi-interpolants". In: *Journal of Approximation Theory* vol. 8, no. 1 (1973), pp. 19–45.

[5] Bracco, C., Giannelli, C., and Sestini, A. "Adaptive scattered data fitting by extension of local approximations to hierarchical splines". In: *Computer Aided Geometric Design* vol. 52-53 (2017), pp. 90–105.

[6] Bracco, C. et al. "Adaptive fitting with THB-splines: Error analysis and industrial applications". In: *Comput. Aided Geom. Des.* vol. 62 (2018), pp. 239–252.

[7] Carr, J. C. et al. "Reconstruction and Representation of 3D Objects with Radial Basis Functions". In: *proceedings of SIGGRAPH '01.* New York, NY, USA: ACM, 2001, pp. 67–76.

[8] Cheney, E. W. "Approximation Theory, Wavelets and Applications". In: ed. by Singh, S. P. Vol. 454. Springer. NATO Science Series (Series C: Mathematical and Physical Sciences), 1995. Chap. Quasi-interpolation, pp. 37–45.

[9] Chui, C. K. *An introduction to wavelets.* Elsevier, 2016.

[10] Davydov, O., Prasiswa, J., and Reif, U. "Two-stage approximation methods with extended B-splines". In: *Mathematics of Computation* vol. 83, no. 286 (2014), pp. 809–833.

[11] Dokken, T., Lyche, T., and Pettersen, K. F. "Polynomial splines over locally refined box-partitions". In: *Computer Aided Geometric Design* vol. 30, no. 3 (2013), pp. 331–356.

[12] DÆhlen, M. and Floater, M. "Iterative polynomial interpolation and data compression". In: *Numerical Algorithms* vol. 5, no. 3 (Mar. 1993), pp. 165–177.

[13] Feng, W., Yang, Z., and Deng, J. "Moving multiple curves/surfaces approximation of mixed point clouds". In: *Commun Math Stat* vol. 2, no. 1 (2014), pp. 107–124.

[14] Fleishman, S., Cohen-Or, D., and Silva, C. "Robust moving least-squares fitting with sharp features". In: *ACM SIGGRAPH 2005 papers, ACM.* 2005, pp. 544–552.

[15] Floater, M. S. and Iske, A. "Multistep scattered data interpolation using compactly supported radial basis functions". In: *Journal of Computational and Applied Mathematics* vol. 73, no. 1 (1996), pp. 65–78.

[16] Forsey, D. R. and Bartels, R. H. "Surface Fitting with Hierarchical Splines". In: *ACM Trans. Graph.* vol. 14, no. 2 (Apr. 1995), pp. 134–161.

[17] Franke, R. “Scattered Data Interpolation: Tests of Some Method”. In: *Mathematics of Computation* vol. 38, no. 157 (1982), pp. 181–200.

[18] Franzetti, M. et al. “Giant dune morphologies and dynamics in a deep continental shelf environment: Example of the banc du four (Western Brittany, France)”. In: *Marine Geology* vol. 346 (2013), pp. 17–30.

[19] Friedman, J. H., Bentley, J. L., and Finkel, R. A. “An Algorithm for Finding Best Matches in Logarithmic Expected Time”. In: *ACM Trans. Math. Softw.* vol. 3, no. 3 (Sept. 1977), pp. 209–226.

[20] Garnero, G. “Comparisons between different interpolation techniques”. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* vol. 5 (2013), W3.

[21] Georgopoulos, A. *Heritage and Archaeology in the DigitalAge. Acquisition, Curation, and Dissemination of Spacial Cultural Heritage Data*. Ed. by Vincent, M. L. et al. Springer, 2017.

[22] Giannelli, C. et al. “THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 299 (2016), pp. 337–365.

[23] Giraudot, S., Cohen-Steiner, D., and Alliez, P. “Noise-Adaptive Shape Reconstruction from Raw Point Sets”. In: *Computer Graphics Forum* vol. 32, no. 5 (2013), pp. 229–238.

[24] Greiner, G. and Hormann, K. *Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines, surface fitting and multiresolution methods.* 1997.

[25] Harrison, D. L., Driscoll, S. J., and Kitchen, M. “Improving precipitation estimates from weather radar using quality control and correction techniques”. In: *Meteorological Applications* vol. 7, no. 2 (2000), pp. 135–144.

[26] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* second. Springer, 2009.

[27] Johannessen, K. A., Kvamsdal, T., and Dokken, T. “Isogeometric analysis using LR B-splines”. In: *Computer Methods in Applied Mechanics and Engineering* vol. 269 (2014), pp. 471–514.

[28] Kazhdan, M., Bolitho, M., and Hoppe, H. “Poisson surface reconstruction”. In: *4^{th} EG Symp. on Geometry Processing.* 2006, pp. 61–70.

[29] Kiss, G. et al. “Adaptive CAD model (re-)construction with THB-splines”. In: *Graphical Models* vol. 76, no. 5 (2014), pp. 273–288.

[30] Lee, S., Wolberg, G., and Shin, S. Y. “Scattered data interpolation with multilevel Bsplines”. In: *IEEE Transactions on Visualization and Computer Graphics* vol. 3, no. 3 (July 1997), pp. 228–244.

[31] Levin, D. “Geometric modeling for scientific visualization”. In: ed. by Brunnett, G. et al. Springer-Verlag, 2003. Chap. Mesh-independent surface interpolation, pp. 37–49.

[32] Lyche, T. and Mørken, K. *Spline Methods Draft.* 2011.

[33] Ohtake, Y. et al. “Multi-level Partition of Unity Implicits”. In: *ACM Transactions on Graphics* vol. 22, no. 3 (July 2003), pp. 463–470.

[34] Oliver, M. A. and Webster, R. “Kriging: a method of interpolation for geographical information systems”. In: *International Journal of Geographical Information Systems* vol. 4, no. 3 (1990), pp. 313–332.

[35] Patané, G. et al. “Comparing methods for the approximation of rainfall fields in environmental applications”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* vol. 127 (2017), pp. 57–72.

[36] Patanè, G. and Spagnuolo, M. “Heterogeneous Spatial Data: Fusion, Modeling, and Analysis for GIS Applications”. In: *Synthesis Lectures on Visual Computing* vol. 8, no. 2 (2016), pp. 1–155.

[37] Patanè, G. and Spagnuolo, M. “Local approximation of scalar functions on 3D shapes and volumetric data”. In: *Computers & Graphics* vol. 36, no. 5 (2012), pp. 387–397.

[38] Raffo, A. and Biasotti, S. “Weighted quasi-interpolant spline approximations: Properties and applications”. In: *Numerical Algorithms* vol. 87, no. 2 (2021), pp. 819–847.

[39] Reitermanova, Z. “Data splitting”. In: *WDS.* Vol. 10. 2010, pp. 31–36.

[40] Sablonnière, P. “Univariate spline quasi-interpolants and applications to numerical analysis”. In: *Rendiconti del Seminario Matematico.* Vol. 63. 2. Università degli studi di Torino / Politecnico di Torino, 2005, pp. 211–222.

[41] Savchenko, V. V. et al. “Function representation of solids reconstructed from scattered surface points and contours”. In: *Computer Graphics Forum* vol. 14, no. 4 (1995), pp. 181–188.

[42] Sederberg, T. W. et al. “T-Splines and T-NURCCS”. In: *ACM Transactions on Graphics* vol. 22, no. 3 (2003), pp. 477–484.

[43] Shen, C., O’Brien, J., and Shewchuk, J. “Interpolating and approximating implicit surfaces from polygon soup”. In: *ACM SIGGRAPH 2004 papers, ACM.* 2004, pp. 896–904.

[44] Skytt, V., Barrowclough, O., and Dokken, T. “Locally refined spline surfaces for representation of terrain data”. In: *Computer & Graphics* vol. 49 (2015), pp. 58–68.

[45] Skytt, V. et al. “Deconfliction and Surface Generation from Bathymetry Data Using LR B-splines”. In: *Mathematical Methods for Curves and Surfaces.* Ed. by Floater, M. et al. Springer, 2017, pp. 270–295.

[46] Sorgente, T. et al. “Topology-driven shape chartification”. In: *Computer Aided Geometric Design* vol. 65 (2018), pp. 13–28.

[47] STARC. *STARC repository.* URL: http://public.cyi.ac.cy/starcRepo/.

Appendix III.A Additional material

When dealing with particularly small data sets, wQISA can still be used for single polynomial approximation. As an example, we consider the 2017 Particulate Matter (PM10) report for the city of Oslo (Norway), which is freely available at the municipal data-bank[1]. Particulates, also known as suspended particulate matter, are microscopic particles of solid or liquid matter suspended in the air. They are classified by size, such as for the particles smaller than 10 μm, called PM10, and the particles less than $2.5\mu m$, called PM2.5. PM10 contains particles that originate from combustion (e.g., heating) and road dust (e.g., car tyres and brakes). As particulates are among the most harmful form of air pollution, their approximation is of great interest.

The data set contains only 11 samples; see in Figure III.10 an Oslo map with their position. The evaluation results give insight into the approximation performance in real condition of sparsity and provide an example of a situation where there is insufficient data to split into three parts.

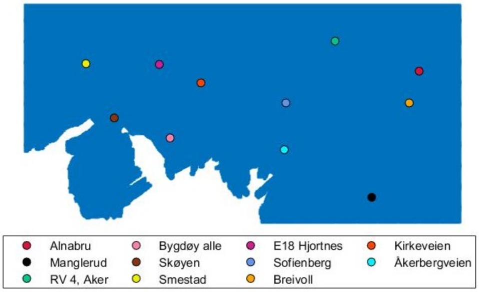


Figure III.10: The 11 stations monitoring the PM10 concentration, owned by *Oslo kommune* and *Statens vegvesen*.

Given the small size of the data set, a Leave-One-Out cross-validation was performed to fix the free parameters and compute the statistics for the punctual error. The optimal parameters were computed by minimizing the mean squared error:

- *wQISA with k-NN weight.* Given the low number of points, we fix the restrict our attention to a degenerate tensor mesh containing just one element. The optimal number of neighbors k is $k = 5$, where we have checked for any $k = 1, \dots, 10$.

- *RBF.* The optimal smoothing parameter α for RBF approximations is computed by minimizing the mean squared error: $\alpha = 679.69 \cdot 10^{-3}$ for

[1] http://statistikkbanken.oslo.kommune.no/webview/

Gaussian kernel (RBF[1]) in 53 iterations; $\alpha = 190.06{\cdot}10^{-3}$ for RBF with multiquadric kernel (RBF[2]) in 51 iterations; $\alpha = 25.00{\cdot}10^{-6}$ for RBF with inverse kernel (RBF[3]) in 7 iterations; $\alpha = 983.44{\cdot}10^{-3}$ for RBF with modified Gaussian kernel (RBF[4]) in 51 iterations.

- *MBA*. The mean squared error start increasing after the first iteration, thus we adopt a tensor mesh containing just one element as for wQISA.

The results are presented in Table III.8. The three best performances with respect to the MSE are given by MBA, wQISA and RBF with Gaussian kernel.

Table III.8: Air pollution, statistics for the error distribution for the LOO cross-validation.

	Mean	Std	MSE	L^∞
Method	[mm]	[mm]	[mm^2]	[mm]
wQISA	3.005	**1.984**	12.967	7.245
Kriging	3.365	2.020	15.401	7.068
RBF^1	7.626	3.346	77.042	16.525
RBF^2	3.320	2.260	16.130	7.921
RBF^3	3.501	2.531	18.660	7.704
RBF^4	14.731	3.714	230.801	21.193
MBA	**2.576**	2.378	**11.775**	**6.784**

Authors' addresses

Andrea Raffo Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, andrea.raffo@ge.imati.cnr.it

Silvia M. Biasotti Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, silvia.biasotti@ge.imati.cnr.it

Paper IV

Extraction of geometric primitives from 3D point clouds for CAD applications

Chiara Romanengo, Andrea Raffo, Silvia Biasotti, Bianca Falcidieno

Manuscript.

Abstract

The automatic creation of geometric models from point clouds has numerous applications in CAD (reverse engineering, manufacturing, assembling) and, in general, in shape modelling and processing. In this work, we propose the use of the Hough transform (HT) framework to fit a dense point cloud with geometric primitives. With respect to the standard Hough approach and the traditional simple geometric primitives used in CAD, we are able to include also algebraic surfaces and more complex analytical surfaces. To the best of our knowledge, this is the first time that this technique is applied to CAD models, thanks to algebraic geometry concepts. The use of both the HT and a clustering strategy also permits a further analysis and interpretation of the geometric primitives found and their aggregation as parts of a unique, more complete primitive or into characteristics patterns. Experiments performed on a variety of CAD models reveal the robustness of the primitive fitting method and the automatic detection of compound primitives.

Keywords: geometric primitives, point clouds, reverse engineering, Hough transform.

Contents

IV.1 Introduction

3D Computer Aided Design (CAD) models are among the most common medium to convey dimensional and geometrical information on designed objects or components. In several situations, unfortunately, the CAD model of an object is not available, it does not even exist, or no longer corresponds to the real geometry of the manufactured object itself.

A strategy to retrieve an object digital model, when this is not available, is to acquire 3D data directly on the object and to use the obtained information to build a digital representation. The reconstruction of digital models from measured data has been a long-term goal of engineering and computer science in general; this process, usually called Reverse Engineering (RE), aims at generating 3D mathematical surfaces and geometric features representing the geometry of real parts.

There are many methods that address this problem and we refer to these surveys that group a large part of the approaches presented so far: [8, 18, 34]. The general RE framework can essentially be decomposed into three general steps: data capture and preprocessing, segmentation and surface fitting, CAD model creation. These phases are generally common to the vast majority of techniques available in the literature.

The phase "segmentation and surface fitting" logically divides the original point set into subsets containing just those points sampled from a particular natural surface, it decides to what type of surface each subset of points belongs (e.g., planar, cylindrical) and it finds which surface of the given type is the best fit to those points in the given subset. No doubt that this step is the most important of the whole RE framework, as the results obtained may significantly differ depending on the strategy adopted to perform this task.

In our context, various specific and effective methods have been used, they exploit the a-priori knowledge of a shape to decompose it into regions approximated by primitives belonging to a given set. According to [18], these approaches can be grouped into four families: stochastic, parameter space, clustering and learning techniques. The first group includes the RANSAC method [31] and several further optimizations, e.g., [23]. The second family includes Hough-like voting methods and parameter space clustering, e.g., [24]. The third includes all the other clustering techniques, from primitive-driven region growing to automatic clustering, to primitive-oblivious segmentation, e.g., [1, 20, 39]. Finally, with the growing popularity of deep learning techniques, supervised fitting methods have been proposed even for multi-class primitives [22]. We refer to [18] for a detailed taxonomy and overview of methods for simple primitive fitting.

Many of these approaches have the problem of preprocessing the acquired data to eliminate noise and outliers and do not deal with point clouds. In addition, they can extract only simple geometric primitives (planes, cylinders, cones, spheres, tori or quadrics), without considering more complex basic shapes, such as helices or generalized cylinders. Most methods for fitting primitives do not aggregate the primitives they recognise in a single one if they are not contiguous. For example, if a pipe is interrupted by another part as in the block model in Figure IV.5(a), traditional methods recognise the two parts of the pipe as two distinct primitives.

Our goal is to aggregate the primitives found on the basis of their size and position, whenever possible. This is particularly important, for instance, when decomposing patterns that correspond to scans of assembly CAD models, because it concurs to the recognition of compound primitives and patterns [25]. To do this, we need a surface primitive recognition method that is particularly robust to noise and outliers and able to recognise multiple instances of the same primitive. The Hough transform meets these needs and, in the extended version proposed in [4], also provides a wide range of primitives to draw on.

Since its introduction, the Hough transform (HT) has gained a large popularity for its capability of recognising multiple instances of a primitive function in a set of points. Originally limited to straight lines in an image [17], it has been generalized and extended in multiple directions, expanding the families of curves that can be treated within the HT framework, including algebraic functions [4], surfaces [36] and, in its most generic formulation, hypersurfaces [3, 33]. In practice, a main limitation to the use of the standard HT approach for large point clouds is the increase of the parameter space size and the computational cost when the number of parameters augments. Indeed, this fact limited the use of the standard Hough transform to planes [7, 24], combinations of linear subspaces [14], spheres [9] and circular cylinders [28]. In the case of cylinders, there was the need of limiting the five parameters of the cylinder representation to three: therefore an a-priori estimation of the cylinder axis was performed. The Generalized Hough transform (GHT) [2] extended the HT to more complex shapes; in 3D a similar extension has been proposed in [38], using points clouds from CAD objects as templates. Unfortunately, while permitting the use of quite complex models, the template shapes used in the GHT are not parametric nor possess an analytic representation, thus limiting the use of this extension to detecting primitives for building a geometric model. The recent advances in the use of the algebraic functions and the potential adaptive sampling of the parameter space, are paving the road to a larger use of the HT for recognising more complex families of geometric primitives. Indeed, we generalize to surface primitives represented either in parametric or implicit form the idea originally proposed in [4] for plane curves. As far as we know, the unique approach of identification of a primitive, based on this idea, has been proposed in [3], but only for the recognition of an ellipsoid in a free-form model.

In this paper, we use the Hough transform systematically for the recognition of simple and complex surface primitives. To the best of our knowledge, this is the first time that this technique, based on the theory defined in [4], is applied to CAD models. In addition, we use a clustering technique to aggregate primitives that are not aligned (e.g., two misaligned cylinders) or slightly differ (for this we also provide a measure of difference within the elements of a pattern). In this regard, we extend the method for aggregating patches defined in [29] to the aggregation of the surface primitives found by using the HT technique. The outcome of our method is a set of surface primitives, gathered because they are compound and/or belong to the same pattern.

The rest of the paper is organised as follows. Section IV.2 briefly overviews the basic concepts of the HT, it provides a list of geometric primitives that can be identified by our method and describes the HT-based recognition algorithm. Section IV.3 describes the clustering method adopted to group the geometric primitives recognized by the HT-based algorithm. Section IV.4 provides experimental results of HT-based recognition and clustering of geometric primitives from point clouds extracted from CAD objects. Discussions and concluding remarks end the paper.

IV.2 Recognising surface primitives using Hough transforms

In this section, we briefly summarise some basic concepts on the Hough transform, provide a list of relevant primitives we can recognise, and introduce the main steps of the HT-based recognition algorithm.

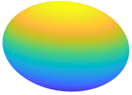
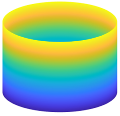
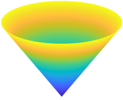
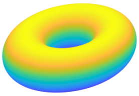
IV.2.1 Background

We denote by $\mathbb{A}^3_{\mathbf{x}}(\mathbb{R})$ and $\mathbb{A}^n_{\mathbf{A}}(\mathbb{R})$, respectively, the 3-dimensional and the n-dimensional affine spaces over $\mathbb{R}$, where $\mathbf{x} := (x, y, z)$ and $\mathbf{A} := (A_1, \dots, A_n)$ vectors of indeterminates. Given a surface $\mathcal{S}$ defined as the zero locus of a function f, a parameter-dependent family of surfaces can be described by functions $f_{\mathbf{a}}$ as $\mathcal{F} = \{\mathcal{S}_{\mathbf{a}} : f_{\mathbf{a}}(x) = 0 \mid \mathbf{a} \in \mathcal{U}\}$, where $\mathcal{U}$ is an open set of the parameter space $\mathbb{A}^n_{\mathbf{A}}(\mathbb{R})$ and $\mathbf{a} := (a_1, ..., a_n)$ is the parameter vector. Then, given a point $P \in \mathbb{A}^3_{\mathbf{x}}(\mathbb{R})$, the HT of the point P with respect to the family $\mathcal{F}$ is $\Gamma_P = \{f_{\mathbf{a}}(P) = 0\}$ in $\mathbb{A}^n_{\mathbf{A}}(\mathbb{R})$. For sufficiently general points $P \in \mathbb{A}^3_{\mathbf{x}}(\mathbb{R})$, it turns out that Γ_P are hypersurfaces in $\mathbb{A}^n_{\mathbf{A}}(\mathbb{R})$. If the set of hypersurfaces Γ_P generated by varying P on a given surface meets in one and only one point $\bar{\mathbf{a}} \in \mathcal{U}$, the family of surfaces $\mathcal{F}$ is called *Hough-regular*. The intersection point $\bar{\mathbf{a}}$ of the hypersurfaces Γ_P defines the parameters of the best fitting surface $\mathcal{S}_{\bar{\mathbf{a}}}$. If the HT regularity is not guaranteed, the point might be non unique and a solution must be selected among more potential parameters solutions.

The general HT-framework deals with the problem of finding a surface – within a family $\mathcal{F}$ of surfaces – that best approximates a particular shape, given in the form of a data set of N points $\mathcal{D}$, where $N >> n$. The HT-based method we propose in this paper can be directly applied to point clouds containing multiple shapes, and it is able to deal with both implicit and parametric representations of surfaces. We consider some basic assumptions for each representation: in case of implicitly-defined surfaces, we assume that the functions $f_{\mathbf{a}}$ are analytical with respect to the Cartesian coordinates x, y, z and the parameter vector $\mathbf{a} = (a_1, \dots, a_n)$, $n \geq 2$, as in [33]; in case of parametric surfaces, we assume that the system defining $\mathcal{S}_{\mathbf{a}}$ with respect to the Cartesian coordinates x, y, and z can be analytically solved with respect to the parameter vector $\mathbf{a} = (a_1, \dots, a_n)$.

The common strategy to identify the solution (or a solution), introduced in [4], is based on the so-called *accumulator function*; it consists in a voting system

Table IV.1: Families of simple geometric primitives expressed in both implicit and parametric form.

(a) ellipsoid	(b) cylinder	(c) cone	(d) torus
 $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}-1=0$ $\begin{cases} x = a\sin v\cos u \\ y = b\sin v\sin u \\ z = c\cos v \end{cases}$	 $\frac{x^2}{a^2}+\frac{y^2}{b^2}-1=0$ $\begin{cases} x = a\cos u \\ y = b\sin u \\ z = v \end{cases}$	 $\frac{x^2}{a^2}+\frac{y^2}{a^2}-\frac{z^2}{b^2}=0$ $\begin{cases} x = av\cos u \\ y = av\sin u \\ z = bv \end{cases}$	 $(a-\sqrt{x^2+y^2})^2+z^2-b^2=0$ $\begin{cases} x = (a+b\cos u)\cos v \\ y = (a+b\cos u)\sin v \\ z = b\sin u \end{cases}$

whereby each point in $\mathcal{D}$ votes a n-uple $\mathbf{a}=(a_1,\ldots,a_n)$; the most voted n-uple corresponds to the most representative surface for the profile. A more detailed description of how this works in our case is given in Section IV.2.3.

IV.2.2 Families of geometric primitives

In addition to the simple geometric primitives (planes, spheres, ellipsoids, cylinders, cones and tori, depicted in table IV.1), we are able to extract some more complex primitives. For the sake of simplicity, some of the surfaces are here presented in their canonical form or with respect to some specific axes; nevertheless, one can easily generalise these equations by applying a generic transformation of the special orthogonal group SO(3).

- *General cylinders.* A cylinder is a surface traced by a straight line of fixed direction, the *generatrix*, while moving along a curve, the *directrix*. Given a curve $(x(u),y(u),z(u)) := (f_1(\mathbf{a},u), f_2(\mathbf{a},u), f_3(\mathbf{a},u))$ and a direction (l,m,n), the parametric representation of the corresponding cylinder is given by:
$$\begin{cases} x = f_1(\mathbf{a},u)+lv \\ y = f_2(\mathbf{a},u)+mv \\ z = f_3(\mathbf{a},u)+nv \end{cases} .$$
Note that a general cylinder depends on the parameters defining generatrix and directrix. The dictionary of curves to be considered as diretrix is extremely rich, see [32]. Table IV.1(b) provides the equations of an elliptic cylinder, while Table IV.2(a) considers a 5−convexity curve as directrix, whose equation can be found in [32].

- *General cones.* A cone is a surface traced by a straight line, the *generatrix*, while gliding along a curve, the *directrix*, and passing through a fixed point, the *vertex*. Given a parametrized curve $(x(u),y(u),z(u)) :=$

$(f_1(\mathbf{a},u), f_2(\mathbf{a},u), f_3(\mathbf{a},u))$ and a point $V = (x_V, y_V, z_V)$, the parametric representation of the corresponding cone is given by:

$$\begin{cases} x = x_V + (f_1(\mathbf{a},u) - x_V)v \\ y = y_V + (f_2(\mathbf{a},u) - y_V)v \\ z = z_V + (f_3(\mathbf{a},u) - z_V)v \end{cases} .$$

As in the case of cylinders, we can exploit the dictionary of plane curves to create families of cones. Then, a general cone depends on the parameters that define the curve and the components of the vertex. The implicit and parametric equations of the family of circular cones are presented in Table IV.1(c). Table IV.2(b) shows a cone generated by a $5-$convexity curve.

- *Surfaces of revolution.* A family of surfaces of revolution can be created by rotating a family of curves around an axis of rotation. For example, given the family of plane curves $(x(u), y(u), z(u)) := (f_1(\mathbf{a},u), 0, f_2(\mathbf{a},u))$ and the $z-$axis, we obtain the parametric equations

$$\begin{cases} x = f_1(\mathbf{a},u)\cos v \\ y = f_1(\mathbf{a},u)\sin v \\ z = f_2(\mathbf{a},u) \end{cases} .$$

One of the most known examples is the *torus*; Table IV.1(d) provides the parametric and implicit equations of a family of tori obtained by rotating a circle on the plane $y = 0$ and centred in $(a, 0, 0)$ around the $z-$axis. Table IV.2(c) shows an example obtained by rotating the curve $(x(u), y(u), z(u)) := (au, 0, b/u^5)$ around the $z-$axis.

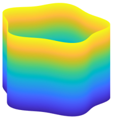
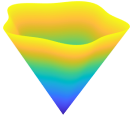
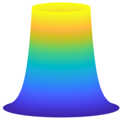
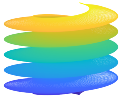
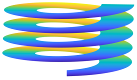
- *Helical surfaces.* Table IV.2(d) presents a family of equations obtained by modifying the parametrization of a circular cylinder. Precisely, the radius R here varies between $[R_1, R_2]$, where $R_1 > 0$, by a cosine function; when the radii are fixed, the slope of the output surface is controlled by the parameters in $z(u, v)$. Note that the radius variation can be adapted to other shapes (e.g., the triangle wave function).

- *Convex combination of curves.* It is possible to define surface primitives by considering the convex combination of a pair of parametrized curves $(f_1(\mathbf{a},u), f_2(\mathbf{a},u), f_3(\mathbf{a},u))$ and $(g_1(\mathbf{b},u), g_2(\mathbf{b},u), g_3(\mathbf{b},u))$. This family has the following parametric equations:

$$\begin{cases} x = vf_1(\mathbf{a},u) + (1-v)g_1(\mathbf{b},u) \\ y = vf_2(\mathbf{a},u) + (1-v)g_2(\mathbf{b},u) \\ z = vf_3(\mathbf{a},u) + (1-v)g_3(\mathbf{b},u) \end{cases} ,$$

where $v \in [0, 1]$. Note that the primitive parametrization depends on the same parameters which define the pair of curves, i.e., $\mathbf{a}$ and $\mathbf{b}$. A planar example is given by the *annulus*, i.e., the region bounded by two concentric circles. A helical strip can be obtained by cutting and bending an annular strip; this corresponds to consider a convex combination of two helices of equal slope but different radii. An example of helical strip is provided in Table IV.2(e).

Table IV.2: Parametrizations of some complex geometric primitives.

(a) cylinder	(b) cone	(c) surface of revolution	(d) helical surface	(e) helical strip
				
$\begin{cases} x = \frac{a\cos u}{1+b\cos(5u)} \\ y = \frac{a\sin u}{1+b\cos(5u)} \\ z = v \end{cases}$	$\begin{cases} x = \frac{av\cos u}{1+b\cos(5u)} \\ y = \frac{av\sin u}{1+b\cos(5u)} \\ z = Av + B \end{cases}$	$\begin{cases} x = au\cos v \\ y = au\sin v \\ z = \frac{b}{u^5} \end{cases}$	$\begin{cases} x = R(u)\cos v \\ y = R(u)\sin v \\ z = k_1(u+nv)+k_2 \end{cases},$ where $R(u) := R_1 + \frac{R_2 - R_1}{2}(\cos u + 1),$ $n \in \mathbb{Z}$	$\begin{cases} x = R(u)\cos v \\ y = R(u)\sin v \\ z = v \end{cases},$ where R(u):=au+(1-u)b

IV.2.3 Algorithm

In this section, we describe a method to fit a dense point cloud with surface primitives via the HT technique. Our approach takes in input a set of points PC and a given threshold ε, to assess which points are significant for a specific primitive, see Algorithm IV.1. This value typically ranges from 1% to 3% of the diagonal of the minimum bounding box of the PC.

For each type of geometric primitive (e.g., plane, circular cylinder, elliptical cone) the parameters of the surfaces that best fit some parts of the point cloud are found through the following procedure:

- *Pre-processing.* The input point cloud PC is rotated so as to align the sides of the minimum bounding box with the main axes. Then, it is translated so as to place the barycenter of the PC in the origin. At this point, the user can select the families of primitives to be used for recognition, otherwise the algorithm automatically selects one at a time all the families of simple primitives.

- *Initialization of the HT parameter space and of the accumulator function.* A region T of the parameter space is fixed and is discretized into cells, which are uniquely identified by the coordinates of their centre. When searching for primitives in their canonical form, the dimension of the parameter spaces under study is up to 3, and initial guesses on where to look for can be obtained by studying the family's geometric characteristics of the family $\mathcal{F}$ (e.g., bounding box, radius, diagonal length). For primitives in their generic position, we assume that one of the following hyphotheses hold true:

- After detecting primitives in the canonical forms, some of the remaining segments contains at most one primitive type.
- A segment that contains more than a primitive type is oversegmented by using a state-of-the-art segmentation method, i.e., a RANSAC algorithm, so as to fall into the previous case.

For a generic segment, initial estimates can be computed by testing a set of primitive types as outlined in Appendix IV.A; in addition to localize the search for maxima in the parameter space, such estimates are exploited to put the segment into its (inferred) canonical form. Finally, an accumulator function H, in the discretized form of a matrix, is initialized; the matrix entries are in a one-to-one correspondence with the cells of the discretization performed in the previous step.

- *Estimation of the accumulator function.* An entry of the accumulator function H is increased by 1 each time the HT of a point intersects the corresponding cell. The evaluation of the intersections of Hough transforms with cells changes depending on whether the family of surfaces is described in implicit or parametric form. For surfaces expressed in implicit form, we take advantage of the local approximation of the zero loci of a set of analytic functions in terms of series of Taylor, following the strategy defined in [33] that exploits the symbolic calculation. In case of surfaces in parametric form we adapt to surfaces the method devised for curves in [30]. In particular, if the system of equations defining the family can be analytically solved with respect to the parameters $\mathbf{a}$, we calculate automatically a sample of Γ_P by exploiting the Moore-Penrose pseudo-inverse of the matrix that defines the coefficients of $\mathbf{a}$. The evaluation of the intersection is translated into an inequality between the components of the sample points of Γ_P and the coordinates of the cells endpoints. In both cases, the families of primitives are considered in a canonical form (i.e., with centers or vertices in the center of Cartesian coordinates and the normal or the principal axis aligned to the z axis of the Cartesian axes), since it limits the number of parameters necessary for the HT.

- *Selection of potential fitting primitives.* The cells corresponding to the peak values of the accumulator function H are identified. When the set of points describes a single surface profile, the HT traditional formulation focuses on the maximum value of the accumulator function H – there might exist several maxima. On the contrary, if the input point cloud is composed of different primitives, the peaks of H identify the potential primitives that might fit different parts. To select these peaks we observe that the peaks of the accumulator function corresponding to the identification of a primitive rise distinctly with respect to their neighbours and are well characterised as isolated peaks. In formal terms, we translated this observation into identifying peaks that have high *topological persistence*[1]. In our implementation, the peaks that correspond to primitives are automatically recognised by keeping the local maxima with a persistence higher than 10% of the maximum value of H, using the algorithm for persistent maxima proposed in [6]. The coordinates of the cell centres of the maxima or the peaks of the accumulator function correspond to the parameters of potentially recognised surface primitives. Figure IV.1(b) shows the accumulator function for a point cloud captured from the gear of Figure IV.1(a). The height represents the value of the accumulator function. Three peaks indicate three potential solutions corresponding to the cylinders shown in Figure IV.1(c): the colours red, green and blue represent the three cylinders, in descending order with respect to the H value .

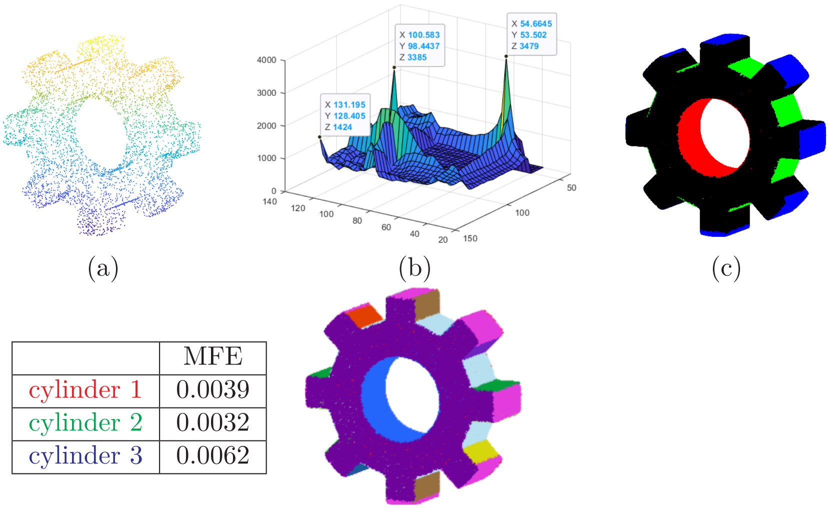
- *Evaluation of the approximation accuracy.* To measure the recognition accuracy of a specific primitive, we select the set of input points $\mathcal{X}$ closer to such a primitive than a given threshold and study its density via the k-Nearest Neighbor algorithm (see, for example, [15]). If $\mathcal{X}$ is sparse, the recognised primitive is considered a false positive; otherwise, for each dense subset $\mathcal{X}_i \subset \mathcal{X}$ we define the *Mean Fitting Error* (MFE) as:

$$E(\mathcal{X}_i, \mathcal{P}) := \frac{1}{|\mathcal{X}_i|} \sum_{\mathbf{x} \in \mathcal{X}_i} d(\mathbf{x}, \mathcal{P})/l_i, \tag{IV.1}$$

  where $\mathcal{P}$ is the current primitive, d is the Euclidean distance, and l_i is the diagonal of the minimum bounding box containing $\mathcal{X}_i$. An example of this quality measure is provided in Figure IV.1(d) for the three identified cylinders, depicted in red, green, and blue colours. What can happen when the selected family of primitives does not fit any part of the point cloud? There are two possibilities:

  - The accumulator function is identically zero, with the result that its persistence is zero; therefore the selection of potential fitting primitives returns the empty solution.
  - The accumulator function H does not present significant peaks, resulting in false positives which are identifiable by studying the sparsity of the fitted points. More precisely, given a set of dense points $\mathcal{X}_i$ and two candidate primitives $\mathcal{P}_{i,1}$ and $\mathcal{P}_{i,2}$, we first calculate the fitting errors $E(\mathcal{X}_i, \mathcal{P}_{i,1})$ and $E_i(\mathcal{X}_i, \mathcal{P}_{i,2})$ between each primitive and $\mathcal{X}_i$; the primitive having lowest error is kept.

[1] The notion of topological persistence was introduced in [13] for encoding and simplifying the points of a filtration f by classifying them as either a feature or noise depending on its lifetime or persistence within the filtration. In practice, given a pair of points p and q, their persistence is defined as $f(p) - f(q)$. Pairing is defined in terms of topological connection between the points, for details on topological persistence and saliency, we refer to [12, 13]. In our case, the domain is represented by the grid of T, the role of filtration is played by the accumulator function H and we are interested only in the peaks of H.



(a) (b) (c)

	MFE
cylinder 1	0.0039
cylinder 2	0.0032
cylinder 3	0.0062

(d) (e)

Figure IV.1: An example of applications of our recognition algorithm. In (a) an example of point cloud captured from a gear; in (b) the accumulator function associated with the search for cylinders and the peaks found by the method for persistent maxima; in (c) the cylinders corresponding to the three peaks. The MFEs for the cylinders, as defined in Equation IV.1, are reported in (d). The outcome of the fitting by 17 segments is shown in (e). To enhance the visibility, plots in (c) and (e) present an increased point thickness compared to (a).

> Finally, we evaluate a global approximation accuracy of a model by considering the minimum, the maximum and the mean of all the MFEs of its segments.

Algorithm IV.1 returns the parameters of the geometric primitives and the corresponding points that possibly fit them. At each step, the points that fit some primitive are discarded from the recognition process updating the set of points PC accordingly; then, the algorithm iterates as long as some geometric primitives are recognised or too few points left. Note that, if more geometric primitives potentially fit the same region of the point cloud, we select the one with the minimum approximation accuracy MFE.

The result of this procedure is the decomposition of an input point cloud into several subsets, called *surface segments* or simply *segments*, in such a way that points of the same segment are well approximated by the same primitive. Figure IV.1(e) shows an example of this result. In this case, we have divided the initial point cloud into different types of primitives (cylinders, axis-aligned planes and non-axis-aligned planes) obtaining a segmentation of 17 segments. Note that the method is able to group some of the non-adjacent parts, which belong to the same primitive as in the example of the two external cylinders and of the axis-aligned planes that form four couples. The approximation accuracy

```
Input: A set of points $PC$, a family of primitives $\mathcal{F}$, a threshold of the
       fitting accuracy $\varepsilon$.
Output: The list Sol of parameters of the primitives in $\mathcal{F}$, the
        corresponding segments K and the approximation accuracy
        MFE.

Variable initialization:
{Sol} ← $\emptyset$, {K} ← $\emptyset$, {MFE}←$\emptyset$;
$\mathcal{T}$=InitParameterSpace($PC$);
$\mathcal{H}^0$=InitAccumulatorFunction($\mathcal{T}$);

Computation of the accumulator function:
$\mathcal{H}$=EstimateAccumulatorFunction($\mathcal{H}^0$,$\mathcal{F}$);

Selection of potential fitting primitives:
for $\bar{\mathbf{m}} \in PersistentMaxima(\mathcal{H})$ do
    $\mathcal{P} \leftarrow \mathcal{F}(\bar{\mathbf{m}})$;
    $\mathcal{X} \leftarrow \{\mathbf{x} \in$ the whole point cloud such that $d(\mathbf{x}, \mathcal{P}) \leq \varepsilon\}$;

    Evaluation of the approximation accuracy:
    for $\mathcal{X}_i \subset \mathcal{X}$, $\mathcal{X}_i$ dense do
        MFE$_i$=EvalutateApproximationAccuracy($\mathcal{X}_i$,$\mathcal{P}$);
        {Sol} ← $\bar{\mathbf{m}}$, {K} ← $\mathcal{X}_i$, {MFE} ← $MFE_i$;
    end
end
return {Sol, K, MFE}
```

Algorithm IV.1: HT-driven primitive fitting.

is provided in Table IV.3 (first column), listing the minimum, the average and the maximum value of the MFE for the 17 primitives recognised on the model.

IV.2.4 Computational complexity

The formulation of the HT for surface primitives embedded in $\mathbb{R}^3$ simply extends the HT definition of plane curves in $\mathbb{R}^2$. The quantization of the region of the parameters determines the size of the accumulator function and dominates the computational space and cost of the Hough transform. Increasing the number of parameters, the size of the accumulator function increases accordingly; thus, it is necessary to balance the samples for each parameter and their number. To reduce the computational cost, the primitives are put in canonical position so that the number of 3 parameters is not exceeded, see Appendix IV.A. Complementarily, an adaptive approach as in [21] can be used, to avoid a brute-force tensor-product split of the parameter space. In summary, denoting M the number of elements of the space of parameters T, the overall computational complexity is $\mathcal{O}(ML)$, where L represents the number of elements of the cluster on which we evaluate the HT accumulator function. However, even sharing the same theoretical complexity, the HT for surfaces in implicit form uses symbolic computations that, in practice, could become a computational bottleneck for densely sampled models.

IV.3 Hierarchical primitive clustering

In this section, we propose the use of cluster analysis to mathematically formulate queries, by means of which further information can be extracted from the input data.

IV.3.1 Background

The term *cluster analysis* (or *clustering*) identifies a family of well-established unsupervised statistical learning techniques. Given a set $X = \{x_1, \ldots, x_N\}$, the scope of (hard) clustering is to gather its elements into a certain number of classes (or *clusters*), in such a way that all elements of a cluster share the same (or closely related) properties (see, for example, [16]). A clustering method can be defined by three main ingredients:

- Firstly, one needs to choose a *dissimilarity* $d : X \times X \to [0, +\infty)$, which is tasked with mathematically encoding which elements are closer to each other. A dissimilarity gives greater importance to certain properties while penalizing others; its choice strongly depends on the problem clustering is applied to. From a mathematical point of view, the use of a dissimilarity rather than a distance leads to a generalized metric space [37].

- Secondly, the chosen dissimilarity needs to be generalized to $D : 2^X \times 2^X \to [0, +\infty)$, to compare any pairs of subsets of X; not only are we interested in comparing elements, but also clusters of elements.

- Lastly, an algorithm for grouping the data through the map D has to be designed, in order to reach a faithful partition of X after a finite number of iterations. Hierarchical approaches, for example, build a hierarchy of clusters which can be easily illustrated by means of a dendrogram; they are often preferred to other methods, because they presuppose very little a-priori knowledge (see, for example, [26]).

IV.3.2 Application to CAD geometric primitives

In our settings, we are interested in grouping those surface primitives identified by the HT with respect to different queries, in order to extract further information. Some examples of simple interrogations are hereby presented.

- *Primitives lying on the same surface.* In this case, the purpose is to define complete primitives. For instance, one can identify all primitives lying on the same plane by comparing (normalized) coefficient vectors by means of the well-known Euclidean distance.

- *Primitives lying on the same surface, up to a roto-translation.* Unlike the previous point, we here aim to assess similarity between primitives regardless of their position in space. We can easily exploit the canonical representations provided by the HT, ignoring roto-translation terms and using the Euclidean distance on the remaining coefficients. In this way we can, for instance, look for holes of the same (or approximately the same) dimensions in a given model.
- *Primitives lying on the same surface, up to other rigid transformations.* The study of any isometries besides roto-translations, such as reflection symmetries, can be performed by considering O(3)/SO(3).

We here apply a well-known (hierarchical) clustering approach, the *complete-linkage*, to compare clusters and build a dendrogram. The method starts with singletons as clusters, and proceeds by merging, step by step, those clusters that are the closest with respect to the map

$$D(C_i, C_j) := \max_{\mathbf{x}_k \in C_i, \mathbf{x}_l \in C_j} d(\mathbf{x}_k, \mathbf{x}_l),$$

where C_i, C_j is a given pair of clusters (of primitives). The use of complete-linkage is here justified by the need of penalizing chaining effects. As an example, we post-process the planar primitives from Figure IV.1(e). Figure IV.2(a) shows the correct identification of all non-axis-aligned complete primitives, resulting in the final clustering of Figure IV.2(b). By analyzing the dendrogram, one can quantify to what extent a set of planar primitives should be clustered together or not, thus revealing possible regions where the point cloud has a lower precision.

Note that cutting the dendrogram with increasing thresholds corresponds to weaken the conditions imposed as query.

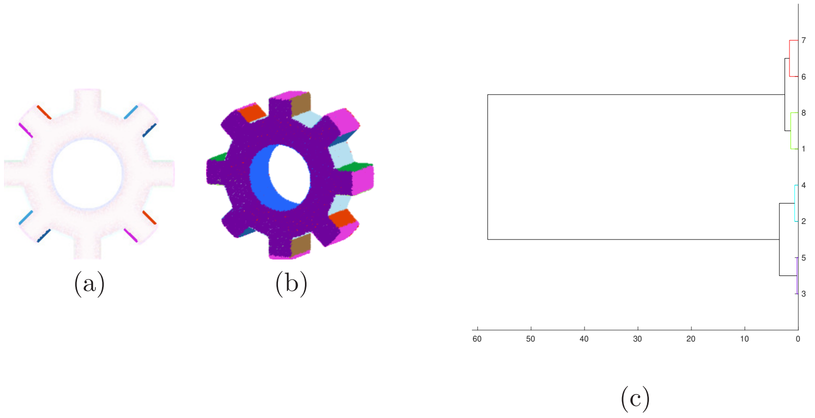
IV.3.3 Computational complexity

Although the dissimilarity-matrix assembly for a set $X = \{x_1, \ldots, x_N\}$ costs $\mathcal{O}(N^2)$, one may note that each entry is computed independently; the task is therefore *embarrassingly parallel*.

An agglomerative hierarchical clustering approach requires, in its naive implementation, $\mathcal{O}(N^3)$ operations (see, for example, [10]). When it comes to complete-linkage clustering, one can consider more efficient implementations, such as the one proposed in [11], which costs $\mathcal{O}(N^2)$.

IV.4 Experimental results

The Hough-based recognition method is here applied to point clouds from different CAD repositories (e.g., [5, 19, 35]).



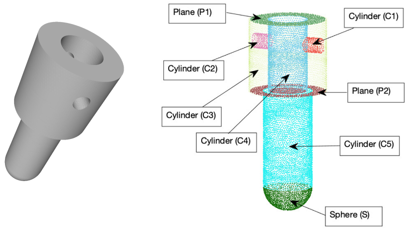
(a) (b)

(c)

Figure IV.2: Post-processing applied to the segmented point cloud of the gear of Figure IV.1(a). In (a), the clusters of segments obtained by applying hierarchical cluster analysis are highlighted by opaque colours, while semi-transparency is used for those segments not requiring post-processing. The result of combining HT and clustering is shown in (b). In (c) an example of dendrogram, whose leaves are the opaque segments shown in (a), is provided.

First, we validate our recognition method by testing it on models from the ground truth proposed in [19], which were selected because containing all common CAD geometric primitives – plane, sphere, cylinder, cone and torus. A first example is provided in Figure IV.3. The point cloud, corresponding to the set of vertices of the original triangle mesh, is segmented in 8 surface segments: 5 cylinders, 2 planes and 1 sphere. The accuracy of our method is proved by comparing the parameters from the HT with those provided in the database. The second example is presented in Figure IV.4. Here, the vertices are segmented into 9 surface segments: 4 cylinders, 1 torus, 3 axis-aligned planes and 1 cone. Again, we are able to recognise all primitives up to a small error in the parameters, with respect to those provided in the dataset.

Figure IV.5(a-d) shows point clouds that can be segmented into complete geometric primitives without the need for the clustering strategy. The first example, shown in Figure IV.5(a), consists of 11 segments – 3 cylinders and 8 planes – and it highlights the robustness of our segmentation method when it comes to detecting intersecting cylinders, here arranged similarly to a Steinmetz solid. Complete geometric primitives, each of which represented by a specific colour, are automatically detected. Another point cloud, displayed in Figure IV.5(c), contains 5 segments: 2 tori, 1 cylinder and 2 axis-aligned planes. As for the previous case, the HT-based recognition successfully detects all the primitives, even those made up of non-adjacent parts. Figures IV.5(b,d) present point clouds with parts recognised by complex analytical surfaces. In particular, in the example of Figure IV.5(b) we identify the yellow segment by a surface

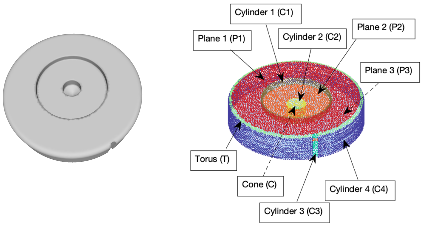


(a) (b)

equation	code	ground truth	HT parameters
$x^2+y^2=r^2$	C1	$r=1.50$	$r=1.50$
	C2	$r=8.00$	$r=8.00$
	C3	$r=4.00$	$r=4.00$
	C4	$r=1.50$	$r=1.50$
	C5	$r=5.00$	$r=5.00$
$z=k$	P1	$k=20.00$	$k=20.00$
	P2	$k=35.00$	$k=35.00$
$x^2+y^2+z^2=r^2$	S	$r=5.00$	$r=5.05$

(c)

Figure IV.3: In (a) a CAD model from the benchmark in [19]; in (b) the vertices of its triangle mesh segmented into 8 surface segments; in (c), for each primitive, the HT parameters are compared with those provided by the database



(a) (b)

equation	code	ground truth	HT parameters
$x^2+y^2=r^2$	C1	$r=15.01$	$r=15.01$
	C2	$r=3.97$	$r=3.97$
	C3	$r=1.28$	$r=1.26$
	C4	$r=29.25$	$r=29.25$
$z=k$	P1	$k=12.00$	$k=12.00$
	P2	$k=9.08$	$k=9.08$
	P3	$k=0.00$	$k=0.00$
$(R-\sqrt{x^2+y^2})^2+z^2-r^2=0$	T	$R=27.25$ $r=2.00$	$R=27.25$ $r=2.00$
$x^2+y^2+a(z-b)^2=0$	C	$a=-2.77$ $b=2.19$	$a=-2.81$ $b=2.22$

(c)

Figure IV.4: In (a) a CAD model from the benchmark in [19]; in (b) the vertices of its triangle mesh segmented into 9 surface segments; in (c), for each primitive, the HT parameters are compared with those provided by the database

of revolution from Table IV.2(c). The remaining points are segmented into 2 cylinders and 2 planes. In the point cloud shown in Figure IV.5(d), we extract the gold part with a cone having as directrix a 5−convexity curve, while the blue and the green segments are fitted with cylinders having the same directrix; the equations of these surfaces can be found in Table IV.2(a,b). These two last point clouds have been segmented into 5 and 7 clusters, respectively.

Figures (IV.6-IV.11) show point clouds wherein the segments produced by the HT approach are post-processed by hierarchical clustering. In Figure IV.6(b), the input point cloud is first segmented into 20 surface primitives via the HT-recognition algorithm (cylinders and planes), which are then grouped by clustering. As expected, no pair of primitives lying on the same surface is found. Despite the presence of imperfections in the original model (see Figure IV.6(a)), we are able to correctly identify repeating primitives, here in the form of circular cylinders of equal radii. Figure IV.6(c) highlights the similarities identified by clustering cylinders: 4 in light blue, 3 in red, 2 in purple and 2 in yellow.

Figure IV.7(a) displays a model which can be accurately described by combining a portion of helical surface, as introduced in Table IV.2(d), with a

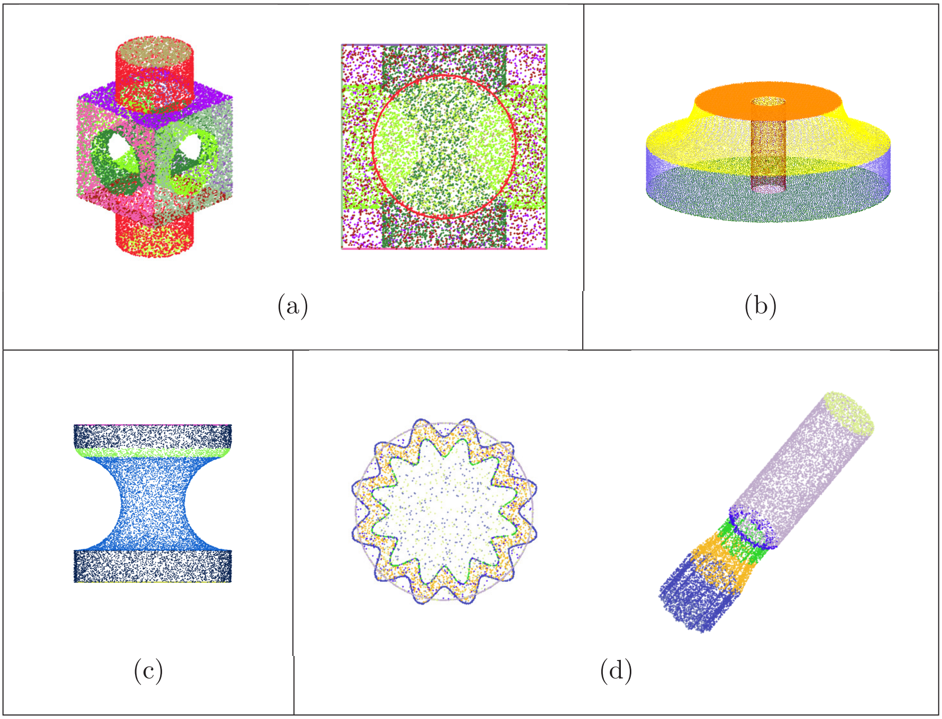


Figure IV.5: Segmentations of CAD objects point clouds. Figures (a-d) present the segmentation of 4 point clouds, ranging from common CAD primitives to more complex ones. The identification of complete segments does not require, in these cases, the application of any clustering algorithm.

pair of planes and a pair of convex combination of helices, see Table IV.2(e). The result is a segmentation of the point cloud into 6 segments, Figure IV.7(b). Two of them are then grouped by the clustering since the helical strips have the same equation up to a translation, as shown in Figure IV.7(c).

Figure IV.8 increases the difficulty by considering a point cloud containing a higher number of segments, some of which are low-quality as the small holes highlighted in Figure IV.8(a). In this example, the geometric primitives identified by the HT algorithm are cylinders, cones and planes. The result is a segmentation in 28 surface segments, see Figure IV.8(b). Note that the central hole presents some grooves, i.e., a surface detail that was not present as a geometric feature in the original CAD model; therefore, we recognise it as a cylinder and the HT is able to ignore the shallow grooves. A final application of clustering makes it possible to identify repeating primitives, up to translations. Figure IV.8(c) illustrates the similarity between 6 cylindrical holes (in green) and between other 6 cylindrical segments (in yellow).

Another example of gear is shown in Figure IV.9(a). Here, the total number of extracted geometric primitives is 68: 19 cylinders; 2 cones; 47 planes, 5 of

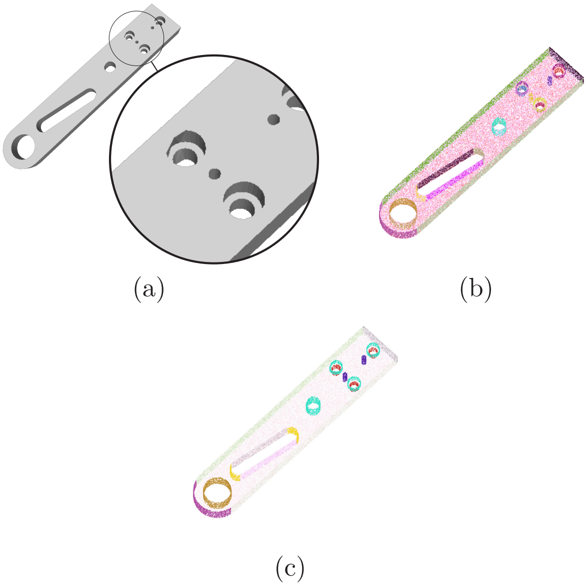

(a) (b) (c)

Figure IV.6: A linkage arm. In (a), the original model is shown, together with a magnification revealing some imperfections. The segments identified by the HT are shown in (b) by different colours. (c) draws the attention to cylinders, among which we can recognize segments lying on the same primitive, up to a translation.

which are axis-aligned. The result is presented in Figure IV.9(b). As highlighted by the original model, in this prototypical version of NuGear, cylinders are roughly approximated by a series of planar primitives; in this specific case, we fit a cylinder instead of many planes, since the former can describe a much larger area without significantly increasing the error. Unlike the gear shown in Figure IV.1(e), the surface segments identified by non-axis-aligned planes are not clustered in pairs; this is because the tooth inclination prevents any alignment. Clustering identifies here a similarity between the 12 cylindrical holes – up to translations – and between 2 external cylinders. Figure IV.9(c) shows this result, colouring the holes in black and the external cylinders in yellow.

Finally, Figure IV.10(a) shows a mechanical part. The HT segmentation of the input point cloud consists of 50 surface segments, all of which are tori, cylinders and planes, see Figure IV.10 (b). The clustering method is able to identify the similarity between 8 red cylindrical holes and between 2 blue cylindrical segments, up to translations – see Figure IV.10(c). Finally, 2 tori are clustered together, because they lie on the same surface primitive up to roto-translations.

Table IV.3 summarises the mean fitting error for all the primitives found in the point clouds processed.

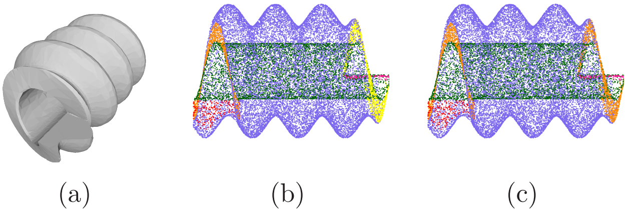


Figure IV.7: A screw-like part. The original model, (a), is sampled. The surface segments detected via HT are shown in (b) by different colours: a helical surface (in purple), two planes (in red and magenta), and two helical strips (in orange and yellow). Although no pair of segments lies on the same parametrized surface, the 2 helical strips have the same equation up to a translation, as shown in (c).

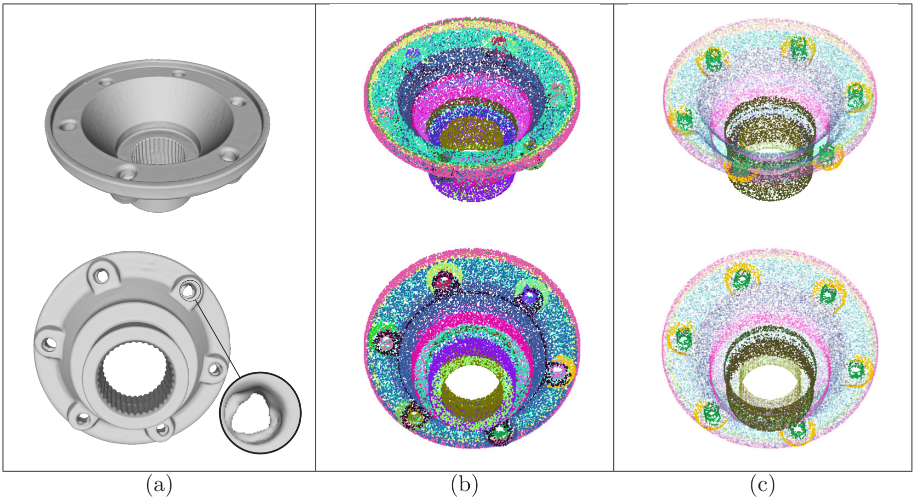


Figure IV.8: A carter. In (a), the original model is shown. The surface segments found by means of the HT approach are depicted in (b), while (c) shows the result of primitive clustering, when one is interested in identifying the same primitive up to a translational transformation. Different rows corresponds to different points of view.

IV.5 Discussion

The use of the HT naturally leads to a robust surface recognition pipeline as shown in the examples of Figures IV.6, IV.8 and IV.11. In particular, in the model of Figure IV.11, the HT recognition correctly identifies the cylinder that fits the central part, without being negatively influenced by the letters in relief – see the original model in Figure IV.11(a). The point cloud is decomposed into 38 segments: 23 cylinders with different axes, 10 planes, 4 cones, and 1 torus that automatically identifies the top and bottom of the cylinder with the “GRAYLOC"

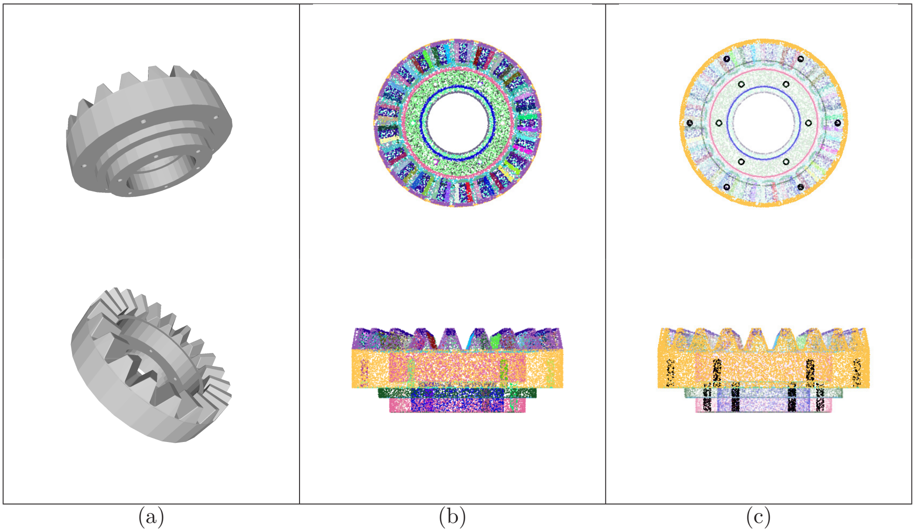


Figure IV.9: A prototype of the NuGear component, courtesy of STAM S.r.l. (Genoa, Italy). The original model is shown in (a). The decomposition in clusters of points produced by the HT approach is given in (b). The output of the additional clustering procedure is shown in (c), it highlights the similarity between 12 cylindrical holes (in black) and between two cylinders (in yellow).

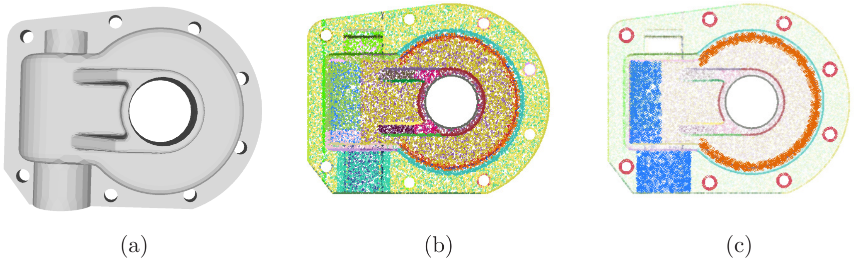


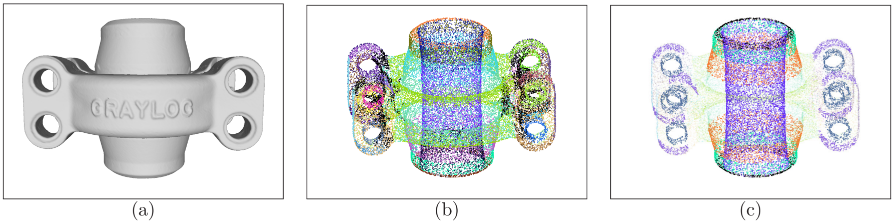
Figure IV.10: A mechanical part. In (a) the original model is shown, while in (b) the decomposition of the corresponding point cloud into segments produced by the HT. In (c) the result of the clustering procedure: 8 cylindrical holes, in red, have a high similarity, up to translations; the same applies for 2 cylindrical segments, in blue; 2 tori, in orange, identify the same primitive, up to a roto-translation.

inscription (see Figure IV.11(b)). The application of the hierarchical clustering allows to group together: 8 grey cylindrical holes (up to roto-translations); 8 purple cylindrical segments; 2 aquamarine circular cylinders; 3 violet circular cylinders; 2 orange circular cones (up to a reflection); 2 black cones (up to a reflection). Moreover, the small imperfections of the manufacture on the central part of the body (recognised by vertical cones, cylinders and tori at the top

Table IV.3: Statistics of the MFEs for all models presented in the paper. Being MFE normalized by definition, we can conclude that the maximum error throughout the paper is 4.48%, which corresponds to the noisy holes in the carter model of Figure IV.8.

	Fig. IV.1(a)	Fig. IV.5(a)	Fig. IV.5(b)	Fig. IV.5(c)	Fig. IV.5(d)
min(E_i)	0.0008	0.0009	0.0006	0.0004	0.0020
mean(E_i)	0.0060	0.0024	0.0031	0.0031	0.0053
max(E_i)	0.0226	0.0056	0.0071	0.0059	0.0081
	Fig. IV.6	Fig. IV.7	Fig. IV.8	Fig. IV.9	Fig. IV.10
min(E_i)	0.0004	0.0033	0.0013	0.0006	0.0008
mean(E_i)	0.0098	0.0086	0.0107	0.0053	0.0035
max(E_i)	0.0300	0.0154	0.0448	0.0178	0.0057

and bottom) and on the lateral holes do not prevent the clustering technique to appropriately aggregate the corresponding segments, correctly dealing with rotations and reflections. However not everything is recognised, mainly in complex objects such as the object of Figure IV.11. In fact the black dots in Figure IV.11(b) correspond to points that do not fit to any of the geometric primitives at our disposal, and originate from irregular elements that act as a connection between better defined segments.



(a) (b) (c)

Figure IV.11: A clamp connector. In (a) the original model. In (b) the decomposition of the corresponding point cloud into 38 segments provided by the HT procedure. In (c), the final grouping obtained by clustering, consisting of 6 groups of primitives (here, singletons of segments are transparent).

A further example of the robustness of the method to noise is presented in Figure IV.12 and quantitatively analysed in Table IV.4. In this example, we perturb the point cloud of Figure IV.5(c) by adding zero-mean Gaussian noise of standard deviation: 0.01 (a), 0.05 (b), 0.10 (c) and 0.20 (d). The first row shows the points classified as noise (in black) and the segments found by our method in the same image, for each level of noise. The second row focuses only on the points that fit the identified primitives, thus providing a denoised segmentation. The robustness to noise is quantitatively studied in Table IV.4: the parameters obtained in the original point cloud are there compared with those found in the perturbed point clouds.

Figure IV.13 makes a comparison with the RANSAC-based method introduced in [31] and with the patch aggregation approach in [20]. In this

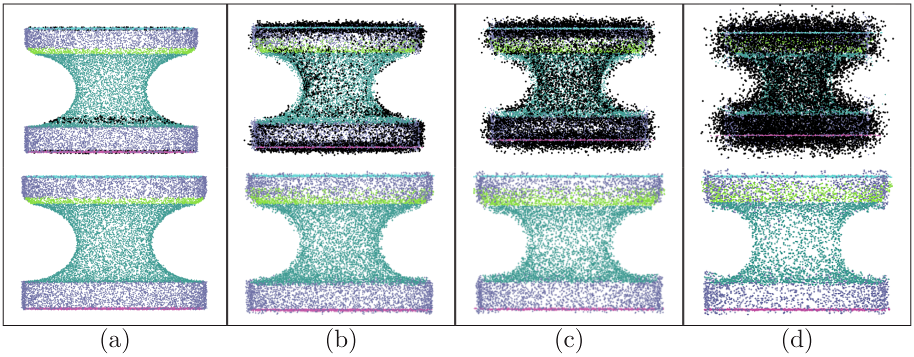


Figure IV.12: The point cloud in Figure IV.5(c) is perturbed by adding zero-mean Gaussian noise of standard deviation: 0.01 (a), 0.05 (b), 0.10 (c) and 0.20 (d). The first row superimposes the points identified as noise (in black) to the final segments found by our method; the second row depicts the points that fit the primitives found and provides a denoised segmentation.

Segment	Original	Noise (a)	Noise (b)	Noise (c)	Noise (d)
Cylinder	$r = 1.80$	$r = 1.79$	$r = 1.78$	$r = 1.79$	$r = 1.78$
Plane 1	$k = -1.28$	$k = -1.28$	$k = -1.29$	$k = -1.28$	$k = -1.31$
Plane 2	$k = 1.28$	$k = 1.28$	$k = 1.28$	$k = 1.28$	$k = 1.26$
Torus 1	$R = 1.49$ $r = 0.72$	$R = 1.49$ $r = 0.73$	$R = 1.47$ $r = 0.70$	$R = 1.48$ $r = 0.67$	$R = 1.56$ $r = 0.74$
Torus 2	$R = 1.05$ $r = 0.79$	$R = 1.09$ $r = 0.78$	$R = 1.02$ $r = 0.78$	$R = 1.17$ $r = 0.74$	$R = 1.13$ $r = 0.80$

Table IV.4: Parameter comparison between the original point cloud from Figure IV.5(c) and the perturbed versions from Figure IV.12.

comparison, we have focused on models that merely present simple primitives, to show that even on these our approach obtains better results than the others; on the other hand, the capability to handle more complex primitives is undoubtedly an added value of our method. Similarly to [20], we use different colours to represent different primitive typologies. For a simple model like the block of Figure IV.13(a), all methods provide acceptable segmentations, although RANSAC [31] misclassifies some primitives. For the remaining more complex models, our method outperforms the competitors. In Figure IV.13(b), our approach is the only capable to correctly identify portions of tori, misclassified by Le and Duan [20] and partly unsegmented by RANSAC [31]. Figures IV.13(c-d) show a RANSAC [31] tendency to over-segment and misclassify complex models. While Le and Duan [20] obtain considerably improved results, their algorithm misses a thin cylinder (see Figure IV.13(c)) and all the torii present in both models.

Finally, Table IV.5 shows the computational time for the HT based recognition algorithm, considering both the simple primitives and the complex ones. The experiments are performed on a laptop equipped with an Intel Core i7 processor

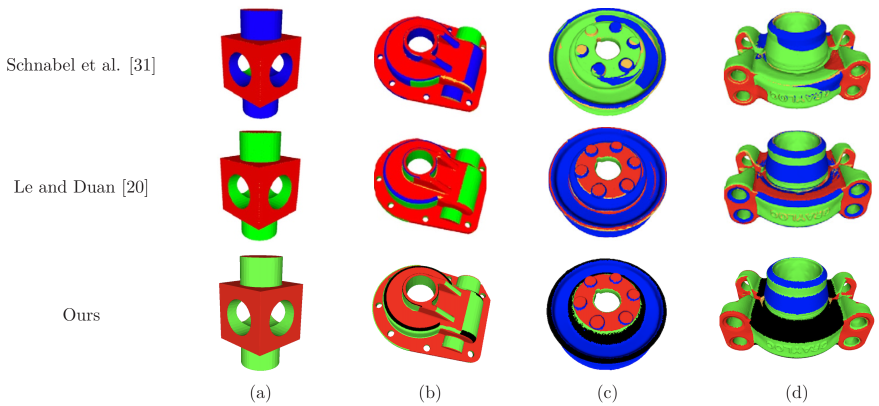


Figure IV.13: Primitive type recognition: a comparison between our approach, a RANSAC-based segmentation [31], and the method in [20]. Different colours correspond to different primitive types: planes (red), cylinders (green), cones (blue), tori (black), unsegmented (yellow).

(at 1.3 GHz).

Table IV.5: Computational time, in seconds, for the recognition of some typologies of primitives shown in the examples.

	# points	# segments	planes	cylinders	spheres	cones	tori	complex	total
Fig. IV.1(e)	25,000	17	0.19	12.48	–	–	–	–	12.67
Fig. IV.3	15,216	8	0.04	4.2	12.81	–	–	–	17.05
Fig. IV.4	15,022	9	0.04	8.22	–	6.64	0.11	–	15.01
Fig. IV.5(a)	25,000	11	0.22	33.01	–	–	–		33.23
Fig. IV.5(b)	67,777	5	0.11	37.88	–	–	–	8.14	46.13
Fig. IV.5(c)	25,000	5	0.06	5.70	–	–	0.20	–	5.96
Fig. IV.5(d)	25,000	7	0.23	3.83	–	–	0.20	1.13	5.96
Fig. IV.6(b)	50,000	20	0.29	2.34	–	–	–	–	2.63
Fig. IV.7(b)	25,000	6	0.15	6.07	–	–	–	44.56	50.78
Fig. IV.8(b)	50,000	28	0.11	14.91	–	39.31	–	–	54.33
Fig. IV.9(b)	50,000	68	0.38	15.54	–	0.57	–	–	16.49
Fig. IV.10(b)	50,000	50	0.24	24.41	–	–	1.42	–	26.07
Fig. IV.11(b)	50,000	38	0.13	17.32	–	37.86	0.07		55.38

IV.6 Concluding remarks

The examples in Sections IV.4 and IV.5 highlight the ability of this method to deal with simple and complex geometric primitives. In addition to the simple CAD primitives that are typical of the *Constructive Solid Geometry* (CSG), the use of the HT in its generalised version provides a larger set of primitives that includes algebraic surfaces and transcendental or exponential ones, as long as they possess a parametric or implicit expression.

Moreover, the combination of the HT with a clustering strategy yields the aggregation of multiple segments; indeed, it permits to recognise as a unique segment parts of the same primitive that are non-contiguous and to identify patterns of repeated segments, even if they slightly differ.

In conclusion, the combination of a primitive fitting strategy based on HT and a hierarchical clustering is leading to a reliable tool suitable for several CAD applications, such as reverse engineering and model annotation. As a possible future development, we foresee to integrate our method with other reasoning, for instance by adding spatial relationships, e.g., the inclusion between parts, to characterise more complex assemblies of parts and support automatic model annotation.

Acknowledgements. This work has been developed in the CNR IMATI research activities DIT.AD004.100 and DIT.AD021.080.001.

References

[1] Attene, M. and Patanè, G. “Hierarchical Structure Recovery of Point-Sampled Surfaces”. In: *Computer Graphics Forum* vol. 29, no. 6 (2010), pp. 1905–1920.

[2] Ballard, D. “Generalizing the Hough transform to detect arbitrary shapes”. In: *Pattern Recognition* vol. 13, no. 2 (1981), pp. 111–122.

[3] Beltrametti, M. et al. “Moore–Penrose approach in the Hough transform framework”. In: *Applied Mathematics and Computation* vol. 375 (2020).

[4] Beltrametti, M. C. and Robbiano, L. “An algebraic approach to Hough transforms”. In: *Journal of Algebra* vol. 371 (2012), pp. 669–681.

[5] Bespalov, D. et al. “Benchmarking CAD Search Techniques”. In: *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling.* SPM ’05. Cambridge, Massachusetts: Association for Computing Machinery, 2005, pp. 275–286.

[6] Biasotti, S. et al. “Tracking the evolution of rainfall precipitation fields using persistent maxima”. In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference.* 2016, pp. 29–37.

[7] Borrmann, D. et al. “The 3D Hough Transform for Plane Detection in Point Clouds: A Review and a New Accumulator Design”. In: *3D Res.* vol. 2, no. 2 (Mar. 2011).

[8] Buonamici, F. et al. “Reverse engineering modeling methods and tools: a survey”. In: *Computer-Aided Design and Applications* vol. 15, no. 3 (2018), pp. 443–464.

[9] Camurri, M., Vezzani, R., and Cucchiara, R. “3D Hough Transform for Sphere Recognition on Point Clouds”. In: *Mach. Vision Appl.* vol. 25, no. 7 (Oct. 2014), pp. 1877–1891.

[10] Day, W. and Edelsbrunner, H. “Efficient algorithms for agglomerative hierarchical clustering methods”. In: *Journal of Classification* vol. 1, no. 1 (1984), pp. 7–24.

[11] Defays, D. “An efficient algorithm for a complete link method”. In: *The Computer Journal* vol. 20, no. 4 (Jan. 1977), pp. 364–366.

[12] Doraiswamy, H. et al. “Topological saliency”. In: *Computers & Graphics* vol. 37, no. 7 (2013), pp. 787–799.

[13] Edelsbrunner, Letscher, and Zomorodian. “Topological Persistence and Simplification”. In: *Discrete Comput. Geom.* vol. 28, no. 4 (Nov. 2002), pp. 511–533.

[14] Fernandes, L. A. and Oliveira, M. M. “A general framework for subspace detection in unordered multidimensional data”. In: *Pattern Recognition* vol. 45, no. 9 (2012), pp. 3566–3579.

[15] Friedman, J. H., Bentley, J. L., and Finkel, R. A. “An Algorithm for Finding Best Matches in Logarithmic Expected Time”. In: *ACM Trans. Math. Softw.* vol. 3, no. 3 (Sept. 1977), pp. 209–226.

[16] Gan, G., Ma, C., and Wu, J. *Data Clustering Theory. Algorithms and Applications.* ASA-SIAM series on statistics and applied probability, 2007.

[17] Hough, P. V. C. *Method and means for recognizing complex patterns.* US Patent 3069654. Dec. 1962.

[18] Kaiser, A., Ybanez Zepeda, J. A., and Boubekeur, T. “A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data”. In: *Computer Graphics Forum* vol. 38, no. 1 (2019), pp. 167–196.

[19] Koch, S. et al. “ABC: A Big CAD Model Dataset For Geometric Deep Learning”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2019.

[20] Le, T. and Duan, Y. “A primitive-based 3D segmentation algorithm for mechanical CAD models”. In: *Computer Aided Geometric Design* vol. 52-53 (2017). Geometric Modeling and Processing 2017, pp. 231–246.

[21] Li, H., Lavin, M. A., and Le Master, R. J. “Fast Hough transform: A hierarchical approach”. In: *Computer Vision, Graphics, and Image Processing* vol. 36, no. 2 (1986), pp. 139–161.

[22] Li, L. et al. “Supervised Fitting of Geometric Primitives to 3D Point Clouds.” In: *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* Computer Vision Foundation / IEEE, 2019, pp. 2652–2660.

[23] Li, Y. et al. “GlobFit: Consistently Fitting Primitives by Discovering Global Relations”. In: *ACM Trans. Graph.* vol. 30, no. 4 (July 2011).

[24] Limberger, F. A. and Oliveira, M. M. “Real-time detection of planar regions in unorganized point clouds”. In: *Pattern Recognition* vol. 48, no. 6 (2015), pp. 2043–2053.

[25] Lupinetti, K. et al. “Content-based CAD assembly model retrieval: Survey and future challenges”. In: *Computer-Aided Design* vol. 113 (2019), pp. 62–81.

[26] Murtagh, F. "A Survey of Recent Advances in Hierarchical CLustering Algorithms". In: *The Computer Journal* vol. 26 (1983), pp. 354–359.

[27] PenroseR. "On best approximate solutions of linear matrix equations". In: *Mathematical Proceedings of the Cambridge Philosophical Society* vol. 52, no. 1 (1956), pp. 17–19.

[28] Rabbani Shah, T. and van den Heuvel, F. "Efficient Hough transform for automatic detection of cylinders in point clouds". Undefined/Unknown. In: *ISPRS Laser Scanning 2005*. Ed. by Vosselman, G. and Brenner, C. null ; Conference date: 12-09-2005 Through 14-09-2005. ISPRS Working Groups, 2005, pp. 60–65.

[29] Raffo, A., Barrowclough, O. J., and Muntingh, G. "Reverse engineering of CAD models via clustering and approximate implicitization". In: *Computer Aided Geometric Design* vol. 80 (2020), p. 101876.

[30] Romanengo, C., Biasotti, S., and Falcidieno, B. "Recognising decorations in archaeological finds through the analysis of characteristic curves on 3D models". In: *Pattern Recognition Letters* vol. 131 (2020), pp. 405–412.

[31] Schnabel, R., Wahl, R., and Klein, R. "Efficient RANSAC for Point-Cloud Shape Detection". In: *Computer Graphics Forum* vol. 26, no. 2 (June 2007), pp. 214–226.

[32] Shikin, E. V. *Handbook and atlas of curves*. Taylor & Francis, 1995.

[33] Torrente, M.-L., Biasotti, S., and Falcidieno, B. "Recognition of feature curves on 3D shapes using an algebraic approach to Hough transforms". In: *Pattern Recognition* vol. 73 (2018), pp. 111–130.

[34] Várady, T., Martin, R. R., and Cox, J. "Reverse engineering of geometric models—an introduction". In: *Computer-Aided Design* vol. 29, no. 4 (1997), pp. 255–268.

[35] Vv, A. *The Shape Repository*. http://visionair.ge.imati.cnr.it/ontologies/shapes/. 2011–2015.

[36] Wang, H. L. and Reeves, A. P. "Three-dimensional generalized Hough transform for object identification". In: *Proceedings of SPIE*. Vol. 1192. SPIE, 1990, pp. 363–374.

[37] Wilson, W. A. "On Semi-Metric Spaces". In: *American Journal of Mathematics* vol. 53, no. 2 (1931), pp. 361–373.

[38] Woodford, O. J. et al. "Demisting the Hough Transform for 3D Shape Recognition and Registration". In: *Int. J. Comput. Vis.* vol. 106, no. 3 (2014), pp. 332–341.

[39] Yan, D.-M. et al. "Variational mesh segmentation via quadric surface fitting". In: *Computer-Aided Design* vol. 44, no. 11 (2012), pp. 1072–1082.

Appendix IV.A Accelerating the Hough Transform via initial geometric estimates

As introduced in Chapter 1, the Hough transform is naturally oriented towards the recognition of a mathematical expression of the primitives. Unfortunately, the variety of primitives that can be used within the HT framework is limited by the number of parameters of the primitive itself, if it is considered in its generic space embedding. To overcome this limitation, we propose a segment pre-processing step[2] that allows us: i) to automatically centre and orient a segment so that it can be fitted with a primitive in standard form; ii) to estimate the primitive parameters so as to localize the search for the optimal solution. Once each segment has been properly roto-translated, we are able to apply the HT technique for a number of primitives otherwise non affordable in terms of computational cost and memory space occupied. For the sake of brevity, we limit the upcoming discussion to the case of simple geometric primitives, i.e, to cylindrical, conic, spherical and toric segments; similar constructions can be applied to the more general list of primitives provided in Section IV.2.2.

IV.A.1 Simple geometric primitives and their standard form

We here provide compact parametric equations for the simple geometric primitives under study.

Spheres The points on the sphere of radius $r > 0$ and center $\mathbf{c} \in \mathbb{R}^3$ can be parametrized as

$$\mathbf{p}(u, v) = \mathbf{c} + r\cos(v)\,(\cos(u)\mathbf{e}_1 + \sin(u)\mathbf{e}_2) + r\sin(v)\mathbf{e}_3,$$

where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ denotes the standard basis for $\mathbb{R}^3$. The standard form is obtained by setting $\mathbf{c} = \mathbf{0}$.

Cylinders The parametric equations of a cylinder may be written as

$$\mathbf{p}(u, v) = \mathbf{l} + r\cos(u)\mathbf{v}_1 + r\sin(u)\mathbf{v}_2 + v\mathbf{a},$$

where: $\mathbf{l}$ is the location vector defining the base plane; r is the cylinder radius; $\mathbf{a}$ is a unit vector that gives the direction of the rotational axis; $\mathbf{v}_1$ and $\mathbf{v}_2$ are chosen so that $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{a}\}$ forms an orthonormal basis. We define the standard form by setting $\mathbf{l} = \mathbf{0}$, $\mathbf{v}_i = \mathbf{e}_i$ for any i, and $\mathbf{a} = \mathbf{e}_3$. Note that this choice is purely individual, as one could impose any of the vectors $\mathbf{e}_i$ to be the standard rotational axis.

[2] The content of this section is extracted from the manuscript “Fitting and recognition of geometric primitives in segmented 3D point clouds using a localized voting procedure" by the same authors, which is currently under fast-track review at Computer Aided Geometric Design.

Cones Cones can be parametrically represented by

$$\mathbf{p}(u,v) = \mathbf{l} + (r + v\sin(\alpha))\,(\cos(u)\mathbf{v}_1 + \sin(u)\mathbf{v}_2) + v\cos(\alpha)\mathbf{a},$$

with $\mathbf{l}$, $\mathbf{v}_i$ and $\mathbf{a}$ having the same geometric meaning as for the cylinders, while r denotes the radius of the circle found by intersecting the cone and the base plane, and α gives the half-angle at the apex of the cone. We call standard form any parametrization obtained by imposing $\mathbf{v}_i = \mathbf{e}_i$ for any i, $\mathbf{a} = \mathbf{e}_3$, and by moving the cone vertex to the origin; note that the latter corresponds to set r to zero (which means that the cone vertex lies on the base plane) and further impose $\mathbf{l} = \mathbf{0}$.

Tori Tori are parametrized as

$$\mathbf{p}(u,v) = \mathbf{c} + (r_{\max} + r_{\min}\cos(v))\,(\cos(u)\mathbf{v}_1 + \sin(u)\mathbf{v}_2) + r_{\min}\sin(v)\mathbf{a},$$

where $r_{\min}$ and $r_{\max}$ are the minor and the major radii, $\mathbf{c}$ is the center of the torus, $\mathbf{a}$ is its rotational axis, and $\mathbf{v}_1$ and $\mathbf{v}_2$ are the remaining axes of the torus coordinate system. The standard forms are here expressed by setting $\mathbf{c} = \mathbf{0}$, $\mathbf{v}_i := \mathbf{e}_i$ for any i, and $\mathbf{a} := \mathbf{e}_3$.

IV.A.2 Segment pre-processing

To (recognize and) fit a surface in a point cloud, it is usually required to have some good initial estimate of the solution which is then refined. A bad initial estimate can lead to a fit of poor quality or, in the worst-case scenario, to a wrong recognition. In our case, initial estimates are fundamental for two reasons. Firstly, they allow you to put a segment in its standard form, thus reducing the number of parameters to be handled by the HT technique. Secondly, they provide a guess to the HT technique of where the optimal solution is, thus solving the problem of unboundedness of the parameter space. We now describe, for each family of geometric primitives introduced in Section IV.A.1, how the initial estimates can be computed.

IV.A.2.1 Segment centering and normal estimation via local HT fits

The steps described in this section are performed independently from the primitive type; they are preliminary to the computation of our initial estimates and, subsequently, to our final recognition.

Point cloud centering At first, the input point cloud $\mathcal{P}$ is centered, i.e., it is translated so that its barycenter coincides with the origin of the Cartesian coordinate system.

Point cloud downsampling The axis-aligned minimum bounding box of $\mathcal{P}$ is split into s equal-sized boxes. Points within the same box $j = 1, ..., s$ will be replaced (i.e., downsampled) by the point which is closest to their barycenter, so as to obtain a uniformly downsampled point cloud. This step is optional, and is meant to handle very dense point clouds with a lower execution time.

Normal estimation For each $\mathbf{p}_j \in \mathcal{P}$, we select all points within a given distance D w.r.t. the usual Euclidean metric; we denote this neighbourhood by $\mathcal{N}_j := \mathcal{N}(\mathbf{p}_j, D)$. We then apply the HT technique to $\mathcal{N}_j$ and select the most voted plane $\hat{\pi}_j$, which gives an approximation of the true tangent plane π_j at $\mathbf{p}_j$ or, equivalently, of the normal vector $\mathbf{n}_j$ at the same point. To this end, we consider the Hesse normal form

$$x \cos\theta \sin\phi + y \sin\theta \sin\phi + z \cos\phi - \rho = 0,$$

where: x, y and z are the Cartesian coordinates of a sample point; $\theta \in [0, 2\pi)$ and $\phi \in [0, \pi]$ are the polar coordinates of the normal vector to the plane; $\rho \in \mathbb{R}_{\geq 0}$ is the distance from the plane to the origin of the coordinate system. The normal at $\mathbf{p}_j$ is approximated by the vector $\hat{\mathbf{n}}_j = [\cos\hat{\theta}_j \sin\hat{\phi}_j, \sin\hat{\phi}_j \sin\hat{\theta}_j, \cos\hat{\phi}_j]$, being $\hat{\theta}_j$ and $\hat{\phi}_j$ estimates of θ_j and ϕ_j obtained via the Hough transform. More details on the application of the HT based on this parametrization can be found in [24].

Normal accuracy As a last step, we compute the accuracy of each candidate tangent plane by using the *Mean Fitting Error* (MFE), defined as:

$$\mathrm{MFE}(\mathcal{N}_j, \pi_j) := \frac{1}{|\mathcal{N}_j|} \sum_{\mathbf{x} \in \mathcal{N}_j} d(\mathbf{x}, \pi_j)/l_j,$$

where d is the Euclidean distance and l_j is the diagonal of the minimum bounding box containing $\mathcal{N}_j$. From here on, we denote $\mathrm{MFE}(\mathcal{N}_j, \pi_j)$ by MFE_j and $\mathbf{MFE} := [\mathrm{MFE}_1, \mathrm{MFE}_2, \ldots]$.

IV.A.2.2 Initial estimates

Once the candidate tangent planes have been estimated, we specialize the processing for each type of primitive.

Sphere For each $\mathbf{p}_j \in \mathcal{P}$, we sample a point $\mathbf{q}_j$ on the (candidate) tangent plane π_j. Then, we consider the plane passing through $\mathbf{p}_j$ and having normal vector $\mathbf{v}_j := \mathbf{n}_j \times \mathbf{t}_j$, where $\mathbf{t}_j = \mathbf{p}_j - \mathbf{q}_j$. A graphical illustration of the three vectors involved is given in Figure IV.14(c): the blue, green and red vectors are, respectively, $\mathbf{n}_j$, $\mathbf{t}_j$ and $\mathbf{v}_j$. This plane intersects the sphere into a set of points outlining a circle, which can be recognized through the classical HT procedure for circles, and whose approximation is evaluated by the Mean Fitting Error. In exact arithmetic, such a plane passes through the sphere center, which corresponds to the circle center as well; in floating-point arithmetic (or when the input segment is perturbed), the center and the radius of the circle give an estimate of the center and the radius of the sphere. By repeating this procedure for all points in $\mathcal{P}$ – or at least, for a representative subset of points – we obtain a set of estimates of the sphere center and radius. By thresholding the MFE, we can discard low-quality estimates; finally, by averaging over the remaining centers and radii we obtain the final estimates of the sphere center and radius, which will be denoted by $\hat{\mathbf{c}}$ and $\hat{r}$.

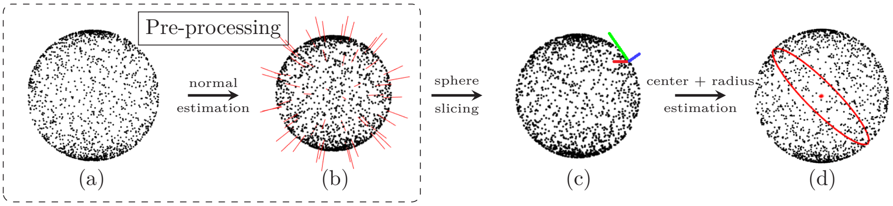


Figure IV.14: Initial estimates for a sphere. The pre-processing step centers the point cloud and approximates the normal vectors at a set of points, see (b). Given a point $\mathbf{p}_j$ and a point $\mathbf{q}_j$ on its tangent plane, (c) shows the vectors $\mathbf{n}_j$, $\mathbf{t}_j$ and $\mathbf{v}_j$ in blue, green and red, respectively. Finally, (d) shows the estimate $\hat{\mathbf{c}}$ of the center.

Cylinder We select the points corresponding to the lowest entries in **MFE**, i.e., having the most accurate estimations of the normal vector. Let $\mathbf{p}_1, \ldots, \mathbf{p}_k$ denote such points. For each pair of points $\mathbf{p}_{j_1}$ and $\mathbf{p}_{j_2}$, where $j_1, j_2 = 1, \ldots, k$ and $j_1 \neq j_2$, we consider the corresponding normal vectors $\hat{\mathbf{n}}_{j_1}$, $\hat{\mathbf{n}}_{j_2}$: their cross product, denoted $\hat{\mathbf{a}}_{j_1,j_2} := \hat{\mathbf{n}}_{j_1} \times \hat{\mathbf{n}}_{j_2}$, is an approximation of the rotational axis of the cylinder up to a translation. Figure IV.15(c) represents a simplified situation, where the two normals (in green and blue) and their cross product (colored red) are positioned so that $\hat{\mathbf{a}}_{j_1,j_2}$ determines the rotational axis; note that this choice is purely illustrative. By iterating over all possible combinations, one can obtain multiple estimates of the rotational axis; we average over all these estimates and return the resulting vector, denoted $\hat{\mathbf{a}}$.

The point cloud $\mathcal{P}$ is now rotated so that $\hat{\mathbf{a}}$ is parallel to the $z-axis$ and, subsequently, projected onto the xy-plane. To estimate the radius r and the center $\mathbf{c}$ of the projected points, which outline (arcs of) a circle if the initial point cloud originated from a circular cylinder, we detect the most voted circle by applying HT-based recognition process, see Figure IV.15(d).

Cone Similarly to the previous case, we select the points corresponding to the lowest entries in **MFE**; let $\mathbf{p}_1, \ldots, \mathbf{p}_k$ denote such points. From basic geometry we know that, in exact arithmetic, the vertex of a cone can be found by intersecting (at least) three tangent planes, see Figure IV.16(c); this is equivalent to solve a linear system $\mathbf{N}\mathbf{v} = \mathbf{k}$, where each row of $\mathbf{N}$ corresponds to a normal vector at one of the points $\mathbf{p}_j$, while $\mathbf{k}$ contains the constant terms. In floating-point arithmetic, such a solution may not exist; therefore, we apply the Moore-Penrose pseudo-inverse and write

$$\hat{\mathbf{v}} = \mathbf{N}^{\dagger} \cdot \mathbf{k}.$$

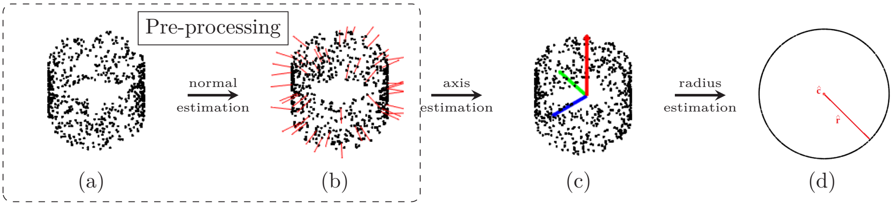


Figure IV.15: Initial estimates for a circular cylinder. The pre-processing step centers the point cloud and approximates the normal vectors at a set of points, see (b). Given two point $\mathbf{p}_{j_1}$ and $\mathbf{p}_{j_2}$, (c) shows the corresponding normal vectors $\hat{\mathbf{n}}_{j_1}$ and $\hat{\mathbf{n}}_{j_2}$ and, in red, their cross product $\hat{\mathbf{a}}_{j_1,j_2}$. Finally, (d) shows the estimate $\hat{r}$ of the radius.

For each pair of points $\mathbf{p}_{j_1}$ and $\mathbf{p}_{j_2}$, where $j_1, j_2 = 1, \ldots, k$ and $j_1 \neq j_2$, we consider the corresponding normal vectors $\hat{\mathbf{n}}_{j_1}$, $\hat{\mathbf{n}}_{j_2}$ and the vectors $\hat{\mathbf{t}}_{j_1} := \mathbf{p}_{j_1} - \hat{\mathbf{v}}$, $\hat{\mathbf{t}}_{j_2} := \mathbf{p}_{j_2} - \hat{\mathbf{v}}$. We compute the cross products $\hat{\mathbf{u}}_{j_1} = \hat{\mathbf{n}}_{j_1} \times \hat{\mathbf{t}}_{j_1}$ and $\hat{\mathbf{u}}_{j_2} = \hat{\mathbf{n}}_{j_2} \times \hat{\mathbf{t}}_{j_2}$. By taking the cross product between $\hat{\mathbf{u}}_{j_1}$ and $\hat{\mathbf{u}}_{j_2}$ we obtain an estimate $\hat{\mathbf{a}}_{j_1,j_2}$ of the rotational axis of the cone, up to a translation by the cone vertex. A simplified graphical illustration, where a triplet of vectors $\hat{\mathbf{n}}_j$ (in blue), $\hat{\mathbf{t}}_j$ (in green) and $\hat{\mathbf{a}}_{j_1,j_2}$ (in red) are moved to rotational axis, is shown in Figure IV.16(d). By iterating over all possible combinations, one can obtain multiple estimates of the rotational axis; we average over all these estimates and return the resulting vector, denoted $\hat{\mathbf{a}}$.To put the point cloud $\mathcal{P}$ in its canonical form, we apply a rototranslation so that the vertex $\hat{\mathbf{v}}$ is moved to the origin of the coordinate axes and $\hat{\mathbf{a}}$ coincides with the z-axis. The estimate $\hat{\alpha}$ is obtained by computing the angle between the $\mathbf{e}_3$ and the vector $[\frac{z_{max}}{r_{max}}\,, 0\,, 1]$, where z_{max} and r_{max} are, respectively, the maximum value of the z-coordinates and the maximum distance from the origin of the projection on the xy-plane of $\mathcal{P}$.

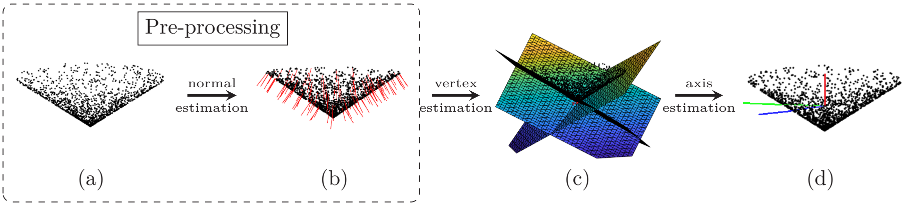


Figure IV.16: Initial estimates for a circular cone. The pre-processing step centers the point cloud and approximates the normal vectors at a set of points, see (b). The intersection of the tangent planes estimates the coordinates of the vertex $\hat{\mathbf{v}}$. Given two points $\mathbf{p}_{j_1}$ and $\mathbf{p}_{j_2}$, in (d) the corresponding blue and green vectors $\hat{\mathbf{u}}_{j_1}$ and $\hat{\mathbf{u}}_{j_2}$ and the resulting cross product $\hat{\mathbf{a}}_{j_1,j_2}$ in red.

Torus In line with the increase in the number of unknown parameters, this primitive requires a more complex handling, as summarized in the following four steps:

- *Upper (or lower) circle recognition.* We search for the best fitting plane to the entire point cloud $\mathcal{P}$ which – unless pathological cases (e.g., very small segments) – intersects the torus in (possibly perturbed arcs of) a circle, as shown Figure IV.17(c, left). This circle can be recognized by the standard HT for circles; the parameters found can be used to generate a new dense set of points, which we will denote by $\mathcal{Q}$; an example is shown in Figure IV.17(c, right).

- *Recognition of small circles.* We select a number of points corresponding to the lowest entries in $\mathbf{MFE}$, and denote them by $\mathbf{p}_1, \ldots, \mathbf{p}_k$. For each of such points $\mathbf{p}_j$, we find its nearest neighbour $\mathbf{q}_j \in \mathcal{Q}$; we then define the vector $\mathbf{v}_j$ as the cross product between the estimated normal vector $\hat{\mathbf{n}}_j$ at $\mathbf{p}_j$ and the vector $\mathbf{t}_j := \mathbf{p}_j - \mathbf{q}_j$. An example is shown in Figure IV.17(d, left image): the green, blue and red vectors represent, respectively, $\mathbf{t}_j$, $\hat{\mathbf{n}}_j$ and $\mathbf{v}_j$. The just-computed vector $\mathbf{v}_j$ identifies a plane – see Figure IV.17(d, right image) – that intersects $\mathcal{P}$ in a set of points outlining two circles, up to some data perturbation. We apply the standard HT to recognise such circles and, more importantly, their radii and centers. For each recognised circle, we compute its Mean Fitting Error and store its center in $\mathcal{C}$ if the MFE is below some given threshold. By averaging the circle radii, we can get an estimate $\widehat{r}_{\min}$ of $r_{\min}$.

- *Towards axis estimation.* We use HT to find the best fitting plane to $\mathcal{C}$. The normal vector to this plane is an estimate of the rotational axis of the torus, up to a translation (see Figure IV.17(e)).

- *Recognition of the big circle for center estimation.* Finally, we recognize the circle outlined by the points in $\mathcal{C}$, see Figure IV.17(f). The center of the torus is approximated by the circle center, which can be also used to fix the rotational axis. The radius of the circle gives us an estimate $\widehat{r}_{\max}$ of $r_{\max}$.

IV.A.3 Examples on synthetic data

Figure IV.18 illustrates the stability of our procedure for parameter estimation against an increasing amount of noise. As we move row by row from top to bottom, we show: a one-eighth portion of a full sphere of radius $r = 1.5$ and center $\mathbf{c} = (0, 0, 0)$; a small portion of a cylinder of radius $r = 1.5$ and aligned to the $z-$axis; a one-eighth portion of a cone of angle $\alpha = 56.24°$, vertex $\mathbf{v} = (0, 0, 0)$ and rotational axis coincident with the $z-$axis; and a one-eighth portion of a full torus aligned with the z-axis, with $r_{\min} = 1$, $r_{\max} = 2$, $\mathbf{c} = (0, 0, 0)$. As we move from left to right, the progressively increasing in noise corresponds to a lower precision of the initial estimates: the cylindrical segment in the second row suggests that the initial estimation can fail when applied to particularly small segments suffering from strong noise levels.

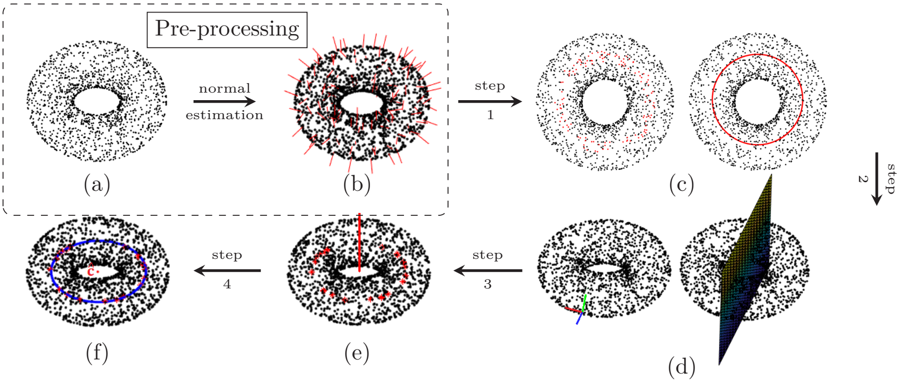


Figure IV.17: Initial estimates for a torus. The pre-processing step centers the point cloud and approximates the normal vectors at a set of points, see (b). In (c) the upper/lower circle recognition, while (d) shows the a plane that identify small circles. Given the centers of small circles, in (e) the estimation of the axis $\hat{\mathbf{a}}$ and in (f) the estimation of center $\hat{\mathbf{c}}$ of the torus.

Authors' addresses

Chiara Romanengo Istituto di Matematica Applicata e Tecnologie Informatiche “E. Magenes”, Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, chiara.romanengo@ge.imati.cnr.it

Andrea Raffo Istituto di Matematica Applicata e Tecnologie Informatiche “E. Magenes”, Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, andrea.raffo@ge.imati.cnr.it

Bianca Falcidieno Istituto di Matematica Applicata e Tecnologie Informatiche “E. Magenes”, Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, bianca.falcidieno@ge.imati.cnr.it

Silvia Biasotti Istituto di Matematica Applicata e Tecnologie Informatiche “E. Magenes”, Consiglio Nazionale delle Ricerche, Genova, Italy, silvia.biasotti@ge.imati.cnr.it

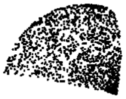
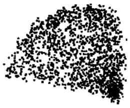
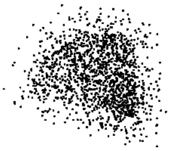
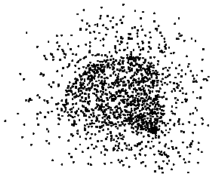
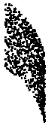
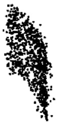
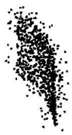
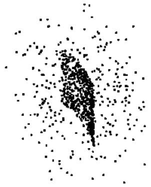
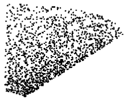
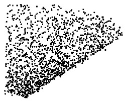
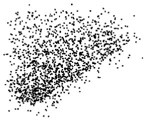
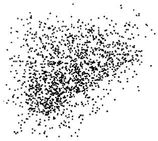
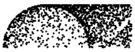
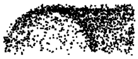
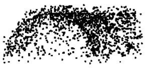
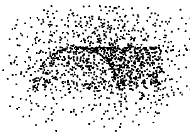
Segments with increasing noise			
Noise-free	$\mathcal{U}(-0.1, 0.1)$	$\mathcal{U}(-0.5, 0.5)$	$\mathcal{U}(-1.0, 1.0)$
 $\lvert r - \hat{r} \rvert = 0.00$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.01$	 $\lvert r - \hat{r} \rvert = 0.02$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.02$	 $\lvert r - \hat{r} \rvert = 0.05$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.02$	 $\lvert r - \hat{r} \rvert = 0.03$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.06$
 $\lvert r - \hat{r} \rvert = 0.00$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.00$	 $\lvert r - \hat{r} \rvert = 0.00$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.00$	 $\lvert r - \hat{r} \rvert = 0.04$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.00$	 $\lvert r - \hat{r} \rvert = 0.47$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.44$
 $\lvert \alpha - \hat{\alpha} \rvert = 0.00$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.00$ $\lvert\lvert \mathbf{v} - \hat{\mathbf{v}} \rvert\rvert_2 = 0.02$	 $\lvert \alpha - \hat{\alpha} \rvert = 0.00$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.00$ $\lvert\lvert \mathbf{v} - \hat{\mathbf{v}} \rvert\rvert_2 = 0.02$	 $\lvert \alpha - \hat{r} \rvert = 0.01$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.19$ $\lvert\lvert \mathbf{v} - \hat{\mathbf{v}} \rvert\rvert_2 = 0.03$	 $\lvert \alpha - \hat{\alpha} \rvert = 0.03$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.37$ $\lvert\lvert \mathbf{v} - \hat{\mathbf{v}} \rvert\rvert_2 = 0.03$
 $\lvert r_{\min} - \hat{r}_{\min} \rvert = 0.00$ $\lvert r_{\max} - \hat{r}_{\max} \rvert = 0.02$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.02$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.00$	 $\lvert r_{\min} - \hat{r}_{\min} \rvert = 0.01$ $\lvert r_{\max} - \hat{r}_{\max} \rvert = 0.02$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.03$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.01$	 $\lvert r_{\min} - \hat{r}_{\min} \rvert = 0.02$ $\lvert r_{\max} - \hat{r}_{\max} \rvert = 0.04$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.04$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.02$	 $\lvert r_{\min} - \hat{r}_{\min} \rvert = 0.09$ $\lvert r_{\max} - \hat{r}_{\max} \rvert = 1.18$ $\lvert\lvert \mathbf{c} - \hat{\mathbf{c}} \rvert\rvert_2 = 0.07$ $\lvert\lvert \mathbf{a} - \hat{\mathbf{a}} \rvert\rvert_2 = 0.07$

Figure IV.18: Initial estimates for various segments with increasing noise.

Paper V

Fit4CAD: A point cloud benchmark for fitting simple geometric primitives in CAD objects

Chiara Romanengo, Andrea Raffo, Yifan Qie, Nabil Anwer, Bianca Falcidieno

Published in *Computers & Graphics*, February 2022, volume 102, pp. 133–143.
DOI: 10.1016/j.cag.2021.09.013.

V

Abstract

We propose Fit4CAD, a benchmark for the evaluation and comparison of methods for fitting simple geometric primitives in point clouds representing CAD objects. This benchmark is meant to help both method developers and those who want to identify the best performing tools. The Fit4CAD dataset is composed by 225 high quality point clouds, each of which has been obtained by sampling a CAD object. The way these elements were created by using existing platforms and datasets makes the benchmark easily expandable. The dataset is already split into a training set and a test set. To assess performance and accuracy of the different primitive fitting methods, various measures are defined. To demonstrate the effective use of Fit4CAD, we have tested it on two methods belonging to two different categories of approaches to the primitive fitting problem: a clustering method based on a primitive growing framework and a parametric method based on the Hough transform.

Keywords: benchmarking, geometric primitive fitting, CAD objects, quality measures.

Contents

The paper has been awarded the Replicability Stamp, see http://www.replicabilitystamp.org.

V.1 Introduction

3D CAD models are among the most common medium to convey dimensional and geometric information on designed objects or components. However, often the CAD model of an object is not available, it does not even exist, or no longer corresponds to the real geometry of the manufactured object itself. One strategy for retrieving a digital model of an object when not accessible is to acquire 3D data directly on the object and use it to create a digital representation. The reconstruction of digital models starting from the measured data is a process, commonly called Reverse Engineering (RE), aiming at reconstructing 3D mathematical surfaces and geometric features that represent the geometry of real parts. Many methods have been proposed to solve this problem; as a reference we cite a recent survey that groups a large part of the approaches presented so far [15].

Given the large number of methods proposed, it becomes important to be able to evaluate their performance by creating standard datasets with a ground truth and a “quality label", thus paving the road for a fair evaluation of the existing technologies and the identification of open research directions not only in reverse engineering but also in shape retrieval, understanding, compression, etc., taking inspiration from other approaches proposed for generic classes of objects, (e.g. [10, 21, 25]).

Here we propose Fit4CAD, a benchmark of point clouds representing CAD objects aimed at evaluating methods for detecting simple (polynomial) geometric primitives (i.e., plane, cylinder, cone, sphere, and torus) in 3D point clouds; by polynomial primitive, we here mean a surface that has an algebraic implicit representation, i.e., it can be defined as the zero set of a polynomial. The dataset consists of 225 high quality point clouds, each of which has been obtained by sampling a CAD object. Each point cloud is equipped of a ground-truth segmentation and, for each primitive, we provide both implicit and parametric forms. The way these elements were created by using existing platforms and datasets makes the benchmark easily expandable. Fit4CAD is designed to be used also by machine learning methods: in fact, the dataset comes in the form of a training set and a test set.

We provide a number of performance measures able to evaluate both the quality of the fitting segments, in term of points correctly recognized as belonging to a primitive, and the quality of the primitive approximation, evaluating the distance between the primitive detected and the ideal one.

Fit4CAD satisfies a certain number of necessary requirements, such as the relevance and representativeness of the elements in the CAD context, the richness and the completeness of the information associated with each primitive, thus enabling fair comparisons for a wide range of geometric primitives recognition algorithms.

The proposed benchmark has been exploited to evaluate and compare two methods of geometric primitive fitting, belonging to two different categories of approaches: a clustering method based on a primitive growing framework and a parametric method based on the Hough transform. By providing an explicit representation of the equations of the primitives, for the second method we are also able to evaluate measures related to the accuracy of the primitives found.

The rest of the paper is organized as follows. Section V.2 examines previous work related to our topic. Section V.3 describes the characteristics of the benchmark: dataset, ground truth, and the performance and accuracy measures chosen to evaluate the identification of primitives. Section V.4 describes the tests carried out on two methods of recognition and fitting of geometric primitives. Some concluding remarks end the paper.

V.2 Prior work

Benchmarking involves sharing of resources, metrics, data and so on, so that the common goals of knowledge creation and furthering the state of the art can be achieved. The creation of standard datasets reduces the amount of work necessary for single researchers to assess the quality of their techniques and compare them with other research groups. The steadily rising participation to contests and open challenges shows the interest and the need for benchmarks (e.g., TreCVID [24]), competitive contests (e.g., on Kaggle.com [5] or the 3D Shape Retrieval Contest (SHREC) [27]) and, more in general, for code sharing (e.g., Graphics Replicability Stamp Initiative[1]). So far, benchmarks for 3D object segmentation [6, 18] have mainly considered generic classes of objects and, therefore, the methods were evaluated for their general-purpose segmentation rather than on CAD objects and their capability of recognizing geometric primivites.

Among the datasets containing general 3D shapes (e.g., toys, mechanisms, jewelry) in the form of triangle meshes it is worth mentioning [28, 29, 30], even if these methods were specifically designed for different goals, e.g. 3D printing, computer vision applications and computer graphics applications, respectively.

The most relevant dataset for our work is the ABC dataset [16]; here, the authors present a massive dataset (over one million models), specifically developed to train data-driven algorithms for geometric deep learning. Models are defined by parametric surfaces, possibly accompanied with the information related to the decomposition into patches, sharp feature annotations, and analytic differential properties. The models were created by using the interface available on the online infrastructure Onshape[2]. All models are stored as triangle meshes, while the associated files (annotations, features, etc.) are not available for all models: more precisely, the file may lack the true list of primitives, or their parametric/implicit representations; this is not a big issue for that specific dataset, as primitive extraction is not their purpose. Being specifically designed for data-driven methods, the models are stored with different resolutions (i.e., different samplings of the same parametric model) or slight variations. Moreover, models in [16] do not present any kind of data perturbation. Lastly, the ABC dataset is not designed for point cloud segmentation, and does not present any specific quality measures for comparing methods.

[1] http://www.replicabilitystamp.org

[2] https://www.onshape.com/en/

V.3 The benchmark

We here introduce our benchmark, geared towards the following desirable properties:

- *Dataset richness and representativeness.* The first and foremost requirement for a thorough evaluation is the availability of a data set characterized by a good sampling of varied shapes: each family of simple geometric primitives (i.e., plane, cylinder, cone, sphere, and torus) should appear in a sufficient number of point clouds. In addition, we consider different point cloud densities as well as data integrity (with/without missing data). Each CAD object is used to generate one and only one point cloud.

- *Ease of expansion.* An ideal benchmark should be able to develop over time, in order to test new paradigms and face new challenges; this requires the capability to generate new data in an easy and efficient way. To satisfy this basic requirement, Section V.3.1 outlines a general pipeline that can be used for data generation.

- *Availability of both implicit and parametric representations.* Modern CAD systems are based on two complementary representations for surfaces, according to the manipulation they are involved in: implicit and parametric representations. Parametrized surfaces are best suited for point generation, while implicit representations allow to check whether a query point lies or not on the surface in a more convenient way. Having both representations makes it possible to answer a wide range of questions (e.g., intersection problems).

- *Completeness of the documentation.* All models are equipped with all and the same information. For each of them, this includes: the files with primitive segments, implicit and parametric primitive representations; a preset split of the dataset into training set and test set. Further details are provided in Section V.3.2.

- *Variety of performance indicators and accuracy measures.* To evaluate and compare methods, it is of vital importance to select measures that highlight strengths and weaknesses. In our case, the problem is twofold: on the one hand, we want to quantify the capability to produce precise segmentations into simple geometric primitives (by performance indicators); on the other hand, we also aim at measuring the fitting accuracy when it comes to implicit and parametric representation of the same shapes (by accuracy measures). Section V.3.3 describes the measures selected to evaluate the detection of simple primitives in CAD objects point clouds.

V.3.1 Dataset

At present, the dataset contains 225 individual high quality point clouds, each of which has been obtained by sampling a CAD object. The dataset is already split into two subsets: a training set, counting 190 point clouds, and a test set, containing the remaining 35 point clouds. Figure V.1 shows the distribution of surface types for both training and test sets.

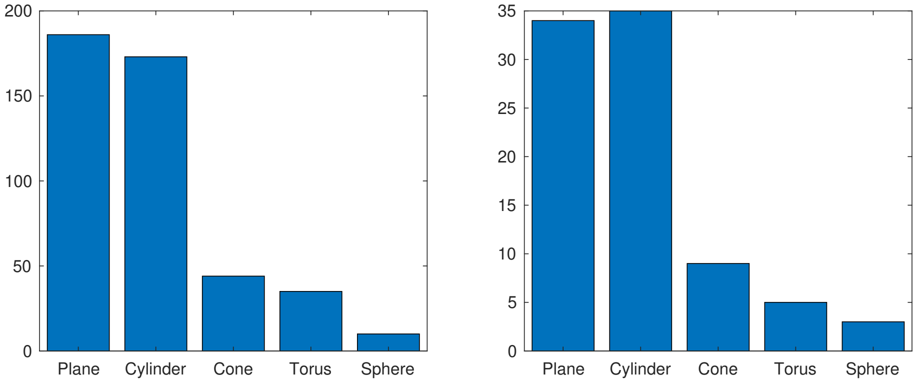


Figure V.1: Surface type distribution. The two bar charts show the distributions for the training set (left) and the test set (right).

The dataset generation process, in the most general form, has been carried out by the following three steps:

1. *Model creation.* We created part of the models, by using the publicly available interface hosted by Onshape, while the remaining part was collected from the ABC dataset [16], which was derived, in turn, from the Onshape public collection. Models gathered from the ABC dataset have been filtered by manually correcting the parts presenting minimum flaws and rejecting low quality models, in order to avoid rare yet bothersome imperfections, such as overlapping or repeating patches. Some examples of CAD objects from Onshape are displayed in Figure V.2.

2. *Parametric and implicit representations.* The generation of B-rep models was crucial to extract the parametric representation behind each geometric primitive; in our case, the parametric representations for each patch have been obtained by processing the STEP files produced by Onshape in GMSH [12]; nevertheless, we emphasize that other software could be considered too (e.g., [19]). Several methods to compute the implicit representation from a parametric form are nowadays available. We here consider the numerical approach known as *approximate implicitization*, introduced in [8] and further delevoped in [1]. One of the advantages of this approach is that it provides exact implicit representations when the exact total degree is selected; we remind that a bivariate polynomial has total degree n if all monomials x^iy^j are such $i+j \leq n$, and there exists at least one monomial x^iy^j such that $i+j=n$.

3. *Point cloud extraction.* CAD objects are sampled at different densities, and optionally manually postprocessed by using CloudCompare[3] to simulate missing data. To give an example, Figure V.3(a) shows a model from Onshape, which is then sampled and postprocessed in V.3(b-c).

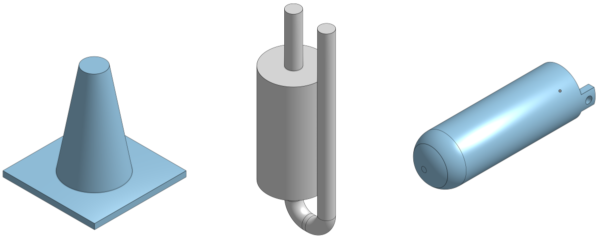


Figure V.2: Example of models obtained using Onshape.

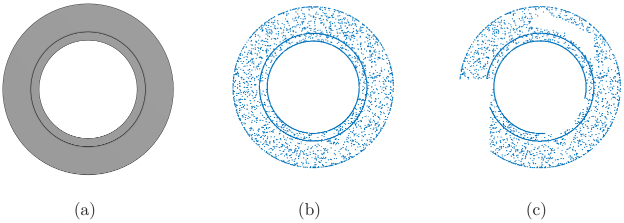


Figure V.3: Example of point cloud creation. The initial object in (a) is sampled at a chosen density (b) and then perturbed by simulating missing data (c).

V.3.2 Ground truth

Each model in the ground truth comes in the form of four TXT files. We here provide a description of each file content for the i-th point cloud.

PCi lists the three-dimensional points forming the point cloud to be segmented.

PCi_primitives contains the list of true primitives. For each primitive, a list of indices is provided; each index corresponds to a point in "*PCi*", with respect to the ordering there introduced. For example,

```
        Primitive6:=[4 9 184 185 186 187 188 189 190 191 192]
```

[3] CloudCompare (version 2.10.2), http://www.cloudcompare.org/

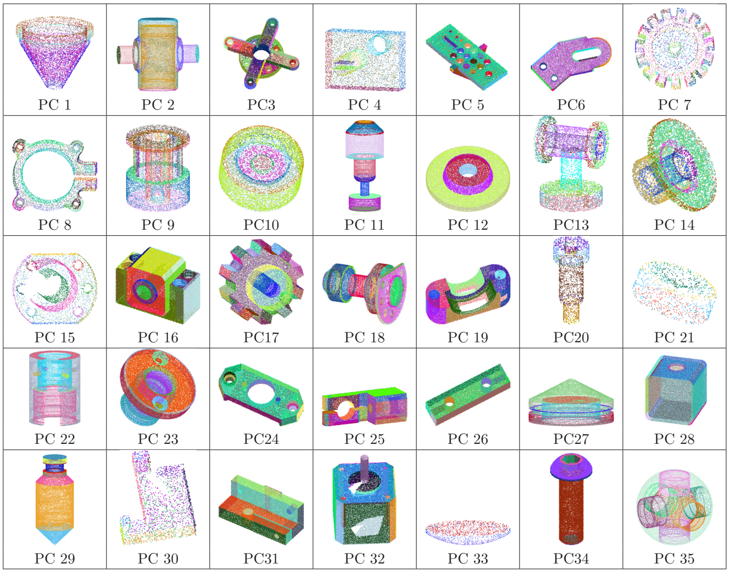


Figure V.4: The 35 point clouds used as a test set. Different colors represent different primitives, as stored in the CAD models, i.e., our ground truth.

means that the sixth primitive contains points number 4, 9, 184, 185, 186, 187, 188, 189, 190, 191 and 192 (where the ordering is the one in the corresponding “*PCi*").

PCi_parametric provides, for each primitive in “*PCi_primitives*" corresponding to a plane, a cylinder, a cone, a sphere or a torus, its parametric representation. To give an example,

```
Primitive6:=[primitive type, v]
```

where **v** is the vector that contains the parameters of the parametric representation (see V.A.1 for further details on the considered ordering).

PCi_implicit provides, for each primitive in “*PCi_primitives*" corresponding to a plane, a cylinder, a cone, a sphere or a torus, its implicit representation. For example,

```
Primitive6:=[primitive type, w]
```

where **w** is the vector that contains the coefficients of the implicit representation (see V.A.2 for further details on the considered ordering).

The points that do not correspond to any of the simple primitives mentioned above (i.e., plane, cylinder, cone, sphere or torus) are classified as unsegmented and not explicitly reported in files *PCi_primitives*, *PCi_parametric* and

PCi_implicit; in the original model, these points usually originate from B-spline surfaces. We intentionally decided to insert some models with non-simple geometric primitives to check whether a candidate method can avoid misclassification.

V.3.3 Quality indicators

To evaluate the detection of simple primitives in CAD objects, we have proposed quality measures selected from [7, 17] with particular care on what concerns their performance and approximation accuracy.

V.3.3.1 Performance measures of the point classification

Any primitive in a model is identified by the list of points belonging to it or, equivalently, by the list of points that do not belong to it. The problem of primitive detection can therefore be easily written in terms of binary classification tasks, one per primitive in the ground truth.

Let $\mathscr{P}_B$ be a a set of points in the benchmark point cloud corresponding to a specific primitive, and let $\mathscr{P}_S$ be the primitive in the segmentation to assess that most overlap with $\mathscr{P}_B$. We can define the following quantities:

- *True positives*, TP: the number of points shared by $\mathscr{P}_B$ and $\mathscr{P}_S$.
- *False positives*, FP: the number of points in $\mathscr{P}_S$ that do not belong to $\mathscr{P}_B$.
- *False negatives*, FN: the number of points in $\mathscr{P}_B$ that do not belong to $\mathscr{P}_S$.
- *True negatives*, TN: the number of points that do not belong to either $\mathscr{P}_B$ nor $\mathscr{P}_S$.

Based on these four quantities, we consider the following measures:

- *Sensitivity*, also called *true positive rate*, measures the proportion of positives which are correctly identified, i.e.,

$$\mathrm{TPR} := \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

  Specificity, or *true negative rate*, measures the proportion of true negatives that are correctly identified as such, i.e.,

$$\mathrm{TNR} := \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}.$$

- *Positive predictive value* is defined as the proportion of predicted positives which are actual positives, i.e.,

$$\mathrm{PPV} := \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}.$$

Similarly, *negative predictive value* is given by

$$\mathrm{NPV} := \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}.$$

- *Accuracy* is the ratio of correct predictions to total predictions made, i.e.,

$$\mathrm{ACC} := \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}.$$

- *Sørensen-Dice index.* It is given by

$$\mathrm{DSC} := \frac{2|\mathscr{P}_B \cap \mathscr{P}_S|}{|\mathscr{P}_B| + |\mathscr{P}_S|}.$$

  In case of binary classification, it is shown to be equivalent to

$$\mathrm{DSC} := \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}},$$

  which is often referred to as F_1 *score*.

For more details, we refer the reader to [17].

V.3.3.2 Approximation accuracy

To measure the recognition accuracy of a specific primitive, we use the parametric and the implicit representations provided in “*PCi_implicit*" and “*PCi_parametric*". Exploiting the notation provided before, let us consider a primitive $\mathscr{P}_S$ to be evaluated, and let $\mathcal{S}$ be the surface described by the corresponding parametric representation. When it comes to the parametric representation, we use the following two measures to evaluate the approximation accuracy of primitive $\mathscr{P}_S$:

- *Mean Fitting Error* (MFE):

$$\mathrm{MFE}(\mathscr{P}_S, \mathcal{S}) := \frac{1}{|\mathscr{P}_S|} \sum_{\mathbf{x} \in \mathscr{P}_S} d(\mathbf{x}, \mathcal{S})/l, \tag{V.1}$$

  where d is the Euclidean distance, and l is the diagonal of the minimum bounding box containing $\mathscr{P}_S$.

- *Directed Hausdorff distance*:

$$d_{\mathrm{dHaus}}(\mathscr{P}_S, \mathcal{S}) = \max_{\mathbf{x} \in \mathscr{P}_S} \min_{\mathbf{y} \in \mathcal{S}} d(\mathbf{x}, \mathbf{y}),$$

  with d the Euclidean distance. To make the measure independent from the primitive size, we normalize it with respect to the diagonal l of the minimum bounding box containing $\mathscr{P}_S$.

The fitting accuracy for the implicit representation is evaluated by the following measure:

- *Coefficient distance*:
$$d_1(\mathbf{v}, \mathbf{v}') = \|\mathbf{v} - \mathbf{v}'\|_1$$
where $\mathbf{v}$ and $\mathbf{v}'$ are the coefficient vectors for the implicit representations of the primitives $\mathscr{P}_S$ and $\mathscr{P}_B$, respectively, and where $\| \cdot \|_1$ is the well-known ℓ^1 norm. In order to make this measure consistent, we assume the coefficient vectors to be normalized, and the first nonzero entry to be positive (where the ordering is the one provided in V.A.2).

We refer the reader to [7] for further details.

V.4 Test of the benchmark on two methods

The proposed benchmark has been used to evaluate and compare two methods dealing with primitive fitting. As guiding examples of how the benchmark works, we have selected two methods that are both available and representative of two classes of methods according to the taxonomy defined in [15]: a clustering method based on a primitive growing framework (Section V.4.1) and a parametric method based on the Hough transform [13] (Section V.4.2). Both methods can be evaluated according to the measures described in Section V.3.3.1, as they explicitly provide the list of the points that form any primitive; on the other hand, the approximation accuracy can be assessed only for methods that can provide parametric or implicit representations, in our examples the Hough-based fitting.

V.4.1 PG: a discrete curvature-based method for point cloud segmentation

As a first approach, we present a curvature-based method based on a primitive growing framework, on the basis of the method proposed in [22] for triangle meshes; for the sake of brevity, we will often use the acronym PG as a shorthand for this method, where PG stands for “Primitive Growing".

The method consists of two main steps: an initial region partitioning process based on high curvature detection and, then, a region refinement process based on slippage analysis, as summarized in Figure V.5. These two steps run as follows:

- *Initial region partition.* Points that identify sharp edges are characterized via a point attribute called *surface variation*, as introduced in [20]. Given a point $\mathbf{p}$ and n neighbouring points, its surface variation is defined as
$$\sigma_n(\mathbf{p}) := \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3},$$

where $\lambda_1 \leq \lambda_2 \leq \lambda_3$ are the eigenvalues of the covariance matrix for the sample point $\mathbf{p}$ and its n neighbouring points; note that λ_l measures the variability of the neighborhood of n points along the direction of the corresponding eigenvector. In our experiments, n is set to 15 as it yields good results when considering the training set. Points on sharp edges are characterized by a high surface variation. These points are here selected by analysing a histogram of the surface variation values, by means of a threshold η: all points having surface variation above η are labelled as sharpe edge points; for example, setting $\eta = 0.8$ means that the points whose surface variation is higher than 80% (top 20%) are considered to belong to a sharp edge. In our implementation, the threshold η is user-defined and taken in the interval $[0.65, 0.95]$. Once sharp edges have been identified, a region growing approach is applied to compute a first coarse pre-segmentation, along the lines of what detailed in [20]: starting from a random seed point, its nearest neighbors are progressively located; those points which does not belong to sharp edges will be labelled and used as new seed points, until all neighboring points are labelled.

- *Region refinement.* According to the ISO GPS invariance class [14], "ideal" features can be categorized into seven invariance classes: planar, cylindrical, helical, spherical, revolute, prismatic, and complex. The seven invariance classes are here captured by local slippage analysis [11]. This step aims at decomposing any coarse segment S from the previous step into simpler geometric parts. Given a point set P of n points from S, the slippable motions of P are found as the motion vector $[\mathbf{r}, \mathbf{t}]$ that, when applied to P, minimizes the motion along the normal direction at each point

$$\min_{[\mathbf{r},\mathbf{t}]} \sum_{i=1}^{n} ((\mathbf{r} \times \mathbf{x}_i + \mathbf{t}) \cdot \mathbf{n}_i)^2, \tag{V.2}$$

where: $\mathbf{r} = (r_x, r_y, r_z)$ is a rotation vector around x, y, and z; $\mathbf{t} = (t_x, t_y, t_z)$ is a translational vector; $\mathbf{p}_i \in P$ are the n samples, and $\mathbf{n}_i$ are their respective normals. Equation V.2 is a least-square problem which can be reduced to the linear system based on the covariance matrix of the second partial derivatives of the function in V.2 with respect to the rotation and translation parameters, see [11] for further details. Slippage analysis permits the detection of 3-, 2-, 1- and 0-slippable motions, see Table V.1. According to the primitives defined in the benchmark, segments identified as prismatic, revolute and complex could be undersegmented; to address this problem, the RANSAC method introduced in [23] is applied; an example of prismatic segment requiring further processing is shown in Figure V.6. Finally, points on sharp edges are assigned to the closest primitives.

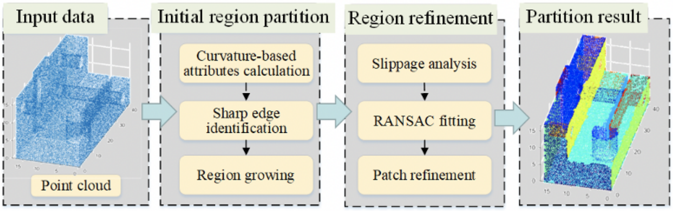


Figure V.5: The framework of the curvature-based surface partitioning method.

ISO GPS Invariance	Slippage	geometric primitives
planar	3	plane
spherical	3	sphere
cylindrical	2	cylinder
helical	2	-
prismatic	1	undersegmented
revolute	1	cone/torus
complex	0	undersegmented

Table V.1: Relation between ISO GPS invariance class [14] and the simple geometric primitives in this benchmark. Note that, in this terminology, prismatic/complex could include include planar, spherical or cylindrical primitives.

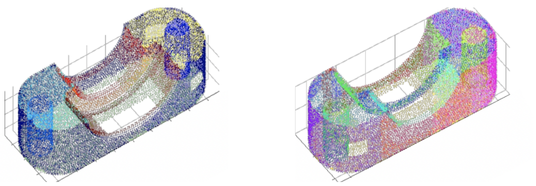


Figure V.6: Example of undersegmented invariance class. On the left, the external blue primitive is classified as prismatic. On the right, the same primitive is split into 2 planes and 2 half cylinders.

V.4.1.1 Computational complexity

The characterization of the sharp edges according to surface variation [20] is done by considering the k-nearest neighbour points with $\mathcal{O}(n \log n)$ operations, where n represents the number of points. The growing algorithm for grouping points inside boundaries costs $\mathcal{O}(n)$ operation while the classification of each point set via slippage analysis is $\mathcal{O}(m)$, where m is the number of points in one surface portion [11]. The further RANSAC based segmentation in case of point sets with low slippage values (such as revolute, prismatic and complex primitives) is $\mathcal{O}(m)$ [23], where $m < n$ in the most common scenario.

V.4.2 HT: Simple primitive fitting based on Hough transform

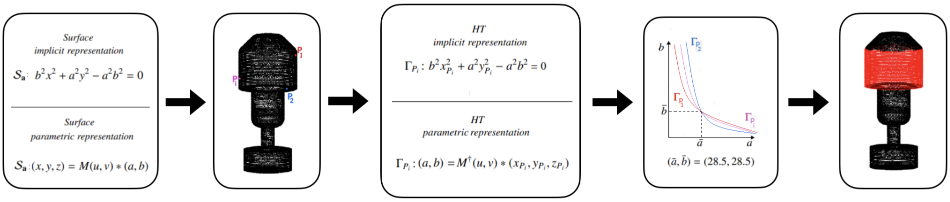

Surface
implicit representation
$\mathcal{S}_{\mathbf{a}}: b^2x^2 + a^2y^2 - a^2b^2 = 0$
Surface
parametric representation
$\mathcal{S}_{\mathbf{a}}: (x,y,z) = M(u,v) * (a,b)$
P_1
P_i
P_2
HT
implicit representation
$\Gamma_{P_i}: b^2x_{P_i}^2 + a^2y_{P_i}^2 - a^2b^2 = 0$
HT
parametric representation
$\Gamma_{P_i}: (a,b) = M^{\dagger}(u,v) * (x_{P_i}, y_{P_i}, z_{P_i})$
b
Γ_{P_2}
Γ_{P_1}
Γ_{P_i}
$\bar{b}$
$\bar{a}$
a
$(\bar{a},\bar{b}) = (28.5, 28.5)$

Figure V.7: The HT-based paradigm: a visual illustration of how a cylinder representation S_a is converted by the Hough transform into $|P_i|$ hypersurfaces Γ_{P_i}; then the intersection of the hypersurfaces Γ_{P_i} identifies the parameters $(\bar{a}, \bar{b})$ that correspond to the red cylinder in the right.

In this section, we consider a method to segment and to fit a point cloud PC with surface primitives using the Hough Transform (HT) technique. The general HT-framework deals with the problem of finding a surface $\mathcal{S}_{\bar{\mathbf{a}}}$ – within a family $\mathcal{F} = \{\mathcal{S}_{\mathbf{a}}\}$ of surfaces dependent on a set of parameters $\mathbf{a} = (a_1, ..., a_n)$ – that best approximates a particular shape. The common strategy to identify the solution (or a solution) consists in a procedure whereby each point in PC votes a n-uple $\mathbf{a}$ in the parameter space; the most voted n-uple $\bar{\mathbf{a}}$ corresponds to the most representative surface $\mathcal{S}_{\bar{\mathbf{a}}}$ for a dense subset of PC. Figure V.7 illustrates how the Hough transform converts the problem of fitting points on a primitive into the problem of fitting the parameters of a family of primitives into points.

This method is based on the theory related to the extension of the Hough transform to general algebraic objects [3]. This theory is very broad and can be used for many types of primitives, for instance in [2] is used for fitting point sets with ellipsoids and can deal with non-simple primitives, such as helical surfaces. The families of primitives included in this benchmark are planes, cylinders, spheres, cones and tori. Once a family of primitives $\mathcal{F}$ is selected, the main steps can be summarized as follows:

- *Inizialization and estimation of the accumulator function.* Once the family $\mathcal{F}$ is chosen, a region T of the parameter space is selected exploiting the knowledge of the geometric characteristics of $\mathcal{F}$ (e.g., bounding box). Then, it is discretized into cells, which are uniquely identified by the coordinates of their centre. This space is associated with an accumulator function $\mathcal{H}$, discretized as a matrix. Its entries are in a one-to-one correspondence with the cells of T. An entry of $\mathcal{H}$ is increased by 1 each time the HT of a point P, Γ_P, intersects the corresponding cell.

- *Selection of potential fitting primitives.* In the case the input point cloud is composed of different primitives, the peaks of $\mathcal{H}$ identify the potential primitives $\mathcal{S}_{\bar{\mathbf{a}}_{\mathbf{i}}}$ that might fit different parts $\mathcal{X}_i \subseteq PC$. Then, the cells corresponding to the peak values of the accumulator function $\mathcal{H}$ are identified by studying its *topological persistence* (see [9]). In our

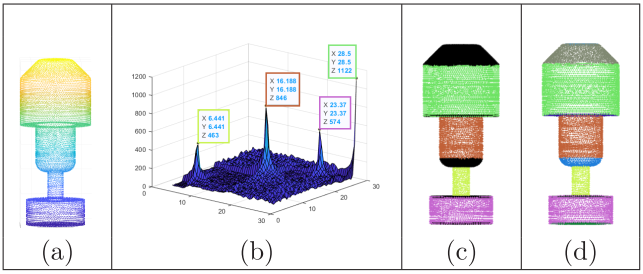


Figure V.8: HT framework. In (a), an example of input point cloud; in (b), the accumulator function associated with the search for cylinders and the peaks found by the method for persistent maxima; in (c), the four cylinders corresponding to the four peaks (black stands for unclassified points). The final outcome, after searching for all simple geometric primitives, is shown in (d).

> implementation, the peaks that correspond to primitives are automatically recognised by keeping the local maxima with a persistence higher than 10% of the maximum value of $\mathcal{H}$, using the algorithm for persistent maxima proposed in [4]. The coordinates of the cell centres of the maxima or the peaks of the accumulator function correspond to the parameters of potentially recognised surface primitives.

Since it can happen that more types of primitives fit the same dense subset $\mathcal{X}_i$ (or a part of it), the *Mean Fitting Error* (see Equation V.1) is used to evaluate the approximation accuracy of each primitive. Then, if $\mathcal{S}_{\bar{\mathbf{a}}_{\mathbf{i},\mathbf{1}}}$ and $\mathcal{S}_{\bar{\mathbf{a}}_{\mathbf{i},\mathbf{2}}}$ are two candidate primitives, the fitting errors $\mathrm{MFE}(\mathcal{X}_i, \mathcal{S}_{\bar{\mathbf{a}}_{\mathbf{i},\mathbf{1}}})$ and $\mathrm{MFE}(\mathcal{X}_i, \mathcal{S}_{\bar{\mathbf{a}}_{\mathbf{i},\mathbf{2}}})$ between each primitive and $\mathcal{X}_i$ are calculated; the primitive having lowest error is kept. The final result is the partitioning of the input point cloud PC into several subsets in such a way that points of the same segment are well approximated by the same primitive. Figure V.8 summarizes the HT framework. In particular, Figure V.8(b) shows an example of accumulator matrix referred to the recognition of the cylinders, while the four cylinders corresponding to the four peaks are highlighted on the original point cloud in Figure V.8(c). Finally, Figure V.8(d) exhibits the resulting segmentation.

V.4.3 Evaluation

We here analyse the performance of the methods outlined in Sections V.4.1 and V.4.2, with the purpose of showing how the benchmark works. Firstly, we compare the quality of the segments/primitives found against a ground-truth; the measures involved do not require an explicit representation of the primitive equation, and thus can be applied to both methods. Secondly, we consider the accuracy of the parametric/implicit representations: in our case, this comes down to the analysis of the method introduced in Section V.4.2.

Performance measures of the point classification

Figures V.11 and V.12 in V.B provide the segmentation results obtained by the primitive growing (PG) and Hough transform (HT) based approaches, colored as follows: given a model and an approach, for each primitive in the benchmark we find the most overlapping segmented primitive; misclassified points are colored in black, while correct matches follows the $1-1$ primitive-color correspondence from Figure V.4.

Table V.2 summarizes the performances of the two methods over all the test set models. Each row correspond to a model; for each model, the table provides information on the number of true and predicted primitives, as well as the accuracy measures introduced in Section V.3.3.1. For each metric, two columns are considered, respectively referring to the PG- and the HT-based approaches.

To ease the analysis, the metrics are studied via boxplots:

- Figure V.9 compares the two methods over the whole test set. A first observation of this analysis is that accuracy measures from the HT-approach have generally a lower variability. At a closer look, one can notice that the quartiles, as well as the minimum and maximum, always assume higher values when it comes to the HT-based method; in particular, the second quartile (i.e., the median) is always above 90%. DSC, TPR and PPV are the three accuracy measures that varies the most; this highlights that the two methods have lower performances in identifying the true positives, compared to true negatives. Both methods exhibit outliers in most of the boxplots.
- Robustness to missing data is analysed in Figure V.10. The HT-based method turns out to be hardly affected by such perturbation, as the inter-quartile range and the whiskers do not significantly vary; the only noteworthy variation is that of TPR, which points out a slightly decreased capability in correctly identifying positives. A more prominent variation can be noted for the PG-method.

Interestingly enough, both methods rarely suffer from oversegmentation, while it is more likely for them to undersegment. The most dramatic undersegmentation is that of point cloud 9 (i.e., PC 9 in Table V.2), where the PG-based and the HT-based methods only manage to detect 43 and 40 primitives, respectively, out of the 104 there expected; this highlights possible issues when the original model has thin or small primitives.

Approximation accuracy

Table V.3 reports the performance of the HT-based method evaluated according to the metrics reported in Section V.3.3.2. Each row corresponds to a segmented point cloud from Figure V.12; each column represents a different accuracy measure; for each point cloud, each measure has been obtained by averaging over all segments.

	# points	# true primitives	# predicted primitives		DSC		PPV		TPR		TNR		NPV		ACC	
			PG	HT	PG	HT	PG	HT	PG	HT	PG	HT	PG	HT	PG	HT
PC 1	7,500	8	12	8	0.370	0.987	0.693	0.989	0.332	0.985	0.968	0.998	0.696	0.995	0.706	0.995
PC 2	20,621	19	13	16	0.599	0.829	0.807	0.890	0.568	0.812	0.995	0.997	0.913	0.964	0.911	0.963
PC 3	9,723	35	36	35	0.843	0.896	0.778	0.828	0.943	0.986	0.994	0.996	0.998	1.000	0.992	0.996
PC 4	10,000	15	12	12	0.736	0.840	0.831	0.981	0.729	0.800	0.993	0.999	0.964	0.972	0.960	0.973
PC 5	20,000	37	32	34	0.480	0.839	0.665	0.872	0.544	0.901	0.990	0.999	0.925	0.978	0.918	0.978
PC 6	9,320	26	13	20	0.465	0.783	0.671	0.857	0.466	0.772	0.994	0.999	0.968	0.997	0.964	0.996
PC 7	5,000	68	41	69	0.465	0.923	0.642	0.871	0.429	0.994	0.993	0.999	0.978	1.000	0.972	0.999
PC 8	7,500	32	28	30	0.390	0.890	0.528	0.896	0.438	0.889	0.984	0.997	0.930	0.998	0.916	0.996
PC 9	17,000	104	43	40	0.403	0.568	0.607	0.848	0.335	0.456	0.998	0.999	0.993	0.996	0.991	0.995
PC 10	10,000	10	7	10	0.627	0.919	0.793	0.926	0.555	0.938	0.988	0.995	0.949	0.996	0.943	0.993
PC 11	13,201	13	9	10	0.705	0.805	0.874	0.959	0.677	0.770	0.993	0.998	0.937	0.973	0.934	0.973
PC 12	12,327	6	6	6	0.894	0.998	0.852	0.997	0.982	1.000	0.984	0.999	0.994	1.000	0.981	0.999
PC 13	10,000	21	17	16	0.582	0.800	0.792	0.991	0.576	0.761	0.993	1.000	0.951	0.983	0.946	0.984
PC 14	7,500	8	6	8	0.623	0.963	0.836	0.997	0.606	0.933	0.987	0.999	0.878	0.990	0.878	0.991
PC 15	5,000	15	15	14	0.533	0.941	0.615	0.972	0.566	0.933	0.981	0.999	0.951	0.993	0.937	0.992
PC 16	29,641	35	28	31	0.676	0.848	0.771	0.842	0.702	0.891	0.997	0.999	0.986	0.994	0.984	0.993
PC 17	21,137	45	45	45	0.916	0.935	0.859	0.889	0.983	0.996	0.997	0.998	1.000	1.000	0.997	0.998
PC 18	16,406	29	20	29	0.555	0.933	0.847	0.912	0.536	0.978	0.994	0.999	0.898	0.999	0.896	0.998
PC 19	16,740	16	14	11	0.751	0.747	0.869	0.967	0.747	0.681	0.990	0.998	0.971	0.960	0.963	0.960
PC 20	2,500	14	8	14	0.525	0.954	0.789	0.917	0.467	1.000	0.991	0.999	0.916	1.000	0.914	0.999
PC 21	1,000	5	3	5	0.677	0.985	0.891	0.983	0.588	0.988	0.977	0.997	0.834	0.998	0.845	0.996
PC 22	26,093	10	6	10	0.561	0.967	0.751	0.949	0.586	0.987	0.994	1.000	0.824	1.000	0.822	0.999
PC 23	19,088	13	14	12	0.721	0.886	0.699	0.878	0.828	0.930	0.985	0.997	0.987	0.991	0.975	0.989
PC 24	13,767	27	21	27	0.742	0.916	0.795	0.861	0.750	0.999	0.995	0.997	0.992	1.000	0.987	0.998
PC 25	18,331	38	26	35	0.677	0.873	0.841	0.862	0.665	0.921	0.997	0.998	0.986	0.998	0.984	0.996
PC 26	17,374	14	10	14	0.663	0.975	0.762	0.953	0.607	1.000	0.988	0.999	0.965	1.000	0.956	0.999
PC 27	19,339	9	7	9	0.789	0.994	0.860	0.995	0.751	0.993	0.988	1.000	0.971	0.999	0.963	0.999
PC 28	46,364	21	15	21	0.589	0.983	0.776	0.975	0.551	0.993	0.996	0.999	0.963	0.999	0.960	0.999
PC 29	12,753	9	7	9	0.785	0.991	0.852	1.000	0.758	0.983	0.992	1.000	0.985	0.998	0.980	0.999
PC 30	2,500	13	8	12	0.468	0.873	0.666	0.873	0.421	0.887	0.974	0.995	0.863	0.978	0.855	0.974
PC 31	22,098	22	18	23	0.729	0.869	0.813	0.906	0.758	0.887	0.992	0.995	0.980	0.991	0.972	0.986
PC 32	18,950	22	18	22	0.682	0.972	0.853	0.948	0.749	1.000	0.994	0.998	0.992	1.000	0.987	0.998
PC 33	1,500	3	3	3	0.811	0.978	0.875	0.983	0.819	0.974	0.910	0.983	0.948	0.997	0.901	0.988
PC 34	12,089	14	13	14	0.788	0.939	0.823	0.892	0.806	0.998	0.994	0.998	0.979	0.999	0.976	0.998
PC 35	24,068	8	15	8	0.866	0.975	0.872	0.958	0.960	0.999	0.983	1.000	0.997	0.999	0.984	0.995

Table V.2: Number of fitted primitives and classification performance metrics: comparison between the PG-based and the HT-based algorithms.

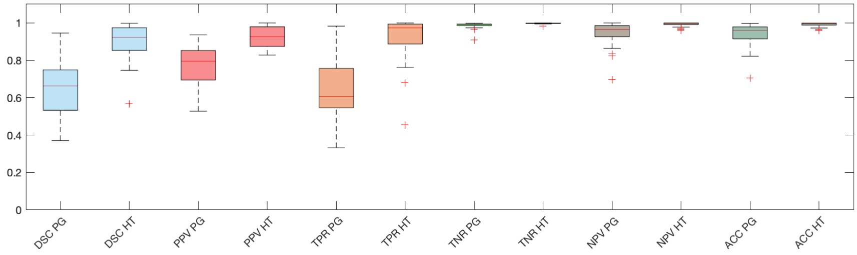


Figure V.9: Boxplot for the classification metrics presented in Table V.2. All 35 models are here considered.

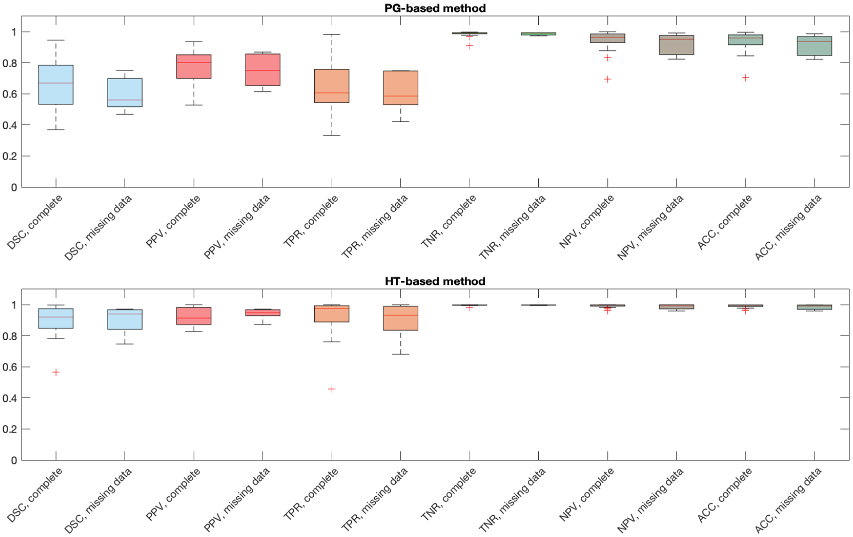


Figure V.10: Performance of the PG- and HT-based methods, with an eye on models suffering from missing data. For these boxplots, we have made use of classification metrics presented in Table V.2.

- Being the MFE normalized by definition, its value can be interpreted as a percentage. From the numbers provided in the table, we can conclude that the MFE ranges from a minimum of 0.1% to a maximum of 1.0%.

- The directed Hausdorff distance, in its normalized version, ranges from 0.2% to 1.9%. The generally higher values, compared to those from the MFE, can be explained by the Hausdorff's sensitivity to outliers.

- The coefficient distance seems to provide a much more fluid situation. By checking the model corresponding to the highest error, we can conclude that the HT-based method has lower precision when applied to point clouds containing tori.

Computational time

All tests are performed on a desktop PC equipped with an Intel Core i9 processor (at 3.6 GHz) and a Windows 10 operating system. The routines have also been tested on a MacBook Pro equipped with macOS Catalina (version 10.15.7). We provide here some statistics of the execution times, obtained on the desktop PC:

- The PG-method has minimum, mean and maximum execution time corresponding to 1.7, 286.0 and 19074.0 seconds, respectively.
- The HT-method has minimum, mean and maximum execution time corresponding to 2.6, 50.7 and 358.0 seconds, respectively.

We observe that, for small point clouds, the PG-method is generally faster, while for big point clouds it is slower.

V.5 Conclusions

In this work we have proposed Fit4CAD, a benchmark for the evaluation and comparison of methods for fitting simple geometric primitives in point clouds representing CAD objects. The ground truth dataset of point clouds is segmented in geometric primitives and subdivided into a training set and a test set. In addition, a set of quality metrics and two fitting methods are given. In this work, evaluation metrics are used to quantify various performance aspects of geometric primitive fitting methods. In our intent, these metrics would assist both comparing with some methods in literature and allowing a parameters fine-tuning of a new method, in order to optimize it on a sufficiently large set of CAD models.

We hope the results of our comparison will inspire the development of new methods for primitive fitting, computational time being the main bottleneck in practice. In particular, it would be interesting to have a comparison with methods that use machine learning approaches, such as [26], because the dataset has been already organized in the form of a training set and a test set.

Regarding the two tested methods, the overall quality of the fitting is satisfactory for both. A rather unexpected conclusion is that over-segmentation is quite limited for both methods, while the combination of small and large primitives is a challenging task that often leads to a significant under-segmentation, see for instance the outcome on the model PC 9.

In future, we plan to continue to expand the dataset, even if we do not aim at a large scale dataset, for example by including more complex primitives and possibly considering specific contexts such as assembly models. Moreover, the flexibility of Fit4CAD permits the insertion of other available methods to reach

Table V.3: Approximation accuracy of the HT method.

	MFE	d_{dHaus}	d_1
PC 1	0.002	0.005	0.008
PC 2	0.007	0.011	0.149
PC 3	0.002	0.004	0.609
PC 4	0.002	0.003	0.000
PC 5	0.004	0.006	0.082
PC 6	0.003	0.005	0.001
PC 7	0.007	0.011	0.116
PC 8	0.004	0.013	0.001
PC 9	0.003	0.006	0.000
PC 10	0.005	0.019	1.203
PC 11	0.003	0.006	0.165
PC 12	0.002	0.004	0.058
PC 13	0.002	0.005	0.000
PC 14	0.004	0.006	1.264
PC 15	0.001	0.003	0.000
PC 16	0.003	0.004	0.029
PC 17	0.003	0.007	0.000
PC 18	0.003	0.004	0.324
PC 19	0.003	0.006	0.019
PC 20	0.002	0.003	0.001
PC 21	0.004	0.007	0.566
PC 22	0.002	0.005	0.000
PC 23	0.006	0.012	0.002
PC 24	0.001	0.003	0.000
PC 25	0.002	0.003	0.001
PC 26	0.001	0.003	0.000
PC 27	0.003	0.005	0.301
PC 28	0.001	0.002	0.000
PC 29	0.010	0.003	0.119
PC 30	0.002	0.009	0.003
PC 31	0.002	0.003	0.000
PC 32	0.006	0.007	0.000
PC 33	0.003	0.006	0.000
PC 34	0.003	0.005	0.004
PC 35	0.004	0.006	0.000

a more complete view of the different typologies of approaches for geometric primitive fitting.

The benchmark is available at https://github.com/chiararomanengo/Fit4CAD.

Acknowledgements. This work has been developed in the CNR IMATI research activities DIT.AD004.100 and DIT.AD021.080.001. The authors thank Michela Spagnuolo for the fruitful discussions and Silvia Biasotti for her substantial contribution to the paper.

References

[1] Barrowclough, O. and Dokken, T. “Approximate implicitization using linear algebra”. In: *Journal of Applied Mathematics* (2012).

[2] Beltrametti, M. et al. “Moore–Penrose approach in the Hough transform framework”. In: *Applied Mathematics and Computation* vol. 375 (2020).

[3] Beltrametti, M. C. and Robbiano, L. “An algebraic approach to Hough transforms”. In: *Journal of Algebra* vol. 371 (2012), pp. 669–681.

[4] Biasotti, S. et al. “Tracking the evolution of rainfall precipitation fields using persistent maxima”. In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference.* 2016, pp. 29–37.

[5] Bojer, C. S. and Meldgaard, J. P. “Kaggle forecasting competitions: An overlooked learning opportunity”. In: *International Journal of Forecasting* vol. 37, no. 2 (Apr. 2021), pp. 587–603.

[6] Chen, X., Golovinskiy, A., and Funkhouser, T. “A Benchmark for 3D Mesh Segmentation”. In: *ACM Trans. Graph.* vol. 28, no. 3 (July 2009).

[7] Deza, M. M. and Deza, E. *Encyclopedia of Distances.* Springer Berlin Heidelberg, 2009.

[8] Dokken, T. “Aspects of Intersection Algorithms and Approximation”. PhD thesis. University of Oslo, 1997.

[9] Edelsbrunner, Letscher, and Zomorodian. “Topological Persistence and Simplification”. In: *Discrete Comput. Geom.* vol. 28, no. 4 (Nov. 2002), pp. 511–533.

[10] Ganapathi-Subramanian, V. and al., et. “Parsing Geometry Using Structure-Aware Shape Templates”. In: *2018 Int. Conf. on 3D Vision (3DV).* Los Alamitos, CA, USA: IEEE Computer Society, Sept. 2018, pp. 672–681.

[11] Gelfand, N. and Guibas, L. J. “Shape Segmentation Using Local Slippage Analysis”. In: *Proceedings of the 2004 Eurographics – ACM SIGGRAPH Symposium on Geometry Processing.* SGP ’04. Nice, France: ACM, 2004, pp. 214–223.

[12] Geuzaine, C. and Remacle, J.-F. “GMSH: A 3-D finite element mesh generator with built-in pre- and post-processing facilities”. In: *Int. Journal for Num. Meth. in Eng.* vol. 79, no. 11 (2009), pp. 1309–1331.

[13] Hough, P. V. C. *Method and means for recognizing complex patterns.* US Patent 3069654. Dec. 1962.

[14] *Geometrical product specifications (GPS)– General concepts- Part 1: Model for geometrical specification and Verification.* Standard. Geneva, CH: ISO, Mar. 2011.

[15] Kaiser, A., Ybanez Zepeda, J. A., and Boubekeur, T. “A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data”. In: *Computer Graphics Forum* vol. 38, no. 1 (2019), pp. 167–196.

[16] Koch, S. et al. “ABC: A Big CAD Model Dataset For Geometric Deep Learning”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2019.

[17] Kuhn, M. and Johnson, K. *Applied Predictive Modeling.* Springer New York, 2018.

[18] Lavoué, G. et al. “SHREC’12 Track: 3D Mesh Segmentation”. In: *Eurographics Workshop on 3D Object Retrieval.* The Eurographics Association, 2012.

[19] Mathur, A., Pirron, M., and Zufferey, D. “Interactive Programming for Parametric CAD”. In: *Computer Graphics Forum* vol. 39, no. 6 (2020), pp. 408–425.

[20] Pauly, M., Gross, M., and Kobbelt, L. P. “Efficient simplification of point-sampled surfaces”. In: *IEEE Visualization, 2002.* 2002, pp. 163–170.

[21] Qi, C. R. and al., et. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference CVPR.* 2017, pp. 77–85.

[22] Qie, Y., Qiao, L., and Anwer, N. “Enhanced Invariance Class Partitioning using Discrete Curvatures and Conformal Geometry”. In: *Computer-Aided Design* vol. 133 (2021), p. 102985.

[23] Schnabel, R., Wahl, R., and Klein, R. “Efficient RANSAC for Point-Cloud Shape Detection”. In: *Computer Graphics Forum* vol. 26, no. 2 (June 2007), pp. 214–226.

[24] Smeaton, A. F., Over, P., and Kraaij, W. “Evaluation campaigns and TRECVid”. In: *MIR ’06: Proc. of the 8th ACM Int. Workshop on Multimedia Information Retrieval.* Santa Barbara, California, USA: ACM Press, 2006, pp. 321–330.

[25] Tal, A. and Zuckerberger, E. “Mesh Retrieval by Components”. In: *Advances in Computer Graphics and Computer Vision.* Ed. by Braz, J. e. a. Springer Berlin Heidelberg, 2007, pp. 44–57.

[26] Tulsiani, S. et al. “Learning Shape Abstractions by Assembling Volumetric Primitives”. In: *Computer Vision and Pattern Regognition (CVPR).* 2017.

[27] Veltkamp, R. C. et al. *SHREC2006: 3D Shape Retrieval Contest.* Tech. rep. UU-CS-2006-030. Utrecht University, 2006.

[28] Vv, A. *The Shape Repository.* http://visionair.ge.imati.cnr.it/ontologies/shapes/. 2011–2015.

[29] Wu, Z. et al. “3D shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015, pp. 1912–1920.

[30] Zhou, Q. and Jacobson, A. “Thingi10K: A Dataset of 10, 000 3D-Printing Models”. In: *CoRR* vol. abs/1605.04797 (2016). arXiv: 1605.04797. URL: http://arxiv.org/abs/1605.04797.

Appendix V.A File type description

The geometrical information of the primitives are provided in the files “*PCi_parametric*" and “*PCi_implicit*". Here, we describe in detail the equations of the parametric and implicit representation that they contain. Notice that, although the simple primitives shapes considered in this benchmark are polynomial (i.e., can be written as the zero set of a bivariate polynomial), the parametric representation we provide for cylinders, cones, spheres and tori are written in terms of trigonometric functions.

V.A.1 Parametric representations

- Plane:

$$\begin{cases} x = a_1 u + b_1 v + c_1 \\ y = a_2 u + b_2 v + c_2 \\ z = a_3 u + b_3 v + c_3 \end{cases}$$

  The parameters for a plane are stored as follows:

```
[Plane, [a1 a2 a3 b1 b2 b3 c1 c2 c3]]
```

- Cylinder:

$$\begin{cases} x = a_1 \cos(u) + b_1 \sin(u) + c_1 v + d_1 \\ y = a_2 \cos(u) + b_2 \sin(u) + c_2 v + d_2 \\ z = a_3 \cos(u) + b_3 \sin(u) + c_3 v + d_3 \end{cases}$$

  The parameters for a cylinder are stored as follows:

```
[Cylinder, [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3]
```

- Cone:

$$\begin{cases} x = a_1 \cos(u) + b_1 \sin(u) + c_1 v \cos(u) + d_1 v \sin(u) + e_1 v + f_1 \\ y = a_2 \cos(u) + b_2 \sin(u) + c_2 v \cos(u) + d_2 v \sin(u) + e_2 v + f_2 \\ z = a_3 \cos(u) + b_3 \sin(u) + c_3 v \cos(u) + d_3 v \sin(u) + e_3 v + f_3 \end{cases}$$

  The parameters for a cone are stored as follows:

```
[Cone, [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3 e1 e2 e3 f1 f2
                          f3]]
```

- Sphere:

$$\begin{cases} x = a_1 \cos(u)\cos(v) + b_1 \sin(u)\cos(v) + c_1 \sin(v) + d_1 \\ y = a_2 \cos(u)\cos(v) + b_2 \sin(u)\cos(v) + c_2 \sin(v) + d_2 \\ z = a_3 \cos(u)\cos(v) + b_3 \sin(u)\cos(v) + c_3 \sin(v) + d_3 \end{cases}$$

The parameters for a sphere are stored as follows:

```
[Sphere, [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3]]
```

- Torus:

$$\begin{cases} x = a_1 \cos(u) + b_1 \sin(u) + c_1 \cos(u)\cos(v) + d_1 \sin(u)\cos(v) + e_1 \sin(v) + f_1 \\ y = a_2 \cos(u) + b_2 \sin(u) + c_2 \cos(u)\cos(v) + d_2 \sin(u)\cos(v) + e_2 \sin(v) + f_2 \\ z = a_3 \cos(u) + b_3 \sin(u) + c_3 \cos(u)\cos(v) + d_3 \sin(u)\cos(v) + e_3 \sin(v) + f_3 \end{cases}$$

The parameters for a torus are stored as follows:

```
[Torus, [a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3 e1 e2 e3 f1 f2
                          f3]]
```

V.A.2 Implicit representations

- Plane:

$$ax + by + cz + d = 0$$

The coefficients for a plane are stored as follows:

```
[Plane, [a b c d]]
```

- Cylinder:

$$ax^2 + by^2 + cy^2 + 2(dxy + exz + fyz) + 2(gx + hy + iz) + l = 0$$

The coefficients for a cylinder are stored as follows:

```
[Cylinder,[a b c d e f g h i l]]
```

- Cone:

$$ax^2 + by^2 + cy^2 + 2(dxy + exz + fyz) + 2(gx + hy + iz) + l = 0$$

The coefficients for a cone are stored as follows:

```
[Cone, [a b c d e f g h i l]]
```

- Sphere:

$$ax^2 + by^2 + cy^2 + 2(dxy + exz + fyz) + 2(gx + hy + iz) + l = 0$$

  The coefficients for a sphere are stored as follows:

```
[Sphere, [a b c d e f g h i l]]
```

- Torus: the coefficients of the implicit representation are provided in the form of a polynomial of degree 4 in x, y and z; they are stored in reverse lexicographic order.

Appendix V.B Results of the test set segmentations

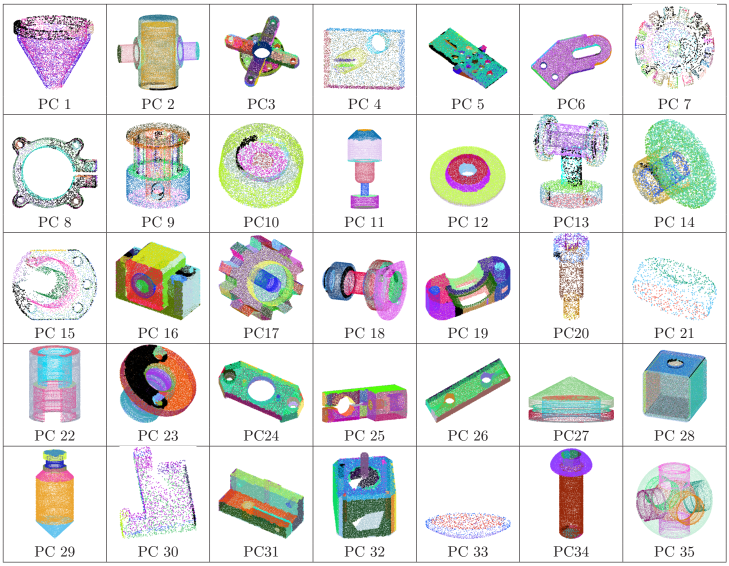


Figure V.11: Segmentations obtained via the PG-based method.

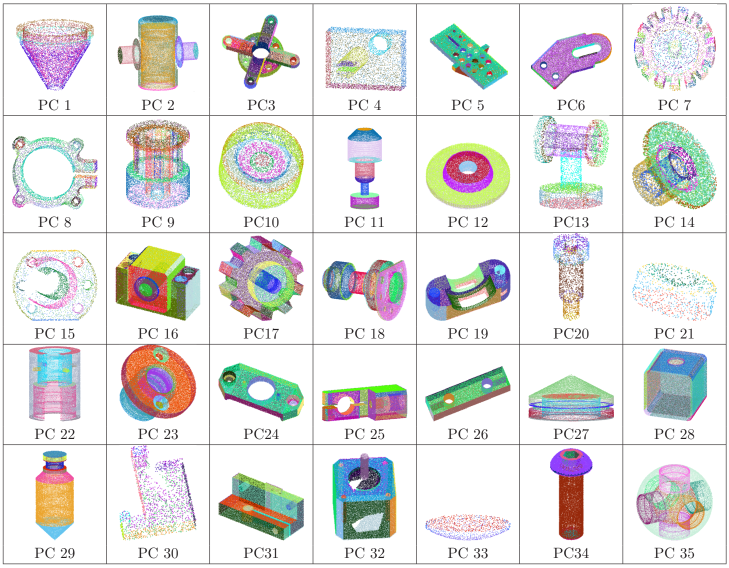


Figure V.12: Segmentations obtained via the Hough-based method.

Authors' addresses

Chiara Romanengo Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, chiara.romanengo@ge.imati.cnr.it

Andrea Raffo Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, andrea.raffo@ge.imati.cnr.it

Yifan Qie Automated Production Research Laboratory (LURPA), ENS Paris-Saclay, Université Paris-Saclay, 91190 Gif-sur-Yvette, France, yifan.qie@ens-paris-saclay.fr

Nabil Anwer Automated Production Research Laboratory (LURPA), ENS Paris-Saclay, Université Paris-Saclay, 91190 Gif-sur-Yvette, France, nabil.anwer@ens-paris-saclay.fr

Bianca Falcidieno Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magene", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy, bianca.falcidieno@ge.imati.cnr.it

Paper VI

SHREC 2021: Retrieval and classification of protein surfaces equipped with physical and chemical properties

Andrea Raffo, Ulderico Fugacci, Silvia Biasotti, Walter Rocchia, Yonghuai Liu, Ekpo Otu, Reyer Zwiggelaar, David Hunter, Evangelia I. Zacharaki, Eleftheria Psatha, Dimitrios Laskos, Gerasimos Arvanitis, Konstantinos Moustakas, Tunde Aderinwale, Charles Christoffer, Woong-Hee Shin, Daisuke Kihara, Andrea Giachetti, Huu-Nghia Nguyen, Tuan-Duy Nguyen, Vinh-Thuyen Nguyen-Truong, Danh Le-Thanh, Hai-Dang Nguyen, Minh-Triet Tran

Published in *Computers & Graphics*, October 2021, volume 99, pp. 1–21. DOI: 110.1016/j.cag.2021.06.010.

Abstract

This paper presents the methods that have participated in the SHREC 2021 contest on retrieval and classification of protein surfaces on the basis of their geometry and physicochemical properties. The goal of the contest is to assess the capability of different computational approaches to identify different conformations of the same protein, or the presence of common sub-parts, starting from a set of molecular surfaces. We addressed two problems: defining the similarity solely based on the surface geometry or with the inclusion of physicochemical information, such as electrostatic potential, amino acid hydrophobicity, and the presence of hydrogen bond donors and acceptors. Retrieval and classification performances, with respect to the single protein or the existence of common sub-sequences, are analysed according to a number of information retrieval indicators.

Keywords: SHREC, protein Surfaces, protein Retrieval, protein Classification, 3D shape analysis, 3D shape descriptor.

The paper has been awarded the Replicability Stamp, see http://www.replicabilitystamp.org.

VI. SHREC 2021: Retrieval and classification of protein surfaces equipped with physical and chemical properties

Contents

VI.1 Introduction

Automatically identifying the different conformations of a given set of proteins, as well as their interaction with other molecules, is crucial in structural bioinformatics. The well-established shape-function paradigm for proteins [33] states that a protein of a given sequence has one main privileged conformation, which is crucial for its function. However, every protein during its time evolution explores a much larger part of the conformational space. The most stable conformations visited by the protein can be experimentally captured by the NMR technique; this is because the hydrogen atoms are already included in the atomic model, thus giving less ambiguities in the charge assignment.

Recognising a protein from an ensemble of geometries corresponding to the different conformations it can assume means capturing the features that are unique to it and is a fundamental step from the structural bioinformatics viewpoint. It is preliminary to the definition of a geometry-based notion of similarity, and, subsequently, complementarity, between proteins. From the application standpoint, the identification of characteristic features can point to protein functional regions and to new target sites for blocking the activity of pathological proteins in the drug discovery field. These features can become more specific if one adds to the geometry of the molecular surface also the information related to the main physicochemical descriptors, such as local electrostatic potential [34], residue hydrophobicity [23], and the location of hydrogen bond donors and acceptors [21].

The aim of this track is to evaluate the performance of retrieval and classification of computational methods for protein surfaces characterized by physicochemical properties. Starting from a set of protein structures in different conformational states generated via NMR experiments and deposited in the PDB repository [2], we build their Solvent Excluded Surface (SES) by the freely available software NanoShaper [9, 10]. Differently from previous SHREC tracks [24, 25, 26, 38] we enrich the protein SES triangulations with scalar fields representing physicochemical properties, evaluated at the surface vertices.

The remainder of this paper is organized as follows. Section VI.2 overviews the previous benchmarks that were aimed at protein shape retrieval aspects. Then, in Section VI.3 we detail the dataset, the ground truth and the retrieval and classification metrics used in the contest. The methods submitted for evaluation to this SHREC are detailed in Section VI.4, while their retrieval and classification performances are presented in Section VI.5. Finally, discussions and concluding remarks are in Section VI.6.

VI.2 Related benchmarks

The interest of recognising proteins and other biomolecules solely based on their structure is a lively challenge in biology and the scientific literature is seeing the rise of datasets and methods for surface-based retrieval of proteins. The Protein Data Bank (PDB) repository [2] is the most widely known public repository for experimentally determined protein and nucleic acid structures. The PDB collects over $175,000$ biological macromolecular 3D structures of proteins, nucleic acids, lipids, and corresponding complex assemblies. A rather small number of proteins in the PDB dataset are captured with the NMR technique, which is very favourable for characterizing the protein also with respect to physicochemical properties. The PDB offers also a number of visualization tools of the contained structures but is not intended to perform either sequence or structure similarity tasks.

Previous benchmarks on protein retrieval based on the shape of their molecular surfaces were provided within the SHape REtrieval Contest (SHREC). In these cases, the molecular surfaces correspond to the protein solvent-excluded surface as defined by Lee and Richards [31] and firstly implemented by Connolly [8]. To the best of our knowledge, the first contest on protein shape retrieval solely based on molecular surfaces was launched in 2017 with 10 query models and a dataset of $5,854$ proteins [38]. A second contest considered a dataset of $2,267$ protein structures, representing the conformational space of 107 proteins [25]. There, the task was to retrieve for each surface the other conformations of the same protein from the whole dataset. In 2019, the SHREC track on protein retrieval [24] envisioned the classification of $5,298$ surfaces representing the conformational space of 211 individual proteins. The peculiarity of this contest was in the classification of the dataset, which took into account two levels of *similarity*. In addition to the mere retrieval of the different conformers of a given protein, the evaluation also took into account the retrieval of orthologous proteins (proteins having the same function in different organisms, e.g., human and murine haemoglobin protein) based on their surfaces. Finally, in 2020 the aim of the SHREC track on protein retrieval [26] moved to the retrieval of related multi-domains protein surfaces. Similarly to the 2019 edition, a 2-level classification (grouping different conformations of the same protein and grouping orthologs together) of the dataset was considered; in contrast to previous years, the 2020 edition included the evaluation of partial similarity, i.e., limited to sub-regions or domains of the entire molecule, thus moving towards a problem of partial correspondence between the proteins.

Compared to the 2019 track on protein retrieval, our benchmark differs in multiple points:

- Our data set does not limit to geometric information, but takes also into account physicochemical properties. Moreover, the data set is already split into a training set and a test set.

- The set of retrieval evaluation measures is considerably extended; a set of classification measures is also introduced.
- We consider two novel ground truths, based on the domain of bioinformatics.

VI.3 The benchmark

During years, we witness the consolidation of the idea that to have a more satisfactory answer to the protein shape retrieval problem it is necessary to combine geometry with patterns of chemical and geometric features [15]. For this reason, we move from the previous SHREC experiences to build a dataset equipped of both characteristics.

VI.3.1 The dataset

The dataset proposed for this challenge consists of 209 PDB entries, each one containing a protein in different conformations experimentally determined via NMR measurements. This leads to about $5,000$ surfaces, annotated with physicochemical properties. Some example of proteins in different conformational states are provided in Figure VI.1.

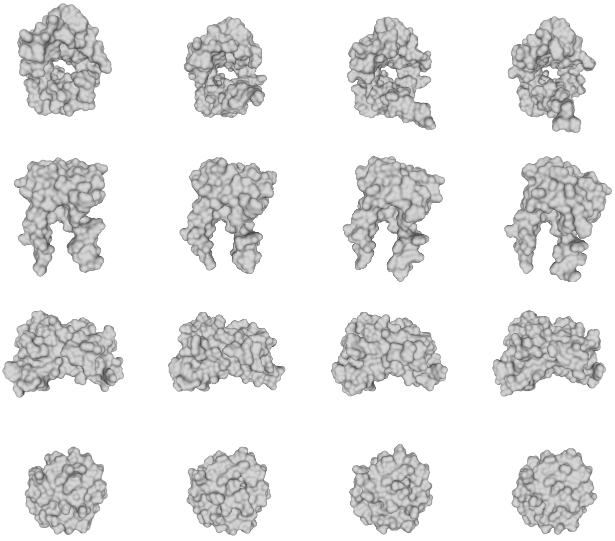


Figure VI.1: Example of 4 proteins in 4 different conformations (each row identifies a protein). Visualization obtained by using MeshLab [7].

Model surfaces were built starting from the PDB files of proteins used in the 2019 SHREC track [24]. These proteins were experimentally captures with the NMR technique and contain also orthologous structures thus making possible to consider multiple levels of similarity. NMR structures natively include hydrogen atoms, which do not need to be modelled. Importantly, these structures encompass a number of energetically favourable conformations of the same protein, representing important regions of the corresponding conformational space. Each individual conformation structure was first separated into a unique PDB file. Then, its molecular surface (MS) was calculated and triangulated by means of the NanoShaper computational tool, choosing the Connolly Solvent Excluded Surface model [8], and default parameters [9]. The vertices of the triangulated surfaces were stored in OFF[1] format.

Each surface model was accompanied by a file with physicochemical information, in TXT format. Each row of the TXT file corresponds to a vertex of the triangulation in the OFF file (in the same order); each row in the TXT file contains the physicochemical properties evaluated at the corresponding vertex in the OFF file. An example of protein surface equipped with physicochemical properties is provided in Figure VI.2: more specifically, Figure VI.2(a) exhibits the original triangulated surface, while Figures VI.2(b-d) represent the three provided physicochemical properties as scalar values on the protein surface.

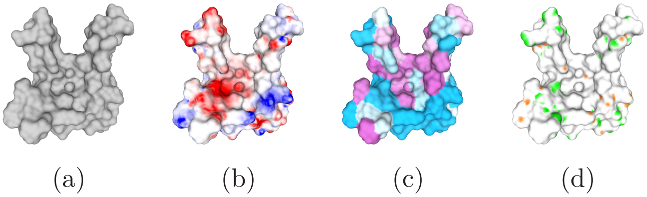


Figure VI.2: Example of protein surface (a) equipped with different physicochemical properties: electrostatic potential (b), hydrophobicity (c) and presence of hydrogen bond donors and acceptors (d). Visualization obtained by using MeshLab [7].

The dataset has been subdivided into a training and a test set (in the proportion of 70%-30%). The distribution of the number of conformations per PDB through the training set and the test set is shown in Figure VI.3.

To enrich the MS information we used the electrostatic potential, which we computed by solving the Poisson-Boltzmann equation (PBE) via the DelPhi finite-differences-based solver [34, 40]. One of the essential ingredients for the solution of the PBE is a good definition of the MS, which is used to separate the high (solvent) from the low (solute) dielectric regions. In order to guarantee the perfect consistency of the approach, we adopted a DelPhi version integrated with NanoShaper [9], so as that the potential is evaluated on the same exact surface

[1] https://segeval.cs.princeton.edu/public/off_format.html

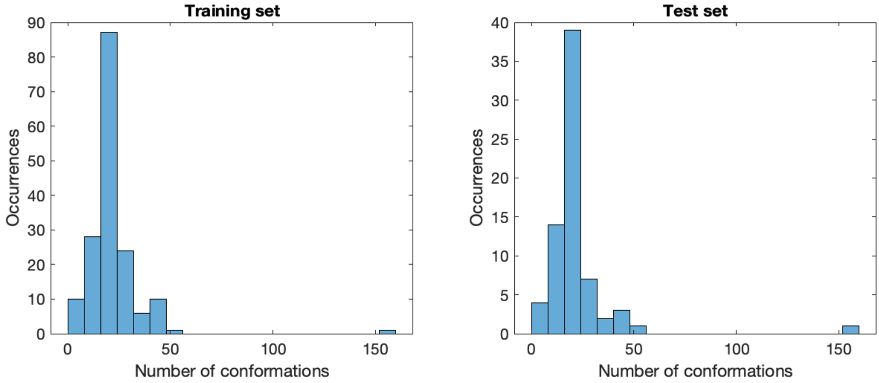


Figure VI.3: Distribution of the number of conformations. The two histograms show the distributions for the training set (left) and the test set (right).

that separates the solute from the solvent. Other necessary ingredients are atom radii and partial charges, which have been assigned using the PDB2PQR tool [20].

A different kind of additional information mapped on the MS was the hydrophobicity [23] of the residues exposed to the solvent. In our setting, we assign to each vertex of the MS the hydrophobicity of the residue of the closest atom, on the basis of the scale given in [23]; this scale ranges from -4.5 (hydrophilic) to 4.5 (hydrophobic).

Lastly, we have computed the location of potential hydrogen bond donors and acceptors in the MS. Firstly, vertices of the MS whose closest atom is a polar hydrogen, a nitrogen or an oxygen were identified. Then, a value between -1 (optimal position for a hydrogen bond acceptor) and 1 (optimal position for a hydrogen bond donor) was assigned to such vertices depending on the orientation between the corresponding heavy atoms (see [21]).

VI.3.2 The ground truth

The performances of the methods that participated to this SHREC contest are evaluated on the basis of two classifications:

- *PDB-based classification.* In the dataset selected, there is a number of entries having different PDB codes that contain the structures of the same protein, possibly interacting with different molecules or having a limited number of point mutations. In these cases it can be expected that the specific condition in which the protein system has been observed impacts on the identified conformations and on the corresponding physicochemical properties. The first classification rewards the techniques that are particularly good at spotting minor differences between similar candidates;

in this classification, a class is made by all the conformations corresponding to the same PDB code. For reference, we refer to this ground truth as PDB-based classification.

- *BLAST-based classification.* Protein sequences fold into unique 3-dimensional (3D) structures and proteins with similar sequences adopt similar structures [35]. Therefore, on the basis of the similarity among the amino acids sequences, we decided to relax the strict relationship that two surfaces are similar only if they correspond to some conformation of the same PDB code. This choice is based on observations coming from the domain of bioinformatics, where a sequence similarity beyond a value of about 30%, and of sufficient length, has a high likelihood of giving rise to the same fold [35]. We derive a second classification and name it BLAST-based classification, since BLASTP is the tool that we used to perform the sequence alignment and to calculate the sequence similarity [5]. The BLAST-based classification represents a classification less fine than the PDB-based one, because it is simply based on the similarity between conformations; in this way, not only the different NMR conformations found in the same PDB file, but also these of the same protein in different PDB files or these pertaining to its mutated isoform(s) may be grouped together. The BLAST-based classification presents four levels. In this setting, two structures are:
  - *Extremely similar (similarity level 3)*, i.e. corresponding to the same protein or very closely related protein isoforms: when they have a sequence similarity greater than 95% on at least the 95% of both sequences.
  - *Highly related (similarity level 2)*, i.e. they are expected to have a similar fold as a whole or in a sub-domain (above what in the bioinformatics jargon is called the “twilight zone"): when they have a sequence similarity greater than 35% and at least 50 aligned residuals, but they do not satisfy the conditions of the previous point.
  - *Similar (similarity level 1)*, i.e. loosely related proteins: when they have a sequence similarity in $[28\%, 35\%]$ and at least 50 aligned residuals.
  - *Dissimilar (similarity level 0)*, i.e. unrelated proteins: when none of the previous conditions holds.

To compare the performance of the methods that make use of the physicochemical properties against the simple geometric models, we asked the participants to perform two tasks:

Task A: only the OFF files of the models are considered (i.e. only the geometry is considered);

Task B: in addition to the geometry, the participant is asked to also consider the TXT files (physicochemical matching).

For a given query, the goal of this SHREC track is twofold: for each Task (A and B), to retrieve the most similar objects (retrieval problem) and to classify the query itself (classification problem). The closeness of the retrieved structures with the ground truth might be evaluated a-priori on the basis of their PDB code or of their sequence similarity (4-level BLAST classification) [35].

Retrieval problem Each model is used as a query against the rest of the dataset, with the goal of retrieving the most relevant surface. For the retrieval problem, a dissimilarity $1,543 \times 1,543$ matrix was required, each element (i, j) recording the dissimilarity value between models i and j in the whole dataset. The relevance with respect to the query of a retrieved surface is evaluated with both the PDB and BLAST classifications previously described.

Classification problem PDB-based and BLAST-based classifications define on the training and on the test sets a decomposition into subsets (that will be referred as communities) consisting of conformations grouped together on the basis of their similarity. The goal of the classification problem is to assign each query of the test set to the correct community with respect to the decompositions induced by the PDB-based and the BLAST-based classifications, respectively.

In the case of the PDB-based classification, each community consists of all the conformations corresponding to the same PDB code.

In the case of the BLAST-based classification, different community decompositions are obtained depending on the choice of the previously described similarity levels. For each level ℓ (with $\ell = 0, 1, 2, 3$), it is possible to retrieve a decomposition into communities referred as BLAST-based community decomposition of level ℓ. Independently from the chosen level ℓ, each community of the BLAST-based decomposition of level ℓ is an aggregation of communities induced by the PDB-based classification.

Having fixed a level ℓ, the communities of the BLAST-based decomposition of level ℓ are computed as it follows. Let us consider a graph G_ℓ for which each node represents a PDB-based community (i.e. models corresponding to the same PDB code) and such that there exists an edge (u, v) whenever the structures u and v have a similarity level greater than or equal to ℓ. Moreover, each edge (u, v) is endowed with a weight $w(u, v)$ coinciding with the percentage of sequence similarity between u and v. The clustering technique for retrieving the BLAST-based decomposition of level ℓ adopts the following recursive strategy which has been specifically designed for the considered framework but it is inspired by classic methods for community detection [14]. Given G_ℓ, compute the connected components of G_ℓ obtained after the removal the edge of G_ℓ of minimum weight (and so representing a low similarity score between models). A connected component C is declared a BLAST-based community of level ℓ if C is a complete graph (i.e. given any two of its nodes there is an edge connecting them). Otherwise, keep removing edges (prioritising the ones with the lowest weight), compute the connected components and denote them as BLAST-based community of level ℓ whenever they are complete. The procedure ends when all the nodes have been inserted in a community.

It is worth to be noticed that the completeness condition has been imposed in order to obtain transitive BLAST-based communities. In this way, we have the theoretical guarantee that any two structures belonging to the same BLAST-based community of level ℓ have necessarily a similarity level greater than or equal to ℓ. Another relevant aspect to be mentioned is related to the fact that, for $\ell = 3$, the proposed algorithm does not remove any edge since the connected components of the graph G_ℓ are already complete (see Figure VI.4). Trivially, the same happens also for $\ell = 0$ since the BLAST-based decomposition of level 0 produces just a unique "giant" community consisting of the entire dataset. Finally, please notice that, by increasing the value ℓ, one obtains BLAST-based decompositions consisting of finer communities.

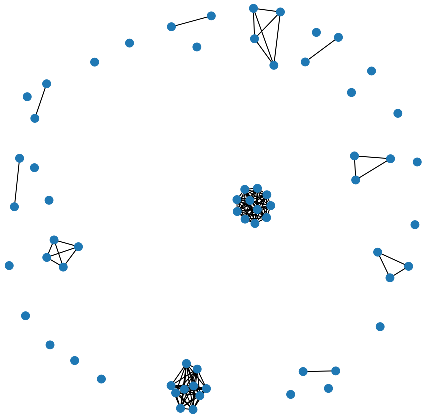


Figure VI.4: The graph G_3 associated with the test set. Each node represents a community induced by the PDB-based classification, while an edge between two nodes occurs whenever the corresponding structures are extremely similar (i.e. they have similarity level equal to 3). Since they are all complete graphs, each connected component of G_3 represents a community of the BLAST-based decomposition of level 3.

Given the PDB and BLAST ground truth for the classification problem, the classification performance of each run is obtained through the nearest neighbour (1-NN) classifier derived from the dissimilarity matrices used in the retrieval problem. For each run, the output consists of two classification arrays for the test set, with 65 labels/communities for the PDB-based decomposition and with 31 labels/communities for the BLAST-based decomposition of level 3 (see Figure VI.4). In these arrays, the element i is set to j if i is classified in class j (that is, the nearest neighbour of the surface i belongs to class j).

VI.3.3 Evaluation measures

We here presents the retrieval and classification evaluation measures that were selected for this benchmark.

VI.3.3.1 Retrieval evaluation measures

3D retrieval evaluation has been carried out according to standard measures, namely precision-recall curves, mean Average Precision (mAP), Nearest Neighbour (NN), First Tier (1T), Second Tier (2T), Normalized Discounted Cumulated Gain (NDCG) and Average Dynamic Recall (ADR) [1, 32, 39].

Precision-recall curves and mean average precision Precision and recall measures are commonly used in information retrieval [32]. Precision is the fraction of retrieved items that are relevant to the query. Recall is the fraction of the items relevant to the query that are successfully retrieved. Being A the set of relevant objects and B the set of retrieved object,

$$\text{Precision} = \frac{|A \cap B|}{|B|}, \quad \text{Recall} = \frac{|A \cap B|}{|A|}.$$

Note that the two values always range from 0 to 1. For a visual interpretation of these quantities we plot a curve in the reference frame recall vs. precision. We can interpret the result as follows: the larger the area below such a curve, the better the performance under examination. In particular, the precision-recall plot of an ideal retrieval system would result in a constant curve equal to 1. As a compact index of precision vs. recall, we consider the mean Average Precision (mAP), which is the portion of area under a precision recall-curve: the mAP value is always smaller or equal to 1.

e-Measure The e-Measure (eM) derives from the precision and recall values for a pre-defined number of retrieved items (32 in our settings), [32, 37]. Given the first 32 items for every query, the e-Measure is defined as $e = \frac{1}{\frac{1}{P}+\frac{1}{R}}$, where P and R represent the precision and recall values over them.

Nearest Neighbour, First Tier and Second Tier These evaluation measures aim at checking the fraction of models in the query's class also appearing within the top k retrievals. Here, k can be 1, the size of the query's class, or the double size of the query's class. Specifically, for a class with $|C|$ members, $k = 1$ for the Nearest Neighbour (NN), $k = |C| - 1$ for the First Tier (1T), and $k = 2(|C| - 1)$ for the Second Tier (2T). Note that all these values necessarily range from 0 to 1. In our this contest, we estimate the NN, FT, ST, and e values using the tools provided in the Princeton Shape Benchmark [37].

Average dynamic recall The idea is to measure how many of the items that should have appeared before or at a given position in the result list actually have appeared. The Average Dynamic Recall (ADR) at a given position averages this measure up to that position. Precisely, we adapt the definition of the ADR to our four level BLAST classification, slightly modifying the definition used in previous datasets equipped with a multi-level classification, such as [3, 4]. For a given query let A be the extremely similar (SR) items, B the set of highly related (HR) items, and let C be the set of similar (MR) items. Obviously $A \subseteq B \subseteq C$. The ADR is computed as:

$$\mathrm{ADR} = \frac{1}{|C|} \sum_{i=1}^{|C|} r_i,$$

where r_i is defined as:

$$r_i = \begin{cases} \frac{|\{\text{SR items in the first } i \text{ retrieved items}\}|}{i}, & \text{if } i \leq |A|; \\ \frac{|\{\text{HR items in the first } i \text{ retrieved items}\}|}{i}, & \text{if } |A| < i \leq |B|; \\ \frac{|\{\text{MR items in the first } i \text{ retrieved items}\}|}{i}, & \text{if } i > |B|. \end{cases}$$

Normalized discounted cumulated gain For its definition we assume that items with highest similarity score according to the BLAST classification are more useful if appearing earlier in a search engine result list (i.e. are first ranked); and, the higher their level of similarity (extremely similar, highly related, similar and dissimilar) the higher their contribution, and therefore their gain. As a preliminary concept we introduce the *Discounted Cumulated Gain (DCG)*. Precisely, the DCG at a position p is defined as:

$$\mathrm{DCG}_p = \mathrm{rel}_1 + \sum_{i=2}^{p} \frac{\mathrm{rel}_i}{\log_2(i)},$$

with rel_i the graded relevance of the result at position i. Obviously, the DCG is query-dependent. To overcome this problem, we normalize the DCG to get the Normalized Discounted Cumulated Gain (NDCG). This is done by sorting elements of a retrieval list by relevance, producing the maximum possible DCG till position p, also called *ideal DCG (IDCG)* till that position. For a query, the NDCG is computed as

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p}.$$

It follows that, for an ideal retrieval system, we would have $\mathrm{NDCG}_p = 1$ for all p.

VI.3.3.2 Classification performance measures.

A set of popular performance metrics in statistical classification is derived by the so-called *confusion matrix* [22]. A confusion matrix is a square matrix whose order equals the number of classes in the dataset (in our case, in the test set). The diagonal element CM(i,i) gives the number of items (i.e. molecular surfaces, in our context) which have been correctly predicted as elements of class i. On the contrary, off-diagonal elements count items that are mislabeled by the classifier: in other words, CM(i,j), with $j \neq i$, represents the number of items wrongly labeled as belonging to class j rather than to class i. The classification matrix CM of an ideal classification system is a diagonal matrix, so that no misclassification occurs.

Sensitivity and specificity These statistical measures are among the most widely used in diagnostic test performance. Sensitivity, also called *True Positive Rate* (TPR), measures the proportion of positives which are correctly identified as such (e.g., the percentage of dogs correctly classified as dogs). Specificity, or *True Negative Rate* (TNR), measures the proportion of negatives which are correctly identified as such (e.g., the percentage of non-dogs correctly classified as non-dogs). A perfect classifier is 100% sensitive and 100% specific.

Positive and negative predicted values Specificity and sensitivity tell how well a classifier can identify true positives and negatives. But what is the likelihood that a test result is a true positive (or true negative) rather than a false-positive (or a false-negative)? *Positive Predictive Rate* (TPR) measures the proportion of *true positives* among all those items classified as positives. Similarly, *Negative Predictive Rate* (NPR) measures the proportion of *true negatives* among all those items classified as negatives.

Accuracy This metric measures how often the classifier is correct: it is the ratio of the total number of correct predictions to the total number of predictions.

F_1 score It takes into account both PPV and TPR, by computing their harmonic mean: this allows to consider both false positive and false negatives. Therefore, it performs well on an imbalanced dataset.

VI.4 Description of methods

Eight groups from five different countries registered to this track. Five of them proceeded with the submission of their results. Each participant was allowed to send us up to three runs for each task, in the form of a dissimilarity matrix per run. All but one submitted three runs per task; one participant delivered three runs for Task A and one for Task B. Overall, Task A has gathered 15 runs, while Task B has 13 runs.

In the following, we will denote the methods proposed by the five participants as P1, P2, . . . , P5.

Specifically,

- method P1 has been proposed by Andrea Giachetti;
- method P2 has been proposed by Tunde Aderinwale, Charles Christoffer, Woong-Hee Shin, and Daisuke Kihara;
- method P3 has been proposed by Yonghuai Liu, Ekpo Otu, Reyer Zwiggelaar, and David Hunter;
- method P4 has been proposed by Evangelia I. Zacharaki, Eleftheria Psatha, Dimitrios Laskos, Gerasimos Arvanitis, and Konstantinos Moustakas;
- method P5 has been proposed by Huu-Nghia Nguyen, Tuan-Duy Nguyen, Vinh-Thuyen Nguyen-Truong, Danh Le-Thanh, Hai-Dang Nguyen, and Minh-Triet Tran.

Lastly, Andrea Raffo, Ulderico Fugacci, Silvia Biasotti, and Walter Rocchia have been the organizers of the SHREC 2021 track on retrieval and classification of protein surfaces on the basis of their geometry and physicochemical properties.

The remaining part of this section is devoted to describe in detail the five proposed methods.

VI.4.1 P1: Joint histograms of curvatures, local properties and area projection transform

VI.4.1.1 Adopted descriptors and overall strategy

The proposed approach is based on the estimation of simple surface- and volume-based shape descriptors, and on their joining with the local surface properties. In a previous contest [17], it has been shown that simple joint (2D) histograms of min/max curvatures (JHC) are extremely effective in characterizing patterns of elements with approximate spherical symmetry and variable size. On the other hand, in a past contest on protein retrieval [24], we used a volumetric descriptor called the Histograms of Area Projection Transform (HAPT) [16] to characterize radial symmetries at different scales providing good results.

In method P1, we tested both descriptors and their combination to evaluate the similarity of the shapes included in the test dataset. Furthermore, having the local information on the physicochemical properties, we can improve the characterization creating joint (3D) histograms counting elements with selected properties in a space characterized by 2 curvature axes and a “property” dimension. Finally, having a labelled dataset, we evaluated the possibility of applying to the descriptor a trained dimensionality reduction based on Linear Discriminant Analysis, e.g., projecting the high-dimensional joint histogram descriptors onto a lower dimensional space maximizing the separation of the training set classes.

A visual description of the pipeline adopted in method P1 is depicted in Figure VI.5.

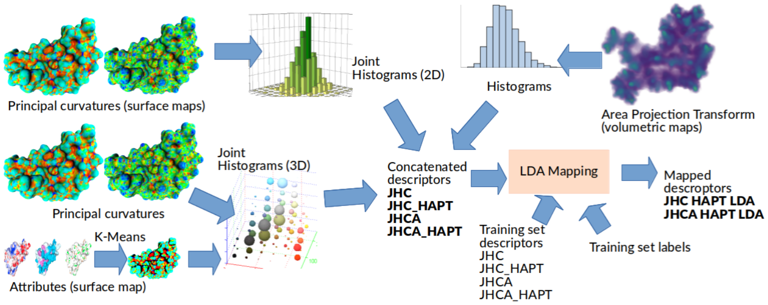


Figure VI.5: A graphical representation of the strategy adopted by group P1 to obtain untrained and trained descriptors. Joint histograms are obtained from the surface-based descriptors, shape only (principal curvature) and attributes (labels derived from K-means clustering of the provided vectors, $K = 50$). Simple histograms are obtained from the volumetric symmetry descriptor (APT). Histograms are concatenated in different ways to obtain the different descriptors compared with the Jeffrey Divergence. Trained descriptors are obtained by estimating LDA mappings of the descriptors of the training set shapes with known labels. The mappings are used to perform dimensionality reduction on the test set elements.

VI.4.1.2 Task A

Joint Histograms of Curvature A basic technique to distinguish surfaces endowed with multiple spherical bumps is to measure curvature values. Minimum and maximum curvatures have been estimated on the mesh vertices at two different scales. The ranges of min curvature and max curvature have been subdivided in 10 bins estimating the joint histogram (size 100). Concatenating the two joint histograms corresponding to the two smoothing levels a final descriptor with 200 elements is obtained. Histograms are compared with Jeffrey divergence [29] to obtain dissimilarity matrices.

Histograms of Area Projection Transform For the mathematical formulation of the technique please refer to the original paper [16]. In a few words, the internal part of the shape is discretized on a regular grid and for each voxel and for a set of discrete radius values r, it is counted how much of the object surface can be considered approximately part of a sphere of radius r centered in the voxel. Looking at some example protein shapes, we decided to apply the technique with voxel size 0.3 and 9 values of r ranging from 0.6 to 3.0 with step 0.3. Then, we binarized the histograms with 12 bins and concatenated the histograms computed at the 9 scales. This resulted in a HAPT descriptor with 108 elements which have been finally concatenated into a 308-element shape descriptor capturing the distribution of curvatures on the surface and radial symmetry inside the volume. The Jeffrey divergence [29] was used to obtain from them the dissimilarity matrices.

Trained descriptors As the data comes with a labelled training set, it is possible to use it to train the dissimilarity metric to maximally separate elements with different labels. This has been achieved by using the Fisher's Linear Discriminant Analysis [13], projecting the original descriptor onto a $C-1$ dimensional space (where C is the number of classes in the training set) maximizing the ratio of the variance between the classes to the variance within the classes. LDA mapping for high-dimensional descriptors has been trained on the training set and used on the test data to evaluate the effectiveness of the approach.

For Task A, three different runs adopting method P1 and generated as it follows have been proposed.

- **Run 1 (JHC):** generated by the Joint Histograms of Curvature (two different levels of smoothing) compared with the Jeffrey divergence.
- **Run 2 (JHC_HAPT):** generated by concatenating Joint Histograms of Curvature and Histograms of Area Projection Transform compared with Jeffrey divergence.
- **Run 3 (JHC_HAPT_LDA):** generated by concatenating JHC and HAPT descriptors mapped with trained LDA projection.

VI.4.1.3 Task B

In method P1, we did not use any prior related to the knowledge of the meaning of the attributes associated with the mesh vertices and we just considered them as generic components of a 3D feature space. The adopted strategy has been to partition this feature space in a set of regions, and estimating for each model histograms counting the number of vertices with features falling in each region. To determine a reasonable partitioning, we just applied K-means clustering with $K = 50$ to the all the dataset vertex attributes and extract corresponding Voronoi cells. Using 50 cells, a 50-elements histogram to describe shape features is retrieved. Joining the attribute dimension to the two curvature dimensions, a 3D joint histogram per vertex with $10 \times 10 \times 50 = 5,000$ elements is obtained. These Joint Histograms of Curvatures and Attributes JHCA can be directly compared with the Jeffrey divergence. However, we also tested the combination of JHCA with HAPT and the LDA-based dimensionality reduction.

For Task B, three different runs adopting method P1 and generated as it follows have been proposed.

- **Run 1 (JHCA):** generated by the Joint Histograms of Curvature (single smoothing level) and Attributes (50 centroids) compared with the Jeffrey divergence.
- **Run 2 (JHCA_HAPT):** generated by concatenating Joint Histograms of Curvature and Attributes and Histograms of Area Projection Transform compared with Jeffrey divergence.

- **Run 3 (JHCA_HAPT_LDA):** generated by concatenating JHCA and HAPT descriptors mapped with trained LDA projection.

VI.4.1.4 Computational aspects

Experiments have been perfomed on a laptop with an Intel®CoreTM i7-9750H CPU running Ubuntu Linux 18.04. The estimation of the descriptors JHC and JHCA took on average 1.5 seconds per model using Matlab code, while the estimation of the HAPT descriptor took on average 15 seconds. Other required operations included: the descriptor comparison whose computation time was negligible; the training of the dissimilarity metric and the LDA mapping both implemented using Matlab and requiring 1 minute in the worst case and 0.1 seconds, respectively; the partitioning based on K-means clustering whose took approximately 30 minutes.

VI.4.2 P2: 3D Zernike descriptor

VI.4.2.1 Adopted descriptors and overall strategy

The approach adopted in method P2 is based on the 3D Zernike Descriptor (3DZD). 3DZD is a rotation-invariant shape descriptor derived from the coefficients of 3D Zernike-Canterakis polynomials [6].

3DZD descriptors are adopted as input of a neural network which will return a prediction of the similarity between any pair of proteins. In a nutshell, neural networks (NN) are a class of tools enabling the estimation of a desired function (in the current case, the similarity between proteins) inspired by the biology of a brain. NNs can be are typically represented by a directed weighted graph consisting of nodes, called neurons and subdivided into layers, and edges connecting neurons of different layers. In a NN, the leftmost layer consists of the so-called input neurons, while the rightmost nodes are called output neurons. In between, there are the hidden layers. In case a NN has at least two hidden layers, it is called a deep neural network. Each neuron of a layer takes a series of inputs, depending on the edges pointing to it, and transmits an activation value by the edges linking the considered node to a different neuron multiplying this value by the weight of the edge. Input neurons receive the features of the input variables and pass them to the next layers while, the activation values of output neurons will form the output of the NN. The desired function is obtained through a training process of the NN in which the weights are attained minimizing a loss function.

We trained two types of neural network, visually depicted in Figure VI.6, to output a score that measures the dissimilarity between a pair of protein shapes, encoded via the 3DZDs.

- The first framework (Extractor model) was previously used in a SHREC track on multi-domain protein shape retrieval, see [26]. The network is structured into multiple layers: an encoder layer, which converts 3DZD

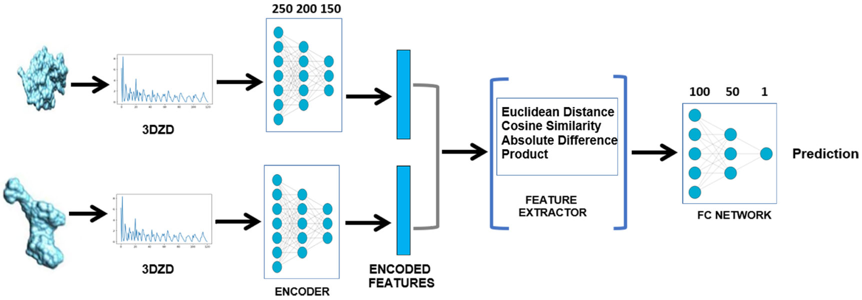


Figure VI.6: A graphical representation of the two types of neural network adopted in method P2 to measure the dissimilarity of protein shapes encoded via the 3DZDs. The Extractor and the EndtoEnd models differ by presence of the feature comparator layer (depicted inside the blue square brackets).

to a vector of 150 features, has 3 hidden units of size 250, 200, and 150, respectively; a feature comparator layer that computes the Euclidean distance, the cosine distance, the element-wise absolute difference, and product; and a fully connected layer with 2 hidden units of size 100 and 50, respectively. There are multiple hidden units in each layer. The ReLU activation function is used in all layers, except for the output of the fully connected layer where the sigmoid activation function has been preferred, This choice allows to interpret the output as the probability, for any pair of proteins, to be in the same class.

- The second framework (EndtoEnd model) is similar to the first one, except for the removal of the feature comparator layer. The output of the encoder layer directly flows into the fully connected layer and the network is trained end-to-end.

The network was trained on the training set of $3,585$ protein structures that was provided by the organizers. The training set was further split into a training set and a validation set, by using respectively 80% and 20% of data (i.e. $2,868$ conformations for training and the remaining 717 for validation). From the 717×717 classification matrix of the validation set, $10,436$ protein pairs were extracted for the purpose of network validation.

A third attempt is made via a simple Euclidean model, where the Euclidean distance between pairs of proteins has been computed directly from the generated 3DZD of the pairs.

VI.4.2.2 Task A

The performance of the networks on the validation set was used to determine models to use for inference on the test set. Training for Task A was performed on the 3DZDs of shape files only.

For Task A, three different runs adopting method P2 and generated as it follows have been proposed.

- **Run 1 (extractor):** generated by the Extractor model.
- **Run 2 (extractor_e2e):** generated by the average between Extractor and EndtoEnd models.
- **Run 3 (extractor_eucl):** generated by the average between Extractor and Euclidean models.

VI.4.2.3 Task B

As for Task A, model selection was carried out on the validation set. Training was performed on input files that concatenate 3DZD of shape with 3DZDs of the three physicochemical properties.

For Task B, three different runs adopting method P2 and generated as it follows have been proposed.

- **Run 1 (extractor):** generated by the Extractor model.
- **Run 2 (extractor_e2e):** generated by the average between Extractor and EndtoEnd models.
- **Run 3 (extractor_e2e_eucl):** generated by the average between Extractor, EndtoEnd, and Euclidean models.

VI.4.2.4 Computational aspects

For each protein in the dataset, we performed some pre-processing step to convert the OFF and TXT files provided by the organizers. The mesh and property files were converted to a volumetric skin representation (the Situs file) where points within 1.7 grid intervals were assigned with values interpolated from the mesh [36]. For the electrostatic features, the interpolated values were the potentials at the mesh vertices. For the shape features, a constant value of 1 was assigned to grids which overlap with the surface. The resulting Situs files were then fed into the EM-Surfer pipeline [11] to compute 3DZD. It took approximately $12 - 13$ minutes to pre-process each file. Generating the 3DZD descriptors took averagely 8 seconds for each protein on an Intel®Xeon®CPU E5-2630 0 @ 2.30GHz.

For Task A, training the extractor model took averagely 6 hours and the EndtoEnd model took about 11 hours. For Task B, training the extractor model took about 9 hours and approximately 14 hours for the EndtoEnd model. Training was performed on a Quadro RTX 800 GPU.

The 3DZD model took averagely 0.22 seconds to predict the dissimilarity between two proteins, using TitanX GPU. The Euclidean model took averagely 0.17 seconds per prediction. Finally, the averaging of the three matrices was virtually instant and negligible.

VI.4.3 P3: Hybrid Augmented Point Pair Signatures and Histogram of Processed Physicochemical Properties of Protein molecules

VI.4.3.1 Adopted descriptors and overall strategy

Considering the twofold nature of this challenge, in P3 we adopted two separate retrieval strategies for the two different tasks. For Task A, we used the Hybrid Augmented Point Pair Signature (HAPPS) [27], a 3D geometric shape descriptor. For Task B, we adopted the Histogram of Processed Physicochemical Properties of Protein molecules descriptor following an Exploratory Data Analysis (HP4-EDA). Both the strategies rely on traditionally hand-crafted feature extraction from the respective datasets, using the knowledge-based approach (i.e. non-learning nor data-driven approach).

The goal of the proposed methods (HAPPS and HP4-EDA) is to provide simple, efficient, robust and compact representations, describing both the 3D geometry and physicochemical properties of protein surfaces, using statistically-based descriptors. Visual descriptions of the pipelines adopted in method P3 for Tasks A and B are depicted in Figures VI.7 and VI.8, respectively.

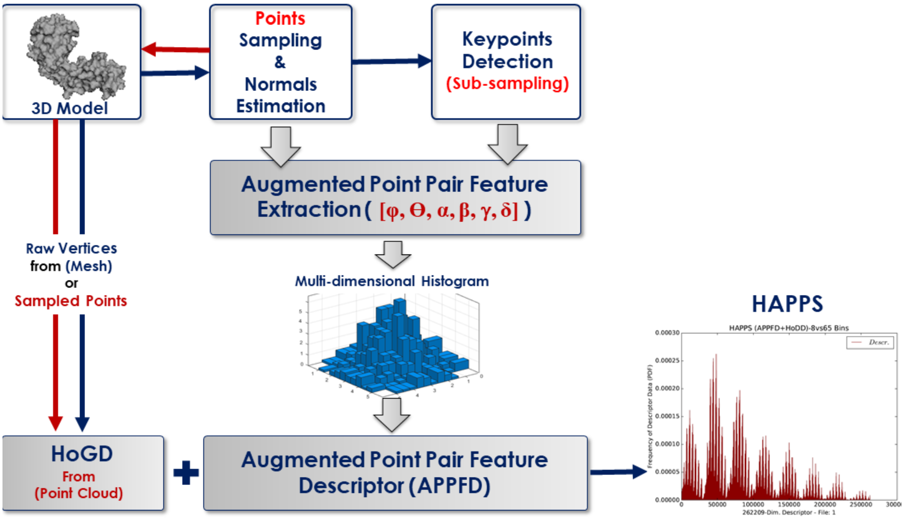

3D Model
Points Sampling & Normals Estimation
Keypoints Detection (Sub-sampling)
Augmented Point Pair Feature Extraction ([φ, Θ, α, β, γ, δ])
Raw Vertices from (Mesh) or Sampled Points
Multi-dimensional Histogram
HAPPS
HAPPS (APPFD+HoDD)-8vs65 Bins
Descr.
Frequency of Descriptor Data (PDF)
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
0
50000
100000
150000
200000
250000
3000(
262209-Dim. Descriptor - File: 1
HoGD From (Point Cloud)
+
Augmented Point Pair Feature Descriptor (APPFD)

Figure VI.7: A graphical representation of the strategy adopted in method P3 for Task A.

VI.4.3.2 Task A

Each 3D geometrical protein surface in this challenge contains an average of 35,000 vertices and 70,000 triangular faces. The HAPPS method (first introduced in [27]) involves a combination of local and global descriptors. Specifically, the Augmented Point Pair Feature Descriptor (APPFD), and the Histogram of Global

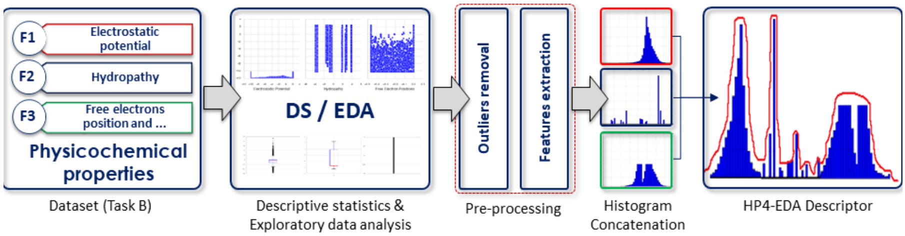


Figure VI.8: A graphical representation of the strategy adopted in method P3 for Task B.

Distances (HoGD). The proposed HAPPS method is particularly interested in providing a robust, compact, and accurate representation of protein structures with as low as N points (i.e., $[N \times 3]$) sampled from the triangular mesh surface of each input protein model, where $N = 3,500$ and $N = 4,500$.

Histogram of Global Distances (HoGD) This descriptor involves binning a set of normalized vectors $\delta_i = \|P_c - p_i\|$ between the centroid P_c of a given 3D object to all other points, p_i on its surface into a 1D histogram with $\sqrt{P} \approx 65$ bins, normalized to give HoGD, following some pre-processing steps, where $p_i \in P$ and P is a 3D point cloud object, with a number of points, $N = 3,500$ or $N = 4,500$. Such normalized vectors δ_i are regarded as global features whose distribution (histogram) is capable of expressing the configuration of the entire shape relative to its centroid, and is a rich description of the global structure of the shape. Pre-processing also involves applying uniform scale, S in all direction to all points in P, such that the root mean square (RMS) distance of each point to the origin is 1, and centering P on its centroid, i.e. $P = p_i - P_c$.

Augmented Point Pair Feature Descriptor (APPFD) The APPFD describes the local geometry around a point, $p_i = [p_{ix}, p_{iy}, p_{iz}]$ in P. Its computation involves a 5-step process: (i) pointcloud sampling and normals estimation; (ii) keypoint, p_{k_i} determination; (iii) local surface region (i.e. LSP), P_i selection; (iv) Augmented Point Pair Feature (APPF) extraction per LSP; (v) bucketing of locally extracted 6-dimensional APPF into a multi-dimensional histogram, with the number of bins, $b_{APPFD} = 8$ in each feature-dimension which is then flattened and normalised to give $8^6 = 262,144$-dimensional single local descriptor (APPFD) per 3D shape.

Finally, HoGD is combined with APPFD to give HAPPS, with a final feature vector of dimension $262,144 + 65 = 262,209$ (see Figure VI.7). For more details regarding the HoGD, APPFD and HAPPS algorithms, the reader is referred to [26, 27].

The APPFD is characterised by four key parameters, which are r, vs, b_{APPFD}, and N. r is the radius value used by the nearest-neighbour algorithm to determine the number of points in a LSP, and is directly proportional to the size of LSP. The voxel-size parameter, vs determines the size (big or small) of an occupied voxel-grid by the pointcloud down-sampling algorithm [42]. It is inversely proportional to the number of sub-sampled points (used as keypoints). r and vs influence the overall performances of the APPFD/HAPPS.

For Task A, three different runs adopting method P3 and generated as it follows varying the values of r, vs, and N have been proposed.

- **Run 1 (HAPPS):** generated by choosing $r = 0.40$, $vs = 0.20$, and $N = 4,500$.
- **Run 2 (HAPPS):** generated by choosing $r = 0.50$, $vs = 0.30$, and $N = 4,500$.
- **Run 3 (HAPPS):** generated by choosing $r = 0.50$, $vs = 0.30$, and $N = 3,500$.

Parameters $b_{APPFD} = 8$ and $b_{HoGD} = 65$ remained the same for all three runs. Overall, the Cosine distance metric between final vectors gave good approximation of the similarity between the HAPPS for Task A datasets.

VI.4.3.3 Task B

The HP4-EDA method involves a descriptive statistics (DS) of the 3-dimensional physicochemical variables or properties, following exploratory data analysis (EDA) of each of these properties.

Let the three physicochemical properties of the dataset in Task B be denoted as f_1, f_2, and f_3, for *Electrostatic Potential*, *Hydrophobicity*, and *Position of potential hydrogen bond donors and acceptors*, respectively. First, we carried out an in-depth EDA of the physicochemical properties to investigate their values distribution, followed by data pre-processing (majorly outliers detection and removal). Next, we investigated the performances of combining some DS, such as the mean, variance, first and third interquartile values, and correlation coefficients between these variables as a final descriptor, including the construction of histograms of these variables, post-processing (see Figure VI.8).

Outliers detection and removal Considering that the presence of outliers would adversely affect the performance of any retrieval system, we checked for the presence of outliers in each of f_1, f_2, and f_3. Unlike f_2, the f_1 and f_3 variables contain lots of outliers with f_3 having almost negligible amount of useful data. Empirically, the presence of outliers in a distribution may not necessarily make the observation a "bad data". For outliers detection and removal, we adopted the Interquartile Range (IQR) Score, represented by the formula $IQR = Q_3 - Q_1$, which is a measure of statistical dispersion calculated as the difference between lower (Q_1) and upper (Q_3) percentiles. Here, any observation that is not in the range of ($Q_1 - 1.5IQR$) and ($Q_3 + 1.5IQR$) is an outlier, and can be removed. We further investigated the effect of using $Q_1 = 10^{th}$ or 25^{th}, and $Q_3 = 75^{th}$ or 90^{th} and recorded better performances with the later option where $Q_1 = 10th$

and $Q_3 = 90th$ for the training set, which parameter settings were also applied to the test data.

For Task B, three different runs adopting method P3 and generated as it follows by applying statistical description techniques for the extraction of statistical features and/or construction of final descriptors from the pre-processed data have been proposed.

- **Run 1 (HP4-EDA):** generated by binning each pre-processed physicochemical variable into a 1D histogram (using 150 bins) and combining the final histograms as the final descriptor for each input physicochemical surface where matching between two descriptors is done using the Earth Mover's Distance (EMD) metric.
- **Run 2 (HP4-EDA):** generated by binning each of the pre-processed values of f_1, f_2, and f_3 into a multi-dimensional histogram, with 5 bins in each feature dimension, where the flattened and normalised histogram frequencies represent the final descriptor for a single input data and descriptors are matched using the Kullback Liebner Divergence (KLD) metric.
- **Run 3 (HP4-EDA):** generated by first normalizing each of the feature (variable) dimensions or columns, and selecting their *mean*, *variance*, Q_1, Q_3, and some correlation coefficient values between f_1, f_2, and f_3 to represent a single input physicochemical surface, with a total of 14-dimensional feature vector, and combining the outcome of Run 1 to have a feature vector of dimension $14 + (150 \times 3) = 464$ as a final descriptor representing a single input.

VI.4.3.4 Computational aspects

For Task A, the HAPPS method has been implemented in Python 3.6 and all experiments have been carried out under Windows 7 desktop PC with Intel Core i7-4790 CPU @ 3.60GHz, 32GB RAM. It took on average, 0.3 and 20.0 seconds to sample point cloud and estimate normals from 3D mesh, and extract features and compute HAPPS, respectively. Matching $1,543 \times 1,543$ testing set HAPPS descriptors took $3,212.3$ seconds using the Cosine metric, which implies an average of 2.1 seconds to match any two HAPPS.

For Task B, the HP4-EDA method has been implemented in Python 3.6 and all experimental run have been performed on 64-bit Windows 10 notebook, Intel Core(TM) i3-5157U CPU @ 2.50GHz, 8GB RAM. The extraction of the features and the computation of the HP4-EDA descriptors took an average of 0.01 seconds. Additionally, it took 322.5 seconds to match $1,543 \times 1,543$ HP4-EDA descriptors for the testing dataset with both the *EMD* and *KLD* distance metrics, an average of 0.2 seconds to match two HP4-EDAs.

VI.4.4 P4: Global and Local Feature (GLoFe) fit

VI.4.4.1 Adopted descriptors and overall strategy

The strategy adopted in method P4 is based on a direct approach. Depending on the track, a collection of local and global features have been calculated. For each feature vector f, a dissimilarity matrix d_f has been computed, while their weighted combination produced the total dissimilarity matrix. The pipeline of the adopted strategy is depicted in Figure VI.9.

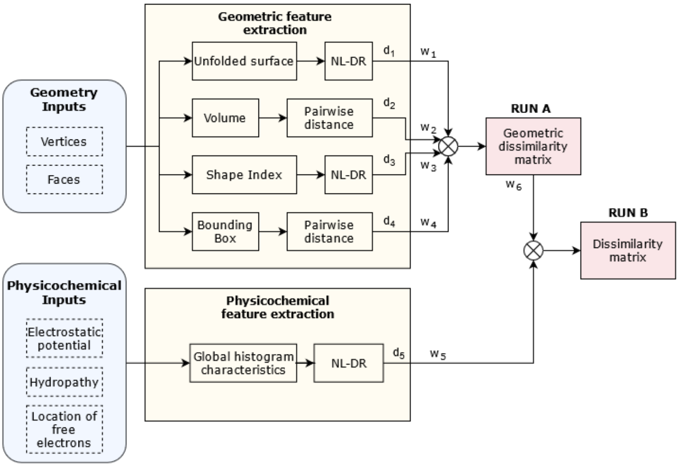


Figure VI.9: Processing pipeline of the strategy adopted in method P4. It includes extraction of multiple global and local shape descriptors, non-linear dimensionality reduction (NL-DR), calculation of pairwise distances, and fusion of the obtained dissimilarity matrices.

The features used for Task A have been 3D shape descriptors from surface unfolding, a shape index describing local curvature, the volume of each protein, and the size of an encompassing bounding box. Differently, the ones taken into account for Task B have been global histogram characteristics of the three provided physicochemical properties.

Using the Euclidean distance normalized by the standard deviation, we calculated the matrices d_f that indicate the pairwise distance between pairs of observations, for each geometric or physicochemical feature f. For some of them, we slightly modified the distance value by examining the inverse consistency of mapping. Specifically, for each protein structure j, we identified the first k-Nearest Neighbours, i.e. $k \in N(j)$, where N denotes the set of neighbours. For each k, we examined whether $j \in N(k)$. If true, the forward and inverse retrieval was consistent, thus the certainty in estimation of pairwise similarity between j and k was considered high. In this case, we increased by a factor, α, the dissimilarity value in $d(j,k)$. The values used in the experiments were $|N| = 5$ for both forward and inverse mapping and $\alpha = 0.3$. The final distance matrix d was constructed from a linear combination of the individual dissimilarity matrices, i.e. $d = \sum w_f \cdot d_f$. where w_f is a weight determining the contribution of each feature f. The weights have been empirically estimated by optimizing the classification performance on the training set.

The classification performance was assessed on the training set in order to allow the optimization of the several (hyper)parameters of the methodology. As evaluation criterion, the percentage of proteins for which the first (or correspondingly the second) closest neighbour belonged to the same class has been adopted.

VI.4.4.2 Task A

For Task A, four local and global geometric features have been extracted.

3D shape descriptors from surface unfolding In order to remove translation and rotation differences across the protein structures, for each data matrix $V^{(j)}$ associated with protein j, we performed Principal Component Analysis (PCA) on the mean centered data and replace them by their projection in the principal component space. This results in globally normalized surface data. Since the vertex coordinates belong to 2D surfaces lying in the 3D space, a dimensionality reduction technique to “unfold” the manifold and embed into a 2-dimensional space has been applied. For this purpose, we used Locality Preserving Projections (LPP) [18] due to the algorithm’s stability, high performance, and mainly its capability to preserve local (neighbourhood) structure. The embedded data $Y^{(j)}$ were calculated as $Y^{(j)} = V^{(j)} \cdot W^{(j)}$ where $W^{(j)}$ is the transformation matrix that maps the set of vertices of protein j from $\mathbb{R}^3$ to $\mathbb{R}^2$. Scores in $Y^{(j)}$ cannot be directly used for protein retrieval because of their high number and variable length across structures. Thus, we used as data representation the multi-variate kernel density estimate, $p_{Y^{(j)}} \in \mathbb{R}^{b_1 \times b_2}$, where b_1 and b_2 are the number of bins (common for all proteins) for the two columns of $Y^{(j)}$, respectively. Then, since the obtained kernel density maps were sometimes anti-symmetric, we augmented the whole dataset by horizontally and vertically flipping $p_{Y^{(j)}}$ resulting in 4 replicates for each protein. Finally, the 4 replicates of $p_{Y^{(j)}}$ for all proteins were linearized and concatenated in a big data matrix, in order to learn the manifold of different proteins and their conformations. The non-linear dimensionality reduction technique t-distributed Stochastic Neighbour Embedding (t-SNE) [19] was used to calculate 5 scores that model each protein structure in the lower dimensional space by retaining data similarity as much as possible. For indexing purposes, the distance of protein structure j to some other protein structure was defined as the smallest distance across the 4 replicates.

Shape index The third incorporated geometric feature is the shape index which was part of the first pre-processing phase of MaSIF [15], a network that combines geometric and physicochemical properties into a single descriptor. The shape index describes the shape of the surface around each vertex with respect to the local curvature, which is calculated in a neighbourhood of geodesic radius 12Å around it. The proposed neighbourhood size was chosen empirically. The shape index of v_i is defined according to

$$\frac{2}{\pi} \tan^{-1} \left(\frac{k_1 + k_2}{k_1 - k_2} \right)$$

where k_1, k_2 (with $k_1 \geq k_2$) are the principal curvature values in v_i's neighbourhood. High shape index values imply that v_i's neighbourhood is highly convex while lower values indicate that is highly concave.

Volume Since the volume of a protein does not significantly change when the protein obtains a different conformation, it has been also used as a feature. Although this feature helps to reduce some possible matches, it has low specificity because the range of volumes across different protein classes overlaps for many of them. Moreover, volume cannot be accurately calculated for the very few protein structures which contain holes.

Global scale In order to characterize global scale of the protein, we fitted a bounding box on the protein surface defined by 8 vertices in the 3D space. We used as global scale descriptor the 3 eigenvalues of the square matrix produced by multiplying the 8×1 vector by its transpose.

For Task A, three different runs adopting method P4 and generated as it follows by a linear combination of the individual dissimilarity matrices of four separate geometric descriptors with weights have been proposed. Specifically, w_1, w_2, w_3, and w_4 refer to unfolded surface, volume, shape index, and bounding box, respectively.

- **Run 1 (GLoFe):** generated by choosing $w_1 = 0.22$, $w_2 = 0.67$, $w_3 = 0.055$, $w_4 = 0.055$.
- **Run 2 (GLoFe):** generated by choosing $w_1 = 0.22$, $w_2 = 0.67$, $w_3 = 0.055$, $w_4 = 0.055$ (without using inverse consistency).
- **Run 3 (GLoFe):** generated by choosing $w_1 = 0.25$, $w_2 = 0.725$, $w_3 = 0$, $w_4 = 0.025$ (without using inverse consistency).

VI.4.4.3 Task B

For each of the three provided physicochemical properties, we extracted global histogram characteristics (first order statistics) that included mean intensity, standard deviation, mode of histogram (i.e. the most frequent intensity value), kurtosis, skewness, and energy. These six features for each of the three physicochemical properties provide global information on the distribution of gray-level intensities. Among these 18 features, the mode for the location of hydrogen bond donors and acceptors assumes the same value for all proteins and so it has been discarded. Differently, the remaining features has been merged into a 17-dimensional vector for each protein.

For Task B, three different runs adopting method P4 and generated as it follows by a linear combination of the geometric and physicochemical dissimilarity matrices with weights have been proposed. Specifically, w_5 refers to the physicochemical dissimilarity matrix and w_6 to the geometric dissimilarity matrix produced by the corresponding run of Task A.

- **Run 1 (GLoFe):** generated by choosing $w5 = 0.065$, $w6 = 0.935$ (geometric dissimilarity matrix produced by Run 1 in Task A).
- **Run 2 (GLoFe):** generated by choosing $w5 = 0.065$, $w6 = 0.935$ (geometric dissimilarity matrix produced by Run 2 in Task A).
- **Run 3 (GLoFe):** generated by choosing $w5 = 0.075$, $w6 = 0.925$ (geometric dissimilarity matrix produced by Run 3 in Task A).

VI.4.4.4 Computational aspects

The experiments on both tracks have been carried out using an AMD Ryzen 7 3700X 8-core Processor @3.59 GHz PC with 16 GB of RAM, except from the extraction of the surface unfolding and the physicochemical features which have been carried out using an Intel i5-6402P @2.80 GHz CPU with 8 GB of RAM. The software has been written in Matlab 2019b.

The total time required for obtaining the results for each task in the test set is indicated in Table VI.1. In addition, an exhaustive search procedure (requiring 54 minutes) was followed to optimize the weights for fusion of the different dissimilarity matrices based on the training set. Notice that the total inference cost (illustrated in Table VI.1) corresponds to the time aggregated due to sequential calculation of the different feature sets, whereas with a multi-threaded implementation it is reasonable to expect that the total computational time would be significantly decreased.

Table VI.1: Computational times required for Tasks A and B by method P4.

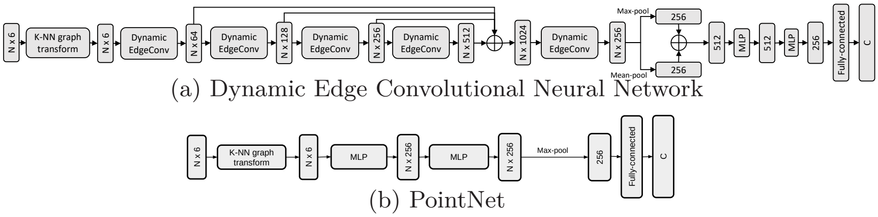
Task			**Time (mins)**
A	Surface unfolding	$n_S^{test} = 1,543$	986
		$n_S^{train} = 3,585$	2,290
		Augment & DR	83
	Volume		961
	Shape index		85
	Global scale		284
	Dissimilarity matrix calculation		0.005
	Total time		4,689
B	Physicochemical features		1
	Dissimilarity matrix calculation		0.001
	Total time		4,690

VI.4.5 P5: Message-Passing Graph Convolutional Neural Networks (MPGCNNs) and PointNet

VI.4.5.1 Adopted descriptors and overall strategy

For the meshes in each 3D model of a protein surface, in method P5 we first sampled 512 points on the surfaces of the meshes based on the area of the meshes. Because a sampled point might not be an original vertex in the 3D meshes, the original physical and chemical properties are not valid for newly sampled points. To generate physical and chemical properties for a sampled point p, trilinear interpolation has been performed from the properties of the three vertices forming the face that p is on. Then, to re-assign the topological structures for sampled points, each node has been connected with their k-Nearest Neighbours based on their original coordinates choosing $k = 16$.

In method P5, we adopted a deep learning strategy by exploiting the availability of protein class labels to optimize the representation of protein surfaces with and without textures. The chosen strategy is based on the use of graph neural networks (GNNs). GNNs are deep learning based methods that, unlike classic NNs, operate on a graph domain rather than on Euclidean domains. Remarkable advancements in NNs can be achieved by including in the network hidden layers that perform convolutions. In such a case, a neural network will be called convolutional neural network (CNN) or graph neural network (GNNs) in the specific considered case. In particular, we designed message-passing graph convolutional neural networks (MPGCNNs) with the Edge Convolution paradigm [41]. A visual description of the pipeline adopted in method P5 is depicted in Figure VI.10.



(a) Dynamic Edge Convolutional Neural Network

(b) PointNet

Figure VI.10: The summary of the graph neural networks employed in method P5 optimized over the classification of the training set. The depicted pipelines are used for the geometry with physicochemical properties tasks, where a graph has N nodes and each node has 6 initial features. For Task A, there are only 3 initial node features, which are the spatial coordinates of points. After training, the 256-dimension vector before the fully-connected layer is used for all the tasks.

Edge convolution In the framework of Task A, the initial node features are the coordinates of sampled points, while in the setting of Task B, the features are concatenated tuples of coordinates and interpolated physicochemical properties. Each protein surface is represented by a k-Nearest Neighbours graph generated in the pre-processing step with 512 vertices. The module that performs the graph message-passing function is the Edge Convolution (EdgeConv) layer [41]. In the EdgeConv layer, the information of a vertex i after layer l is calculated as $x_i^{l+1} = \max_{j \in N} h(x_i^l, x_j^l)$ where N is the neighbouring vertices of vertex i and

$$h(x_i^l, x_j^l) = ReLU(MLP(x_i^l \oplus x_j^l))$$

where ReLU is Rectified Linear Unit (in the implementation, LeakyReLU, a variant of ReLU, has been used), MLP is a standard multi-layer perceptron (MLP), $\oplus$ is the concatenation operator. In the implementation of method P5, we adopted a dynamic variant of EdgeConv instead of the standard EdgeConv described above. At each Dynamic EdgeConv layer, the each vertex's k-Nearest Neighbours is re-calculated in the feature space produced by the previous layer, before applying the standard EdgeConv operation. After the graph is recomputed, standard EdgeConv operation is performed. After the pre-processing phase, the vertex features first go through 4 layers of Dynamic EdgeConv. The dimensions of output features for each vertex after these first 4 layers are 64, 64, 128, and 256, respectively. Then, the outputs of these 4 layers are concatenated to become a 512-dimensional vector for each vertex. This 512-dimensional vector is then fed through another Dynamic EdgeConv layer, creating the output vector with 512 dimensions v. The feature vector v is pooled using the concatenation of the outputs of a max-pooling and a mean pooling layer to generate the first graph-level feature vector. This vector is passed through two MLP blocks with BatchNorm, Leaky-ReLU, and Dropout layers. The retrieval tasks can then be performed by exploiting the L2-distances between the output vectors.

PointNet Adopting the same data pre-processing procedure, we also implemented PointNet [30], a well-known graph-based learning strategy for 3D data. In this network architecture, the vertex features first pass through 2 message-passing modules. Each messages-passing module contains a MLP block that uses ReLU as activation function. The module captures spatial information between a node and its neighbours by performing subtractions between each pair of the center's position and its neighbour's position. After having performed calculation, the information of a vertex i after layer l is calculated as $x_i^{l+1} = \max_{j \in N} h(x_j^l, p_j, p_i)$ where N is the neighbouring vertices of vertex i and

$$h(x_j^l, p_j, p_i) = ReLU(MLP(x_j^l \oplus (p_j - p_i)))$$

where p_j is the position of the vertex j and p_i is the position of the vertex i. For a MLP block that the vertex features pass through, its output further goes through a ReLU function. After two MLP blocks, the feature vector is pooled using a single global max-pooling layer. Then, retrieval tasks can be performed by the same way as with the Dynamic EdgeConv strategy.

VI.4.5.2 Task A

For Task A, three different runs adopting method P5 and generated as it follows have been proposed.

- **Run 1 (EdgeConv):** generated by the Dynamic EdgeConv strategy where distances between output vectors are L2-distances between embeddings.
- **Run 2 (PointNet):** generated by the PointNet strategy but choosing the number of mesh surface sample points as 258 instead of 512.

- **Run 3 (Ensemble):** generated by taking the weighted average of embedding distances from the above Dynamic EdgeConv and PointNet embedding distances. Specifically, the distances from Dynamic EdgeConv are empirically weighted by 0.6, while those from PointNet are weighted by 0.4.

VI.4.5.3 Task B

For Task B, one run adopting method P5 and generated as it follows has been proposed.

- **Run 1 (EdgeConv):** generated by the Dynamic EdgeConv strategy where the concatenation of spatial coordinates and properties made up of initial vertex features. The distances between output vectors are L2-distances between embeddings.

VI.4.5.4 Computational aspects

All of the methods have been implemented in Python 3.8, using Pytorch [28] and Pytorch Geometric [12] libraries. The experiments have been carried out a machine with an Intel Core i7-8700K 6-core CPU Processor @3.70 GHz PC with 32 GB of RAM and an NVIDIA TITAN V with 12 GB of VRAM. The training and test set's embedding extraction used both the CPU and the GPU, whose time is represented in Table VI.2. The computation of distance matrix only used the CPU and it required approximately 15 minutes for Run 3 of Task A while just approximately 7 minutes for the other runs.

Table VI.2: The (approximated) training and extraction times of employed strategies in method P5.

Task	Strategy	Training	Test Set Extr.
A	PointNet	720 mins	0.5 mins
	Dyn. EdgeConv	1,100 mins	3 mins
B	Dyn. EdgeConv	1,100 mins	3 mins

VI.5 Comparative analysis

The performances of each run presented in Section VI.4 are here quantitatively evaluated on the basis of the measures described in Section VI.3.3. We remind the reader that: Task A refers to the mere use of geometry, while Task B includes both geometry and physicochemical properties; for any run, the method name and its specific settings are given in Section VI.4. The performance measures are presented for both the PDB and BLAST classifications detailed in Section VI.3.2.

An additional analysis, reported in the supplementary material in VI.C, is performed on a 3-level BLAST-based classification: we introduce a further relaxation by merging the classes containing the same proteins or their isoforms with structures of proteins that have a significant sequence similarity, according to what introduced in Section VI.3.2. In this case, the BLAST-based decomposition of level 2 consists of 25 communities.

VI.5.1 Retrieval evaluation measures

Table VI.3 summarizes the retrieval performances of all the runs submitted for evaluation with respect to the PDB classification. More specifically, the table provides the following information: the Nearest Neighbour (NN), the First Tier (1T), the Second Tier (2T), the e-measure (eM), the Discounted Cumulated Gain (DCG), and the mean Average Precision (mAP). For each task, method and retrieval measure, the best performance is highlighted in bold; for each task and retrieval measure, the best performance among all methods is highlighted in red. All values are averaged for all queries. Many methods achieve great or excellent performances. For instance:

- For Task A, 10 out of 15 runs have an NN value above 0.9, i.e. their classification rate is above 90%.
- For Task B, 11 out of 13 runs have the NN value above 0.9.

The same methods have mAP and DCG values above, respectively, 0.6 and 0.8. Precision-Recall plots are provided in Figure VI.11.

Table VI.3: Summary of results by method and property type (only geometry vs. geometry and physicochemical properties) for the PDB classification. Here: NN = Nearest Neighbour, 1T = First Tier, 2T = Second Tier, eM = e-Measure, DCG = Discounted Cumulated Gain, mAP = mean Average Precision. For each task and for each measure, the best value for each method is in bold. The best among them is highlighted in red.

	Geometry							Geometry and chemistry						
	method	NN	1T	2T	eM	DCG	mAP	method	NN	1T	2T	eM	DCG	mAP
P1	run 1	0.837	0.605	0.778	0.504	0.845	0.675	run 1	0.982	0.873	0.951	0.685	0.971	0.921
	run 2	**0.947**	**0.815**	**0.940**	**0.654**	**0.947**	**0.877**	run 2	**0.989**	**0.922**	**0.979**	**0.714**	**0.985**	**0.958**
	run 3	0.927	0.729	0.884	0.597	0.921	0.806	run 3	0.585	0.364	0.518	0.306	0.670	0.414
P2	run 1	0.914	0.735	0.888	0.607	0.916	0.802	run 1	0.951	0.815	**0.938**	0.653	0.949	0.874
	run 2	0.894	0.723	0.880	0.605	0.908	0.791	run 2	0.947	0.800	0.927	0.649	0.942	**0.895**
	run 3	**0.924**	**0.748**	**0.889**	**0.613**	**0.921**	**0.813**	run 3	**0.979**	**0.839**	0.937	**0.665**	**0.962**	0.858
P3	run 1	0.920	0.683	0.836	0.562	0.897	0.756	run 1	0.902	**0.696**	**0.848**	**0.572**	**0.893**	**0.764**
	run 2	**0.930**	**0.711**	**0.858**	**0.586**	**0.911**	**0.782**	run 2	**0.923**	0.592	0.720	0.486	0.847	0.663
	run 3	0.922	0.692	0.846	0.572	0.903	0.767	run 3	0.903	0.683	0.820	0.560	0.887	0.757
P4	run 1	**0.927**	**0.593**	**0.716**	**0.493**	**0.865**	**0.684**	run 1	**0.941**	**0.684**	**0.791**	**0.550**	**0.901**	**0.761**
	run 2	**0.927**	0.586	0.705	0.487	0.862	0.675	run 2	**0.941**	0.676	0.785	0.544	0.899	0.755
	run 3	0.907	0.549	0.672	0.453	0.840	0.634	run 3	0.933	0.653	0.758	0.529	0.888	0.730
P5	run 1	**0.755**	**0.537**	**0.734**	**0.468**	**0.806**	**0.541**	run 1	**0.718**	**0.532**	**0.731**	**0.466**	**0.798**	**0.751**
	run 2	0.437	0.300	0.477	0.263	0.633	0.485							
	run 3	0.713	0.494	0.699	0.435	0.783	0.452							

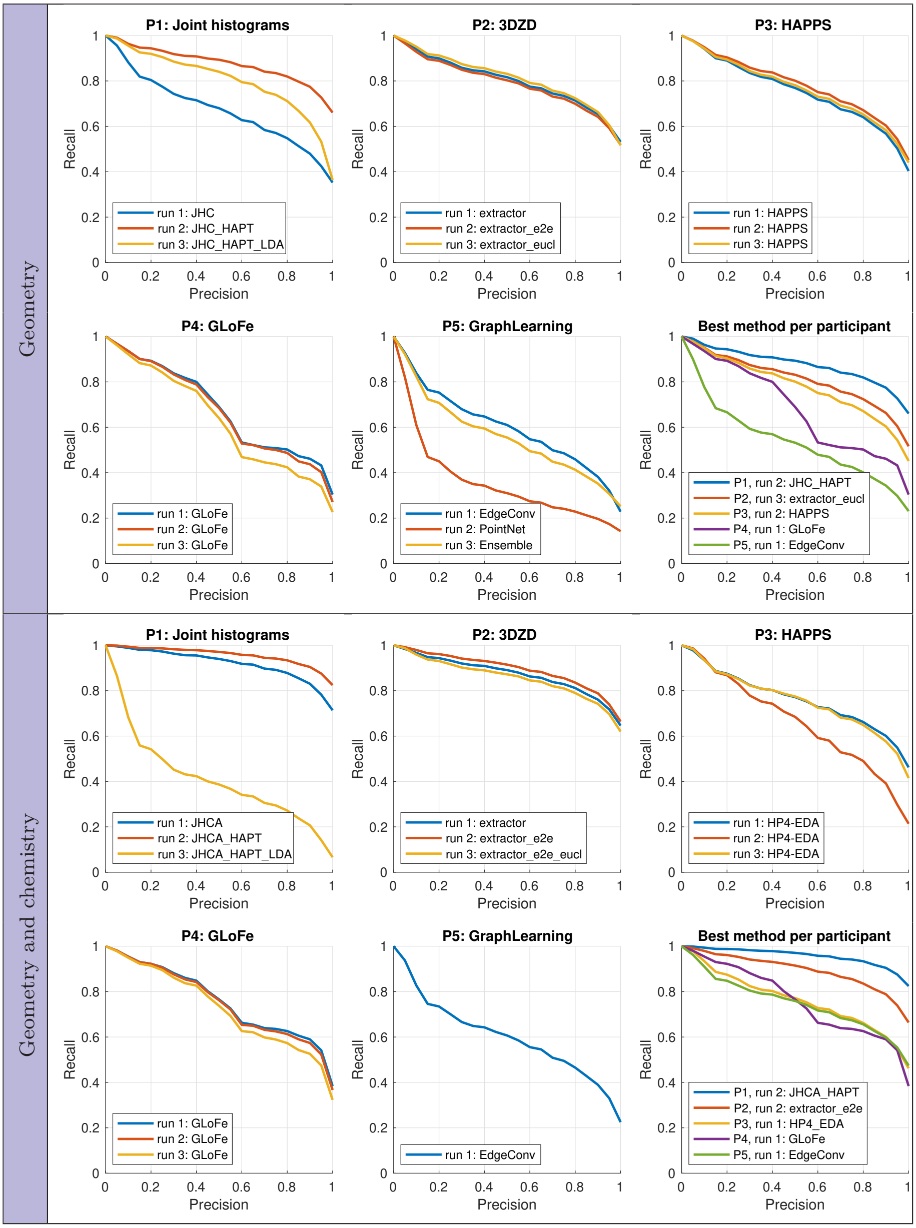


Figure VI.11: Precision-recall curves for Task A (geometry only) and Task B (geometry and physicochemical properties), with respect to the PDB classification.

For sake of conciseness, we do not list the analogous values of Table VI.3 and plots in Figure VI.11 for the BLAST classification, rather we focus on multi-level indicators, such as the ADR values and the NDCG plots. Interpreting the elements of a query queue in terms *identity or isoform*, *highly related*, *similar* and *dissimilar* surfaces with respect to the BLAST classification, Table VI.4 reports the average dynamic recall (ADR) values for the runs of all methods. All the ADR scores, which range from 0 (worst case) to 1 (ideal performance), are averaged over all the models in the dataset.

Table VI.4: Summary of average dynamic recalls (ADRs) for the 4-level BLAST classification. For each task, the best ADR for each method is in bold. The best among them is highlighted in red.

	Geometry						Geometry and chemistry				
	P1	P2	P3	P4	P5		P1	P2	P3	P4	P5
run 1	0.640	**0.700**	0.697	**0.681**	0.631	run 1	**0.809**	**0.770**	0.756	**0.719**	**0.733**
run 2	0.721	0.688	**0.706**	0.676	0.543	run 2	0.755	0.738	**0.804**	0.715	-
run 3	**0.729**	0.670	0.699	0.660	**0.641**	run 3	0.635	**0.770**	0.751	0.704	-

A more comprehensive analysis of the retrieval queue with respect to the BLAST classification is provided by the normalized discounted cumulative gain plots in Figure VI.12. The NDCG measure is represented as a function of the rank p. The NDCG values for all queries are averaged to obtain a measure of the average performance for each submitted run. Remind that, for an ideal run, it would be NDCG $\equiv 1$. The NDCG measure takes BLAST classification performances into larger account than PDB one, as all surfaces corresponding to the same PDB code are the same protein for the BLAST classification.

In VI.C, we include the same multi-level indicators, namely ADR and NDCG plots, for the 3-level BLAST classification. Since this classification aggregates communities that are extremely similar and highly related, the ADR scores slightly increase but the overall relationships between the different methods and runs are confirmed.

VI.5.2 Classification performance measures

Table VI.5 summarizes the classification performances for both tasks A and B, when considering the PDB classification; the confusion matrices originating such values are shown, for the sake of completeness, in Figures VI.13 and VI.14. More precisely, the table contains the following information: True Positive Rate (TPR), True Negative Rate (TNR), Positive Predicted Values (PPV), Negative Predicted Values (NPV), Accuracy (ACC) and F_1 score (F1). One fact we can immediately note is that, maybe not surprisingly, methods that showed robustness when evaluated with the retrieval measures yet exhibit strong performances when tested with classification measures. However, we can additionally notice that:

- All methods have TNR higher than TPR, making them more reliable in correctly finding true negatives rather than true positives.

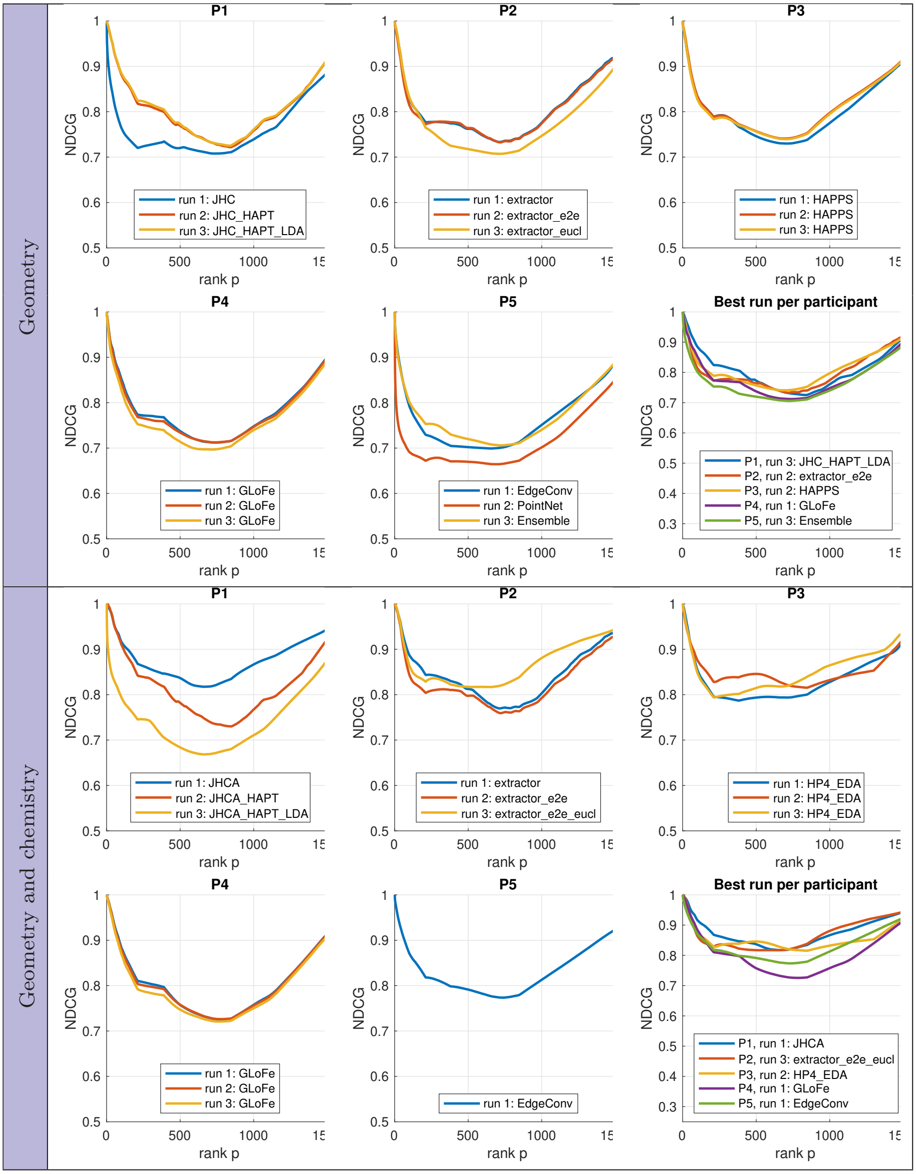


Figure VI.12: Normalized discounted cumulated gain (NDCG) for Task A (geometry only) and Task B (geometry and physicochemical properties), with respect to the 4-level BLAST classification.

- All methods have NPV higher than TPV: it is more likely for these methods to be right when reporting a negative rather than a positive.

- All methods have accuracy over 97.5%, although they are not equally “accurate" in finding true/false positives/negatives. This a sign – rather obvious from the nature of the problem – that positives and negatives are not in equal proportion; methods showing a greater difference between TPR and TNR (and between PPV and NPV) are more inclined to privilege the negatives (the predominant class) at the expense of the positives (the minor class). In our context, accuracy is therefore an overoptimistic estimation.

- F_1 score provides a “better” metric than the accuracy, in the sense that it suffers more from imbalance.

Enriching the protein SES triangulation with physicochemical properties does not always lead to an improvement. For example, in the third run by P1 it dramatically decreases the performances. On the other hand, run 2 from the same method shows a marked improvement to deal with (true) positives.

Table VI.6 summarizes the classification performance for both tasks A and B, when considering the (4-level) BLAST classification; the confusion matrices originating such values are shown, for the sake of completeness, in Figures VI.15 and VI.16. As expected, a decrease in the number of classes leads to an improvement of the classification measures. However, it is also worth noting that this improvement is not the same in all methods: by comparing Tables VI.5 and VI.6, one can indeed note changes in the best run per method (and in the best overall run).

The supplementary material, reported in VI.C, includes classification measures (see Table VI.7) and the corresponding confusion matrices (see Figures VI.17 and VI.18) in the case of a 3-level BLAST classification; one can notice that this latter leads to the same considerations as in the 4-level BLAST classification.

VI.5.3 Discussion

The methods that participated in this SHREC contest are representative of various types of approaches to the 3D object retrieval problem, ranging from purely feature-based engineered methods, mainly based on features represented with histograms (P1, P3, and P4), to the combination of features and dimensional reduction techniques (P1 Run 3), to deep neural networks (P2) and transfer learning from deep graph convolutional networks (P5).

On the one hand, the retrieval performances are positive for all methods, in either the PDB or BLAST classifications. On the other hand, the NDCG and ADR measures are specifically designed for interpreting a multi-level dataset classification as in this case, and thus offer a complementary evaluation of the classical retrieval measures (e.g., NN, FT, ST, precision-recall plots, etc.) and classification measures (TPR, TNR, PPV, NPV, ACC, confusion matrices, etc.). These performance indicators show that the highest ADR scores vary from 0.729 (geometric) to 0.809 (geometry and pysico-chemical properties) being 1 the best

Table VI.5: Summary of statistical measures by method and property type (only geometry vs. geometry and physicochemical properties) for the PDB-based community decomposition. Here: TPR = True Positive Rate, TNR = True Negative Rate, PPV = Positive Predictive Value, NPV = Negative Predictive Value, ACC = ACCuracy, F1 = F_1 score. For each task and for each measure, the best value for each method is in bold. The best among them is highlighted in red.

	Geometry							Geometry and physicochemical properties						
	method	TPR	TNR	PPV	NPV	ACC	F1	method	TPR	TNR	PPV	NPV	ACC	F1
P1	run 1	0.8373	0.9967	0.8401	0.9973	0.9941	0.8354	run 1	0.9825	0.9997	0.9860	0.9997	0.9994	0.9832
	run 2	**0.9475**	**0.9991**	**0.9489**	**0.9992**	**0.9983**	**0.9467**	run 2	**0.9890**	**0.9999**	**0.9921**	**0.9999**	**0.9998**	**0.9893**
	run 3	0.9274	0.9990	0.9304	0.9988	0.9974	0.9239	run 3	0.5839	0.9885	0.6086	0.9912	0.9807	0.5727
P2	run 1	0.9145	0.9979	0.9159	0.9984	0.9965	0.9119	run 1	0.9514	0.9989	0.9525	0.9992	0.9981	0.9504
	run 2	0.8944	0.9975	0.8977	0.9976	0.9954	0.8931	run 2	0.9469	0.9989	0.9470	0.9991	0.9981	0.9460
	run 3	**0.9242**	**0.9983**	**0.9253**	**0.9985**	**0.9969**	**0.9222**	run 3	**0.9793**	**0.9995**	**0.9819**	**0.9996**	**0.9991**	**0.9791**
P3	run 1	0.9196	**0.9985**	0.9205	0.9986	**0.9971**	0.9169	run 1	0.9015	0.9977	0.9037	0.9979	0.9957	0.9007
	run 2	**0.930**0	0.9982	**0.9333**	**0.9988**	**0.9971**	**0.9276**	run 2	**0.9216**	**0.9985**	**0.9238**	**0.9987**	**0.9974**	**0.9207**
	run 3	0.9222	0.9982	0.9258	0.9986	0.9969	0.9199	run 3	0.9015	0.9979	0.9005	0.9985	0.9966	0.8987
P4	run 1	**0.9274**	**0.9983**	**0.9284**	**0.9987**	**0.9971**	**0.9262**	run 1	**0.9410**	0.9987	**0.9424**	**0.9990**	**0.9978**	0.9406
	run 2	0.9268	**0.9983**	0.9277	**0.9987**	**0.9971**	0.9252	run 2	**0.9410**	**0.9988**	0.9422	**0.9990**	**0.9978**	**0.9409**
	run 3	0.9067	0.9979	0.9059	0.9983	0.9963	0.9046	run 3	0.9326	0.9984	0.9336	0.9988	0.9974	0.9323
P5	run 1	**0.7537**	**0.9944**	**0.7539**	**0.9943**	**0.9892**	**0.7507**	run 1	**0.7187**	**0.9937**	**0.7215**	**0.9940**	**0.9886**	**0.7160**
	run 2	0.4362	0.9870	0.4412	0.9873	0.9754	0.4328							
	run 3	0.7123	0.9927	0.7109	0.9930	0.9868	0.7044							

Table VI.6: Summary of statistical measures by method and property type (only geometry vs. geometry and physicochemical properties) for the BLAST-based community decomposition of level 3. Here: TPR = True Positive Rate, TNR = True Negative Rate, PPV = Positive Predictive Value, NPV = Negative Predictive Value, ACC = ACCuracy, F1 = F_1 score. For each task and for each measure, the best value for each method is in bold. The best among them is highlighted in red.

	Geometry							Geometry and physicochemical properties						
	method	TPR	TNR	PPV	NPV	ACC	F1	method	TPR	TNR	PPV	NPV	ACC	F1
P1	run 1	0.9086	0.9949	0.9082	0.9959	0.9917	0.9069	run 1	0.9961	0.9998	0.9962	**1.0000**	0.9997	0.9960
	run 2	**0.9890**	**0.9996**	**0.9891**	0.9996	**0.9993**	**0.9888**	run 2	**0.9981**	**1.0000**	**0.9981**	**1.0000**	**1.0000**	**0.9980**
	run 3	0.9844	**0.9996**	0.9869	**0.9997**	**0.9993**	0.9840	run 3	0.8529	0.9904	0.8562	0.9931	0.9854	0.8452
P2	run 1	0.9760	**0.9992**	0.9766	**0.9993**	**0.9985**	0.9758	run 1	0.9929	0.9997	0.9930	0.9999	0.9996	0.9928
	run 2	0.9728	0.9991	0.9746	0.9991	0.9983	0.9723	run 2	0.9909	0.9998	0.9909	0.9999	0.9997	0.9908
	run 3	**0.9767**	0.9985	**0.9770**	0.9989	0.9977	**0.9764**	run 3	**0.9987**	**0.9999**	**0.9987**	**1.0000**	**0.9999**	**0.9987**
P3	run 1	0.9689	0.9981	0.9690	0.9985	0.9970	0.9680	run 1	**0.9942**	0.9996	**0.9943**	**0.9999**	0.9995	**0.9942**
	run 2	**0.9747**	0.9980	**0.9754**	**0.9989**	0.9972	**0.9740**	run 2	0.9916	**0.9997**	0.9921	0.9998	**0.9996**	0.9916
	run 3	0.9721	**0.9987**	0.9738	0.9985	**0.9976**	0.9716	run 3	0.9806	0.9981	0.9809	0.9994	0.9980	0.9803
P4	run 1	**0.9799**	**0.9987**	**0.9805**	**0.9991**	**0.9980**	**0.9798**	run 1	**0.9903**	**0.9995**	**0.9909**	**0.9996**	**0.9992**	**0.9904**
	run 2	0.9793	**0.9987**	0.9801	0.9990	0.9979	0.9791	run 2	**0.9903**	**0.9995**	0.9908	**0.9996**	0.9991	**0.9904**
	run 3	0.9734	0.9984	0.9738	0.9987	0.9974	0.9732	run 3	0.9870	0.9993	0.9876	0.9994	0.9988	0.9871
P5	run 1	**0.9209**	**0.9953**	**0.9209**	**0.9958**	**0.9923**	**0.9197**	run 1	**0.9501**	**0.9975**	**0.9506**	**0.9988**	**0.9968**	**0.9491**
	run 2	0.7168	0.9852	0.7201	0.9853	0.9734	0.7156							
	run 3	0.9047	0.9944	0.9031	0.9953	0.9910	0.9019							

possible value for the ADR: this confirms that these approaches are good but not optimal. Similarly, the highest possible area under a NDCG curve equals 1, while the best scores in this contribution are around 0.9 (for the Task B).

In this SHREC contest, a training dataset was explicitly provided having with the ground truth based on the PDB classification. The surface distribution in the classes mirrored the distribution of classes in the test set, see Figure VI.3;

the number of conformations per PDB ranges from 2 to 160. This highlights one of the difficulties that learning methods have faced, namely the presence of classes of very heterogeneous size, which makes prediction very difficult. The severity of the PDB classification is then mitigated by the BLAST one, but this classification has been used only for the interpretation of the results and not previously provided to the participants.

A further difficulty for learning methods is the dataset design choice of using different proteins (and their conformations) between training set and test set. This was done to investigate the ability of 3D retrieval approaches to reason about and predict the conformations of a protein, even if not yet “seen" by the training system. This probably motivates that the best overall performance for the PDB classification was obtained by a technique based on engineered features. This fact is further confirmed by the lower prediction ability of the same descriptor when combined with a dimensional reduction technique as demonstrated by the method P1 (run 3), which show a particular decrease when the geometry is enriched with physicochemical properties. Conversely, when we consider the BLAST classification, i.e. proteins that share fairly long amino acids sequences are considered similar altogether their conformations, we see that learning-based methods improve their performance proportionally more than direct methods, such as the P1 (run 3) method. In our view, this reflects the fact that similarities between sequences are reflected in similarities of 3D structure, and with this classification comes greater homogeneity between the features of the “extended” classes.

Additional considerations can be derived from the geometric-only and mixed geometry and physicochemical properties comparison. Not surprisingly, we notice an improvement in the performance of the various approaches when switching from runs purely geometry-driven (Task A) to runs that consider both geometry and physicochemical properties (Task B). Nevertheless, the direct comparison between the proposed runs is not always possible because for some participants the geometric method may vary between the 2 tasks. We notice that the most widely adopted solution is the introduction of a histogram for the physicochemical properties, that is then used as an additional feature vector whose outcome is combined with the dissimilarity scores given by the geometric description. Furthermore, from the experiments available to us, we note that the most significant improvements are seen for methods that are based on learning. This is particularly reflected in the methods P2 and P5. This suggests that physicochemical properties play an important role in the characterisation of a protein but, perhaps, more research is still needed to deeply understand their role and how best to integrate them into engineered descriptors; for instance, considering a joint description as currently proposed in P1, that adopts joint 3D histograms.

VI.6 Concluding remarks

In this paper, we have provided a detailed analysis and evaluation of state-of-the-art retrieval and classification algorithms dealing with protein similarity assessment based on molecular surfaces, which we believe deserve attention from the research community. The introduction of physicochemical properties into the benchmark, represents an element of originality in the available benchmarks for structural biology and provides a more complete representation of the protein. To enable the participation of learning-based methods, both a training and a test set were provided for this benchmark dataset. Moreover, we are aware that in some of the PDB codes we used, the underlying structures may correspond to mutations of the same protein, or be isoform, or share a common fold; for this reason, we performed a multi-level performance analysis, comparing the performance of the proposed methods to both a classification made according to the protein PDB code and an aggregation between proteins made by using BLASTP.

Beyond the extensive analysis that has been carried out throughout the paper, we hope that the experimental results presented here may offer interesting hints for further investigation. For instance, a better and more informed definition of similarity can be preliminary to a better and more effective definition of the complementarity between binding biomolecules.

The benchmark, as well as the dissimilarity matrices that originated the results described in Section VI.5 and in the appendices, are available at https://github.com/rea1991/SHREC2021.

Acknowledgements. The track organisers thank Dr. Michela Spagnuolo and Dr. Davide Boscaini for the fruitful discussions. Special thanks go to Ms. Daniela Bejan, for her help in using the software PyMol.

This project is co-funded by the project “TEACUP: Metodi e TEcniche innovative per lo sviluppo di librerie per la modellazione, l’Analisi e il confronto CompUtazionale di Proteine", POR FSE, Programma Operativo Regione Liguria 2014-2020, No RLOF18ASSRIC/68/1. The CNR-IMATI research is partially developed in the activities DIT.AD021.080.001 and DIT.AD009.091.001. This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215, and by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number DS2020-42-01.

References

[1] Baeza-Yates, R. A. and Ribeiro-Neto, B. *Modern Information Retrieval.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[2] Berman, H. M. et al. “The Protein Data Bank”. In: *Nucleic Acids Research* vol. 28, no. 1 (Jan. 2000), pp. 235–242.

[3] Biasotti, S. et al. “Tracking the evolution of rainfall precipitation fields using persistent maxima”. In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. 2016, pp. 29–37.

[4] Biasotti, S. et al. *Mathematical Tools for Shape Analysis and Description*. Vol. 6. Williston, VT, USA: Morgan & Claypool, 2014, pp. 1–138.

[5] Camacho, C. et al. “BLAST+: architecture and applications”. In: *BMC Bioinformatics* vol. 10 (2009), p. 421.

[6] Canterakis, N. “3D Zernike moments and Zernike affine invariants for 3D image analysis and recognition”. In: *In 11th Scandinavian Conf. on Image Analysis*. Citeseer. 1999, pp. 85–93.

[7] Cignoni, P. et al. “MeshLab: an Open-Source Mesh Processing Tool”. In: *Eurographics Italian Chapter Conference*. Ed. by Scarano, V., Chiara, R. D., and Erra, U. The Eurographics Association, 2008, pp. 129–136.

[8] Connolly, M. L. “Analytical molecular surface calculation”. In: *Journal of Applied Crystallography* vol. 16, no. 5 (1983), pp. 548–558.

[9] Decherchi, S. and Rocchia, W. “A general and Robust Ray-Casting-Based Algorithm for Triangulating Surfaces at the Nanoscale”. In: *PLOS ONE* vol. 8, no. 4 (Apr. 2013), pp. 1–15.

[10] Decherchi, S. et al. “NanoShaper-VMD interface: computing and visualizing surfaces, pockets and channels in molecular systems”. In: *Bioinform.* vol. 35, no. 7 (2019), pp. 1241–1243.

[11] Esquivel-Rodriguez, J. et al. “Navigating 3D electron microscopy maps with EM-SURFER”. In: *BMC bioinformatics* vol. 16, no. 1 (2015), pp. 1–9.

[12] Fey, M. and Lenssen, J. E. “Fast graph representation learning with PyTorch Geometric”. In: *arXiv preprint arXiv:1903.02428* (2019).

[13] Fisher, R. A. “The use of multiple measurements in taxonomic problems”. In: *Annals of eugenics* vol. 7, no. 2 (1936), pp. 179–188.

[14] Fortunato, S. “Community detection in graphs”. In: *Physics reports* vol. 486, no. 3-5 (2010), pp. 75–174.

[15] Gainza, P. et al. “Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning”. In: *Nature Methods* vol. 17, no. 2 (2020), pp. 184–192.

[16] Giachetti, A. and Lovato, C. “Radial symmetry detection and shape characterization with the multiscale area projection transform”. In: *Computer Graphics Forum*. Vol. 31.5. Wiley Online Library. 2012, pp. 1669–1678.

[17] Giachetti, A. et al. “SHREC 2020 Track: River Gravel Characterization”. In: *Eurographics Workshop on 3D Object Retrieval*. Ed. by Schreck, T. et al. The Eurographics Association, 2020.

[18] He, X. and Niyogi, P. “Locality preserving projections”. In: *Advances in neural information processing systems* vol. 16, no. 16 (2004), pp. 153–160.

[19] Hinton, G. and Roweis, S. T. “Stochastic neighbor embedding”. In: *NIPS*. Vol. 15. Citeseer. 2002, pp. 833–840.

[20] Jurrus, E. et al. “Improvements to the APBS biomolecular solvation software suite”. In: *Protein Science* vol. 27, no. 1 (2018), pp. 112–128.

[21] Kortemme, T., Morozov, A. V., and Baker, D. “An orientation-dependent hydrogen bonding potential improves prediction of specificity and structure for proteins and protein–protein complexes”. In: *Journal of molecular biology* vol. 326, no. 4 (2003), pp. 1239–1259.

[22] Kuhn, M. and Johnson, K. *Applied Predictive Modeling*. Springer New York, 2018.

[23] Kyte, J. and Doolittle, R. F. “A simple method for displaying the hydropathic character of a protein”. In: *Journal of molecular biology* vol. 157, no. 1 (1982), pp. 105–132.

[24] Langenfeld, F. et al. “Protein Shape Retrieval Contest”. In: *Eurographics Workshop on 3D Object Retrieval*. Ed. by Biasotti, S., Lavoué, G., and Veltkamp, R. The Eurographics Association, 2019.

[25] Langenfeld, F. et al. “SHREC 2018 - Protein Shape Retrieval”. In: *Proceedings of the 11th Eurographics Workshop on 3D Object Retrieval*. 2018, pp. 53–61.

[26] Langenfeld, F. et al. “SHREC 2020: Multi-domain protein shape retrieval challenge”. In: *Computers & Graphics* vol. 91 (2020), pp. 189–198.

[27] Otu, E. et al. “Nonrigid 3D Shape Retrieval with HAPPS: A Novel Hybrid Augmented Point Pair Signature”. In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2019, pp. 662–668.

[28] Paszke, A. et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *arXiv preprint arXiv:1912.01703* (2019).

[29] Puzicha, J. et al. “Empirical evaluation of dissimilarity measures for color and texture”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, pp. 1165–1172.

[30] Qi, C. R. et al. “Pointnet: Deep learning on point sets for 3D classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[31] Richards, F. M. “Areas, volumes, packing, and protein structure”. In: *Annual Review of Biophysics and Bioengineering* vol. 6 (1977), pp. 151–176.

[32] Rijsbergen, C. J. V. *Information Retrieval*. 2nd. Newton, MA, USA: Butterworth-Heinemann, 1979.

[33] Roberts, K. et al. *Molecular biology of the cell*. New York: Garland Science, 2002.

[34] Rocchia, W., Alexov, E., and Honig, B. “Extending the Applicability of the Nonlinear Poisson-Boltzmann Equation: Multiple Dielectric Constants and Multivalent Ions”. In: *The Journal of Physical Chemistry B* vol. 105, no. 28 (2001), pp. 6507–6514.

[35] Rost, B. “Twilight zone of protein sequence alignments”. In: *Protein Engineering, Design and Selection* vol. 12, no. 2 (1999), pp. 85–94.

[36] Sael, L. et al. “Fast protein tertiary structure retrieval based on global surface shape similarity”. In: *Proteins: Structure, Function, and Bioinformatics* vol. 72, no. 4 (2008), pp. 1259–1273.

[37] Shilane, P. et al. “The Princeton Shape Benchmark”. In: *Shape Modeling International.* June 2004.

[38] Song, N. et al. “Protein Shape Retrieval: SHREC’17 Track”. In: *Proceedings of the Workshop on 3D Object Retrieval.* 3Dor ’17. Lyon, France: Eurographics Association, 2017, pp. 67–74.

[39] Veltkamp, R. C. et al. *SHREC2006: 3D Shape Retrieval Contest.* Tech. rep. UU-CS-2006-030. Utrecht University, 2006.

[40] Wang, L. et al. “Using DelPhi capabilities to mimic protein’s conformational reorganization with amino acid specific dielectric constants”. In: *Communications in Computational Physics* vol. 13, no. 1 (2013), pp. 13–30.

[41] Wang, Y. et al. “Dynamic graph CNN for learning on point clouds”. In: *Acm Transactions On Graphics (tog)* vol. 38.5 (2019), pp. 1–12.

[42] Zhou, Q.-Y., Park, J., and Koltun, V. “Open3D: A modern library for 3D data processing”. In: *arXiv preprint arXiv:1801.09847* (2018).

Appendix VI.A Confusion matrices (PDB-based community decomposition)

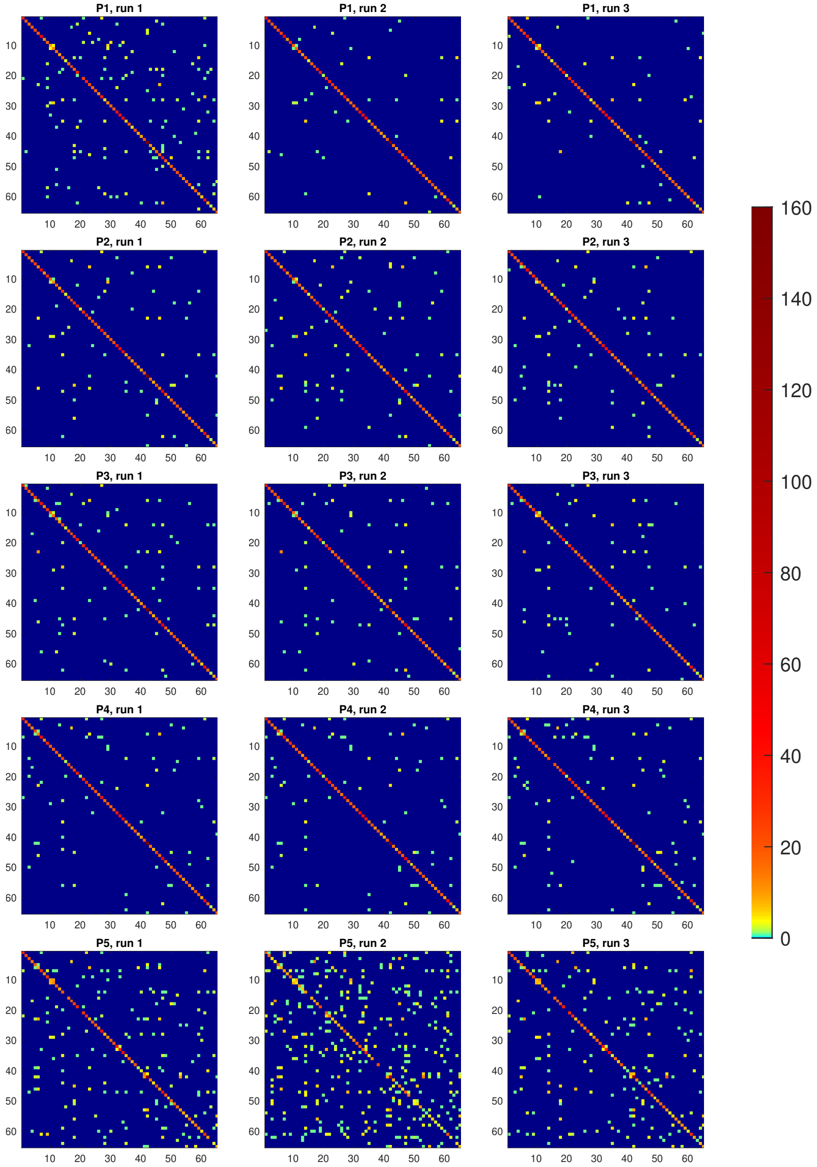


Figure VI.13: Confusion matrices for task A (geometry only), with respect to the PDB-based community decomposition.

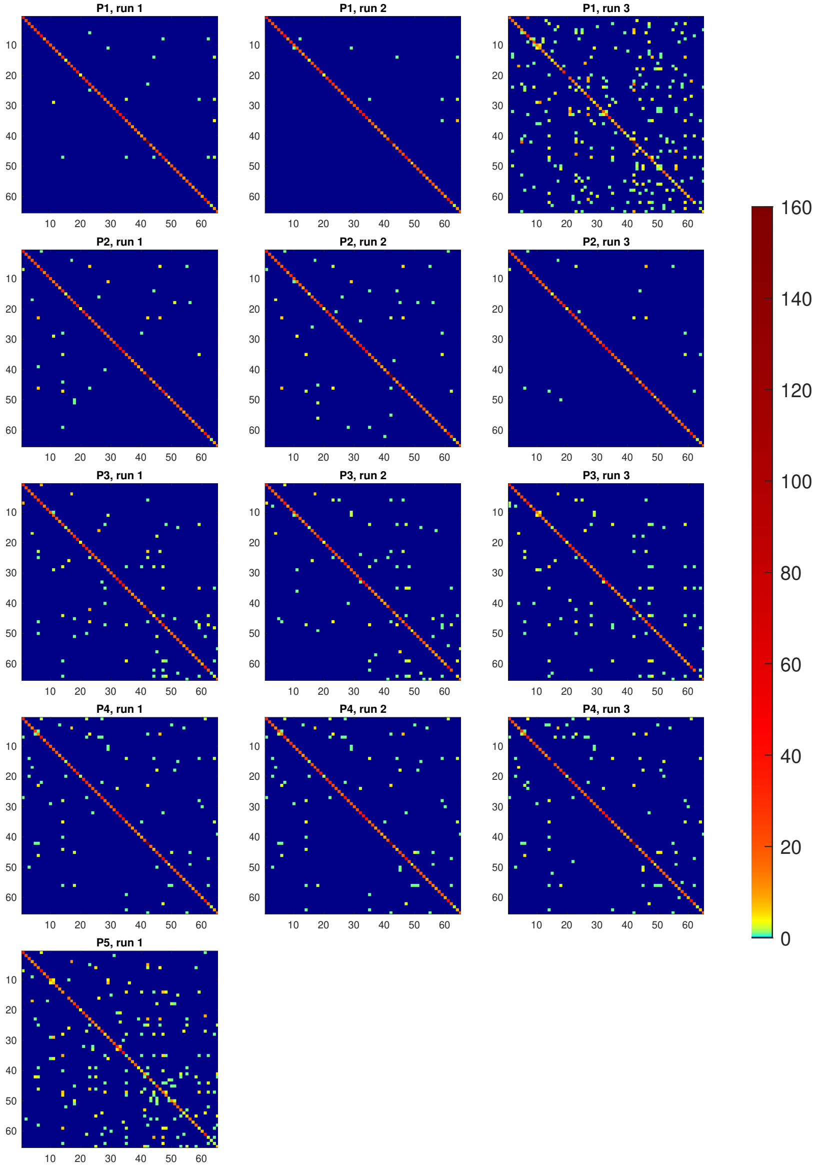


Figure VI.14: Confusion matrices for task B (geometry and physicochemical properties), with respect to the PDB-based community decomposition.

Appendix VI.B Confusion matrices (BLAST-based community decomposition of level 3)

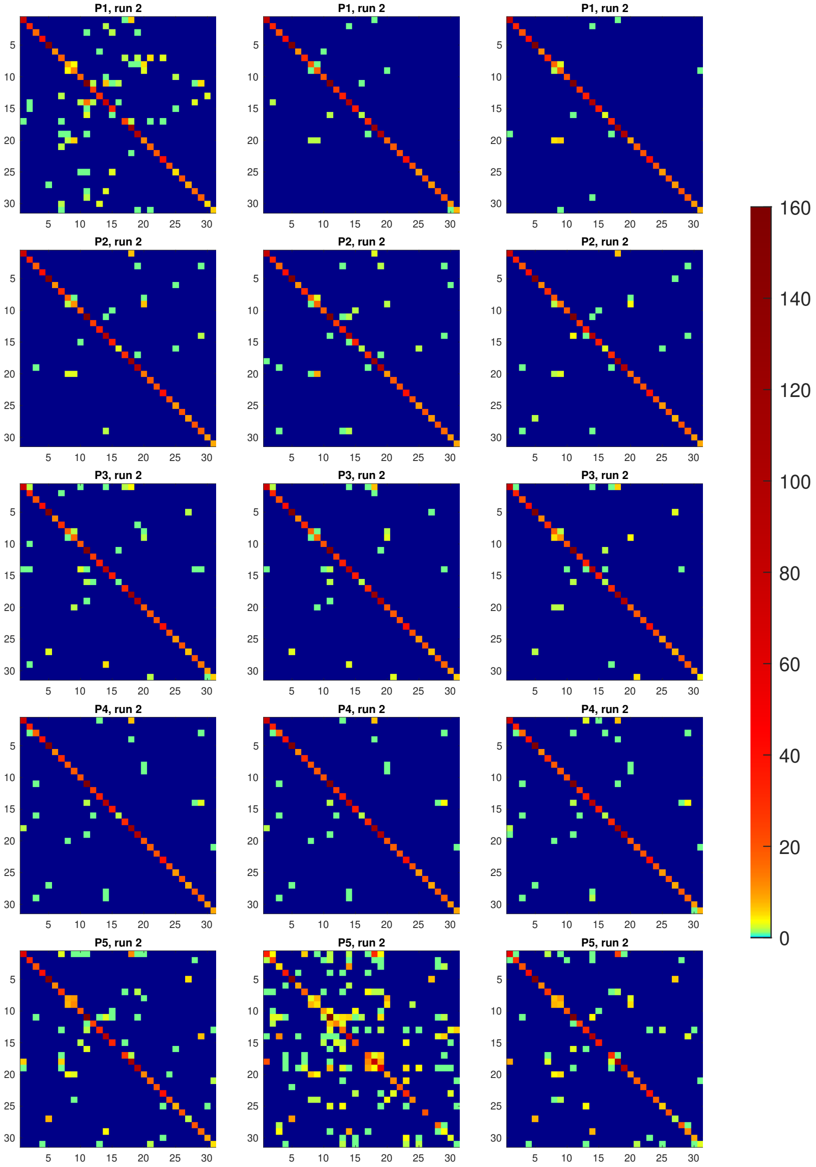


Figure VI.15: Confusion matrices for task A (geometry only), with respect to the BLAST-based community decomposition of level 3.

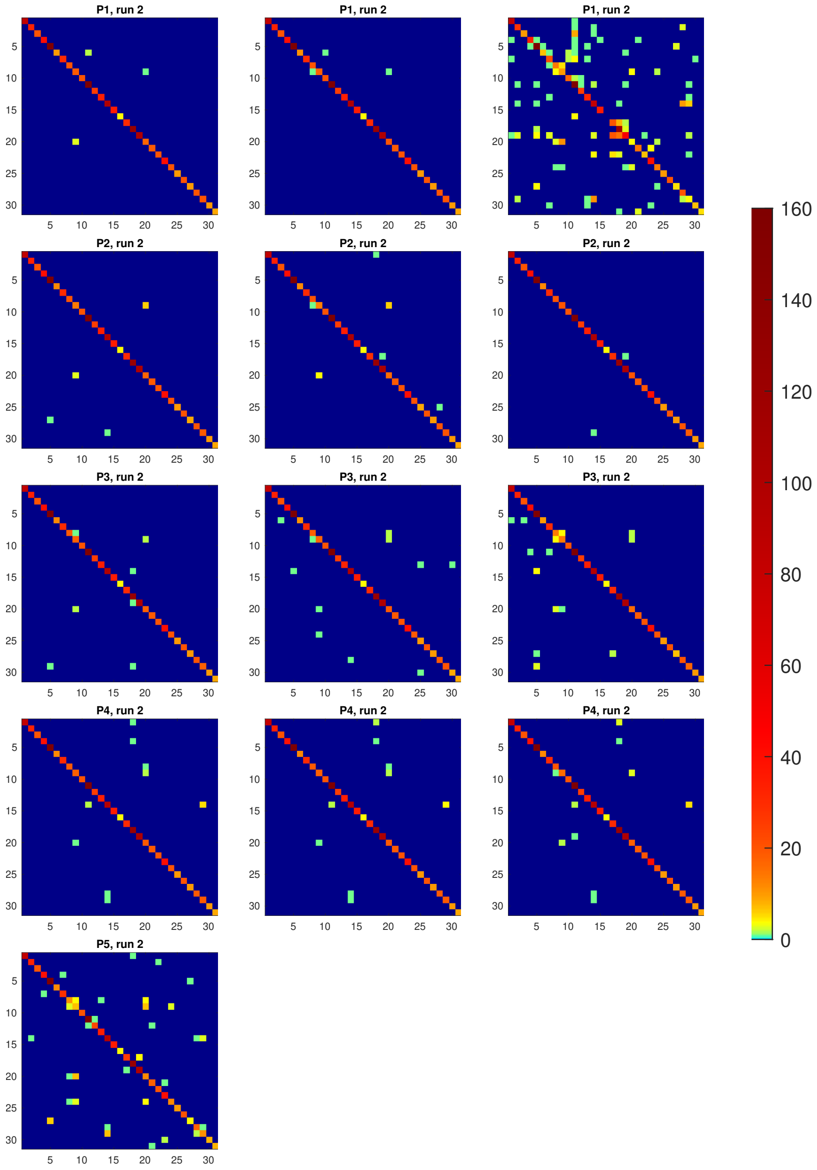


Figure VI.16: Confusion matrices for task B (geometry and physicochemical properties), with respect to the BLAST-based community decomposition of level 3.

Appendix VI.C Supplementary material: Performances with respect to a 3-level BLAST classification

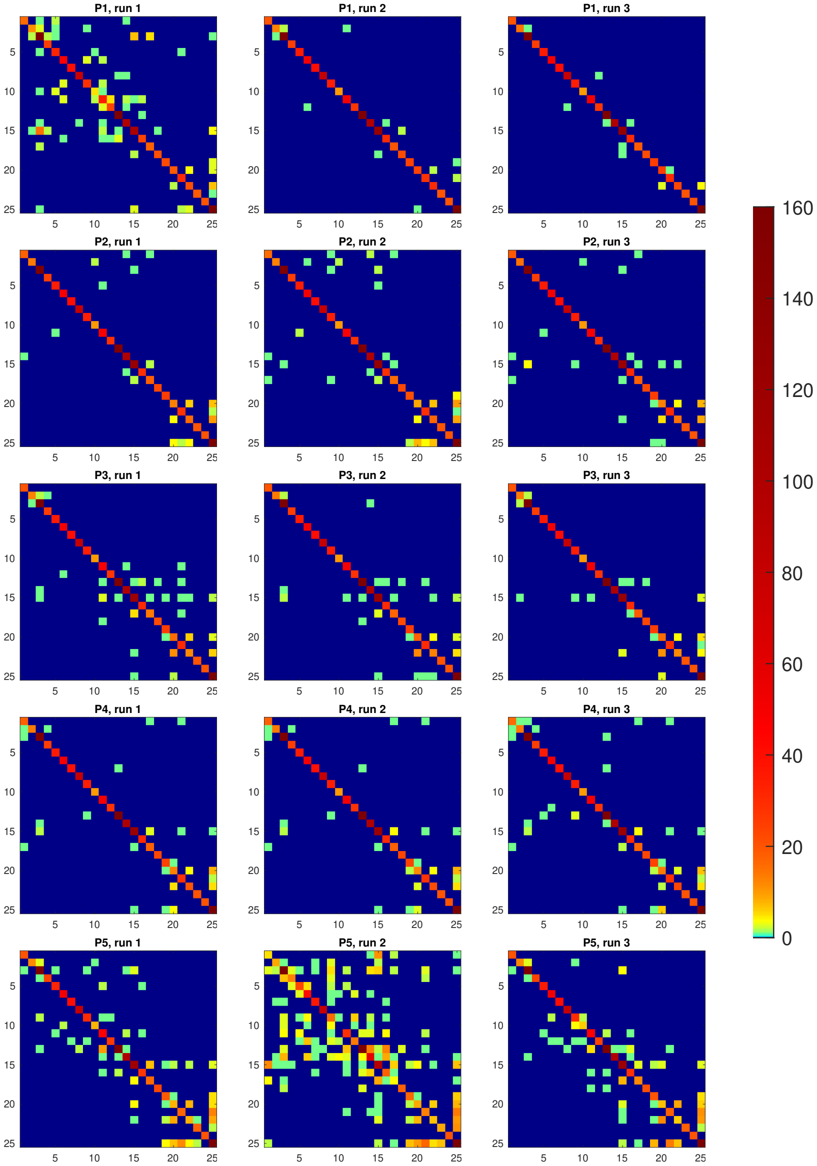


Figure VI.17: Confusion matrices for task A (geometry only), with respect to the BLAST-based community decomposition of level 2.

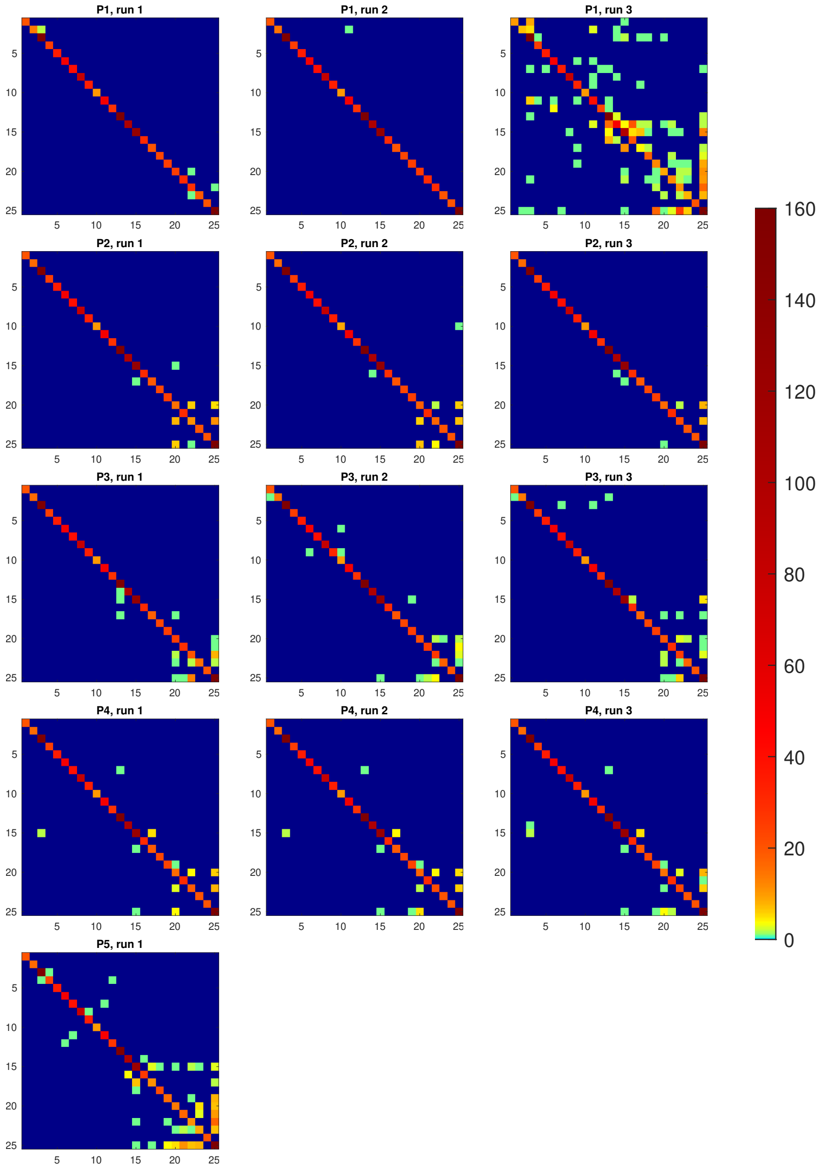


Figure VI.18: Confusion matrices for task B (geometry and physicochemical properties), with respect to the BLAST-based community decomposition of level 2.

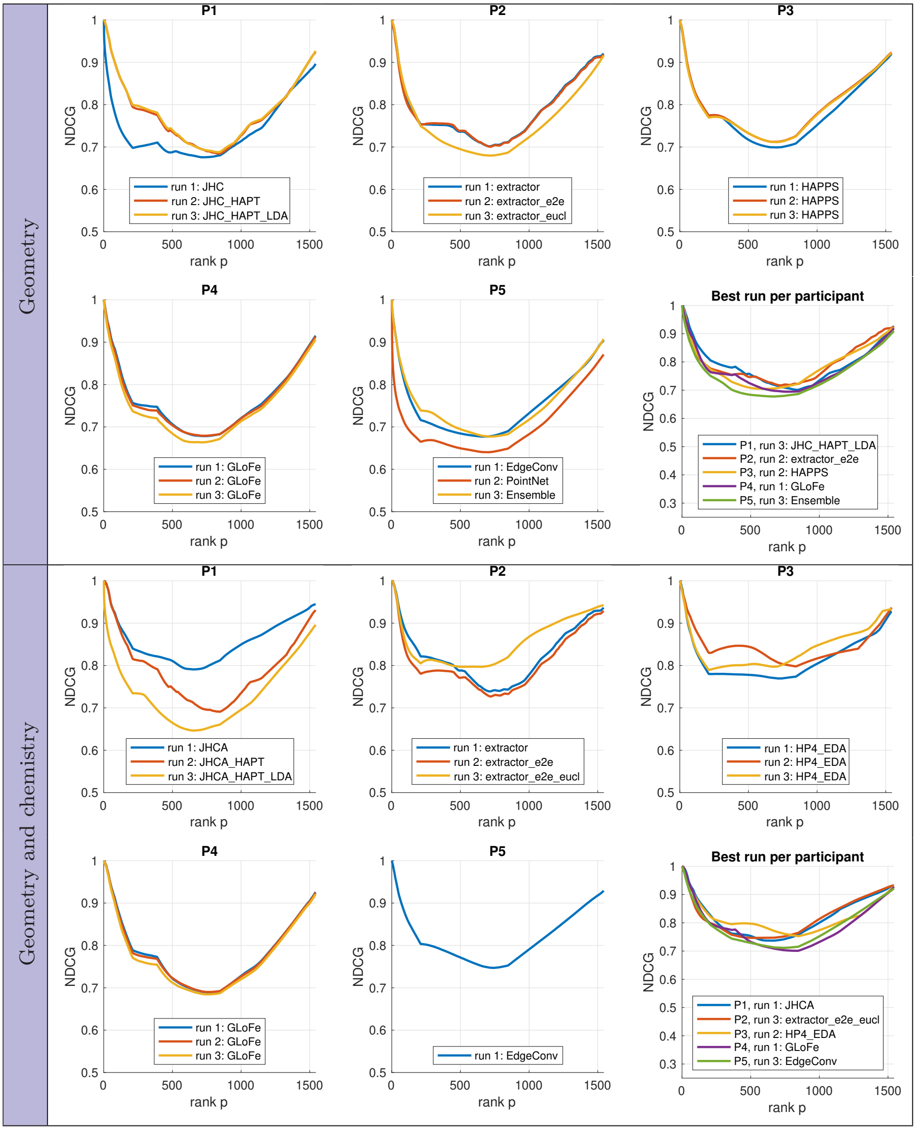


Figure VI.19: Normalized discounted cumulated gain (NDCG) for Task A (geometry only) and Task B (geometry and physicochemical properties), with respect to the 3-level BLAST classification.

Table VI.7: Summary of statistical measures by method and property type (only geometry vs. geometry and physicochemical properties) for the BLAST-based community decomposition of level 2. Here: TPR = True Positive Rate, TNR = True Negative Rate, PPV = Positive Predictive Value, NPV = Negative Predictive Value, ACC = ACCuracy, F1 = F_1 score. For each task and for each measure, the best value for each participant is in bold. The best among them is highlighted in red.

	Geometry							Geometry and chemistry						
	method	TPR	TNR	PPV	NPV	ACC	F1	method	TPR	TNR	PPV	NPV	ACC	F1
P1	run 1	0.9125	0.9922	0.9123	0.9940	0.9878	0.9113	run 1	0.9968	0.9996	0.9968	**1.0000**	0.9996	0.9967
	run 2	**0.9903**	**0.9992**	0.9904	0.9995	0.9988	**0.9902**	run 2	**0.9993**	**1.0000**	**0.9994**	**1.0000**	**1.0000**	**0.9993**
	run 3	**0.9903**	0.9990	**0.9905**	**0.9997**	**0.9989**	0.9900	run 3	0.7991	0.9796	0.8081	0.9842	0.9689	0.7953
P2	run 1	0.9663	**0.9967**	0.9648	0.9978	0.9953	0.9652	run 1	0.9747	0.9974	0.9733	0.9986	0.9966	0.9733
	run 2	0.9495	0.9955	0.9484	0.9959	0.9927	0.9485	run 2	0.9780	**0.9977**	0.9782	0.9981	0.9965	0.9778
	run 3	**0.9701**	**0.9967**	**0.9680**	**0.9987**	**0.9960**	**0.9682**	run 3	**0.9864**	0.9976	**0.9863**	**0.9996**	**0.9976**	**0.9854**
P3	run 1	0.9592	0.9969	0.9579	0.9977	0.9952	0.9575	run 1	0.9767	0.9977	0.9784	0.9975	0.9960	0.9771
	run 2	0.9682	0.9972	0.9675	0.9980	0.9959	0.9671	run 2	**0.9825**	**0.9983**	**0.9831**	**0.9986**	**0.9974**	**0.9825**
	run 3	**0.9702**	**0.9974**	**0.9697**	**0.9983**	**0.9963**	**0.9687**	run 3	0.9760	0.9980	0.9774	0.9980	0.9966	0.9763
P4	run 1	0.9682	**0.9969**	0.9675	0.9981	0.9957	0.9675	run 1	0.9767	0.9974	0.9762	**0.9986**	0.9966	0.9760
	run 2	**0.9689**	**0.9969**	**0.9679**	**0.9982**	**0.9958**	**0.9678**	run 2	**0.9786**	**0.9978**	**0.9784**	0.9984	**0.9968**	**0.9783**
	run 3	0.9631	0.9966	0.9627	0.9973	0.9948	0.9626	run 3	0.9754	0.9971	0.9750	0.9981	0.9960	0.9748
P5	run 1	**0.8996**	**0.9915**	**0.8996**	0.9916	**0.9855**	**0.8986**	run 1	**0.9132**	**0.9906**	**0.9125**	**0.9928**	**0.9861**	**0.9120**
	run 2	0.7155	0.9792	0.7250	0.9778	0.9623	0.7175							
	run 3	0.8983	0.9913	0.8965	**0.9918**	**0.9855**	0.8963							

Table VI.8: Summary of average dynamic recalls (ADRs) for the 3-level BLAST-based classification. For each task, the best ADR for each participant is in bold. The best among them is highlighted in red.

	Geometry					Geometry and chemistry					
	P1	P2	P3	P4	P5		P1	P2	P3	P4	P5
run 1	0.677	**0.735**	0.741	**0.728**	0.686	run 1	**0.829**	0.800	0.800	**0.756**	**0.779**
run 2	0.749	0.728	**0.751**	0.723	0.608	run 2	0.771	0.768	**0.849**	0.751	-
run 3	**0.755**	0.715	0.746	0.708	**0.698**	run 3	0.674	**0.806**	0.810	0.742	-

Authors' addresses

Andrea Raffo Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Genova, Italy, andrea.raffo@ge.imati.cnr.it

Ulderico Fugacci Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Genova, Italy, ulderico.fugacci@ge.imati.cnr.it

Silvia Biasotti Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Genova, Italy, silvia.biasotti@ge.imati.cnr.it

Walter Rocchia Computational Modelling of Nanoscale and Biophysical Systems, Istituto Italiano di Tecnologia, Genova, Italy, walter.rocchia@iit.it

Yonghuai Liu Department of Computer Science, Edge Hill University, Ormskirk, UK, liuyo@edgehill.ac.uk

Ekpo Otu Department of Computer Science, Aberystwyth University, Aberystwyth, UK, eko@aber.ac.uk

Reyer Zwiggelaar Department of Computer Science, Aberystwyth University, Aberystwyth, UK, rrz@aber.ac.uk

David Hunter Department of Computer Science, Aberystwyth University, Aberystwyth, UK, dah56@aber.ac.uk

Evangelia I. Zacharaki Department of Electrical and Computer Engineering, University of Patras, Patras, Greece, ezachar@upatras.gr

Eleftheria Psatha Department of Electrical and Computer Engineering, University of Patras, Patras, Greece, elefpsatha@ece.upatras.gr

Dimitrios Laskos Department of Electrical and Computer Engineering, University of Patras, Patras, Greece, d.laskos@upnet.gr

Gerasimos Arvanitis Department of Electrical and Computer Engineering, University of Patras, Patras, Greece, arvanitis@ece.upatras.gr

Konstantinos Moustakas Department of Electrical and Computer Engineering, University of Patras, Patras, Greece, moustakas@upatras.gr

Tunde Aderinwale Department of Computer Science, Purdue University, West Lafayette, USA, taderinw@purdue.edu

Charles Christoffer Department of Computer Science, Purdue University, West Lafayette, USA, christ35@purdue.edu

Woong-Hee Shin Department of Chemical Science Education, Sunchon National University, Suncheon, Republic of Korea, rainmaker1207@gmail.com

Daisuke Kihara Department of Biological Sciences, Purdue University, West Lafayette, USA, dkihara@purdue.edu

Andrea Giachetti Department of Computer Science, University of Verona, Verona, Italy, andrea.giachetti@univr.it

Huu-Nghia Nguyen University of Science, VNU-HCM, Ho Chi Minh City, Vietnam, nhhnghia@apcs.vn

Tuan-Duy Nguyen University of Science, VNU-HCM, Ho Chi Minh City, Vietnam, nhtduy@apcs.vn

Vinh-Thuyen Nguyen-Truong University of Science, VNU-HCM, Ho Chi Minh City, Vietnam, ntvthuyen@apcs.vn

VI. SHREC 2021: Retrieval and classification of protein surfaces equipped with physical and chemical properties

Danh Le-Thanh University of Science, VNU-HCM, Ho Chi Minh City, Vietnam, ltdanh19@apcs.vn

Hai-Dang Nguyen University of Science, VNU-HCM, Ho Chi Minh City, Vietnam, nhdang@selab.hcmus.edu.vn

Minh-Triet Tran University of Science, VNU-HCM, Ho Chi Minh City, Vietnam, tmtriet@fit.hcmus.edu.vn