

University of Genoa
Department of Mathematics
Ph.D. in Mathematics and Applications
Curriculum in Mathematical Methods for Data Analysis
XXXV Cycle



Recognition and representation of curve and surface primitives in digital models via the Hough transform

Advisors:

Dr. Silvia Biasotti

Dr. Bianca Falcidieno

Ph.D. Student: Chiara Romanengo
Freshman Number 4003155

January 2023

ABSTRACT

Curve and surface primitives have an important role in conveying an object shape and their recognition finds significant applications in manufacturing, art, design and medical applications. When 3D models are acquired by scanning real objects, the resulting geometry does not explicitly encode these curves and surfaces, especially in the presence of noise or missing data. Then, the knowledge of the parts that compose a 3D model allows the reconstruction of the model itself. The problem of recognising curves and surfaces and providing a mathematical representation of them can be addressed using the Hough transform technique (HT), which in literature is mainly used to recognise curves in the plane and planes in space. Only in the last few years, it has been explored for the fitting of space curves and extended to different families of surfaces. Such a technique is robust to noise, does not suffer from missing parts and benefits from the flexibility of the template curve or surface. For these reasons, our approach is inspired by a generalisation of the Hough transform defined for algebraic curves.

In this thesis, we present the methods we implemented and the results we obtained about the recognition, extraction, and representation of feature parts that compose a 3D model (both meshes and point clouds). Specifically, we first study the recognition of plane curves, simple and compound, expressed both in implicit and parametric form, with a focus on the application of cultural heritage and geometric motifs. Then, we analyse the extension of the method to space curves, concentrating on the improvement of the model through the insertion of the recognised curves directly on its surface. To overcome the limitation of knowing in advance the family of curves to be used with the HT, we introduce a piece-wise curve approximation using specific parametric, low-degree polynomial curves. Finally, we analyse how to recognise simple and complex geometric surface primitives on both pre-segmented and entire point clouds, and we show a comparison with state-of-the-art approaches on two benchmarks specifically created to evaluate existing and our methods.

TO MY GRANDFATHER TONY

ACKNOWLEDGEMENTS

I would especially like to thank my advisors Silvia Biasotti and Bianca Falcidieno for being very supportive mentors that taught me everything they could. They introduced me to this beautiful work and they constantly supported and encouraged me during these years.

I would like to thank my brother Carlo, my mother Cristina and her husband Luca for supporting me and putting up with me during this journey. My thanks are also dedicated to my grandparents, Tony and Nanda, and my uncle Dario who always encouraged me in my studies.

I would like to thank all people from IMATI who have become more friends than just colleagues.

I would like to thank the other researchers with whom I had the privilege and the pleasure to work, in particular my colleagues.

I would like to thank my "second family" who always believed in me.

I would like to thank my life-long friends of Cavour who have encouraged me and who have always forgiven me for my carelessness.

I would like to thank my life-long friends of Piampa, who have always supported my work, even though they thought I was crazy when I enrolled in the math course.

TABLE OF CONTENTS

	Page
List of Tables	xi
List of Figures	xiii
Introduction	1
Motivation	1
Contribution	3
Organization of the thesis	4
1 Background material	7
1.1 Theoretical representation and properties of curves and surfaces	7
1.1.1 Some properties of curves	9
1.1.2 Some properties of surfaces	11
1.2 Computational representation and properties of curves and surfaces	15
1.2.1 Notations	15
1.2.2 Mathematical representations in case of codimension 1 in the plane and in the space	16
1.2.3 Mathematical representations in case of codimension 2 in the space	17
1.2.4 Geometric properties	18
1.2.5 Colorimetric properties	19
1.3 The Hough transform for families of curves and surfaces	20
1.3.1 Preliminary concepts on the HT	20
1.3.2 Mathematical representations of the HT in case of codimension 1 in the plane and in the space	22
1.3.3 Mathematical representations of the HT in case of codimension 2 in the space	22
1.3.4 Voting procedure	23
2 Recognition, extraction and representation of plane curves	27
2.1 Previous work	27

TABLE OF CONTENTS

2.2	Families of plane curves	29
2.2.1	Simple plane curves	29
2.2.2	Compound plane curves	33
2.3	A method for recognising plane curves	35
2.3.1	Estimation of the Accumulator Function	37
2.3.2	Computational cost	39
2.4	Testing and validating the method on data from real objects	39
2.5	A comparative analysis for plane curves	41
2.6	Applications	44
2.6.1	Recognition of decorations in archaeological finds	44
2.6.2	Recognition of motifs and symbols	49
2.7	Concluding remarks	51
2.8	Related publications	52
3	Recognition, extraction and representation of space curves	55
3.1	Previous work	55
3.2	Families of space curves	57
3.2.1	Space curves of type I	57
3.2.2	Space curves of type II	58
3.3	A method for recognising space curves	60
3.3.1	Estimation of the Accumulator Function	62
3.3.2	Computational complexity	64
3.4	Testing the method on digital models of real objects	64
3.5	A comparative analysis for space curves	67
3.6	Curve insertion into 3D meshes	70
3.6.1	Feature curves insertion method	71
3.6.2	Curve insertion examples	74
3.7	Concluding remarks	77
3.8	Related publications	78
4	Piece-wise curve approximation using the Hough transform	79
4.1	Pipeline of the piece-wise curve approximation method	79
4.2	Description of the feature approximation method	84
4.2.1	Variation from the standard HT	84
4.2.2	Approximation method considering one projection	85
4.2.3	Approximation method considering two projections	86
4.3	Examples	89
4.4	Feature-preserving point cloud simplification and resampling	89
4.5	Concluding remarks	89

5	Recognition, extraction and representation of geometric primitives	93
5.1	Previous work	94
5.2	Geometric primitives	95
5.2.1	Simple geometric primitives	95
5.2.2	Complex geometric primitives	97
5.3	Recognition and fitting primitives from 3D segmented point clouds	99
5.3.1	Description of the method	99
5.3.2	Computational complexity	106
5.3.3	Performance over different datasets	107
5.3.4	Comparison with the method [127]	111
5.3.5	Tests on point clouds segmented by different methods	111
5.4	Recognition and fitting primitives from 3D point clouds	114
5.4.1	Description of the method	115
5.4.2	Computational complexity	119
5.4.3	Experimental results	120
5.4.4	Comparative analysis	128
5.5	Concluding remarks	130
5.6	Related publications	131
6	Conclusions	133
6.1	Ongoing activities and future directions	135
A	Benchmarking activities	141
A.1	The SHREC2022 track on fitting and recognising geometric primitives in point clouds	142
A.1.1	Dataset and performance measures	142
A.1.2	Evaluation	147
A.2	The Fit4CAD benchmark	152
A.2.1	Dataset and performance measures	154
A.2.2	Evaluation	158
A.3	Related publications	161
	Bibliography	165

LIST OF TABLES

TABLE	Page
2.1 A set of plane curves expressed in both implicit and parametric forms.	34
2.2 Comparison between the parameters of the mathematical curve and those recognised by our algorithm over exact or perturbed curves with respect to both implicit and parametric curve formulation. The values in bold represent the target parameters. The symbol = indicates when the parameter identified by the HT-based algorithm equals the target one.	42
2.3 Quality measures and computational time for the HT-based recognition algorithm for the same plane curve in implicit (I) and parametric (P) form. L represents the number of characteristic points.	43
2.4 Quality measures and computational time for the HT-based recognition algorithm for the same plane curves of Table 2.3, perturbed by the 5% of the diagonal of the curve bounding box. The curve numbers are the same as in Table 2.2. L represents the number of characteristic points.	43
2.5 The similarity matrix among the 6-petal flowers recognised on the four fragments in Figure 2.10(a). The following naming convention is adopted: the first number represents the flower label and the second one corresponds to the model number. . . .	48
3.1 A set of space curves of type I expressed in both implicit and parametric forms.	59
3.2 Comparison between the mathematical curve parameters and those recognised by our algorithm over exact or perturbed curves, with respect to both the implicit and parametric formulation. The labels of the parameters are those used in the curve formulation in Section 2.2. The values in bold represent the target parameters. The symbol = indicates when the parameter identified by the HT-based algorithm equals the target one.	68
3.3 Quality measures and computational time for the HT-based recognition algorithm for the same skew curves in implicit (I) and parametric (P) form. L represents the number of feature points. The curve numbers are the same as in Table 3.2.	69

LIST OF TABLES

3.4	Quality measures and computational time for the HT-based recognition algorithm for the same mathematical curves of Table 3.3, perturbed by the 5% of the diagonal of the curve bounding box. The curve numbers are the same as in Table 3.2. L represents the number of feature points.	70
3.5	Statistics on the mesh properties before and after a curve insertion. The number of vertices refers to the decrease in the mesh vertices after inserting a feature curve. The average edge length and the minimum angle width are reported only for the region modified by our algorithm. The models have different sizes: the models in Figures 3.5(a) and 3.7(c) are normalised in a unit sphere (radius 1); the unit measures are millimetres, metres and centimetres for the models in Figures 3.7(a), 3.7(b) and 3.9(a), respectively.	76
5.1	Parametric representation of some complex geometric primitives.	99
5.2	Basic distances for the primitives considered in the Section 5.2.1, using the notation from Figure 5.1. The subscripts 1 and 2 in the parameters refer to the two segments τ_1 and τ_2	106
5.3	Average classification performances for different correlation queries on the Fit4CAD dataset.	110
5.4	Classification performance: comparison between our method and the approach proposed in [127].	111
5.5	Statistics of the MFES for all models of Section 5.4.3.1. Being the MFE normalized by definition, we can conclude that the maximum error for the fitting of the simple primitives is 4.48%, which corresponds to the noisy holes in the carter of Figure 5.18.	124
5.6	Statistics of the MFES for all models of Section 5.4.3.2. Since the MFE is normalized by definition, we can conclude that the maximum error for the fitting of the simple primitives is 1,54%, which corresponds to the helical surface of Figure 5.22.	126
5.7	Parameter comparison between the original point cloud from Figure 5.16(b) and the perturbed versions from Figure 5.24.	127
A.1	Classification metrics for the whole test set, with respect to each primitive type: T1=plane, T2=cylinder, T3=sphere, T4=cone, T5=torus. Here, gold, silver and bronze identify the first, second and third best performance. The last column contains the macro averages.	150
A.2	Statistics of the fitting errors for the whole test set. Here, gold, silver and bronze identify the first, second and third best performance.	151
A.3	Number of fitted primitives and classification performance metrics: comparison between the PG-based and the HT-based algorithms.	159
A.4	Approximation accuracy of the HT method.	162

LIST OF FIGURES

FIGURE	Page	
1.1	Examples of Dupin's indicatrix. In each of the cases, the principal directions and the corresponding value for the discriminant $LN - M^2$ are given. In (a) the indicatrix is a circle, in (b) a pair of conjugate hyperbolas, and in (c) a pair of parallel lines.	14
1.2	The "classical" shapes of a surface according to the values of mean H and Gaussian K curvature.	15
1.3	The distribution of colours considered "opposites" according to CIELAB.	19
1.4	RGB space colours (left) represented according to CIELAB coordinates (right).	20
1.5	An illustrative example of the point-line duality.	21
1.6	An illustrative example of the voting procedure, considering the profile that outlines the threads of a screw (on the left); in the middle, the Hough transforms of points in the space abc ; on the right, the recognised curve corresponding to the coefficient \bar{a} , \bar{b} and \bar{c} identified by the intersection point in the abc space.	24
2.1	An example of recognition of a complex pattern through the product of curves composing it (courtesy of [132]).	33
2.2	An example of recognition of a complex pattern through the use of repetition rules (courtesy of [132]).	35
2.3	Recognition of various feature curves: in (a) an Archimedean spiral; in (b) a geometric petal of type (A); in (c) a 5-convexities curve; in (d) an egg curve; in (e) a mouth curve; in (f) an elliptic curve. For each pair of pictures, we show on the left the projected points, the curve that best fits them and we highlight in red the points closest to it. This latter set of points is then shown on the original model on the right.	40
2.4	Recognition of Greek fret elements with 8 (a) and 6 (b) straight lines, respectively. In (c) the recognition of the element complementary to (a).	45
2.5	An archaeological fragment (a). Petal-like clusters (b) recognised with citrus curves (c). Measures and GoF of the six citrus curves (d).	46
2.6	A pictorial representation of the criteria adopted to detect which petals belong to the same flower(a), and some examples of flowers recognised on two different fragments (b,c).	46

LIST OF FIGURES

2.7	A detail of floral band annotation. Each petal is numbered (a) for each flower we list its petals (b).	46
2.8	(a) A query element. In (b-g) the retrieved elements are ordered with respect to their increasing distance.	47
2.9	Comparing the parameters of the recognised curves (Figures 2.8(a,b)) it is possible to associate two different fragments as parts of the same moulding.	47
2.10	(a) Four different fragments with the same stylistic character of a floral band; (b) the 6-petal flower is recognised with a geometric petal (type B) curve; (c) a visual representation of the distance among the models shown through a dendrogram plot. .	48
2.11	Examples of recognition of complex motifs: (a) the symbol of Toyota (courtesy of the Toyota Company); (b) the symbol of the Mathematical Olympiad; (c) the <i>Third Paradise</i> . 50	
2.12	The sequence of the three different mathematical representations of the Third Paradise used in the recognition process.	50
2.13	In (a) the smart table Ta.Bi with tags. In (b) and (c) some examples of symbol manipulation varying the value of the parameter a of equation 2.5.	51
2.14	The letters of the word "Toyota" are highlighted inside the symbol.	51
3.1	In (a) an example of an intersection between a hyperbolic paraboloid and a cylinder with a mouth curve as its directrix, with the same axis; in (b) an example of the intersection between an elliptical paraboloid and a cylinder with a Lamet curve as its directrix, with the same axis.	60
3.2	In (a) a 3D model of a screw; in (b) the clusters of points which correspond to a mean curvature value less than 0.1. The column on the right lists some of the clusters found, identified by a number and a colour (black for cluster #1). In (c) the cluster \mathcal{P}_1 represented with the minimum bounding box (in red) that contains it; in (d) the accumulator function \mathcal{H}_i ; in (f) the recognised curve, corresponding to the maximum value of \mathcal{H}_i	62
3.3	Results obtained by the recognition method applied to some 3D models. In (a, d, f, g, h) various types of helices, in (b) a clelia curve and in (c) a pancake curve.	65
3.4	Results obtained by our method on some 3D models.	67
3.5	An overview of the curve insertion method. In (a) a 3d model of a vase with its feature curves is highlighted in red. In (b) its initial triangulation with the blue vertices corresponds to 2 clusters of points extracted by our method. In (c) a detail of the 2 feature curves (notice that these curves are characterised by a stripe of thin triangles). In (d) the insertion into the model of the two recognised curves and a detail (e) of the curve tessellation obtained near them.	71

3.6	Pipeline of the curve insertion algorithm: (a,b) in blue we highlight the boundary \mathcal{B} obtained after the removal of the vertices close to $\mathcal{C}_{\mathbf{a}}$ for two feature curves recognised in the objects of Figure 3.5 and Figure 3.3(c), respectively; (c) identification of the vertices of the boundary \mathcal{B} of (a), highlighted in the picture with small blue dots; (d) the samples on the curve $\mathcal{C}_{\mathbf{a}}$ are shown with red dots; (e,f,g) insertion of the new vertices and the local triangulation criteria; (h) the final triangle mesh with curvilinear elements and in (i) a detail of the curve elements for the red curve in Figure 3.5(e) . . .	72
3.7	Some examples of feature curve insertion. The first two columns show some 3D models and their original triangulation. The last column highlights the new triangles and the curvilinear elements.	75
3.8	Visual comparison between the initial mesh (a) and the one obtained after the feature curves insertion (c) along the border of the decoration. The three inserted curves are highlighted in different colours on the mesh (b).	75
3.9	(a) A model with a colour decoration of a flower and (b) its petals recognised on it. (c) The original mesh tessellation does not reflect any decorative element. (d) The mesh after the insertion of the feature curves, with a detail around the curves inserted. . .	76
3.10	Approximation of curves projected, respectively, on a paraboloid and on a plane: in (a)-(b) using a geometric petal curve; in (c)-(d) using an m-convexities curve (see [150]).	77
4.1	The smoothing step for a point \mathbf{p} of \mathcal{P}_i . In (a) the set $\mathcal{N}(\mathbf{p})$ is highlighted in red, while the first PCA component \mathbf{v} is designed in blue. In (b) the point \mathbf{p} is in blue and its projection point is in green.	80
4.2	Example of a search for a point of self-intersection. In (a) the graph G associated to the input point cloud; in (b,c) a node \mathbf{p} in violet and the vertices of the subgraph $G_{\mathbf{p}}$ in orange; in (d) the points with $branch_number(\mathbf{p}) = 4$ in red, the points with $branch_number(\mathbf{p}) = 3$ in green and the points with $branch_number(\mathbf{p}) = 2$ in blue.	82
4.3	The first three steps of the Douglas-Peucker algorithm considering an open, in (a), and a closed, in (b), profile.	83
4.4	A visual illustration of the steps of our algorithm. In (a) the result of the clustering method: for the following steps we select Cluster #5. In (b) the projection on the (x,y) -plane and the normalization. In (c) the smoothing operation; in (d) the result of the segmentation; finally, in (e) the output of the algorithm.	83
4.5	216 curves of the families $\mathcal{F}_{3,x}$ in blue and 216 curves of the families $\mathcal{F}_{3,y}$ in black with parameters included in the range $[-1, 1]$	85
4.6	Curves recognised by the HT-based recognition algorithm for the first segment of \mathcal{P}_i of Figure 4.4 considering the families $\mathcal{F}_{g,y}$ (a) and $\mathcal{F}_{g,x}$ (b).	87

LIST OF FIGURES

4.7	A visual illustration of the steps of our algorithm in case of space curve approximation. In (a) the projection on the (x, z) -plane and the normalization. In (b) the smoothing operation; in (c,d) the curves recognised by the HT-based recognition algorithm for the first segment considering the families $\mathcal{G}_{g,z}$ (c) and $\mathcal{G}_{g,x}$. In (e) the approximation of all segments. Finally, in (f) the output of the algorithm on the input point cloud \mathcal{P}	88
4.8	Some results of our piece-wise curve approximation method. In the first column the original model from which the point cloud is extracted. In the central columns the resulting approximation of two projections. In the final column, the outcome curves are highlighted on the model.	90
4.9	An example of a decimation process performed on a point cloud. In (a), the input point cloud with the feature points is highlighted in red. In (b) - (f), the output is obtained by reducing the number of points by a factor of 10, 20, 50, 100, and 200, respectively.	91
5.1	Simple geometric primitives: plane, cylinder, cone, sphere and torus respectively, along with their attributes (geometric descriptors).	97
5.2	Initial estimates for a sphere. The preprocessing step centres 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 centre.	101
5.3	Initial estimates for a circular cylinder. The preprocessing step centres the point cloud and approximates the normal vectors at a set of points, see (b). Given two points \mathbf{p}_{j_1} and \mathbf{p}_{j_2} , (c) shows the corresponding normal vectors $\hat{\mathbf{n}}_{j_1}$ and $\hat{\mathbf{n}}_{j_2}$, in blue and green respectively, and, in red, their cross product $\hat{\mathbf{a}}_{j_1, j_2}$. Finally, (d) shows the estimate \hat{r} of the radius.	102
5.4	Initial estimates for a circular cone. The preprocessing step centres 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} , (d) shows 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.	102
5.5	Initial estimates for a torus. The preprocessing 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 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.	103
5.6	Pipeline of the method for fitting and recognising primitives using the Hough transform.	105
5.7	A model from [91]: we show segments of the same primitive type exhibiting different similarities. Colours are used to visually represent primitives sharing the same property.	108
5.8	Two models from [91]: segments exhibiting different similarities are shown.	109

5.9	A model from [91]: a Gaussian noise is applied at three levels of intensity.	109
5.10	Examples of segmentations from [148]: the original segmentations (first column) are post-processed by our method to overcome the problem of oversegmentation.	112
5.11	Examples of queries for planar, cylindrical, conical, and spherical segments obtained from the RANSAC segmentation of a scanned industrial object as provided in [100].	113
5.12	Examples of queries for planar, cylindrical, and spherical segments obtained from the RANSAC segmentation over a challenging point cloud acquired from an industrial object as provided in [100].	114
5.13	Pipeline of the method.	117
5.14	In (a) a mechanical CAD model from the benchmark in [91]; in (b) the vertices of its triangle mesh decomposed into 8 surface segments; in (c), for each primitive, the HT parameters are compared with those provided by the database	120
5.15	In (a) a mechanical CAD model from the benchmark in [91]; in (b) the vertices of its triangle mesh decomposed into 9 surface segments; in (c), for each primitive, the HT parameters are compared with those provided by the database	121
5.16	Recognition of CAD point clouds containing only simple geometric primitives. The identification of maximal segments does not require, in these cases, the application of any clustering algorithm.	121
5.17	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) in different colours. (c) draws attention to cylinders, among which we can recognise segments lying on the same primitive, up to a translation.	122
5.18	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 correspond to different points of view.	123
5.19	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).	123
5.20	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.	124
5.21	Recognition of complex geometric primitives in CAD point clouds. Their identification does not require, in these cases, the application of any clustering algorithm.	125

5.22	A screw-like part. The original model, (a), is sampled. The surface primitives detected via HT are shown in (b) in 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 them lies on the same parametrised surface, the 2 helical strips have the same equation up to a translation, as shown in (c) (both in orange).	125
5.23	A clamp connector. In (a) the original model. In (b) the decomposition of the corresponding point cloud into 38 segments is provided by the HT procedure. In (c), the final grouping is obtained by clustering, consisting of 6 groups of primitives (here, singletons of segments are transparent).	126
5.24	The point cloud in Figure 5.16(b) is perturbed by adding zero-mean Gaussian noise of standard deviation: 0.01, 0.05, 0.10 and 0.20. 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.	127
5.25	Primitive type recognition: a comparison between our approach, a RANSAC-based segmentation [144], and the method in [96]. Different colours correspond to different primitive types: planes (red), cylinders (green), cones (blue), tori (black), splines (pink), and unsegmented (yellow).	129
5.26	Comparison of our approach (HT) with other three methods: a primitive growing approach (PG), ParSeNet (PN) and HPNet (HN). The analysis is performed over the Fit4CAD benchmark [140]	130
6.1	Examples of images present in our dataset.	136
6.2	In (a) the validation metrics and in (b) the confusion matrix associated to the classification problem.	137
6.3	The validation metrics associated with the regression problem.	138
6.4	Three examples of images in our dataset and in red the line that represent the estimation of the rotation angle.	138
A.1	Examples of point clouds from the dataset. Columns identify primitive types: plane (T1), cylinder (T2), sphere (T3), cone (T4) and torus (T5). Rows correspond to point cloud artefacts: none (A0), uniform noise (A1), Gaussian noise (A2), undersampling (A3), missing data (A4), uniform noise + undersampling (A5), Gaussian noise + undersampling (A6), uniform noise + missing data (A7), Gaussian noise + missing data (A8), small deformations (A9).	144
A.2	Confusion matrices CM for the whole test set, with respect to each primitive type. Here, the entry $CM(i, j)$ indicates the number of samples with the true label being the i -th class and the predicted label being the j -th class.	148
A.3	Bar graphs of the macro averages for the classification measures, grouped by method. The legend reflects the colour encoding of the perturbation types.	149

A.4	Bar graphs of the log macro averages for the recognition and fitting measures, grouped by method. The legend reflects the colour encoding of the perturbation types.	153
A.5	Example of models obtained using Onshape.	155
A.6	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).	155
A.7	The 35 point clouds used as a test set. Different colours represent different primitives, as stored in the CAD models, i.e., our ground truth.	156
A.8	Boxplot for the classification metrics presented in Table A.3. All 35 models are here considered.	160
A.9	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 the classification metrics presented in Table A.3.	160

INTRODUCTION

The analysis and the recognition of characteristic parts of a 3D model are fundamental steps for classifying, grouping, and possibly indexing digital models in proprietary or online datasets. Curve and surface primitives play a crucial role in conveying an object shape, and their recognition finds substantial uses in manufacturing, art, design and medical fields. For instance, in reverse engineering, 3D scanning devices are used to digitise and validate a manually optimised physical prototype. It is extremely important that no details of the scanned object, such as sharp edges, corners and, in general, surface features, are lost during the acquisition process. Indeed, when 3D models are acquired by scanning real objects, the reconstructed point cloud does not explicitly encode these curves and surfaces, especially when it is affected by noise or missing parts. Measurement uncertainty, sampling resolution, or occlusions during the acquisition occur in applications like the digitisation of archaeological artefacts, often damaged, and whose details are partially missing. Then, a large number of points, the presence of noise and outliers, the occurrence of missing or redundant parts and the non-uniform distribution of the data are the main problems to be addressed in many application areas. In this thesis, we propose a solution based on the Hough transform (HT), that is able to recognise curves and surface primitives and to provide a mathematical representation of them, ensuring the robustness to noise, outliers and missing parts.

Motivation

The characterisation and recognition of curve patterns on 3D models is a well-known issue in computer graphics. Feature curves, i. e., lines that characterise a shape feature, are useful for visual shape illustration [92], and perception studies support these curves as an effective choice for representing the salient parts of a 3D model [39, 72]. In our context, as feature curves we consider sets of points of a model that jointly define a contour, a valley, or a ridge on the model, thus giving local information about the surface. If the model has some boundary, it is not considered a feature curve, as well as noise artefacts (e.g., from scan inaccuracy and/or smoothing/resampling operations).

In the field of cultural heritage, the identification and recognition of feature elements support the experts in documenting the relationships among multiple fragments or different digital representations. Currently, the annotation of 3D cultural heritage artefacts is performed manually

[162] or limited to a few anatomical features, like in the dashboard implemented for the project GRAVITATE [35]. Indeed, there is a lack of tools that help to convert intuitive reasoning into objective criteria, measures, and semantic annotations. Existing methods for the analysis of digital artefacts are mainly limited to reconstruction [123], visual enhancement of carvings [95], the definition of a global similarity score [23] or analysis of specific features, like constant radius decorations over potteries [47].

Besides archaeology, the need of interpreting the feature of an artist or a culture comes from many fields. For instance, the lines that are the signature of Gaudi's architectural arches have been subject to many discrepancies regarding the arch geometry; to address disputes, specific studies have been proposed to deduce which mathematical curve better approximates them, see [141]. In decorative arts, there exists a myriad of possible motifs and those motifs have been distinguished based on formal properties and divided according to different categorisations. Crafts such as weaving, which are intrinsically pattern-like, show that humans can apply their ability of geometric abstraction to different objects and phenomena and create visual motifs on any drawing, carving, cloth, tapestry or other artefacts. The cultural practice also indicates that the capability to recognise and make geometric decorative patterns has always been inherent in human nature. Indeed, today geometric motifs are often used to design logos, trademarks and symbols, which identify products, works, or events and some of these motifs are graphic elements obtained from the composition of mathematical curves.

Another common problem is the creation of a high-level model representation, in which the characteristic parts are annotated. 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 it 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 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: [31, 88]. 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. This step is very important in the RE framework, as the results obtained may

significantly differ depending on the strategy adopted to perform this task.

Contribution

The main contribution of this thesis lies in the solutions proposed to recognise curves and surfaces in 3D digital models through the use of the Hough transform technique. A recognition method based on the HT is able to deal with both curve and surface primitives, in order to provide a mathematical representation of them, and it is robust to noise, outliers and missing parts. Then, it can be an answer to the problems mentioned before.

Approaches based on the HT in the literature mainly focus on the recognition of hypersurfaces, that is objects of codimension 1 such as plane curves in images and surfaces in space. One of the challenges is to extend them to codimension 2 cases, such as the case of space curves in space. Regarding the recognition of surfaces, HT-based approaches mainly focus on the identification of planes and spheres. In this context, the challenge is to effectively extend the recognition method to a larger set of geometric primitives.

Preliminary, our study focuses on the recognition of space feature curves that can be projected onto the regression plane, exploiting a dictionary of plane curves. Specifically, we first extend the method in [154] by introducing new plane curves and new rules of composition and aggregation of curves based on rotations and translations of the pattern elements. Then, we provide a compatibility measure among the curves that belong to the same family, even if located on objects of an overall different shape. To do this, we exploit the parameters that appear in the mathematical representation of these curves, since they are strictly related to the geometric properties of the curves. From the compatibility point of view, we show the effectiveness of our approach by applying it to the archaeological domain and finding similarities among fragments.

Then, we extend the nowadays well-established Hough-type technique for plane curves to space curves in 3D digital objects. Specifically, we extend the technique mentioned before to the identification of space curves that cannot be projected onto the regression plane. The limited availability of templates for space curves has probably reduced the interest in such a fitting problem. To overcome this problem, we extend the dictionary of space curves by also including those whose profile can be approximated as the intersection of a quadric surface with a cylinder generated by a plane curve. Spatial curves are then distinguished into classical (type I) and approximated with quadric surfaces (type II).

Since our technique for identifying curves on 3D shapes works for both parametric and implicit curve representations, the pros and cons of both forms are also discussed, highlighting when to select one instead of the other or when the two curve formulations are not interchangeable. In particular, the quality of the curve approximation through the HT-based recognition for the two forms of representation is compared, evaluating their advantages and drawbacks.

Recognition methods based on the HT require prior knowledge of the family of curves to look

for. In some application contexts such as cultural heritage, this suggestion can come directly from the expert, through a template or a drawing, or from the technical documentation associated with the find. To overcome this limitation, we work on a method that provides a piece-wise space curve approximation using specific parametric polynomial curves of degree 3 or 4.

Regarding the recognition of surfaces, our first contribution concerns the introduction of a dictionary of surfaces made of simple and complex geometric primitives. Then, we implement two different methods. The first one is a new technique to reduce the dimension of the parameter space and localise the search for the optimal solution, thus making the application of the HT algorithm possible to a larger number of primitives. The pipeline is able to compute initial estimates for the parameters of spheres, cylinders, cones, and tori. The second method is able to recognise multiple instances of the same primitive presented on a single point cloud and extracts complex primitives, in addition to the most common ones (planes, cylinders, cones, spheres, tori). Both approaches can identify global relations among the recognised geometric primitives, such as lying on the same primitive, sharing the same axis or radius, or being on parallel planes, etc..

Simultaneously, we create datasets and metrics aimed at evaluating methods for detecting simple geometric primitives both in 3D point clouds representing CAD objects and in 3D point clouds representing simple geometric primitives with different kinds of perturbations.

Organization of the thesis

This thesis is divided into 6 chapters and an appendix. Chapter 1 provides a brief introduction of the basic notions required to frame our work. The first two parts are dedicated to the representations of curves and surfaces, from both theoretical and computational points of view, together with the definition of their geometric and colorimetric properties. The third part focuses on the main concepts of the Hough transform technique and its representations.

Chapters 2 and 3 deal with the recognition of curves through the HT and are organised in the same way: they provide an overview of the current literature, the dictionary available in each method, a detailed description of the approach and some examples of applications. Specifically, Chapter 2 deals with the recognition of space curves projectable on the regression plane via a technique that exploits a dictionary composed of both simple and compound plane curves. Since this approach can consider curves with implicit or parametric representation, we also provide a comparative analysis between two expressions in terms of the quality of the approximation and computational complexity. Finally, we show how the parameters that uniquely identify the recognised curves can be exploited in the archaeological domain and in an interactive experience.

Chapter 3 focuses on the recognition of space curves, applying the HT to the codimension 2 case in the space. In this context, we consider two classes of space curves: the first type of curves is equipped with a known representation, while the second one is defined starting from the large dictionary of plane curves. Also in this case, a comparative analysis between the methods

in implicit and parametric form is provided. Finally, as an application, we show the insertion of curvilinear elements as constraints on a mesh, either to modify the local mesh tessellation, emphasising some characteristics, or complete missing parts.

Since the previous two approaches require a priori-knowledge of which family of curves to look for, in Chapter 4 we introduce a method for piece-wise curve approximation of curvilinear profiles in point clouds, able to approximate both planar and spatial profiles. Also, in this case, we show an example of application that exploits the mathematical expression of the curves to drive a constrained point cloud simplification and resampling.

Chapter 5 deals with the recognition of surfaces, introducing a dictionary that contains both simple geometric primitives (such as planes, cylinders, spheres, cones, and tori) and complex geometric primitives (general cylinders, general cones, surfaces of revolution, helical surfaces and convex combination of curves). In this context, we describe two methods. The first one takes in input a pre-segmented point cloud and provides, for each segment, the geometric descriptors that uniquely identify it. The second one is applied to a point cloud representing a CAD object that might contain different types of primitives and returns a segmentation together with the mathematical expressions of segments. For each proposed method, we show some applications and comparisons with the state-of-the-art approaches. In Appendix A we describe our two benchmarks that concern the identification of simple geometric primitives and that we created to test and compare our methods introduced in Chapter 5.

Chapter 6 summarises the achievements obtained, draws the conclusions of this work and addresses open issues and potential research directions for the future.

Authorship declaration

This thesis contains published and unpublished works that have been co-authored. In order, the material for the second chapter has been supplied by [133], [135] co-authored with my advisors Silvia Biasotti and Bianca Falcidieno, and [137] co-authored with Silvia Biasotti, Bianca Falcidieno, Chiara Eva Catalano and Erika Brunetto. The third chapter contains the material in [136] and [138], co-authored with Silvia Biasotti and Bianca Falcidieno. Chapter 4 is an ongoing work in collaboration with Silvia Biasotti, Bianca Falcidieno and Ulderico Fugacci. The first method presented in Chapter 5 is provided by [128], while the second approach is described in a work submitted for publication. Both works are co-authored by Andrea Raffo, Silvia Biasotti and Bianca Falcidieno. Finally, the material for Appendix A has been supplied by [139] and [140], co-authored with different researchers from various research groups.

BACKGROUND MATERIAL

This chapter is devoted to a brief presentation of basic notions useful to better understand the sequel. It is organised as follows. In the first section, we introduce some preliminary concepts about manifolds, curves and surfaces and their properties, such as curvatures. In Section 1.2 we show the mathematical representations of curves and surfaces that allow us to analytically represent them and we describe how we use and estimate their geometric and colorimetric properties. Finally, Section 1.3 introduces the main concepts underlying the Hough transform technique and its representations.

1.1 Theoretical representation and properties of curves and surfaces

From the theoretical point of view, the model usually referred to when discussing the representation of a geometric shape is that of *manifold or variety*. For the definition of manifold, we need to introduce some preliminary concepts (see [25, 70, 103, 115]).

Definition 1.1.1 (Function C^∞). Let U be an open set of \mathbb{R}^n and $f : U \rightarrow \mathbb{R}^m$ a function. This function is said to be of class C^∞ or *smooth* if its partial derivatives exist and are continuous for each order of differentiation.

Definition 1.1.2 (Topological Hausdorff space or T_2). A topological space X is a *topological Hausdorff space* or T_2 if for all $\mathbf{x}, \mathbf{y} \in X$, $\mathbf{x} \neq \mathbf{y}$, there exist two neighborhoods $U_{\mathbf{x}}, U_{\mathbf{y}}$ containing respectively \mathbf{x} and \mathbf{y} such that $U_{\mathbf{x}} \cap U_{\mathbf{y}} = \emptyset$. In other words, it is a topological space in which for two distinct points one can always find disjointed open neighborhoods.

Definition 1.1.3 (Manifold without boundary). A topological Hausdorff space M is called a k -dimensional topological manifold if each point $\mathbf{x} \in M$ admits a neighborhood $U_i \subseteq M$ homeomorphic to the open disk $D^k = \{\mathbf{x} \in \mathbb{R}^k \mid \|\mathbf{x}\|_2 < 1\} \subseteq \mathbb{R}^k$ and $M = \bigcup_{i \in \mathbb{N}} U_i$. Given ϕ_i the homeomorphisms relative to U_i , a *chart* is defined as the pair (ϕ_i, U_i) . The set of charts of a single covering is called an *atlas*.

The number k represents the dimension of the manifold. In practice, by this definition, we require that M locally possesses the same topological properties as a Euclidean space of appropriate dimension. It should be emphasised that covering is not necessarily done by disjoint open U_i . How two maps must be connected is expressed with the transition functions.

Definition 1.1.4 (Transition maps). Let X be an atlas and two charts related to the open sets U_i e U_j , $i \neq j$. The *transition maps between i and j* is the function

$$\phi_{i,j} = \phi_i(U_i \cap U_j) \rightarrow \phi_j(U_i \cap U_j).$$

Transition functions describe how to move from one chart to another. The regularity of the transition functions determines how smooth such a transition is, and in the case where $\phi_{i,j} \in C^\infty$, $\forall i \neq j$, the manifold is called *differentiable manifold*.

Definition 1.1.5 (Manifold with boundary). A topological Hausdorff space M is called a k -dimensional topological manifold with boundary if each point $\mathbf{x} \in M$ admits a neighborhood $U_i \subseteq M$ homeomorphic to the open disk D^k or to $\mathbb{R}^{k-1} \times \{y \in \mathbb{R} \mid y > 0\}$. Moreover M admits a numerable covering of such neighborhoods.

The definitions of chart and atlas are analogous to the case without boundary, but the space to which the neighborhoods are homeomorphic changes: for the interior points the disk of appropriate dimensions remains, while the contours involving the boundary are homeomorphic to the k -dimensional half-space. Points on a manifold with boundary are classified either *interior*, if they have a neighborhood homeomorphic to an open disk, or *boundary*, if their neighborhood is homeomorphic to a half-disk.

Examples of 3-dimensional manifolds with boundary are the solid sphere and the solid torus, while the boundary, the usual sphere S^2 and the torus T^2 , are two closed 2-manifolds. The circle and the open interval $(0, 1)$ are examples of 1-manifolds. In general terms, a 2-manifold is called a *surface*.

Definition 1.1.6 (Smooth Manifold). A k -topological manifold without (respectively with) boundary is called a *smooth manifold* without (respectively with) boundary if all transition functions are smooth.

Accordingly, a k -topological manifold without (respectively with) boundary is called a manifold of class C^m without (respectively with) boundary, if all transition functions are of class C^m .

Definition 1.1.7 (Orientability). A manifold M is called *orientable* if there exists an atlas $\{(U_i, \phi_i)\}$ on it such that the Jacobian of all transition functions $\phi_{i,j}$ from a chart to another is positive for all interesting pairs of regions. Manifolds that do not satisfy this property are called *non-orientable*.

Intuitively, a surface is orientable if you can orient the normal to the surface consistently at every point, for example, by the right-hand rule. As a common convention, for the representation of real objects, the normal is used to identify the external boundary. Indeed, in case of surfaces without boundary, the direction of the surface is determined by the direction of its normals (the outer direction is the one that sees the normals as outgoing, vice versa the inner). In case of surfaces with boundary, there is an ambiguity of sign for the normals. In practice, one of the two directions is chosen as outward (and thus an orientation is fixed).

In this thesis, we will consider only orientable surfaces, with or without boundary, and curves. Often the elements we are going to consider will not be smooth, but will have a lower level of continuity.

1.1.1 Some properties of curves

In general, a curve is defined through a continuous function $\gamma : I \rightarrow X$ from an interval I of the real numbers into a topological space X . In other words, a curve is a topological space which is locally *homeomorphic* to a line. If the function γ is differentiable, then the curve is called differentiable. In case X is of three dimensions, such as the Euclidean space \mathbb{R}^3 , the curve is called *space curve*, while if X is a plane, it is called *plane curve*. Space curves that do not lie on a plane are called *skew curves*. For the material provided in this section, we refer to [49, 58, 90].

Definition 1.1.8 (Regular curve). A differentiable curve γ is called *regular* curve if γ' is continuous and $\gamma'(t) \neq \mathbf{0}$ for all $t \in I$. Similarly, a differentiable curve is said to be *m-regular* if the function γ is of class C^m and $\gamma'(t) \neq \mathbf{0}$ for all $t \in I$. In case $m = \infty$, the curve is called *smooth*.

The points that satisfy the regularity condition are called regular points. If there exists $t_0 \in I$ such that $\gamma'(t_0) = \mathbf{0}$, then the point $\gamma(t_0)$ is said to be a *singular* point of the curve. Note that, the condition of regularity $\gamma'(t) \neq \mathbf{0}$ for all $t \in I$ can be expressed in an equivalent way through the norm of $\gamma'(t)$, then requiring that $\|\gamma'(t)\|_2^2 \neq 0$ for all $t \in I$.

Definition 1.1.9 (Geometric continuity). A curve is said to be of class G^m (has geometric continuity of order m , where $m \geq 1$) if there exists a local regular parametrization of class C^m of this curve in a neighbourhood of each point of this curve.

Definition 1.1.10 (Simple curve). A regular curve γ is said to be *simple* if there are no multiple points, that is if $t_1 \neq t_2$ implies $\gamma(t_1) \neq \gamma(t_2)$.

In the rest of the section, we assume $X = \mathbb{R}^3$. We introduce a special local coordinate system, linked to a point $\boldsymbol{\gamma} = \boldsymbol{\gamma}(t)$ on the curve, that will significantly facilitate the description of local curve properties at that point. Let us assume that all derivatives needed below do exist. The first terms of the Taylor expansion of $\boldsymbol{\gamma}(t + \Delta t)$ at t are given by

$$\boldsymbol{\gamma}(t + \Delta t) = \boldsymbol{\gamma} + \boldsymbol{\gamma}' \Delta t + \boldsymbol{\gamma}'' \frac{1}{2} (\Delta t)^2 + \boldsymbol{\gamma}''' \frac{1}{6} (\Delta t)^3 + \dots$$

Let us assume that the first three derivatives are linearly independent. Then $\boldsymbol{\gamma}'$, $\boldsymbol{\gamma}''$, $\boldsymbol{\gamma}'''$ form a local affine coordinate system with origin $\boldsymbol{\gamma}$. From this local affine coordinate system, one easily obtains a local Cartesian (orthonormal) system with origin $\boldsymbol{\gamma}$ and axes \mathbf{t} , \mathbf{n} , \mathbf{b} by the Gram-Schmidt process of orthonormalization

$$\mathbf{t} = \frac{\boldsymbol{\gamma}'}{\|\boldsymbol{\gamma}'\|}, \quad \mathbf{n} = \mathbf{b} \wedge \mathbf{t}, \quad \mathbf{b} = \frac{\boldsymbol{\gamma}' \wedge \boldsymbol{\gamma}''}{\|\boldsymbol{\gamma}' \wedge \boldsymbol{\gamma}''\|}$$

where \mathbf{t} is called tangent vector, \mathbf{n} is called main normal vector and \mathbf{b} is called binormal vector. The frame \mathbf{t} , \mathbf{n} , \mathbf{b} is called the *Frenet frame*. Letting the Frenet frame vary with t provides a good idea of the curve's behavior in space. The plane spanned by the point $\boldsymbol{\gamma}$ and the two vectors \mathbf{t} and \mathbf{n} is called the *osculating plane*.

A change $\tau = \tau(t)$ of the parameter t where τ is a differentiable function of t will not change the shape of the curve. This reparametrization will be regular if $\tau' \neq 0$ for all $t \in I$, where $I = [a, b]$. Let

$$s = s(t) = \int_a^t \|\boldsymbol{\gamma}'\| dt$$

be such a parametrization. Then, s is independent of any regular reparametrization because

$$\boldsymbol{\gamma}' dt = \frac{d\boldsymbol{\gamma}}{d\tau} \frac{d\tau}{dt} dt = \frac{d\boldsymbol{\gamma}}{d\tau} d\tau.$$

It is an invariant parameter and it is called *arc length* parametrization of the curve.

The following formulas may be defined both in terms of arc length s and in terms of the actual parameter t , but they are particularly simple if one uses arc length parametrization. Some simple calculations yield the so-called *Frenet-Serret* formulas:

$$\mathbf{t}' = \mathbf{t}'(s) = \kappa \mathbf{n}, \quad \mathbf{n}' = \mathbf{n}'(s) = -\kappa \mathbf{t} + \tau \mathbf{b}, \quad \mathbf{b}' = \mathbf{b}'(s) = -\tau \mathbf{n}$$

where the terms κ and τ are called *curvature* and *torsion*. The curvature κ measures the deviance of $\boldsymbol{\gamma}$ from being a straight line relative to the osculating plane; the torsion τ measures the deviance of $\boldsymbol{\gamma}$ from being a plane curve. They may be defined both in terms of arc length s and in terms of the actual parameter t . We give both definitions:

$$\kappa = \kappa(s) = \|\boldsymbol{\gamma}''\|, \quad \kappa = \kappa(t) = \frac{\boldsymbol{\gamma}' \wedge \boldsymbol{\gamma}''}{\|\boldsymbol{\gamma}'\|^3},$$

$$\tau = \tau(s) = \frac{1}{\kappa^2} \det(\boldsymbol{\gamma}', \boldsymbol{\gamma}'', \boldsymbol{\gamma}'''), \quad \tau = \tau(t) = \frac{\det(\boldsymbol{\gamma}', \boldsymbol{\gamma}'', \boldsymbol{\gamma}''')}{\|\boldsymbol{\gamma}' \wedge \boldsymbol{\gamma}''\|^2}.$$

1.1.2 Some properties of surfaces

A surface is defined through a continuous function $\mathbf{f}: \Omega \rightarrow X$ from a subset Ω of \mathbb{R}^2 into a topological space X of dimension at least three. In other words, a surface is a topological space which is locally *homeomorphic* to a plane or an half-plane. If the function \mathbf{f} is differentiable, then the surface is called differentiable. In the rest of this section, we will assume that $X = \mathbb{R}^3$. For more details on the material provided in this section, we refer to [25, 49, 58] and, for curvatures, to [103], Chapter 4.

Definition 1.1.11 (Regular surface). A differentiable surfaces \mathbf{f} is called *regular* surface if $\mathbf{f}_t(t, u)$ and $\mathbf{f}_u(t, u)$ are continuous and $\mathbf{f}_t(t, u) \wedge \mathbf{f}_u(t, u) \neq \mathbf{0}$ for all $(t, u) \in \Omega$, that is both families of isoparametric lines are regular and are nowhere tangent to each other.

The points that satisfy the regularity condition are called regular points. If it exists $(t_0, u_0) \in \Omega$ such that $\mathbf{f}_t(t_0, u_0) \wedge \mathbf{f}_u(t_0, u_0) = \mathbf{0}$, then the point $\mathbf{f}(t_0, u_0)$ is said to be a *singular* point of the surface.

The definition of curvature of a surface is derived from the concept of curvature for a curve. The definition of the curvatures of a surface passes through two local invariants, called fundamental forms.

1.1.2.1 Fundamental forms

Definition 1.1.12 (Coordinate patch). Let S be a surface and U an open set in the uv plane. The map $\phi: U \rightarrow S$ is called *coordinate patch* of class C^m , with $m \geq 1$, of U in S , $\phi = \phi(u, v)$, such that:

- ϕ is C^m on U ;
- $\partial_u \phi \times \partial_v \phi \neq \mathbf{0} \quad \forall (u, v) \in U$;
- ϕ is injective and bicontinuous on U .

Let $\phi = \phi(u, v)$ be a C^m coordinate patch of a neighborhood U of $(u, v) \in S$, $m \geq 1$ and $d\phi = \phi_u du + \phi_v dv$ its differential, which generates vectors parallel to the tangent plane in $\phi(u, v)$. Committing a little abuse of notation, one can use $d\phi$ to denote both the vector generated by $\phi_u du + \phi_v dv$ and the differential. Remembering that

$$\phi(u + du, v + dv) = \phi(u, v) + d\phi + o((du^2 + dv^2)^{\frac{1}{2}}),$$

it can be said that $d\phi$ is the first-order approximation of the vector $\phi(u + du, v + dv) - \phi(u, v)$, from $\phi(u, v)$ and $\phi(u + du, v + dv)$.

Definition 1.1.13 (First fundamental form). Considering the notations introduced before, we define the *first fundamental form* of ϕ by the product $I = d\phi \cdot d\phi$. In more explicit terms, we have:

$$I = Edu^2 + 2Fdudv + Gdv^2 = I(du, dv),$$

where

$$E = \phi_u \cdot \phi_u, \quad F = \phi_u \cdot \phi_v, \quad G = \phi_v \cdot \phi_v.$$

This definition does not strictly depend on the representation chosen, but only on the surface considered, as demonstrated in [103]. The first fundamental form can be used to calculate arc length, angles, and surface areas, via the coefficients E, F , and G .

To define the second fundamental form, let us consider a coordinate patch of class C^m with $m \geq 2$. This allows us to determine the normal to each point of the coordinate patch ϕ , i.e., $\mathbf{N} = \frac{\phi_u \times \phi_v}{\|\phi_u \times \phi_v\|}$, which is a function of u and v of class at least C^1 , with the differential equal to $d\mathbf{N} = \mathbf{N}_u du + \mathbf{N}_v dv$.

Definition 1.1.14 (Second fundamental form). We define the *second fundamental form* of ϕ as the quantity $II = -d\phi \cdot d\mathbf{N}$, which in more explicit terms is expressed as

$$II = Ldu^2 + 2Mdudv + Ndv^2 = II(du, dv)$$

with

$$L = -\phi_u \cdot \mathbf{N}_u, \quad M = -\frac{1}{2}(\phi_u \cdot \mathbf{N}_v + \phi_v \cdot \mathbf{N}_u), \quad N = -\phi_v \cdot \mathbf{N}_v.$$

The second fundamental form is invariant with respect to the chosen parametrization, minus the sign.

1.1.2.2 Normal curvature

The two fundamental forms are used to determine the normal curvature, which in turn allows the principal curvatures to be derived. Let us consider a curve C of parametrization $\boldsymbol{\gamma} = \boldsymbol{\gamma}(u(t), v(t))$ of class at least C^2 passing through a point \mathbf{p} , belonging to the appropriate coordinate patch of S containing \mathbf{p} , at least of class C^2 . In this case, the *curvature normal vector* to C in \mathbf{p} is defined by $\mathbf{k}_n = (\mathbf{k} \cdot \mathbf{N})\mathbf{N}$, where \mathbf{k} is curvature vector of C in \mathbf{p} , that is, the vector associated to \mathbf{p} with respect to C that lies on the plane normal to the curve in \mathbf{p} and points in the direction in which it "continues", and \mathbf{N} is the normal in \mathbf{p} .

Definition 1.1.15 (Normal curvature). We define *normal curvature* of C in \mathbf{p} the quantity

$$\kappa_n = \mathbf{k} \cdot \mathbf{N}$$

where in this case the sign depends on the sign of \mathbf{N} .

Considering the definitions for tangent curves \mathbf{t} and curvature vector \mathbf{k} (see [103]), we obtain an alternative writing of κ_n that exploits the coefficients of I and II :

$$\kappa_n = \mathbf{k} \cdot \mathbf{N} = \frac{L(du/dt)^2 + 2M(du/dt)(dv/dt) + N(dv/dt)^2}{E(du/dt)^2 + 2F(du/dt)(dv/dt) + G(dv/dt)^2}.$$

This depends strictly on the direction of the tangent line to C at \mathbf{p} (i.e., $(du/dt)/(dv/dt)$). The only other dependence for this quantity is on \mathbf{p} .

The following theorem can be formulated.

Theorem 1.1.1. *All curves passing through the point \mathbf{p} that are tangent to the same line through \mathbf{p} have the same normal curvature in \mathbf{p} .*

In conclusion, since the normal curvature to C in \mathbf{p} depends only on \mathbf{p} and the direction of the tangent to C in \mathbf{p} , we can talk about normal curvature in \mathbf{p} in the direction given by $(dv/dt, du/dt)$ and thus

$$\kappa_n = \frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2} = \frac{II}{I}.$$

The latter changes sign according to II , being I defined positive. This identity also allows us to state that κ_n is invariant with respect to parametric transformations that maintain the sign of \mathbf{N} (or that change sign if they reverse that of \mathbf{N}).

1.1.2.3 Principal curvatures

In order to determine the principal curvatures, we assume that a neighborhood of a point \mathbf{p} can be described in the form

$$ue_1 + ve_2 + f(u, v)e_3,$$

which does not alter κ_n , thanks to the invariant properties of the fundamental forms discussed previously and given the regularity of the surface assumed. With this parametrization we translate \mathbf{p} into the origin of the reference system, and the plane x_1x_2 becomes the one tangent to the surface at \mathbf{p} . Referring to the definitions of I and II and in particular their coefficients, we obtain the simplified form:

$$\kappa_n = \frac{Ldu^2 + 2Mdudv + Ndv^2}{du^2 + dv^2}.$$

Assuming that $du^2 + dv^2 = 1$, since κ_n depends solely on the ratio du/dv , and imposing $du = \cos\theta$ and $dv = \sin\theta$, we have:

$$\kappa_n = L \cos^2\theta + 2M \cos\theta \sin\theta + N \sin^2\theta.$$

Finally, imposing $|\kappa_n| = 1/r^2$, $x_1 = r \cos\theta$ and $x_2 = r \sin\theta$ we have

$$\pm 1 = Lx_1^2 + 2Mx_1x_2 + Nx_2^2$$

which determines a conic section on x_1x_2 called *Dupin's indicatrix*.

Thanks to these indicators, shown in Figure 1.1, in the case where they exist and are not circles, we can identify two values, called *principal curvatures* and denoted by κ_1 and κ_2 , equal to

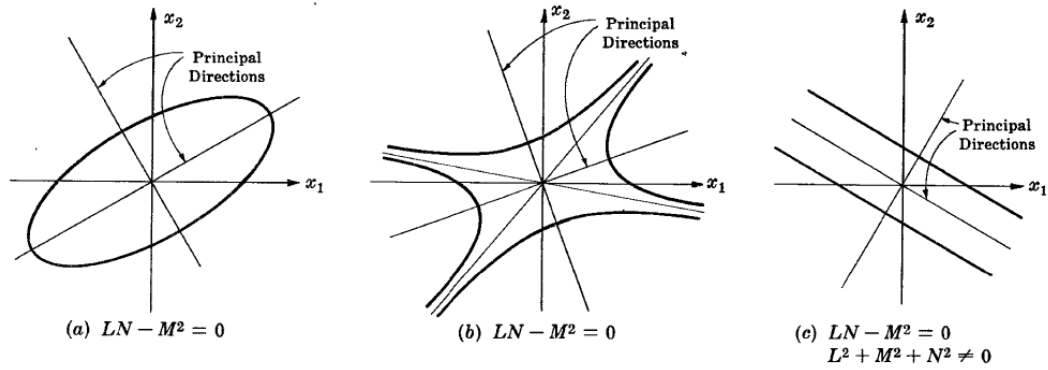


Figure 1.1: Examples of Dupin's indicatrix. In each of the cases, the principal directions and the corresponding value for the discriminant $LN - M^2$ are given. In (a) the indicatrix is a circle, in (b) a pair of conjugate hyperbolas, and in (c) a pair of parallel lines.

the maximum and minimum values for κ_n , respectively. The directions corresponding to these values are called *principal directions*. Knowledge of principal curvatures allows us to make a classification of the points of a surface, distinguishing them into elliptic, parabolic and hyperbolic points, which allows us to tie the curvature of a surface to the local nature of the surface. In cases where there is no indicator or it is a circumference, all directions are considered principal.

To conclude the discussion, we report a result that allows the theoretical calculation of principal curvatures through the use of fundamental forms.

Theorem 1.1.2. *A value κ is a principal curvature if and only if it is a solution of*

$$(EG - F^2)\kappa^2 - (EN + GL - 2FM)\kappa + (LN - M^2) = 0.$$

1.1.2.4 Mean and Gaussian curvature

From the principal curvatures, the mean curvature and Gaussian curvature can be derived.

Definition 1.1.16 (Mean and Gaussian curvature). *The mean curvature and the Gaussian curvature of an S surface in \mathbf{p} correspond, respectively, to the values*

$$H = \frac{\kappa_1 + \kappa_2}{2} = \frac{EN + GL - 2FM}{2(EG - F^2)}.$$

$$K = \kappa_1 \kappa_2 = \frac{LN - M^2}{2(EG - F^2)}.$$

Figure 1.2 shows how the surface varies at the point \mathbf{p} considered with respect to the value assumed by the two curvatures.

Following a roto-translation of the surface, the maximum and minimum curvature can be reversed. Since Gaussian curvature is defined as the product of principal curvatures, it is invariant by roto-translation because the product of signs remains constant, while the mean


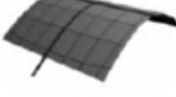


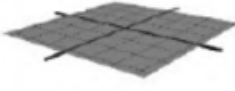

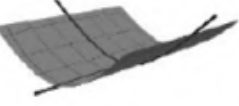

	$K < 0$	$K = 0$	$K > 0$
$H < 0$	Saddle (Ridge) 	Ridge 	Peak 
$H = 0$	Saddle 	Planar 	Not possible
$H > 0$	Saddle (Valley) 	Valley 	Pit 

Figure 1.2: The "classical" shapes of a surface according to the values of mean H and Gaussian K curvature.

curvature can change in absolute value. This transformation is an example of isometry, then the Gaussian curvature is an intrinsic property of the surface.

1.2 Computational representation and properties of curves and surfaces

As a first step toward a mathematical description of an object, one therefore defines a coordinate system in which it will be described analytically. Indeed, the main tool for the development of general results is the use of local coordinate systems, in terms of which geometric properties are easily described and studied. This step allows us to represent and work analytically on the elements introduced in the previous section.

1.2.1 Notations

Let us consider a set of multivariate real functions $F_j : \mathbb{R}^{k+n} \rightarrow \mathbb{R}$, $j = 1, \dots, m$, that depend on two sets of variables $\mathbf{x} = (x_1, \dots, x_k)$ and $\mathbf{a} = (a_1, \dots, a_n)$. Replacing \mathbf{x} and \mathbf{a} , respectively, by points $\mathbf{p} \in \mathbb{R}^k$ and $\mathbf{A} \in \mathbb{R}^n$, two new sets of functions are originated $F_{j,\mathbf{p}}(\mathbf{a}) := F_j(\mathbf{p}, \mathbf{a})$, $F_{j,\mathbf{p}} : \mathbb{R}^n \rightarrow \mathbb{R}$, being continuous on an open convex set $U' \in \mathbb{R}^n$, and $F_{j,\mathbf{A}}(\mathbf{x}) := F_j(\mathbf{x}, \mathbf{A})$, $F_{j,\mathbf{A}} : \mathbb{R}^k \rightarrow \mathbb{R}$, being continuous on an open convex set $U \in \mathbb{R}^k$. Then, there are two families of zero loci $\mathcal{F} = \{\mathcal{C}_{\mathbf{A}}\}_{\mathbf{A} \in \mathbb{R}^n}$

and $\mathcal{H} = \{\Gamma_{\mathbf{p}}\}_{\mathbf{p} \in \mathbb{R}^k}$, where:

$$\mathcal{C}_{\mathbf{A}} = \{\mathbf{x} \in U \mid F_{j,\mathbf{A}}(\mathbf{x}) = 0, j = 1, \dots, m\},$$

$$\Gamma_{\mathbf{p}} = \{\mathbf{a} \in U' \mid F_{j,\mathbf{p}}(\mathbf{a}) = 0, j = 1, \dots, m\}.$$

To simplify the notations, from here on we denote $\mathcal{F} = \{\mathcal{C}_{\mathbf{a}}\}$ and we consider

- $m = 1, k = 2$, assuming that the zero loci $\mathcal{C}_{\mathbf{a}}$ describes a plane curve;
- $m = 1, k = 3$, assuming that the zero loci $\mathcal{C}_{\mathbf{a}}$ describes a surface;
- $m = 2, k = 3$, assuming that the zero loci $\mathcal{C}_{\mathbf{a}}$ describes a space curve.

1.2.2 Mathematical representations in case of codimension 1 in the plane and in the space

Terminology, definitions and concepts on hypersurfaces are presented in this section. Following the notation introduced in Section 1.3.1, a hypersurface in the plane ($k = 2$) or in the space ($k = 3$) is defined as the zero locus \mathcal{Z} of a real function F ,

$$\mathcal{Z} = \{\mathbf{x} \in \mathbb{R}^k \mid F(\mathbf{x}) = 0\}.$$

In order to avoid pathological cases, the same hypothesis as [151] are adopted, that is we assume that the sets of *regular* points of F are dense in \mathcal{Z} and the function F is smooth.

Another way for representing a hypersurface is to use a parametric representation. In case the hypersurface is a curve in the plane, it is described by the set of points $\{(f_1(t), f_2(t)), t \in I\}$, whose components f_1 and f_2 are continuous functions of a common variable $t \in I$, with I an interval of \mathbb{R} . If the hypersurface represents a surface in the space, its parametric representation is given by $\{(f_1(t, u), f_2(t, u), f_3(t, u)), (t, u) \in \Omega\}$, whose components f_1, f_2 and f_3 are continuous functions of common variables $(t, u) \in \Omega$, with Ω a subset of \mathbb{R}^2 . We can think of a parametric curve as a map from a straight line with points t to a curve in the (x, y) -plane and of a parametric surface as a map from a plane with points (t, u) to a surface in the (x, y, z) -space. As said in [78], the class of parametric algebraic curves and surfaces is smaller than the class of implicit algebraic curves and surfaces, then the transition from implicit to parametric equations and vice versa is generally not simple or impossible to solve. Under certain assumptions, it is possible to switch from a parametric representation to an implicit one and vice versa. In case of plane curve, according to the implicit-function theorem and the notations and definitions introduced in [150](Ch.1), if it is m -*regular* (see Definition 1.1.8) with respect to a parametrization, the equation $x = f_1(t)$ is solvable with respect to t in an appropriate open set, therefore $t = t(x)$. Vice versa, if F is a smooth function and

$$F(\mathbf{x}_0) = 0, F_x^2(\mathbf{x}_0) + F_y^2(\mathbf{x}_0) > 0$$

at same point \mathbf{x}_0 , it is well known that the fulfillment of these conditions is sufficient for expressing one variable as a function of another. In particular, if $F_y(\mathbf{x}_0) \neq 0$, then it exists a solution of the equation $F(\mathbf{x}) = 0$ in the form $y = y(x)$ in a neighborhood of \mathbf{x}_0 . Therefore, we may choose the coordinate x , as a parameter of this curve.

In the case of surface, if it is regular (see Definition 1.1.11) with respect to a parametrization, and specifically $f_{1_t}f_{2_u} - f_{1_u}f_{2_t} \neq 0$, then the system

$$\begin{cases} x = f_1(t, u) \\ y = f_2(t, u) \end{cases}$$

can be solvable with respect to t and u

$$\begin{cases} t = t(x, y) \\ u = u(x, y) \end{cases}$$

for the implicit-function theorem. Vice versa, according to the implicit-function theorem, if F is smooth and

$$F(\mathbf{x}_0) = 0, F_x^2(\mathbf{x}_0) + F_y^2(\mathbf{x}_0) + F_z^2(\mathbf{x}_0) > 0,$$

in particular imposing $F_z(\mathbf{x}_0) \neq 0$, then it exists a solution of the equation $F(\mathbf{x}) = 0$ in the form $z = z(x, y)$ in a neighborhood of \mathbf{x}_0 . Therefore, we may choose the coordinates x and y as parameters of this surface.

1.2.3 Mathematical representations in case of codimension 2 in the space

In this section, terminology, definitions and concepts on space curves are presented, which differ from the case of hypersurfaces in the affine space since space curves have codimension 2. According to the terminology introduced before, a space curve in implicit form is represented as the zero loci \mathcal{Z} of two real functions F_1 and F_2 ,

$$\mathcal{Z} = \{\mathbf{x} \in \mathbb{R}^3 \mid F_1(\mathbf{x}) = 0, F_2(\mathbf{x}) = 0\}.$$

To avoid pathological cases, we adopt the same hypothesis as [151], therefore let us assume that the functions F_1 and F_2 are smooth (or at least there exist first- and second- derivatives at every point) and the sets of *regular* points of (F_1, F_2) are dense in \mathcal{Z} . Note that, an implicit representation is not unique as a space curve can be represented as the intersection of many pairs of surfaces. Specifically, a space curve is set-theoretic complete intersection of three surfaces, two of which are cylinders (see [17, 27, 143]).

Another common method for representing a space curve is to use a parametric representation. A space curve in parametric form is described by the set of points $\{(f_1(t), f_2(t), f_3(t)) \mid t \in I\}$, whose components f_1 , f_2 and f_3 are continuous functions of a common variable $t \in I$, with I an interval of \mathbb{R} . The implicit and parametric curve representations are not interchangeable and the transition

from implicit to parametric equations and vice versa is generally not simple or impossible to solve, at least with elementary procedures [78]. Algebraic curves offer a remarkable class of curves for which it is possible to switch from a parametric representation to an implicit one [157]. In general, according to the implicit-function theorem and the notations and definitions introduced in [150](Ch.3), if a curve is *m-regular* (see Definition 1.1.8) with respect to a parametrization, the equation $x = f_1(t)$ is solvable with respect to t in an appropriate open set, therefore $t = t(x)$. Vice versa, the fulfillment of the inequality

$$\begin{vmatrix} F_{1_x}(x_0, y_0, z_0) & F_{1_y}(x_0, y_0, z_0) \\ F_{2_x}(x_0, y_0, z_0) & F_{2_y}(x_0, y_0, z_0) \end{vmatrix} \neq 0,$$

with (x_0, y_0, z_0) a point such that $F_1(x_0, y_0, z_0) = 0$ and $F_2(x_0, y_0, z_0) = 0$, is sufficient for expressing any two variables as a function of the third (again for the implicit-function theorem) in a neighborhood of (x_0, y_0, z_0) . Therefore, we may choose one of the coordinates, e.g., x , as a parameter of this curve

$$y = y(x), z = z(x).$$

However, even when possible, the algorithmic conversion from an implicit to a parametric form is, in general, computationally expensive [78]. Depending on the task, a parametric representation is best suited for generating points along a curve, whereas an implicit representation is most convenient for determining whether a given point lies on a specific curve [146]. These facts motivate the definition of a recognition method able to handle both implicit and parametric curve representations.

1.2.4 Geometric properties

In the literature, methods for evaluating curvatures are distinguished into three main groups: the first goes from discrete to continuous, approximating locally with a smooth surface to which one can apply the definitions given in Section 1.1.2; the second denotes descriptors from the characteristics of the operators used in the continuous case, applying them directly in the discrete and the third looks for discrete approximations of differential properties of surfaces, from which the curvature can be derived, such as normal tensors, see [65].

The computation of curvatures on a discrete surface representation such as a triangular mesh focuses mainly on the edges and vertices of triangles, since for all other points one can find a neighborhood homeomorphic to a planar object, thus with null curvature. Regarding the point clouds, methods to estimate the curvatures are often based on moving least squares (MLS) fitting of algebraic surfaces. We refer to [65, 107] and [156] for an overview of methods for curvature approximation and to [112] for a detailed comparison among eight representatives of the various curvature estimator strategies.

Most of these methods are designed only for triangle meshes and not for point clouds, but there are many algorithms in the literature to obtain a mesh from a cloud (see for example [22]). For more details on curvature approximation strategies in point clouds, we refer to [48].

In our experiments, regarding the estimation of curvatures on meshes, we use the *Toolbox Graph* [121], which approximates curvature through the theory of normal cycles [38]. As regards the computation of curvatures on point clouds, we use the Algebraic Point Set Surface fitting method [69] provided by Meshlab [37].

1.2.5 Colorimetric properties

The choice of a colour-related description was made with respect to a theory related to human perception of colour, the so-called *colour space Lab* (or *CIELAB*) [85]. The space in question is colour-opponent: in practice, the distribution in a colour space the "opposite" hues are in opposite positions in the representation space. For example, thinking of distributing colours in a plane disk, we define two or more equi-spaced poles on it. On each pair of opposite poles one colour, e.g. red, is on one side and its opposite, e.g. green, is on the other. The definition of "opposite colours" and the choice of space is what differentiate the various colour-opponent spaces.

The strength of CIELAB space is its excellent performance in merging (or distancing) colours that according to human perception are similar (or dissimilar). For example, unlike the RGB colour system, colours with close coordinates in CIELAB space are perceptually similar, whereas the same is not necessarily true in RGB space [9]. The way colours are considered opposites for CIELAB is illustrated in Figure 1.3¹.

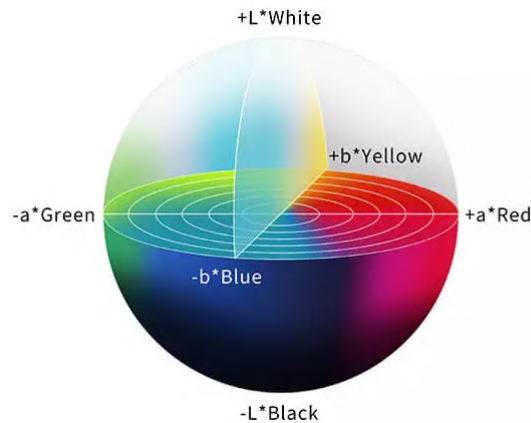


Figure 1.3: The distribution of colours considered "opposites" according to CIELAB.

Given a colour, the coordinates (or channels) of the CIELAB space are L, which is similar to the "human" concept of luminosity, a, which denotes the amount of red/green, and b, which denotes the amount of yellow/blue. As can be seen in Figure 1.4, RGB space is distributed in CIELAB space so that the opposite colours, according to this colour interpretation, are as far apart as possible and vice versa.

¹<https://www.linshangtech.com/tech/color-space-tech1439.html>

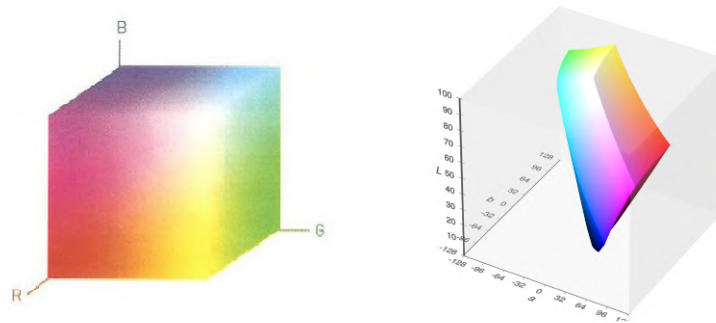


Figure 1.4: RGB space colours (left) represented according to CIELAB coordinates (right).

In some experiments in this thesis, the channel L is used as the surface property that identifies a decorative pattern.

1.3 The Hough transform for families of curves and surfaces

The Hough transform has been introduced in 1962 [81] originally to recognise straight lines in images. Then it has been extended to circles and ellipses [53] and later generalised to template curves even without explicit curve representation [14]. One of the salient features of HT is its ability to recognise one or more instances of a shape, even partial or in presence of noise, within an image. This has led the HT to become, for dozen years, one of the basic techniques of pattern recognition, regularly taught in many computer vision courses. A key turning point is the seminal work [20] that extended the HT to algebraic objects, thus enabling the use of the HT to larger classes of curves, surfaces and hypersurfaces. Notice that the HT recognition techniques in the literature generally deal with hypersurfaces of codimension 1, i.e., plane curves in images, surfaces in space, etc. [19]. The use of HT for space curves – then the approach to elements of codimension 2 in the space – is still in its infancy and it has been addressed only in the last few years.

1.3.1 Preliminary concepts on the HT

The problem to define a method for recognising a (plane/space) curves or surfaces on objects represented as point clouds or meshes can be formalised as follows: let \mathcal{D} be a curve or surface of interest embedded in \mathbb{R}^k , $k = 2, 3$ and $\mathcal{P} \subset \mathcal{D}$ a set of L points, i.e. $\mathcal{P} = \{\mathbf{p}_l \in \mathcal{D}, l = 1, \dots, L\}$, $L \gg n$, the aim is to find the curve or the surface that best fits \mathcal{P} . The Hough transform (HT) yields the formalism and the basis for such a method.

The original HT definition is based on the concept of *point-line duality*, which is formalized in the affine plane by the classical duality in the projective plane. It can be summarised as follows: points on a straight line, defined by an equation, correspond to lines in the parameter space that

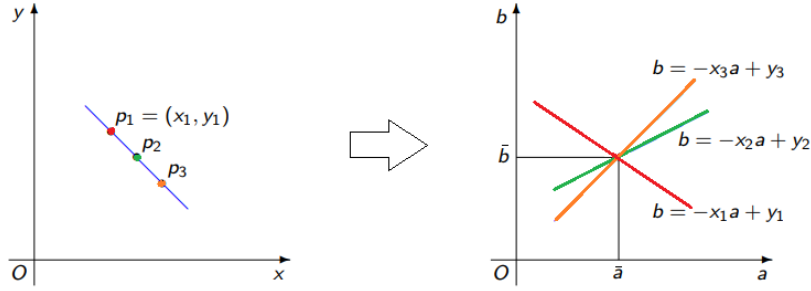


Figure 1.5: An illustrative example of the point-line duality.

intersect at a single point (see Figure 1.5). This point uniquely identifies the coefficients in the equation of the original straight line.

The duality concept extends to curves and surfaces, both in implicit [152] and parametric form [20]. In these works, the HT formulation is expressed either in the field of complex numbers \mathbb{C} or the field of real numbers \mathbb{R} and in affine spaces over these fields; being mainly interested into applications, in this thesis, we restrict our notation to \mathbb{R} and on real spaces. Specifically, given a family $\mathcal{F} = \{\mathcal{C}_{\mathbf{a}}\}$ of curves or surfaces that depend on a set of parameters $\mathbf{a} = (a_1, \dots, a_n) \in U' \subset \mathbb{R}^n$, U' an open set of \mathbb{R}^n , a general point \mathbf{p} in the plane/space corresponds to a locus $\Gamma_{\mathbf{p}}$ in the parameter space U' .

Definition 1.3.1. Fixed the point \mathbf{p} , $\Gamma_{\mathbf{p}}$ is called the *Hough transform* of \mathbf{p} with respect to the family \mathcal{F} .

Lemma 1.3.1 (Key Lemma). *The following conditions are equivalent:*

- for any $\mathcal{C}_{\mathbf{a}}, \mathcal{C}_{\mathbf{a}'}$, we have $\mathcal{C}_{\mathbf{a}} = \mathcal{C}_{\mathbf{a}'} \implies \mathbf{a} = \mathbf{a}'$;
- for any $\mathcal{C}_{\mathbf{a}}$, we have $\bigcap_{\mathbf{p} \in \mathcal{C}_{\mathbf{a}}} \Gamma_{\mathbf{p}} = \{\mathbf{a}\}$.

Definition 1.3.2. If a family \mathcal{F} satisfies one of the two equivalent conditions of the Key Lemma it is called *Hough regular*.

The duality concept is fundamental for the HT-based recognition algorithm. In the case the functions F_j are polynomials, it is possible to explicitly verify if the family is Hough regular.

Note that, in practice, the exact intersection is not infinite and $\bigcap_{\mathbf{p} \in \mathcal{D}} \Gamma_{\mathbf{p}} = \emptyset$, since the real data can be affected by noise or can be inaccurate. Then, we relax the request and compute an approximation of $\bigcap_{\mathbf{p} \in \mathcal{D}} \Gamma_{\mathbf{p}} \approx \{\bar{\mathbf{a}}\}$, where $\bar{\mathbf{a}}$ identifies the curve or the surface $\mathcal{C}_{\bar{\mathbf{a}}} \in \mathcal{F}$ that fits \mathcal{D} . Indeed, the HT translates the recognition problem into detecting which value of the parameters that determine the family \mathcal{F} corresponds to the curve or surface that best fits a given set of points (such a value may be non-unique) through a voting procedure (see details in Section 1.3.4).

1.3.2 Mathematical representations of the HT in case of codimension 1 in the plane and in the space

Following the notations introduced in Section 1.2, to analytically derive the HT for implicit and parametric representations, let us initially consider the family \mathcal{F} with $\mathcal{C}_{\mathbf{a}}$ given in the implicit form:

$$\mathcal{C}_{\mathbf{a}} : F_{\mathbf{a}}(\mathbf{x}) = 0$$

where F is a function with respect to the variables $\mathbf{x} = (x, y)$ in an open set $U \in \mathbb{R}^2$ if $\mathcal{C}_{\mathbf{a}}$ is a plane curve, while $\mathbf{x} = (x, y, z)$ in an open set $U \in \mathbb{R}^3$ if $\mathcal{C}_{\mathbf{a}}$ is a surface. For each point \mathbf{p} , placing $F_{\mathbf{a}}(\mathbf{x}_{\mathbf{p}}) = F_{\mathbf{p}}(\mathbf{a})$, the HT of the point \mathbf{p} with respect to the family \mathcal{F} is given by the implicit equation

$$(1.1) \quad \Gamma_{\mathbf{p}} : F_{\mathbf{p}}(\mathbf{A}) = 0$$

with $F_{\mathbf{p}}$ a function in an open set of the space of the parameters U' in \mathbb{R}^n with respect to the variables $\mathbf{A} = (A_1, \dots, A_n)$. In practice, the estimation of $\Gamma_{\mathbf{p}}$ can be seen as the estimation of the zero loci of the system in equation (1.1).

Similarly, let us consider the family \mathcal{F} with $\mathcal{C}_{\mathbf{a}}$ given in the parametric form:

$$\mathcal{C}_{\mathbf{a}} : \begin{cases} f_1(\mathbf{a}, \mathbf{t}) \\ \vdots \\ f_k(\mathbf{a}, \mathbf{t}) \end{cases} \quad \mathbf{t} \in U,$$

where f_1, \dots, f_k are continuous functions in \mathbf{t} , with $k = 2$ and U an open set of \mathbb{R} in case of plane curve, while $k = 3$ and U an open set of \mathbb{R}^2 in case of surface. If the curve equation system admits an analytical solution, the analytic expression of $\Gamma_{\mathbf{p}}$ can be derived directly from the parametric equations of \mathcal{F} :

$$\Gamma_{\mathbf{p}} : \begin{cases} A_1 = h_1(\mathbf{x}_{\mathbf{p}}, \mathbf{t}) \\ \vdots \\ A_n = h_n(\mathbf{x}_{\mathbf{p}}, \mathbf{t}) \end{cases} \quad \mathbf{t} \in U,$$

with h_1, \dots, h_n continuous functions in \mathbf{t} . Note that $\Gamma_{\mathbf{p}}$ is expressed in terms of the same parameter \mathbf{t} that appears in the equation of $\mathcal{C}_{\mathbf{a}}$.

1.3.3 Mathematical representations of the HT in case of codimension 2 in the space

Following the notations introduced in Section 1.2, to analytically derive the HT for implicit and parametric curve representations, let us initially consider the traditional system of Cartesian

coordinates (x, y, z) and the family of curves \mathcal{F} with $\mathcal{C}_{\mathbf{a}}$ given in the implicit form:

$$\mathcal{C}_{\mathbf{a}} : \begin{cases} F_{1,\mathbf{a}}(x, y, z) = 0 \\ F_{2,\mathbf{a}}(x, y, z) = 0 \end{cases}$$

with $F_{1,\mathbf{a}}$ and $F_{2,\mathbf{a}}$ functions with respect to the variables (x, y, z) in an open set $U \in \mathbb{R}^3$. For each point \mathbf{p} , placing $F_{1,\mathbf{a}}(x_{\mathbf{p}}, y_{\mathbf{p}}, z_{\mathbf{p}}) = F_{1,\mathbf{p}}(\mathbf{a})$ and $F_{2,\mathbf{a}}(x_{\mathbf{p}}, y_{\mathbf{p}}, z_{\mathbf{p}}) = F_{2,\mathbf{p}}(\mathbf{a})$, the HT of the point \mathbf{p} with respect to the family \mathcal{F} is given by the implicit equation

$$(1.2) \quad \Gamma_{\mathbf{p}} : \begin{cases} F_{1,\mathbf{p}}(\mathbf{A}) = 0 \\ F_{2,\mathbf{p}}(\mathbf{A}) = 0 \end{cases}$$

with $F_{1,\mathbf{p}}$ and $F_{2,\mathbf{p}}$ functions in an open set of the space of the parameters U' in \mathbb{R}^n with respect to the variables $\mathbf{A} = (A_1, \dots, A_n)$. In practice, the estimation of $\Gamma_{\mathbf{p}}$ can be seen as the estimation of the zero loci of the system in equation (1.2).

Similarly, let us consider the family of curves \mathcal{F} with $\mathcal{C}_{\mathbf{a}}$ given in the parametric form:

$$(1.3) \quad \mathcal{C}_{\mathbf{a}} : \begin{cases} x = f_1(\mathbf{a}, t) \\ y = f_2(\mathbf{a}, t) \\ z = f_3(\mathbf{a}, t) \end{cases} \quad t \in U,$$

with U an open set of \mathbb{R} and $f_j, j = 1, 2, 3$, continuous functions in t . If the curve equation system admits an analytical solution, the analytic expression of $\Gamma_{\mathbf{p}}$ can be derived directly from the parametric curve equations of \mathcal{F} :

$$(1.4) \quad \Gamma_{\mathbf{p}} : \begin{cases} A_1 = h_1(\mathbf{x}_{\mathbf{p}}, t) \\ \vdots \\ A_n = h_n(\mathbf{x}_{\mathbf{p}}, t) \end{cases} \quad t \in U,$$

with h_1, \dots, h_n continuous functions in t . Note that $\Gamma_{\mathbf{p}}$ is expressed in terms of the same parameter t that appears in the equation of the curve $\mathcal{C}_{\mathbf{a}}$.

Section 3.3 details how the Hough transform expressed either in the form of equation (1.2) or equation (1.4) can be estimated for space curves.

1.3.4 Voting procedure

Given a set of points $\mathcal{P} = \{\mathbf{p}_l \in \mathcal{D}, l = 1, \dots, L\}$ and once a family \mathcal{F} is selected, the generalised HT-based recognition method ([18, 153]) can be summarised in three main steps.

1. *Initialisation of the HT space parameters and the accumulator function.* A region \mathcal{T} of the parameter space of \mathcal{F} is identified by exploiting the geometric characteristics of the

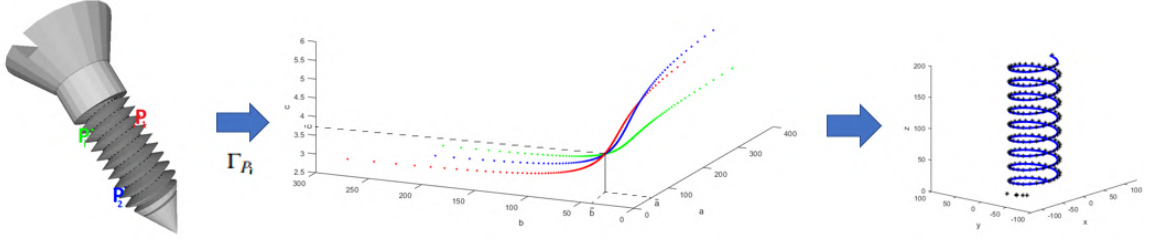


Figure 1.6: An illustrative example of the voting procedure, considering the profile that outlines the threads of a screw (on the left); in the middle, the Hough transforms of points in the space abc ; on the right, the recognised curve corresponding to the coefficient \bar{a} , \bar{b} and \bar{c} identified by the intersection point in the abc space.

chosen family. Let n be the number of parameters that characterise the curve family \mathcal{F} , \mathcal{T} is sampled as follows:

$$a_{j,k_j} = a_j^* + k_j \delta_j, \quad k_j = 0, \dots, N_j - 1; \quad j = 1, \dots, n$$

where δ_j are the sampling distances from j -th component of \mathbf{a} , $\mathbf{a}^* = (a_1^*, \dots, a_n^*)$ an estimate of the parameters and N_j indicates the number of samples considered for the j -th component. Then, the cell centred in one of the points a_{j,k_j} is denoted as follows:

$$c(k_1, \dots, k_n) = \prod_{j,k_j} [a_{j,k_j} - \frac{\delta_j}{2}, a_{j,k_j} + \frac{\delta_j}{2}).$$

Note that this notation is coherent with the notation used in [109] for families of curves with two parameters and extends to curves with n parameters.

Assuming the set \mathcal{P} is made of L points, the HT accumulator function $\mathcal{H} := \mathcal{H}^{(L)}$ of dimension $\prod_{j=1}^n N_j$ is defined and initialised as $\mathcal{H}_{k_1, \dots, k_n} = 0$ for each $k_j = 0, \dots, N_j$, $j = 1, \dots, n$.

2. *Estimation of the accumulator function.* The estimation of the accumulator function \mathcal{H} , in the discretized form of a matrix, is crucial because every entry of this matrix corresponds to the number of times that $\Gamma_{\mathbf{p}_l}$, varying $\mathbf{p}_l \in \mathcal{P}$, crosses the corresponding cell of the discretization. Therefore, the ideal solution to the HT problem is to detect the cell of \mathcal{H} that is crossed by the greatest number of points \mathbf{p}_l . The Hough transforms of a set of points \mathcal{P} are then evaluated and the matrix \mathcal{H} is updated by placing each entry of the matrix equal to the number of transforms passing through the cell corresponding to this entry:

$$\mathcal{H}_{k_1, \dots, k_n}^{(l)} = \begin{cases} \mathcal{H}_{k_1, \dots, k_n}^{(l-1)} + 1 & \text{if } (\Gamma_{\mathbf{p}_l} \cap c(k_1, \dots, k_n)) \neq \emptyset \\ \mathcal{H}_{k_1, \dots, k_n}^{(l-1)} & \text{if } (\Gamma_{\mathbf{p}_l} \cap c(k_1, \dots, k_n)) = \emptyset. \end{cases}$$

To estimate \mathcal{H} , two different approaches are used, depending on whether the family \mathcal{F} is represented in parametric or implicit form.

3. *Identification of the potential fitting curve or surface.* Given the accumulator matrix \mathcal{H} , let us consider $\bar{\mathbf{a}} = \operatorname{argmax}_{k_1, \dots, k_n} \mathcal{H}_{k_1, \dots, k_n}$, which identifies the parameters of the curve or the surface $\mathcal{C}_{\bar{\mathbf{a}}}$ within the family that best approximates the set of points \mathcal{P} . Local maxima can be non-unique, and then more solutions can be found by the HT. In this case, different quality measures are used to determine the best one.

An illustrative example of this procedure is provided in Figure 1.6. Specifically, considering a 3D model of a screw and a set of points that characterise one of its threads it is possible to recognise the profile by selecting the family of cylindrical helices (see Section 3.2). As shown in Figure 1.6, to each point of the profile corresponds a curve on the parameter space. The coordinates of the intersection point of these curves correspond to the coefficients of the equation of the recognised cylindrical helix.

RECOGNITION, EXTRACTION AND REPRESENTATION OF PLANE CURVES

The characterisation and recognition of curve patterns on surfaces is a well-known problem in computer graphics. Feature curves are useful for visual shape illustration [92] and perception studies support these curves as an effective choice for representing the salient parts of a 3D model [39, 72].

The problem faced in this chapter is focused on the extraction of feature curves on 3D shapes that can be projected onto the regression plane and on images, then the HT-based recognition approach can take advantage of a rich dictionary of families of plane curves [150]. In this context, we first introduce the dictionary available in our template (Section 2.2), made of both simple and compound plane curves, and we detail the main steps of our method (Section 2.3) dealing with implicit and parametric representations. Then, in Section 2.4 we exhibit some tests of our approach on real objects and in Section 2.5 we provide a comparative analysis of the two strategies. Finally, we show two examples of applications that exploit the mathematical representations of curves provided by our algorithm (Section 2.6).

2.1 Previous work

Plane curves fitting. Most methods for feature curve detection focus on the curve fitting problem, i. e., they look for the curve that fits best a profile. There are many existing works for extracting feature curves from meshes. Methods for feature curves detection aim at finding curve segments that represent characteristic shape features (e.g., ridge/valleys, crest lines, sharp lines, demarcating curves, etc.) [34, 39, 71, 77, 92, 94, 161]. Even if these curves often undergo smoothing operations (for instance, based on Laplacian smoothing [77] or energy minimisation

[116]), their profile is often identified by a set of ordered points [34, 77, 161] or by piecewise splines of low degree polynomials, e.g., [10, 58]. However, low degree polynomials cannot span a large number of points, therefore many small segments need to be blended together to build the desired curve, and such a decomposition is not unique, for instance, it depends on the starting point. Due to the ease of construction, spline interpolation is frequently used to approximate data [145]. B-splines are now a popular tool in CAGD and offer a common method for handling spline interpolation [30]. Other generalisations to non-polynomial splines include non-uniform rational B-splines (NURBS) [122] and generalised splines [29], which also permit trigonometric or exponential bases.

Another important class of curves are Pythagorean–hodograph (PH) curves, which have become an active research area in recent years [7, 59, 129]. Finally, regarding the curve fitting it is worth mentioning subdivision schemes [54], that are widely used in various applications such as computer graphics and solid modeling.

Hough Transform for plane curves. Since its original definition for straight line detection [81], the HT has been extensively used for the recognition of circles and ellipses in [53] and then generalised to the identification of non-analytic profiles in images [14]. This generalisation of the HT is able to detect an arbitrary (but fixed) shape using a look-up table to drive the template matching, still retaining the HT robustness. Nevertheless, this HT generalisation adopts a brute force approach that considers all the possible orientations and scales of the input shape. Thus, the number of parameters in its process is considerably high. Further, being based on a single-shape template it cannot adequately handle similar shapes, as in the case of different instances of the same shape, which are comparable but not identical, e.g. petals and leaves. For an extended survey on the HT theory and its extensions, we refer to [114].

The theoretical foundations introduced in [20] and [18] have been laid to extend the HT technique to the detection of special classes of curves whose algebraic forms are known, but significantly more complex than straight lines or conics. Specifically, irreducible algebraic plane curves like elliptic curves, curves with 3 convexities, Wassenaar curves, conchoids of Slüse and piriform curves have been considered and the standard line and conic detection algorithm has been extended to detect these curves [109]. In addition, also collections of different algebraic pieces from the same family or low-degree piecewise polynomial curves have been introduced in [40, 131].

To deal with real data sets, an approximation strategy of the HT for algebraic curves has been defined, for instance, we refer to [109, 153]. There is software freely available for plane curves represented in both parametric form http://mida.dima.unige.it/g_software_htbone.html according to the algorithm [18, 109] and in implicit form <https://github.com/CNR-IMATI/FeatureExtraction> as detailed in [153, 154].

2.2 Families of plane curves

The HT-based recognition procedure looks for the curve $\mathcal{C}_{\mathbf{a}}$ that better fits a set of points \mathcal{P} within a selected family of curves \mathcal{F} . Therefore, we need to define a dictionary of curves in which it is possible to choose the family of curves most appropriate for the recognition process. In this section, we describe the library of families of mathematical plane curves developed in our system. In [154], the first set of families of mathematical curves suitable for recognising features on surfaces was already proposed, namely the *citrus curve*, the *Archimedean spiral*, the *Lamet curve*, the *m-convexities curve*, and the *geometric petal curve* of type (A). This library also includes basic families of curves like straight lines, circles and ellipses. Then, we have extended this dictionary to eight additional families of curves in [137]: the *geometric petal curve* of type (B), the *elliptic curve*, the *lemniscate of Bernoulli*, the *egg of Keplero*, the *mouth curve*, the *astroid* and the *bullet-nose curve*.

In Sections 2.2.1 and 2.2.2 we detail the families of plane curves composing our dictionary. Note that the dictionary is flexible and easily extendable according to application needs by choosing the appropriate families of curves in an atlas such as [150]. We have included those suitable for our case studies.

2.2.1 Simple plane curves

For each curve in our library, we highlight the parameters that drive the HT and show how the knowledge of the main geometric characteristics of each family allows estimating a priori these parameters in which \mathbf{a} varies for the set \mathcal{P} . Indeed, we use the bounding box of the set \mathcal{P} to evaluate which curve parameters would generate a curve tangent to the bounding box. In the following, the lengths of the horizontal and vertical edges of the bounding box of \mathcal{P} are denoted as $D1$ and $D2$. In the same way, we define the minimum and the maximum distances of points from the origin, respectively, $R1$ and $R2$.

These a priori estimates are considered as a kind of centre of the parameter space: a window of the parameter space is automatically selected and opportunely discretized into cells to detect the value of \mathbf{a} corresponding to the best fitting curve. Note that, according to the notation in Section 1.3.1, \mathbf{a} is a vector of parameters: in the examples listed in this section \mathbf{a} has dimension one or two (i.e., one or two scalar parameters).

In the following, for each curve we show its equation, parameters and a representation with specific parameters. When convenient, the implicit representation is in polar coordinates instead of the traditional Cartesian ones.

- The *citrus curve* has the implicit and parametric equations:

$$I : a^4 b^2 y^2 + \left(x - \frac{a}{2}\right)^3 \left(x + \frac{a}{2}\right)^3 = 0, \quad P : \begin{cases} x = t - \frac{a}{2} \\ y = \pm \sqrt{\frac{(a-t)^3 t^3}{a^4 b^2}} \end{cases}$$

with a and b real parameters (with respect to the previous notation $\mathbf{a} = (a, b)$, the double notation in the following will be omitted). It is a symmetric and limited curve, contained in the rectangle $[-\frac{a}{2}, \frac{a}{2}] \times [-\frac{a}{8b}, \frac{a}{8b}]$. The parameters a and b can be estimated through the size of the bounding box of the set of points \mathcal{P} . The parameters a and b are limited by the relation $a \approx D1$ and $b \approx \frac{a}{4D2}$; in particular, b determines how the curve squeezes along the y axis.

- The *Archimedean spiral* is expressed in the form:

$$I : \rho - a - b\theta = 0, \quad P : \begin{cases} x = (a + bt)\cos t \\ y = (a + bt)\sin t \end{cases}$$

with a and b real parameters and the implicit representation in polar coordinates. This curve is connected and not limited, with a singularity at the point $(a, 0)$. Two consecutive turnings of the spiral have a constant separation distance, which is equal to $2\pi b$ and, therefore, the k -th turning of the spiral is contained in a region bounded by two concentric circles of radius $a + 2(k-1)\pi b$ and $a + 2k\pi b$. Therefore the parameters a and b depend on the length of the edges of the bounding box of the set \mathcal{P} and the number of turnings of the spiral k .

- The *Lamé curve* has implicit and parametric equations:

$$I : bx^m + a^m y^m = a^m b, \quad P : \begin{cases} x = at \\ y = \pm (b - bt^m)^{\frac{1}{m}} \end{cases}$$

with $a, b \in \mathbb{R}_{>0}$ and $m \in \mathbb{N}$. It is a curve of degree m and represents a rectangle with rounded corners, which increases when m grows. This curve is connected, closed and equipped with two symmetry axes (the x axis and the y axis). Furthermore, it is contained in a rectangle of edges $[-a, a] \times [-b^{\frac{1}{m}}, b^{\frac{1}{m}}]$. In our applications, the length of the edges of the bounding box of the set \mathcal{P} allows the computation of the parameters a and b as $a \approx \frac{D1}{2}$ and $b \approx (\frac{D2}{2})^m$.

- The *m-convexities curve* is defined in implicit form (in polar coordinates) and in parametric form by the equations:

$$I : \rho = \frac{a}{1 + b \cos(m\theta)}, \quad P : \begin{cases} x = \frac{a}{1 + b \cos(mt)} \cos t \\ y = \frac{a}{1 + b \cos(mt)} \sin t \end{cases}$$

with $a, b \in \mathbb{R}_{>0}$, $b < 1$ and $m \in \mathbb{N}_+$, $m \geq 2$. The parameter a plays the role of the scale factor, while b regulates the fineness of the convexities. This curve is connected and bounded. It has m symmetry axes and it is contained in the region between two concentric circumferences of radius $\frac{a}{1+b}$ and $\frac{a}{1-b}$. To estimate the parameters, we use the maximum and the minimum radius (respectively, $R2$ and $R1$) of the set \mathcal{P} written in polar coordinates.

- The *geometric petal curve* of type (A) is defined in implicit form (in polar coordinates) and in parametric form by the equations:

$$I : \rho = a + b \cos^{2n} \theta, \quad P : \begin{cases} x = (a + b \cos^{2n} t) \cos t \\ y = (a + b \cos^{2n} t) \sin t \end{cases}$$

with $n \in \mathbb{N}_+$ e $a, b \in \mathbb{R}$, which for our purposes is reduced to the case $b = -a$. It is a bounded and symmetrical curve, with a singularity in the origin, and completely contained in a circle of radius $\sqrt{2a}$. Replacing a with c^2 , we get the curve in Cartesian coordinates, with the usual substitution $\rho = \sqrt{x^2 + y^2}$ and $\cos(\theta) = \frac{x}{\sqrt{x^2 + y^2}}$, of equation

$$(x^2 + y^2)^{2n+1} - c^4[(x^2 + y^2)^n - x^{2n}]^2 = 0.$$

This curve is bounded by the rectangle $[-\frac{2n}{2n+1}c^2\sqrt{[2n]\frac{1}{2n+1}}, \frac{2n}{2n+1}c^2\sqrt{[2n]\frac{1}{2n+1}}] \times [0, c^2]$. In this case, it is convenient to use the equation in polar form. Remembering the substitution of a with c^2 , a limit for the parameter a is given by the length of the vertical edge of the bounding box of the set \mathcal{P} , therefore $a \approx D2$.

- The *geometric petal* of type (B) is defined in implicit form (in polar coordinates) and in parametric form by the equations:

$$I : \rho = a + b \cos 2n\varphi, \quad P : \begin{cases} x = (a + b \cos 2nt) \cos t \\ y = (a + b \cos 2nt) \sin t \end{cases}$$

with $a > 0$, $b > 0$ and $n \in \mathbb{N}$. This curve is contained in a circle with radius $a + b$ and the origin is the center of symmetry, which becomes a singular point if $a \leq |b|$ while if $a > |b|$ there are no singularities. As for the m-convexities curve, the maximum and the minimum radius (respectively, $R2$ and $R1$) of the points written in polar coordinates are used to estimate a and b . Note that this family of curves can deal with an even number of petals, differently from the m-convexities one.

- The *lemniscate of Bernoulli* is an algebraic curve in the form of a lying eight. Its implicit and parametric equations are:

$$I : (x^2 + y^2)^2 = 2a^2(x^2 - y^2), \quad P : \begin{cases} x = a \frac{\sin t}{1 + \cos^2 t} \\ y = a \frac{\sin t \cos t}{1 + \cos^2 t} \end{cases}$$

with a positive real parameter. It is a symmetric and bounded curve, contained in the rectangular region $[-\sqrt{2}a, \sqrt{2}a] \times [-\frac{a}{2}, \frac{a}{2}]$. Then the parameter a can be estimated as $a \approx \frac{\sqrt{2}}{2}D1$. For convenience, in this case we adopt the equation of the curve in polar form

$$\rho^2 = 2a^2 \cos 2\theta.$$

- The *egg of Keplero* has the implicit and parametric equations:

$$I : (x^2 + y^2)^2 = ax^3, \quad P : \begin{cases} x = \frac{a}{(1+t^2)^2} \\ y = \frac{at}{(1+t^2)^2} \end{cases}$$

with $a \in \mathbb{R}$, $a > 0$. It is symmetric with respect to the x -axis and bounded, in particular, it is contained in the rectangle $[0, a] \times [-\frac{3\sqrt{3}}{16}a, \frac{3\sqrt{3}}{16}a]$. Therefore, the value of the parameter a coincides with the length of the horizontal edge of the bounding box of the set \mathcal{P} .

- The *mouth curve* has the implicit and parametric equations:

$$I : a^4 y^2 = (a^2 - x^2)^3, \quad P : \begin{cases} x = a \cos t \\ y = a \sin^3 t \end{cases}$$

with a real positive parameter which is half the length of the mouth. It is a symmetric and bounded curve, contained in the square $[-a, a] \times [-a, a]$. Even in this case, the parameter a is estimated through the length of the edges of the bounding box of the set \mathcal{P} . This curve has a shape similar to the citrus curve, but it extends more along the y -axis.

- The *elliptic curve* has implicit and parametric equations:

$$I : y^2 = x^3 + ax + b, \quad P : \begin{cases} x = \frac{t}{a} \\ y = \pm \sqrt{\frac{t^3}{a^3} + t + b} \end{cases}$$

with a and b real parameters. It is symmetric with respect to the x -axis and unbounded. We only consider the case in which the curve is not singular, i.e., when the determinant $4a^3 + 27b^2$ is positive. The parameters a and b have specific geometric interpretations: the parameter b is the square of the ordinate of the intersection points of the curve with the y -axis, while the parameter a appears in the abscissas of the points of maximum of the semi-curve $y = \sqrt{x^3 + ax + b}$.

- The *bullet-nose curve* has implicit and parametric equations:

$$I : a^2 y^2 - b^2 x^2 = x^2 y^2, \quad P : \begin{cases} x = a \cos t \\ y = b \sin t \end{cases}$$

with $a > 0$ and $b > 0$. It is symmetric with respect to the origin and unbounded. The parameters a and b are estimated through the lengths of the minimum bounding box of the set \mathcal{P} , specifically the horizontal length with regard to a and the vertical one with regard to b .

- The *astroid* is a bounded curve given by the equations

$$I : (x^2 + y^2 - a^2)^3 + 27a^2x^2y^2, \quad P : \begin{cases} x = a \cos^3 t \\ y = a \sin^3 t \end{cases}$$

with $a > 0$. It is contained in the rhombus of vertices $(a, 0)$, $(0, a)$, $(-a, 0)$ and $(0, -a)$. In applications, an estimate of a can be obtained with respect to both the maximum abscissa and the ordinate of the set \mathcal{P} .

Table 2.1 summarises the mathematical expression of the plane curves listed in this section and provides a graphical example for each of them.

2.2.2 Compound plane curves

Combining the simple curves listed in Section 2.2.1, it is possible to recognise complex patterns made of several elements. In these patterns, more families of curves or more occurrences of the same family might be present, possibly with different parameters. We address the multiple occurrences of curves in two different ways, as explained in [135]. The first approach builds a new family of curves by combining the equations of the corresponding curve families. In the case of implicit representation, the new family is built through the product of the equations of the curves composing it. For the parametric representation, the new family of curves is formed by the union of parametric equations of the curves that compose it. The use of the curve product or union is very simple and yields a strategy to enrich the dictionary of curves, but increases the number of parameters. The idea is that the simultaneous detection of two or more curves limits the ambiguities. Figure 2.1(b) shows the outcome of the recognition of a rosette in all its parts in terms of the product of a circle and a geometric petal of type (B) with $a > |b|$. In this case, the family used is defined by the implicit and parametric equations:

$$I : (\rho - a - b \cos 2n\varphi)(\rho - r) = 0, \quad P : \begin{cases} x = (a + b \cos 2nt) \cos t \\ y = (a + b \cos 2nt) \sin t \end{cases} \cup \begin{cases} x = r \cos t \\ y = r \sin t \end{cases}.$$

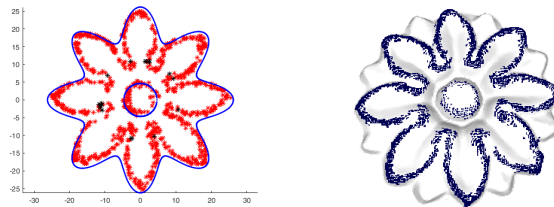
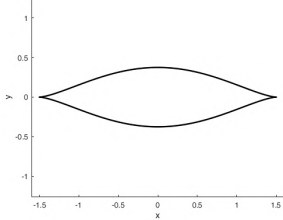
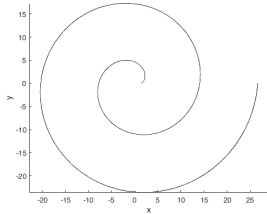
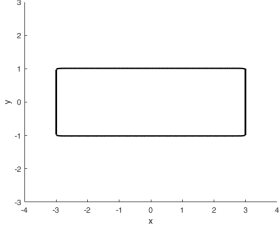
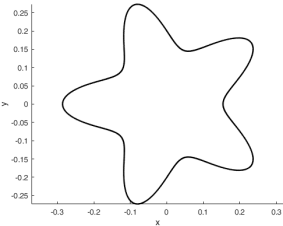
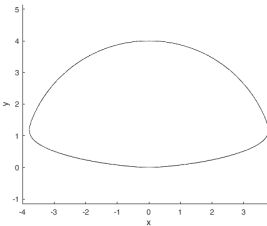
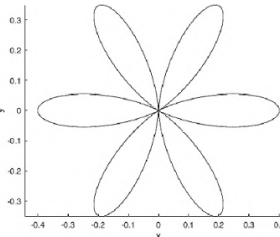
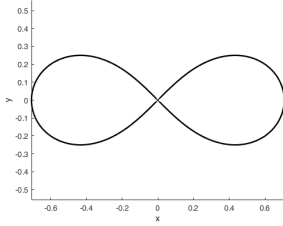
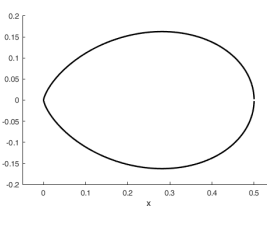
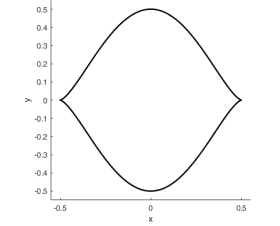
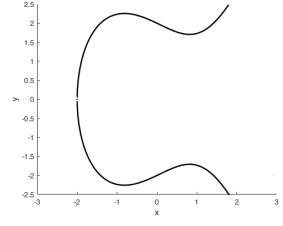
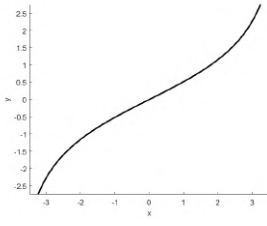
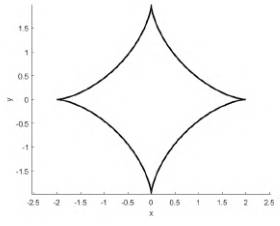


Figure 2.1: An example of recognition of a complex pattern through the product of curves composing it (courtesy of [132]).

The second method looks for the rules and parameters that characterise a pattern and replicates them, for instance repeating a pattern by translation, reflection, or rotation rules. In

Table 2.1: A set of plane curves expressed in both implicit and parametric forms.

<p style="text-align: center;">citrus curve</p>  <p>I: $a^4 b^2 y^2 + (x - \frac{a}{2})^3 (x + \frac{a}{2})^3 = 0$</p> <p>P: $\begin{cases} x = t - \frac{a}{2} \\ y = \pm \sqrt{\frac{(a-t)^3 t^3}{a^4 b^2}} \end{cases}$</p>	<p style="text-align: center;">Archimedean spiral</p>  <p>I: $\rho - a - b\theta = 0$</p> <p>P: $\begin{cases} x = (a + bt) \cos t \\ y = (a + bt) \sin t \end{cases}$</p>	<p style="text-align: center;">Lamet curve</p>  <p>I: $b x^m + a^m y^m - a^m b = 0$</p> <p>P: $\begin{cases} x = at \\ y = \pm (b - bt^m)^{\frac{1}{m}} \end{cases}$</p>
<p style="text-align: center;">m-convexities curve</p>  <p>I: $\rho - \frac{a}{1 + b \cos(m\theta)} = 0$</p> <p>P: $\begin{cases} x = \frac{a}{1 + b \cos(mt)} \cos t \\ y = \frac{a}{1 + b \cos(mt)} \sin t \end{cases}$</p>	<p style="text-align: center;">geometric petal A</p>  <p>I: $\rho - a + a \cos^{2n} \theta = 0$</p> <p>P: $\begin{cases} x = (a - a \cos^{2n} t) \cos t \\ y = (a - a \cos^{2n} t) \sin t \end{cases}$</p>	<p style="text-align: center;">geometric petal B</p>  <p>I: $\rho - a - b \cos 2n\varphi = 0$</p> <p>P: $\begin{cases} x = (a + b \cos 2nt) \cos t \\ y = (a + b \cos 2nt) \sin t \end{cases}$</p>
<p style="text-align: center;">lemniscate of Bernoulli</p>  <p>I: $(x^2 + y^2)^2 - 2a^2(x^2 - y^2) = 0$</p> <p>P: $\begin{cases} x = a \frac{\sin t}{1 + \cos^2 t} \\ y = a \frac{\sin t \cos t}{1 + \cos^2 t} \end{cases}$</p>	<p style="text-align: center;">egg of Keplero</p>  <p>I: $(x^2 + y^2)^2 - ax^3 = 0$</p> <p>P: $\begin{cases} x = \frac{a}{(1+t^2)^2} \\ y = \frac{at}{(1+t^2)^2} \end{cases}$</p>	<p style="text-align: center;">mouth curve</p>  <p>I: $a^4 y^2 - (a^2 - x^2)^3 = 0$</p> <p>P: $\begin{cases} x = a \cos t \\ y = a \sin^3 t \end{cases}$</p>
<p style="text-align: center;">elliptic curve</p>  <p>I: $y^2 - x^3 - ax - b = 0$</p> <p>P: $\begin{cases} x = \frac{t}{a} \\ y = \pm \sqrt{\frac{t^3}{a^3} + t + b} \end{cases}$</p>	<p style="text-align: center;">bullet-nose curve</p>  <p>I: $a^2 y^2 - b^2 x^2 = x^2 y^2$</p> <p>I: $a^2 y^2 - \frac{b^2}{34} x^2 = x^2 y^2$</p>	<p style="text-align: center;">astroid</p>  <p>I: $(x^2 + y^2 - a^2)^3 + 27a^2 x^2 y^2$</p> <p>P: $\begin{cases} x = a \cos^3 t \\ y = a \sin^3 t \end{cases}$</p>

this case, the parameters that characterise a single pattern and the rules for their aggregation and repetition are learned. In particular, the method can recognise the decorations made of elements that are repeated in the space in a geometric way, by locating the individual components and then aggregating them according to decoration-specific rules. Figure 2.2 illustrates the recognition of a repeated pattern. The pattern is made of two elements: the oval-like curve and the half of a mouth-like curve. The pattern is then horizontally repeated to form a frieze of four elements. The parameter a of the mouth-like curve corresponds to half of the curve width, therefore the amplitude of the curve is $2a$. Then, we use the parameter $2a$ obtained for the recognition of a single element to infer the entity of the horizontal translation and to recognise the entire moulding.

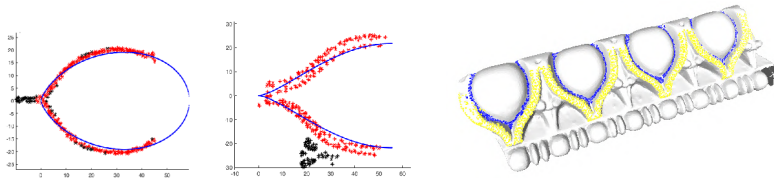


Figure 2.2: An example of recognition of a complex pattern through the use of repetition rules (courtesy of [132]).

2.3 A method for recognising plane curves

In this section, the main steps of the algorithm for the HT-based recognition of plane curves are briefly summarised. The input of our method is a 3D model, from which we extract feature points that can be projected onto the plane. As a working assumption, we assume that the input model can be locally represented by an explicit function and therefore locally flattened. An overview of methods for extracting feature points is presented in [39, 94], while the SHREC19 benchmark presents an evaluation of methods for feature curve estimation [113]. Note that the curve recognition method is independent of the technique adopted to select the feature points and the data representation (both meshes or point clouds).

Our approach works in four main steps, namely the extraction of the feature points; the projection of the potential feature points on a plane; the HT curve recognition; and an evaluation step to determine the quality of the approximation. It requires in input a family of curves and some thresholds to assess which points are significant or not, to establish if a set of feature points can be grouped or not, and to assess the quality of the recognition result.

1. *Extraction of the feature points.* Unless point clusters are provided directly, a 3D model (a mesh or a point cloud) is preprocessed to identify sets of feature points. In the case of geometric curves, the mean curvature is used as the filter criterion: for triangle meshes the algorithm in Toolbox Graph package [121] is applied; for point clouds, the polynomial

fitting of osculating jets provided by MeshLab [37] is used. Depending on the type of feature (if a ridge or a groove) the high or low curvature values are selected through the analysis of a histogram of the curvature distribution and the points corresponding to the histogram queues (the size of the queue is an input parameter) are kept. In the case of colorimetric decorations, we adopt the L-channel (luminosity) of the CIELAB space [85] to determine which points are selected.

Specifically, in the range of variation of the values assumed by the mean curvature or the luminosity, feature points are defined as points whose value falls within the first $m\%$ or the last $M\%$ of the values. The thresholds m and M influence the number of feature points; in our implementation, the usual values are 15% and 85%, respectively.

The feature points are then grouped into connected components through a clustering operation. Our implementation adopts the method *Density-Based Spatial Clustering of Application with Noise* DBSCAN, [57], which aggregates nearby points with a certain density and eliminates the isolated ones considered noise. This clustering choice permits the aggregation into groups of points of even irregular shape but also aggregate curves that intersect; in the latter case, another clustering strategy could be adopted or a composite curve should be considered. The DBSCAN algorithm requires two parameters: the threshold used as the radius of the density region (a real positive number), and the minimum number of points necessary to form a region (a positive integer). In our implementation, we use the DBSCAN routine provided in MATLAB [111]. The outcome of this preprocessing phase is a set of num clusters \mathcal{P}_i of feature points, $i = 1, \dots, num$. Each cluster of points \mathcal{P}_i is supposed to represent a curve or some of its parts.

2. *Projection on the regression plane.* This step consists in reducing the three-dimensional problem to a two-dimensional one, projecting every single cluster \mathcal{P}_i on a plane. Even if every point on a surface admits a neighbourhood that is homeomorphic to a disk, this method holds only for curves that can be flattened. Each cluster is automatically projected into its regression plane using the Matlab function *regress*; in case the estimation of the normal of a cluster is numerically unstable, an alternative method for the plane approximation is the RANSAC algorithm [62]. Then, the minimal bounding box is used to determine its main axes and estimate its size. Subsequently, the points of every cluster are translated and/or rotated to place them in the default position of the curve family selected for recognition.
3. *Recognition of the feature curve.* The generalised HT technique described in Section 1.3.4 is applied to every cluster \mathcal{P}_i . The estimation of the accumulator function \mathcal{H}_i changes according to the representation of the family \mathcal{F} . Finally, the points in the cluster that are close to the curve recognised $\mathcal{C}_{\mathfrak{a}}$ for less than a user-defined threshold (ϵ) are selected.

4. *Evaluation of the goodness of fit.* To determine if the curve identified at the previous step satisfactorily approximates a set of feature points \mathcal{P}_i , the notion of *Goodness of Fit* (GoF) introduced in [109] is used. Formally, for each point $\mathbf{p} \in \mathcal{P}_i$ let us consider the Euclidean distance d from the curve $\mathcal{C}_{\mathbf{a}}$ recognised in the previous step defined as follows:

$$d(\mathbf{p}, \mathcal{C}_{\mathbf{a}}) = \inf_{\mathbf{p}_c \in \mathcal{C}_{\mathbf{a}}} \|\mathbf{p}_c - \mathbf{p}\|_2.$$

Then the *GoF* is defined as:

$$GoF := \frac{d_1 + d_2 + \dots + d_{|\mathcal{P}_i|}}{|\mathcal{P}_i|},$$

where $|\mathcal{P}_i|$ is the cardinality of the set \mathcal{P}_i . The curve recognised is a good approximation of the profile outline if the value of the *GoF* is smaller than a given threshold (for example, an automatic way to fix such a threshold is to select the 10% of the lowest curve parameter).

Finally, the translation and rotation operations are done backward to identify on the original model the initial coordinates of the feature points recognised. The algorithm returns the parameters of the recognised curve and the vertices of the model closest to the curve identified.

2.3.1 Estimation of the Accumulator Function

The estimation of the accumulator function changes according to the type of representation of the family of curves. In this section, we describe the method for curves in parametric and implicit forms.

Estimation of the accumulator function for plane curves in parametric form. For curves in parametric form, the generic analytical expression of $\Gamma_{\mathbf{p}}$ needs to be derived directly from the parametric curve equations of the family [109]. A particular case of great interest occurs when the system of parametric equations is linear with respect to the parameters \mathbf{a} . In this case, a general rule for the analytic solution of the system with respect to \mathbf{a} is derived and therefore an explicit expression for $\Gamma_{\mathbf{p}}$ in equation (1.4). To this purpose, the Moore-Penrose pseudo-inverse [68, 120] of the system that defines \mathcal{F} is considered. In our experiments we use the Moore-Penrose inverse of real matrices M^\dagger that can be computed using the singular value decomposition. The advantage of M^\dagger is that it can deal with $2 \times n$ matrices and provides the least squares solution to a system of linear equations.

In particular, exploiting the Moore-Penrose pseudo-inverse [120], it is possible to automatically derive $\Gamma_{\mathbf{p}}$ by of the matrix $M(t)$ that defines the coefficients of \mathbf{a} . Then, if the family of curves in parametric form admits a matrix representation as

$$(x(t), y(t)) = M(t)\mathbf{a},$$

where $M(t)$ is a matrix $2 \times n$ that depends on the variable of the parametric equations t , $\Gamma_{\mathbf{p}}$ can be calculated directly

$$\Gamma_{\mathbf{p}} : \mathbf{A}(t) = M^\dagger(t)(x_{\mathbf{p}}, y_{\mathbf{p}}).$$

Note that the parameter t that appears in the parametric formulation of $\Gamma_{\mathbf{p}}$ is the same as in the parametric equation of the family of curves \mathcal{F} . Using the notation introduced in Section 1.3.4, the condition of crossing a cell $c(k_1, \dots, k_n)$ is translated into the following inequality

$$(2.1) \quad a_{j,k_j} - \frac{\delta_j}{2} \leq A_j(t) < a_{j,k_j} + \frac{\delta_j}{2}.$$

If the inequality is verified for at least one value of the parameter t for each component j of $\mathbf{A} = (A_1, \dots, A_n)$, the value of \mathcal{H}_i corresponding to $c(k_1, \dots, k_n)$ increases by 1.

Note that, if the family of curves is formed by the union of parametric equations of the curves composing it, the evaluation of the intersection between the cells of the parameter space and the HT of the points is slightly different. In particular, considering the family of curves represented by the union of q parametric equations, the system $\mathbf{x} = M(t)\mathbf{a}$ has to be solved for each system of equations that make up the family of curves by obtaining q results $\mathbf{A}_s = (A_{1_s}, \dots, A_{n_s})$, with $s = 1, \dots, q$. The inequality (2.1) has to be verified for at least one parameter t for each component j and for at least one value of s .

Estimation of the accumulator function for plane curves in implicit form. In the case of plane curves expressed in implicit form, $\Gamma_{\mathbf{p}}$ is a hypersurface, and it can be seen as the zero loci of its implicit function. Then, $\Gamma_{\mathbf{p}}$ crosses the cell $c(k_1, \dots, k_n)$ if its evaluation on the cell centre is zero. Indeed, exploiting the local approximation of the zero loci of a set of analytic functions in terms of series of Taylor, an algebraic theoretical result is implemented based on comparing the evaluation of the functions present in $\Gamma_{\mathbf{p}}$ at the cell centres with two theoretical bounds. These bounds depend on the Jacobian and Hessian matrices of these functions and the pseudo-inverse of the Jacobian.

Exploiting the notation introduced in Section 1.2 and referring to [153], we consider $F_{\mathbf{p}}(\mathbf{A})$ and $F_{\mathbf{p}}(\hat{\mathbf{a}})$, with $\hat{\mathbf{a}}$ the point center of the selected cell. This cell is a subset of the (∞, \mathcal{E}) -unit ball centred at point $\hat{\mathbf{a}}$, that is $\mathbf{B}_{(\infty, \mathcal{E})} = \{\mathbf{A} \in \mathbb{R}^n \mid \|\mathbf{A} - \hat{\mathbf{a}}\|_{(\infty, \mathcal{E})} \leq 1\}$, where $\|\cdot\|_{(\infty, \mathcal{E})}$ is the weighted infinite norm with respect to the positive diagonal matrix \mathcal{E} with entries the positive real numbers $\frac{1}{\varepsilon_1}, \dots, \frac{1}{\varepsilon_n}$. Then, we fixed $\varepsilon_{min} = \min \varepsilon_1, \dots, \varepsilon_n$ and $\varepsilon_{max} = \max \varepsilon_1, \dots, \varepsilon_n$ and we denote $Jac_{F_{\mathbf{p}}}(\mathbf{A})$ and $H_{F_{\mathbf{p}}}(\mathbf{A})$ the Jacobian and the Hessian matrices of F . Finally, we indicate H the value $H = \max_{\mathbf{A} \in \mathbf{B}_{(\infty, \mathcal{E})}} \{\|H_{F_{\mathbf{p}}}(\mathbf{A})\|_{\infty}\}$ and J the value $J = \sup_{\mathbf{A} \in \mathcal{D}(\hat{\mathbf{a}}, R)} \|Jac_{F_{\mathbf{p}}}^\dagger(\mathbf{A})\|_{\infty}$, with $\mathcal{D}(\hat{\mathbf{a}}, R) = \{\mathbf{A} \in \mathbb{R}^n \mid \|\mathbf{A} - \hat{\mathbf{a}}\|_{\infty} < R\}$, $R < \min\{\varepsilon_{min}, \frac{\|Jac_{F_{\mathbf{p}}}(\hat{\mathbf{a}})\|_1}{H}\}$ and $Jac_{F_{\mathbf{p}}}^\dagger(\mathbf{A})$ the pseudo-inverse of the Jacobian $Jac_{F_{\mathbf{p}}}(\mathbf{A})$. If:

- $|F_{\mathbf{p}}(\hat{\mathbf{a}})| > \|Jac_{F_{\mathbf{p}}}(\hat{\mathbf{a}})\|_1 \varepsilon_{max} + \frac{H}{2} \varepsilon_{max}^2 =: B_1$, then $\Gamma_{\mathbf{p}}$ does not cross the cell and the corresponding entry of \mathcal{H}_i does not increase;

- $|F_{\mathbf{p}}(\hat{\mathbf{a}})| < \frac{2R}{J(c+\sqrt{n}HJR)} =: B_2$, with $c = \max 2, \sqrt{n}$, then $\Gamma_{\mathbf{p}}$ crosses the cell and the corresponding entry of \mathcal{H}_i increases by 1;
- $B_2 < |F_{\mathbf{p}}(\hat{\mathbf{a}})| < B_1$ the corresponding entry of \mathcal{H}_i increases by a value of indeterminacy ξ . By convention, this value of indeterminacy is fixed at 0.5.

The implementation of the method for curves in implicit form requires the symbolic computation that we handle with the CoCoA library [6].

2.3.2 Computational cost

The cost of the HT recognition algorithm is dominated by the size of the discretization of the region of the parameter space. Following the notation introduced in Section 1.3.4, such a discretization consists of $M = \prod_{j=1}^n N_j$ elements, where n is the number of parameters (in the curves proposed, $n = 1, 2, 3$) and N_j is the number of subdivisions for the j th parameter. Since the accumulator function \mathcal{H}_i has the same dimension as \mathcal{T} , the evaluation of \mathcal{H}_i requires $O(M)$ operations for each point. Therefore, the computational complexity for each cluster is $O(ML)$, where L represents the number of elements of \mathcal{P}_i on which we evaluate the HT accumulator function. Moreover, as described in [153], in the case of curves in the implicit form we need to evaluate, once for each curve, the symbolic expression of the Jacobian, the Moore-Penrose pseudo-inverse and the Hessian matrices which have the same order of complexity of M and, therefore, the cost of the HT-based recognition is $O(M)$ for each point. Then, the computational complexity for each cluster is $O(ML)$. In summary, the algorithm has the same theoretical complexity for curves in implicit and parametric form, but, in the first case, it resorts to using symbolic computations (managed with CoCoA) that, in practice, could become a computational bottleneck for densely sampled curves.

2.4 Testing and validating the method on data from real objects

This section presents some results we obtained by the recognition method described in Section 2.3 on a set of models collected from the web and various repositories, in particular the benchmark proposed in the *SHape REtrieval Contest SHREC'19* track on feature curve extraction [113], the VISIONAIR shape repository [1], the STARC repository [2], and the ornaments in the Regency collection. The original models of the ornaments from the Regency collection are courtesy of professor K. Rodriguez Echavarria. All models are represented as triangle meshes or point clouds.

Figure 2.3 shows some examples of recognition of different patterns using the single mathematical curves listed in Section 2.2.1. In these examples, the feature points are extracted using different thresholds of the mean curvature. In the right column, we show the projected points and the curve that best approximates the profile \mathcal{P} they outline. The red points are the closest to the curve with respect to the threshold ε , as detailed in Section 2.3. Backward operations

are performed on these points to obtain the initial coordinates of the corresponding vertices. The outcome of the recognition algorithm is shown in the next column: the recognised vertices are highlighted on the model. The models in Figure 2.3(a,d-f) have a known unit of measure (millimetres): depending on the curve, the parameters a and b give a precise estimation of the size of the curve. The curves and their parameters are: (a) an Archimedean spiral with $a = 10.67$ and $b = 5.51$, (b) a geometric petal curve of type (A) with $a = 7.85$, $c = 0.56$ and $n = 50$, (c) a m -convexities curve with $a = 8.51$, $b = 0.30$ and $m = 5$, (d) an egg of Keplero with $a = 8.87$, (e) a mouth curve with $a = 55.67$, (f) an elliptic curve with $a = 28.38$ and $b = 76.05$. The models in the first column ((a), (b), (c)) have been proposed in the *SHape REtrieval Contest SHREC'19*.

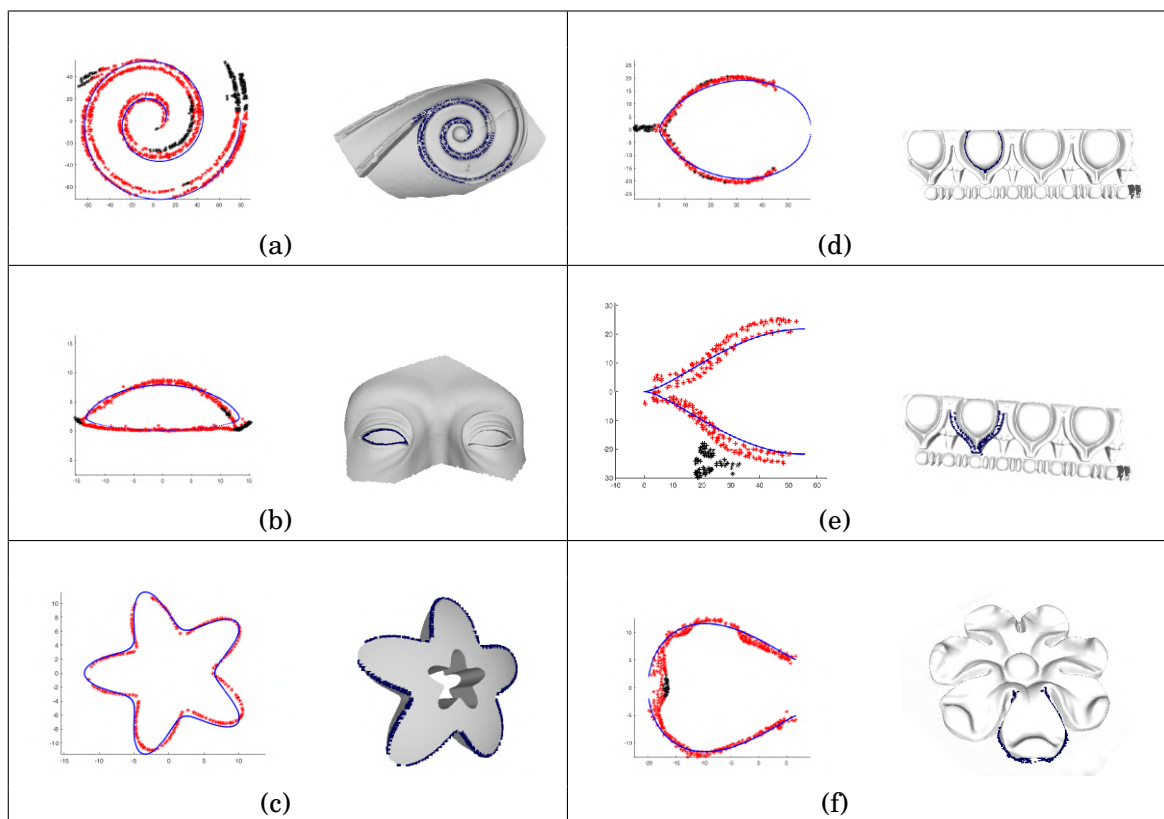


Figure 2.3: Recognition of various feature curves: in (a) an Archimedean spiral; in (b) a geometric petal of type (A); in (c) a 5-convexities curve; in (d) an egg curve; in (e) a mouth curve; in (f) an elliptic curve. For each pair of pictures, we show on the left the projected points, the curve that best fits them and we highlight in red the points closest to it. This latter set of points is then shown on the original model on the right.

The track of the SHREC'19 had a twofold purpose: the main request was to propose a method able to extract the feature curves and to highlight one or more subsets of vertices of the meshes in the dataset; the second and optional task was to find similarities among the feature curves extracted across all the models. The results obtained by the HT-based method have been evaluated by the organizers through evaluation measures and compared to those of the other participants.

The analysis showed that a good balance between precision and vertex clustering was obtained by the HT-based method, which recognised most of the expected feature curves, balancing the number of vertices recognised and the curve fragmentation (with respect to the ground truth). Furthermore, this method was also able to answer the optional task, finding similarities between curves of the same family, belonging to the same or different models.

2.5 A comparative analysis for plane curves

In this section, we propose a comparative analysis between the methods for plane curves in parametric and implicit form for the estimation of the HT, in terms of the quality of the approximation and computational time.

Quality measures. In Section 2.3 we have proposed the *GoF* measure as the termination criterion of our method. In addition, here we quantify the quality of the HT approximation obtained with two other distances, namely the direct Hausdorff distance and an average of the distance among the characteristic points and the fitting curve, called $mean(D_{knn})$.

The *Direct Hausdorff distance* from the points $a \in A \subset \mathbb{R}^3$ to the points $b \in B \subset \mathbb{R}^3$ is defined as follows:

$$(2.2) \quad d_{dHaus}(A, B) = \max_{(a \in A)} \min_{(b \in B)} d(a, b),$$

with d the Euclidean distance.

The $mean(D_{knn})$ distance between two sets of points $A \subset \mathbb{R}^3$ and $B \subset \mathbb{R}^3$ searches for the k nearest neighbors in B to each point in A using a three dimensional Kd -tree with the Euclidean distance [64]. In particular, in our test we have put $k = 1$, then we search for the nearest neighbor in B for each point in A and we get a set of distances D_{knn} . Then, the $mean(D_{knn})$ is calculated by averaging this set.

We also tested the Fréchet distance [56], which has been proposed to measure how two curves match also in terms of their curve parameterization. Unfortunately, when dealing with curves on real models, we noticed that the dependency of this measure on the curve parameterization makes it unsuitable for comparing a mathematical curve with an unordered cluster of points.

Comparative analysis. Tables 2.2, 2.3 and 2.4 compare the HT-based algorithms in implicit and parametric form for the families of plane curves shown in Table 2.1 when the data are samples on an exact mathematical expression of a curve or a random perturbation error of the 5% of the diagonal of the curve bounding box. The parameters of a mathematical curve and those recognised by the HT-based algorithm on these samples are listed in Table 2.2, while the quantitative distances between the identified curves and the samples are shown in Tables 2.3 and 2.4. Specifically, the parameters of the mathematical curves, those recognised by the HT algorithm on samples of the exact mathematical curve, and the ones of the same curve perturbed

with noise are listed in Table 2.2. In each column, we report the family of space curves, the parameters of the mathematical curve, and the curve parameters that differ from the original ones as recognised by the HT algorithms for the implicit and parametric curve expressions respectively on exact curve samples and perturbed curve samples. The results in Table 2.2 show that the algorithm in implicit form identifies exactly the parameters corresponding to the original curves, while the parametric version gets the exact parameters only in four cases. Then, the implicit version obtains a better approximation than the parametric one considering the same discretization of the parameter space.

Table 2.2: Comparison between the parameters of the mathematical curve and those recognised by our algorithm over exact or perturbed curves with respect to both implicit and parametric curve formulation. The values in bold represent the target parameters. The symbol = indicates when the parameter identified by the HT-based algorithm equals the target one.

Mathematical Curve		HT on exact curve samples		HT on perturbed samples	
Curve	Parameters	implicit	parametric	implicit	parametric
(1) citrus curve	a=2 b=1	=	=	$a = 1.92$ $b = 1.020$	$a = 2.120$ $b = 1.010$
(2) Archimedean spiral	a=1 b=1	=	$a = 1.040$ $b = 0.990$	$a = 0.98$ $b = 1.020$	$a = 1.070$ $b = 0.97$
(3) Lamet curve $m = 16$	a=2 b=1	=	= $b = 0.990$	$a = 2.040$ $b = 0.960$	$1 = 860$ $b = 1.02$
(4) m-convexities curve $m = 5$	a=0.2 b=0.1	=	= $b = 0.106$	$a = 1.96$ =	= $b = 0.094$
(5) geometric petal A $n = 10$	a=2	=	$a = 1.95$	=	$a = 1.92$
(6) geometric petal B $n = 3$	a=1 b=1	=	$a = 1.030$ $b = 0.970$	$a = 1.010$ $b = 0.990$	$a = 1.010$ $b = 1.010$
(7) lemniscate of Bernoulli	a=2	=	$a = 1.95$	=	$a = 2.1$
(8) egg of Keplero	a=2	=	$a = 1.96$	$a = 1.980$	$a = 1.980$
(9) mouth curve	a=1	=	=	$a = 0.98$	$a = 0.94$
(10) elliptic curve	a=2 b=4	=	=	$a = 2.040$ =	= $b = 3.88$
(11) bullet-nose curve	a=4 b=2	=	$a = 3.960,$ $b = 1.980$	$a = 4.016$ $b = 1.980$	$a = 4.0480$ $b = 2.056$
(12) astroid	a=2	=	=	$a = 2.040$	$a = 1.960$

In the columns of Tables 2.3 and 2.4 we list, from left to right, the family of curves; the GoF, the d_{dHaus} , the $mean(D_{knn})$ distances; the computational time in seconds; all these measures are reported for both the implicit and parametric curve expressions. Finally, in the rightmost column, we indicate the number of points L in each cluster. The measurements in Table 2.3 refer to the quality of the HT approximation when applied to a set of L points uniformly sampled on a mathematical curve. We evaluate the quality of the curve approximation by uniformly sampling the curve recognised with 200 points (in general, these points do not correspond to the L samples used to fit the curve) and estimate the GoF , $mean(D_{knn})$ and direct Hausdorff distances between these samples and the original set of points. For simplicity of notation in the Tables, we represent

with 0.0 a distance when its value is close to the machine precision.

Table 2.3: Quality measures and computational time for the HT-based recognition algorithm for the same plane curve in implicit (I) and parametric (P) form. L represents the number of characteristic points.

Curve	GoF		d_{dHaus}		$mean(D_{knn})$		Time (s)		L
	I	P	I	P	I	P	I	P	
(1)	0.0	0.0	0.0	0.0	0.0	0.0	8	0.5	68
(2)	0.0	0.014	0.0	0.084	0.0	0.035	10	1	63
(3)	0.0	$3 * 10^{-4}$	0.0	$6 * 10^{-4}$	0.0	$6 * 10^{-4}$	20	0.5	68
(4)	0.0	0.001	0.0	0.002	0.0	0.001	8	0.5	63
(5)	0.0	0.061	0.0	0.050	0.0	0.074	0.8	0.5	63
(6)	0.0	$8 * 10^{-4}$	0.0	0.060	0.0	0.027	15	1.5	126
(7)	0.0	0.014	0.0	0.070	0.0	0.040	0.7	0.3	63
(8)	0.0	0.005	0.0	0.050	0.0	0.011	1	0.4	84
(9)	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.3	63
(10)	0.0	0.0	0.0	0.0	0.0	0.0	15	0.5	104
(11)	0.0	0.006	0.0	0.061	0.0	0.027	13	1	80
(12)	0.0	0.0	0.0	0.0	0.0	0.0	1	0.5	63

Table 2.4: Quality measures and computational time for the HT-based recognition algorithm for the same plane curves of Table 2.3, perturbed by the 5% of the diagonal of the curve bounding box. The curve numbers are the same as in Table 2.2. L represents the number of characteristic points.

Curve	GoF		d_{dHaus}		$mean(D_{knn})$		Time (s)		L
	I	P	I	P	I	P	I	P	
(1)	0.025	0.028	0.075	0.076	0.031	0.031	8	0.5	68
(2)	0.030	0.031	0.237	0.333	0.089	0.135	8	0.5	63
(3)	0.015	0.027	0.101	0.178	0.041	0.051	20	0.5	68
(4)	0.004	0.007	0.015	0.015	0.008	0.008	8	0.5	63
(5)	0.002	0.019	0.010	0.063	0.002	0.039	0.8	0.5	63
(6)	0.014	0.019	0.070	0.075	0.033	0.037	15	1.5	126
(7)	0.018	0.038	0.063	0.171	0.039	0.089	0.7	0.3	63
(8)	0.015	0.015	0.078	0.078	0.028	0.028	1	0.4	84
(9)	0.016	0.027	0.078	0.114	0.036	0.055	0.9	0.3	63
(10)	0.034	0.049	0.485	0.317	0.049	0.054	15	0.5	104
(11)	0.060	0.062	0.151	0.154	0.060	0.062	13	1	80
(12)	0.007	0.045	0.179	0.097	0.024	0.04	1	0.5	63

Similarly, the measures in Table 2.4 report the distances of the HT curve approximation with respect to a set of L randomly perturbed samples of a smooth curve; note that these L points do not exactly lie on the original curve. The GoF measure and the $mean(D_{knn})$ highlight that, even

in this case, better precision is achieved by the HT-based algorithm for approximating curves in implicit form with respect to the parametric ones. On the other hand, the algorithm for curves in parametric form is computationally more efficient. Moreover, in general practice, the family of curves in implicit form allows one to recognise globally the profile and thus makes it easier to complete the missing parts. On the other hand, the parametric form allows for identifying more precisely the portion of the curve that best approximates the profile.

The experiments are performed on the same laptop equipped with an Intel Core i7 processor (at 1.3 GHz). All the routines are implemented in MATLAB [111], except for the symbolic calculations implemented on the open source toolkit CoCoA [6]. We also tested the use of the symbolic computation part in MATLAB, but CoCoA was more efficient, although it is less developed.

2.6 Applications

We applied our plane curves recognition method to two specific domains, that is cultural heritage finds (Section 2.6.1) and artistic or design motifs (Section 2.6.2). In both domains, the recognised parameters have an important role. In the first case, they are used to compare the decorations present on the surface of different archaeological fragments [135], then it is possible to assess if they could be part of the same ornament. In the second, the curves in parametric form can be used as an input for the interactive visualization and manipulation of the symbols through a multi-touch smart table [137]. Some of these motifs are graphic elements obtained from the composition of mathematical curves and can be recognised through our method.

2.6.1 Recognition of decorations in archaeological finds

In this section, we present some examples that show how to translate archaeological knowledge into effective algorithms. The models were proposed within the GRAVITATE project [3], which had the specific purpose to support the archaeologist's knowledge with automated mathematical tools. Many fragments exhibit decorations composed of feature curves organised according to specific distribution rules; the aim is to encode these patterns into routines that can be applied to parts having a different overall shape, structure and functionality. To this end, in the first part, we present examples of aggregation techniques that automatically recognise and annotate compound curves and frets (both floral and Greek-like styles), while the second paragraph shows how to use the method for addressing decoration compatibility measures.

Recognition of compound curves. Figure 2.4 represents the HT fitting of parts of two Greek frets, respectively with 8 (Figure 2.4(a) and its complementary Figure 2.4(c)) and 6 (Figure 2.4(b)) straight lines. In this case, the lines are orthogonal to the Cartesian axes and therefore, the degree of the HT curves is still limited. However, in general, the product approach can produce

curves of a very high degree. For instance, to recognise as a unique element the 6-petal flower depicted in Figure 2.5 it needs the product of 6 citrus curves of degree 3, thus generating the HT transform for a family of curves of degree 18. The higher the curve degree, the higher the number of parameters and, as a consequence, the computational effort.

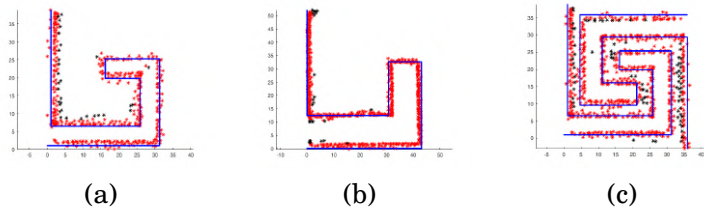


Figure 2.4: Recognition of Greek fret elements with 8 (a) and 6 (b) straight lines, respectively. In (c) the recognition of the element complementary to (a).

While [154] limited the recognition of compound curves to the product of curves, here some automatic rules and parameters that characterise style elements are introduced. In particular, the method can recognise the decorations made of elements that are geometrically repeated in the space, by locating the individual components and then aggregating them according to decoration-specific rules. For instance, for the 6-petal flower of Figure 2.5(a) that characterises many fragments of votive statues from the Salamis island [89] (approximately 50 fragments), every single petal is first recognised. Given the peculiarity of the decoration (light on a dark background), the vertices corresponding to a high value of luminosity are selected in the preprocessing step. The resulting clusters are shown in Figure 2.5(b). These clusters are then processed in the recognition step, using a family of citrus curves that better fit their petal-like shape. Since this decoration is hand-made and comes from an abraded fragment, the petals have slightly different sizes and shapes. In Figure 2.5(c), the numbers in the petals represent the petal labels while the measures of the two main axes are detected based on the parameters of the six fitting curves obtained with the HT as shown in Figure 2.5(d).

In particular, the recognised parameters appear in the coordinates of the salient points of this curve, i.e. the endpoints of the curve symmetry axes, that is $(\pm \frac{a}{2}, 0)$ and $(0, \pm \frac{a}{8c})$ (Figure 2.5(c)). Then it is possible to evaluate the rays of the circular crown within which the salient points have to lie if the petals belong to the same flower (see Figure 2.6(a)). In this way, 6-petals flowers even incomplete can be identified and annotated (see Figure 2.6(b,c)).

Finally, it is possible to recognise and annotate repeated decorations. Based on the petal contiguity, the single flowers are annotated to recognise the whole floral band and automatically annotate it, the flowers and the petals. Note that most of the petals belong to two flowers, as detailed in Figure 2.7 on an example of a floral band, thus making explicit which flower is close to another.

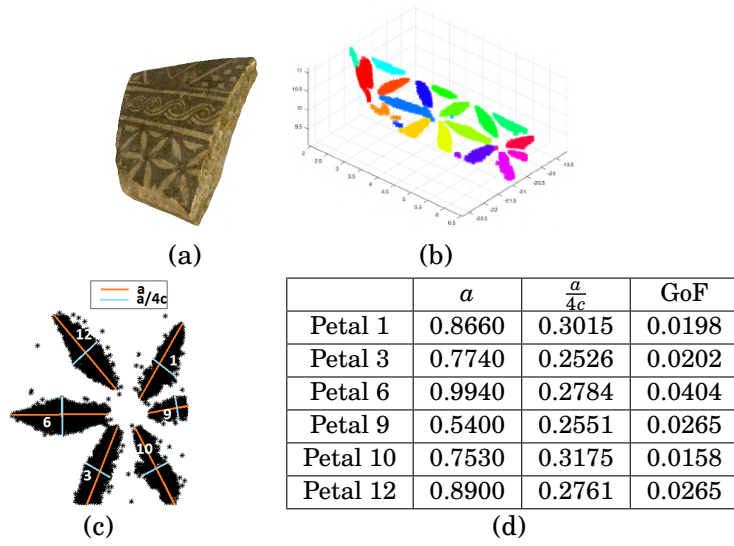


Figure 2.5: An archaeological fragment (a). Petal-like clusters (b) recognised with citrus curves (c). Measures and GoF of the six citrus curves (d).

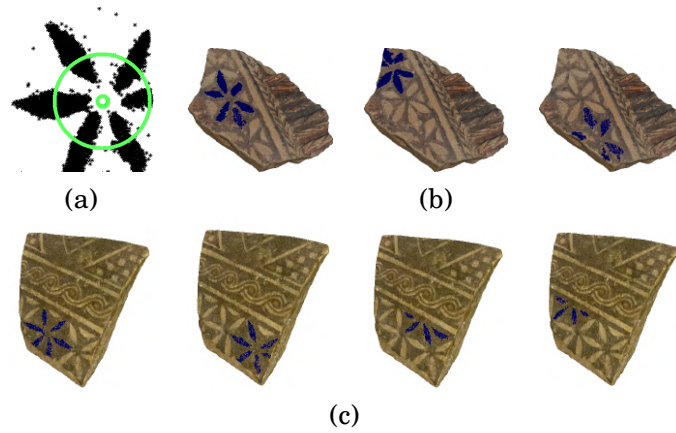


Figure 2.6: A pictorial representation of the criteria adopted to detect which petals belong to the same flower(a), and some examples of flowers recognised on two different fragments (b,c).

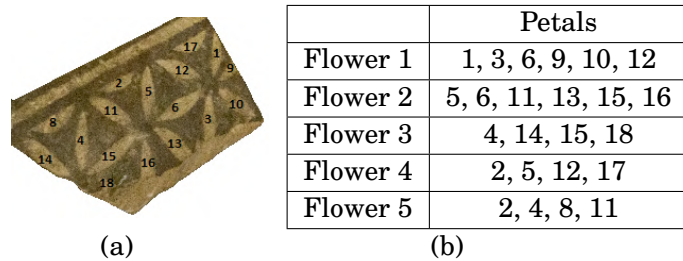


Figure 2.7: A detail of floral band annotation. Each petal is numbered (a) for each flower we list its petals (b).

Similarity of feature curves. The curve parameters obtained with the HT technique are indicators of the most representative curve and therefore they can be used to compare elements even if recognised on different objects. Indeed, thanks to the distinctive power of the HT parameters, a distance between the two curves \mathcal{C}_1 and \mathcal{C}_2 is defined as the norm L^1 of the parameters corresponding to these curves, i.e., $d(\mathcal{C}_1, \mathcal{C}_2) = \|\lambda_{\mathcal{C}_1}, \lambda_{\mathcal{C}_2}\|_1$, where $\lambda_{\mathcal{C}_1}$ and $\lambda_{\mathcal{C}_2}$ are the parameters of the curves \mathcal{C}_1 and \mathcal{C}_2 , respectively. Note that such a notion of distance assumes that the curve parameters are homogeneous in terms of the measured properties; this implies that the distance between two feature curves is computed if they belong to the same family.

These estimates provide insights into the style compatibility of the objects themselves. Figure 2.8 shows an example of the curve rating obtained on the spiral-like details of the models contained in the SHREC'19 benchmark on feature recognition [113]. For recognising these elements the family of Archimedean spirals is considered; the parts are sorted in increasing order of distance from the leftmost one (that acts as the query element). In particular, the two closest spirals (see Figures 2.8(a,b)) could be part of the same ornament, even if located in different model areas, see in Figure 2.9 a possible reconstruction.

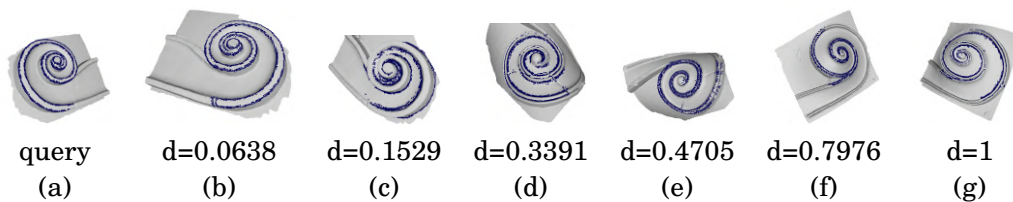


Figure 2.8: (a) A query element. In (b-g) the retrieved elements are ordered with respect to their increasing distance.



Figure 2.9: Comparing the parameters of the recognised curves (Figures 2.8(a,b)) it is possible to associate two different fragments as parts of the same moulding.

Figure 2.10 and Table 2.5 sketch the effective usefulness of the method for fragment re-association. Starting from four different fragments (Figure 2.10(a)), potentially coming from different votive statues, but with the same stylistic character of a floral band, these fragments are compared in terms of the distance among the 6-petal flowers in each decoration. These fragments contain 3, 2, 1, and 2 complete 6-petal flowers, respectively.

To automatically assess the HT parameters of a 6-petal flower, the family of geometric petal (type B) curves is adopted. As it can be seen in Figure 2.10(b), the recognition is not always perfect as the flowers have been drawn by hand and then inaccurate, while the family of curves

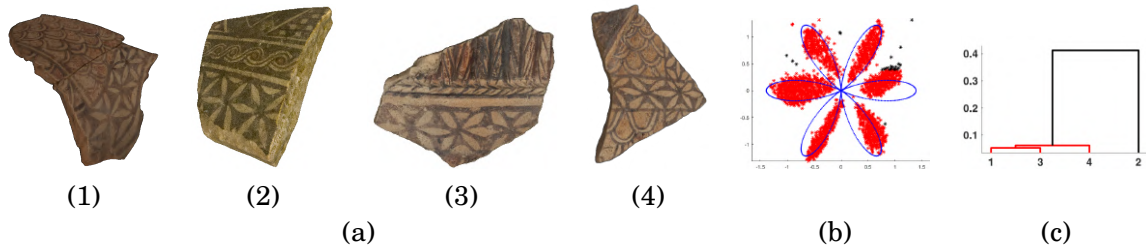


Figure 2.10: (a) Four different fragments with the same stylistic character of a floral band; (b) the 6-petal flower is recognised with a geometric petal (type B) curve; (c) a visual representation of the distance among the models shown through a dendrogram plot.

follows precise geometric rules. Despite this, the parameters recognised give enough information on the size of the flowers, allowing the production of a similarity matrix, see Table 2.5(a) for numerical details on the distances among the 6-petal flowers on these 4 fragments. To highlight that in this specific case the floral band is hand-crafted and the decorative elements slightly differ even on the same surface, Table 2.5(a) shows the distances among all the flowers recognised on the four models. To compare the models the distances of the flowers that make up their floral bands are averaged or the minimum of them is taken (Table 2.5(b,c)). Starting from the similarity scores obtained, it is possible to conclude that the floral bands recognised on the fragments (1), (3) and (4) are compatible, and therefore they could belong to the same part of a statue, while the one in the fragment (2) is different, for instance by simply clustering the fragments based on the distance matrices in Table 2.5. Figure 2.10(c) visually depicts the distance among the models using hierarchical clustering, a dendrogram plot, which consists of many U-shaped lines that connect the models in a hierarchical tree. The height of each U represents the distance between the two connected models.

Table 2.5: The similarity matrix among the 6-petal flowers recognised on the four fragments in Figure 2.10(a). The following naming convention is adopted: the first number represents the flower label and the second one corresponds to the model number.

Similarity scores among 6-petal flowers							
	2_1	3_1	1_2	2_2	1_3	1_4	2_4
1_1	0.204	0.076	0.553	0.461	0.052	0.266	0.087
2_1	0	0.128	0.757	0.665	0.256	0.062	0.215
3_1	-	0	0.629	0.537	0.128	0.190	0.087
1_2	-	-	0	0.092	0.504	0.722	0.543
2_2	-	-	-	0	0.412	0.630	0.451
1_3	-	-	-	-	0	0.318	0.139
1_4	-	-	-	-	-	0	0.179

(a)

Fragment similarity scores (mean)			
	2	3	4
1	0.6008	0.1457	0.1509
2	0	0.4579	0.5863
3	-	0	0.2286

(b)

Fragment similarity scores (min)			
	2	3	4
1	0.4615	0.0524	0.0617
2	0	0.4508	0.4508
3	-	0	0.1391

(c)

2.6.2 Recognition of motifs and symbols

This section describes how we applied our method to recognise three symbols, as shown in [137]. In this case, the feature points are extracted from an image by an edge detection method, e.g., the Canny edge detection algorithm [33]; then in this case the projection of points is not required since they are planar. All these motifs can be represented as a union of parametric curves. The first one is the symbol of Toyota (Fig. 2.11(a)), made up of several occurrences of the same type of curve, ellipses in this case. Then a family composed by the union of three ellipses is considered, of which two are centred in the origin and the third one shifted by half the length of the semi-axis minor of the external ellipse:

$$(2.3) \quad \begin{cases} x = a \cos t \\ y = b \sin t \end{cases} \quad t \in [0, 2\pi] \cup \begin{cases} x = a_1 \cos t \\ y = b_1 \sin t + \frac{b}{2} \end{cases} \quad t \in [0, 2\pi] \cup \begin{cases} x = a_2 \cos t \\ y = b_2 \sin t \end{cases} \quad t \in [0, 2\pi].$$

The second symbol represents the logo of the Mathematical Olympiad (Fig. 2.11(b)) and it has been recognised using a family of curves expressed in parametric form by the union of a circle centred in the origin and a lemniscate of Bernoulli:

$$(2.4) \quad \begin{cases} x = \sqrt{2}a \frac{\sin t}{1+\cos^2 t} \\ y = \sqrt{2}a \frac{\sin t \cos t}{1+\cos^2 t} \end{cases} \quad t \in [0, 2\pi] \cup \begin{cases} x = b \cos t \\ y = b \sin t \end{cases} \quad t \in [0, 2\pi].$$

Finally, the recognition of artwork is tackled. It is named *Third Paradise*¹ and it is created by the artist Michelangelo Pistoletto (Fig. 2.11(c)). A first attempt to recognise the symbol with a composition of curves was presented in [133]. In this work, the purpose was not to reproduce the motif exactly but only to recognise it, using a family of curves expressed in an implicit form composed of the product of two eggs of Keplero and one mouth curve, then with the family of equation (see Fig. 2.12(a)):

$$\begin{aligned} &(((x+a)^2 + y^2) + b(x+a)^3) (a^4 y^2 - (a^2 - x^2)^3) \\ &(((x-a)^2 + y^2)^2 - b(x-a)^3) = 0. \end{aligned}$$

To fit more accurately the shape of the motif, in [137] the external curves are modified with a lemniscate of Bernoulli, and the central part is replaced with a citrus curve (see Fig. 2.12(b)):

$$\begin{cases} x = \sqrt{2}a \frac{\sin t}{1+\cos^2 t} \\ y = \sqrt{2}a \frac{\sin t \cos t}{1+\cos^2 t} \end{cases} \quad t \in [0, 2\pi] \cup \begin{cases} x = t - \frac{a}{2} \\ y = \pm \sqrt{\frac{(a-t)^3 t^3}{a^4 b^2}} \end{cases} \quad t \in [0, a]$$

¹<http://terzoparadiso.org/en/what-is>

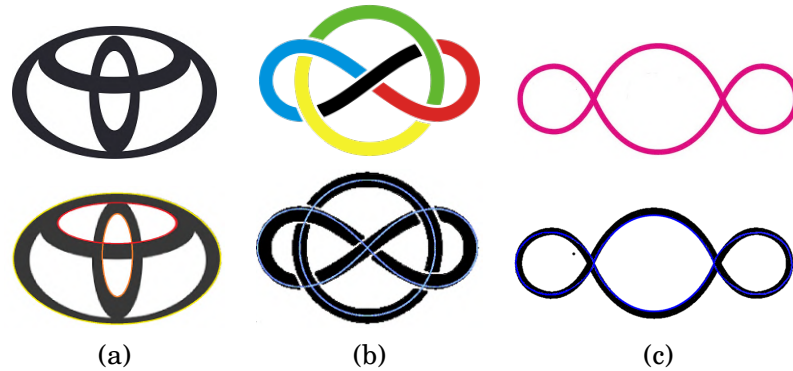


Figure 2.11: Examples of recognition of complex motifs: (a) the symbol of Toyota (courtesy of the Toyota Company); (b) the symbol of the Mathematical Olympiad; (c) the *Third Paradise*.

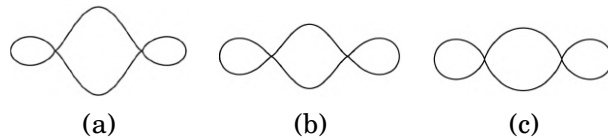


Figure 2.12: The sequence of the three different mathematical representations of the Third Paradise used in the recognition process.

and, starting from the parametric equations of these curves, the coefficients and exponents are changed to approximate better the curvature and shape of each component. At last, the most accurate representation is given by a family of curves expressed in parametric form and composed of the union of six curves:

$$(2.5) \quad \begin{cases} x = \pm \frac{113}{100} a \frac{\sin^{\frac{7}{5}} t}{1 + \cos^2 t} \pm a \\ y = \frac{363}{250} a \frac{\sin t \cos t}{1 + \cos^2 t} \end{cases} \quad t \in [0, \pi] \cup \begin{cases} x = \pm a t^{\frac{6}{5}} \mp a \\ y = \pm \frac{4a(2-t)t}{5} \end{cases} \quad t \in [0, 1].$$

The mathematical representation of the motif represented with Eq. (2.5) is shown in Fig. 2.12(c).

This improved approximation allowed us to meet the demands of the project [4], whose purpose was to create an interactive experience for museum visitors focusing on the case study of the Third Paradise. The formulation of the three curves in parametric form has been used as an input for the interactive visualisation and manipulation of the symbols by their geometrical features: users can manipulate the curves such that the proportions and dimensions are modified by varying the constitutive parameters of the equations. By changing the dimensions of the curve or modifying colour and thickness, they can visualise different variations of the symbol, explore the geometry, and deform it. Ta.Bi² is used for this purpose, a multi-touch smart table optimised to display and control 2D content. In addition to the traditional finger touch, Ta.Bi is able to recognise specific tangible objects on its surface, called "tags" (Fig. 2.13(a)), by means of the "tangible bit" principle [87]. An example of this application is shown in Figure 2.14: starting from

²<https://tabi.spxlab.com/>

the computation of the intersection points between the ellipses defined in 2.3, it is possible to colour the parts of the curves that correspond to each letter of "Toyota" and create an animated sequence on the interactive table.

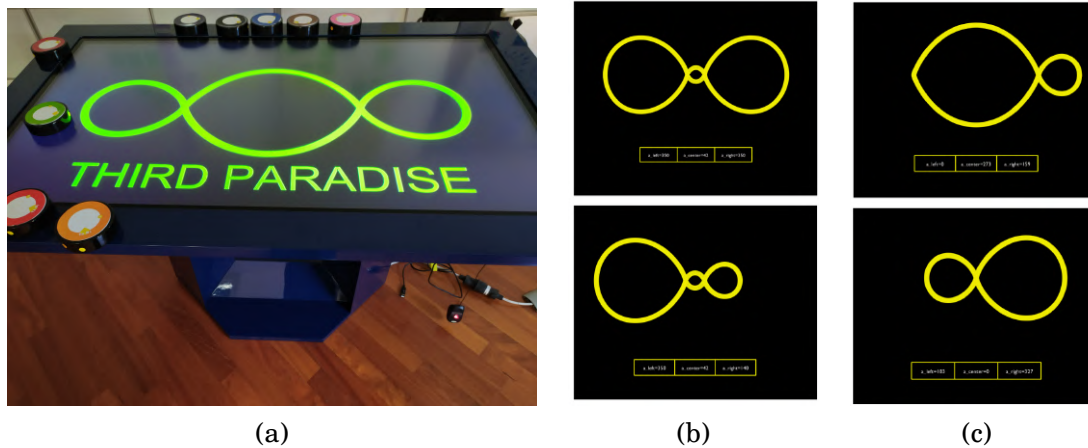


Figure 2.13: In (a) the smart table Ta.Bi with tags. In (b) and (c) some examples of symbol manipulation varying the value of the parameter α of equation 2.5.

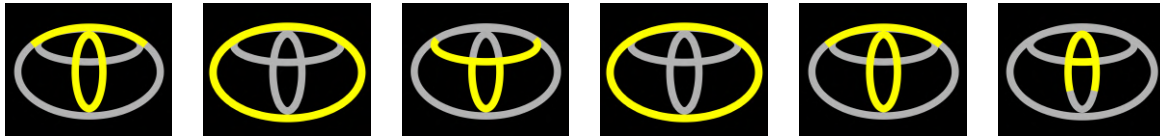


Figure 2.14: The letters of the word "Toyota" are highlighted inside the symbol.

Another example is shown in Figure 2.13(b,c) considering the artwork *Third Paradise*. In particular, by analytically computing the area of the curves with the integral calculation, it is possible to obtain that the area of the central curve (A_{center}) corresponds to the sum of the areas of the two lateral curves (A_{right}, A_{left}), up to small approximation errors. Therefore, according to the experimental result, the constraint is $A_{center} = A_{right} + A_{left}$ and in this way, naming a_{center} , a_{right} and a_{left} the parameters of the corresponding curves, it is possible to operate constraint-based deformations. More specifically, if the user modifies one of the lateral curves, also the central one will change in such a way that its area is still the sum of the areas of the two lateral curves. The user can modify the two lateral curves by applying consecutive transformations as desired, even exasperating the constraint until the curves come out of the screen, as it is shown in Figure 2.13.

2.7 Concluding remarks

The case studies shown in Section 2.6.1 were given by the GRAVITATE European project [3]. The aim of this project was to provide a set of tools to increase and promote the use of semantic-

based solutions for responding to archaeological research queries. In addition, it was required to automatically find similarities between fragments scattered in different museums, with the goal of virtually reconstructing the artefacts.

The case study related to the Third Paradise artwork (Section 2.6.2) is developed in the INTER-CH project [4], whose goal was to create an interactive experience for museum visitors to communicate the message and content of the artwork in an innovative way.

The method and the results on plane curves presented in this chapter have been developed during the first year of my PhD. The dictionary expansion can be found in [133] and the work was presented at the Smart Tools and Applications in Graphics conference 2019 (STAG 2019). The results regarding the application to the archaeological domain are presented in [135], in which we show more examples of compound curves and similarity analysis. Finally, in [137] we introduce the application to the recognition of motifs and their visualization and manipulation on a multi-touch smart table. Specifically, more examples of the deformation of symbols are shown, exploring their geometrical features through the parametric formulation of curves. The work was presented at Smart Tools and Applications in Graphics conference 2020 (STAG 2020).

In conclusion, this technique is useful in some application contexts, such as the recognition of geometric patterns in archaeological finds, the analysis of brand logos, and the interpretation of artistic elements. Indeed, the method we developed offers many advantages: it can recognise a pattern of feature curves, even in the presence of noise and partial data; it provides the parameters and the equations of the recognised feature curves, and then it can detect the similarity of decorations and permits the exploration of their geometry. Furthermore, in the case of archaeological finds, the outcome of this method can be used to support the automatic annotation of the digital fragments models.

In addition, our curve recognition method was the only technique proposed in the SHREC'19 retrieval contest [113] able to deal with feature curve comparison. In the literature, other methods are either specific to a family of curves (e.g., spirals [72] or parabolic curves [105]) or focus on local curve fitting (e.g., spline-based curve fitting).

However, our method requires to have in input the family of curves to be used for the recognition, it needs to know a priori what kind of patterns to look for. This suggestion can be provided by the user through a template or by choosing in a curves catalogue the one or the ones most similar to what is sought. In the archaeological domain, this suggestion can come directly from the expert, through a template or a drawing, or from the technical documentation associated with the find.

2.8 Related publications

- E.Moscoso Thompson, G. Arvanitis, K. Moustakas, N. Hoang-Xuan, E. R. Nguyen, M. Tran, Thibault Lejemble, Loic Barthe, Nicolas Mellado, C. Romanengo, S. Biasotti, B. Falcidieno,

SHREC'19 track: Feature Curve Extraction on Triangle Meshes, EG19 Workshop 3DOR19 proceedings, 2019.

- C. Romanengo, S. Biasotti, B. Falcidieno, *HT-based Recognition of Patterns on 3D Shapes Using a Dictionary of Mathematical Curves*, STAG2019 proceedings, 2019 – orally presented work.
- C. Romanengo, S. Biasotti, B. Falcidieno, *Recognising decorations in archaeological finds through the analysis of characteristic curves on 3D models*, Pattern Recognition Letters, vol. 131, pp. 405-412.
- C. Romanengo, E. Brunetto, S. Biasotti, C. E. Catalano, B. Falcidieno, *Recognition, Modelling and Interactive Manipulation of Motifs or Symbols Represented by a Composition of Curves*, STAG2020 proceedings, 2020 – orally presented work.

RECOGNITION, EXTRACTION AND REPRESENTATION OF SPACE CURVES

In this chapter, we address the problem of recognising space curves on the surface of 3D digital models (meshes and point clouds) and providing their mathematical representation. When 3D models are acquired by scanning real objects, the resulting geometry does not explicitly encode these curves, especially when it is affected by noise, due to measurement uncertainty and sampling resolution, or misses some parts, due to the occlusion during the acquisition or other factors. For these purposes, we make use of the Hough transform, which is well known for recognising curves in the plane and surfaces in space but has not yet been sufficiently explored for space curves. Compared to the previous chapter, the challenge addressed here is how the HT can practically deal with space curves, i.e., the codimension 2 case in the space.

In the first part of this chapter, we introduce the dictionary available in our method (Section 3.2), dividing it into two classes, and we detail the main steps of our approach (Section 3.3) dealing with both implicit and parametric equations. In Section 3.4, we show some tests on digital models of real objects, while in Section 3.5 we provide a comparative analysis of the two strategies, in implicit and parametric representations. Finally, we show an example of the application of our algorithm that exploits the parametric expression of curves provided by our algorithm and inserts them directly into the model (Section 3.6).

3.1 Previous work

Space curve fitting. Beyond the context of the HT, space curve fitting is still an open problem. Whilst the fitting problem is largely addressed in the literature for plane curves [58, 122]

less effort has been made for space curves, probably because there are not the large curve atlases available on the contrary for plane curves [150]. Moreover, only a few techniques are extensible to this problem. Indeed, projections onto planes of particular families of curves have often been used in this context as well. For instance, we mention the piece-wise splines of low-degree polynomials like the spline approximation in [42]; however this approach needed a local, planar curve interpolation, therefore it was not able to recognise entire curves and to complete missing parts. Piece-wise splines of low-degree polynomials are the most common curves that can be combined to construct a space curve [63], for instance, cubic B-spline curves [149]. Often, point interpolation is combined with provided tangent and curvature information in the curve points/knots, such as in the geometric Hermite interpolation problem [86, 118, 158] or minimal variation of curvature constraints [130], or exact measurement of the arc-length [8, 59, 60].

Some methods fitted the data with some specific family of curves, such as Samper et al. did in [141], for studying the best curve fitting the Gaudí architecture among three conical and three hyperbolic-cosine curve types; Harary and Tal did in [73] and [72], for fitting line drawings and silhouettes with, respectively, *3D Euler spirals* and *natural 3D spiral*; Lv et al. did in [106] with nose curves defined as geodesic curves crossing the nose bridge on triangle meshes. However, these methods define only specific case-driven solutions and algorithms for setting the curve parameters.

To the best of our knowledge, also methods based on a learning process are limited to planar curves or curves that are projected on a plane, and to curves represented as groups of polylines. For instance, approaches based on co-occurrence analysis relied on an interactive learning phase or assumed there are repetitions for every type of curve to be identified and sketched [66]. Due to the characteristics of the detected curves, this kind of methods was adopted mainly for recognising parts of buildings (such as windows, doors, etc.) and features in architectural models that are similar to strokes. However, low-degree polynomials cannot span a large number of points, therefore many small segments need to be blended together to build the desired curve [98], and such a decomposition is not unique, for instance, it depends on the starting point.

Hough Transform for space curve. The use of HT for space curves – i.e. the approach to elements of codimension 2 in space – is still in its infancy, and it has been addressed only in the last few years, but most of the proposed methods do not exploit the large variety of possible curves. For instance, in [117] the HT has been employed to identify recurring straight line elements on the walls of buildings. In that application, the HT is applied only to planar point sets and line elements are clustered according to their angle with respect to the main wall direction; in this sense, the Hough aggregator is used to select the directions of the feature lines (horizontal, vertical, slanting) one at a time. A first attempt to recognise curvilinear space profiles from their plane projections is provided in [143]. A relevant part of the described procedure consists in the definition of a suitable projection of a 3-dimensional image onto a coordinate plane. To test the

methodologic validity of this approach, it is applied to the recognition of a given Viviani's curve and of a spherical curve with three convexities.

3.2 Families of space curves

As in the case of plane curves, the method for space curves requires in input a family of curves \mathcal{F} . Therefore, we define a dictionary of space curves dividing it into two classes of curves suitable for our case studies. *Type I curves* are families of space curves equipped with a known representation. In Section 3.2.1 the families of curves belonging to type I are listed. The set of space curves with an explicit representation is quite limited, while in the literature there is a large variety of plane curves [150]. We take advantage of this richness by considering as a second type of space curves (*type II curves*) the intersection of a paraboloid and a cylinder having a plane curve as its directrix. Section 3.2.2 describes how to obtain type II curve representations.

3.2.1 Space curves of type I

The space curves listed in this section have an explicit representation. Although non-exhaustive, this list constitutes the first set of space curves suitable for curve recognition. For each curve, the main geometric characteristics and the parameters that drive the HT are highlighted.

- The *helix* of implicit and parametric equations

$$I: \begin{cases} b^2 x^2 \left(\frac{z}{c}\right)^{\frac{2m}{s}} + a^2 y^2 \left(\frac{z}{c}\right)^{\frac{2n}{s}} - a^2 b^2 \left(\frac{z}{c}\right)^{\frac{2m}{s}} \left(\frac{z}{c}\right)^{\frac{2n}{s}} = 0 \\ x - a \left(\frac{z}{c}\right)^{\frac{n}{s}} \cos\left(\left(\frac{z}{c}\right)^{\frac{1}{s}}\right) = 0 \end{cases}, \quad P: \begin{cases} x = at^n \cos t \\ y = bt^m \sin t \\ z = ct^s \end{cases}$$

is a space curve that depends on three real positive parameters a , b and c , with $c \neq 0$. Using different values of n and m , different families of curves are obtained. The *helix* curve is an unbounded, connected curve. It is a space curve that always lies on a symmetrical surface with respect to the z -axis. In the special case of the cylindrical helix corresponding to the selection of the parameters $n = m$ and $a = b$, we take advantage of the cylindrical system of coordinates. The curve is formulated as follows:

$$\begin{cases} \rho - a\theta^n = 0 \\ z - c\theta^s = 0. \end{cases}$$

- The *sphero-cylindrical curve* is a space curve corresponding to the intersection of a sphere, centred in the origin O and radius a , with the elliptical cylinder with half-axes of lengths b and c (respectively x -axis and y -axis) and axis of symmetry at distance d from O . The

implicit and parametric equations of this curve are:

$$I: \begin{cases} x^2 + y^2 + z^2 - a^2 = 0 \\ c^2(x-d)^2 + b^2y^2 - b^2c^2 = 0 \end{cases}, \quad P: \begin{cases} x = d + b \cos t \\ y = c \sin t \\ z = \pm \sqrt{a^2 - (d + b \cos t)^2 - c^2 \sin^2 t} \end{cases}.$$

It depends on four real positive parameters a, b, c and d . The *hippopede* is a special case of the sphero-cylindrical curve, in which the distance d of the cylinder axis from the origin O is equal to $a - b$.

- The curve called *clelia* is obtained, in its generalised form, by intersecting a Plücker's conoid $z = a \sin n\theta$ with an ellipsoid of centre O and x, y -half-axes of equal length a . The implicit (in cylindrical coordinates) and parametric equations of this curve are:

$$I: \begin{cases} \rho - a \cos n\theta = 0 \\ z - b \sin n\theta = 0 \end{cases}, \quad P: \begin{cases} x = a \cos nt \cos t \\ y = a \cos nt \sin t \\ z = b \sin nt \end{cases}.$$

It depends on two positive real parameters a and b .

- Another space curve of type I is a curve that resembles the edge of a curved circular pancake then called the *pancake curve*. It is obtained by intersecting a cylinder with a parabolic cylinder having perpendicular axes. The implicit and parametric equations of this curve are:

$$I: \begin{cases} x^2 + y^2 - a^2 = 0 \\ a^2z - b(2x^2 - a^2) = 0 \end{cases}, \quad P: \begin{cases} x = a \cos t \\ y = a \sin t \\ z = b \cos 2t \end{cases}.$$

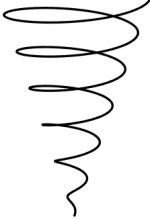

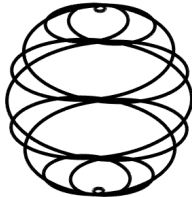
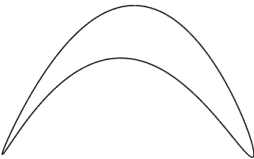
This curve depends on two real positive parameters a and b .

Table 3.1 summarises the spatial curves described so far with their implicit and parametric equations.

3.2.2 Space curves of type II

We define a larger variety of space curves by taking advantage of the richness of plane curves. To do that, the space curve representation proposed in this section is modelled as the intersection of a paraboloid and a cylinder with a plane curve (presented in Section 2.2.1) as its directrix. Therefore, space curves of type II are those curves that can be recognised through such a ploy that uses simpler primitives. This approach generalises the idea of projections of points on the regression plane (see Chapter 2) and improves the quality of the approximation of the spatial profile.

Table 3.1: A set of space curves of type I expressed in both implicit and parametric forms.

<p style="text-align: center;">helix curve</p>  $\text{I:} \begin{cases} b^2 x^2 \left(\frac{z}{c}\right)^{\frac{2m}{s}} + a^2 y^2 \left(\frac{z}{c}\right)^{\frac{2n}{s}} - a^2 b^2 \left(\frac{z}{c}\right)^{\frac{2m}{s}} \left(\frac{z}{c}\right)^{\frac{2n}{s}} = 0 \\ x - a \left(\frac{z}{c}\right)^{\frac{n}{s}} \cos\left(\left(\frac{z}{c}\right)^{\frac{1}{s}}\right) = 0 \end{cases}$ $\text{P:} \begin{cases} x = at^n \cos t \\ y = bt^m \sin t \\ z = ct^s \end{cases}$	<p style="text-align: center;">sphero-cylindrical curve</p>  $\text{I:} \begin{cases} x^2 + y^2 + z^2 - a^2 = 0 \\ c^2(x-d)^2 + b^2 y^2 - b^2 c^2 = 0 \end{cases}$ $\text{P:} \begin{cases} x = d + b \cos t \\ y = c \sin t \\ z = \pm \sqrt{a^2 - (d + b \cos t)^2 - c^2 \sin^2 t} \end{cases}$
<p style="text-align: center;">clelia</p>  $\text{I:} \begin{cases} \rho - a \cos n\theta = 0 \\ z - b \sin n\theta = 0 \end{cases}$ $\text{P:} \begin{cases} x = at^n \cos t \\ y = bt^m \sin t \\ z = ct^s \end{cases}$	<p style="text-align: center;">pancake curve</p>  $\text{I:} \begin{cases} x^2 + y^2 - a^2 = 0 \\ a^2 z - b(2x^2 - a^2) = 0 \end{cases}$ $\text{P:} \begin{cases} x = a \cos t \\ y = a \sin t \\ z = b \cos 2t \end{cases}$

In practice, a constructive strategy to build new families of space curves is proposed. To limit the number of parameters that define a quadric surface, the set of points \mathcal{P} is roto-translated in the origin of the Cartesian coordinates and we use families of paraboloids, represented in the form: $z = cx^2 + dy^2$. To do this, the regression plane of the set \mathcal{P} is calculated and then rotated so that the normal of the regression plane coincides with the z axis of the Cartesian coordinate system. The generic parametric equation of this type of space curve is

$$(3.1) \quad \begin{cases} x = f(a, b, t) \\ y = g(a, b, t) \\ z = c(f(a, b, t))^2 + d(g(a, b, t))^2 \end{cases}$$

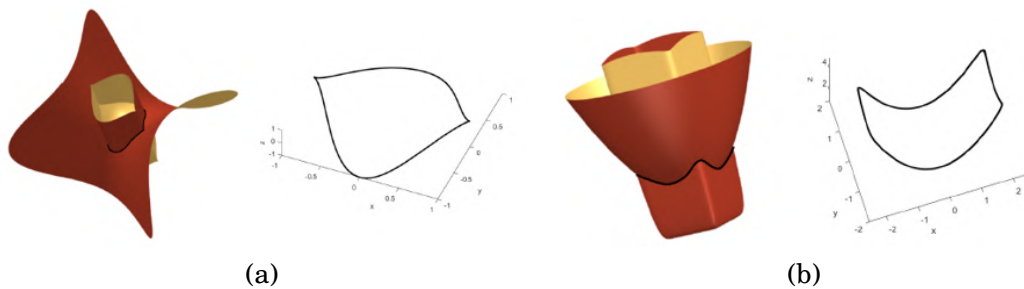


Figure 3.1: In (a) an example of an intersection between a hyperbolic paraboloid and a cylinder with a mouth curve as its directrix, with the same axis; in (b) an example of the intersection between an elliptical paraboloid and a cylinder with a Lamet curve as its directrix, with the same axis.

where $(f(a, b, t), g(a, b, t))$ is a parametric representation of a family of plane curves. Note that plane curves are a special case of this class, in which the paraboloid corresponds to a degenerate quadric surface. An implicit representation of type II space curves can be derived directly from the parametric expression or it can be constructed as an intersection of the cylinder generated from a plane curve and the surface $(z - cx^2 - dy^2 = 0)$. Figure 3.1 provides two examples of type II curves.

3.3 A method for recognising space curves

Some working hypotheses are assumed by our HT-based recognition method for space curves. In case a curve $\mathcal{C}_{\mathbf{a}}$ is in parametric form, it is assumed that the system defining $\mathcal{C}_{\mathbf{a}}$ with respect to the Cartesian coordinates x , y , and z can be analytically solved with respect to the parameter variables $\mathbf{a} = (a_1, \dots, a_n)$. In the case of curves in implicit form, it is assumed that the functions F_1 and F_2 in the definition of $\mathcal{C}_{\mathbf{a}}$ and $\Gamma_{\mathbf{p}}$ are analytical with respect to the Cartesian coordinates x , y , z and the parameter variables $\mathbf{a} = (a_1, \dots, a_n)$, $n \geq 2$.

Also, in this case, the method is independent of the technique adopted to select the feature points and the data representation (both meshes or point clouds).

In this section, the four main steps of the algorithm for our HT-based space curve recognition are briefly summarised. It requires as input a family of curves, some thresholds to establish if a set of feature points can be grouped or not and to assess the quality of the recognition result, an array $\Delta_i = (\Delta_{i,1}, \dots, \Delta_{i,n})$ of integers to define the size of initial discretisation of \mathcal{T} .

1. *Extraction of the feature points.* Unless point clusters are provided directly, a 3D model (a mesh or a point cloud) is preprocessed to identify sets of feature points as shown in the first step of the method for plane curves described in Section 2.3. In this case, only the mean curvature is used as the filter criterion and then, the points are then grouped into connected components through a clustering operation, adopting the method *Density-Based Spatial*

Clustering of Application with Noise DBSCAN, [57]. The outcome of this preprocessing phase is a set of num clusters \mathcal{P}_i of feature points, $i = 1, \dots, num$.

2. *Recognition of the feature curve.* The generalised HT technique described in Section 1.3.4 is applied to each cluster \mathcal{P}_i . In this case, considering the notation introduced in Section 1.3.4, the parameter space $\mathcal{T} = \prod_{j=1}^n [(1 - \bar{p})a_j^*, (1 + \bar{p})a_j^*]$ is divided into intervals of length δ_j derived from the input array Δ_i as $\delta_j = \frac{2\bar{p}}{(\Delta_i)_j} a_j^*$, where \bar{p} is a percentage which is set differently depending on the family considered (a typical value is 0.1). The estimation of the accumulator function \mathcal{H}_i changes according to the representation of the family \mathcal{F} .
3. *Evaluation of the approximation accuracy.* To automatically determine if a curve $\mathcal{C}_{\bar{\mathbf{a}}}$ satisfactorily approximates the cluster of points \mathcal{P}_i , we define the *Mean Fitting Error* (MFE) as:

$$(3.2) \quad MFE(\mathcal{P}_i, \mathcal{C}_{\bar{\mathbf{a}}}) := \frac{1}{|\mathcal{P}_i|} \sum_{\mathbf{p} \in \mathcal{P}_i} d(\mathbf{p}, \mathcal{C}_{\bar{\mathbf{a}}})/l_i,$$

where d is the Euclidean distance and l_i is the diagonal of the minimum bounding box containing \mathcal{P}_i . Note that the MFE provides an estimation of the deviation between $\mathcal{C}_{\bar{\mathbf{a}}}$ and \mathcal{P}_i and it is a size-independent percentage. The smaller it is, the better the curve approximation. If \mathcal{H}_i presents two or more local maxima, this distance has to be calculated for each potential curve solution. The curve having the lowest error is kept.

4. *Goodness of the approximation and refinement strategy.* The output of the first three steps is the best fitting curve $\mathcal{C}_{\bar{\mathbf{a}}} \in \mathcal{F}$ and the quality of the approximation MFE. The value of the MFE has to be compared with the fitting threshold ε (typical values of ε are 3 – 5%). In the case the value of MFE is smaller than ε the curve is a good solution, while if the MFE is greater than 2ε the family \mathcal{F} is excluded and the algorithm starts again by selecting another family of curves of our dictionary. Finally, if $MFE \in (\varepsilon, 2\varepsilon)$ a new denser sampling of the parameter space is considered unless the values $(\Delta'_i)_j$ exceed a value C that represents the highest possible refinement for each parameter (in our experiments, we observed that a good choice for the value C is 256). In particular, for curves expressed in implicit form the refinement could adaptively split only the cells with uncertainty, while in the case of curves in parametric form, all cells are bisected (i.e., $\Delta'_i = 2\Delta_i$). The refinement process repeats until $MFE \in (\varepsilon, 2\varepsilon)$ and $(\Delta'_i)_j$ is smaller than C for all j . The refinement process successfully ends if a curve $\mathcal{C}_{\bar{\mathbf{a}}} \in \mathcal{F}$ with $MFE \leq \varepsilon$ is found. A new family of curves \mathcal{F}' is selected if the values of the array Δ'_i become too large or $MFE > 2\varepsilon$. Note that in the plane curve recognition algorithm (Chapter 2), only the GoF has been considered for the evaluation of the approximation accuracy. However, using the MFE makes the algorithm more automatic, since it does not require any input parameter.

Figure 3.2 provides an illustrative pipeline of our method.

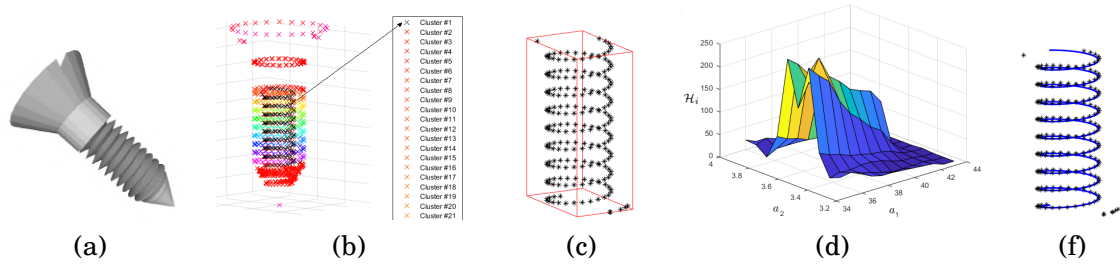


Figure 3.2: In (a) a 3D model of a screw; in (b) the clusters of points which correspond to a mean curvature value less than 0.1. The column on the right lists some of the clusters found, identified by a number and a colour (black for cluster #1). In (c) the cluster \mathcal{P}_1 represented with the minimum bounding box (in red) that contains it; in (d) the accumulator function \mathcal{H}_i ; in (e) the recognised curve, corresponding to the maximum value of \mathcal{H}_i .

3.3.1 Estimation of the Accumulator Function

The estimation of the accumulator function changes according to the type of representation of the family of curves. In this section, we provide a description of the method for curves in parametric and implicit form.

Estimation of the accumulator function for space curves in parametric form. In case the family of curves \mathcal{F} is represented in parametric form, it is supposed that the system of equations can be analytically solved with respect to the parameters \mathbf{a} and the expressions of $\mathcal{C}_{\mathbf{a}}$ and $\Gamma_{\mathbf{p}}$ are known. Note that, the voting algorithm, originally proposed in [109] for families of plane curves with only two parameters, here is formulated to deal also with space curves with an arbitrary number of parameters.

Given a family of curves in the parametric form represented as follows:

$$\mathbf{x}(t) = M(t)\mathbf{a}$$

where $\mathbf{x}(t) = (x(t), y(t), z(t))^T$, $\mathbf{a} = (a_1, \dots, a_n)^T$ and $M(t)$ is a matrix $3 \times n$ that depends on the variable of the parametric equations t , then Γ_P can be calculated using the Moore-Penrose pseudo-inverse, as:

$$(3.3) \quad \mathbf{A}(t) = M^\dagger(t)\mathbf{x}_{\mathbf{p}}.$$

In practice, we solve the system in equation (3.3) for each cell of the space of the parameters \mathcal{F} . The solution of the linear system can be optimised when the matrix $M(t)$ is invertible. In particular, in case of space curves, this means that the matrix $M(t)$ is 3×3 and its determinant is not zero, thus implying that the parameters \mathbf{a} must be of the form $\mathbf{a} = (a_1, a_2, a_3)$. In these cases, several optimisations can be put in place, for instance, using Cramer's rule as in [134] or other direct methods. Using the notation introduced in Section 1.3.4, the evaluation of the

intersection is translated into an inequality between the components of the sample points of $\Gamma_{\mathbf{p}}$ and the coordinates of the endpoints of the cells for each component:

$$a_{j,k_j} - \frac{\delta_j}{2} \leq A_j(t) < a_{j,k_j} + \frac{\delta_j}{2}$$

$k_j = 0, \dots, N_j - 1$. This inequality must be verified for at least one parameter t for each component j of $\mathbf{A} = (A_1, \dots, A_n)$.

Estimation of the accumulator function for space curves in implicit form. When space curves are represented in implicit form, we consider that $\Gamma_{\mathbf{p}}$ can be evaluated through the zero loci of its implicit functions and $\Gamma_{\mathbf{p}}$ crosses the cell $c(k_1, \dots, k_n)$ if its value on the point centre of the cell is zero.

Indeed, for estimating if $\Gamma_{\mathbf{p}}$ crosses $c(k_1, \dots, k_n)$, 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 [152], where it has been formally proven that for each cell $c(k_1, \dots, k_n)$, a bound of $\Gamma_{\mathbf{p}}$ depends on the norms of the Jacobian and Hessian matrices of \mathcal{F} and the Moore-Penrose pseudo-inverse of the Jacobian matrix.

Referring to the equation (1.2), for each point \mathbf{p} we consider the vectors $F_{\mathbf{p}}(\mathbf{A}) = (F_{1,\mathbf{p}}(\mathbf{A}), F_{2,\mathbf{p}}(\mathbf{A}))$ and $F_{\mathbf{p}}(\dot{\mathbf{a}}) = (F_{1,\mathbf{p}}(\dot{\mathbf{a}}), F_{2,\mathbf{p}}(\dot{\mathbf{a}}))$, namely $\dot{\mathbf{a}}$ the point center of the considered cell, and $\mathbf{B} = \{\mathbf{A} \in \mathbb{R}^n \mid \|\mathbf{A} - \dot{\mathbf{a}}\|_{\infty} \leq \varepsilon\}$ the ball centered at point $\dot{\mathbf{a}}$ of radius $\varepsilon = \max_{j=1, \dots, n}(\delta_j)$. Then our cell is a subset of \mathbf{B} whose interior coincides with the interior of \mathbf{B} . Then, let us consider the Jacobians $Jac_{F_{1,\mathbf{p}}}(\mathbf{A})$ and $Jac_{F_{2,\mathbf{p}}}(\mathbf{A})$ and the Hessian matrices $H_{F_{1,\mathbf{p}}}(\mathbf{A})$ and $H_{F_{2,\mathbf{p}}}(\mathbf{A})$ of, respectively, $F_{1,\mathbf{p}}(\mathbf{A})$ and $F_{2,\mathbf{p}}(\mathbf{A})$. Let us fix

$$Jac_F(\mathbf{A}) = \begin{pmatrix} Jac_{F_{1,\mathbf{p}}}(\mathbf{A}) \\ Jac_{F_{2,\mathbf{p}}}(\mathbf{A}) \end{pmatrix}, \quad J = \begin{pmatrix} Jac_{F_{1,\mathbf{p}}}(\dot{\mathbf{a}}) \\ Jac_{F_{2,\mathbf{p}}}(\dot{\mathbf{a}}) \end{pmatrix} \quad \text{and} \quad H^{\infty} = \|(H_1, H_2)\|_{\infty}$$

with $H_1 = \max_{\mathbf{A} \in \mathbf{B}} \{\|H_{F_{1,\mathbf{p}}}(\mathbf{A})\|_1\}$ and $H_2 = \max_{\mathbf{A} \in \mathbf{B}} \{\|H_{F_{2,\mathbf{p}}}(\mathbf{A})\|_1\}$.

Finally, let us fix $J^{\infty} = \max_{\mathbf{A} \in \mathbf{B}} \{\|Jac_{F_{\mathbf{p}}}^{\dagger}(\mathbf{A})\|_{\infty}\}$.

If:

- $\|F_{\mathbf{p}}(\dot{\mathbf{a}})\|_{\infty} > \|J\|_{\infty}\varepsilon + \frac{3}{2}\varepsilon^2 H^{\infty} =: B_1$, then $\Gamma_{\mathbf{p}}$ does not cross the cell;
- $\|F_{\mathbf{p}}(\dot{\mathbf{a}})\|_{\infty} < \frac{2R}{J^{\infty}(2+6R J^{\infty} H^{\infty})} =: B_2$, with $R < \min\{\varepsilon, \frac{\sigma}{\sqrt{6}H^{\infty}}\}$ and σ the minimum singular value of J , then $\Gamma_{\mathbf{p}}$ crosses the cell;
- $B_2 < \|F_{\mathbf{p}}(\dot{\mathbf{a}})\|_{\infty} < B_1$, then neither Theorems 3.2 and 4.6 in [152] can be applied; this indeterminacy is reported in the accumulator function adding ξ instead of 1 in the corresponding entry.

Since the above quantities depend on the Jacobian and the Hessian matrices, the cell $c(k_1, \dots, k_n)$ that contributes to the computation of H must be a point for which these values are

non-trivial. In our implementation, we use the system *CoCoA* [6] for the symbolic manipulation of polynomials and matrices, since it gives an exact estimation of the zeros of a curve in an implicit form on the cells of the parameter space. Note that, compared to previous works [19, 154], in our implementation, it is necessary to manage a vector of functions and therefore the non-trivial calculation of the Jacobian and Hessian matrices.

3.3.2 Computational complexity

As for the case of plane curves, the complexity of the algorithm depends on the size of the discretization of the parameter space \mathcal{T} . As previously discussed, \mathcal{T} is subdivided into $M = \prod_{j=1}^n \delta_{i,j}$ cells, with n the number of parameters of the family \mathcal{F} , and the accumulator function \mathcal{H}_i has the same dimension of \mathcal{T} . The evaluation of \mathcal{H}_i requires $O(M)$ operations for each point \mathbf{p}_l , $l = \dots, L$, in a cluster \mathcal{P}_i . Then, the computational complexity for each cluster is $O(ML)$.

The theoretical complexity is the same for curves in implicit and parametric form, but in practice, the first case requires symbolic computations, that can slow down considerably the algorithm. Indeed, in this case, we need to evaluate, once for each point of the curve, the symbolic expression of the Jacobian, the Moore-Penrose pseudo-inverse and the Hessian matrices because these entities are involved in the determination of the zero sets of an implicit function. For curves expressed in parametric form, the estimation of the HT accumulator function corresponds to the explicit estimation of $\Gamma_{\mathbf{p}}$, which is done either solving a linear system or analytically solving the system a priori and, therefore, it corresponds to the evaluation of a set of inequalities.

In conclusion, the computational complexity seems to be the same as the plane curves method introduced in Section 2.3. However, in the implementation of the method for space curves in the implicit form, it is necessary to manage a vector of functions and therefore the non-trivial calculation of the Jacobian and Hessian matrices.

3.4 Testing the method on digital models of real objects

The examples of models selected here are mainly meshes because the repositories considered are mainly made up of these models. The models shown here represent man-made objects, natural objects, and industrial components, mainly stored in various repositories (e.g. Visionair [1], Sketchfab¹ and the STARC [2] repository). For these models, a representation in smooth patches such as B-splines or other CAD representations is not available, therefore, the exact, mathematical formulation of their space curves is not known. Figure 3.3 shows the results obtained by applying our method to eight 3D models. Once the feature points are aggregated in clusters, the curve recognition method described in Section 3.3 is applied to each group. The two columns show, respectively, the set of feature points identified by the clustering step with the recognised curve and the points highlighted on the corresponding model. In particular, Figures

¹<https://sketchfab.com/>

3.3 present some examples of recognition by means of families of classical space curves. The feature points in Figures 3.3(a,d,f,g,h) are approximated with a family of helices (see Section 3.2) with different values of n , m and s : $n = \frac{2}{5}$, $m = \frac{1}{5}$, $s = \frac{16}{25}$ for Figure 3.3(a), $n = m = \frac{1}{2}$, $s = \frac{3}{2}$ for Figure 3.3(d), $n = m = \frac{1}{2}$, $s = 1$ (that is a parabolic helix) for Figure 3.3(f), $n = 1$, $m = \frac{3}{5}$ and $s = \frac{7}{10}$ for Figure 3.3(g) and $n = m = \frac{3}{2}$, $s = \frac{9}{4}$ for Figure 3.3(h). The points selected on the model in Figure 3.3(b) are recognised with a clelia curve with $n = \frac{4}{5}$, while the points selected on the model in Figure 3.3(c) are recognised with a pancake curve. Finally, the points in Figure 3.3(e) are approximated with a family of sphero-cylindrical curves, called hippopede in the special case in which $d = a - b$.

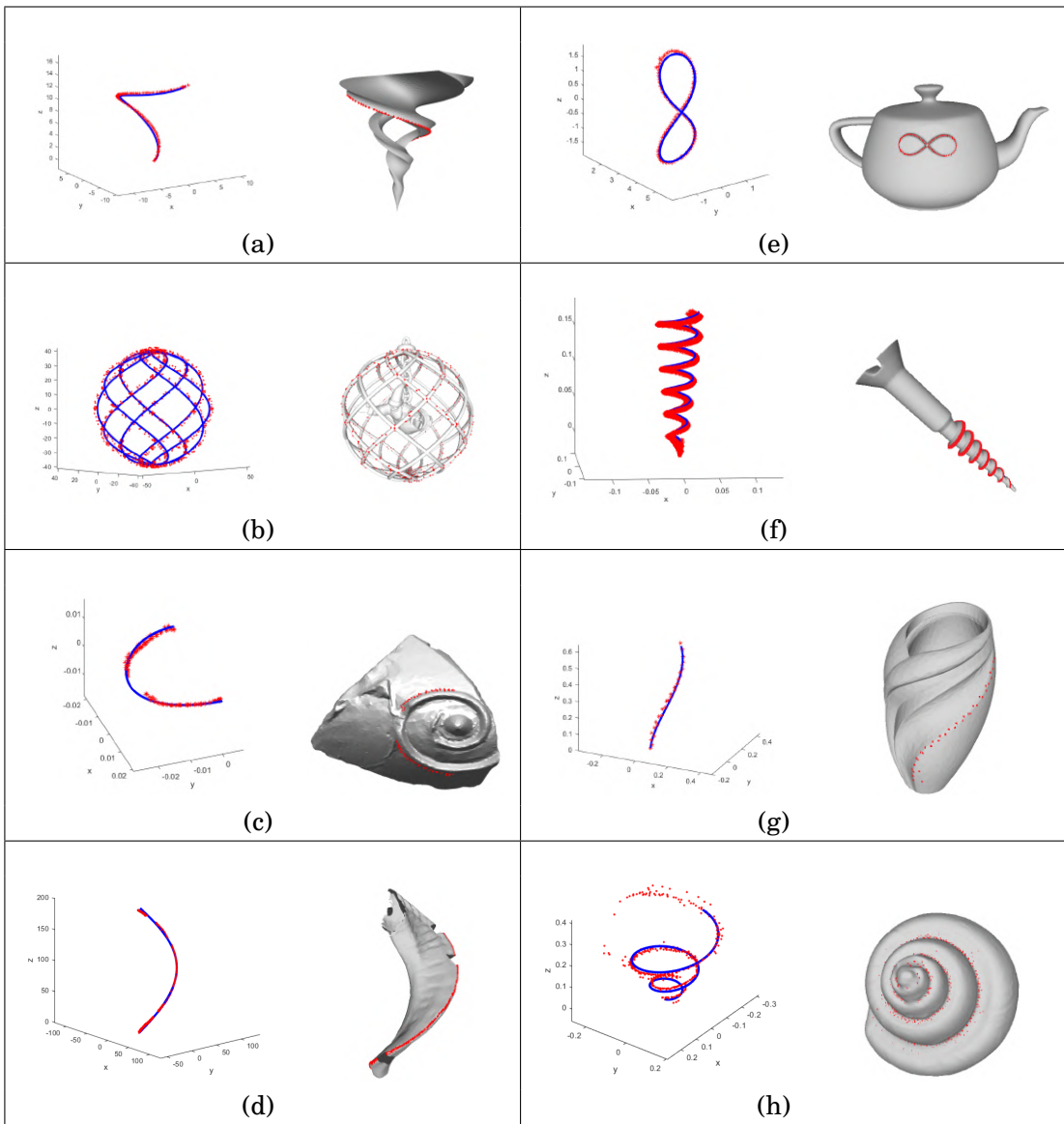


Figure 3.3: Results obtained by the recognition method applied to some 3D models. In (a, d, f, g, h) various types of helices, in (b) a clelia curve and in (c) a pancake curve.

Note that all the curves used in the experiments of this section have a parametric representation, which is linear with respect to the parameters \mathbf{a} , with the exception of the sphero-cylindrical and hippopede ones, recognised in the examples in Figure 3.3(e). In these cases, the expression of $\Gamma_{\mathbf{p}}$ needs to be derived family by the family of curves. As an example, the parametric expression of $\Gamma_{\mathbf{p}}$ is obtained by solving the system of equations that corresponds to the hippopede curve:

$$\Gamma_{\mathbf{p}} : \begin{cases} A = \sqrt{x_P^2 + y_P^2 + z_P^2} \\ B = -\frac{x_P}{1-\cos t} + \frac{\sqrt{x_P^2 + y_P^2 + z_P^2}}{1-\cos t} \\ C = \frac{y_P}{\sin t} \end{cases} .$$

Figure 3.4 shows some examples of feature points recognised with a family of curves of type II obtained as projections of families of plane curves on particular types of surfaces. The feature points in Figures 3.4(a,b) are approximated with a family of curves obtained by the intersection of a circular paraboloid and a cylinder with a 5-convexities and 20-convexities curve, respectively, as directrix:

$$I : \begin{cases} \rho - \frac{a}{1+b \cos 5t} = 0 \\ z - c\rho^2 = 0 \end{cases} , \quad P : \begin{cases} x = \frac{a}{1+b \cos 5t} \cos t \\ y = \frac{a}{1+b \cos 5t} \sin t \\ z = c \left(\frac{a}{1+b \cos 5t} \right)^2 \end{cases}$$

where the implicit equations are expressed in cylindrical coordinates. Their use implies a reduction of the degree of the curve and provides a general formula since the equation in Cartesian coordinates has to be derived based on the number of convexities.

The points in Figure 3.4(c) are recognised with the family of curves of equations:

$$I : \begin{cases} bx^m + a^m y^m - a^m b = 0 \\ z - cx^2 - cy^2 = 0 \end{cases} , \quad P : \begin{cases} x = at \\ y = \pm (b - bt^m)^{\frac{1}{m}} \\ z = ca^2 t^2 + c(b - bt^m)^{\frac{2}{m}} \end{cases}$$

obtained as the intersection of a paraboloid and a cylinder with a Lamet curve as its directrix.

The feature points in Figures 3.4(d) are approximated with a family of curves obtained by the intersection of a circular paraboloid and a cylinder with a geometric petal of type (A) as directrix:

$$I : \begin{cases} \rho - a + b \cos^{2n} \theta = 0 \\ z - c\rho^2 = 0 \end{cases} , \quad P : \begin{cases} x = (a + b \cos^{2n} \theta) \cos t \\ y = (a + b \cos^{2n} \theta) \sin t \\ z = c (a + b \cos^{2n} \theta)^2 \end{cases}$$

where, also in this case, the implicit equations are expressed in cylindrical coordinates.

In particular, the recognition of space curves with families of type II extends the approach described in Chapter 2, which detects families of space curves necessarily projecting them on a plane. It also improves the approximation by adding the possibility of using surface primitives,

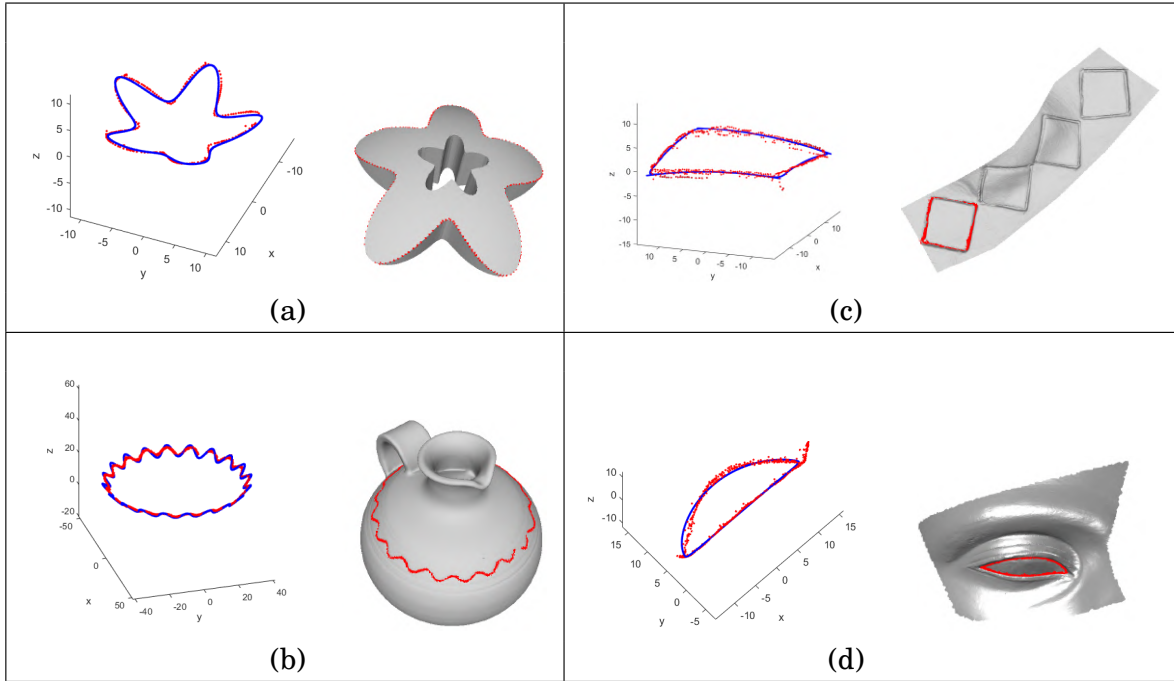


Figure 3.4: Results obtained by our method on some 3D models.

such as paraboloids. Considering the recognised curves of the models presented in Figure 2.3(b,c) and Figure 3.4(d,a), respectively, the use of a paraboloid better fits the set of feature points when the surface is curved, in particular in the star extremities and in the inner part of the eye. This result is also demonstrated by the value of the MFE: in Figure 3.4(d) it is 0.0131, while in Figure 2.3(b) it is 0.0180; in Figure 3.4(a) it is 0.0116, while in 2.3(c) it is 0.0129.

3.5 A comparative analysis for space curves

This section provides a comparative analysis between the methods for space curves in implicit and parametric forms. Tables 3.2, 3.3 and 3.4 compare the HT-based algorithms for the same families of space curves listed in Section 3.2, when the data are sampled on an exact mathematical expression of the curve or sampled on a random perturbation error of the 5% of the diagonal of the curve bounding box. Specifically, Table 3.2 lists the parameters of the mathematical curves, those recognised by the HT algorithm on samples of the exact mathematical curve and the ones of the same curve perturbed with noise. The results in Table 3.2 show that the algorithm in implicit form identifies exactly the parameters corresponding to the original curves, while the parametric version gets the exact parameters only in the case of sphero-cylindrical curves because the HT of these families is analytically solved and no approximations are introduced with the estimations of the Moore Penrose pseudo-inverse.

The quantitative distances between the identified curves and the samples are shown in

Table 3.2: Comparison between the mathematical curve parameters and those recognised by our algorithm over exact or perturbed curves, with respect to both the implicit and parametric formulation. The labels of the parameters are those used in the curve formulation in Section 2.2. The values in bold represent the target parameters. The symbol = indicates when the parameter identified by the HT-based algorithm equals the target one.

Mathematical Curve		HT on exact curve samples		HT on perturbed samples	
Curve	Parameters	implicit	parametric	implicit	parametric
(1) conical helix	a = 1.5 b = 1.5 c = 1	=	$a = 1.485$ $b = 1.515$ =	=	$a = 1.485$ $b = 1.515$ $c = 0.990$
(2) parabolic helix	a = 1.5 b = 1.5 c = 1	=	$a = 1.495$ = =	=	$a = 1.490$ $b = 1.510$ =
(3) cylindrical helix	a = 2 b = 1	=	$a = 1.980$ $b = 0.990$	=	$a = 1.960$ $b = 0.980$
(4) pancake curve	a = 2 b = 1	=	$a = 1.980$ $b = 1.010$	=	$a = 1.980$ $b = 1.060$
(5) clelia, $n = 0.8$	a = 2 b = 2.1	=	$a = 1.980$ =	=	$a = 1.940$ $b = 2.058$
(6) spherocylindrical curve $k = 2$	a = 2 b = 1 c = 0.5	=	=	=	=
(7) hippopede	a = 2 b = 1 c = 0.9	=	=	=	$b = 1.020$ =

Tables 3.3 and 3.4, considering the same measures of Section 2.5. More in detail, in the columns of Tables 3.3 and 3.4 we list, from left to right, the family of curves; the GoF , the d_{dHaus} , the $mean(D_{knn})$ distances; the computational time in seconds. All these measures are reported for both the implicit and parametric curve expressions. Finally, in the rightmost column, we indicate the number of points L in each cluster. The measurements in Table 3.3 refer to the quality of the HT approximation when applied to a set of L points uniformly sampled on a mathematical curve. We evaluate the quality of the curve approximation by uniformly sampling the curve recognised with 200 points (in general, these points do not correspond to the L samples used to fit the curve) and estimate the GoF , $mean(D_{knn})$ and direct Hausdorff distances between these samples and the original set of points. For simplicity of notation in the Tables, we represent with 0.0 a distance when its value is close to the machine precision.

Similarly, the measures in Table 3.4 report the distances of the HT curve approximation with respect to a set of L randomly perturbed samples of a smooth curve; note that these L points do not exactly lie on the original curve. The GoF measure and the $mean(D_{knn})$ highlight that, even in this case, better precision is achieved by the HT-based algorithm for approximating curves in implicit form with respect to the parametric ones. On the other hand, the algorithm for curves in

parametric form is computationally more efficient. Additionally, in practice, the family of curves in implicit form makes it simpler to complete the missing parts because it enables global profile recognition. The parametric form allows more accurate identification of the area of the curve that most closely resembles the profile.

The experiments are performed on the same laptop equipped with an Intel Core i7 processor (at 1.3 GHz). All the routines are implemented in MATLAB [111], with the exception of the symbolic calculations that are implemented on the open source toolkit, CoCoA [6]. The computational time of the HT-based recognition algorithm for curves in parametric form can be further optimised when a family of space curves can be expressed with three, linear, independent parameters. In this case, we can use more efficient linear system solvers rather than the Moore-Penrose pseudo inverse; for instance, we experimentally observed that if the analytic system derived from the curve equations is solved with Cramer’s rule as proposed in [134] the computational time decreases by 60% for a set of 400 feature points.

Table 3.3: Quality measures and computational time for the HT-based recognition algorithm for the same skew curves in implicit (I) and parametric (P) form. L represents the number of feature points. The curve numbers are the same as in Table 3.2.

Curve	GoF		d_{dHaus}		$mean(D_{knn})$		Time (s)		L
	I	P	I	P	I	P	I	P	
(1)	0.0	0.094	0.0	0.186	0.0	0.095	178	7	63
(2)	0.0	0.001	0.0	0.017	0.0	0.001	180	7	63
(3)	0.0	0.062	0.0	0.127	0.0	0.067	170	1.5	63
(4)	0.0	0.021	0.0	0.022	0.0	0.021	178	1	63
(5)	0.0	0.012	0.0	0.020	0.0	0.013	850	15	315
(6)	0.0	0.0	0.0	0.0	0.0	0.0	1700	4	64
(7)	0.0	0.0	0.0	0.0	0.0	0.0	1000	4	64

Our experiments confirm that the GoF distance generally provides a more satisfactory convergence rate. Indeed, such a distance has been tailored for contours in images that might be affected by some noise and outliers; indeed, it happens that some points of a cluster \mathcal{P}_i do not really lie on the profile we are looking for [109]. For this reason, the GoF distance was designed to be evaluated only on the most representative feature points. The $mean(D_{knn})$ yields an interpretation of the distances similar to the GoF, but differently from the GoF, it averages the distances between the fitting curve and the closest feature points. As displayed in Table 3.3, the GoF and the $mean(D_{knn})$ qualitatively reflect the same property when a uniform sampling of a mathematical curve is fitted with a smooth curve, while the average becomes more sensitive to perturbations when the curve samples are affected by some noise, see Table 3.4. On the contrary, the Hausdorff distance is designed to measure if two sets of points are globally close and

Table 3.4: Quality measures and computational time for the HT-based recognition algorithm for the same mathematical curves of Table 3.3, perturbed by the 5% of the diagonal of the curve bounding box. The curve numbers are the same as in Table 3.2. L represents the number of feature points.

Curve	GoF		d_{dHaus}		$mean(D_{knn})$		Time (s)		L
	I	P	I	P	I	P	I	P	
(1)	0.045	0.127	0.351	0.261	0.043	0.125	178	7	63
(2)	0.038	0.044	0.805	0.540	0.039	0.047	180	7	63
(3)	0.046	0.129	0.258	0.334	0.049	0.125	170	1.5	63
(4)	0.044	0.052	0.182	0.175	0.046	0.053	178	1	63
(5)	0.034	0.056	0.168	0.178	0.043	0.057	850	15	315
(6)	0.013	0.015	0.203	0.326	0.040	0.048	1700	4	64
(7)	0.048	0.014	0.160	0.168	0.048	0.053	1000	5	64

depends on the extrema kept by the Euclidean distance. As confirmed by our tests, this distance is sensitive to outliers and data perturbation, while it well reflects if two dense curve samples approximate each other.

3.6 Curve insertion into 3D meshes

Here we show how the recognised feature curves on the surface of a 3D model and their parametric equations can be exploited to locally remesh the model itself by inserting these curvilinear elements in a constrained way.

In the literature, when a feature curve is represented as a sequence of vertices on the mesh, if the points lie on mesh elements (edges or faces) these elements are split, and the points are inserted into the model constraining the curve edges to become mesh elements [34]. If the curve is a smooth curve and the underlying triangle mesh is dense enough, e.g., an Euler spiral detected over the laser scan reconstruction of an archaeological artefact [75], a visually pleasing representation of the curve can be obtained by opportunely moving the mesh vertices to fit the closest point on the spiral. In this case, the local mesh connectivity does not vary and only the vertices coordinates are modified to belong to the curve. Of course, this strategy is suitable to deal only with dense and well-shaped meshes. If the curve crosses some missing parts, as in the case of archaeological findings, a local retriangulation strategy can be considered; for instance, in [74] an ad-hoc hole-filling strategy was proposed in the case of the Euler spiral curve taking advantage of the feature curve properties (in the case of the Euler spiral, the tangent and the curvature are constant everywhere along the curve). In this case, a *bridge of triangles* is built along the curve segment to cross the mesh missing part and create two simpler holes that are then filled with a standard hole filling method [101].

Most of the methods that consider the insertion of a curvilinear constraint into a surface mesh, usually approximate the curve segments with linear edges. On the contrary, here we propose the use of both linear and curvilinear elements to exactly represent the curve segments. In this direction, the application is inspired by the seminal work in [83], where 2D curved triangle elements are inserted into a flat mesh. Differently from this method, thanks to the curve recognition method that acts directly on space curves, the method directly edits the surface of a 3D model, proposing a locally constrained re-mesh algorithm similar to the one used for sharp edges in [12]. Moreover, thanks to the flexibility of the HT, we are able to consider also non-polynomial curves thus exactly approximating also circles and ellipses, which would require NURBS approximations and was impossible with the Bézier curves used in [83]. If we are dealing with 3D models obtained from dense laser scans, our method permits the creation of meshes generally coarser than the original one and that exactly represent the curve recognised by the HT. However, our method can also be successfully used to thicken feature curves in very coarse models.

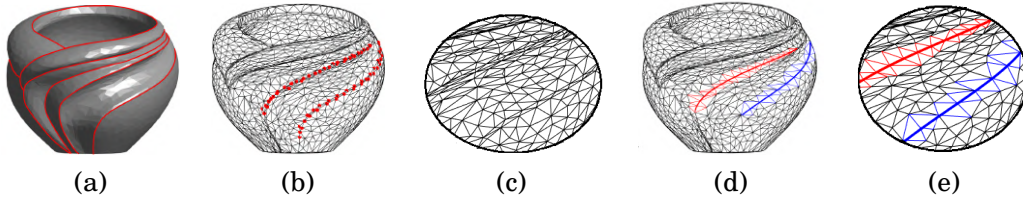


Figure 3.5: An overview of the curve insertion method. In (a) a 3d model of a vase with its feature curves is highlighted in red. In (b) its initial triangulation with the blue vertices corresponds to 2 clusters of points extracted by our method. In (c) a detail of the 2 feature curves (notice that these curves are characterised by a stripe of thin triangles). In (d) the insertion into the model of the two recognised curves and a detail (e) of the curve tessellation obtained near them.

In Figure 3.5 we show an overview of our method. In the following section we detail the steps of our method.

3.6.1 Feature curves insertion method

Once a set of feature curves has been recognised, it is possible to explicitly insert them as constraints into a 3D model, either to modify the local mesh tessellation or to emphasise some specific characteristic, or to complete some missing parts. In this phase, smooth, curve elements are inserted within a linear, triangle mesh, thus creating a hybrid mesh representation.

For the feature curve insertion phase, some restrictions are imposed on the type of surfaces we are able to manage. First, the model mesh \mathcal{M} is considered orientable and 2-manifold everywhere, a fact that is not necessary for HT-based curve recognition. Then it is imposed that every point \mathbf{p} on the curve $\mathcal{C}_{\mathbf{a}}$ is closer than β to \mathcal{M} and the intersection between \mathcal{M} and a ball centred in a point p of the curve ($\mathbf{p} \in \mathcal{C}_{\mathbf{a}}$) of radius β is a simply connected component. In the experiments, β is

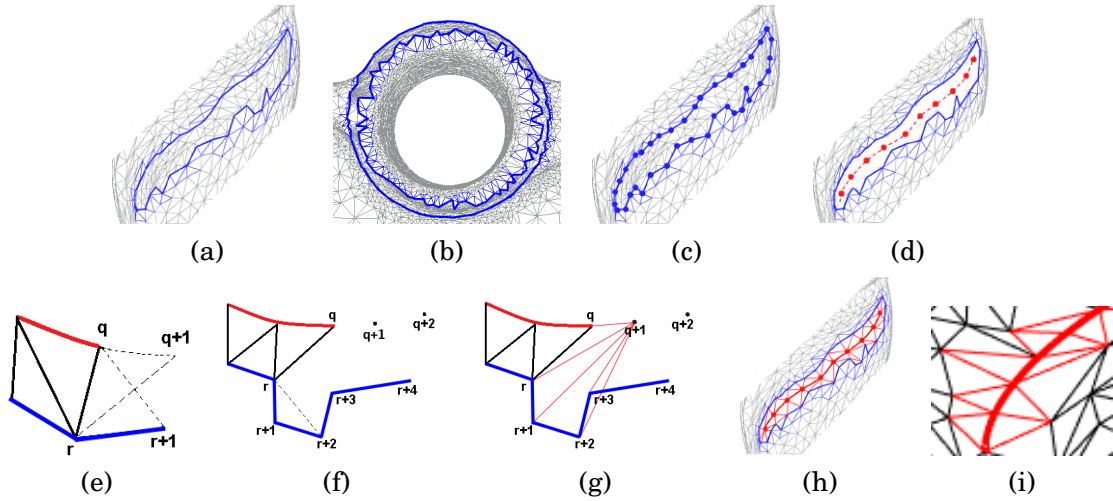


Figure 3.6: Pipeline of the curve insertion algorithm: (a,b) in blue we highlight the boundary \mathcal{B} obtained after the removal of the vertices close to $\mathcal{C}_{\bar{a}}$ for two feature curves recognised in the objects of Figure 3.5 and Figure 3.3(c), respectively; (c) identification of the vertices of the boundary \mathcal{B} of (a), highlighted in the picture with small blue dots; (d) the samples on the curve $\mathcal{C}_{\bar{a}}$ are shown with red dots; (e,f,g) insertion of the new vertices and the local triangulation criteria; (h) the final triangle mesh with curvilinear elements and in (i) a detail of the curve elements for the red curve in Figure 3.5(e)

selected in the range of 1% – 5% of the diameter of the bounding box of the model. Intuitively, we do not consider surfaces that are self-intersecting or almost bow on themselves and it is imposed that the curve projection on the mesh is locally injective.

The presented curvilinear meshing algorithm locally re-paves the part of the mesh closest to the fitting curve. The method can be summarised in the following steps: i) to identify the vertices of the mesh that are close to the curve to be inserted; ii) to remove the triangles that are incident to these vertices in order to create a hole around the curve; iii) to detect the boundary of the hole created in the mesh; iv) to sample the smooth curve so that a set of curve edges is created; v) to generate a valid mesh by sewing the curve samples and the boundary previously generated.

In Figure 3.6 we show the pipeline of the method whose five steps are detailed in the following.

1. *Identification of the vertices close to the recognised curve.* Once the curve $\mathcal{C}_{\bar{a}}$ that best approximates a set \mathcal{P} of feature points is identified, the vertices of the mesh that are at a lower distance to $\mathcal{C}_{\bar{a}}$ than a threshold β are selected, which depends on the bounding box size of the mesh itself.
2. *Mesh cutting around the curve.* The vertices identified in the previous step are removed from the mesh together with the faces incident to them. Therefore a mesh \mathcal{M}' with a hole is obtained (see Figure 3.6(a,b) showing the resulting meshes for two curves recognised in the objects of Figure 3.5 and Figure 3.7(c) respectively).

3. *Detection of the hole boundary.* Then, the vertices on the hole boundary \mathcal{B} are identified (see Figure 3.6(c) for the hole in 3.6(a)). Note that, as shown in the Figures 3.6(a,b) the boundary \mathcal{B} can be made of one or two connected components. In particular, the second case happens if $\mathcal{C}_{\mathbf{a}}$ is closed.
4. *Curve sampling.* Since we have the parametric equation of $\mathcal{C}_{\mathbf{a}}$, it is possible to sample it in a preferred way, without any restriction. In our settings, the curve is sampled to obtain a number of samples equal to a percentage of the number of vertices of the boundary, varying such a percentage from 30% to 50%, as shown for example in Figure 3.6(d). These percentages are selected because the examples presented in this section are dense meshes obtained from laser scans; on the contrary, if dealing with very coarse meshes or models from sketches it is possible to over-fit the curve and even choose percentages higher than 100%. These samples will become the endpoints of the new curve edges.
5. *Generation of a valid mesh.* To fill the hole boundary, an approach similar in spirit to the surface tiling from contours [15] is adopted, sewing, alternatively, points on the curve $\mathcal{C}_{\mathbf{a}}$ and the mesh boundary \mathcal{B} . Starting from the first sample \mathbf{q} on $\mathcal{C}_{\mathbf{a}}$, for each connected component of the boundary \mathcal{B} , the vertex on \mathcal{B} the closest to \mathbf{q} is selected and all the vertices of that component are ordered according to the model normal and counterclockwise with respect to the curve $\mathcal{C}_{\mathbf{a}}$. To simplify the notation, we indicate with $\mathbf{q} + 1$ the vertex following \mathbf{q} with respect to the considered sorting. Starting from \mathbf{q} and from the first point of the ordered boundary vertices \mathbf{r} , a new triangle $t = (\mathbf{q}, \mathbf{r}, \mathbf{w})$ is inserted by choosing the new vertex \mathbf{w} between the vertices $\mathbf{q} + 1$ and $\mathbf{r} + 1$, selecting them on the basis of an edge length criterion. In particular, the new vertex is chosen by computing the lengths of the edges $(\mathbf{q}, \mathbf{r} + 1)$ and $(\mathbf{r}, \mathbf{q} + 1)$ and the vertex belonging to the smaller edge is selected (Figure 3.6(e)). Furthermore, if the edge $(\mathbf{r}, \mathbf{r} + 2)$ is not an edge of \mathcal{M}' , the length of the edges $(\mathbf{q}, \mathbf{r} + 1)$ and $(\mathbf{r}, \mathbf{q} + 1)$ are compared also with the length of $(\mathbf{r}, \mathbf{r} + 2)$ to possibly insert also $t = (\mathbf{r}, \mathbf{r} + 1, \mathbf{r} + 2)$ as a new triangle of the final mesh (in this case we assume there is a kind of loop on \mathcal{B}), as shown in Figure 3.6(f,g). In case the length of the new edge exceeds a prescribed threshold, the new vertex is chosen by computing the width of the angles and choosing the largest minimum angle. The process is iterated till the last sample of $\mathcal{C}_{\mathbf{a}}$ is reached.

Once the last sample on $\mathcal{C}_{\mathbf{a}}$ is reached, if \mathcal{B} has a single connected component, the vertex of \mathcal{B} the closest to this point is found and the previous steps are repeated by scrolling the indices of the curve in a decreasing way, obtaining a triangle mesh with curve elements such as in the example in Figure 3.6(h). Otherwise, the sewing operation is repeated for the second connected component of \mathcal{B} , as shown in Figure 3.7(c)(right). The holes generated by edge loops are filled during the visit of \mathcal{B} by adopting a standard hole-filling approach, such as the one implemented in [11].

Since the explicit formulation of \mathcal{C}_a is known, the edges between two curve samples can be represented directly as curve segments. Then, a new representation of the initial mesh that contains both curvilinear and linear elements is proposed. The curve triangulation is stored using the standard .off format for the linear elements and a parametric representation of the curve elements is added. Specifically, for each curve segment the coordinates of the starting point, the curve parametrization and the range of the curve parameter t are stored. Apart from this coding, to visualise the results, the curve triangles as polygonal elements are plotted in MATLAB, as shown in Figure 3.6(i) and in the third column of Figure 3.7.

3.6.2 Curve insertion examples

Figure 3.7 presents some examples of the presented curved triangles re-meshing algorithm. The central column represents the original triangulation while in the right column we highlight the new triangles and the new curve elements. In Figure 3.7(c) the outcome of the insertion of two circles is presented (in this case the new elements are depicted in red and blue). Further examples of the outcome of this method are available in Figure 3.5; in particular, Figure 3.5(d) presents the insertion into the model of two curves and the curve tessellation obtained.

As shown in Figure 3.7(b), the HT permits the recognition of curves even partially corrupted. In the example in Figure 3.8, the same curve approximates a profile before and after the missing part. Figure 3.8(b) show three curves recognized on the model. Figure 3.8(c) shows the repairing effect of the curve mesh tessellation obtained with our method when the three curves recognised around the abraded part are used to conjoint the profiles. As depicted in Figure 3.8(c) the new model gives a pleasant continuation of the curvilinear borders of the decoration.

Finally, in Figure 3.9 an example of curve insertion for a decorative pattern corresponding to a set of floral petals is shown. In this case, the curves represent the border of a coloured pattern; the colour information is stored in the mesh vertices as RGB values. In this case, the curve recognition is done by grouping the vertices according to their colorimetric information, see for instance the six petals in the floral band decoration in Figure 3.9(b). Then, the decorative elements are inserted as curves in the mesh. As shown in Figure 3.9 the overall model geometry does not change but the local connectivity is modified to follow the decorative elements. The views in Figures 3.9(c,d) show the mesh tessellation before and after the curve insertion. As visible in the detail, the petals are quite deteriorated and irregular because hand made, in particular, since one of the broken petals reaches the fragment fracture, the curve element is recognised and inserted as an open curve.

Table 3.5 compares the mesh properties of the models considered so far, before and after our curve insertion. The number of vertices in the second column of Table 3.5 corresponds to the difference of vertices of the final mesh in the area affected by the feature curves insertion; the average edge length and the width of the minimum angle are evaluated on the stripes of triangles removed and the new patches inserted. All the models considered in the examples

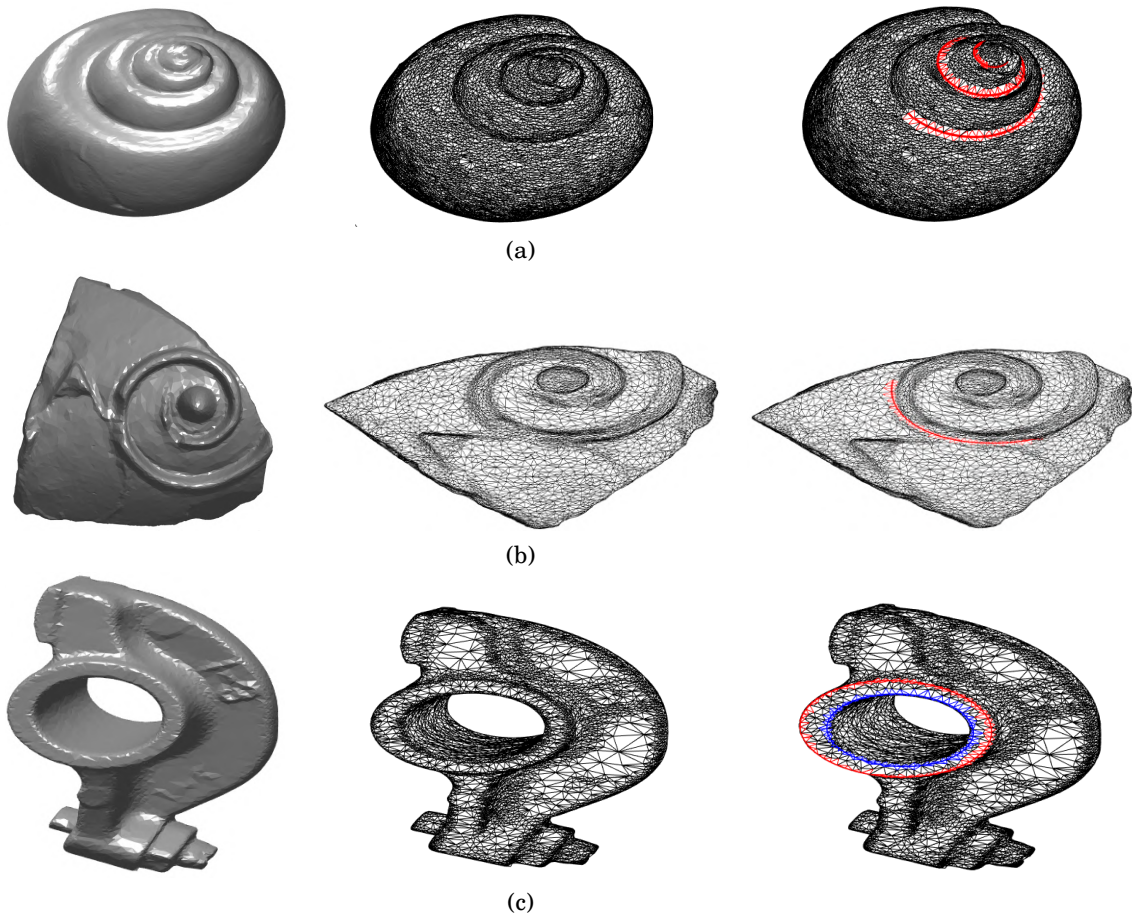


Figure 3.7: Some examples of feature curve insertion. The first two columns show some 3D models and their original triangulation. The last column highlights the new triangles and the curvilinear elements.

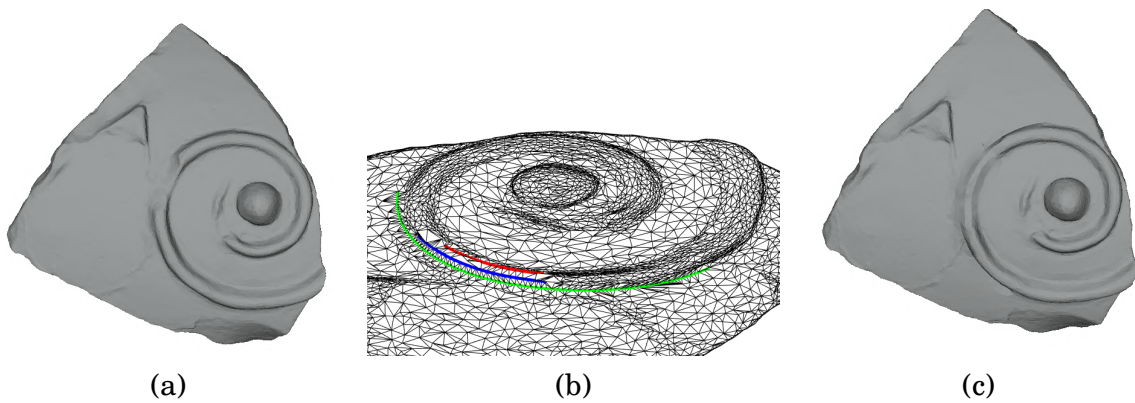


Figure 3.8: Visual comparison between the initial mesh (a) and the one obtained after the feature curves insertion (c) along the border of the decoration. The three inserted curves are highlighted in different colours on the mesh (b).

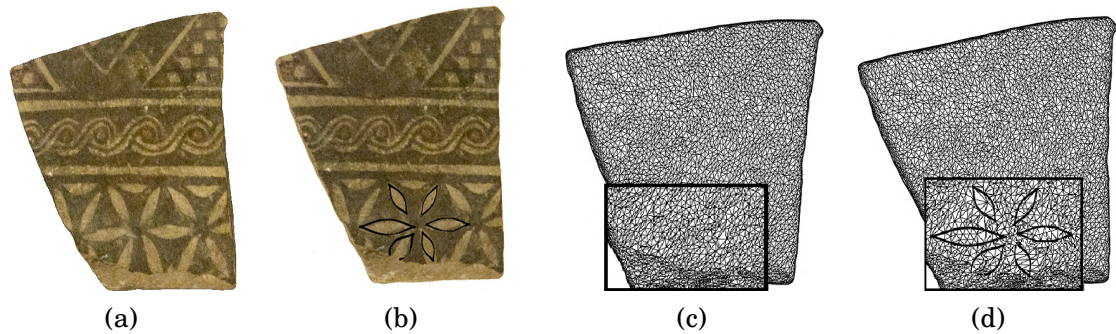


Figure 3.9: (a) A model with a colour decoration of a flower and (b) its petals recognised on it. (c) The original mesh tessellation does not reflect any decorative element. (d) The mesh after the insertion of the feature curves, with a detail around the curves inserted.

contain about 10K vertices with the sole exception of the model in Figure 3.5(a) which contains about 2.5K vertices. The number of vertices removed is proportional to the extension of the curve on the model. The absolute value of the edge length varies from model to model because they have different units of measure and normalisation. Anyway, considering the percentage of the increment of the edge length before and after the curve insertion, it is possible to notice that the edge length always grows more than 15%, also in the case the curve does not correspond to a geometric feature but to a colorful decoration. Similarly, the minimum angle in the triangle strip increases of more than 40%.

Table 3.5: Statistics on the mesh properties before and after a curve insertion. The number of vertices refers to the decrease in the mesh vertices after inserting a feature curve. The average edge length and the minimum angle width are reported only for the region modified by our algorithm. The models have different sizes: the models in Figures 3.5(a) and 3.7(c) are normalised in a unit sphere (radius 1); the unit measures are millimetres, metres and centimetres for the models in Figures 3.7(a), 3.7(b) and 3.9(a), respectively.

Model	# vertices removed	initial average edge length	final average edge length	initial minimum angle	final minimum angle
Figure 3.5(a)	45	0.0367	0.0420	0.0785	0.3120
Figure 3.7(a)	598	0.0112	0.0218	0.0426	0.0616
Figure 3.7(b)	71	$9.8 * 10^{-4}$	0.0014	0.0335	0.2059
Figure 3.7(c)	379	0.0090	0.0140	0.0626	0.1100
Figure 3.9(a)	89	0.1196	0.1423	0.0062	0.0200

The result of our method is an improvement of the original model; for example, Figure 3.8(c) shows the repairing effect of a model of a fragment, while Figure 3.7(c) provides the restoration of sharp edges that had probably been lost after a series of remesh operations. Also, Figure 3.6(c,e) shows that the strips of thin triangles have been replaced by two sharp edges well-defined.

3.7 Concluding remarks

Some of the models of fragments were given by the GRAVITATE project European project [3]. The method and the results presented in this chapter have been obtained during the first and the second year of my PhD. The division of the dictionary of space curves into type I and type II curves are exhibited in [138], while the description of two approaches in parametric and implicit form is provided in [136]. Finally, the method and the results of the space curves insertion on the surface of a 3D model are presented in [134] and it was presented at the Shape Modeling International conference (SMI 2020). Some of these results have been discussed with some experts at the Doctoral Consortium 2021, which took place at the Eurographics'2021 conference.

The proposed method extends the approach in Chapter 2: indeed, it detects families of space curves without necessarily projecting them on a plane and it also improves the approximation using a surface primitive as a projecting surface, such as a paraboloid. Figure 3.10 compares the HT-based recognition of the feature curves extracted from the 3D models provided on the first row, considering the projection on a paraboloid (in (a) and (c)) or a plane (in (b) and in (d)). The use of a paraboloid provides a better fitting of the set of feature points when the surface is curved, in particular in the star extremities and the inner part of the eye. This result is also demonstrated by the value of the MFE.

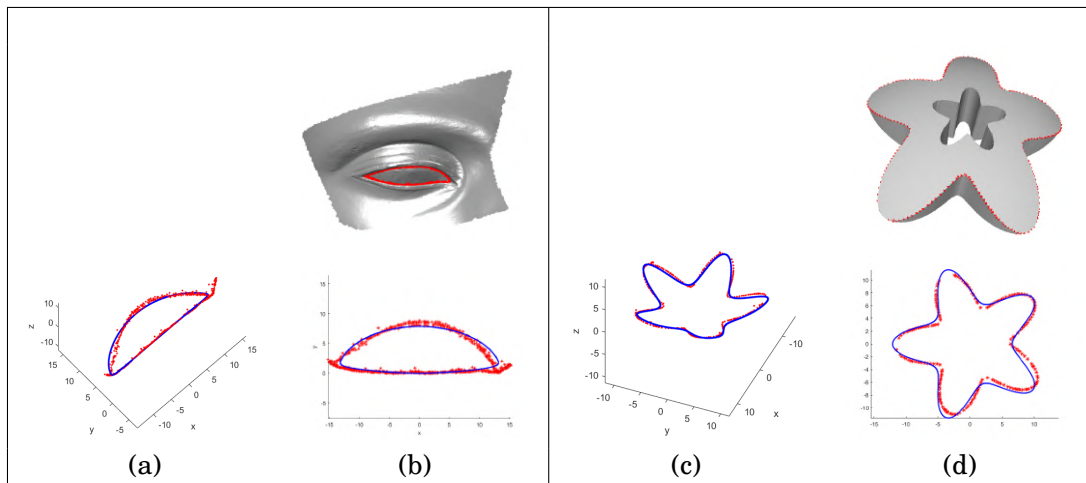


Figure 3.10: Approximation of curves projected, respectively, on a paraboloid and on a plane: in (a)-(b) using a geometric petal curve; in (c)-(d) using an m -convexities curve (see [150]).

In conclusion, the proposed method has several advantages. First, it permits the use of both the parametric and the implicit form in a single flow. Furthermore, the use of the HT naturally leads to a curve recognition pipeline robust to noise, outliers and missing data. Thanks to these characteristics, it is particularly suitable for the analysis of digital models deriving from 3D scans. In addition to archaeological fragments, the method has been tested on digital models used in real application contexts, such as CAD objects and objects scanned for reverse engineering.

Moreover, the recognised curves can become the key driving the model simplification and inserting curve elements directly into the model. Note that, in this application, the parametric representation adds large flexibility in the way the curves are sampled and in their possible subdivision into smaller segments. Unlike traditional techniques that insert a feature curve in the model as a piecewise-linear curve, to keep low the number of model elements, we propose a new representation that contains both curve and linear elements. In particular, the curve elements we consider can also be non-algebraic curves and transcendental or exponential curve segments: in this way, we overcome the limitation of using only Bézier segments that are not able to exactly represent circles or ellipses. Different from spline-based methods that obtain a better local approximation, our approach provides a global representation of the recognised curves and is more suitable for recognition.

3.8 Related publications

- C. Romanengo, B. Falcidieno, S. Biasotti, *Hough transform based recognition of space curves*, Journal of Computational and Applied Mathematics (2022), vol. 64, pp. 284–297.
- C. Romanengo, S. Biasotti, B. Falcidieno, *Hough Transform for Detecting Space Curves in Digital 3D Models*, Journal of Mathematical Imaging and Vision (2022), vol. 415, pp. 114504.

PIECE-WISE CURVE APPROXIMATION USING THE HOUGH TRANSFORM

The traditional HT-based recognition algorithm requires the pre-knowledge of which family of curves to look for. To overcome this limitation, we were inspired by the work [40] that applies a method based on the HT for recognising curvilinear profiles in digital images considering polynomial pieces of degree between 3 and 6. In this chapter, we define a method for the piece-wise curve approximation of curvilinear profiles in point clouds, able to deal with both planar and spatial profiles. Specifically, we create a segmentation of the initial profile made of subsets of regular points and we provide a piece-wise curve approximation, composed of parametric polynomial curves whose degree is 3 or 4 (see Section 4.1). If the profile is planar, then only one projection of the feature points is required for the approximation. In case the profile is spatial since the segmentation is composed of sets of regular points, we can use the results provided in <https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node6.html>, for which a regular space curve can be seen as the intersection of two cylinders. Then, we consider two projections to approximate the profile (see Section 4.2). In this thesis, we consider profiles whose projection onto the planes does not collapse to a point, postponing these particular cases to a later study.

4.1 Pipeline of the piece-wise curve approximation method

Given a point cloud \mathcal{P} , we preprocess it in a way similar to the preprocessing strategy described in Section 2.3. The result is a set of clusters \mathcal{P}_i , $i = 1, \dots, n$ with spatial coordinates (x, y, z) (see for example Figure 4.4(a)). The following procedure is considered for each cluster \mathcal{P}_i .

1. *Projection.* First, principal components analysis (PCA) is applied to \mathcal{P}_i to find the best fitting

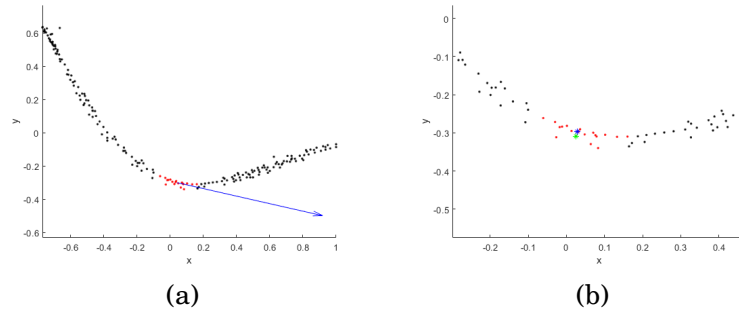


Figure 4.1: The smoothing step for a point \mathbf{p} of \mathcal{P}_i . In (a) the set $\mathcal{N}(\mathbf{p})$ is highlighted in red, while the first PCA component \mathbf{v} is designed in blue. In (b) the point \mathbf{p} is in blue and its projection point is in green.

plane of the set of points \mathcal{P}_i . The first two eigenvectors are used to define the projection plane to reduce the problem in the 2-dimensional space. \mathcal{P}_i is rotated and translated, to make the projection plane coincident with the plane $z = 0$. In case the variation in the z -coordinates of the points is less than a certain threshold, that is a percentage of the lengths of the axis-aligned bounding box, the profile is considered planar, so we consider only this projection (see Section 4.2.2). Otherwise, the profile is considered spatial, and then the second projection given by the components of the PCA will be considered (see Section 4.2.3).

2. *Normalisation.* On the plane identified in the previous step, the profile is represented using two Cartesian coordinates, for simplicity we denote as x and y these coordinates and as \mathcal{P}'_i the set of planar points. Then, \mathcal{P}'_i is centred in the origin of the Cartesian axes and normalised, so that all points lie within the unit circle (see for example Figure 4.4(b)). Our method benefits from normalisation because it restricts the range of variation for most parameters.
3. *Smoothing the feature points.* Since the set of points \mathcal{P}'_i may be affected by noise, we adopt a local smoothing filter similar to [41]. Specifically, given a point $\mathbf{p} \in \mathcal{P}'_i$, we select a neighborhood of \mathbf{p} identified by $\mathcal{N}(\mathbf{p}) = \{\mathbf{p}' \in \mathcal{P}'_i \mid \|\mathbf{p}' - \mathbf{p}\|_2 < \epsilon\}$, where ϵ is fixed starting from the mean distance among points of \mathcal{P}'_i . Then, the cross-correlation coefficient ρ of $\mathcal{N}(\mathbf{p})_x$ and $\mathcal{N}(\mathbf{p})_y$ is computed to evaluate how $\mathcal{N}(\mathbf{p})$ is distributed along a line. High absolute values of ρ indicate that there is a linear relationship between $\mathcal{N}(\mathbf{p})_x$ and $\mathcal{N}(\mathbf{p})_y$, therefore we fix 0.7 as a good threshold for the absolute value of ρ (as suggested in [41]). If ρ is under this threshold, we iteratively add points closest to \mathbf{p} to $\mathcal{N}(\mathbf{p})$. Once a good ρ value is reached, the point \mathbf{p} is projected into the line passing through the mean point $\bar{\mathbf{p}}$ of $\mathcal{N}(\mathbf{p})$ and aligned with a vector \mathbf{v} , the first PCA component for $\mathcal{N}(\mathbf{p})$. An example of this step is shown in Figure 4.1 and the final result is provided in Figure 4.4(c).

4. *Classification of the feature points.* To identify intersection points, we use an algorithm strategy that, given the point cloud \mathcal{P}'_i , returns a function $branch_number : \mathcal{P}'_i \rightarrow \mathbb{N}$ assigning to each $\mathbf{p} \in \mathcal{P}'_i$ the number $branch_number(\mathbf{p})$ of “branches” originating in \mathbf{p} . The points assuming values strictly greater than 2 will be identified as intersection points of different curves.

The function $branch_number$ is computed as follows. Given a threshold value $\delta > 0$, we define the graph $G = (\mathcal{P}'_i, E)$ as the undirected graph whose:

- nodes coincide with the points of \mathcal{P}'_i ;
- edges consist of the pairs of points in \mathcal{P}'_i having a Euclidean distance lower than δ .

Given a node \mathbf{p} of G and two threshold values δ_1, δ_2 with $0 < \delta_1 < \delta_2$, we define the subgraph $G_{\mathbf{p}} = (\mathcal{P}_{\mathbf{p}}, E_{\mathbf{p}})$ of G whose:

- set of nodes $\mathcal{P}_{\mathbf{p}}$ consists of the nodes $\mathbf{q} \in \mathcal{P}'_i$ such that $\delta_1 < \|\mathbf{p} - \mathbf{q}\|_2 < \delta_2$;
- set of edges $E_{\mathbf{p}}$ consists of the edges $(\mathbf{q}, \mathbf{r}) \in E$ such that both $\mathbf{q}, \mathbf{r} \in \mathcal{P}_{\mathbf{p}}$.

The value $branch_number(\mathbf{p})$ is defined as the number of (not too small) connected components of $G_{\mathbf{p}}$.

The approach clearly depends on the choice of the parameters $\delta, \delta_1, \delta_2$. Preliminary evaluations show that there exists a suitable choice of these parameters providing (even in the presence of noise) a quite limited (w.r.t. the actual number) number of points \mathbf{p} assuming $branch_number(\mathbf{p})$ value strictly greater than 2. The points \mathbf{p} with $branch_number(\mathbf{p}) = 2$ are classified as regular points. End-points, if they exist, have $branch_number(\mathbf{p}) = 1$. An example of the application of this procedure is provided in Figure 4.2. Finally, we use this point classification to sort the points of \mathcal{P}'_i . Starting from a point such that $branch_number(\mathbf{p}) > 2$, we visit the graph following a path made of points with $branch_number(\mathbf{p}) = 2$ until we reach another point with $branch_number(\mathbf{p}) > 2$ or the same point (loop). As we visit the graph, the paths found are discarded from further visits and are seen as sets $\mathcal{P}'_{i_j}, j = 1, \dots, J$, of ordered points (the order is induced by the visit priority).

5. *Segmentation.* We use a variation of the Douglas-Peucker algorithm [52] to segment each ordered set of points \mathcal{P}'_{i_j} detected in \mathcal{P}'_i . The variation allows us to deal also with a closed curve as shown in Figure 4.3(b). We first select two points \mathbf{p}_1 and \mathbf{p}_2 of \mathcal{P}'_{i_j} . In case the curve is open, we choose the first and the last point of the sorted \mathcal{P}'_{i_j} , while if the curve is closed, we select the first and the middle point of the sorting. Then, the error induced over the line that connects \mathbf{p}_1 and \mathbf{p}_2 , denoted by $\overline{\mathbf{p}_1\mathbf{p}_2}$, that approximates the portion $(\mathbf{p}_1, \mathbf{p}_2)$ is computed as $\varepsilon = \max_{\mathbf{p} \in (\mathbf{p}_1, \mathbf{p}_2)} d(\overline{\mathbf{p}_1\mathbf{p}_2}, \mathbf{p})$, where d is the point-line distance. The farthest point of \mathcal{P}'_{i_j} from the line $\overline{\mathbf{p}_1\mathbf{p}_2}$ is denoted as \mathbf{p}_3 and the set $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ is a first sampling

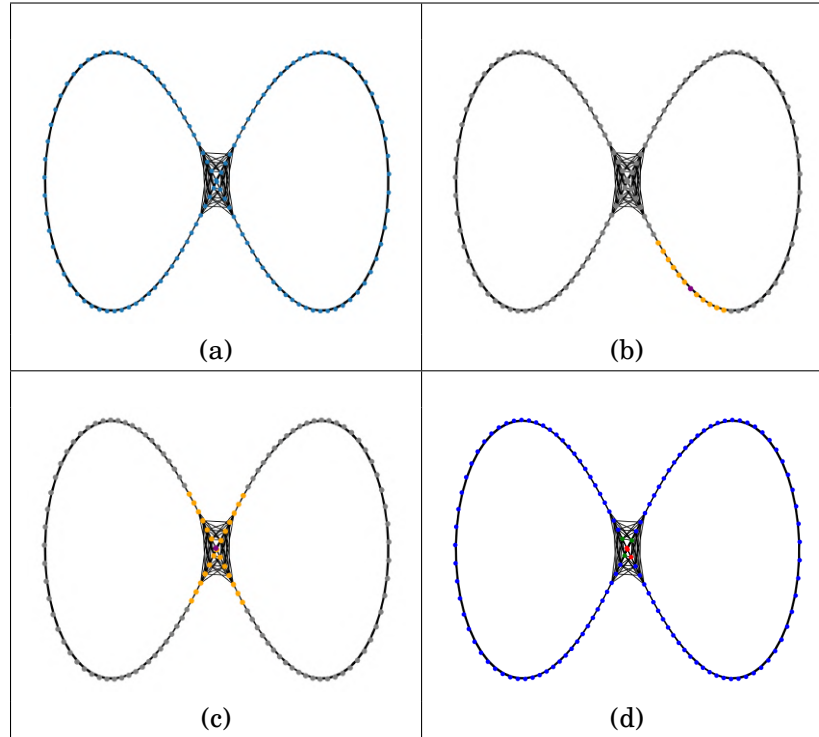


Figure 4.2: Example of a search for a point of self-intersection. In (a) the graph G associated to the input point cloud; in (b,c) a node \mathbf{p} in violet and the vertices of the subgraph $G_{\mathbf{p}}$ in orange; in (d) the points with $\text{branch_number}(\mathbf{p}) = 4$ in red, the points with $\text{branch_number}(\mathbf{p}) = 3$ in green and the points with $\text{branch_number}(\mathbf{p}) = 2$ in blue.

of \mathcal{P}'_{i_j} . The same procedure is iterated to the sub-portions $(\mathbf{p}_1, \mathbf{p}_3)$ and $(\mathbf{p}_2, \mathbf{p}_3)$ until the approximation error ε falls below a certain fixed threshold, which in our context is set as a percentage of the length of \mathcal{P}'_{i_j} (see Figure 4.3). In case the curve is closed, the iterations are applied also to the portion $(\mathbf{p}_2, \mathbf{p}_1)$ and iterated to its sub-portions. Applying this method to each ordered set \mathcal{P}'_{i_j} , the result is a sampling $\{\mathbf{p}_1, \dots, \mathbf{p}_K\}$ of \mathcal{P}'_i and consequently $K - 1$ subsets $\mathcal{S}_k = (\mathbf{p}_k, \mathbf{p}_{k+1})$ of \mathcal{P}'_i if the profile is open, such that $\mathcal{P}'_i = \bigcup_{k=1}^{K-1} \mathcal{S}_k$, or K subsets \mathcal{S}_k if the curve is closed, such that $\mathcal{P}'_i = \bigcup_{k=1}^K \mathcal{S}_k$. An example of this result is shown in Figure 4.4(d). Note that each subset \mathcal{S}_k is a set of points that outlines a regular plane curve.

6. *HT-based recognition method.* The HT-based recognition method is applied to each subset \mathcal{S}_k , using specific families of curves. The result is a set of pieces of curve \mathcal{C}'_k , $k = 1, \dots, L$, such that $\bigcup_{k=1}^L \mathcal{C}'_k$, $L = K - 1$ (open profile) or $L = K$ (closed profile), correctly approximates \mathcal{P}'_i (see for example Figure 4.4(e)).

In case the profile considered is spatial, the second projection given by the PCA is applied and all steps of the pipeline are repeated, except for the segmentation step, which is repeated on the second plane only if necessary, i.e. splitting the segments if they contain non-regular points in the second plane. Then, they are good candidates to be approximated through the two projections.

4.1. PIPELINE OF THE PIECE-WISE CURVE APPROXIMATION METHOD

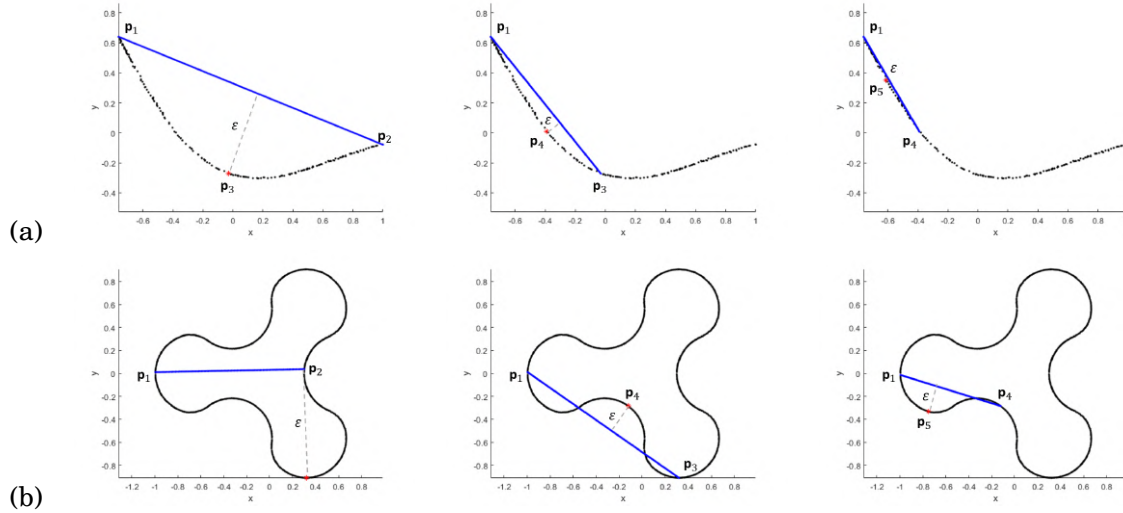


Figure 4.3: The first three steps of the Douglas-Peucker algorithm considering an open, in (a), and a closed, in (b), profile.

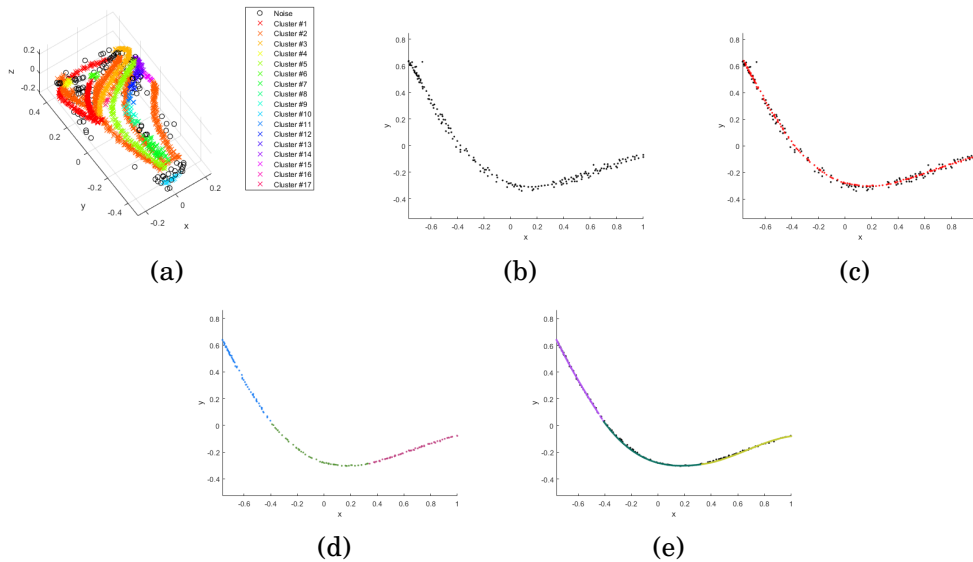


Figure 4.4: A visual illustration of the steps of our algorithm. In (a) the result of the clustering method: for the following steps we select Cluster #5. In (b) the projection on the (x, y) -plane and the normalization. In (c) the smoothing operation; in (d) the result of the segmentation; finally, in (e) the output of the algorithm.

4.2 Description of the feature approximation method

In this section, we elaborate on the sixth step of the algorithm described in the previous one. Specifically, in Section 4.2.1 we highlight the variations from the classical HT-based recognition algorithm, while in Section 4.2.2 and 4.2.3 we illustrate how the method works considering, respectively, the only (x, y) -projection and the combination of the (x, y) -projection and the second projection given by the components of the PCA. In the second case, we exploit the fact that a regular space curve can be seen as the intersection of two cylinders. Specifically, given a curvilinear spatial profile, we can consider two cylindrical projections and approximate the plane profile with a plane curve in order to create the cylinders, projecting the detected curves along the direction perpendicular to each projection plane. This fact is guaranteed by the condition of regularity: let us consider the implicit form of a space curve

$$I: \begin{cases} f(x, y, z) = 0 \\ g(x, y, z) = 0 \end{cases}.$$

If the functions f and g satisfy the regularity condition $f_y g_z - f_z g_y \neq 0$, then the system can be written as

$$I: \begin{cases} y = Y(x) \\ z = Z(x) \end{cases}$$

for the implicit-function theorem. Therefore, the explicit equation of a space curve can be expressed as a curve intersection of two cylinders projecting the curve onto the xy and xz planes. We refer to <https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node6.html> for details. Note that, for simplicity, we detail only one type of representation, making explicit x .

The fifth step of our pipeline ensures that the segments that we are going to approximate outline regular plane curves. However, we show that the regularity of the projected curve also guarantees the regularity of the original curve in the space. Indeed, let us consider a generic space curve $\gamma(t) = (f_1(t), f_2(t), f_3(t))$ and, for simplicity of notation, we suppose its projection onto the plane (x, y) given by $\gamma_{x,y}(t) = (f_1(t), f_2(t))$ is regular. Similar reasoning holds for the other plane. The regularity of $\gamma_{x,y}$ is equivalent to the condition $f_1'(t)^2 + f_2'(t)^2 \neq 0$ for all $t \in I$. Then, it is also true that

$$f_1'(t)^2 + f_2'(t)^2 + f_3'(t)^2 \neq 0 \text{ for all } t \in I$$

because $f_3'(t)^2 \geq 0$, which corresponds to the regularity condition of the space curve γ . Consequently, the fifth step of our pipeline guarantees that the spatial segments approximated through the two projections outline regular space curves.

4.2.1 Variation from the standard HT

The algebraic families of curves that we use for the HT-base recognition process are:

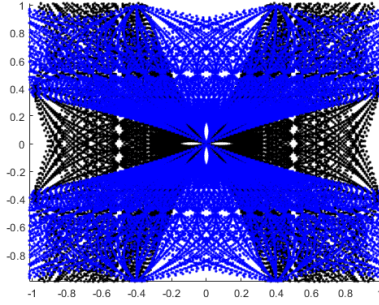


Figure 4.5: 216 curves of the families $\mathcal{F}_{3,x}$ in blue and 216 curves of the families $\mathcal{F}_{3,y}$ in black with parameters included in the range $[-1, 1]$.

$$\mathcal{F}_{g,x} := \{\mathcal{C}_{\mathbf{a}}^g : x = a_g y^g + a_{g-1} y^{g-1} + \dots + a_1 y + a_0\} \quad g = 3, 4$$

$$\mathcal{F}_{g,y} := \{\mathcal{C}_{\mathbf{b}}^g : y = b_g x^g + b_{g-1} x^{g-1} + \dots + b_1 x + b_0\} \quad g = 3, 4.$$

The choice of two families allows for a wider variety of possible curves (see Figure 4.5). Note that considering these two families of curves, the parameters for the HT can be 4 or 5, but in the steps of our algorithm some of these are fixed, imposing conditions of continuity and/or G^1 -continuity.

In the standard HT-based method each entry of the accumulation matrix increases by 1, each time the HT of a point intersects the corresponding cell, so all points have the same weight. In our method, we give each point a particular weight, which is a variation of the one proposed in [40]. Specifically, we give greater weight to the extreme points of the segment we want to approximate. Given \mathbf{p}_0 the starting point of the segment, the weight of a generic point \mathbf{p} is:

$$\omega_{\mathbf{p}_0}(\mathbf{p}) := \frac{1}{\exp(\|\|\mathbf{p}_0 - \mathbf{p}\|_2 - \overline{dist})^2)}$$

with $\|\cdot\|_2$ the L^2 norm and \overline{dist} the mean of the distances of all points of the segment from the starting point \mathbf{p}_0 .

In this case, we use the point-distance weighted HT and then, given a starting point \mathbf{p}_0 , each entry of the accumulation matrix increases by $\omega_{\mathbf{p}_0}(\mathbf{p})$, each time the HT of \mathbf{p} intersects the corresponding cell.

4.2.2 Approximation method considering one projection

To each subset \mathcal{S}_k , the HT-based recognition method is applied using the families of curves $\mathcal{F}_{g,x}$ and $\mathcal{F}_{g,y}$ introduced in Section 4.2.1. The one recognised with a lower value of MFE (between \mathcal{S}_k and a sample of the curve, as defined in Eq. (3.2)) is kept, and considered as the recognised curve for the segment \mathcal{S}_k . In Figure 4.6(a,b) the two families have produced two different results, but the one shown in Figure 4.4(a) has a lower value of MFE. Note that, if the minimum value of the

MFE is higher than a threshold, the Douglas Peucker algorithm is applied to refine the input segment.

We impose some conditions on the two families of curves to ensure that the outcome of the algorithm is a G^1 -continuous curve, eventually containing some singular or intersection points, in which the continuity is C^0 . These conditions change according to the position of the segments, that is initial, central or final.

If the segment considered is the initial one \mathcal{S}_1 , we only impose the passage to the initial point \mathbf{p}_1 , then the parameters a_0 and b_0 are fixed for the respective family:

$$a_0 = x_{\mathbf{p}_1} - a_g y_{\mathbf{p}_1}^g - \dots - a_1 y_{\mathbf{p}_1}, \quad b_0 = y_{\mathbf{p}_1} - b_g x_{\mathbf{p}_1}^g - \dots - b_1 x_{\mathbf{p}_1}.$$

We denote the resulting piece of curve as \mathcal{C}_1 . In case the set of points \mathcal{S}_k represent a central segment, we call \mathbf{p}_k the final point of the previous curve \mathcal{C}_{k-1} and the new starting point of this step. Then we compute \mathbf{u}_k the unit tangent vector of \mathcal{C}_{k-1} in \mathbf{p}_k to impose the G^1 -joined as follows:

$$\begin{aligned} a_0 &= x_{\mathbf{p}_k} - a_g y_{\mathbf{p}_k}^g - \dots - a_1 y_{\mathbf{p}_k}, \quad b_0 = y_{\mathbf{p}_k} - b_g x_{\mathbf{p}_k}^g - \dots - b_1 x_{\mathbf{p}_k}, \\ a_1 &= \frac{1}{\tan(\mathbf{u}_k)} - g a_g y_{\mathbf{p}_k}^{g-1} - \dots - 2a_2 y_{\mathbf{p}_k} - a_1, \\ b_1 &= \tan(\mathbf{u}_k) - g b_g x_{\mathbf{p}_k}^{g-1} - \dots - 2b_2 y_{\mathbf{p}_k} - b_1 \end{aligned}$$

where $\tan(\mathbf{u}_k)$ denotes the tangent of the angle between \mathbf{u}_k and the x -axis. We denote the resulting piece of the curve as \mathcal{C}_k . Finally, in case the segment is the final \mathcal{S}_L , the procedure changes based on whether the global curve is open ($L = K - 1$) or closed ($L = K$). If it is open, the imposed condition is the same as the central segment, while if the curve is closed, we consider the families $\mathcal{F}_{g,x}$ and $\mathcal{F}_{g,y}$ with $g = 4$ and we impose conditions of C^0 -continuity on both the initial and the final points. The only remaining free parameter to be recognised with HT is a_4 .

For each curve, the algorithm returns a vector containing its geometrical information: $[1 a b c d e]$ (with $a = b_4$, $b = b_3$, $c = b_2$, $d = b_1$ and $e = b_0$) if the curve belongs to the family $\mathcal{F}_{g,y}$ or $[2 a b c d e]$ (with $a = a_4$, $b = a_3$, $c = a_2$, $d = a_1$ and $e = a_0$) if the curve belongs to the family $\mathcal{F}_{g,x}$.

In case the profile is planar, to each curve \mathcal{C}'_k , translations and/or rotations are applied backwards in order to have a good approximation of the original cluster of points \mathcal{P}_i . Otherwise, the algorithm proceeds with the step described in Section 4.2.3.

4.2.3 Approximation method considering two projections

Upstream of the results obtained from the algorithm described in Section 4.2.2, if the feature points describe a space curve there are still steps to be taken. Starting from the roto-translated point cloud \mathcal{P}_i , it is possible to consider the second projection suggested by the PCA obtaining a new planar point cloud \mathcal{P}''_i (see for example Figure 4.7(a)). In order not to make the notation too

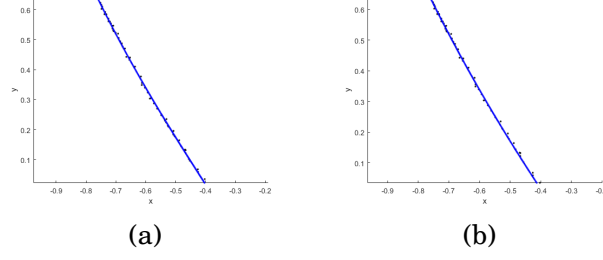


Figure 4.6: Curves recognised by the HT-based recognition algorithm for the first segment of \mathcal{P}_i of Figure 4.4 considering the families $\mathcal{F}_{g,y}$ (a) and $\mathcal{F}_{g,x}$ (b).

heavy, in this section we will consider the (x, z) -projection. At first, the smoothing step of the algorithm in Section 4.2.2 is applied to \mathcal{P}''_i in order to obtain a noiseless point cloud, as shown in Figure 4.7(b). Then, we consider the same segmentation $\mathcal{P}''_i = \bigcup_{k=1}^L \mathcal{S}_k$, with $L = K - 1$ or $L = K$, and we apply the HT-based recognition method to each segment \mathcal{S}_k using the following families of curves:

$$\mathcal{G}_{g,x} := \{\mathcal{C}_{\mathbf{c}}^g : x = c_g z^g + c_{g-1} z^{g-1} + \dots + c_1 z + c_0\} \quad g = 3, 4$$

$$\mathcal{G}_{g,z} := \{\mathcal{C}_{\mathbf{d}}^g : z = d_g x^g + d_{g-1} x^{g-1} + \dots + d_1 x + d_0\} \quad g = 3, 4.$$

An example of these types of curves is provided in Figure 4.7(c,d). The recognised curve with a low value of MFE is denoted as \mathcal{C}_k'' . If the minimum value of the MFE is higher than a threshold, the Douglas-Peucker algorithm is applied to the input segment to divide it into regular sub-segments. The result is a set of curves \mathcal{C}_k'' , $k = 1, \dots, L$, such that $\bigcup_{k=1}^L \mathcal{C}_k''$ ($L = K - 1$ or $L = K$) correctly approximate \mathcal{P}''_i . For each curve, the algorithm returns a vector containing its geometrical information: $[1 f h i l m]$ (with $f = d_4$, $h = d_3$, $i = d_2$, $l = d_1$ and $m = d_0$) if the curve belongs to the family $\mathcal{G}_{g,z}$ or $[2 f h i l m]$ (with $f = c_4$, $h = c_3$, $i = c_2$, $l = c_1$ and $m = c_0$) if the curve belongs to the family $\mathcal{G}_{g,x}$.

At this point, the curves \mathcal{C}_k' and \mathcal{C}_k'' , have to be combined to obtain a unique curve \mathcal{C}_k that represents the segment \mathcal{S}_k in the space. We consider three possibilities:

- $\mathcal{C}_k' \in \mathcal{F}_{g,y}$ and $\mathcal{C}_k'' \in \mathcal{G}_{g,z}$:

$$\mathcal{C}_k : \begin{cases} x = t \\ y = at^4 + bt^3 + ct^2 + dt + e \\ z = ft^4 + ht^3 + it^2 + lt + m \end{cases} \quad t \in [\min(\mathcal{S}_{k,x}), \max(\mathcal{S}_{k,x})]$$

- $\mathcal{C}_k' \in \mathcal{F}_{g,x}$ and $\mathcal{C}_k'' \in \mathcal{G}_{g,z}$:

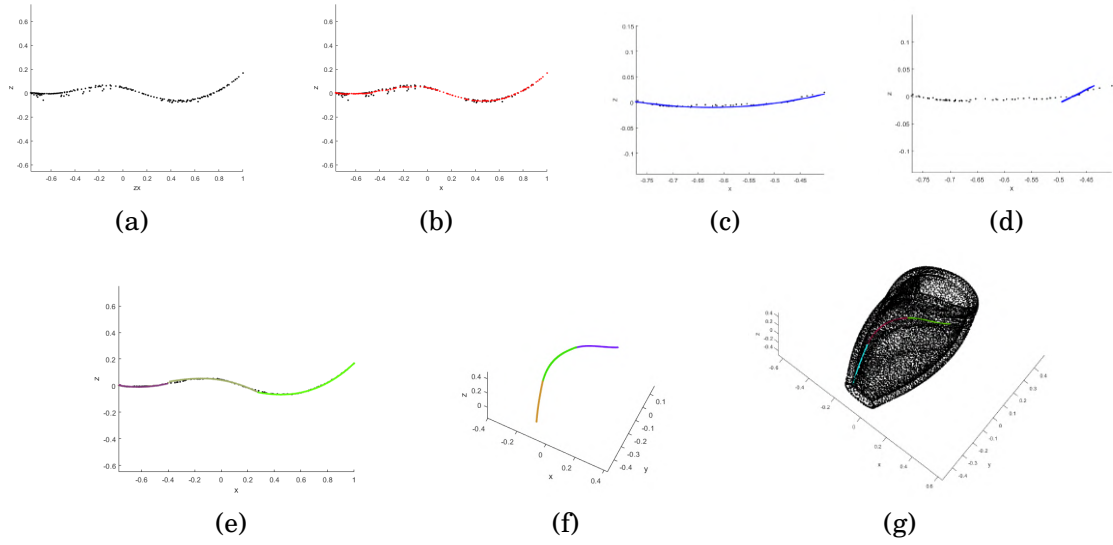


Figure 4.7: A visual illustration of the steps of our algorithm in case of space curve approximation. In (a) the projection on the (x, z) -plane and the normalization. In (b) the smoothing operation; in (c, d) the curves recognised by the HT-based recognition algorithm for the first segment considering the families $\mathcal{G}_{g,z}$ (c) and $\mathcal{G}_{g,x}$. In (e) the approximation of all segments. Finally, in (f) the output of the algorithm on the input point cloud \mathcal{P} .

$$\mathcal{C}_k : \begin{cases} x = at^4 + bt^3 + ct^2 + dt + e \\ y = t \\ z = f(at^4 + bt^3 + ct^2 + dt + e)^4 + \dots \\ \dots + h(at^4 + bt^3 + ct^2 + dt + e)^3 + \dots \\ \dots + i(at^4 + bt^3 + ct^2 + dt + e)^2 + \dots \\ \dots + l(at^4 + bt^3 + ct^2 + dt + e) + m \end{cases} \quad t \in [\min(\mathcal{S}_{k_y}), \max(\mathcal{S}_{k_y})]$$

- $\mathcal{C}'_k \in \mathcal{F}_{g,y}$ and $\mathcal{C}''_k \in \mathcal{G}_{g,x}$:

$$\mathcal{C}_k : \begin{cases} x = ft^4 + ht^3 + it^2 + lt + m \\ y = a(ft^4 + ht^3 + it^2 + lt + m)^4 + \dots \\ \dots + b(ft^4 + ht^3 + it^2 + lt + m)^3 + \dots \\ \dots + c(aft^4 + ht^3 + it^2 + lt + m)^2 + \dots \\ \dots + d(ft^4 + ht^3 + it^2 + lt + m) + e \\ z = t \end{cases} \quad t \in [\min(\mathcal{S}_{k_z}), \max(\mathcal{S}_{k_z})]$$

An example of $\bigcup_{k=1}^L \mathcal{C}_k$ is shown in Figure 4.7(f). To each curve \mathcal{C}_k , translations and/or rotations are applied backward to have a good approximation of the original cluster \mathcal{P}_i (see Figure 4.7(g)).

4.3 Examples

Figure 4.8 provides some results of our piece-wise curve approximation method applied to point clouds extracted from the models shown in the first column. In the central columns, we exhibit the two projections, except for the model in the first row, which needs only one projection since the profile is planar. The final column shows the resulting curve on the initial model. Part of the proposed models come from the Visionair 3D Shape Repository [1] and the ABC dataset [91].

4.4 Feature-preserving point cloud simplification and resampling

The outcome of the piece-wise approximation method can be used by a point cloud simplification algorithm to drive a constrained point cloud simplification and resampling. The simplification process that we consider aims at drastically reducing the number of points representing the original point cloud while preserving the identified curves. Given a point cloud \mathcal{P} , the previous steps of the proposed process allow us to identify the feature curves of \mathcal{P} . Specifically, for each identified curve \mathcal{C}^i the previous steps provide us:

- a decomposition of the curve \mathcal{C}^i in sub-curves $\mathcal{C}_1^i, \mathcal{C}_2^i, \dots, \mathcal{C}_L^i$, each of which is described in terms of its parametric equation, starting and ending points;
- a collection of points $\mathcal{P}_i \subseteq \mathcal{P}$ representing the points of \mathcal{P} identified as points of the curve \mathcal{C}^i . Each point $\mathbf{p} \in \mathcal{P}_i$ is properly labelled according to the specific sub-curve \mathcal{C}^i it belongs to.

The considered curve-preserving simplification process is a decimation algorithm considering at any new iteration a pair \mathbf{u}, \mathbf{v} of points of \mathcal{P} and properly replacing them with a new point \mathbf{w} . The decimation process is driven by a density function to first remove vertices belonging to regions of \mathcal{P} with high density and takes advantage of a graph representation of the point cloud \mathcal{P} . Each removal aims to preserve the identified curve \mathcal{C}^i .

4.5 Concluding remarks

The experimental examples provided in this chapter come from the Visionair 3D Shape Repository [1] and the ABC dataset [91]. The presented method was developed at the end of the third year of my PhD, to overcome the limitation of the traditional HT for curves that requires the pre-knowledge of the family to be used for the recognition. Indeed, it permits to approximate feature points extracted from a point cloud with pieces of plane and space curves given by specific polynomials. Except for the correspondence of the specified singular points, where the order of continuity is C^0 , the output curve is composed of polynomial pieces with the order of continuity G^1


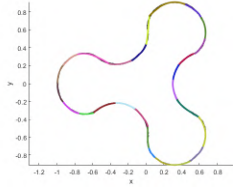


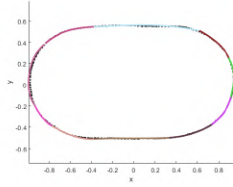
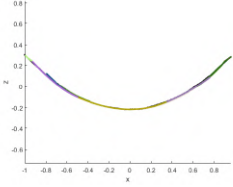
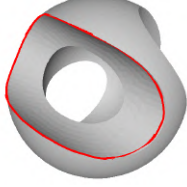

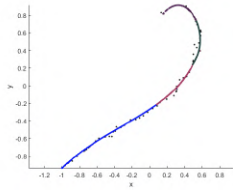
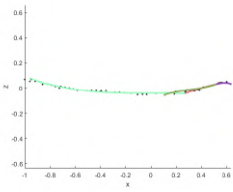


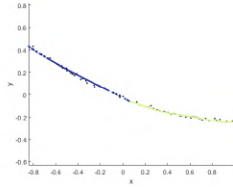
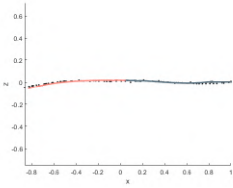


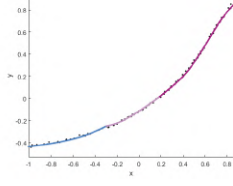
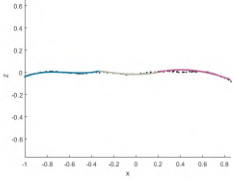

initial model	first projection	second projection	final curve on the model
			
			
			
			
			

Figure 4.8: Some results of our piece-wise curve approximation method. In the first column the original model from which the point cloud is extracted. In the central columns the resulting approximation of two projections. In the final column, the outcome curves are highlighted on the model.

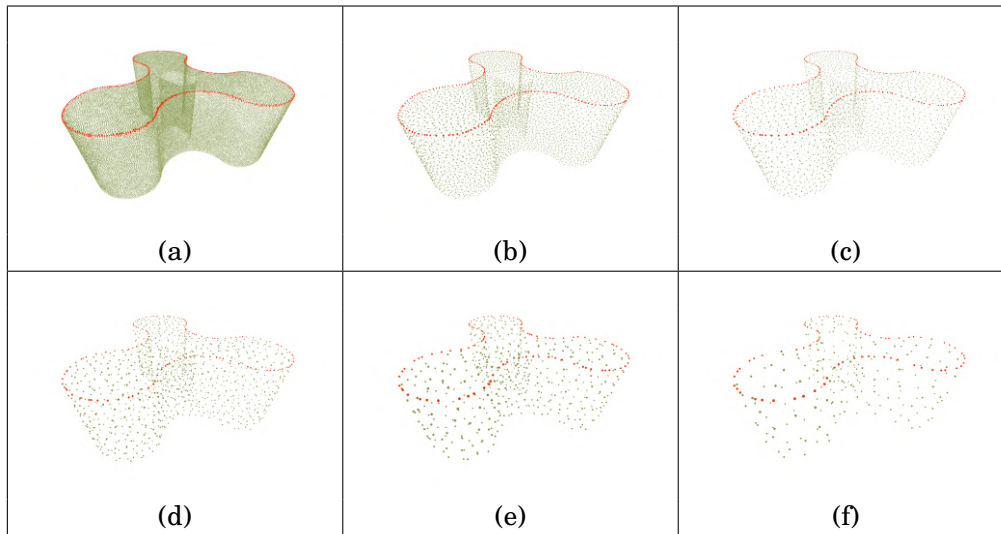


Figure 4.9: An example of a decimation process performed on a point cloud. In (a), the input point cloud with the feature points is highlighted in red. In (b) - (f), the output is obtained by reducing the number of points by a factor of 10, 20, 50, 100, and 200, respectively.

in each projection. We have yet to investigate the problem of the type and the order of continuity from the spatial point of view of the resulting curve approximation. The resulting curves can be input to some point cloud simplification algorithm to drive a constrained point cloud simplification and resampling.

Currently, the presented method does not include the case of lines orthogonal to each other and parallel to the axes of the projection plane, such as the edges of a cube parallel to the Cartesian axes. A relevant part of our ongoing research activity is devoted to extending our approach by including these particular cases, as well as more complex examples that also contain self-intersection points.

RECOGNITION, EXTRACTION AND REPRESENTATION OF GEOMETRIC PRIMITIVES

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 it no longer corresponds to the real geometry of the manufactured object itself.

A strategy to retrieve an object's 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, aims at generating 3D mathematical surfaces and geometric features representing the geometry of real parts. Many methods address this problem and we refer to [88], a survey that groups a large part of the approaches presented so far.

Our goal is to define model fitting algorithms that, given a set of points, find the most likely model primitives that generated those points. In this chapter, we first address the problem of recognition and fitting simple geometric primitives from segmented point clouds (Section 5.3.1). Then, we face the problem of recognition and fitting primitives from the entire point cloud representing a CAD object (Section 5.4.1).

The methods proposed in this chapter have been compared with other methods in [139] and [140]. The first mentioned paper is the result of an international contest: the SHREC (Shape Retrieval Challenge), using appropriately built benchmarks. The details of this activity are described in Appendix A.

5.1 Previous work

Representing an object with a set of geometric components is a long-standing problem in computer vision, computer graphics and CAD. As we are interested in recognising (simple or complex) geometric primitives in the manufacturing domain, here we focus our attention on those approaches that share the same goal.

A considerable variety of algorithms have been devised to decompose digitalised point clouds or meshes representing CAD objects into regions approximated by primitives belonging to some given sets. According to [88], these approaches can be grouped into four families: stochastic, parameter space, clustering and learning techniques. The first group includes the RANSAC method [144] and its optimizations. The second family includes Hough-like voting methods and parameter space clustering, e.g., [102]. The third class gathers all the other clustering techniques, and can be classified into three main types: primitive-driven region growing, e.g., [13]; automatic clustering and Lloyd-based algorithms, e.g., [159]; primitive-oblivious segmentation, e.g., [96]. Finally, with the growing popularity of deep learning techniques, supervised fitting methods have been proposed even for multi-class primitives [99, 155]. We refer to [88] for a comprehensive historical taxonomy of methods for simple primitive detection, which is beyond the scope of this thesis.

Despite a large number of available solutions, the problem is far from being solved; novel approaches have been proposed in the very last few years to address the shortcomings of existing paradigms or to propose novel directions to move in. An example of a recent approach dealing with the recognition of geometric primitives is [125]. It consists of a curvature-based partitioning method that decomposes an input triangle mesh into maximal surface portions; the decomposition is performed so that each segment corresponds to one of the following seven invariance classes of surfaces: plane, sphere, cylinder, surface of revolution, prismatic, helix, and complex surface. The method was adapted to handle point clouds in [140]. In [108], a method for the identification and fitting of planes and cylinders from unstructured point clouds in manufacturing is presented. It consists of three subsequent phases: point cloud segmentation; merging of over-segmented regions and estimation of surface parameters; extraction of cylinders and planes. Being the method able to handle only two primitive types, its applicability is however restricted. To handle the increasing availability of acquired data, [124] introduces a region-growing based system for the segmentation of large point clouds in planar regions. Other approaches, devised to detect only specific types of primitives, are: [26], which deals with quadric surfaces; [104], which fits surfaces of revolution; and [21], which extracts cylindrical shapes from non-oriented point clouds.

In the field of deep learning, two methods have shown promising results in the problems of segmentation and recognition: ParSeNet [148] and HPNet [160]. Beyond their performance, outstanding merit is their capability of handling – together with the more classical simple geometric primitives – open and closed spline surfaces. Another learning-based method lately proposed for fitting primitives is PriFit [147], which learns to decompose a point cloud of various

3D shape categories into a set of geometric primitives, such as ellipsoids and cuboids, or deformed planes. However, the natural flexibility of supervised learning approaches in identifying geometric primitives comes with a considerable cost: the need of having gigantic labelled training sets, whose difficulty of gathering limits their current application.

As for the methods that use the Hough transform to identify geometric primitives, they were limited, until recently, to planes [28, 102], combinations of linear subspaces [61], spheres [32] and circular cylinders [126]; in the case of cylinders, however, the rotational axis needs to be known before the application of the HT. The recent advances in the use of the algebraic functions [20] are paving the road to a larger use of the HT for recognising more complex families of geometric primitives. To the best of our knowledge, the sole approach that exploit this idea were proposed in [19], for the recognition of an ellipsoid in a free-form model.

5.2 Geometric primitives

In this section, we present the geometric primitives we use in the recognition process. Specifically, in Section 5.2.1 we first show a dictionary of simple geometric primitives, while in Section 5.2.2 we provide a set of complex geometric primitives. For each primitive type, we specify the constraints used to obtain the corresponding standard forms. The expression *standard form* – sometimes referred to as *canonical form* – refers to a standard way of presenting some sets of mathematical objects. A standard form is usually expected to be simpler than the elements it is equivalent to. For instance, such (simplified) expressions usually require less memory and possess nice features, which make cleaner and more precise an algorithm design [36]. In analytic geometry, standard forms can help study curves and surfaces. Conics and quadrics can be classified by their orbits under roto-translation. Their general implicit equation can be greatly simplified by a suitable rotation – which eliminates mixed terms – and a suitable translation – which removes one or more monomials of degree 1; from a purely geometric perspective, vertices and centres are translated to the origin of the coordinate system, while axes are aligned to the coordinate axes. When dealing with parametric representations, standard forms allow to set of some of the parameters. The number of parameters is reduced, in both implicit and parametric cases. A similar argument can be applied to tori and, more in general, to surfaces of revolution.

5.2.1 Simple geometric primitives

This section introduces the parametric equations used for planes, cylinders, cones, spheres, and tori. For each primitive type, we specify the constraints used to obtain the corresponding standard forms, except for the planes that are represented with the Hesse normal form. Figure 5.1 shows an example of the simple geometric primitives considered with their attributes that we call geometric descriptors.

Planes. We consider the Hesse normal form

$$(5.1) \quad x \cos \theta \sin \phi + y \sin \theta \sin \phi + z \cos \phi - \rho = 0,$$

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.

Spheres. The points on the sphere of radius $r > 0$ and center $\mathbf{c} \in \mathbb{R}^3$ can be parametrised 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.

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 parametrised 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 $\mathbf{a} := \mathbf{e}_3$.

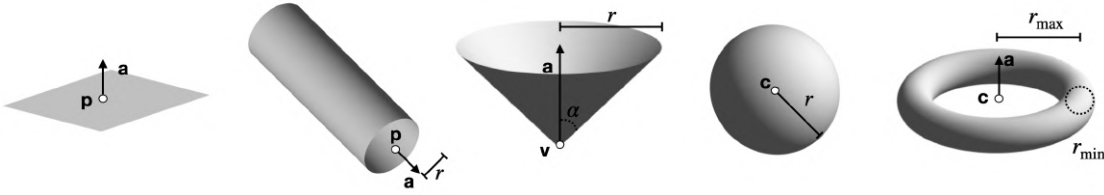


Figure 5.1: Simple geometric primitives: plane, cylinder, cone, sphere and torus respectively, along with their attributes (geometric descriptors).

5.2.2 Complex geometric primitives

In addition to the simple geometric primitives of Figure 5.1, in this section we introduce a set of complex geometric primitives (general cylinders and cones, surfaces of revolution, convex combination of curves, helical surfaces). To the best of our knowledge, these primitives have never been used before for recognition by using the HT.

For the sake of simplicity, some of the surfaces are here presented in their standard 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 directrix is extremely rich, see [150]. As an example, Table 5.1(a) considers a 5-convexity curve as a directrix.

- *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. Table 5.1(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*; another example is given by ellipsoids, which are obtained by rotating an ellipse with respect to one of its principal axes. Table 5.1(c) shows the case of a surface obtained by rotating the curve $(x(u), y(u), z(u)) := (au, 0, b/u^5)$ around the z -axis.


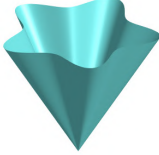

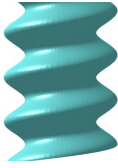

- *Convex combination of curves.* It is possible to define surface primitives by considering the convex combination of a pair of parametrised curves $(f_1(\mathbf{a}, u), f_2(\mathbf{a}, u), f_3(\mathbf{a}, u))$ and $(g_1(\mathbf{b}, u), g_2(\mathbf{b}, u), g_3(\mathbf{b}, u))$. This family has the following parametric equations:

$$\begin{cases} x = v f_1(\mathbf{a}, u) + (1 - v) g_1(\mathbf{b}, u) \\ y = v f_2(\mathbf{a}, u) + (1 - v) g_2(\mathbf{b}, u) \\ z = v f_3(\mathbf{a}, u) + (1 - v) g_3(\mathbf{b}, u) \end{cases} ,$$

where $v \in [0, 1]$. Note that the primitive parametrisation 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 considering a convex combination of two helices of equal slope but different radii. An example of a helical strip is provided in Table 5.1(e).

- *Helical surfaces.* Table 5.1(d) presents a family of equations obtained by modifying the parametrisation of a circular cylinder. Precisely, the radius R here varies between $[R_1, R_2]$, where $R_1 > 0$, by a cosine function; when 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).

Table 5.1: Parametric representation of some complex geometric primitives.

(a) generalised cylinder	(b) generalised cone	(c) surface of revolution	(d) helical surface	(e) helical strip
				
$\begin{cases} x = \frac{a \cos u}{1+b \cos(5u)} \\ y = \frac{a \sin u}{1+b \cos(5u)} \\ z = v \end{cases}$	$\begin{cases} x = \frac{av \cos u}{1+b \cos(5u)} \\ y = \frac{av \sin u}{1+b \cos(5u)} \\ z = Av + B \end{cases}$	$\begin{cases} x = au \cos v \\ y = au \sin v \\ z = \frac{b}{u^5} \end{cases}$	$\begin{cases} x = R(u) \cos v \\ y = R(u) \sin v \\ z = k_1(u + nv) + k_2 \end{cases},$ <p style="text-align: center;">where $R(u) := R_1 + \frac{R_2 - R_1}{2}(\cos u + 1),$ $n \in \mathbb{Z}$</p>	$\begin{cases} x = R(u) \cos v \\ y = R(u) \sin v \\ z = v \end{cases},$ <p style="text-align: center;">where $R(u) := au + (1-u)b$</p>

5.3 Recognition and fitting primitives from 3D segmented point clouds

We propose a new method based on the Hough transform that, given a segmented point cloud $\mathcal{P} = \bigcup_{i=1}^{num} \mathcal{P}_i$ representing a man-made object, recognises simple geometric primitives (see Section 5.2.1) and their interrelationships. Specifically, we introduce a novel technique able to provide an initial estimate of the geometric parameters characterising each primitive type. By using these estimates, we localise the search for the optimal solution in dimensionally-reduced parameter spaces, thus making it efficient to extend the HT to more primitives than those generally found in the literature, i.e., planes and spheres.

5.3.1 Description of the method

In the following, we detail the steps of our method to recognise and fit geometric primitives in a segmented point cloud presented in [128], denoting with \mathcal{P}_i each segment. We first describe, for each family of geometric primitives introduced in Section 5.2.1, how to compute the initial estimates and then, we show how to exploit the parameters estimation in the method flow. The result in the output is a set of geometric descriptors characterising each recognised primitive. Finally, we provide details on a post-processing step that uses geometric descriptors to find relationships between different primitives of the same type.

Segment centring and normal estimation via local HT fits. The following steps 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 segment \mathcal{P}_i is centred, i.e., it is translated so that its barycentre coincides with the origin of the Cartesian coordinate system.
- *Point cloud downsampling.* The axis-aligned minimum bounding box of \mathcal{P}_i is split into s equal-sized boxes. Points within the same box $j = 1, \dots, s$ will be replaced (i.e., downsampled) by the point 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 to reduce the execution time.
- *Normal estimation.* For each $\mathbf{p}_j \in \mathcal{P}_i$, 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 as Eq. 5.2.1, then 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.
- *Normal accuracy.* As the last step, we compute the accuracy of each candidate tangent plane by using the *Mean Fitting Error* $\text{MFE}(\mathcal{N}_j, \pi_j)$, defined as Eq. 3.2. From here on, we denote $\text{MFE}(\mathcal{N}_j, \pi_j)$ by MFE_j and $\mathbf{MFE} := [\text{MFE}_1, \text{MFE}_2, \dots]$. Finally, we select the points corresponding to the lowest entries in \mathbf{MFE} , i.e., having the most accurate estimations of the normal vector. More precisely, the entries are selected through a threshold that depends on a percentage (a typical value is 2%) of the maximum among the length, width, and height of the bounding box of \mathcal{P}_i . Let $\mathbf{p}_1, \dots, \mathbf{p}_k$ denote such points.

Initial estimates. Once the candidate tangent planes have been estimated, we specialize the processing for each type of primitive. For the sake of clarity, we show the procedure on complete primitives; however, the method can also deal with parts of primitives.

- **Sphere.** For each $\mathbf{p}_j \in \mathcal{P}_i$, we sample a point \mathbf{q}_j on the (candidate) tangent plane π_j . Then, we consider the¹ plane $\tilde{\pi}_j$ 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 5.2(c): the blue, green and red vectors are, respectively, \mathbf{n}_j , \mathbf{t}_j and \mathbf{v}_j . The plane $\tilde{\pi}_j$ intersects the sphere into a set of points outlining a circle, which can be recognised through the classical HT procedure for circles, and whose approximation is evaluated by the Mean Fitting Error. In exact arithmetic, $\tilde{\pi}_j$ passes through the sphere centre, which corresponds to the circle centre as well; in floating-point arithmetic (or when the input segment is perturbed), the centre and the radius of the circle give an estimate of the centre and the radius of the sphere.

¹Despite the existence of infinitely many planes that are perpendicular to π_j , here we choose one by fixing a point on π_j and its normal vector.

By repeating this procedure for all points in \mathcal{P}_i – or at least, for a representative subset of points – we obtain a set of estimates of the sphere centre and radius. By thresholding the MFE, we can discard low-quality estimates; finally, by averaging over the remaining centres and radii we obtain the final estimates of the sphere centre and radius, which will be denoted by $\hat{\mathbf{c}}$ and \hat{r} .

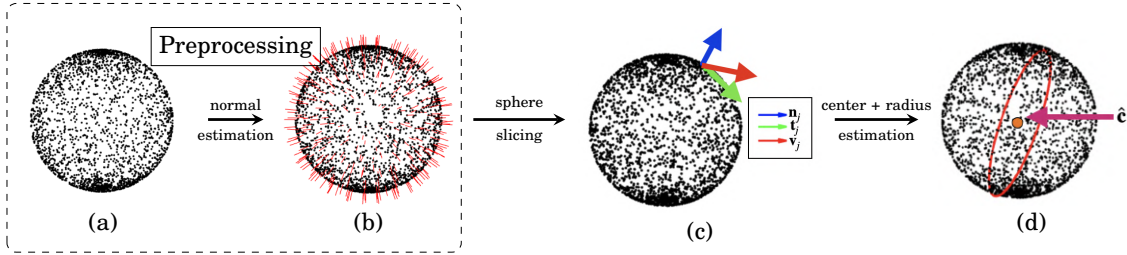


Figure 5.2: Initial estimates for a sphere. The preprocessing step centres the point cloud and approximates the normal vectors at a set of points, see (b). Given a point \mathbf{p}_j and a point \mathbf{q}_j on its tangent plane, (c) shows the vectors \mathbf{n}_j , \mathbf{t}_j and \mathbf{v}_j in blue, green and red, respectively. Finally, (d) shows the estimate $\hat{\mathbf{c}}$ of the centre.

- **Cylinder.** For each pair of points \mathbf{p}_{j_1} and \mathbf{p}_{j_2} , where $j_1, j_2 = 1, \dots, k$ 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 5.3(c) represents a simplified situation, where the two normals (in green and blue) and their cross product (in 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}_i 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 centre \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 the HT-based recognition process, see Figure 5.3(d).

- **Cone.** From basic geometry, we know that, in exact arithmetic, the vertex of a cone can be found by intersecting (at least) three tangent planes. In case of data perturbation, however, such an intersection will be most likely empty. To overcome this problem, we define a voting procedure that exploits the representations of the tangent planes of the points $\mathbf{p}_1, \dots, \mathbf{p}_k$. More specifically, since for each tangent plane π_j the normal \mathbf{n}_j and the term ρ_j are known, the voting procedure considers the coordinates x , y and z of the vertex as the parameters to be estimated. The most voted coordinates correspond to the vertex $\hat{\mathbf{v}}$.

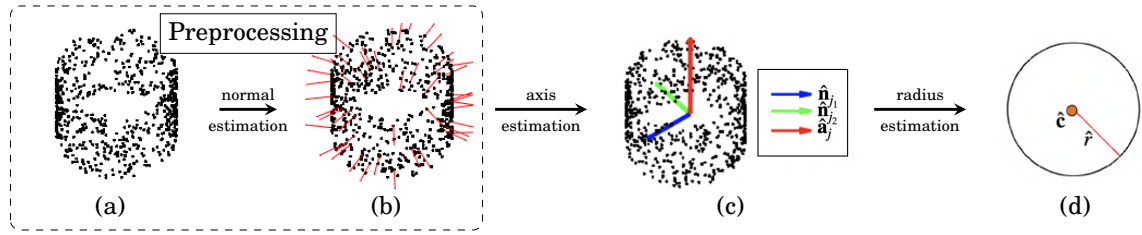


Figure 5.3: Initial estimates for a circular cylinder. The preprocessing step centres the point cloud and approximates the normal vectors at a set of points, see (b). Given two points \mathbf{p}_{j_1} and \mathbf{p}_{j_2} , (c) shows the corresponding normal vectors $\hat{\mathbf{n}}_{j_1}$ and $\hat{\mathbf{n}}_{j_2}$, in blue and green respectively, and, in red, their cross product $\hat{\mathbf{a}}_{j_1,j_2}$. Finally, (d) shows the estimate \hat{r} of the radius.

Then, for each pair of points \mathbf{p}_{j_1} and \mathbf{p}_{j_2} , where $j_1, j_2 = 1, \dots, k$ 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 a rotational axis, is shown in Figure 5.4(c). 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}_i in its canonical form, we apply a roto-translation 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{a} 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}_i .

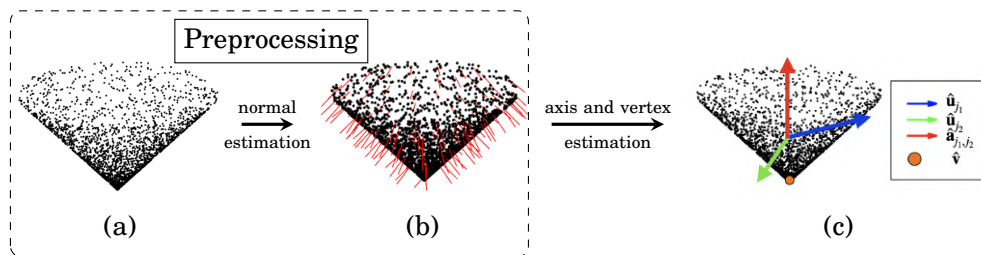


Figure 5.4: Initial estimates for a circular cone. The preprocessing step centres the point cloud and approximates the normal vectors at a set of points, see (b). The intersection of the tangent planes estimates the coordinates of the vertex $\hat{\mathbf{v}}$. Given two points \mathbf{p}_{j_1} and \mathbf{p}_{j_2} , (d) shows 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 more complex handling, as summarised in the following four steps:

- *Upper (or lower) circle recognition.* We search for the best fitting plane to the entire

point cloud \mathcal{P}_i , which – unless pathological cases (e.g., very small segments) – intersects the torus in (possibly perturbed arcs of) a circle, as shown in Figure 5.5(c, left). This circle can be recognised 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 5.5(c, right).

- *Recognition of small circles.* For each of the points $\mathbf{p}_j, j = 1, \dots, k$, 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 5.5(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 5.5(d, right image) – that intersects \mathcal{P}_i 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 centres. For each recognised circle, we compute its Mean Fitting Error and store its centre in \mathcal{C} if the MFE is below some given threshold. By averaging the circle radii, we can get an estimate \hat{r}_{\min} of r_{\min} .
- *Towards axis estimation.* We use the 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 5.5(e)).
- *Recognition of the big circle for centre estimation.* Finally, we recognise the circle outlined by the points in \mathcal{C} , see Figure 5.5(f). The centre of the torus is approximated by the circle centre, which can be also used to fix the rotational axis. The radius of the circle gives us an estimate \hat{r}_{\max} of r_{\max} .

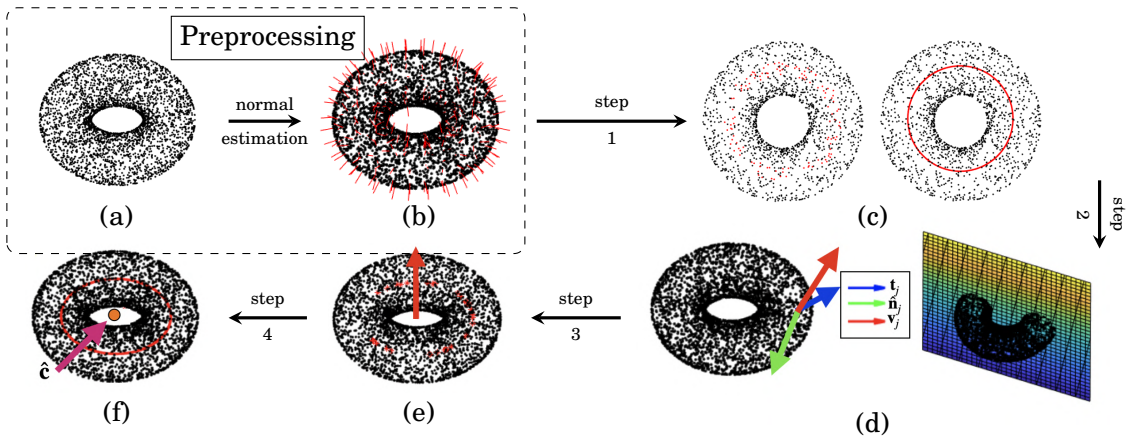


Figure 5.5: Initial estimates for a torus. The preprocessing step centers the point cloud and approximates the normal vectors at a set of points, see (b). In (c) the upper/lower circle recognition, while (d) shows the plane that identify small circles. Given the centers of small circles, in (e) the estimation of the axis $\hat{\mathbf{a}}$ and in (f) the estimation of center $\hat{\mathbf{c}}$ of the torus.

Recognising primitives using the Hough transform. We can now exploit the preprocessing step and the parameter estimation procedures to apply the HT technique to each segment \mathcal{P}_i ; this is particularly relevant when no prior information on the primitive type to look for is available. For each segment, our pipeline returns its type (i.e., plane, cylinder, cone, sphere, torus), and its geometric descriptors. Figure 5.6 illustrates the flow we follow to recognise primitives using the HT.

Given a segment \mathcal{P}_i and a specific family of surfaces $\mathcal{F} = \{\mathcal{S}_\beta\}$ of the dictionary of surface primitives (e.g., a family of tori), our method runs in three main steps:

Step 1: Preprocessing and initial estimates. First, the segment \mathcal{P}_i is preprocessed and then, we find initial estimates for the current family of surfaces by applying the corresponding procedure as described previously. The point cloud obtained from the preprocessing step, here denoted by \mathcal{P}'_i , is roto-translated to put it in the standard position. Note that, by working on \mathcal{P}'_i rather than on \mathcal{P}_i , we are able to deal with a dimensionally reduced parameter space.

Step 2: HT-based surface recognition. The new set of points \mathcal{P}'_i is the input of the classical HT-based recognition algorithm introduced in Section 1.3.4. Since we use the parametric representations for the families of primitives, the estimation of the accumulator matrix is done by adapting to surfaces the strategy devised for plane curves in Section 2.3. In particular, we compute automatically a sample of $\Gamma_{\mathbf{p}}$ through the Moore-Penrose pseudo-inverse [120] and the intersection is evaluated via an inequality between the components of the sample of $\Gamma_{\mathbf{p}}$ and the coordinates of the cell endpoints. The output consists of the optimal parameters $\hat{\beta}'$, i.e., the parameters that best fit \mathcal{P}'_i w.r.t. the given family of surfaces (in canonical position). The initial estimates from the previous step give a hint to the HT technique about where the optimal solution is, thus eliminating the problem of the unboundedness of the parameter space.

Step 3: Evaluation of the approximation accuracy. To measure the recognition accuracy of a specific primitive, we use the Mean Fitting Error $\text{MFE}(\mathcal{P}'_i, \mathcal{S}_{\hat{\beta}'})$, as defined in Eq. 3.2. When the accumulator function exhibits more global maxima, the Hough transform returns more optimal solutions; in our case, we keep only the one having the lowest MFE.

When no prior information on the primitive type is available, the three steps above are repeated for each family of surfaces (in this work: planes, cylinders, spheres, cones and tori); the surface with the lowest MFE is returned as the best fitting surface $\mathcal{S}_{\hat{\beta}'}$ for the segment \mathcal{P}'_i . We admit that none of the primitive types at our disposal offers a satisfying fit of the segment if all the computed MFEs are above a global threshold ε , here defined as the 5% of the main diagonal of the model bounding box.

Finally, the roto-translation from Step 2 is applied backward in order to obtain the parametric representation and the geometric descriptors for the segment \mathcal{P}_i in its original position.

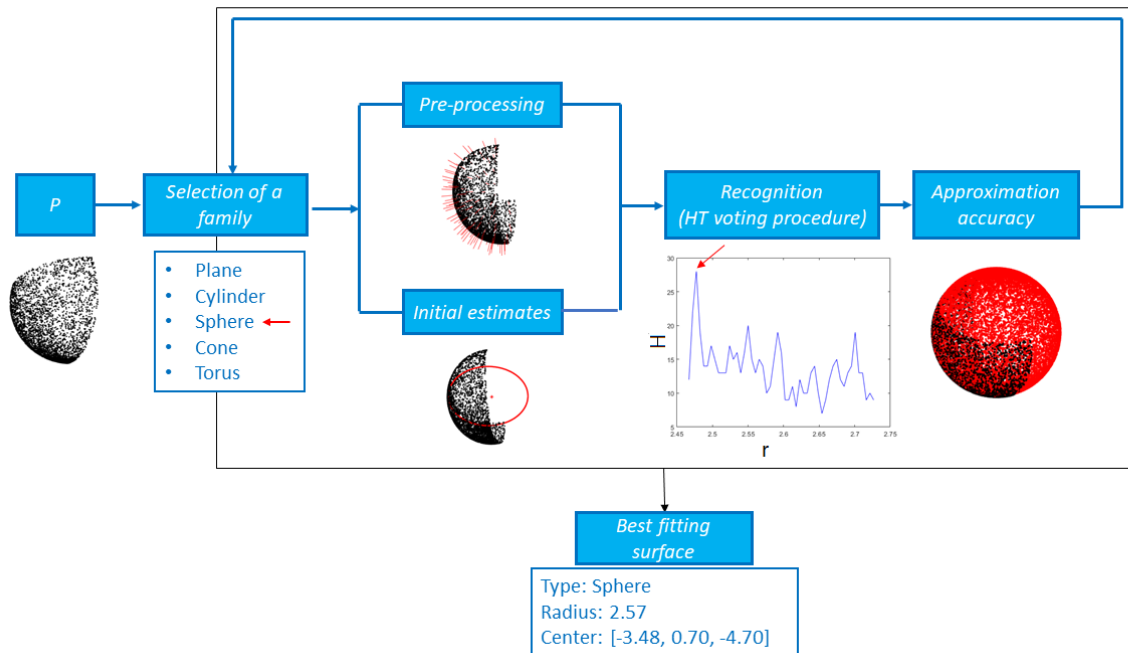


Figure 5.6: Pipeline of the method for fitting and recognising primitives using the Hough transform.

Post-processing: primitive aggregation. The parameters provided by the recognition process uniquely characterise the segments and can be used as geometric descriptors allowing them to be aggregated using a clustering approach. The pipeline of such a method is built over two consecutive steps.

Step 1. Firstly, we apply the algorithm described before to each input segment. This allows us to label each segment with its most likely primitive type, as well as obtain its parametric representation and its shape description.

Step 2. Once all segments have been processed, we apply a well-known (hierarchical) clustering approach – the *complete-linkage* – to compare clusters and build a dendrogram. The use of complete linkage is here justified by the need of penalising chaining effects. 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

$$D(C_h, C_j) := \max_{\tau_k \in C_h, \tau_l \in C_j} d(\tau_k, \tau_l),$$

where C_h, C_j is a given pair of clusters (of segments) and d is a user-defined distance or dissimilarity. For any pair of segments τ_1, τ_2 belonging to the same family, several distances $d(\tau_1, \tau_2)$ are possible. For each type of primitive, Table 5.2 lists some simple distances and

the intuitive concepts they intend to measure, using the notation introduced in Section 5.2.1.

In practice, we have at our disposal a set of distances, each one corresponding to a condition we are interested to measure. Note that all distances listed in Table 5.2 are metrics and $d(\tau_1, \tau_2) = 0$ implies that the primitives τ_1 and τ_2 are equal with respect to that criterion.

Table 5.2: Basic distances for the primitives considered in the Section 5.2.1, using the notation from Figure 5.1. The subscripts 1 and 2 in the parameters refer to the two segments τ_1 and τ_2 .

Primitives	Distance $d(\tau_1, \tau_2)$	Query
Planes	$\left\{ \begin{array}{l} \ \mathbf{a}_1 \times \mathbf{a}_2\ _2 \\ \mathbf{a}_1 \cdot (\mathbf{p}_1 - \mathbf{p}_2) \end{array} \right.$	parallel incident
Cylinders	$\left\{ \begin{array}{l} r_1 - r_2 \\ \ \mathbf{a}_1 \times \mathbf{a}_2\ _2 \\ \ \mathbf{a}_1 \times (\mathbf{p}_1 - \mathbf{p}_2)\ _2 \end{array} \right.$	equal radii parallel rotational axes incident rotational axes
Cones	$\left\{ \begin{array}{l} \alpha_1 - \alpha_2 \\ \ \mathbf{a}_1 \times \mathbf{a}_2\ _2 \\ \ \mathbf{v}_1 - \mathbf{v}_2\ _2 \end{array} \right.$	equal apertures parallel rotational axes equal vertices
Spheres	$\left\{ \begin{array}{l} r_1 - r_2 \\ \ \mathbf{c}_1 - \mathbf{c}_2\ _2 \end{array} \right.$	equal radius equal centers
Tori	$\left\{ \begin{array}{l} r_{\min,1} - r_{\min,2} \\ r_{\max,1} - r_{\max,2} \\ \ \mathbf{a}_1 \times \mathbf{a}_2\ _2 \\ \ \mathbf{c}_1 - \mathbf{c}_2\ _2 \end{array} \right.$	equal smallest radii equal largest radii parallel rotational axes equal centers

In addition, more complex queries can be formulated by summing simple distances. For instance, one can check whether two segments lie:

- On the same torus by using the metric $d(\tau_1, \tau_2) := |r_{\min,1} - r_{\min,2}| + |r_{\max,1} - r_{\max,2}| + \|\mathbf{a}_1 \times \mathbf{a}_2\|_2 + \|\mathbf{c}_1 - \mathbf{c}_2\|_2$.
- On the same torus, up to a translation, by considering the distance $d(\tau_1, \tau_2) := |r_{\min,1} - r_{\min,2}| + |r_{\max,1} - r_{\max,2}| + \|\mathbf{a}_1 \times \mathbf{a}_2\|_2$.
- On the same torus, up to a roto-translation, by employing the metric $d(\tau_1, \tau_2) := |r_{\min,1} - r_{\min,2}| + |r_{\max,1} - r_{\max,2}|$.

Note that the sum of distance is still a distance and that cutting the dendrogram with increasing thresholds corresponds to weakening the conditions imposed as the query.

5.3.2 Computational complexity

The method includes segment preprocessing and the actual primitive fitting. The segment preprocessing includes the estimation of distances, the axis-aligned bounding box and an approximation

of the tangent plane via the HT. For spheres, cones, cylinders and tori, the parameter estimation includes the recognition of circle(s), again by using the HT. The segment recognition procedure basically consists of the HT procedure for the surface primitives in standard form. In practice, both the preprocessing and the fitting procedure mainly depend on the HT, being the other operations in $O(S)$ or in $O(S \log S)$, where S represents the number of points of the segment. The computational complexity of the HT voting procedure is dominated by the size of the accumulator function: denoting M the number of cells of the space of parameters, the computational complexity of the HT recognition on a segment is $O(MS)$. The number of parameters for the HT directly influences the size of M . In our method, their number is 3 for planes and circles (they are used in the preprocessing), 2 for tori, and 1 for spheres, cylinders and cones. Through this parameter reduction, we are able to deal with parametric primitives that would otherwise have 4 parameters (sphere) or at least 9 parameters (cylinder, cone and torus), as deducible from the representations in Section 5.2.1. This reduction in parameters makes it possible to apply the HT to primitives that would otherwise not be computable in practice due to the explosion of the spatial complexity of the accumulator function.

The preprocessing and the HT recognition steps are repeated for each segment and for each geometric primitive. It is worth noting that, being each segment and each primitive fitting performed independently, the task is *embarrassingly parallel*.

Regarding the clustering step, the implementation of agglomerative hierarchical clustering requires $O(num^3)$ operations, where num is the number of segments given in input (see [43]). One can think about more effective implementations of complete-linkage clustering, such as the one suggested in [44], which costs $O(num^2)$.

5.3.3 Performance over different datasets

We evaluate the capability of our method to fit and aggregate primitives according to different correlation queries. For all models, in this section, we show how primitives are aggregated if they belong to the same geometric primitive or according to different relations, such as co-planarity, co-axiality, and parallelism. Despite much broader experimentation, we show only some figures with the most significant relationships found. To better evaluate the behaviour of our method, we used datasets available online. For all examples, the following thresholds for cutting the dendrograms have been selected: 10^{-10} for planes; 10^{-1} for spheres, cylinders, conic, and tori.

The first dataset we considered was the ABC dataset [91], which contains a collection of one million CAD models created for research of geometric deep learning methods and applications. Figure 5.7 presents a point cloud of 22,803 points containing 18 segments: planes, cylinders, cones, and tori. Segments satisfying the same correlation query are represented by the same colour. As shown in the images, our method can successfully recognise the primitive type and use the segment parameters to infer various relations. The expressions “same plane”/“same cylinder”/“same cone”/“same sphere”/“same torus” are henceforth used to test the (possible)

presence of segments originating from the same underlying primitive (e.g., to group together toric segments having the same centre, radii and rotational axis). The mean MFE over all segments is 0.0019.


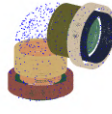
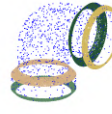


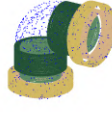






Model and planes	Cylinders	Cones	Tori
 Original model	 Same cylinder	 Same cone	 Same torus
 Same plane	 Same radius	 Same radius	 Same radius
 Parallel planes	 Same rotational axis	 Same rotational axis	 Same rotational axis

Figure 5.7: A model from [91]: we show segments of the same primitive type exhibiting different similarities. Colours are used to visually represent primitives sharing the same property.

The point set in Figure 5.8(a) is composed of 13,455 points and contains 26 segments, of which: 2 are extracted from planes, 8 from cylinders, and 16 from tori. This model is perfectly handled by our method, without misclassification in any correlation query. The mean MFE over all segments is 0.0036. The model in Figure 5.8(b) counts 9 segments for a total of 15,726 points. In this example, a pair of segments obtained from the same sphere is present. Again, the grouping proceeds smoothly, except for two queries where two cylinders with very similar radii are clustered together. This misclassification can be partly justified by the intrinsic approximation that voting procedures introduce when discretizing the parameter space. On the other hand, it is also worth noting that our procedure for (initial) parameter estimation generally exhibits a lower precision when applied to small and low-sampled segments, due to a higher error in the tangent plane approximation. In this case, the mean MFE over all segments is 0.0042.

Finally, we test the resilience to increasing noise in Figure 5.9 for a point cloud composed of 9,723 points and 35 segments. We added synthetic Gaussian noise of fixed mean $\mu = 0$ and standard deviation σ equals to 0.10, 0.25 and 0.50. We note that, as the noise intensity increases, some segments start being misclassified; more specifically, when $\sigma = 0.50$, two cylinders are mislabeled as cones and thus wrongly clustered. The mean MFE over all segments in the three cases is, respectively, 0.0106, 0.0212 and 0.0347.

5.3. RECOGNITION AND FITTING PRIMITIVES FROM 3D SEGMENTED POINT CLOUDS

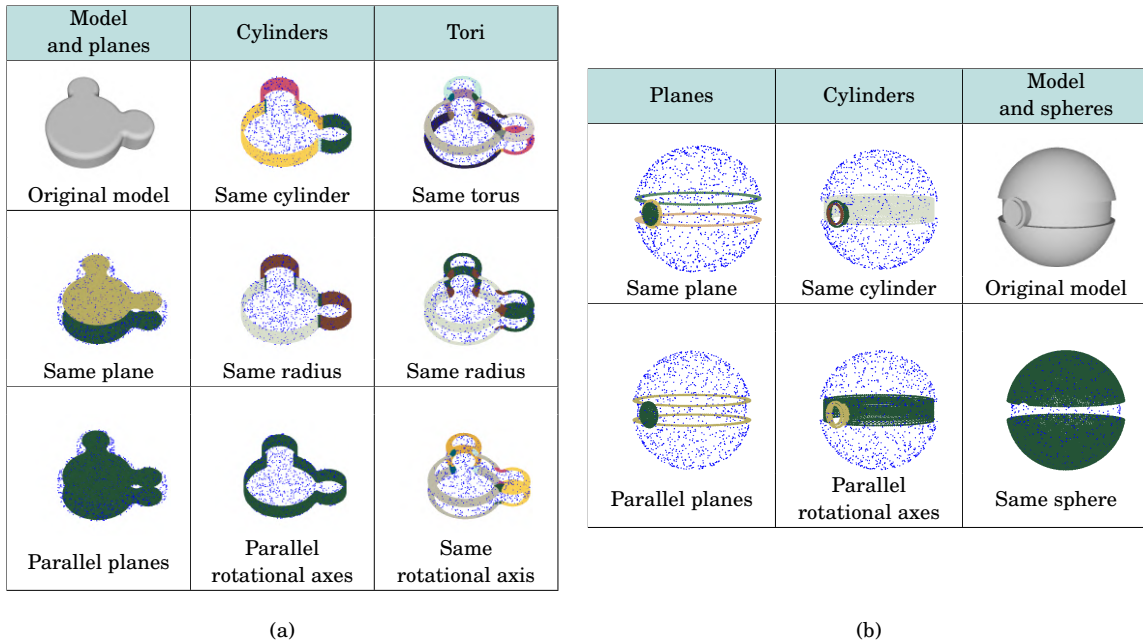


Figure 5.8: Two models from [91]: segments exhibiting different similarities are shown.

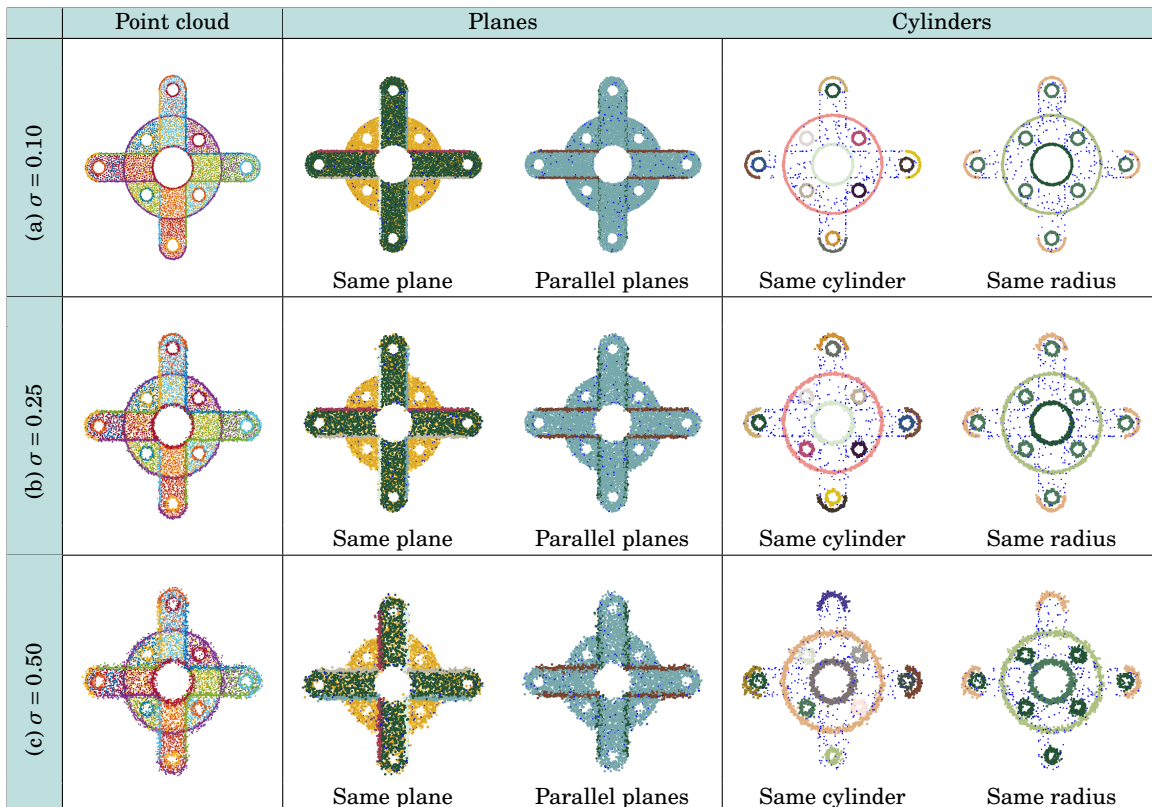


Figure 5.9: A model from [91]: a Gaussian noise is applied at three levels of intensity.

The second dataset we considered for our tests is Fit4CAD [140] (see in the Appendix, Section A.2). Fit4CAD is a benchmark created specifically for the evaluation and comparison of methods for fitting simple geometric primitives in point clouds representing CAD objects. This dataset contains models affected by some point cloud artefacts (e.g., undersampling or missing data). Exploiting the provided ground truth, we tested our method on the models coming from this dataset, and we were able to evaluate the results.

Specifically, Table 5.3 reports the value of five common classification measures employed in [140]: True Positive and Negative Rates (TPR and TNR), Positive and Negative Predicted Values (PPV and NPV), and accuracy (ACC). The values presented in the table are obtained by averaging the single measures over all models. The average accuracy ACC is always above 90%; the only cases where it is below 95% correspond to queries involving tori, suggesting reduced robustness for the corresponding geometric descriptors. The average TNR is always above 98%, while the average TPR is always lower than the average TNR. These results show that the method is great to spot true negatives, while it is less efficient to cluster together segments. This is particularly an issue for spheres and tori. A possible solution to alleviate this problem would be to allow the use of user-defined thresholds for the dendrogram. Average PPV and NPV show the degree of correctness of the method in indicating possible positives or negatives. Again, the method has a slightly lower performance for tori.

Table 5.3: Average classification performances for different correlation queries on the Fit4CAD dataset.

Primitive	Query	PPV	TPR	TNR	NPV	ACC
Plane	Same plane	0.998	0.980	1.000	0.999	0.999
	Parallel planes	0.975	0.962	0.992	0.994	0.987
Cylinder	Same cylinder	0.995	0.980	0.999	0.999	0.998
	Same radius	0.974	0.970	0.995	0.996	0.991
	Parallel rotational axes	0.977	0.976	0.991	0.980	0.984
	Same rotational axis	0.992	0.985	0.995	0.998	0.995
Cone	Same cone	1.000	0.944	1.000	0.994	0.994
	Same aperture	1.000	0.922	1.000	0.981	0.985
	Same apex	1.000	0.944	1.000	0.994	0.994
	Parallel rotational axes	1.000	0.944	1.000	0.986	0.988
	Same rotational axis	1.000	0.833	1.000	0.942	0.952
Sphere	Same sphere	1.000	0.667	1.000	0.976	0.976
	Same radius	1.000	0.667	1.000	0.976	0.976
	Same center	1.000	0.667	1.000	0.976	0.976
Torus	Same torus	1.000	0.583	1.000	0.958	0.958
	Same radii	1.000	0.583	1.000	0.958	0.958
	Same center	0.889	0.600	0.985	0.955	0.943
	Parallel rotational axes	0.875	0.583	0.993	0.896	0.901
	Same rotational axis	0.875	0.583	0.993	0.896	0.901

5.3.4 Comparison with the method [127]

We compared our method on the Fit4CAD dataset with [127] because it adopts a pipeline similar to ours and its implementation is available online.

The method [127] consists of a combination of approximate implicitization – which reduces to a least squares minimization in its discrete formulation [16] – and hierarchical clustering. Unlike our method, it uses the coefficients of implicit representations as geometric descriptors; although computationally efficient, this choice results in problems of instability when point cloud artefacts are present, as pointed out by the authors. Moreover, [127] only checks which segments lie on the same surface, but it does not identify the segment type (e.g., cylinder vs. cone), nor formulate more complex queries.

In Table 5.4, the comparison is drawn for the five classification measures mentioned before. We can notice a generally better performance of our approach, especially in the case of noisy or perturbed data. Indeed, polynomial estimations used in [127] have been proven to be particularly sensitive to data perturbation, while the HT paradigm is popularly known for its robustness. On the other hand, [127] has lower computational complexity, making it preferable for an initial inspection when the input is clean or when the user is not interested in other correlation queries.

Table 5.4: Classification performance: comparison between our method and the approach proposed in [127].

Model	Method	PPV	TPR	TNR	NPV	ACC
Fig 5.7	Ours	1.000	1.000	1.000	1.000	1.000
	[127]	0.611	1.000	0.948	1.000	0.951
Fig 5.8(a)	Ours	1.000	1.000	1.000	1.000	1.000
	[127]	0.739	0.957	0.937	0.996	0.936
Fig 5.8(b)	Ours	0.875	1.000	0.964	1.000	0.969
	[127]	0.875	1.000	0.964	1.000	0.969
Fig 5.9(a)	Ours	1.000	1.000	1.000	1.000	1.000
	[127]	0.079	0.988	0.094	0.952	0.160
Fig 5.9(b)	Ours	1.000	1.000	1.000	1.000	1.000
	[127]	0.076	1.000	0.000	0.000	0.076
Fig 5.9(c)	Ours	0.905	0.905	1.000	0.998	0.998
	[127]	0.076	1.000	0.000	0.000	0.076
Fit4CAD	Ours	0.995	0.997	0.999	0.999	0.999
	[127]	0.791	0.992	0.946	0.998	0.950

5.3.5 Tests on point clouds segmented by different methods

To evaluate the performance of our method, we applied it to point clouds segmented with different techniques. The first set of tests includes segments obtained with the learning technique presented in [148]. The second set of tests considers segments obtained by applying RANSAC [144] on industrial scans.

5.3.5.1 Tests on point clouds segmented by a learning approach

We tested our method on some of the segmented point clouds provided by a learning approach [148] that are composed of 10,000 points.

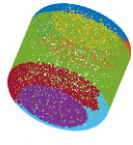
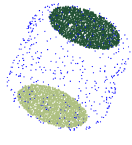
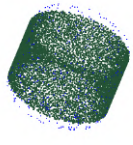
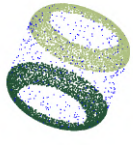
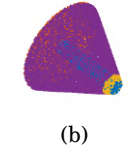
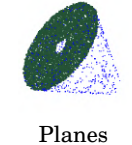
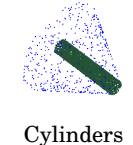
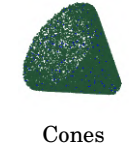
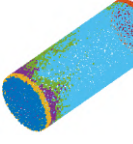
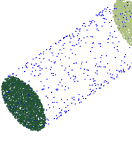
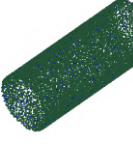
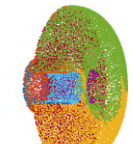


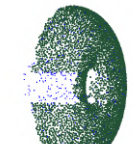

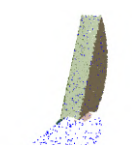
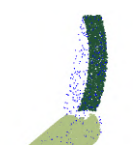
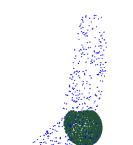
Segmentation	Primitives		
 (a)	 Planes	 Cylinders	 Tori
 (b)	 Planes	 Cylinders	 Cones
 (c)	 Planes	 Cylinders	
 (d)	 Planes	 Cylinders	 Tori
 (e)	 Planes	 Cylinders	 Spheres

Figure 5.10: Examples of segmentations from [148]: the original segmentations (first column) are post-processed by our method to overcome the problem of oversegmentation.

In this case, our technique is useful as post-processing to overcome the problem of oversegmentation and may confirm or modify the classification of the primitive. Indeed, as shown in Figure 5.10, the combination of the geometric descriptors identification and clustering procedure permits aggregate segments belonging to the same primitive. Specifically, Figure 5.10(a) presents a point cloud divided into 7 segments, where four of them can be grouped two by two since they

belong to the same torus. A similar grouping of segments belonging to the same torus is provided in Figure 5.10(d). Figure 5.10(b) shows a model divided into 4 segments and composed of a plane, a cylinder and a cone split into two parts, of which one is very small. Our method groups these two pieces since they belong to the same primitive. In Figure 5.10(c) the point cloud is divided into 7 segments, where five of them belong to the same cylinder. Finally, in Figure 5.10(e) a point cloud made of 12 segments is reduced to 9 by grouping four pieces of the same cylinder. The mean of the MFE over all segments is (a) 0.0097, (b) 0.0102, (c) 0.0036, (d) 0.0048, (e) 0.0061.

5.3.5.2 Tests on industrial scans segmented by the RANSAC technique

We tested the method on two industrial scanned objects used in [100].



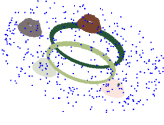
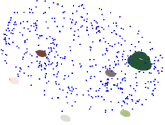
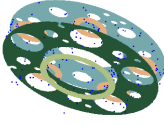
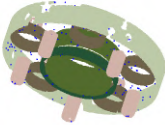
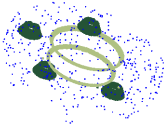
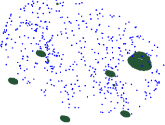
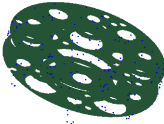

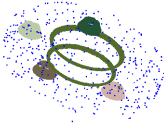
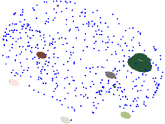
Point cloud and planes	Cylinders	Cones	Spheres
 Original model	 Same cylinder	 Same cone	 Same sphere
 Same plane	 Same radius	 Same radius	 Same radii
 Parallel planes	 Same rotational axis	 Same rotational axis	 Same centre

Figure 5.11: Examples of queries for planar, cylindrical, conical, and spherical segments obtained from the RANSAC segmentation of a scanned industrial object as provided in [100].

Unfortunately, no ground truth is available for these objects, and we will only report the average MFE. As a starting point, we used the same input as [100], i.e., RANSAC segmentations made available online by the authors on their GitHub page. The simplest example, shown in Figure 5.11, contains 282,534 points and 39 segments. The main axes of planes, cylinders and cones are found to be parallel. Besides parallelism, this model also exhibits cylinders, cones and


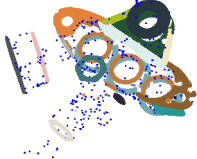




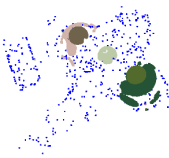
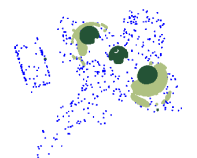
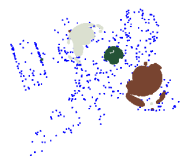
Model and planes	 Original model	 Same plane	 Parallel planes
Cylinders	 Same cylinder	 Same radius	 Same rotational axis
Spheres	 Same sphere	 Same radius	 Same center

Figure 5.12: Examples of queries for planar, cylindrical, and spherical segments obtained from the RANSAC segmentation over a challenging point cloud acquired from an industrial object as provided in [100].

spheres with the same radius. Note that only one segment is wrongly labelled as a sphere instead of a truncated cone. Note that the same considerations about axis-aligned cylinders are provided in Figure 14 of [100], but we can also find other types of correlations. For this point cloud, the mean of the MFE over all segments is 0.0091.

The point set in Figure 5.12 contains 529,006 points for a total of 51 segments, and corresponds to a machined part. Not only is our approach able to find segments lying on the same plane or cylinder, but also to identify cylinders having the same radii or axes and spheres characterised by the same centers. The mean of the MFE over all segments is 0.0120.

5.4 Recognition and fitting primitives from 3D point clouds

In this section, we describe a new method to recognise an input point cloud \mathcal{P} representing a CAD object with geometric primitives via the HT technique. The result is a set of N_{seg} recognised segments \mathcal{S}_j , such that $\mathcal{P} = \bigcup_{j=1}^{N_{seg}} \mathcal{S}_j$. This method is an extension of the one described in Section 5.3.1, since it takes in input the entire CAD objects, and it is able to recognise multiple instances

of the same primitive presented on a single point cloud. Specifically, it contains in a certain step the procedure described in Section 5.3.1.

5.4.1 Description of the method

The proposed method consists of the following main stages: an initial point cloud preprocessing, followed by the iteration of a recognition step and a splitting phase, and a final clustering step. A graphical illustration of its pipeline is given in Figure 5.13.

Point cloud preprocessing. First, the input point cloud \mathcal{P} is translated so as to place its barycenter in the origin of the Cartesian coordinate system. Then, normals at the input points are estimated as follows: if \mathcal{P} is clean, we consider the method presented in [80], via the MATLAB function `pcnormals`; while we use the approach shown in [128], in case \mathcal{P} is noisy. The point cloud is rotated so that the most voted direction among the (just-estimated) normal vectors coincides with the z -axis. This process is necessary to test the existence – as a first check – of primitives in their standard form (i.e., with centres or vertices in the origin of the Cartesian axes and with the normal or the principal axis aligned to the z -axis): indeed, it limits the number of parameters to be estimated by the HT. Finally, \mathcal{P} is scaled into a unit cube.

Recognition step. After being preprocessed, \mathcal{P} becomes the input of the HT-based recognition step. Since \mathcal{P} can contain different instances of the same primitive type and primitives of different types, the standard HT procedure must be adjusted to allow such cases. This step can be summarised as follows:

- *Selection of one or more families of primitives.* The user can select the families of primitives to be used for recognition; in its default configuration, the algorithm tests the presence of simple geometric primitives – one family at a time. By studying the geometric properties of \mathcal{P} and by relating them to the geometric characteristics of the selected family (e.g., bounding box, radius, diagonal length), it is possible to initialise a region T of the parameter space. The region T is discretised into cells, which are uniquely identified by the coordinates of their centre. Then, an accumulator function \mathcal{H} , in the discretised form of a matrix, is initialized. The matrix entries are in a one-to-one correspondence with the cells of the discretisation performed in the previous step.
- *Estimation of the accumulator function.* An entry of the accumulator function \mathcal{H} is increased by 1 each time the Hough transform $\Gamma_{\mathbf{p}}$ of a point \mathbf{p} intersects the corresponding cell. In our case, surfaces are expressed in parametric form; then, to check if and where a Hough transform intersects some cells we adapt to surfaces the method devised for curves in Section 2.3. Specifically, if the system of equations defining the family can be solved analytically with respect to the unknown parameters \mathbf{a} , we calculate automatically a sample

of $\Gamma_{\mathbf{p}}$ by exploiting the Moore-Penrose pseudo-inverse [120] 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 $\Gamma_{\mathbf{p}}$ and the coordinates of the endpoints of the cells.

- *Selection of potential fitting primitives.* The cells corresponding to the peak values of the accumulator function \mathcal{H} have to be identified. When the set of points is assumed to represent a single surface profile, the traditional HT formulation aims at finding the maximum of the accumulator function \mathcal{H} ; when the family of surfaces is not Hough-regular, there might exist several maxima. On the other hand, if the point cloud is composed of different primitives, different peaks of \mathcal{H} identify potential primitives that might fit different parts. To select these peaks, we observe that peaks of the accumulator function corresponding to primitive shapes rise distinctly with respect to their neighbours and are well-characterised as isolated peaks. Formally said, we proceed by identifying peaks that have high *topological persistence*². In our implementation, peaks that correspond to primitives are automatically recognised by keeping the local maxima having a persistence higher than 10% of the maximum value of \mathcal{H} , by using the algorithm for persistent maxima proposed in [24]. The coordinates of the cell centres of the maxima correspond to the parameters of potentially recognised surface primitives.
- *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, [64]). If \mathcal{X} is sparse, the recognised primitive is considered a false positive; otherwise, for each cluster $\mathcal{X}_i \subset \mathcal{X}$ we define the *Mean Fitting Error* $\text{MFE}(\mathcal{X}_i, \mathcal{P})$, as defined in Eq. 3.2, where \mathcal{P} is the current primitive. 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 \mathcal{H} does not present predominant peaks, resulting in false positives which are identifiable by studying the sparsity of the fitted points.

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 $\text{MFE}(\mathcal{X}_i, \mathcal{P}_{i,1})$ and $\text{MFE}(\mathcal{X}_i, \mathcal{P}_{i,2})$ between each primitive and \mathcal{X}_i ; the primitive having lowest error is kept.

²The notion of topological persistence was introduced in [55] 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 the topological connection between the points, for details on topological persistence and saliency, we refer to [51, 55]. In our case, the domain is represented by the grid of T , the role of filtration is played by the accumulator function \mathcal{H} and we are interested only in the peaks of \mathcal{H} .

Each time a primitive is recognised, the points of \mathcal{P} close to the recognised primitive less than a threshold ε are discarded from \mathcal{P} . The value of ε typically ranges from 1% to 3% of the diagonal of the minimum bounding box of the \mathcal{P} , according to the type of primitive.

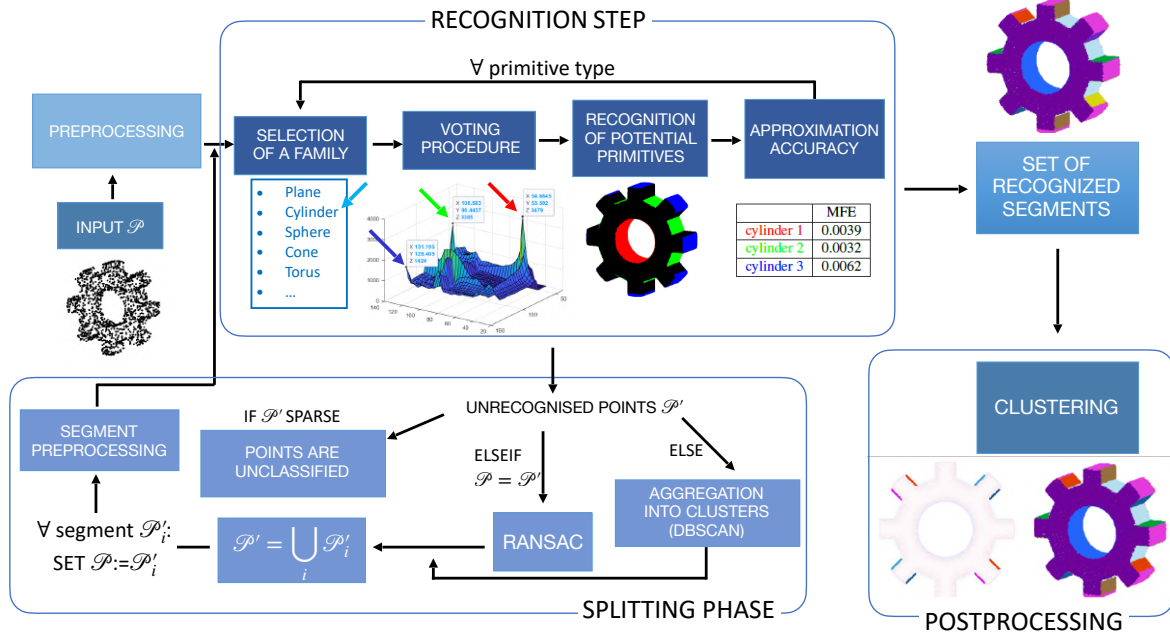


Figure 5.13: Pipeline of the method.

The algorithm returns the parameters of the geometric primitives and the corresponding points fitted by them, as well as the set of points that were not fitted by any primitive – denoted by \mathcal{P}' . 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.

Splitting phase. The algorithm chooses the next step according to the resulting \mathcal{P}' :

- If \mathcal{P}' is sparse, then its points are returned as unclassified.
- If $\emptyset \subsetneq \mathcal{P}' \subsetneq \mathcal{P}$, i.e., if \mathcal{P}' is a proper nonempty subset of \mathcal{P} , we proceed by aggregating points in \mathcal{P}' into clusters \mathcal{P}'_j , with $j = 1, \dots, N_{\text{clust}}$, by adopting the DBSCAN method [57]. Then, the recognition step mentioned before is iterated over each cluster \mathcal{P}'_j as long as some geometric primitives are recognised. Before proceeding with a new recognition round, we preprocess each cluster \mathcal{P}'_j ; the preprocessing adopted here differs from that applied to the entire point cloud \mathcal{P} : we exploit the strategy presented in [128] and detailed in Section 5.3.1 to estimate its standard form, thus reducing the number of parameters that have to be estimated in the new iteration of the recognition step. In case the recognition step involves complex primitives, the strategy proposed in Section 5.3.1 can be naturally extended since

they are characterised by the presence of symmetry axes that can be estimated in a similar way through the use of the normals of points.

- It is possible that, in some steps, no primitive is recognised. This can happen in two cases: (i) when applying our algorithm to an input point cloud that has no primitives in their standard form, or (ii) when the estimation of the standard position of some cluster \mathcal{P}'_j fails – because \mathcal{P}'_j contains more than one primitive. In both cases, we proceed by over-segmenting the model: in our experiments, we use the RANSAC algorithm proposed in [144], because of its efficiency and its tendency to over-segment point clouds (see, for example, [96]); however, we proceed by estimating primitive types and geometric parameters with a new round of the recognition step, as voting procedures have shown greater robustness to point cloud artefacts.

This step ends by transforming the parameters found by the Hough transform with respect to the inverse translations, rotations and scaling applied in the previous steps of the algorithm. The result of this procedure is the decomposition of an input point cloud \mathcal{P} into several subsets, called *segments*, in such a way that points of the same segment are well approximated by the same primitive. We denote these final segments by \mathcal{S}_j .

Segment clustering based on geometric relationships. After decomposing the input point cloud into the segments \mathcal{S}_j , a clustering method is applied to unveil geometric relationships. The goal of this step is to find maximal primitives (i.e., segments composed also of non-adjacent parts, belonging to the same primitive) that are not automatically detected in the recognition process or to find patterns of primitives. Specifically, we aggregate primitives based on their positions, orientation, and dimension, as described in Section 5.3.1 and in [128] for simple geometric primitives.

An illustrative example. Figure 5.13 provides a graphical illustration of the pipeline of our method. The main steps of the algorithm are associated to an example of a point cloud representing a gear. After preprocessing the point cloud, we start with the first round of the recognition step – which aims at recognising the presence of primitives in their canonical position. For the sake of clarity, however, the graphical illustration only focuses on the recognition of cylinders. Specifically, once the family of cylinders is selected, the corresponding accumulator function is computed; it exhibits three peaks, which indicate the presence of three potential solutions: the three cylinders highlighted in the colours red, green and blue. The approximation accuracy for this type of primitive is less than 1%.

Since some segments (the non-axis-aligned planar segments) are not in their canonical position, their points are not recognised in the first round of the recognition step. Instead, these points are collected in \mathcal{P}' and aggregated into dense clusters in the splitting phase; each of such clusters is individually studied by a new round of the recognition step and recognised as planar

segments. However, being these segments studied separately, we lose some geometric information. We then apply clustering to recognise that some planar segments lie on the same parametric plane, by exploiting the geometric descriptors provided by the HT.

The final result is a segmentation of 17 segments. It is worth mentioning that the method is able to group some of the non-adjacent parts, which belong to the same primitive in canonical position – as in the example of the two external cylinders, and of the axis-aligned planes that form four couples. However, for primitives not in their canonical position, postprocessing based on clustering is required.

5.4.2 Computational complexity

In the preprocessing step, our method includes the estimation of the normal vectors and the individuation of the most voted normal, which is implemented as explained in [80] – in the case of clean point clouds – and in [128] – in the presence of point cloud artefacts. For a thorough analysis of the computational complexities of this estimation, the reader is referred to the corresponding papers.

The formulation of the HT for surface primitives embedded in \mathbb{R}^3 naturally extends that for plane curves in \mathbb{R}^2 . In the pipeline presented in the previous section, for each point cloud \mathcal{P} (both in the case of the initial point cloud and of single clusters at subsequent iterations) we apply the HT by considering the different types of geometric primitives one at a time; thus, the dimension of the parameter space changes accordingly to the type of primitive considered. The quantization of a region of interest and the dimension of the parameter space determines the size of the accumulator function, and dominate both the memory usage and the computational complexity of the Hough transform: it is, therefore, necessary to balance the samples for each parameter and their number. To reduce the computational cost, primitives are put in their (estimated) standard positions in this way the number of parameters does not exceed 3, as explained in [128]; complementarily, adaptive approaches can be used to further speed up the search (e.g., [97]). The overall computational complexity of the HT-based recognition step is $O(ML)$, where M denotes the number of cells of the parameter space and L represents the number of points (in the initial point cloud or in a cluster obtained at a subsequent iteration) at which we evaluate the HT accumulator function. More precisely:

- At the first iteration the HT is applied to the whole input point cloud. Supposing that the number of families of primitives to be used for the recognition is K and the number of points in \mathcal{P} is N , the overall complexity of the first iteration is $O(N \sum_{k=1}^K M_k)$, where M_k is the dimension of the parameter space for the k -th primitive type.
- In the subsequent steps, the HT-based recognition method is applied to each cluster \mathcal{P}'_j and then the computational cost corresponds to $O(\sum_{j=1}^{N_{\text{clust}}} N_j \sum_{k=1}^K M_k)$, where N_j denotes the number of points in cluster \mathcal{P}'_j while N_{clust} is the number of clusters.

Notice that, being the recognition step performed independently w.r.t. the primitive type, the task is *embarrassingly parallel*. The same applies to the clusters of points obtained in the same iteration.

Finally, agglomerative hierarchical clustering approaches require, in their naive implementation, $O(N_{\text{seg}}^3)$ operations, where N_{seg} denotes the number of segments in the output segmentation; see, for example, [43]. When it comes to complete-linkage clustering, one can consider more efficient implementations, such as the one proposed in [44], which costs $O(N_{\text{seg}}^2)$. Although the dissimilarity-matrix assembly costs $O(N_{\text{seg}}^2)$, one may note that each entry is computed independently; again, the task is therefore *embarrassingly parallel*.

5.4.3 Experimental results

5.4.3.1 Simple geometric primitives

As a sanity check, we first apply our method to two models from [91], which were selected because containing all simple geometric primitives – plane, sphere, cylinder, cone and torus – and because of the availability of ground truth. A first example is provided in Figure 5.14. The point cloud, corresponding to the set of vertices of the original triangle mesh, is decomposed in 8 surface segments: 5 cylinders, 2 planes and 1 sphere. The high accuracy of our method is proved by comparing the parameters from the HT with those in the database. In the second example, presented in Figure 5.15, the corresponding point cloud is subdivided into 9 surface primitives: 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.

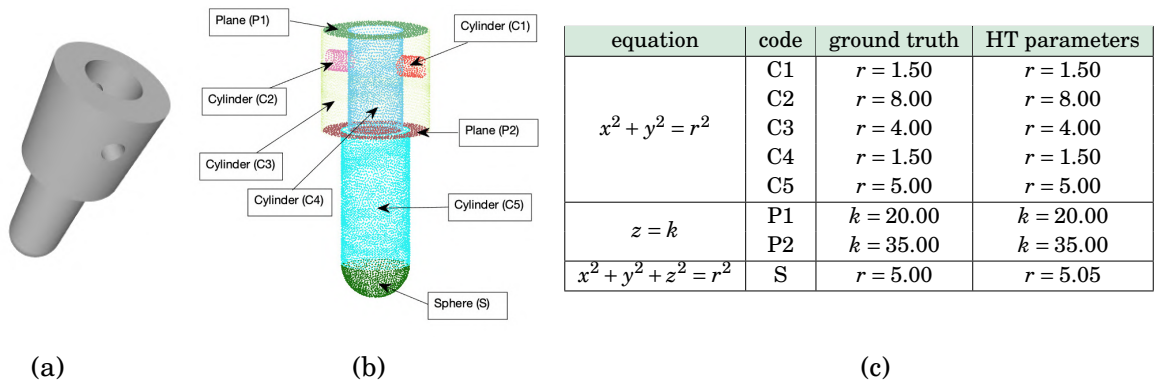


Figure 5.14: In (a) a mechanical CAD model from the benchmark in [91]; in (b) the vertices of its triangle mesh decomposed into 8 surface segments; in (c), for each primitive, the HT parameters are compared with those provided by the database

Figure 5.16(a-b) shows point clouds that can be segmented into complete geometric primitives without the need for the clustering strategy. The first example, shown in Figure 5.16(a), consists of 11 segments – 3 cylinders and 8 planes – and it highlights the robustness of our method when it

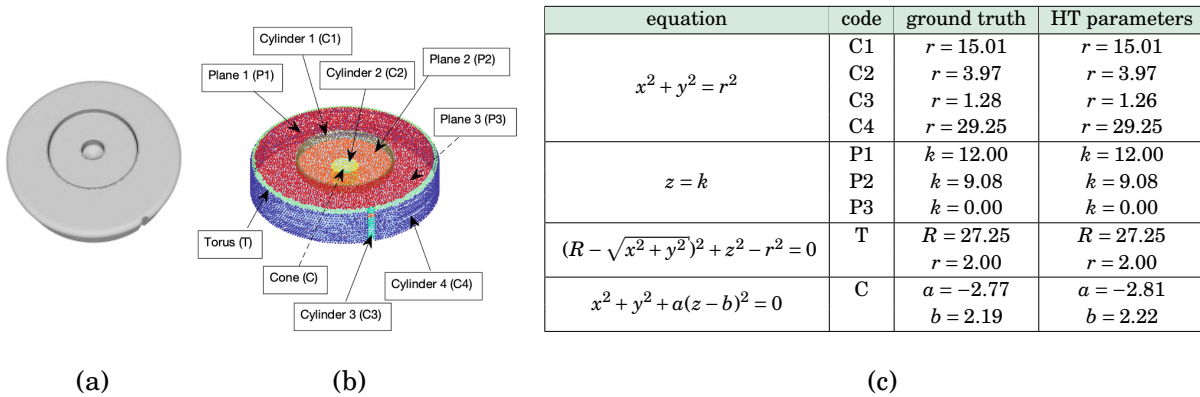


Figure 5.15: In (a) a mechanical CAD model from the benchmark in [91]; in (b) the vertices of its triangle mesh decomposed into 9 surface segments; in (c), for each primitive, the HT parameters are compared with those provided by the database

comes to detecting intersecting cylinders, here arranged similarly to a Steinmetz solid. Complete geometric primitives, each of which is represented by a specific colour, are automatically detected. Another point cloud, displayed in Figure 5.16(b), 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.

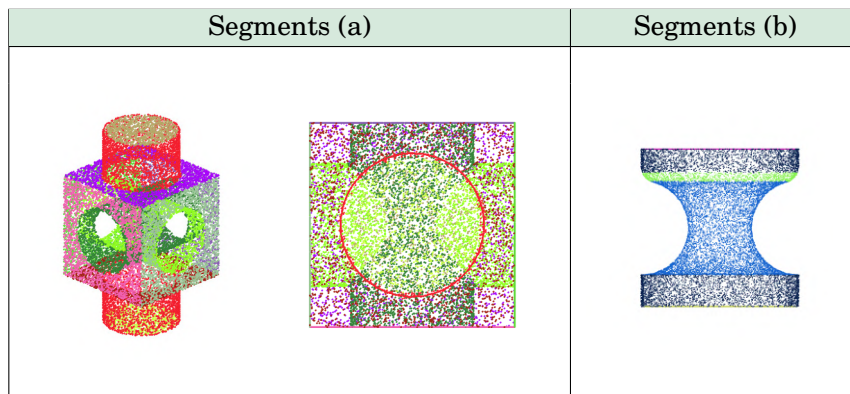


Figure 5.16: Recognition of CAD point clouds containing only simple geometric primitives. The identification of maximal segments does not require, in these cases, the application of any clustering algorithm.

Figures (5.17-5.20) show point clouds wherein the segments produced by the HT approach are post-processed by the hierarchical clustering. In Figure 5.17(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 5.17(a)), we are able to correctly identify repeating primitives, here in the form of circular cylinders of equal radii. Figure 5.17(c) highlights the similarities identified by clustering cylinders: 4 in light blue, 3 in

red, 2 in purple and 2 in yellow.

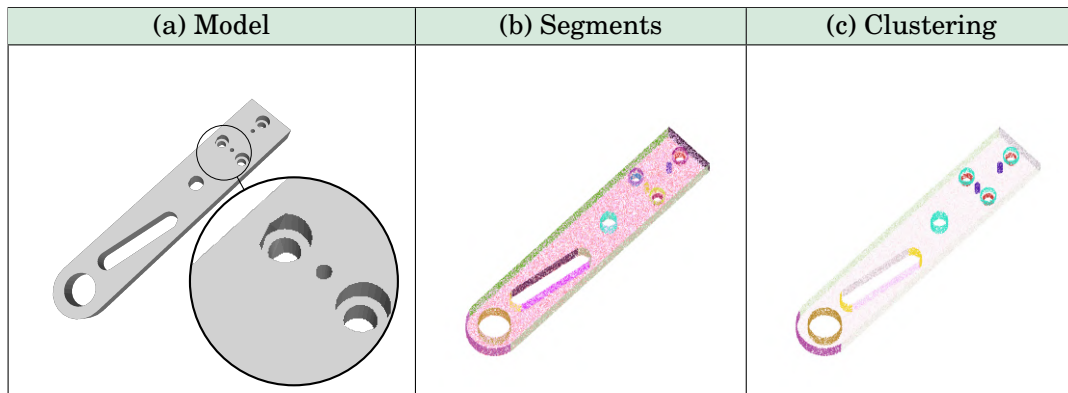


Figure 5.17: A linkage arm. In (a), the original model is shown, together with a magnification revealing some imperfections. The segments identified by the HT are shown in (b) in different colours. (c) draws attention to cylinders, among which we can recognise segments lying on the same primitive, up to a translation.

Figure 5.18 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 5.18(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 5.18(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 5.18(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 5.19(a). Here, the total number of extracted geometric primitives is 68: 19 cylinders; 2 cones; 47 planes, 5 of which are axis-aligned. The result is presented in Figure 5.19(b). As highlighted by the original model, in this prototypical version of the NuGear component, 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. Clustering identifies here a similarity between the 12 cylindrical holes – up to translations – and between 2 external cylinders, while the surface segments identified by non-axis-aligned planes are not clustered in pairs; this is because the tooth inclination prevents any alignment. Figure 5.19(c) shows this result, colouring the holes in black and the external cylinders in yellow.

Finally, Figure 5.20(a) shows another mechanical part. The HT decomposition of the input point cloud consists of 50 surface segments, all of which are tori, cylinders and planes, see Figure 5.20 (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 5.20(c). Finally, 2 tori

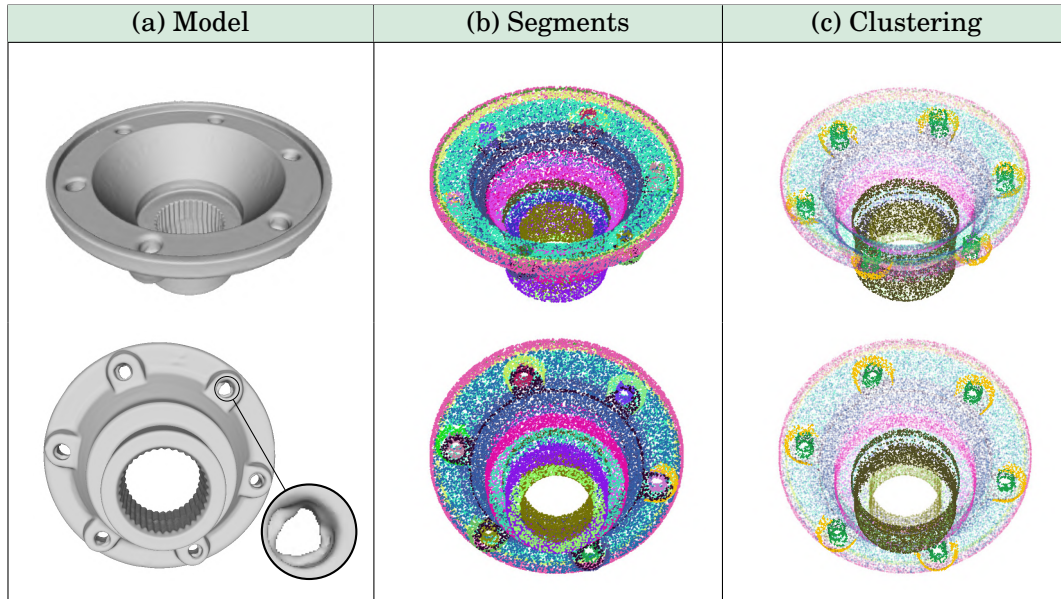


Figure 5.18: A carter. In (a), the original model is shown. The surface segments found by means of the HT approach are depicted in (b), while (c) shows the result of primitive clustering when one is interested in identifying the same primitive up to a translational transformation. Different rows correspond to different points of view.

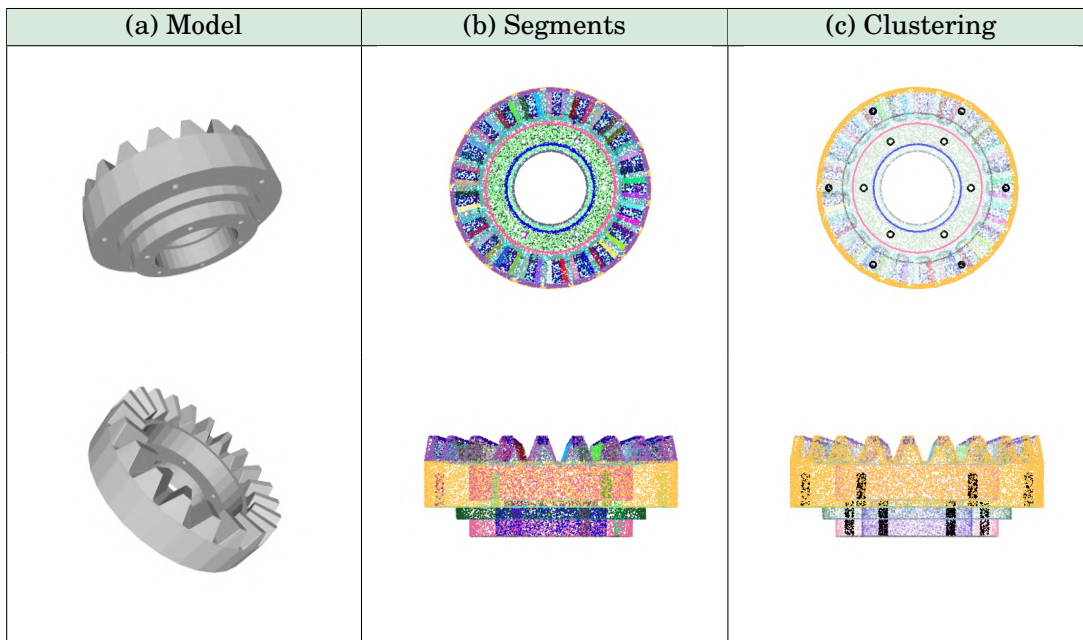


Figure 5.19: A prototype of the NuGear component, courtesy of STAM S.r.l. (Genoa, Italy). The original model is shown in (a). The decomposition in clusters of points produced by the HT approach is given in (b). The output of the additional clustering procedure is shown in (c), it highlights the similarity between 12 cylindrical holes (in black) and between two cylinders (in yellow).

are clustered together, because they lie on the same surface primitive up to roto-translations.

Table 5.5 summarises the characteristics of each point cloud processed in this section and the mean fitting error for all the simple primitives recognised on it.

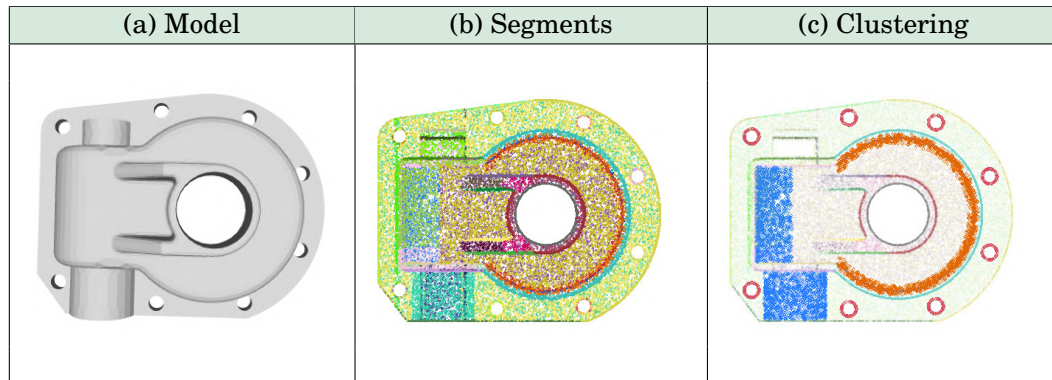


Figure 5.20: A mechanical part. In (a) the original model is shown, while in (b) the decomposition of the corresponding point cloud into segments produced by the HT. In (c) the result of the clustering procedure: 8 cylindrical holes, in red, have a high similarity, up to translations; the same applies for 2 cylindrical segments, in blue; 2 tori, in orange, identify the same primitive, up to a roto-translation.

Table 5.5: Statistics of the MFES for all models of Section 5.4.3.1. Being the MFE normalized by definition, we can conclude that the maximum error for the fitting of the simple primitives is 4.48%, which corresponds to the noisy holes in the carter of Figure 5.18.

Model	# points	# segs	$\min(E_i)$	$\text{mean}(E_i)$	$\max(E_i)$
Fig. 5.14	15,216	8	0.0006	0.0021	0.0046
Fig. 5.15	15,022	9	0.0008	0.0029	0.0051
Fig. 5.16(a)	25,000	11	0.0009	0.0024	0.0056
Fig. 5.16(b)	25,000	5	0.0004	0.0031	0.0059
Fig. 5.17(b)	50,000	20	0.0004	0.0098	0.0300
Fig. 5.18(b)	50,000	28	0.0013	0.0107	0.0448
Fig. 5.19(b)	50,000	68	0.0006	0.0053	0.0178
Fig. 5.20(b)	50,000	50	0.0008	0.0035	0.0057

5.4.3.2 Complex geometric primitives

As anticipated, our method is able to recognise other primitives in addition to the simple ones shown in Section 5.4.3.1. Here we show some of the complex primitives identified in our experiments.

The first example – shown in Figure 5.21(a) – contains an ellipsoid, which is easily recognised by our method at the price of an additional parameter in the parameter space; additionally, the point cloud can be partitioned into 4 planes, 1 torus and 1 cylinder. In the point cloud from Figure 5.21(b), we are able to identify correctly the yellow segment as a surface of revolution from Table

5.1(c); the remaining points are segmented into 2 cylinders and 2 planes. In the point cloud shown in Figure 5.21(c), we recognise the gold part as a generalised cone, while the blue and the green segments are fitted with generalised cylinders: all of the three have the same directrix, a 5-convexity curve. This last point cloud has been segmented into 7 clusters.

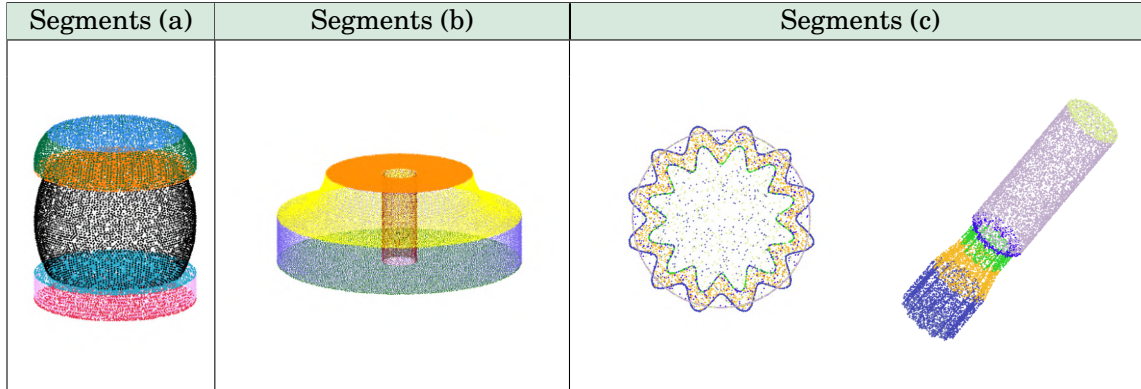


Figure 5.21: Recognition of complex geometric primitives in CAD point clouds. Their identification does not require, in these cases, the application of any clustering algorithm.

Figure 5.22(a) displays a mechanical part which can be accurately described by combining a portion of a helical surface, as introduced in Table 5.1(d), with a pair of planes and a pair of a convex combination of helices, see Table 5.1(e). The result is a segmentation of the point cloud into 6 primitives, Figure 5.22(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 5.22(c).

Table 5.6 provides the main characteristics of each point cloud processed in this section and the mean fitting error for all the simple and complex primitives recognised on it.

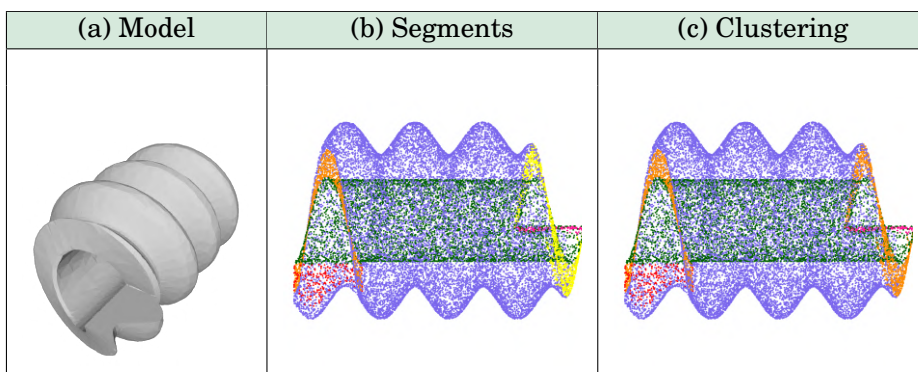


Figure 5.22: A screw-like part. The original model, (a), is sampled. The surface primitives detected via HT are shown in (b) in different colours: a helical surface (in purple), two planes (in red and magenta), and two helical strips (in orange and yellow). Although no pair of them lies on the same parametrised surface, the 2 helical strips have the same equation up to a translation, as shown in (c) (both in orange).

Table 5.6: Statistics of the MFES for all models of Section 5.4.3.2. Since the MFE is normalized by definition, we can conclude that the maximum error for the fitting of the simple primitives is 1,54%, which corresponds to the helical surface of Figure 5.22.

Model	# points	# segs	$\min(E_i)$	$\text{mean}(E_i)$	$\max(E_i)$
Fig. 5.21(a)	25,524	7	0.0009	0.0042	0.0063
Fig. 5.21(b)	67,777	5	0.0006	0.0031	0.0071
Fig. 5.21(c)	25,000	7	0.0020	0.0053	0.0081
Fig. 5.22(b)	25,000	6	0.0033	0.0086	0.0154

5.4.3.3 Robustness of the pipeline to perturbed data

The use of the HT naturally leads to a robust method for the recognition of mathematical surfaces, as suggested in the examples of Figures 5.17 and 5.18 – which were characterized by spurious parts. In the point cloud of Figure 5.23, 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 5.23(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” inscription (see Figure 5.23(b)). The application of the hierarchical clustering allows us 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 and bottom) and on the lateral holes do not prevent the clustering technique from appropriately aggregating the corresponding segments, correctly dealing with rotations and reflections. However not everything is recognised: the black dots in Figure 5.23(b) correspond to points that are not fitted by any of the geometric primitives at our disposal, as they originate from irregular elements that act as a connection between better defined segments. We label such points as “unsegmented” because of the high mean fitting error.

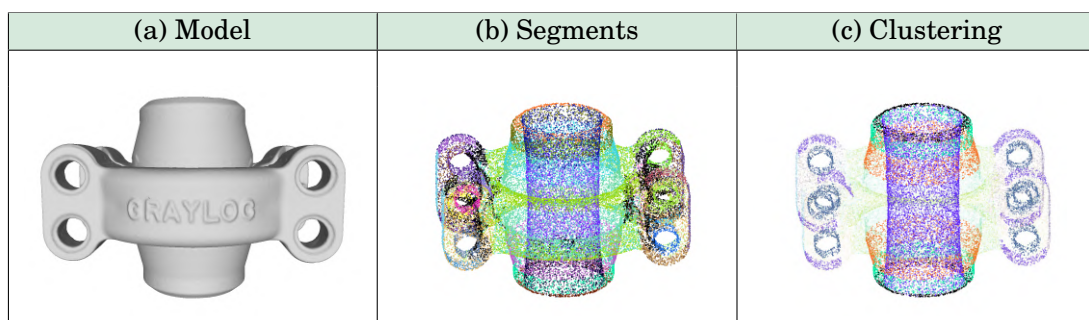


Figure 5.23: A clamp connector. In (a) the original model. In (b) the decomposition of the corresponding point cloud into 38 segments is provided by the HT procedure. In (c), the final grouping is obtained by clustering, consisting of 6 groups of primitives (here, singletons of segments are transparent).

Further proof of the robustness of our method applied to raw data is presented in Figure 5.24 and quantitatively analysed in Table 5.7. In this example, we perturb the point cloud of Figure 5.16(b) by adding zero-mean Gaussian noise of standard deviation: 0.01, 0.05, 0.10 and 0.20. 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 5.7: the parameters obtained in the original point cloud are there compared with those found in the perturbed point clouds.

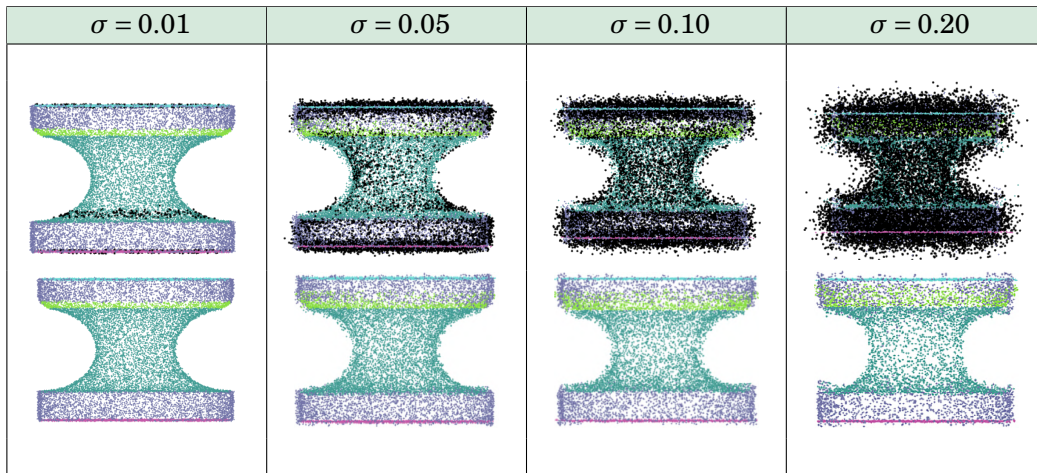


Figure 5.24: The point cloud in Figure 5.16(b) is perturbed by adding zero-mean Gaussian noise of standard deviation: 0.01, 0.05, 0.10 and 0.20. The first row superimposes the points identified as noise (in black) to the final segments found by our method; the second row depicts the points that fit the primitives found and provides a denoised segmentation.

Table 5.7: Parameter comparison between the original point cloud from Figure 5.16(b) and the perturbed versions from Figure 5.24.

Segment	Original	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.10$	$\sigma = 0.20$
Plane 1	$k = -1.28$	$k = -1.28$	$k = -1.29$	$k = -1.28$	$k = -1.31$
Plane 2	$k = 1.28$	$k = 1.28$	$k = 1.28$	$k = 1.28$	$k = 1.26$
Cylinder	$r = 1.80$	$r = 1.79$	$r = 1.78$	$r = 1.79$	$r = 1.78$
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	$R = 1.05$ $r = 0.79$	$R = 1.09$ $r = 0.78$	$R = 1.02$ $r = 0.78$	$R = 1.17$ $r = 0.74$	$R = 1.13$ $r = 0.80$

5.4.4 Comparative analysis

Figure 5.25 compares our method with the RANSAC-based method introduced in [144], the patch aggregation approach in [96] and two recent deep learning architectures: ParSeNet [148] and HPNet [160]. In this comparison, we have focused on models that merely present simple primitives, to show that even on these our approach gives a great performance; on the other hand, the capability to handle more complex primitives is undoubtedly an added value of our method. Similarly to [96], we use different colours to represent different primitive typologies: red for planes, green for cylinders, blue for cones, black for tori, pink for (open and closed) B-splines and yellow for unsegmented. For a simple model like the block of Figure 5.25(a), all methods provide decent decompositions, although RANSAC misclassifies some primitives while the deep learning frameworks run into problems along sharp edges – because of an erroneous estimation of the surface normals. For the remaining more complex models, our method outperforms the competitors. In Figure 5.25(b), our approach is the only one capable of correctly identifying portions of tori, misclassified by Le and Duan [96] and partly unsegmented by RANSAC; ParSeNet and HPNet are able to reveal the presence of tori, but the resulting segmentation is unreliable along sharp edges. Figures 5.25(c-d) show a RANSAC tendency to over-segment and misclassify complex models. While Le and Duan [96] obtain considerably improved results, their algorithm misses a thin cylinder (see Figure 5.25(c)) and all the tori in both models. Being these two objects acquired by low-quality scanners, the corresponding point clouds (and meshes) are affected by point cloud artefacts which, in turn, leads to erroneous normal estimates. The resulting segmentations are unreliable and are mostly associated with spline segments; note that the two architectures were trained on the ABC dataset, which does not take into account the presence of noise or other types of perturbation in the data. Due to the lack of freely-downloadable implementations for some methods, it is not possible to present this comparison other than qualitatively.

To offer a quantitative analysis of the robustness of our pipeline, we compare its performance with that of three state-of-the-art methods whose implementation was made freely available by the authors: a direct method approach on a primitive-growing (PG) framework [125], adapted to handle point clouds as described in [140], and the two learning-based methods from the previous comparison: ParSeNet (PN) and HPNet (HN). We conduct the study over the whole Fit4CAD benchmark [140], which contains CAD point clouds defined by simple geometric primitives. In addition, Fit4CAD contains a selection of meaningful and validated models from [91], converted from meshes to point clouds. Figure 5.26 summarises the performance of each method over the test set with respect to the following classification measures: Sørensen-Dice index (DSC), Positive Predicted Value (PPV), True Positive Rate (TPR), Negative Predicted Value (NPV), True Negative Rate (TNR) and accuracy (ACC). Given a point cloud, the benchmark defines these measures by comparing each point cloud segment in the ground truth to the most overlapping segment returned by a segmentation method, and then by averaging over all segments contained in that

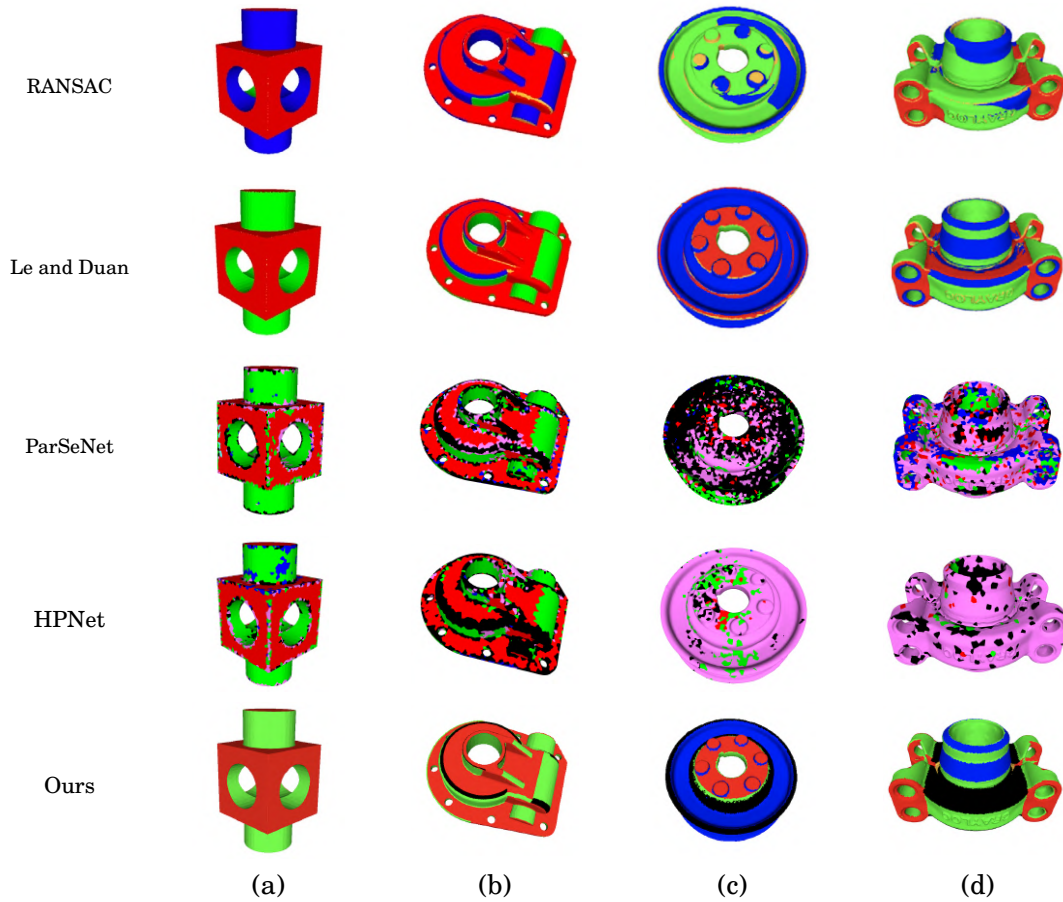


Figure 5.25: Primitive type recognition: a comparison between our approach, a RANSAC-based segmentation [144], and the method in [96]. Different colours correspond to different primitive types: planes (red), cylinders (green), cones (blue), tori (black), splines (pink), and unsegmented (yellow).

point cloud. Note that, given a point cloud segment in the ground truth, the points inside that segment are the positives while the points outside that segment are the negatives. We arrive at the following conclusions:

- Learning-based methods show remarkably high TNRs – said otherwise, the points they predict as being negative are almost always true negatives. On the other hand, they are more penalized by NPV: while it is ParSeNet that exhibits the highest variability and the lowest quartiles, both methods are seriously affected by outliers – with some point clouds having this score below 0.2. A low NPV means that quite a few points predicted as negative are false negatives (e.g., common in heavily over-segmented models).
- Our approach performs significantly better than the competitors in terms of TPR, i.e., it has similar proportions of correct (positive) predictions among positives. When it comes to the PPV, differences are more modest.

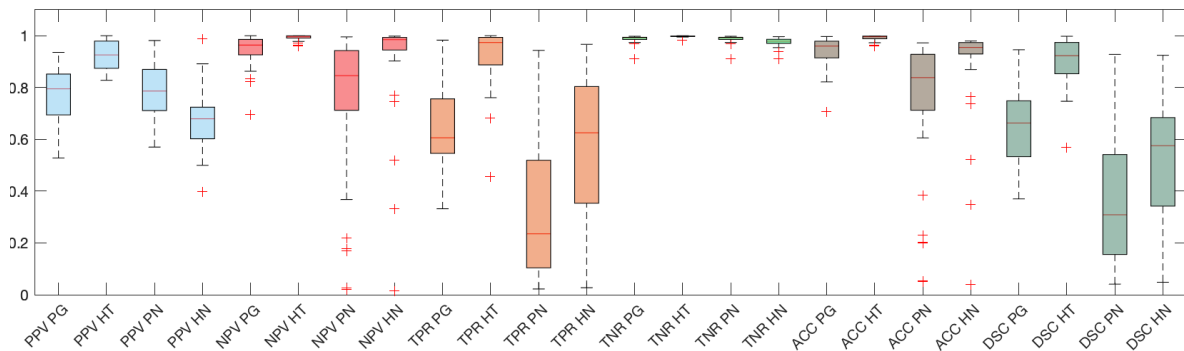


Figure 5.26: Comparison of our approach (HT) with other three methods: a primitive growing approach (PG), ParSeNet (PN) and HPNet (HN). The analysis is performed over the Fit4CAD benchmark [140]

- In terms of accuracy, the four methods reach high scores, with the direct methods being the less prone to outliers: however, this metric is not completely reliable as this is a naturally unbalanced binary classification problem. A more reliable measure is provided by the Sørensen-Dice index (DSC). Our method visibly outperforms the competitors in terms of the DSC. Intuitively, the higher the DSC, the more accurate segments returned by a segmentation method are – with respect to the most overlapping segment in the ground truth; to put it differently, DSC penalises greatly both under- and over-segmentation.

When it comes to execution time, our Hough-based method and the primitive-growing approach were run on a desktop PC equipped with an Intel Core i9 processor (at 3.6 GHz) and a Windows 10 operating system. The average execution time, per model, are 286.0 seconds for the PG-method and 50.7 seconds for our pipeline. ParSeNet and HPNet were run on Google Colab pro equipped with NVIDIA-SMI 460.32.03 and CUDA 11.2. Regarding the average execution times, we have 5 seconds for ParSeNet and 257.1 seconds for HPNet (when the preprocessing of normals is applied).

5.5 Concluding remarks

The methods and the results presented in this chapter have been investigated during the third year of my PhD. Specifically, the approach shown in Section 5.3.1 is described in [128] and our implementation is available as a GitHub repository at https://github.com/chiararomanengo/fitting_geometric_primitives, while its extension exhibited in Section 5.4.1 is submitted for publication.

In the first part of this chapter, we describe a method able to provide a parametric representation and several parameters (centres, axes, vertices, etc.) that uniquely define it, thus feeding the clustering algorithm with a reliable segment description. As the method is applied to segmented

point clouds, the better the presegmentation technique, the better the results obtained. In any case, it helps to improve the model in case of over-segmentation. Since the method is based on an aggregation technique, in the case of under-segmentation, it would be necessary to integrate it with an adaptive splitting strategy.

Thanks to the devised segment preprocessing technique, we can fit geometric primitives in a standard form (thus limiting the number of parameters needed by the HT) as well as provide an estimate of the true solution (thus limiting the search for the optimal solution to a specific region which, in turn, allows us to solve the problem of unboundedness of the parameter space). Our experiments confirm the robustness of the HT-based method to deal with various types of point cloud artefacts.

In the second part of this chapter, we address the problem of the recognition of simple and complex primitives in a point cloud, opportunely extending the family of geometric primitives to which the HT technique can be applied. Thanks to an opportune preprocessing of the point cloud and its sub-parts, we can limit the number of parameters that are necessary to represent the primitive, thus reducing the computational complexity of the HT computation. In addition, the explicit extraction of position-independent primitive parameters and the use of a hierarchical clustering strategy permits the identification of maximal and compound primitives, thus reducing the output over-segmentation. Indeed, differently from spline-based primitives, our strategy is suited for primitive similarity reasoning and permits the finding of maximal primitives.

Although learning methods have performed very well in recent years, when models present complex combinations of primitives (such as the examples in Figure 5.25(c,d)), direct methods perform even better, and our method is very competitive. In addition, our method has been validated on a whole benchmark and, compared with the others, it turns out to be the best.

5.6 Related publications

- A. Raffo, C. Romanengo, B. Falcidieno, S. Biasotti, *Fitting and recognition of geometric primitives in segmented 3D point clouds using a localized voting procedure*, Computer Aided Geometric Design (2022), vol. 64, pp. 102–123.
- C. Romanengo, A. Raffo, S. Biasotti, B. Falcidieno, *Recognising geometric primitives in 3D point clouds of mechanical CAD objects*, submitted.

CONCLUSIONS

In this thesis, we faced the problem of recognising curves and surfaces on 3D digital models providing their mathematical representation. However, when 3D models are acquired by scanning real objects, the resulting geometry does not explicitly encode these curves and surfaces, especially when it is affected by noise, due to measurement uncertainty and sampling resolution, or missing some parts, due to occlusions during the acquisition or other factors. In particular, in applications like the digitisation of archaeological artefacts, these objects might be damaged, and then curves are partially missing. In our recognition approach, we use the generalised Hough transforms that identify the best fitting curve or surface in a dictionary containing different families. The HT answers to both the need for robustness to noise and data incompleteness, and it benefits from the flexibility of the template curve or surface.

At first, in Chapter 2 we focused on the recognition of feature curves on 3D shapes that can be projected into the plane by using a rich dictionary of families of plane curves. This technique shows its effectiveness in some application contexts, such as the recognition of geometric patterns in archaeological finds, the analysis of brand logos, and the interpretation of artistic elements. Since the parameters and the equation of the recognised feature curves are provided, it is possible to detect the similarity of decorations and explore their geometry. Indeed, in the case of archaeological fragments, the outcome of this method can be used to support the automatic annotation of digital models and to compare the decorations present on the surface of different fragments. In the case of artistic and design motifs, the curves in parametric form can be used as an input for the interactive visualisation and manipulation of the symbols through a multi-touch smart table. However, this method needs to know in advance which kind of patterns to look for. This suggestion can be provided by the user through a template or by choosing in a curve catalogue the one or the ones most similar to what is sought. In the archaeological domain, this

input can come directly from the expert, through a template or a drawing, or from the technical documentation associated with the find.

To overcome the necessity of projecting feature curves on the plane, in Chapter 3 we extend the previous method to the recognition of space curves. In this vein, we propose a template of space curves dividing them into two classes. The first type is composed of space curves equipped with a known representation. Since in the literature the dictionary of space curves is quite limited and there is a large variety of plane curves rather than space curves, we exploit this richness by introducing as a second type of space curve the intersection of a paraboloid with a cylinder that has a plane curve as directrix. This second family improves the approximation by using a surface primitive as a projecting surface. The use of a paraboloid provides a better fitting of the set of feature points when the surface is curved, with respect to the previous method that approximates the curves projecting them onto the regression plane. In this case, in addition to archaeological fragments, the method has been also tested on digital models coming from real application contexts, such as CAD objects. The mathematical representation of curves provided by our method can drive model simplification and can be exploited to add curve elements directly into the mesh.

However, also this method needs to know a priori what family of curves has to be used for the recognition. This limitation has been overcome by implementing a method for piece-wise curve approximation using specific families of polynomial curves of degree 3 or 4, explained in Chapter 4. The proposed method can approximate both planar and spatial profiles, by considering one or two projections of the feature points, respectively. The result is a curve of continuity G^1 on each projection, except for some specific points, in which the continuity is C^0 . In this context, we show as an example of an application a feature-preserving point cloud simplification, and resampling.

Regarding the recognition of surfaces (discussed in Chapter 5), we introduced a dictionary made of both simple (planes, spheres, cylinders, cones, and tori) and complex geometric primitives (general cylinder, general cones, surfaces of revolution, helical surfaces, and convex combination of curves). Then, we describe two strategies. The first approach is applied to a pre-segmented point cloud and provides the geometric descriptors that uniquely identify each segment. The primitive fitting is obtained by considering the mathematical representation of families in standard form, thus limiting the number of parameters needed by the HT. These descriptors are exploited by a clustering method to find different relations among segments, improving the model in the case of over-segmentation. The second approach takes in input a whole point cloud representing a CAD object, that can contain multiple instances of the same primitive, and provides a segmentation together with the mathematical representations of segments. The output over-segmentation can be reduced by explicitly extracting position-independent primitive parameters and using a hierarchical clustering approach to discover maximal and compound primitives. In contrast to spline-based primitives, our approach is suitable for reasoning about primitive similarity and enables the search for maximal primitives (i.e., segments composed also of non-adjacent parts,

belonging to the same primitive).

Parallel to this, we have developed datasets and metrics to evaluate approaches for identifying simple geometric primitives in 3D point clouds representing CAD objects as well as in 3D point clouds representing simple geometric primitives with various types of perturbations. The benchmarks and the comparisons among our and existing methods are provided in Appendix A.

6.1 Ongoing activities and future directions

Recognition methods based on the HT require to priori know the family of curves or surfaces to be used. In Chapter 4 we propose the first solution to overcome this limitation in the case of space curves, by providing a piece-wise polynomial curve approximation. It is a good answer to the problem of fitting, but it loses the uniqueness of the recognition. Then, we would like to find a solution that automatically selects the family of curves a priori. The second difficulty to be addressed is the search for the standard form of the selected family and the parameters estimate that allows the construction of the parameter space. In this context, we are combining our method with a learning-based approach able to select automatically the template, through a classification step, and estimate the parameters, through a regression step. We are doing some tests about this in the context of plane curves. Specifically, we first created a dataset composed of images obtained from the parametric equations of the families of curves available in our dictionary. These images present differently shaped dots with different thicknesses and with various sampling densities and values of uniform noise. Then, we use a traditional pre-trained Convolutional Neural Networks (CNNs) architecture [119]. One of the major advantages of CNNs is their ability to "autonomously" learn the convolutional kernels necessary for the extraction of features in an image dataset (therefore no prior feature engineering process is necessary). The second great advantage of modern CNNs is the possibility of exploiting the so-called transfer learning effect, i.e., being able to reuse - up to a certain degree of performance - old kernels pre-trained on "generic" datasets composed of natural images also on very specific datasets composed of synthetic images such as those we generate. Being able to take advantage of transfer learning allows us to train CNNs much faster and with smaller datasets than what would be normally necessary without transfer learning.

All of our experiments were performed on a machine equipped with $2 \times$ Nvidia Tesla V100 GPUs with 32GB of RAM each. As for the software, all the training phases took place inside Jupyter Notebooks using Python 3.8 and the popular Fast.ai library [82], a handy abstraction layer on top of Pytorch that allows training neural networks in a sufficiently simple and fast way.

In the following, we describe the two steps that allow us to automatically select the family of curves necessary for the recognition (classification) and estimate a geometric parameter (regression).

- **Classification step.** To begin exploring the problem, we generated a toy dataset of 990

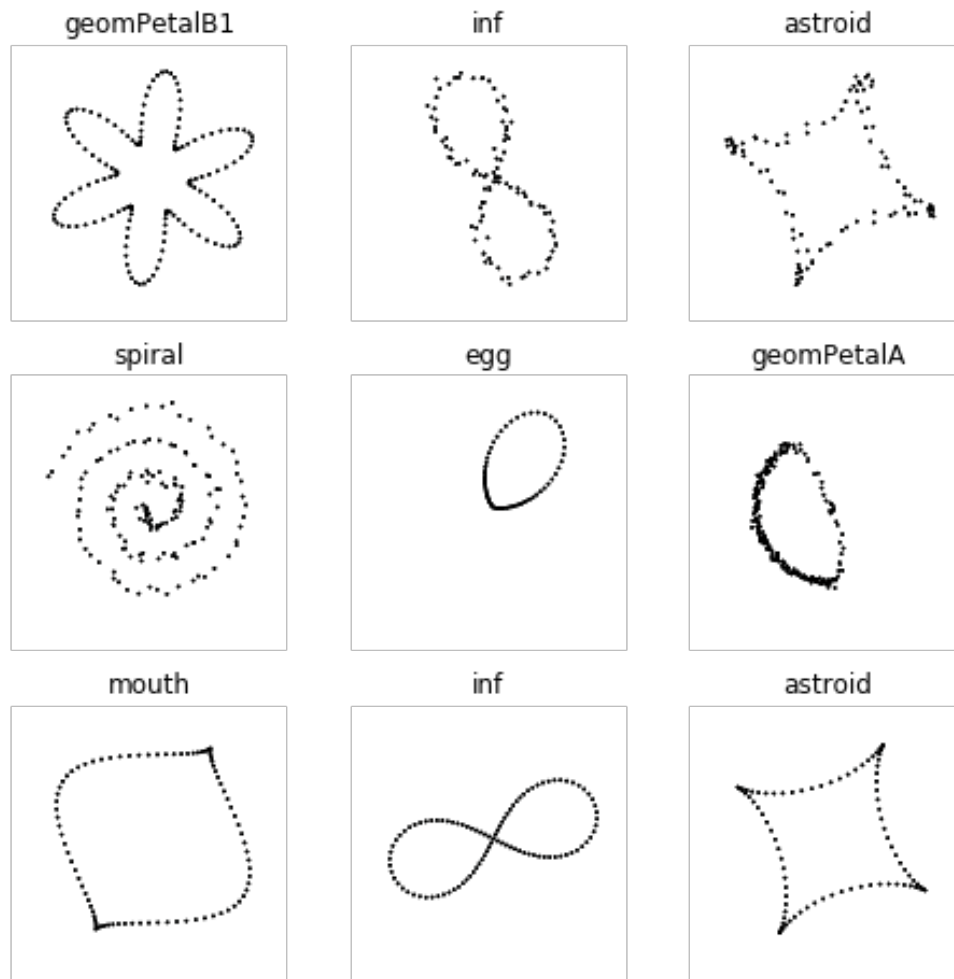


Figure 6.1: Examples of images present in our dataset.

images divided into 9 classes (astroid, citrus, egg, geomPetalA, geomPetalB1, geomPetalB2, inf, mouth, spiral). Figure 6.1 shows an example of batch given as input to the neural network. Already in this phase we started to insert a minimum of noise in the images, both to act as "offline data augmentation" and to start testing the generalization capacity of the network.

Given the very simple task and the toy dataset created, the architecture we chose was a standard ResNet-34 [76] available in the Fast.ai library. The training took place on images with a size of 256×256 px with a learning rate (LR) of $10e - 3$ and a batch size of 64 images per batch. The training ended with a Fast.ai callback after the validation loss did not improve over 10 epochs. As expected, the network is perfectly capable of learning to classify images in the dataset, even reaching 100% accuracy, precision, recall and an F-Score of 1 on this toy dataset (see Figure 6.2(a)). Finally, the confusion matrix confirms what is reported

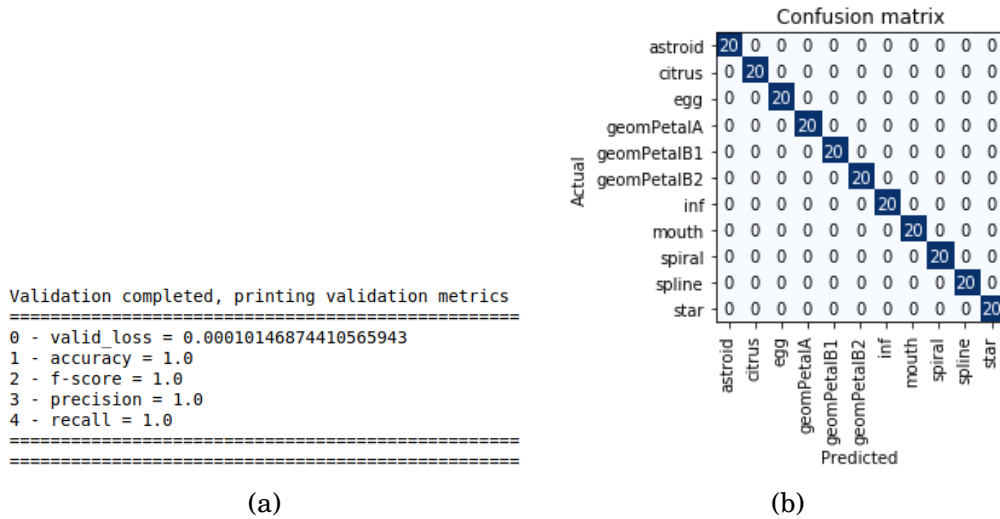


Figure 6.2: In (a) the validation metrics and in (b) the confusion matrix associated to the classification problem.

by the metrics (see Figure 6.2(b)).

- Regression step.** Having had full success with the first toy dataset classification experiment, we decided to expand the dataset and start working on the parameter estimation of curves. As a starting point, we estimate only one parameter: the rotation angle, which translated into the language of Deep Learning is called image regression. The important difference compared to regression analysis in statistics is that the independent variables are not two- or three-dimensional points but are the feature representation of the images after they have been processed and condensed into a feature vector in the forward phase of the neural network.

To address this problem, we have generated more than 190k images containing curves belonging to 6 classes (astroid, citrus, egg, lemniscate, geomPetalA, geomPetalB1). To perform offline data augmentation we used an initial set of simple transformations (e.g. by varying the type of marker that makes up the curve, its size and the distance between the marker symbols) but without injecting further noise in this phase. The training took place on images with a size of 256×256 px with a batch size of 32 images per batch. This time the training took place in two phases: in the first phase, with a learning rate (LR) of $10e - 3$, only the weights of the output layer of the network were trained (thus exploiting the transfer learning from ImageNet [45]). In the second phase, all the weights of the neural network were trained with LR ranging from $10e - 6$ to $10e - 7$. Again, the end of training triggered at epoch 18 of the “unfreeze” phase due to a callback from Fast.ai after the validation loss has not improved in 10 epochs (see Figure 6.3).

Figure 6.4 visually shows some estimation results of the rotation angle, highlighting in red

```

Validation completed, printing validation metrics
=====
0 - valid_loss = 0.004581906832754612
1 - _rmse = 0.06768976897001266
2 - mae = 0.05230538174510002
3 - r2_score = 0.9982571374546618
=====

```

Figure 6.3: The validation metrics associated with the regression problem.

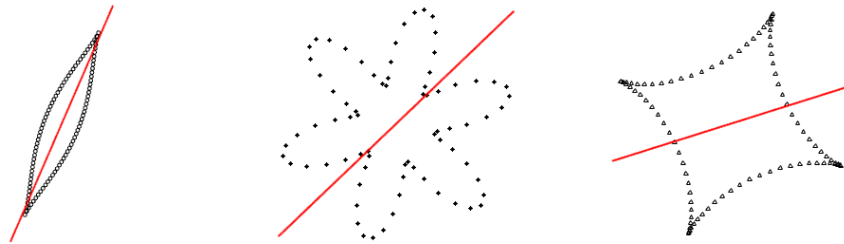


Figure 6.4: Three examples of images in our dataset and in red the line that represent the estimation of the rotation angle.

the line representing it. Although these results are very encouraging, much work remains before we can obtain a model capable of predicting multiple parameters beyond the angle of rotation (e.g. position, size, etc.), the class of the object and that is very robust to any kind of noise, so not just the simple "synthetic" noise we have used so far. Furthermore, each additional parameter we add as an estimate is one less degree of freedom that we have to use in the data augmentation phase, so it will be necessary to get very creative to produce sufficiently varied datasets in the future.

The approach just described automatically selects the family of curves to be used for the recognition and to estimate a priori the curve parameters. Although we have extended the number of families of curves and surfaces that can be used with the HT, we still need to limit the number of parameters in curve and surface representations because the computational cost of HT increases as the number of parameters involved grows up. To reduce the number of parameters, in this thesis we exploited several strategies such as reparameterization techniques (e.g., the change of variables) or the search for a standard form of the selected family of curves or surfaces. In the future, we plan to investigate further possibilities and other techniques for parameter space reduction, for instance by exploiting geometric properties of curves or geometric invariants such as symmetries, centres of rotation, or properties of the projective space.

In Section 3.6, we describe a method able to insert the recognised space curves into the model to drive model simplification and to create a hybrid mesh representation. Analogously,

we envisage the application to model resampling and simplification for the piece-wise curve approximation presented in Chapter 4, too. On the wave of those achievements, we plan to extend our approach to the insertion of both curve and surface primitives to create a hybrid model with curvilinear and surface profiles. This process would permit the improvement of model quality and fair appearance. The use of exact primitive representations would produce a faithful model simplification as well as repair a model with the completion of missing parts and be robust to noise. In our view, these hybrid models would mix linear and curvilinear edges and faces, and faces would not be necessarily triangular, thus allowing greater flexibility. This would extend the representation models traditionally used in a CAD modeller, such as [5, 67]. However, we think that hybrid meshes are particularly suitable for the emerging use of finite element methods for physical simulations in the context of isogeometric analysis [84]. In these cases, it is possible to have a better (or even exact) approximation of the geometry and, therefore, a better simulation over there.

Another possible field of interest is remote sensing, in particular the processing of the point clouds coming from the acquisition of large-scale outdoor/indoor scenes. This procedure is becoming popular for the creation of a digital model of an urban area or a complex building, falling into an area of research that goes by the name of Urban Intelligence. Urban Intelligence fosters the creation of a cyber-physical counterpart of all the city/building systems and sub-systems. Within this scenario, it is necessary to integrate heterogeneous data and predict the evolution of the model with the physical city, supporting the monitoring of the current status and predicting/anticipating possible future scenarios. In this context, the potential support of the work presented in this thesis goes in the provision of several tools for point cloud processing and annotation, thus contributing to the creation of the geometric layer of the city digital twin. Indeed, the more the geometry adheres to the real counterpart, the more accurate measures and simulations related to the urban space will be. As an example, in the civil industry, the use of Building information models (BIM) is becoming popular for its modularity, flexibility, and the possibility of making simulations over it. The 3D geometric representation of a building structure is the fundamental information under the BIM models [79, 142]. In this context, the most widely used approach for modelling already-existing structures is to create 3D geometric structures from point clouds that are acquired through laser scanners or photogrammetry. The fundamental principle behind the BIM representation is the recognition of structures (e.g., walls, pillars, staircases, etc.) that are shaped as basic geometric primitives, such as planes, spheres, and cylinders. To guarantee faithful calculations on the material required and effective numerical simulations, it is crucial to recognize these elements as accurately as possible and to annotate the characteristic parts for the model construction with precise measures. The issues that have to be addressed so that the methods presented in this thesis can be effectively applied to remote sensing and urban intelligence are the ability to deal with scene complexity since it might be composed of not isolated heterogeneous objects and could present also unexpected elements, and

CONCLUSIONS

an increase of the computational efficiency because it could be necessary to process hundred million and even billions of points.



BENCHMARKING ACTIVITIES

To develop a competitive and fair evaluation system and to identify new research areas, the creation of appropriate datasets and ground truth are essential building blocks. The fundamental purpose of creating a new benchmark is not unique; it can range from the requirement for a summary of the approaches capable of handling a certain task to identifying what is still lacking or requires additional development. Benchmarks have played a crucial role in the advancement of object recognition and other fields of computer vision and graphics. The challenges posed by these standard datasets have helped identify and overcome the deficiencies of existing approaches, and have led to great advances in the state of the art. Even the recent growth of interest in deep learning methods can be attributed to their success. In particular, the availability of such benchmarks helps the development of algorithms processing shape models, allowing a direct fair comparison of different approaches.

In our case, we wanted to compare the performance of our methods, in recognising geometric primitives in point clouds, with other algorithms. To do this, we have proposed a track for the international 3D SHape REtrieval Challenge (SHREC), whose general objective is to evaluate the effectiveness of 3D-shape retrieval algorithms. This track [139] was submitted to test recognition algorithms related to our study topic. Similarly, we created a benchmark [140], composed of dataset and performance measures, to test and compare our method, since the available datasets contain mostly general shape models and few real CAD artefacts, or lack ground truth or performance measures.

This appendix is divided into two sections: the first (Section A.1) regards the recognition and fitting of simple geometric primitives on point clouds representing planes, cylinders, spheres, cones and tori (or their subparts), to compare our approach outlined in Section 5.3.1 with literature methods; the second (Section A.2) concerns the fitting of geometric primitives in point cloud representing CAD objects to compare our method explained in Section 5.4.1 to another

direct algorithm.

For each proposed track, in this appendix we describe the benchmark, that is dataset and performance measures, and we briefly summarise the comparative analysis among the participant methods. Both benchmarks are currently available on GitHub and have been awarded the replicability stamp by the graphics Replicability Stamp Initiative.

A.1 The SHREC2022 track on fitting and recognising geometric primitives in point clouds

Since we needed to compare our method described in Section 5.3.1 with other state-of-the-art approaches, we have proposed a track in the SHape REtrieval Challenge SHREC (previously Contest) by creating an appropriate benchmark.

This SHREC track aims to evaluate the performance of automatic algorithms for fitting and recognising geometric primitives in point clouds. The goal is to identify, for each point cloud, its primitive type (i.e., planes, spheres, cylinders, cones and tori) and some geometric descriptors. Three tasks are considered in the evaluation phase:

- *Classification task.* Methods are evaluated based on their capability to identify, for each point cloud of the test set, its primitive type.
- *Recognition task.* Methods are studied in terms of their ability to recover the geometric descriptors (e.g, radii of a torus, vertex of a cone) of each segment.
- *Fitting task.* Regardless of the primitive type, the performance of the methods is studied with respect to the approximation of each point cloud.

A.1.1 Dataset and performance measures

The dataset was specifically designed to allow the training of data-driven methods, with a training set of 46000 primitives and a test set of 925 models. Nine types of perturbations were randomly applied to each type of primitive. To quantify the performance of automatic algorithms in classifying, recognising and fitting simple geometric primitives on point clouds under different conditions, we, therefore, consider a variety of classification and approximation measures. This analysis will be further specified according to the type of geometric primitive and point cloud artefacts.

The benchmark is available at <https://github.com/chiararomanengo/SHREC2022.git>.

A.1.1.1 Dataset

The dataset is composed of 46,925 three-dimensional segments represented as point clouds; it is divided into a training set and a test set that contain, respectively, 46,000 and 925 point clouds.

Each point cloud is provided in a TXT file listing one triplet per line.

The point clouds are generated by sampling classical surface primitives derived from constructive solid geometry (i.e., planes, cylinders, spheres, cones and tori), using the following procedure. First, point clouds are sampled by using parametric equations in their canonical form, i.e., centred at the origin of the coordinate axes and with the rotational axis aligned with the z -axis; the geometric quantities that define each primitive (i.e., amplitudes and radii, if any) are assigned randomly. Second, to obtain segments of different shapes, several cuts are enforced by exploiting random planes. Third, we randomly apply translations and/or rotations to recover primitives in their general position. Lastly, each point cloud is (potentially) processed so that it can be:

- *A0 - Clean.* No perturbation is applied.
- *A1 - Perturbed by uniform noise of different intensities.* The noise is obtained by sampling uniform distributions of the form $\mathcal{U}(-\frac{1}{n}, \frac{1}{n})$, being $3 \leq n \leq 20$ random, and adding such perturbations to a random percentage of the points.
- *A2 - Perturbed by Gaussian noise of different intensities.* The noise is obtained by sampling normal distributions among $\mathcal{N}(-\frac{1}{n}, \frac{4}{n^2})$, being $10 \leq n \leq 30$ random and adding such perturbations to a random percentage of the points.
- *A3 - Clean but affected by undersampling.* We randomly select a percentage of points to be removed.
- *A4 - Clean but affected by missing parts.* We randomly choose a point of the point cloud; then, we remove all points contained inside the sphere having the selected point as the centre and radius assigned randomly.
- *A5 - Perturbed by uniform noise of different intensities and undersampled.*
- *A6 - Perturbed by Gaussian noise of different intensities and undersampled.*
- *A7 - Perturbed by uniform noise of different intensities and affected by missing parts.*
- *A8 - Perturbed by Gaussian noise of different intensities and affected by missing parts.*
- *A9 - Clean but with local deformations.* We randomly select a point of the point cloud and we apply a bivariate Gaussian centred at the selected point with a random covariance matrix.

For the sake of brevity, we will often refer to these point cloud artefacts as *perturbation types*. For each perturbation type A0, ..., A8, training set and test set containing 5,000 and 100 point clouds, respectively; for perturbation type A9, training set and test set count 1,000 and 25 point clouds, respectively.

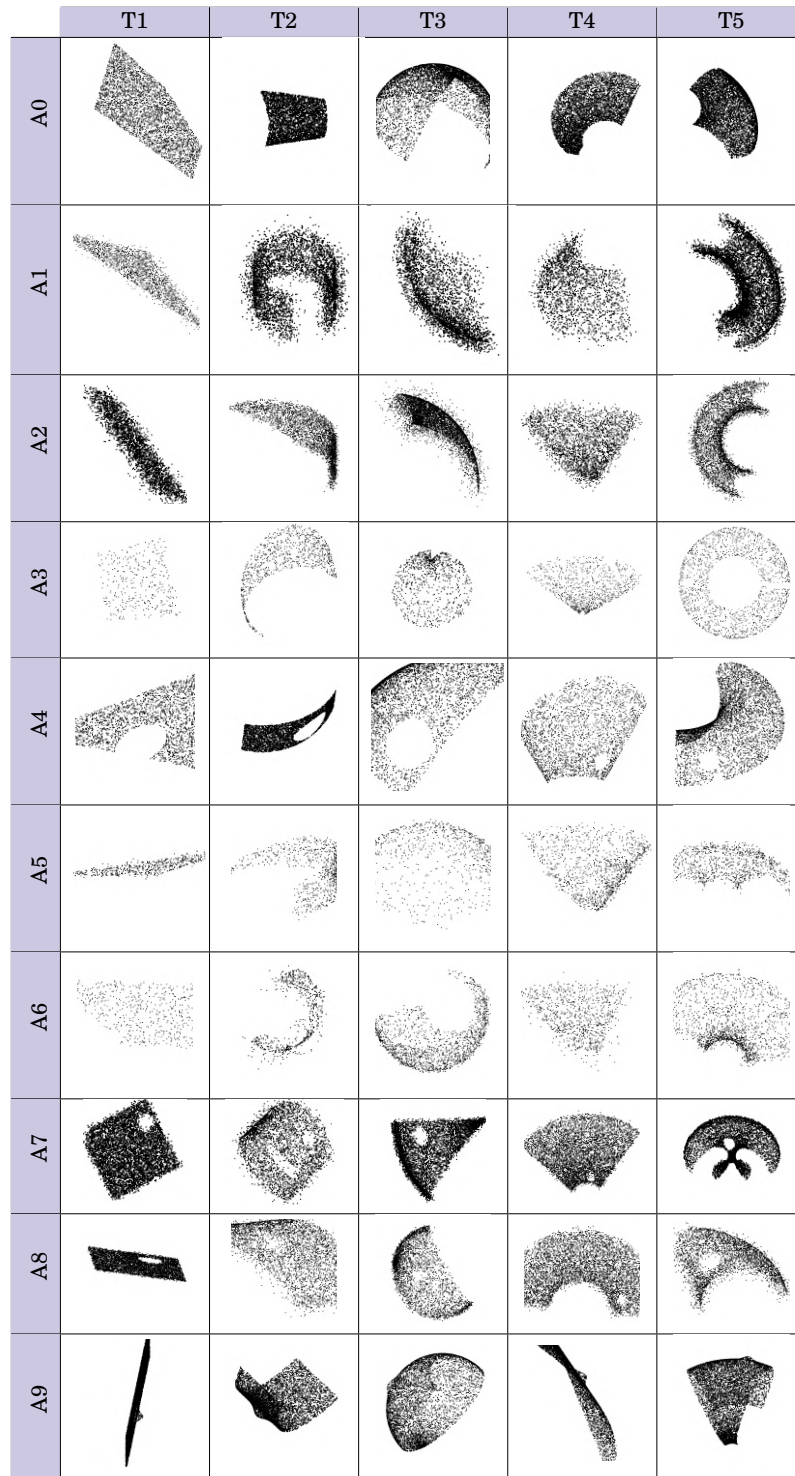


Figure A.1: Examples of point clouds from the dataset. Columns identify primitive types: plane (T1), cylinder (T2), sphere (T3), cone (T4) and torus (T5). Rows correspond to point cloud artefacts: none (A0), uniform noise (A1), Gaussian noise (A2), undersampling (A3), missing data (A4), uniform noise + undersampling (A5), Gaussian noise + undersampling (A6), uniform noise + missing data (A7), Gaussian noise + missing data (A8), small deformations (A9).

A.1.1.2 Ground truth

For each point cloud of the training set, the track participants were provided with a ground truth TXT file that includes the primitive type and the geometric parameters required to identify each segment. More precisely, each file contains a column vector \mathbf{v} defined as follows:

- *Plane*. We list the primitive type (for planes, codified as 1), the unit normal vector¹ \mathbf{a} and a point sampled on the plane \mathbf{P} . Then, the file provides the vector $\mathbf{v} = [1, \mathbf{a}^t, \mathbf{P}^t]^t$.
- *Cylinder*. We list the primitive type (for cylinders, codified as 2), the radius r , the unit vector determining the rotational axis \mathbf{a} , and a point \mathbf{P} sampled on the axis. In this case, the file contains the vector $\mathbf{v} = [2, r, \mathbf{a}^t, \mathbf{P}^t]^t$.
- *Sphere*. We list the primitive type (for spheres, codified as 3), the radius r and the centre \mathbf{C} . The ground truth corresponds to the vector $\mathbf{v} = [3, r, \mathbf{C}^t]^t$.
- *Cone*. We list the primitive type (for cones, codified as 4), half the aperture² α , the unit vector determining the rotational axis \mathbf{a} and the vertex \mathbf{C} . Then, the file provides the vector $\mathbf{v} = [4, \alpha, \mathbf{a}^t, \mathbf{C}^t]^t$.
- *Torus*. We list the primitive type (for tori, codified as 5), the major and minor radii, respectively r and R , the unit vector determining the rotational axis \mathbf{a} and the centre \mathbf{C} . In this case, the file contains the vector $\mathbf{v} = [5, r, R, \mathbf{a}^t, \mathbf{C}^t]^t$.

For each point cloud, the track participants were required to return a TXT file per point cloud (of the test set), with the same syntax employed for the ground truth files of the training set.

A.1.1.3 Classification measures

The *confusion matrix* [93] is a popular way to visualise the classification performance of a method. A confusion matrix CM is a square matrix whose order equals the number of classes in the dataset under study. Diagonal elements count true positives, i.e., all those items which have been correctly labelled as members of their ground truth classes: precisely, $\text{CM}(i, i)$ is the number of items that have been correctly predicted as elements of class i . Off-diagonal elements give the numbers of items mislabeled by the classifier; precisely, $\text{CM}(i, j)$, with $j \neq i$, is the number of elements wrongly labelled as belonging to class j rather than to class i . Ideal classifiers have diagonal classification matrices.

¹To be precise, the definite article “the” is here used improperly. Indeed, there are two unique unit normal vectors that can be used to define the same plane or rotational axis. In our case, we recover uniqueness by enforcing the first nonzero component to be positive.

²The aperture of a right circular cone, denoted by 2α , is the maximum angle between two generatrix lines; we here report half the aperture, i.e., α .

- *True Positive and Negative Rates.* True Positive Rate (TPR) refers to the method’s ability to correctly identify positives (e.g., the percentage of cats correctly classified as cats). Likewise, True Negative Rate (TNR) measures the classifier’s ability to identify negatives (e.g., the percentage of non-cats correctly classified as non-cats). In statistics, TPR and TNR are also called *sensitivity* and *specificity*. A perfect classifier is 100% sensitive and 100% specific.
- *Positive and negative predicted values.* Besides TPR and TNR, to quantify the likelihood that a method returns a true positive (or true negative) rather than a false-positive (or a false-negative) we consider the Positive Predictive Value (PPV) and the Negative Predictive Value (NPV). PPV is the ratio of *true positives* to the number of items labelled as positives by the classifier. Similarly, NPV is the ratio of *true negatives* to the number of items classified as negatives.
- *Accuracy.* Accuracy describes how often a classifier is correct in its predictions: precisely, it is defined as the proportion of predictions that the classifier got right.
- *Macro-average.* Macro-average permits to evaluation the overall performance of the classifier treating all classes equally, it is defined as the arithmetic mean of one of the metrics mentioned above, computed independently for each class.

A.1.1.4 Fitting and recognition measures

To measure the approximation accuracy of a specific primitive, we exploit the geometric descriptors predicted by each method and the corresponding parametric representation to sample densely the recognised surface primitive, say \mathcal{S}_P . Let us consider a point cloud \mathcal{P} to be evaluated; we use the following two measures to evaluate the approximation accuracy:

- *Mean Fitting Error* $\text{MFE}(\mathcal{P}, \mathcal{S}_P)$, as defined in Eq. (3.2);
- *Directed Hausdorff distance* $d_{\text{dHaus}}(\mathcal{P}, \mathcal{S}_P)$, as defined in Eq. (2.2).

Finally, the recognition accuracy is evaluated in terms of L^2 norm between the column vectors of the ground truth and that obtained by each method. Specifically, we compare the geometric descriptors recognised with those of the ground truth considering the L^2 distance:

$$d(\mathbf{v}_G, \mathbf{v}_P) = \|\mathbf{v}_G - \mathbf{v}_P\|_2$$

where \mathbf{v}_G and \mathbf{v}_P here denote the ground truth and the prediction vectors, respectively, where we remove the first entry and, for planes and cylinders, the point (since they are not comparable).

A.1.2 Evaluation

Six methods submitted their results, namely: accelerated Hough transform (M1), Histograms of local features and parameter fitting (M2), PointNet (M3), 3D ShapeNets (M4), PointNet bundle and parameter estimation with least square fitting (M5), AlexNet and ISO reconstruction standards (M6). Two approaches – M1 and M2 – base their whole pipeline on direct methods; M3 and M4 are fully based in neural networks; M5 and M6 exploit both methodologies, that is, neural architectures for the classification task and direct methods for the fitting and recognition tasks. The approach M1 is described in Section 5.3.1, while the others are detailed in [139].

A.1.2.1 Classification task

Figure A.2 shows the confusion matrices over the whole test set. All six methods have a high number of true positives, being cones and tori the most difficult primitive types to be recognised. A more thorough analysis is provided by Table A.1, which contains the following information: Positive Predicted Value (PPV), Negative Predicted Value (NPV), True Positive Rate (TPR), True Negative Rate (TNR), Accuracy (ACC) and their macro-averages. We can notice that:

- Methods M2, M3, M4 and M5 have NPV (slightly) higher than PPV: it is more likely for these methods to be right when reporting a negative rather than a positive. Methods M1 and M6 have mixed outcomes.
- Methods M3 and M5 have TNR higher than TPR, making them more reliable in correctly finding true negatives rather than true positives. Methods M1, M2, M4 and M6 present, for some primitive types, a TPR higher than the TNR: this means that, in some cases, the classifier is more likely to identify correctly true positives rather than true negatives.
- All methods have an accuracy over 90%, with slightly lower scores for cylinders, cones and tori. Macro-averaged values of accuracy are always above 95%, with two of the three best scores reached by neural methods.

Figure A.3 clarifies the (global) impact of different point cloud artifacts upon the six methods, with respect to the classification measures. We observe that:

- M1, M2, M4 and M6 reach their best performance on clean data. Surprisingly enough, M2, M4 and M6 perform best on data with small deformations: these three approaches appear to be able to extract some additional hidden information to improve their prediction capability.
- M1 suffers the most from undersampled data. On the other hand, coherently with the Hough paradigm on which the method relies on, it is not much affected by missing data.

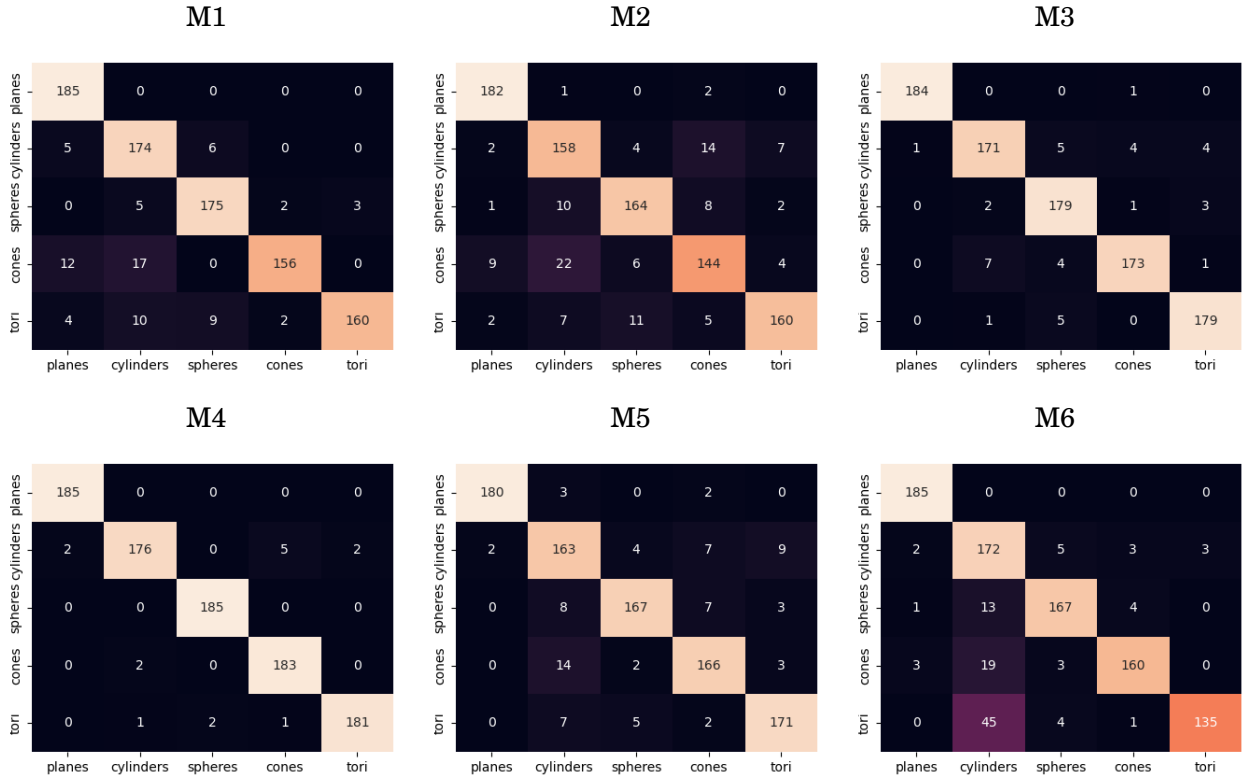


Figure A.2: Confusion matrices CM for the whole test set, with respect to each primitive type. Here, the entry $CM(i, j)$ indicates the number of samples with the true label being the i -th class and the predicted label being the j -th class.

- M3 and M4 have their lowest performance, respectively, in case of uniform and Gaussian noise. However, the grouped bar charts suggest these two methods are the least prone to misclassification. This result confirms the power of neural methods in handling classification tasks.
- In spite of having, in most cases, the lowest classification performance, M2 has still no measure that goes below 80%. Compared to the other method using PointNet (M3), here the implementation seems to overfit way more the training data.
- M5 has a relatively high performance on undersampled data, when compared to other point cloud artifacts.
- The combination of noise and undersampling has proved to be the most challenging perturbation for M6 which, however, exhibits one of the best overall performance on small deformations.

By checking which segments are misclassified, it can be additionally observed that the segments which results being more problematic are, in many cases, the small ones.

A.1. THE SHREC2022 TRACK ON FITTING AND RECOGNISING GEOMETRIC PRIMITIVES
IN POINT CLOUDS

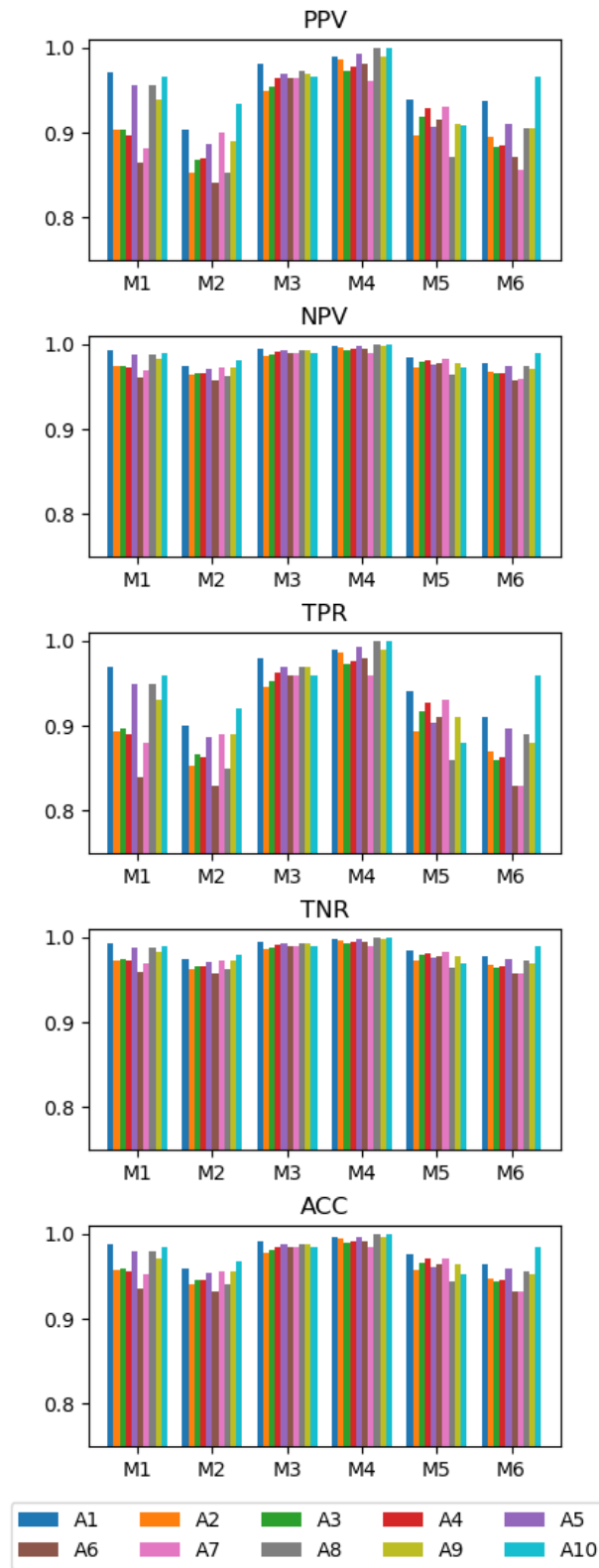


Figure A.3: Bar graphs of the macro averages for the classification measures, grouped by method. The legend reflects the colour encoding of the perturbation types.

Table A.1: Classification metrics for the whole test set, with respect to each primitive type: T1=plane, T2=cylinder, T3=sphere, T4=cone, T5=torus. Here, gold, silver and bronze identify the first, second and third best performance. The last column contains the macro averages.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.8981	0.8447	0.9211	0.9750	0.9816	0.9241
	M2	0.9286	0.7980	0.8865	0.8324	0.9249	0.8741
	M3	0.9946	0.9448	0.9275	0.9665	0.9572	0.9581
	M4	0.9893	0.9832	0.9893	0.9683	0.9891	0.9838
	M5	0.9890	0.8359	0.9382	0.9022	0.9194	0.9169
	M6	0.9686	0.6908	0.9330	0.9524	0.9783	0.9046
NPV	M1	1.0000	0.9847	0.9864	0.9621	0.9672	0.9801
	M2	0.9959	0.9629	0.9716	0.9455	0.9668	0.9685
	M3	0.9986	0.9812	0.9918	0.9839	0.9919	0.9895
	M4	1.0000	0.9879	1.0000	0.9973	0.9946	0.9960
	M5	0.9933	0.9699	0.9759	0.9744	0.9811	0.9789
	M6	1.0000	0.9808	0.9759	0.9670	0.9365	0.9720
TPR	M1	1.0000	0.9405	0.9459	0.8432	0.8649	0.9189
	M2	0.9838	0.8541	0.8865	0.7784	0.8649	0.8735
	M3	0.9946	0.9243	0.9676	0.9351	0.9676	0.9578
	M4	1.0000	0.9514	1.0000	0.9892	0.9784	0.9838
	M5	0.9730	0.8811	0.9027	0.8973	0.9243	0.9157
	M6	1.0000	0.9297	0.9027	0.8649	0.7297	0.8854
TNR	M1	0.9716	0.9568	0.9797	0.9946	0.9959	0.9797
	M2	0.9811	0.9459	0.9716	0.9608	0.9824	0.9684
	M3	0.9986	0.9865	0.9811	0.9919	0.9892	0.9895
	M4	0.9973	0.9959	0.9973	0.9919	0.9973	0.9959
	M5	0.9973	0.9568	0.9851	0.9757	0.9797	0.9789
	M6	0.9919	0.8959	0.9838	0.9892	0.9959	0.9714
ACC	M1	0.9773	0.9535	0.9730	0.9643	0.9697	0.9676
	M2	0.9816	0.9276	0.9546	0.9243	0.9589	0.9494
	M3	0.9978	0.9741	0.9784	0.9805	0.9849	0.9831
	M4	0.9978	0.9870	0.9978	0.9914	0.9935	0.9935
	M5	0.9924	0.9416	0.9686	0.9600	0.9686	0.9663
	M6	0.9935	0.9027	0.9676	0.9643	0.9427	0.9542

A.1.2.2 Recognition and fitting tasks

Table A.2 provides various statistics of the measures introduced in Section A.1.1.4, when the whole test set is considered. It contains the following information: first, second and third quartiles, mean value and standard deviation. More specifically, these quartiles split a set of sorted real numbers into four parts of approximately equal cardinality: the first (Q1) and the third (Q3) quartiles are defined as the values such that, respectively, 25% and 75% of the numbers lie below them; the second quartile (Q2) is the median of the set. Quartiles' significance is due to their ability to identify possible outliers. Based on quartiles, we can notice that:

- The L^2 distance provides low values for all methods with regard to the three quartiles. The lowest Q1 has an order of magnitude of -14 and is obtained by method M6; the highest order of magnitude is that of method M3, and corresponds to -1 . Q2 and Q3 have order of magnitudes between -3 and -1 .

A.1. THE SHREC2022 TRACK ON FITTING AND RECOGNISING GEOMETRIC PRIMITIVES
IN POINT CLOUDS

- With the sole exception of the third quartile of M5, quartiles of the MFEs lie under the order of magnitude of -2 . The lowest values are obtained by M2.
- When it comes to quartiles, the Hausdorff distance the order of magnitude is generally (non-strictly) lower than -1 ; the only exceptions are the third quartiles of methods M3 and M5.

Table A.2: Statistics of the fitting errors for the whole test set. Here, gold, silver and bronze identify the first, second and third best performance.

	Method	Q1	Q2	Q3	mean	std
L^2	M1	2.16e-02	6.82e-02	1.78e-01	2.06e-01	3.71e-01
	M2	8.95e-10	5.17e-03	9.27e-02	1.21e+11	2.78e+12
	M3	1.77e-01	4.73e-01	8.73e-01	6.03e-01	5.54e-01
	M4	2.37e-07	1.54e-02	4.43e-02	5.18e-02	1.84e-01
	M5	6.25e-04	2.15e-02	9.01e-01	2.46e+02	1.64e+03
	M6	1.49e-14	1.49e-02	1.05e-01	1.28e+06	7.54e+06
MFE	M1	3.71e-03	5.96e-03	1.05e-02	9.19e-03	9.42e-03
	M2	0.00	1.94e-03	5.11e-03	1.39e+10	3.51e+11
	M3	1.71e-02	4.49e-02	8.67e-02	6.57e-02	7.46e-02
	M4	2.32e-03	3.59e-03	5.87e-03	5.82e-03	1.43e-02
	M5	3.45e-03	1.49e-02	1.70e-01	2.62e+00	4.46e+01
	M6	2.04e-03	3.43e-03	9.51e-03	2.56e+05	1.46e+06
d_{dHaus}	M1	9.49e-02	1.67e-01	3.05e-01	2.50e-01	2.58e-01
	M2	6.12e-02	9.25e-02	2.68e-01	1.07e+11	2.60e+12
	M3	3.72e-01	6.62e-01	1.09e+00	8.13e-01	6.34e-01
	M4	5.82e-02	7.90e-02	1.51e-01	2.23e-01	4.74e-01
	M5	7.18e-02	2.98e-01	4.24e+00	3.87e+01	5.77e+02
	M6	5.85e-02	8.95e-02	2.53e-01	1.31e+06	7.63e+06

Based on the arithmetic mean and standard deviation, we can conclude that:

- Methods M2, M5 and M6 are affected by numerical instability issues, which result in the presence of outliers; it is worth noting, however, this problem is limited to the top 25% of the errors.

- M2 presents 11 outliers corresponding to the recognition of cones in which the estimated half-angle estimate is close to zero and the estimated vertex is far away from the origin of the coordinate axes.
- M5 provides different outliers depending on the type of task. Specifically, it has 61 outliers in the parameter recognition of various cylinders, cones and tori; however, for these clouds, fitting errors are low. We therefore conclude that, in all these cases, the approximation is successful but at the cost of a poor recognition. From the fitting task point of view, MFE and directed Hausdorff distance have high values in correspondence of 18 outliers correctly identified as planes: here, the normal is correctly identified (thus, the low value of L^2 distance); on the other hand, the method fails in returning a point passing through each returned plane.
- M6 presents 30 outliers, which correspond to point clouds classified as cones; the problem these estimates suffer from is analogue to that of M2.
- M1 and M4 alternate the first and second places in terms of means and standard deviations, suggesting that direct and neural methods can both provide competitive solutions to the problem under study.

By checking the bar graphs of the log mean errors with respect to the perturbation type, see Figure A.4, it is possible to note a generally low variation of the (average) performance of each method, with the exception of M2 and M6; for these results, the logarithmic scale was preferred because of the presence of outliers. It is indicative that, when the methods (i.e., all but M2 and M6) are right in predicting the primitive type, their estimates are not much influenced by the perturbation type. Moreover:

- Methods M2 and M6 have better recognition and fitting performance when dealing with clean data or with point clouds having just small deformations. However, it is worth noticing once more that the poor performance mostly depends on the presence of outliers (see Table A.2).
- M5 is on average better in fitting than in recognition: despite being the mean L^2 errors rather high, mean MFEs and directed Hausdorff distances are lower.
- M1 and M4 behave similarly in both recognition and fitting tasks.

A.2 The Fit4CAD benchmark

In this section we describe 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

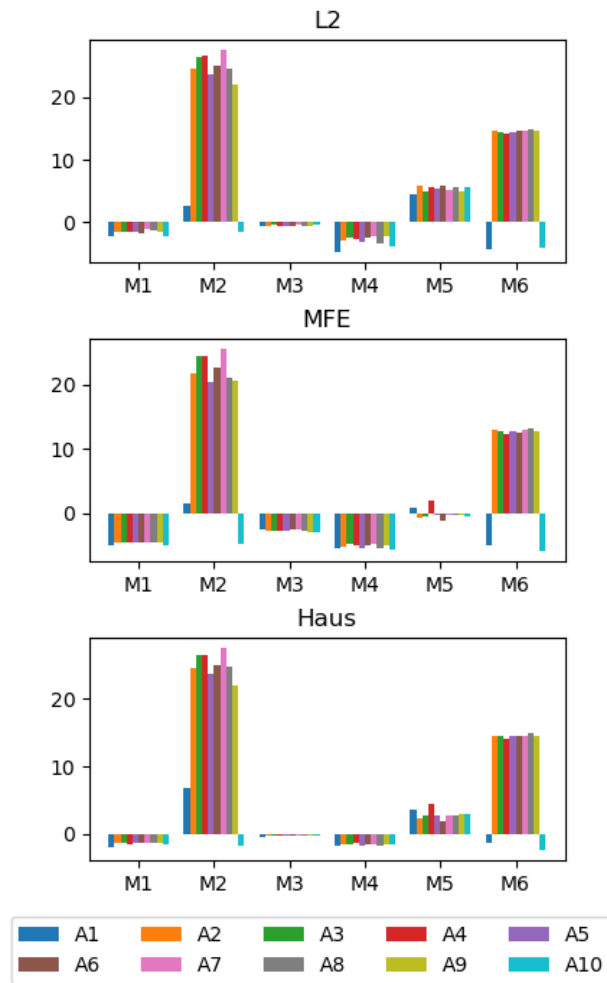


Figure A.4: Bar graphs of the log macro averages for the recognition and fitting measures, grouped by method. The legend reflects the colour encoding of the perturbation types.

set and a test set. In addition, a set of quality metrics and two fitting methods are given. This benchmark is meant to help both method developers and those who want to identify the best performing tools. 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, because the dataset has been already organised in the form of a training set and a test set. Our benchmark is available at <https://github.com/chiararomanengo/Fit4CAD>.

A.2.1 Dataset and performance measures

A.2.1.1 Dataset

At present, the dataset contains 225 individual high quality point clouds, each of which has been obtained by sampling a CAD model. 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. 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 [91], 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 A.5.
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 [67]; nevertheless, we emphasise that other software could be considered too (e.g., [110]). 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 [50] and further developed in [16]. 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 $x^i y^j$ are such $i + j \leq n$, and there exists at least one monomial $x^i y^j$ such that $i + j = n$.
3. *Point cloud extraction.* CAD objects are sampled at different densities, and optionally manually postprocessed by using CloudCompare³ to simulate missing data. To give an example, Figure A.6(a) shows a model from Onshape, which is then sampled and postprocessed in A.6(b-c).

A.2.1.2 Groundtruth

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.

PC_i lists the three-dimensional points forming the point cloud to be segmented.

³CloudCompare (version 2.10.2), <http://www.cloudcompare.org/>

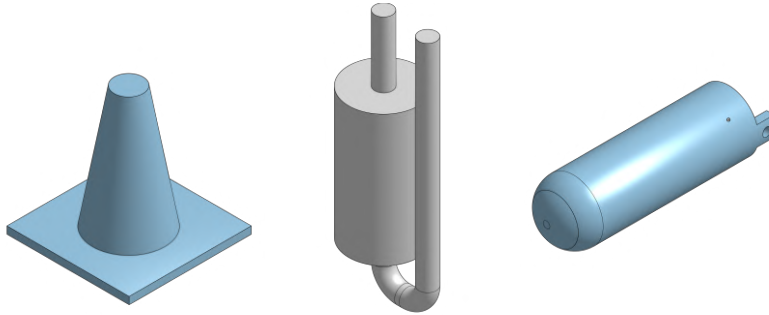


Figure A.5: Example of models obtained using Onshape.

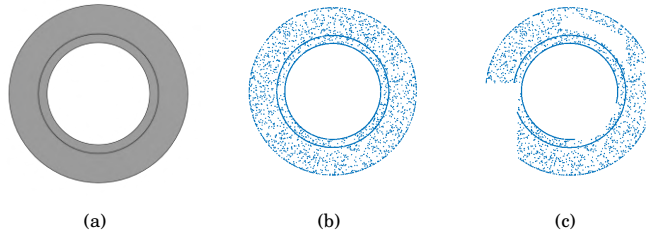


Figure A.6: Example of point cloud creation. The initial object in (a) is sampled at a chosen density (b) and then perturbed by simulating missing data (c).

$PCi_primitives$ 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]
```

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,  $\mathbf{v}$ ]
```

where \mathbf{v} is the vector that contains the parameters of the parametric representation (see [140] 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,  $\mathbf{w}$ ]
```

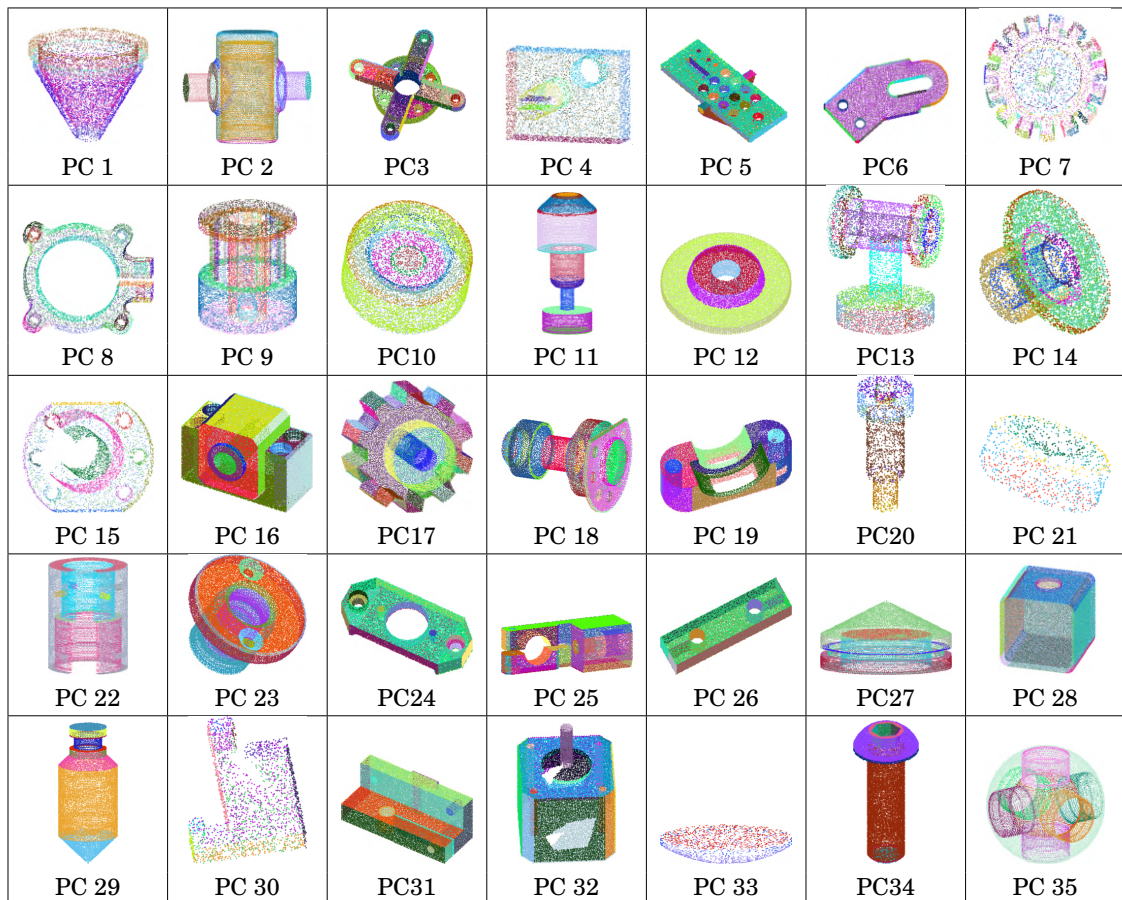


Figure A.7: The 35 point clouds used as a test set. Different colours represent different primitives, as stored in the CAD models, i.e., our ground truth.

where \mathbf{w} is the vector that contains the coefficients of the implicit representation (see [140] 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.

A.2.1.3 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 \mathcal{P}_B be a set of points in the benchmark point cloud corresponding to a specific primitive, and let \mathcal{P}_S be the primitive in the segmentation to assess that most overlap with \mathcal{P}_B . We can

define the following quantities:

- *True positives*, TP: the number of points shared by \mathcal{P}_B and \mathcal{P}_S .
- *False positives*, FP: the number of points in \mathcal{P}_S that do not belong to \mathcal{P}_B .
- *False negatives*, FN: the number of points in \mathcal{P}_B that do not belong to \mathcal{P}_S .
- *True negatives*, TN: the number of points that do not belong to either \mathcal{P}_B nor \mathcal{P}_S .

Based on these four quantities, we consider the following measures:

- *Sensitivity*, also called *true positive rate*, measures the proportion of positives that are correctly identified, i.e.,

$$\text{TPR} := \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Specificity, or *true negative rate*, measures the proportion of true negatives that are correctly identified as such, i.e.,

$$\text{TNR} := \frac{\text{TN}}{\text{TN} + \text{FP}}.$$

- *Positive predictive value* is defined as the proportion of predicted positives that are actual positives, i.e.,

$$\text{PPV} := \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Similarly, *negative predictive value* is given by

$$\text{NPV} := \frac{\text{TN}}{\text{TN} + \text{FN}}.$$

- *Accuracy* is the ratio of correct predictions to total predictions made, i.e.,

$$\text{ACC} := \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

- *Sørensen-Dice index*. It is given by

$$\text{DSC} := \frac{2|\mathcal{P}_B \cap \mathcal{P}_S|}{|\mathcal{P}_B| + |\mathcal{P}_S|}.$$

In case of binary classification, it is shown to be equivalent to

$$\text{DSC} := \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}},$$

which is often referred to as F_1 score.

For more details, we refer the reader to [93].

A.2.1.4 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 \mathcal{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 \mathcal{P}_S :

- *Mean Fitting Error* ($\text{MFE}(\mathcal{P}_S, \mathcal{S})$) as defined in Eq. 3.2;
- *Directed Hausdorff distance* ($d_{\text{dHaus}}(\mathcal{P}_S, \mathcal{S})$) as defined in Eq. 2.2. To make the measure independent from the primitive size, we normalize it with respect to the diagonal l of the minimum bounding box containing \mathcal{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 \mathcal{P}_S and \mathcal{P}_B , respectively, and where $\|\cdot\|_1$ is the well-known ℓ^1 norm. 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 [140]).

We refer to [46] for further details.

A.2.2 Evaluation

We here analyse the performance of two methods, to show how the benchmark works. The first approach (HT) is the one described in Section 5.4.1, while the second one (PG) is a curvature-based method based on a primitive growing framework, based on the method proposed in [125] for triangle meshes. 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 5.4.1.

Performance measures of the point classification

Table A.3 summarizes the performances of the two methods over all the test set models. Each row corresponds 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 A.2.1.3. For

each metric, two columns are considered, respectively referring to the PG- and the HT-based approaches.

Table A.3: Number of fitted primitives and classification performance metrics: comparison between the PG-based and the HT-based algorithms.

	# 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	17	13	16	0.656	0.921	0.795	0.960	0.632	0.937	0.994	0.999	0.947	0.998	0.944	0.998
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	0.999
PC 4	10,000	15	14	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	0.533	0.848	0.615	0.842	0.566	0.891	0.981	0.999	0.951	0.994	0.937	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	7,500	8	8	8	0.946	0.975	0.936	0.958	0.970	0.999	0.989	1.000	0.997	0.999	0.990	0.995

To ease the analysis, the metrics are studied via boxplots:

- Figure A.8 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 vary 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 A.9. 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 A.3), 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.

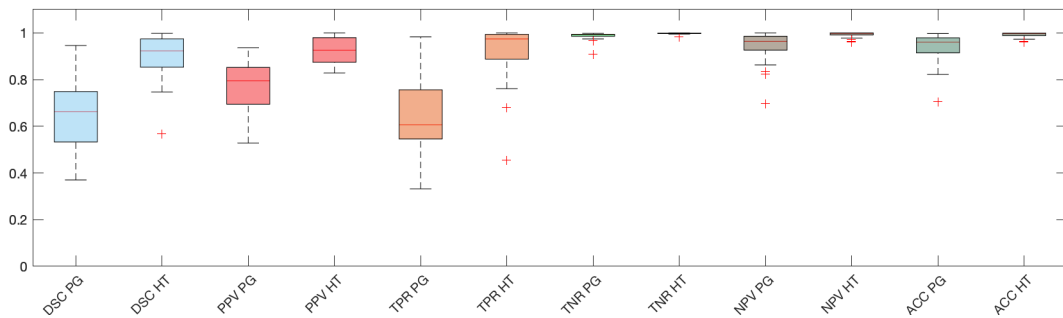


Figure A.8: Boxplot for the classification metrics presented in Table A.3. All 35 models are here considered.

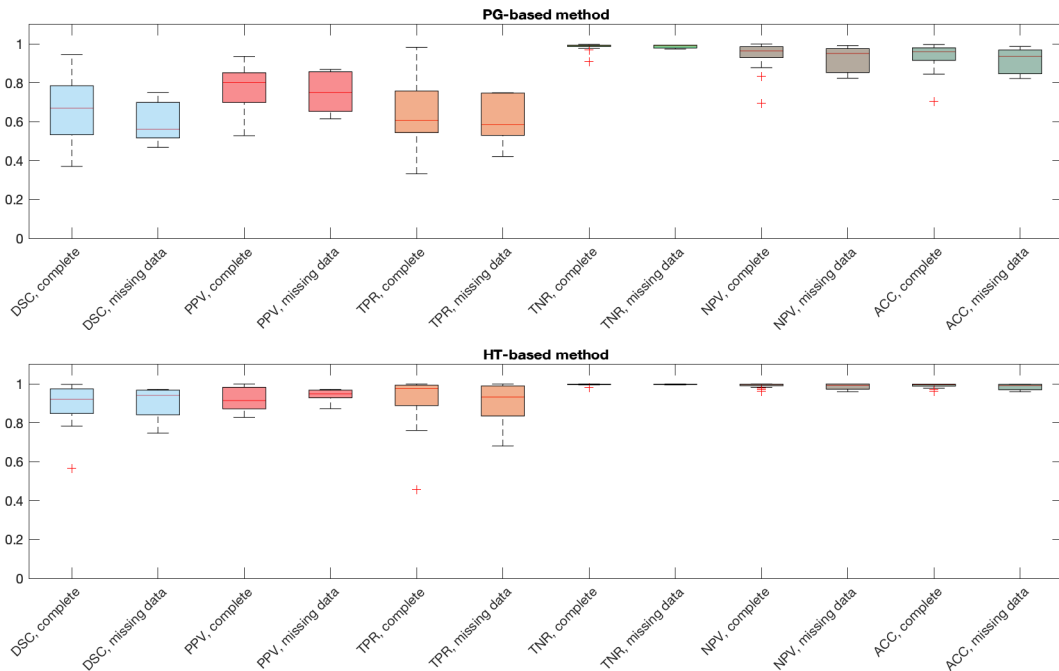


Figure A.9: Performance of the PG- and HT-based methods, with an eye on models suffering from missing data. For these boxplots, we have made use of the classification metrics presented in Table A.3.

Approximation accuracy

Table A.4 reports the performance of the HT-based method evaluated according to the metrics reported in Section A.2.1.4. Each row corresponds to a segmented point cloud; each column represents a different accuracy measure; for each point cloud, each measure has been obtained by averaging over all segments.

- 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 0.7%.
- 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 a 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.

A.3 Related publications

- C. Romanengo, A. Raffo, S. Biasotti, B. Falcidieno, V. Fotis, I. Romanelis, E. Psatha, K. Moustakas, I. Sipiran, Q.-T. Nguyen, C.-B. Chu, K.-N. Nguyen-Ngoc, D.-K. Vo, T.-A. To, N.-T. Nguyen, N.-Q. Le-Pham, H.-D. Nguyen, M.-T. Tran, Y. Qie, N. Anwer. , *Fitting and recognition of simple geometric primitives on point clouds*, Computer and Graphics (2022), vol. 107, pp. 32-49.

Table A.4: 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	0.132
PC 3	0.002	0.004	0.621
PC 4	0.002	0.003	0.000
PC 5	0.007	0.011	0.082
PC 6	0.003	0.005	0.001
PC 7	0.007	0.011	0.232
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	0.003	0.010	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

- C. Romanengo, A. Raffo, Y. Qie, N. Anwer, B. Falcidieno, *Fit4CAD: A point cloud benchmark for fitting simple geometric primitives in CAD objects*, *Computer and Graphics* (2022), vol. 102, pp. 133-143.

BIBLIOGRAPHY

- [1] *The Shape Repository*.
Available at <http://visionair.ge.imati.cnr.it/ontologies/shapes/>, 2011–2015.
- [2] *STARC repository*.
Available at <http://public.cyi.ac.cy/starcrepo/>, 2014.
- [3] *GRAVITATE: Discovering relationships between artefacts using 3D and semantic data*, 2015-2018.
EU H2020 REFLECTIVE project.
- [4] *INTER-CH: interfacce innovative per la valorizzazione del patrimonio storico-artistico ligure*.
Available at <http://www.imati.cnr.it>, 2020-2022.
POR Liguria FSE 2014- 2020.
- [5] *Open Cascade*.
Available at <https://dev.opencascade.org/doc/overview/html/index.html>, 2021.
- [6] J. ABBOTT, A. M. BIGATTI, AND G. LAGORIO, *CoCoA-5: a system for doing Computations in Commutative Algebra*.
Available at <http://cocoa.dima.unige.it>, 2019.
- [7] G. ALBRECHT, C. V. BECCARI, J.-C. CANONNE, AND L. ROMANI, *Planar pythagorean-hodograph b-spline curves*, *Computer Aided Geometric Design*, 57 (2017), pp. 57–77.
- [8] G. ALBRECHT, C. V. BECCARI, AND L. ROMANI, *Spatial pythagorean-hodograph b-spline curves and 3d point data interpolation*, *Computer Aided Geometric Design*, 80 (2020), p. 101868.
- [9] E. ALBUZ, E. D. KOCALAR, AND A. A. KHOKHAR, *Quantized cielab* space and encoded spatial structure for scalable indexing of large color image archives*, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2000)*, vol. 6, IEEE, 2000, pp. 1995–1998.
- [10] A. ANDREADIS, G. PAPAIOANNOU, AND P. MAVRIDIS, *Generalized digital reassembly using geometric registration*, in *Digital Heritage*, vol. 2, IEEE, 2015, pp. 549–556.

BIBLIOGRAPHY

- [11] M. ATTENE AND B. FALCIDIENO, *Remesh: An interactive environment to edit and repair triangle meshes*, in IEEE International Conference on Shape Modeling and Applications 2006 (SMI 2006), IEEE, 2006, pp. 271–276.
- [12] M. ATTENE, B. FALCIDIENO, J. ROSSIGNAC, AND M. SPAGNUOLO, *Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces*, in Eurographics Symposium on Geometry Processing, (SGP 2003), L. Kobbelt, P. Schroeder, and H. Hoppe, eds., 2003, pp. 62–69.
- [13] M. ATTENE AND G. PATANÈ, *Hierarchical structure recovery of point-sampled surfaces*, Computer Graphics Forum, 29 (2010), pp. 1905–1920.
- [14] D. H. BALLARD, *Generalizing the Hough transform to detect arbitrary shapes*, Pattern recognition, 13 (1981), pp. 111–122.
- [15] G. BAREQUET AND M. SHARIR, *Piecewise-linear interpolation between polygonal slices*, Computer Vision and Image Understanding, 63 (1996), pp. 251 – 272.
- [16] O. J. D. BARROWCLOUGH AND T. DOKKEN, *Approximate implicitization using linear algebra*, Journal of Applied Mathematics, (2012), pp. 1 – 25.
- [17] M. BELTRAMETTI, E. CARLETTI, D. GALLARATI, AND G. MONTI BRAGADIN, *Lectures on Curves, Surfaces and Projective Varieties—A Classical View of Algebraic Geometry*, European Mathematical Society, 9th ed., 2009.
- [18] M. BELTRAMETTI, A. MASSONE, AND M. PIANA, *Hough transform of special classes of curves*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 391–412.
- [19] M. BELTRAMETTI, J. SENDRA, J. SENDRA, AND M. TORRENTE, *Moore–penrose approach in the hough transform framework*, Applied Mathematics and Computation, 375 (2020), p. 125083.
- [20] M. C. BELTRAMETTI AND L. ROBBIANO, *An algebraic approach to Hough transforms*, Journal of Algebra, 37 (2012), pp. 669–681.
- [21] F. BERGAMASCO, M. PISTELLATO, A. ALBARELLI, AND A. TORSELLO, *Cylinders extraction in non-oriented point clouds as a clustering problem*, Pattern Recognition, 107 (2020), p. 107443.
- [22] F. BERNARDINI, J. MITTLEMAN, H. RUSHMEIER, C. SILVA, AND G. TAUBIN, *The ball-pivoting algorithm for surface reconstruction*, IEEE Transactions on Visualization and Computer Graphics, 5 (1999), pp. 349–359.

-
- [23] S. BIASOTTI, A. CERRI, B. FALCIDIENO, AND M. SPAGNUOLO, *3D artifacts similarity based on the concurrent evaluation of heterogeneous properties*, Journal on Computing and Cultural Heritage, 8 (2015), pp. 1–19.
- [24] S. BIASOTTI, A. CERRI, S. PITTALUGA, D. SOBRERO, AND M. SPAGNUOLO, *Tracking the evolution of rainfall precipitation fields using persistent maxima*, in Proceedings of the Conference on Smart Tools and Applications in Computer Graphics, 2016, pp. 29–37.
- [25] S. BIASOTTI, B. FALCIDIENO, D. GIORGI, AND M. SPAGNUOLO, *Mathematical tools for shape analysis and description*, Synthesis Lectures on Computer Graphics and Animation, 6 (2014), pp. 1–138.
- [26] T. BIRDAL, B. BUSAM, N. NAVAB, S. ILIC, AND P. STURM, *Generic primitive detection in point clouds using novel minimal quadric fits*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 42 (2020), pp. 1333–1347.
- [27] J. BOCHNAK, M. COSTE, AND M.-F. ROY, *Real Algebraic Geometry*, Springer Berlin, Heidelberg, 1st ed., 1998.
- [28] D. BORRMANN, J. ELSEBERG, K. LINGEMANN, AND A. NÜCHTER, *The 3D Hough transform for plane detection in point clouds: A review and a new accumulator design*, 3D Research, 2 (2011).
- [29] C. BRACCO, T. LYCHE, C. MANNI, F. ROMAN, AND H. SPELEERS, *Generalized spline spaces over t -meshes: Dimension formula and locally refined generalized b -splines*, Applied Mathematics and Computation, 272 (2016), pp. 187–198.
- [30] A. BUFFA AND G. SANGALLI, *IsoGeometric Analysis: A New Paradigm in the Numerical Approximation of PDEs*, Springer, 2nd ed., 2012.
- [31] F. BUONAMICI, M. CARFAGNI, R. FURFERI, L. GOVERNI, A. LAPINI, AND Y. VOLPE, *Reverse engineering modeling methods and tools: a survey*, Computer-Aided Design and Applications, 15 (2018), pp. 443–464.
- [32] M. CAMURRI, R. VEZZANI, AND R. CUCCHIARA, *3D Hough Transform for Sphere Recognition on Point Clouds*, Machine Vision and Applications, 25 (2014), p. 1877–1891.
- [33] J. CANNY, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8 (1986), pp. 679–698.
- [34] Y. CAO, D.-M. YAN, AND P. WONKA, *Patch layout generation by detecting feature networks*, Computers & Graphics, 46 (2015), pp. 275 – 282.

- [35] C. E. CATALANO, A. REPETTO, AND M. SPAGNUOLO, *A Dashboard for the Analysis of Tangible Heritage Artefacts: a Case Study in Archaeology*, in Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage, (GCH 2017), D. Fellner, ed., The Eurographics Association, 2017.
- [36] B. F. CAVINESS, *On Canonical Forms and Simplification*, Journal of the ACM, 17 (1970), p. 385–396.
- [37] P. CIGNONI, M. CALLIERI, M. CORSINI, M. DELLEPIANE, F. GANOVELLI, AND G. RANZUGLIA, *MeshLab: an Open-Source Mesh Processing Tool*, in Eurographics Italian chapter conference, (EGIT 2008), V. Scarano, R. De Chiara, and U. Erra, eds., The Eurographics Association, 2008, pp. 129–136.
- [38] D. COHEN-STEINER AND J.-M. MORVAN, *Restricted delaunay triangulations and normal cycle*, in Proceedings of the Nineteenth Annual Symposium on Computational Geometry, Association for Computing Machinery, 2003, p. 312–321.
- [39] F. COLE, A. GOLOVINSKIY, A. LIMPAECHER, H. S. BARROS, A. FINKELSTEIN, T. FUNKHOUSER, AND S. RUSINKIEWICZ, *Where do people draw lines?*, ACM Transaction on Graphics, 27 (2008), pp. 1–11.
- [40] C. CONTI, L. ROMANI, AND D. SCHENONE, *Semi-automatic spline fitting of planar curvilinear profiles in digital images using the Hough transform*, Pattern Recognition, 74 (2018), pp. 64–76.
- [41] J. I. DANIELS, L. K. HA, T. OCHOTTA, AND C. T. SILVA, *Robust smooth feature extraction from point clouds*, in IEEE International Conference on Shape Modeling and Applications 2007 (SMI 2007), 2007, pp. 123–136.
- [42] J. DANIELS II, T. OCHOTTA, K. L. HA, AND T. C. SILVA, *Spline-based feature curves from point-sampled geometry*, The Visual Computer, 24 (2008), pp. 449–462.
- [43] W. DAY AND H. EDELSBRUNNER, *Efficient algorithms for agglomerative hierarchical clustering methods*, Journal of Classification, 1 (1984), pp. 7–24.
- [44] D. DEFAYS, *An efficient algorithm for a complete link method*, The Computer Journal, 20 (1977), pp. 364–366.
- [45] J. DENG, W. DONG, R. SOCHER, L.-J. LI, K. LI, AND L. FEI-FEI, *Imagenet: A large-scale hierarchical image database*, in The IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2009), IEEE, 2009, pp. 248–255.
- [46] M. M. DEZA AND E. DEZA, *Encyclopedia of Distances*, Springer Berlin Heidelberg, 2009.

-
- [47] L. DI ANGELO, P. DI STEFANO, A. E. MORABITO, AND C. PANE, *Measurement of constant radius geometric features in archaeological pottery*, *Measurement*, 124 (2018), pp. 138 – 146.
- [48] J. J. DIGNE AND J.-M. MOREL, *Numerical analysis of differential operators on raw point clouds*, *Numerische Mathematik*, 127 (2014), pp. 255–289.
- [49] M. P. DO CARMO, *Differential geometry of curves and surfaces.*, Prentice Hall, 1976.
- [50] T. DOKKEN, *Aspects of intersection algorithms and approximation*, PhD thesis, University of Oslo, 1997.
- [51] H. DORAISWAMY, N. SHIVASHANKAR, V. NATARAJAN, AND Y. WANG, *Topological saliency*, *Computers & Graphics*, 37 (2013), pp. 787 – 799.
- [52] D. H. DOUGLAS AND T. K. PEUCKER, *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10 (1973), pp. 112–122.
- [53] R. O. DUDA AND P. E. HART, *Use of the Hough transformation to detect lines and curves in pictures*, *Communications of the ACM*, 15 (1972), pp. 11–15.
- [54] N. DYN, J. GREGORY, AND D. LEVIN, *Analysis of uniform binary subdivision schemes for curve design*, *Constructive Approximation*, 7 (1991), pp. 127–147.
- [55] EDELSBRUNNER, LETSCHER, AND ZOMORODIAN, *Topological persistence and simplification*, *Discrete Computational Geometry*, 28 (2002), p. 511–533.
- [56] A. EFRAT, L. GUIBAS, S. HAR-PELED, J. MITCHELL, AND S. MURALI, *New similarity measures between polylines with applications to morphing and polygon sweeping*, *Discrete & Computational Geometry*, 28 (2002), pp. 535–569.
- [57] M. ESTER, H. P. KRIEGEL, J. SANDER, AND X. XU, *A density-based algorithm for discovering clusters in large spatial databases with noise*, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, AAAI Press, 1996, pp. 226–231.
- [58] G. FARIN, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Academic Press, 3rd ed., 1993.
- [59] R. T. FAROUKI, *Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable*, Springer Berlin, Heidelberg, 2008.

BIBLIOGRAPHY

- [60] R. T. FAROUKI, M. AL-KANDARI, AND T. SAKKALIS, *Hermite interpolation by rotation-invariant spatial Pythagorean-Hodograph curves*, *Advances in Computational Mathematics*, 17 (2002), pp. 369–383.
- [61] L. A. FERNANDES AND M. M. OLIVEIRA, *A general framework for subspace detection in unordered multidimensional data*, *Pattern Recognition*, 45 (2012), pp. 3566 – 3579.
- [62] M. A. FISCHLER AND R. C. BOLLES, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, *Communications of the ACM*, 24 (1981), pp. 381–395.
- [63] A. FORREST, *The twisted cubic curve: a computer-aided geometric design approach*, *Computer-Aided Design*, 12 (1980), pp. 165 – 172.
- [64] J. H. FRIEDMAN, J. L. BENTLEY, AND R. A. FINKEL, *An algorithm for finding best matches in logarithmic expected time*, *ACM Transactions on Mathematical Software*, 3 (1977), pp. 209–226.
- [65] T. GATZKE AND C. M. GRIMM, *Estimating curvature on triangular meshes*, *International Journal of Shape Modeling*, 12 (2006), pp. 1–28.
- [66] A. GEHRE, I. LIM, AND L. KOBBELT, *Feature curve co-completion in noisy data*, *Computer Graphics Forum*, 37 (2018), pp. 1–12.
- [67] C. GEUZAIN AND J.-F. REMACLE, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, *International Journal for Numerical Methods in Engineering*, 79 (2009), pp. 1309–1331.
- [68] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, 3rd ed., 1996.
- [69] G. GUENNEBAUD AND M. GROSS, *Algebraic point set surfaces*, *ACM Transactions on Graphics*, 26 (2007), p. 23–32.
- [70] V. GUILLEMIN AND A. POLLACK, *Differential Topology*, Englewood Cliffs, 2010.
- [71] S. GUMHOLD, X. WANG, AND R. MACLEOD, *Feature extraction from point clouds*, in *Proceeding of 10th International Meshing Roundtable*, Sandia National Laboratories, 2001, pp. 293–305.
- [72] G. HARARY AND A. TAL, *The Natural 3D Spiral*, *Computer Graphics Forum*, 30 (2011), pp. 237–246.
- [73] G. HARARY AND A. TAL, *3D Euler spirals for 3D curve completion*, *Computational Geometry*, 45 (2012), pp. 115 – 126.

-
- [74] G. HARARY, A. TAL, AND E. GRINSPUN, *Context-based coherent surface completion*, ACM Transactions on Graphics, 33 (2014), pp. 1–12.
- [75] G. HARARY, A. TAL, AND E. GRINSPUN, *Feature-preserving surface completion using four points*, in Computer Graphics Forum, vol. 33, Wiley Online Library, 2014, pp. 45–54.
- [76] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), 2016, pp. 770–778.
- [77] K. HILDEBRANDT, K. POLTHIER, AND M. WARDETZKY, *Smooth feature lines on surface meshes*, in EG Symposium on geometry processing, (SGP 2005), The Eurographics Association, 2005, pp. 85–90.
- [78] C. M. HOFFMANN, *Conversion methods between parametric and implicit curves and surfaces*, tech. rep., Purdue Univ Lafayette, Dept of Computer Sciences, 1990.
- [79] R. HONTI, J. ERDÉLYI, AND A. KOPÁČIK, *Semi-automated segmentation of geometric shapes from point clouds*, Remote Sensing, 14 (2022).
- [80] H. HOPPE, T. DEROSE, T. DUCHAMP, J. McDONALD, AND W. STUETZLE, *Surface reconstruction from unorganized points*, in Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, Association for Computing Machinery, 1992, p. 71–78.
- [81] P. V. C. HOUGH, *Method and means for recognizing complex patterns*, 1962. US Patent 3,069,654.
- [82] J. HOWARD AND S. GUGGER, *Fastai: A layered api for deep learning*, Information, 11 (2020).
- [83] Y. HU, T. SCHNEIDER, X. GAO, Q. ZHOU, A. JACOBSON, D. ZORIN, AND D. PANOZZO, *Triwild: Robust triangulation with curve constraints*, ACM Transactions on Graphics, 38 (2019), pp. 1–15.
- [84] T. HUGHES, J. COTTRELL, AND Y. BAZILEVS, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 4135 – 4195.
- [85] R. W. G. HUNT AND M. R. POINTER, *Measuring Colour*, Wiley, 4th ed., 2011.
- [86] K. HÖLLIG AND J. KOCH, *Geometric Hermite interpolation*, Computer Aided Geometric Design, 12 (1995), pp. 567 – 580.

BIBLIOGRAPHY

- [87] H. ISHII AND B. ULLMER, *Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms.*, in Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, Association for Computing Machinery, 1997, pp. 234–241.
- [88] A. KAISER, J. A. YBANEZ ZEPEDA, AND T. BOUBEKEUR, *A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data*, Computer Graphics Forum, 38 (2019), pp. 167–196.
- [89] V. KARAGEORGHIS, J. KARAGEORGHIS, AND A. L. FOUNDATION, *The coroplastic art of ancient Cyprus*, Nicosia : A.G. Leventis Foundation, 1991.
- [90] P. KICIAK, *Geometric continuity of curves*, Springer International Publishing, Cham, 2017, pp. 7–37.
- [91] S. KOCH, A. MATVEEV, Z. JIANG, F. WILLIAMS, A. ARTEMOV, E. BURNAEV, M. ALEXA, D. ZORIN, AND D. PANOZZO, *ABC: A big CAD model dataset for geometric deep learning*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2019), 2019, pp. 9601–9611.
- [92] M. KOLOMENKIN, I. SHIMSHONI, AND A. TAL, *Demarcating curves for shape illustration*, ACM Transaction on Graphics, 27 (2008), pp. 1–9.
- [93] K. KUHN, MAX;JOHNSON, *Applied Predictive Modeling*, Springer New York, 2018.
- [94] Y. K. LAI, Q. Y. ZHOU, S. M. HU, J. WALLNER, AND H. POTTMANN, *Robust feature classification and editing*, IEEE Transactions on Visualization and Computer Graphics, 13 (2007), pp. 34–45.
- [95] K. LAWONN, E. TROSTMANN, B. PREIM, AND K. HILDEBRANDT, *Visualization and extraction of carvings for heritage conservation*, IEEE Transactions on Visualization and Computer Graphics, 23 (2017), pp. 801–810.
- [96] T. LE AND Y. DUAN, *A primitive-based 3D segmentation algorithm for mechanical CAD models*, Computer Aided Geometric Design, 52-53 (2017), pp. 231–246.
- [97] H. LI, M. A. LAVIN, AND R. J. LE MASTER, *Fast Hough transform: A hierarchical approach*, Computer Vision, Graphics, and Image Processing, 36 (1986), pp. 139–161.
- [98] H. LI, H. ZHANG, Y. WANG, J. CAO, A. SHAMIR, AND D. COHEN-OR, *Curve style analysis in a set of shapes*, Computer Graphics Forum, 32 (2013), pp. 77–88.
- [99] L. LI, M. SUNG, A. DUBROVINA, L. YI, AND L. J. GUIBAS, *Supervised Fitting of Geometric Primitives to 3D Point Clouds*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2019), 2019, pp. 2652–2660.

-
- [100] Y. LI, X. WU, Y. CHRYSATHOU, A. SHARF, D. COHEN-OR, AND N. J. MITRA, *GlobFit: Consistently Fitting Primitives by Discovering Global Relations*, ACM Transactions on Graphics, 30 (2011).
- [101] P. LIEPA, *Filling Holes in Meshes*, in Eurographics Symposium on Geometry Processing, (SGP 2003), L. Kobbelt, P. Schroeder, and H. Hoppe, eds., 2003, pp. 200–205.
- [102] F. A. LIMBERGER AND M. M. OLIVEIRA, *Real-time detection of planar regions in unorganized point clouds*, Pattern Recognition, 48 (2015), pp. 2043–2053.
- [103] M. LIPSCHUTZ, *Schaum’s Outline of Differential Geometry*, McGraw-Hill Education, 1969.
- [104] C. LIU AND W. HU, *Real-time geometric fitting and pose estimation for surface of revolution*, Pattern Recognition, 85 (2019), pp. 90–108.
- [105] E. LÓPEZ-RUBIO, K. THURNHOFER-HEMSI, E. B. BLÁZQUEZ-PARRA, O. D. DE CÓZAR-MACÍAS, AND M. C. L. DE GUEVARA-MUÑOZ, *A fast robust geometric fitting method for parabolic curves*, Pattern Recognition, 84 (2018), pp. 301 – 316.
- [106] C. LV, Z. WU, X. WANG, M. ZHOU, AND K.-A. TOH, *Nasal similarity measure of 3D faces based on curve shape space*, Pattern Recognition, 88 (2019), pp. 458 – 469.
- [107] E. MAGID, O. SOLDEA, AND E. RIVLIN, *A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data*, Computer Vision and Image Understanding, 107 (2007), pp. 139 – 159.
- [108] V. MARKOVIC, Z. JAKOVLJEVIC, AND I. BUDAK, *Automatic recognition of cylinders and planes from unstructured point clouds*, The Visual Computer, (2021), pp. 1–24.
- [109] A. M. MASSONE, A. PERASSO, C. CAMPI, AND M. C. BELTRAMETTI, *Profile detection in medical and astronomical images by means of the Hough transform of special classes of curves*, Journal of Mathematical Imaging and Vision, 51 (2015), pp. 296–310.
- [110] A. MATHUR, M. PIRRON, AND D. ZUFFEREY, *Interactive Programming for Parametric CAD*, Computer Graphics Forum, 39 (2020), pp. 408–425.
- [111] THE MATHWORKS, INC., *MATLAB version 9.7.0.1247435 Update 2 (R2019b)*, Natick, Massachusetts, 2019.
- [112] E. MOSCOSO THOMPSON AND S. BIASOTTI, *A Preliminary Analysis of Methods for Curvature Estimation on Surfaces With Local Reliefs*, in Eurographics 2019 - Short Papers, P. Cignoni and E. Miguel, eds., The Eurographics Association, 2019.

BIBLIOGRAPHY

- [113] E. MOSCOSO THOMPSON, A. GERASIMOS, K. MOUSTAKAS, E. R. NGUYEN, M. TRAN, T. LEJEMBLE, L. BARTHE, N. MELLADO, C. ROMANENGO, S. BIASOTTI, AND B. FALCIDIENO, *SHREC'19 track: Feature Curve Extraction on Triangle Meshes*, in Eurographics Workshop 3D Object Retrieval, (3DOR 2019), S. Biasotti, G. Lavoué, and R. C. Veltkamp, eds., The Eurographics Association, 2019, pp. 85–92.
- [114] P. MUKHOPADHYAY AND B. B. CHAUDHURI, *A survey of Hough transform*, Pattern Recognition, 48 (2015), pp. 993 – 1010.
- [115] J. R. MUNKRES, *Topology*, Prentice Hall, Inc., 2nd ed., 2000.
- [116] M. NIESER, C. SCHULZ, AND K. POLTHIER, *Patch layout from feature graphs*, Computer-Aided Design, 42 (2010), pp. 213–220.
- [117] S. OESAU, F. LAFARGE, AND P. ALLIEZ, *Indoor Scene Reconstruction using Feature Sensitive Primitive Extraction and Graph-cut*, ISPRS Journal of Photogrammetry and Remote Sensing, 90 (2014), pp. 68–82.
- [118] S. OKANIWA, A. NASRI, H. LIN, A. ABBAS, Y. KINERI, AND T. MAEKAWA, *Uniform B-spline curve interpolation with prescribed tangent and curvature vectors*, IEEE Transactions on Visualization and Computer Graphics, 18 (2012), pp. 1474–1487.
- [119] W. PEDRYCZ AND S. CHEN, *Deep Learning: Concepts and Architectures*, Springer International Publishing, 2019.
- [120] R. PENROSE, *On best approximate solutions of linear matrix equations*, in Mathematical Proceedings of the Cambridge Philosophical Society, vol. 52, Cambridge University Press, 1956, p. 17–19.
- [121] G. PEYRE, *Toolbox graph - A toolbox to process graph and triangulated meshes*. Available at [http://www.ceremade.dauphine.fr/~sim\\$peyre/matlab/graph/content.html](http://www.ceremade.dauphine.fr/~sim$peyre/matlab/graph/content.html), 2007.
- [122] L. PIEGL AND W. TILLER, *The NURBS Book*, Springer Berlin, Heidelberg, 2nd ed., 1997.
- [123] M. POTENZIANI, M. CALLIERI, M. DELLEPIANE, M. CORSINI, F. PONCHIO, AND R. SCOPIGNO, *3DHOP: 3D heritage online presenter*, Computers & Graphics, 52 (2015), pp. 129 – 141.
- [124] F. POUX, C. MATTES, Z. SELMAN, AND L. KOBBELT, *Automatic region-growing system for the segmentation of large point clouds*, Automation in Construction, 138 (2022), p. 104250.
- [125] Y. QIE, L. QIAO, AND N. ANWER, *Enhanced invariance class partitioning using discrete curvatures and conformal geometry*, Computer-Aided Design, 133 (2021), p. 102985.

-
- [126] T. RABBANI SHAH AND F. VAN DEN HEUVEL, *Efficient Hough transform for automatic detection of cylinders in point clouds*, in ISPRS Laser Scanning 2005, G. Vosselman and C. Brenner, eds., ISPRS Working Groups, 2005, pp. 60–65.
- [127] A. RAFFO, O. J. BARROWCLOUGH, AND G. MUNTINGH, *Reverse engineering of CAD models via clustering and approximate implicitization*, Computer Aided Geometric Design, 80 (2020), p. 101876.
- [128] A. RAFFO, C. ROMANENGO, B. FALCIDIENO, AND S. BIASOTTI, *Fitting and recognition of geometric primitives in segmented 3d point clouds using a localized voting procedure*, Computer Aided Geometric Design, 97 (2022), p. 102123.
- [129] A. RAMANANTOANINA AND K. HORMANN, *New shape control tools for rational bézier curve design*, Computer Aided Geometric Design, 88 (2021), p. 102003.
- [130] R. J. RENKA, *Shape-preserving interpolation by fair discrete G^3 space curves*, Computer Aided Geometric Design, 22 (2005), pp. 793 – 809.
- [131] G. RICCA, M. C. BELTRAMETTI, AND A. M. MASSONE, *Piecewise recognition of bone skeleton profiles via an iterative Hough transform approach without re-voting*, in Medical Imaging 2015: Image Processing, vol. 9413, SPIE, 2015, pp. 706–713.
- [132] K. RODRIGUEZ ECHAVARRIA AND R. SONG, *Analyzing the decorative style of 3d heritage collections based on shape saliency*, Journal on Computing and Cultural Heritage, 9 (2016), pp. 1–17.
- [133] C. ROMANENGO, S. BIASOTTI, AND B. FALCIDIENO, *HT-based Recognition of Patterns on 3D Shapes Using a Dictionary of Mathematical Curves*, in Proceedings of the Conference on Smart Tools and Applications in Graphics 2019, (STAG 2019), M. Agus, M. Corsini, and R. Pintus, eds., The Eurographics Association, 2019, pp. 31–40.
- [134] C. ROMANENGO, S. BIASOTTI, AND B. FALCIDIENO, *HT-based identification of 3D feature curves and their insertion into 3D meshes*, Computers & Graphics, 89 (2020), pp. 105–116.
- [135] C. ROMANENGO, S. BIASOTTI, AND B. FALCIDIENO, *Recognising decorations in archaeological finds through the analysis of characteristic curves on 3d models*, Pattern Recognition Letters, 131 (2020), pp. 405 – 412.
- [136] C. ROMANENGO, S. BIASOTTI, AND B. FALCIDIENO, *Hough transform for detecting space curves in digital 3d models*, Journal of Mathematical Imaging and Vision, 64 (2022), p. 284–297.

- [137] C. ROMANENGO, E. BRUNETTO, S. BIASOTTI, C. E. CATALANO, AND B. FALCIDIENO, *Recognition, modelling and interactive manipulation of motifs or symbols represented by a composition of curves*, in Proceedings of the Conference on Smart Tools and Applications in Graphics 2020, (STAG 2020), S. Biasotti, R. Pintus, and S. Berretti, eds., Eurographics Association, 2020, pp. 27–35.
- [138] C. ROMANENGO, B. FALCIDIENO, AND S. BIASOTTI, *Hough transform based recognition of space curves*, Journal of Computational and Applied Mathematics, 415 (2022), p. 114504.
- [139] C. ROMANENGO, A. RAFFO, S. BIASOTTI, B. FALCIDIENO, V. FOTIS, I. ROMANELIS, E. PSATHA, K. MOUSTAKAS, I. SIPIRAN, Q.-T. NGUYEN, C.-B. CHU, K.-N. NGUYEN-NGOC, D.-K. VO, T.-A. TO, N.-T. NGUYEN, N.-Q. LE-PHAM, H.-D. NGUYEN, M.-T. TRAN, Y. QIE, AND N. ANWER, *Shrec 2022: Fitting and recognition of simple geometric primitives on point clouds*, Computers & Graphics, 107 (2022), pp. 32–49.
- [140] C. ROMANENGO, A. RAFFO, Y. QIE, N. ANWER, AND B. FALCIDIENO, *Fit4CAD: A point cloud benchmark for fitting simple geometric primitives in CAD objects*, Computers & Graphics, 102 (2022), pp. 133–143.
- [141] A. SAMPER, G. GONZÁLEZ, AND B. HERRERA, *Determination of the geometric shape which best fits an architectural arch within each of the conical curve types and hyperbolic-cosine curve types: The case of Palau Güell by Antoni Gaudí*, Journal of Cultural Heritage, 25 (2017), pp. 56 – 64.
- [142] A. SCALAS, D. CABIDDU, M. MORTARA, AND M. SPAGNUOLO, *Potential of the geometric layer in urban digital twins*, ISPRS International Journal of Geo-Information, 11 (2022).
- [143] D. SCHENONE, *Detection of space profiles by means of the hough transform technique*, master’s thesis, Università degli Studi di Genova, 2015.
- [144] R. SCHNABEL, R. WAHL, AND R. KLEIN, *Efficient RANSAC for Point-Cloud Shape Detection*, Computer Graphics Forum, 26 (2007), pp. 214–226.
- [145] L. SCHUMAKER, *Spline Functions: Basic Theory*, Cambridge University Press, 3rd ed., 2007.
- [146] T. SEDERBERG, D. ANDERSON, AND R. GOLDMAN, *Implicit representation of parametric curves and surfaces*, Computer Vision, Graphics, and Image Processing, 28 (1984), pp. 72 – 84.
- [147] G. SHARMA, B. DASH, A. ROYCHOWDHURY, M. GADELHA, M. LOIZOU, L. CAO, R. WANG, E. G. LEARNED-MILLER, S. MAJI, AND E. KALOGERAKIS, *PriFit: Learning to Fit Primitives Improves Few Shot Point Cloud Segmentation*, Computer Graphics Forum, 41 (2022), pp. 39–50.

- [148] G. SHARMA, D. LIU, S. MAJI, E. KALOGERAKIS, S. CHAUDHURI, AND R. MECH, *ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds*, in European Conference on Computer Vision, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, eds., vol. 12352, Springer, 2020, pp. 261–276.
- [149] L.-Y. SHEN, C.-M. YUAN, AND X.-S. GAO, *Certified approximation of parametric space curves with cubic b-spline curves*, *Computer Aided Geometric Design*, 29 (2012), pp. 648 – 663.
- [150] E. V. SHIKIN, *Handbook and atlas of curves*, CRC, 1995.
- [151] G. TAUBIN, *Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation*, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13 (1991), pp. 1115–1138.
- [152] M. TORRENTE, M. BELTRAMETTI, AND J. SENDRA, *r-norm bounds and metric properties for zero loci of real analytic functions*, *Journal of Computational and Applied Mathematics*, 336 (2018), pp. 375 – 393.
- [153] M.-L. TORRENTE AND M. C. BELTRAMETTI, *Almost vanishing polynomials and an application to the Hough transform*, *Journal of Algebra and Its Applications*, 13 (2014), p. 1450057.
- [154] M.-L. TORRENTE, S. BIASOTTI, AND B. FALCIDIENO, *Recognition of feature curves on 3D shapes using an algebraic approach to hough transforms*, *Pattern Recognition*, 73 (2018), pp. 111 – 130.
- [155] M. A. UY, Y.-Y. CHANG, M. SUNG, P. GOEL, J. G. LAMBOURNE, T. BIRDAL, AND L. J. GUIBAS, *Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2022), 2022, pp. 11850–11860.
- [156] L. VÁŠA, P. VANĚČEK, M. PRANTL, V. SKORKOVSKÁ, P. MARTÍNEK, AND I. KOLINGEROVÁ, *Mesh statistics for robust curvature estimation*, *Computer Graphics Forum*, 35 (2016), pp. 271–280.
- [157] R. J. WALKER, *Algebraic curves*, Springer Verlag, 1978.
- [158] L. XU AND J. SHI, *Geometric Hermite interpolation for space curves*, *Computer Aided Geometric Design*, 18 (2001), pp. 817 – 829.
- [159] D.-M. YAN, W. WANG, Y. LIU, AND Z. YANG, *Variational mesh segmentation via quadric surface fitting*, *Computer-Aided Design*, 44 (2012), pp. 1072 – 1082.

BIBLIOGRAPHY

- [160] S. YAN, Z. YANG, C. MA, H. HUANG, E. VOUGA, AND Q. HUANG, *HPNet: Deep Primitive Segmentation Using Hybrid Representations*, in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2021, pp. 2753–2762.
- [161] S. YOSHIKAWA, A. BELYAEV, H. YOKOTA, AND H.-P. SEIDEL, *Fast, robust, and faithful methods for detecting crest lines on meshes*, Computer Aided Geometric Design, 25 (2008), pp. 545–560.
- [162] C.-H. YU AND J. HUNTER, *Documenting and sharing comparative analyses of 3D digital museum artifacts through semantic web annotations*, Journal on Computing and Cultural Heritage, 6 (2013), pp. 1–20.