Universidad EAFIT

Thesis of Master in Science

Compendium on:

Computational Geometry Application in Dimensional Assessment and Boundary Elements

Master Student
Cristian Camilo Rendón Cardona

Supervisor
Prof. Dr. Ing. Oscar E. Ruiz Salguero
Laboratory of CAD CAM CAE
Universidad EAFIT

Co-supervisor
Dr. Ing. Jorge Eduardo Correa Panesso
Industry and Cloud Computing in Computer Graphics
Manufactura Cohesiva S.A.S.

Dissertation
Submitted in partial fulfillment of the requirements for the degree of Master of Engineering in the College of Engineering of Universidad EAFIT. This thesis results from joint research collaboration between Universidad EAFIT (Colombia) and Manufactura Cohesiva S.A.S.(Colombia).

UNIVERSIDAD EAFIT
COLLEGE OF ENGINEERING
MASTER PROGRAM IN ENGINEERING
MEDELLIN, COLOMBIA
JULY 2022

Dedication

To my friends and family.

Acknowledgments

Keep Ithaca always in your mind.
Arriving there is what you're destined for.
But don't hurry the journey at all. Better
if it lasts for years, so you're old by the
time you reach the island, wealthy with
all you've gained on the way, not
expecting Ithaca to make you rich. Ithaca
gave you the marvelous journey. Without
her you wouldn't have set out. She has
nothing left to give you now.

Constantine Peter Cavafy, 1911.
Translated by Edmud Keeley

To my mother, my father, and my sister: the people in my life that have always supported me. Without you and your love, this would not have been possible.

I want to express my gratitude to Prof. Dr. Ing. Oscar Ruiz Salguero for his guidance, patience, compromise, and constant attention. For making me not only a better professional but also, a better human being.

I am grateful to Dr. Jorge Correa, it has been a honor to work with you and I hope we keep creating incredible technology together.

I am sincerely thankful to Universidad EAFIT and Manufactura Cohesiva for helping and providing the funding for my research and my studies.

I am grateful to the whole team of Laboratory of CAD CAM CAE. Thank you for your friendship, help, and humor. All of you are marvelous people and deserve the best.

Contents

List of Figures

List of Tables

1 Introduction

Computational geometry plays a crucial role in digital manufacturing. Two-manifold representation of manufacturing elements presents several challenges. In many cases, the mesh conditioning of 3D surfaces produces massive datasets which are complex and slow to process. Digital collaborative environments require rapid and reliable technologies for detecting features and analytic forms in different datasets. Therefore, special interest is placed on heterogeneous and anisotropic statistically poor triangular meshes.

The Fluid Dynamics problem has always been a field of interest and research, providing numerous ways to predict it. Numerical simulations are one of the most common for the simulation of complex flows. The Boundary Element Method (BEM) is a broad method that can be applied to such a field. The BEM equations present singularities on their integrals that must be addressed.

This work presents a compilation of different contributions to the problems stated in this introduction. Sections 1.1 and 1.2 summarize the articles included in this compendium and list all the co-authors associated to each manuscript. Section 1.3 summarizes the industrial and academic research projects associated with the Master Thesis. Section 1.4 reports the distinctions of the student during his Master Thesis. Finally, Section 1.6 explains to the reader how to follow this document.

1.1 Summary of Manuscripts

In Table 1.1 are listed the published and submitted articles with their respective authorship and bibliographic information.

Table 1.1: List of manuscripts related to this Master Thesis.

Item	Bibliographic Information	Type / Status
1	Cristian Rendon-Cardona, Jorge Correa, Diego A. Acosta, Oscar Ruiz-Salguero. Algorithms, p-ISSN: 1999-4893, volume 14, issue 11, Publisher MDPI. url= https://www.mdpi.com/1999-4893/14/11/304, DOI: https://doi.org/10.3390/ a14110304, Received: 10 July 2021; Published: 22 October 2021. Indexed in Scopus (Q2), Scimago (Q3), Publindex (B).	Journal Article / Published.
2	Samuel Velez-Sanin, Cristian Rendon-Cardona, Juan Diego-Jaramillo, Nicolás Guarín Zapata, Oscar Ruiz Salguero. Computation, ISSN: 2079-3197, Journal webpage: https://www.mdpi.com/journal/computation/about.	Submitted to Journal.

1.2 List of Co-authors of this Compendium

The names and affiliations of the co-author of the articles presented in this compendium are listed in Table 1.2.

Table 1.2: Co-authors of this compendium.

Name	Affiliation
Prof. Dr. Eng. Diego Andrés Acosta Maya	Process Development and Design Research Group (DDP), Universidad EAFIT, Colombia
Dr. Eng. Jorge Eduardo Correa Panesso	Manufactura Cohesiva S.A.S., Medellín, Colombia
Eng. Samuel Vélez-Sanin	Laboratory of CAD CAM CAE, Universidad EAFIT, Colombia
Eng. Cristian Rendon-Cardona	Laboratory of CAD CAM CAE, Universidad EAFIT, Colombia
Prof. Dr. Eng. Juan Diego-Jaramillo	Applied Mechanics Laboratory, Universidad EAFIT, Colombia
Dr. Eng. Nicolás Guarín Zapata	Applied Mechanics Laboratory, Universidad EAFIT, Colombia
Prof. Dr. Eng. Oscar Ruiz Salguero	Laboratory of CAD CAM CAE, Universidad EAFIT, Colombia

1.3 Projects

The student executed the previously cited research in the context of industrial projects hosted by Manufactura Cohesiva S.A.S. The student was granted an 24-month research internship at Cohesiva in the context of the colaboration agreement between Manufactura Cohesiva and Universidad EAFIT for Master and PhD students. Part of this work is product of research conducted by the student during his undergraduate studies. The student participated in the following projects:

1. **Project: COHESIVE VIEWER 2020-2021**: development a rapid algorithm for segmentation of triangular meshes representing 2-manifolds. *Institutions:* Manufactura Cohesiva, Colombia; Laboratory of CAD CAM CAE, Universidad EAFIT, Colombia.

2. **Boundary Element Method**: implementation of the equations that tackle the boundary element problem in the field of Fluid Dynamics. *Institutions:* Laboratory of CAD CAM CAE, Universidad EAFIT, Colombia.

1.4 Distinctions

The following table presents a list of the universities where the student was accepted to continue his postgraduate studies. The acceptances were possible due to the abilities acquired by the student during his master's and the industrial internship.

Table 1.3: Universities acceptance of the student.

University	**Level of Education**	**Date**
Southern University of Denmark	Posgraduate Program	April 23rd 2021
Chalmers University of Technology	Posgraudate Program	April 7th 2022
University of Illinois at Urbana-Champaign	Postgraduate Program	April 17th 2020
University Paris-Saclay	PhD Program	June 17th 2022

1.5 Academic Impact

The table below lists the industrial and informatics abilities acquired by the effect of the master and the industrial internship.

Table 1.4: Informatics and industrial abilities

Abilities	**Period**	**Description**
Computational geometry for web technologies	August 2020 - June 2022	Development of computer graphics technologies applied in the web and cloud computing industry.
Servers security protocols	February 2021 - November 2021	Implementation servers protocol https and configuration of endpoints to effectively communicate with web applications.
Data structures and algorithms	November 2020 - June 2022	Necessary for all projects
Computational simulations	August 2021 - June 2022	Development of an engine for simulation of human-cloth interactions.

Table 1.5 presents the courses that provided the student with the necessary tools and knowledge to conduct the research projects.

Table 1.5: Relevant Courses

Course	**Semester**
IM0242 Introduction to CAD CAM Systems*	2017-1
IC0916 Introduction to the Boundary Element Method	2020-1
IC0717 Introduction to the Finite Elements Method	2020-1
IC0896 Mechanics of Advanced Continuous Media	2020-1
IC0920 Advanced Mathematics for Engineers	2020-1
IM0904 Optimization Techniques	2020-2
IM0912 Numerical Solution of Differential Equations	2020-2
IM0603 Geometric Modeling	2021-1
EI0813 Academic Writing and Research Skills	2021-1
ST1004 Interactive Visualization of Point Clouds	2021-2

*Although the IM0242 course was completed during the undergraduate studies of the student, it established the basis in computational geometry, programming and computer science for his master and the industrial internship.

1.6 How to Read this Document

This document presents the developments of a research executed at the Laboratory of CAD CAM CAE at Universidad EAFIT (Colombia), and Manufactura Cohesiva S.A.S. The results are a combination of data structures and algorithms, computational geometry, mathematics, and numerical simulations. **All the articles included in this compendium have been submitted peer-reviewed journals.**

Chapter 2 proposes a methodology to fit cylinders, spheres, and cones to statistically poor triangular meshes. The devised algorithm executes a segmentation of the input set and applies the respective fitting to the obtained submesh. There are two segmentation methods depending on the continuity of the submesh supporting the analytic form. C-0 continuity allows real-time execution. Otherwise, the segmentation and fitting are codependent.

Chapter 3 proposes a methodology to overcome the singularities present on the Green functions in the boundary and surface integrals when using Boundary Element Method in fluid dynamics simulations. The implemented methodology applies a displacement d to the source node and evalutes de integral when $d \rightarrow 0$. It shows a detailed description of the implementations and results of the simulation compared with the commercial software ANSYS.

Finally, relevant conclusions of this work as well as possible future improvements on this research are presented in Chapter 4.

2
Analytic Form Fitting in Poor Triangular Meshes

Cristian Rendon-Cardona[1,2], Jorge Correa[2], Diego A. Acosta[3] and Oscar Ruiz-Salguero[1].

[1] CAD CAM CAE Laboratory - Universidad EAFIT, Colombia

[2] Manufactura Cohesiva SAS, Medellín Colombia

[3] Grupo de Desarrollo y Diseño de Procesos (DDP) - Universidad EAFIT, Colombia

Context

Cristian Rendon-Cardona, Jorge Correa, Diego A. Acosta, Oscar Ruiz-Salguero. Algorithms, p-ISSN: 1999-4893, volume 14, issue 11, Publisher MDPI. url= https://www.mdpi.com/1999-4893/14/11/304, DOI: https://doi.org/10.3390/a14110304, Received: 10 July 2021; Published: 22 October 2021.

Abstract

Fitting of analytic forms to point or triangle sets is central to computer-aided design, manufacturing, reverse engineering, dimensional control, etc. The existing approaches for this fitting assume an input of statistically strong point or triangle sets. In contrast, this manuscript reports the design (and

industrial application) of fitting algorithms whose inputs are specifically poor triangular meshes. The analytic forms currently addressed are planes, cones, cylinders and spheres. Our algorithm also extracts the support submesh responsible for the analytic primitive. We implement spatial hashing and boundary representation for a preprocessing sequence. When the submesh supporting the analytic form holds strict C^0-continuity at its border, submesh extraction is independent of fitting, and our algorithm is a real-time one. Otherwise, segmentation and fitting are codependent and our algorithm, albeit correct in the analytic form identification, cannot perform in real-time.

2.1 Introduction

2.1.1 Research Target

This manuscript reports an algorithm for analytic form (cone, cylinder, sphere) submesh extraction and fitting, using as input low statistical quality triangular meshes. Our algorithm has been successfully applied in actual industrial low-quality datasets, when good quality ones are impractical due to their large size and processing times. Our method favors sequential small submesh probing and vector closure formulation for the identification of analytic form parameters. Our method avoids, as much as possible, multi-variable regression, which requires a statistically rich input.

2.1.2 Context

Point samples of a two-manifold (surface) in 3D are achieved by contact, ultrasound or optical scanning. Under Nyquist–Shannon conditions of this geometrical point cloud information, topological (i.e., connectivity) information may be inferred. A widely used result of the topology inference is the triangular mesh surface approximation. This manuscript addresses the next step of shape inference, namely the fitting of a analytic form $g()$ to a subset of the triangular mesh M.

This manuscript is concerned with triangular meshes whose facets strongly depart from equilateral, isotropic, and homogeneous size conditions. The reason for this choice is that in many cases, mesh conditioning is not only labor-expensive but is produces massive triangle sets. Fitting analytic form to heavy datasets is prohibitively slow for many industrial applications.

A strong demand exists in industrial applications for planar, cylindrical, conical and spherical forms, identified in low-quality triangle sets. Therefore, this manuscript addresses those surface types, tessellated with such low-quality triangles. The implicit form ($g(x, y, z) = k$) is pursued. In this manuscript, $g()$ equally notes the parametric or implicit forms, as well as the defining feature set of the form, as follows:

a For cylinders: axis pivot point p_0, axis direction $\hat{v}$, radius R.

b For cones: axis pivot point p_0, axis direction $\hat{v}$, cone angle γ.

c For spheres: center pivot point p_0, radius R.

This manuscript addresses the following problem:

Given:

1 $M(\mathcal{T}, \mathbf{P})$: triangular mesh representing a shell with manifold properties.

2 $t_s \in \mathcal{T}$: seed triangle (selected by the user).

3 $\mathbb{T}$: type of analytic form (selected by the user from the options cylinder, sphere, and cone).

Goal:

i S: largest connected subset of M, formed by triangles from M, containing t_s and fitting the analytical form of $\mathbb{T}$.

ii $g()$: analytic form of the chosen primitive, fit to subset S.

where the analytic form is expressed by the identified parameters in (a)–(c) above.

It is important to note that the continuity level (C^0, C^1 or higher) at the borders ∂S triangle subset S and $M-S$ has a dramatic impact on the decoupling of solutions for items (i) and (ii) above. If the surface M has only C^0-continuity at border ∂S, this fact determines S, without influence from the fitting stage. Then, the fitting of $g()$ proceeds in real time. On the other hand, if M presents continuity C^1 or higher at border ∂S, there is a codependency between S and $g()$. S can be extended only as far as it produces a good quality fit $g()$. Low-quality $g()$ fitting indicates that the subset S includes mistaken triangles. Those triangles must be eliminated from S, other triangles might be included, and a new guess for $g()$ is computed. This iteration is obviously more expensive than the one in which identification of S does not depend on the quality of $g()$ fitting.

This manuscript is organized as follows: Section 2.2 reviews the existing literature and argues our claim for novelty in our initiative. Section 2.3 explains the methodology followed. Section 2.4 conveys the results of our implementation and compares it in some aspects against similar competitor approaches. Section 2.5 concludes the manuscript and discusses relevant future endeavors.

2.2 Literature Review

This section executes a taxonomy on the prevailing methods for fitting of analytic forms to point of triangle sets. The available literature may be divided into (1) stochastic, (2) parameter-space, and (3) mesh segmentation fitting methods. This taxonomy is based on the one described by Ref. [2].

2.2.1 Stochastic Methods

These methods randomly segment the input (point or triangle) set. They apply local regressions with alternative goal primitives (cylinder, cone, sphere, etc.), keeping the one with the best fit. The methods allow for the evolution of the primitive type and its parameters as well as evolution of the partition of the input mesh into local support submeshes. The methods require a dense, isotropic, homogeneous input set.

Ref. [3] requires point clouds with surface normal information. It fits cones and cylinders by solving local linear regressions. It intends to keep a small support point set, at the expense of actually producing a collection of guesses for the user to choose from. Ref. [4] presents analytical fitting of cones, cylinders and ellipsoids from dense, noisy point clouds. Ref. [4] finds an initial guess for the analytical surface by calculating their characteristic variables. Then, tuning of the analytic form parameters takes place by minimizing the point vs. analytic surface accumulated distances.

2.2.2 Mapping to Parameter Space

In these methods, type $\mathbb{T}$ of the analytic form is assumed. For a given point sample support set, a regression is performed, thus finding an estimate of the n parameters which that particular form type has. A point in the parameter space R^n is then located with such values. As other neighborhoods of the input point set are used as support sets, a point population of R^n arises, with clusters in the most likely parameter combinations. These high-density clusters are identified with standard statistical tools and the parameters of the analytic form are determined. Mapping to parametric space is particularly expensive when computing (storage), sharply growing with n, the dimension of the R^n parameter space. They also require dense input point sets. Due to these reasons, these methods are mostly applied to identification of polygonal flat faces in a polytope.

Ref. [5] uses the Hough transform as mapping to parametric space to identify planes. Ref. [6] lowers the high cost of detecting cylinders (parameter space R^6) by breaking the six parameter sets (i.e., 6D Hough transform) into axis vector, axis pivot, and cylinder radius and carrying sequential lower-dimension Hough transforms.

2.2.3 Mesh Segmentation Fitting Methods

In these methods [7–9], the priority is to segment the input (point-normal vector set or point graph). The fitted analytic form is a by-product. These methods create input clusters using diverse strategies: fitting-segmentation iteration, grouping using X-means or mean shift (separate segmentation), or classification by ranges of additional information (scalar or vector fields) such as color, depth, abrupt dihedral angle, etc. These methods have limitations as they need at least one of the additional pieces of information mentioned above and also require dense datasets.

Ref. [7] describes a hierarchical segmentation of dense point clouds. This method generates the k-nearest graph by merging edges of the input connectivity graph. This merging uses as criteria a penalty function related to the quality of fitting of the point clusters to a given primitive (one of sphere, cone, plane, torus). Ref. [8] applies segmentation-fitting iterations seeking to simultaneously segment the full input set and to fit quadratic analytic primitive options (plane, sphere, cylinder, ellipsoid, paraboloid) to the support subsets. It executes an initial segmentation of the input dataset before the iterative process. Ref. [8] does not provide detailed information on such a preprocessing. Ref. [9] executes a color-based segmentation of indoor 3D depth map scenarios. Then, a matching is conducted between the generated patches and a template database of predefined 3D indoor furniture models. Finally, the user refines the segmentation. The authors train a matching algorithm of random-regression forest to obtain the desired result. The process of matching the support set S to various reference models is an expensive task and is out of our scope since we do not need to train any learning algorithm. Ref. [10] executes a rough segmentation of a dense good quality point cloud or triangular mesh. This segmentation seeks to find neighborhoods of the input data which present *slippage* compatibility, i.e., which can slide onto themselves (e.g., spheres, cylinders, planes). The segmentation is carried out by using a greedy clustering, based on the local slippage compatibility. This reference aims for an approximate segmentation. It does not determine the $g()$ analytic form, nor presents execution times or complexity information.

2.2.4 Conclusions of Literature Review

The literature reveals that most fitting techniques require statistically dense datasets complying with the Nyquist–Shannon theorem. Additionally, these input datasets must include supplementary information (i.e., scalar maps or vector fields). Therefore, our approach offers (1) fitting of analytic surfaces to statistically poor triangulations as the industry demands it. (2) When the continuity and the border ∂S is exclusively C^0, we provide real-time performance. (3) When the border ∂S presents C^1 or higher continuity with $M - S$, segmentation and fitting are codependent. The fitted form $g()$ guarantees the perfecting of the support subset S. In turn, the support subset S provides the grounds for the $g()$ fitting. Table 2.1 presents a summary of the different approaches with their advantages and drawbacks.

Table 2.1: Competitor approaches vs. current approach.

Approach	**Refs.**	**Advantages**	**Disadvantages**
Stochastic Methods. Randomly apply local regressions of alternative goal primitives and keeping the best fit.	[3, 4, 11–13]	(1) Robust to outliers; (2) application-specific	(1) Requires dense, isotropic, homogeneous input dataset.
Mapping to Parameter Space. Implement a mapping to the parameter space, identify the number of parameters needed and search for clusters of similar parameters.	[5, 6, 14, 15]	(1) Robust to noise and outliers; (2) work with missing data	(1) High computational cost; (2) requires dense datasets
Mesh Segmentation Fitting Methods. Create clusters of the dataset using different grouping techniques. Fitting can occur during or after the segmentation.	[7–10, 16–18]	(1) Spatial consistency	(1) Sensitive to noise and outliers; (2) require supplementary information in the dataset.
Our approach. A primitive fitting in poor triangular meshes. Segmentation driven by dihedral angle is implemented when continuity is strict C^0. Segmentation and fitting are codependent for C^1 or higher continuity.		(1) Fitting in poor datasets; (2) low computational cost; (3) real-time performance for C^0-continuity; (4) perfecting of S for C^1 or higher continuity.	(1) Requires the type of analytic form intended to be fitted as an input.

2.3 Methodology

Fitting of an analytic primitive $g()$ to the support (triangle) subset S requires (obviously) the determination of S. Extraction of support subset S from the input set M is heavily based on triangle neighborhood interrogations on M. The acceleration of neighborhood queries on M was implemented by building the triangular boundary representation of the triangle set. This construction was itself accelerated by applying a spatial hashing process on input M (Spatial Hashing based on techniques from [19, 20]). Spatial hashing and B-Rep construction are reported in Section 2.3.1 on preprocessing. Section 2.3.2 discusses the specific problem of inferring the analytic form $g()$ from a triangle set S. Sections 2.3.3 and 2.3.4 address the determination of triangle subset S for applying Section 2.3.2. Section 2.3.4 focuses on cases where the border between support subset S and $M - S$ is sharp (C^0-continuity), thus allowing identification of S by using continuity information alone. Section 2.3.4 addresses more difficult cases where continuity at the border between S and $M - S$ is C^1 or better. In these cases, S is legitimated by a good fitting $g()$, but at the same time, $g()$ depends on S. In these cases, fitting ($g()$) and segmentation (S) are codependent and cannot be executed independently from each other.

Preprocessing

The process of extracting and identifying analytic forms from a manifold triangle set (Figure 2.1) is preceded by a preprocessing stage encompassing: (1) a spatial hashing; (2) boundary representation (B-Rep) construction.

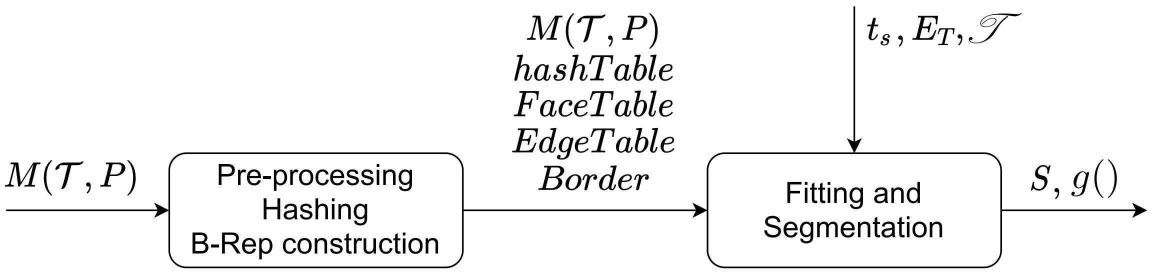


Figure 2.1: Overall primitive extraction process.

Analytic Form Fitting

If the triangle subset S supporting the analytic form has C^0 -continuity alone at the border ∂S between S and M_S, the determination of S is exclusively based on the dihedral angle at ∂S (red box, left portion of Figure 2.2). Otherwise, if the continuity at ∂S is C^1 or higher, the identification of support subset S and the fitting of $g()$ on that S are codependent (blue box, right portion of Figure 2.2). Triangles are added to S, or rejected, according to their effect on the fitting quality of the analytic form $g()$. Growth of the support subset S stops when all triangles at border ∂S have an adverse effect on $g()$.

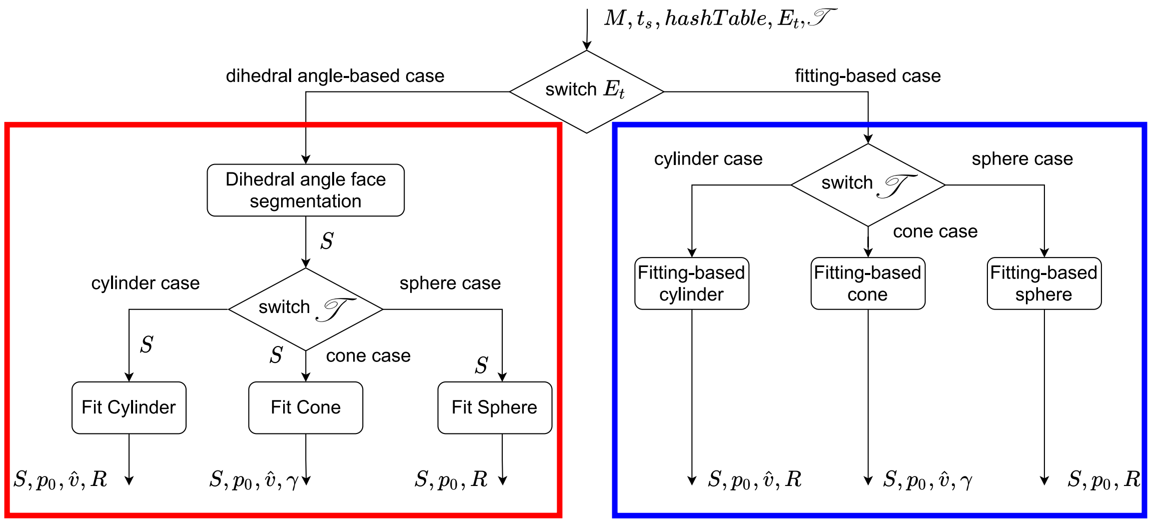


Figure 2.2: Fitting process. **Left** (red) block: segmentation/fitting decoupled. Real-time identification of $g()$. **Right** (blue) block: codependent segmentation and fitting.

2.3.1 Preprocessing

Spatial Hashing

Spatial hashing on the input triangle set M is used in this work for accelerating the construction of explicit triangle neighborhood information (boundary representation). The I/O specification of the spatial hashing preprocessing follows.

Input:

1 $M(\mathcal{T}, \mathbf{P})$: Triangle (two-manifold) set with geometry $\mathbf{P}$ and topology $\mathcal{T}$.

2 Ω: Rectangular prism in R^3, orthogonally oriented w.r.t. coordinate axes, equipped with a regular grid of voxels with dimension $delta_x$, $delta_y$, $delta_z$.

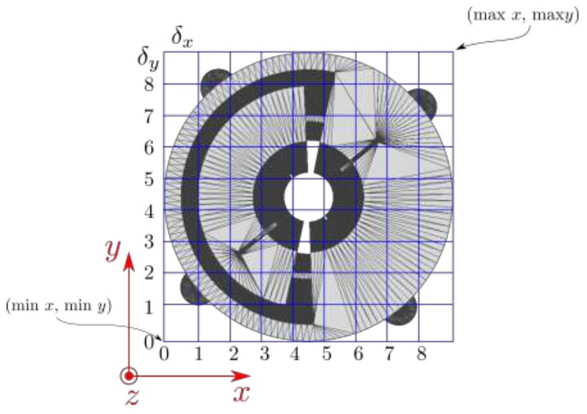
3 N_v : A predefined number of voxels per dimension of Ω.

Output:

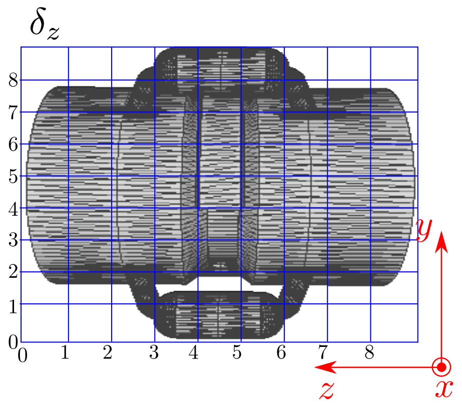
1 h: Hashing function: $h : P \to N \times N \times N$, which maps each vertex of the input triangle set M into the (i,j,k) indices of the voxel $v_{ijk} \subset \Omega$, which contains such a vertex.

2 H: $\Omega \to T^2$, which maps each voxel v_{ijk} of Omega to the set of triangles of M which contain a vertex inside voxel v_{ijk}. 2^T denotes the power set of T (all sets made with triangles from T).

The algorithm divides the prism Ω into a set of rectangular voxels. These voxels are represented by a three-dimensional array where each cell contains the indices of the triangles with vertices in that voxel. Figure 2.3 shows a graphical representation of the voxel subdivision for an input triangle set.



(a)



(b)

Figure 2.3: (**a**) Bounding box and (**b**) rectangular voxel subdivision.

The hash function $[i, j, k] = h(p_i)$ maps a point $p_i \in \Omega$ to a position $[i, j, k]$ in the hash table H.

$$\begin{aligned} i &= floor((p_i.x - \Omega.x_{min})/\delta_x), \\ j &= floor((p_i.y - \Omega.y_{min})/\delta_y), \\ k &= floor((p_i.z - \Omega.z_{min})/\delta_z). \end{aligned} \tag{2.1}$$

where $\Omega.x_{min}$, $\Omega.y_{min}$, $\Omega.z_{min}$ are constants corresponding to the minimal positions of the prism Ω. δ_x, δ_y, δ_z are the x, y, and z voxel side dimensions.

Figure 2.4 represents a block a diagram of the algorithm and how the vertices are stored in the hash table. As the input triangles sets are representation of two-manifold shells, H is a sparse array.

Notice that the spatial hashing $[H, h]$ ensures that all triangles incident on (EDGEs of) a given triangle are stored in at most three voxels of the hash array H, and they are directly accessible via the function $h()$. Due to this reason, the spatial hashing accelerates the construction/weaving of the boundary representation of the input triangle set M.

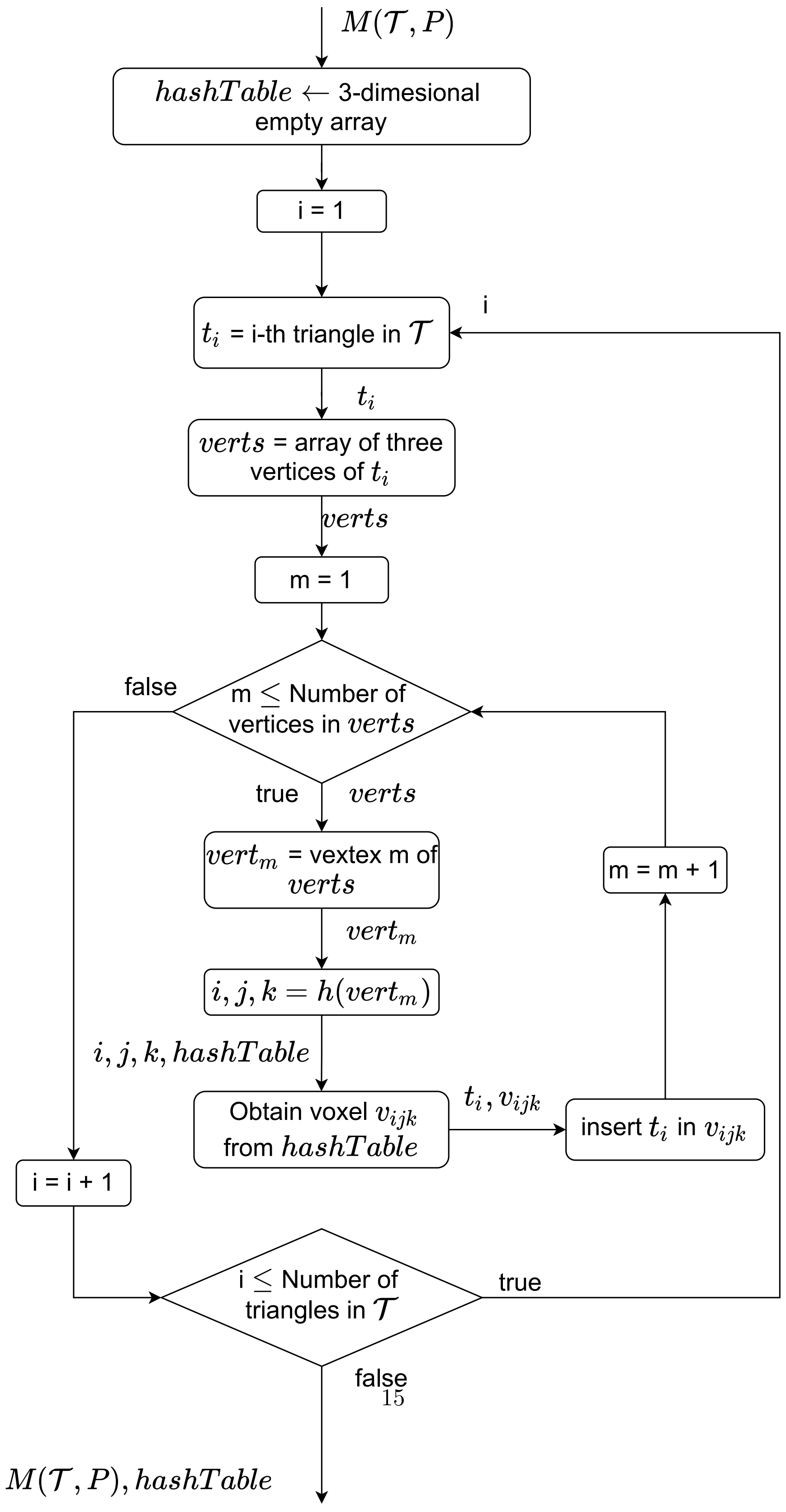

$M(\mathcal{T},P)$
$hashTable \leftarrow$ 3-dimesional empty array
i = 1
t_i = i-th triangle in $\mathcal{T}$
i
t_i
$verts$ = array of three vertices of t_i
$verts$
m = 1
false
m $\leq$ Number of vertices in $verts$
true
$verts$
$vert_m$ = vextex m of $verts$
m = m + 1
$vert_m$
$i,j,k = h(vert_m)$
$i,j,k,hashTable$
Obtain voxel v_{ijk} from $hashTable$
t_i, v_{ijk}
insert t_i in v_{ijk}
i = i + 1
i $\leq$ Number of triangles in $\mathcal{T}$
true
false
$M(\mathcal{T},P), hashTable$

Figure 2.4: Spatial hashing preprocessing. The hashing is executed before the fitting process starts.

Boundary Representation Construction

After the spatial hashing process, a boundary representation, B-Rep, of the input triangle set M is constructed. We use a variation of the half-edge data structure [21]. The construction of the B-Rep produces the FACEs, EDGEs, VERTEXs and BORDER tables (see Tables 2.2–2.5).

Input:

- A two-manifold triangular mesh $M(\mathcal{T}, P)$.

Output:

- Boundary representation for M (Tables 2.2–2.5).

Table 2.2: VERTEX table.

Vertex	x	y	z	**Edge**
1	2.34	3.00	1.12	13
2	2.22	9.00	10.36	27
3	5.20	4.00	9.12	53

Table 2.3: FACE table constructed from B-Rep.

Face	**Edge 1**	**Edge 2**	**Edge 3**
1	1	2	3
2	3	5	4
3	3	1	4

Table 2.4: EDGE table. If counter-EDGE cell is empty, indicates border.

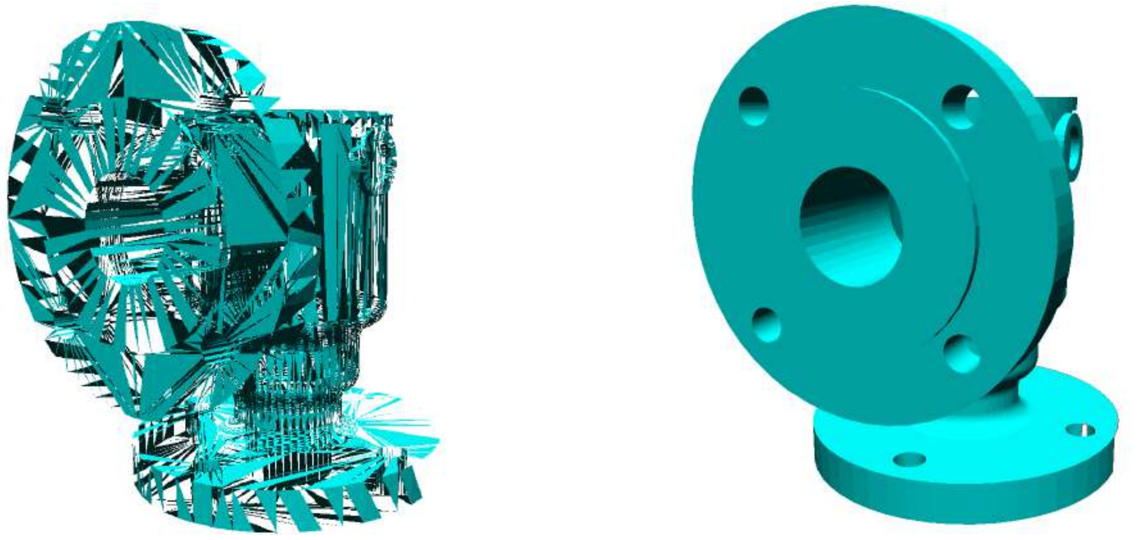
Edge	**Vertex 1**	**Vertex 2**	**Counter-EDGE**	**Face**
1	1	2	4	1
2	2	3	5	1
3	3	4	-	1

Table 2.5: BORDER table.

Border Loop	**Vertex 1**	**Vertex 2**	**Vertex 3**	**Vertex 4**	**Vertex 5**	**Vertex 6**	...
1	4	7	8	91	<End-of-loop>		
2	10	71	31	11	1	90	<End-of-loop>
3	63	83	47	121	9	<End-of-loop>	

In the present case, the SHELL table is omitted because the input mesh is connected (i.e., there is only one SHELL). The B-Rep guarantees that all FACEs within a SHELL have consistent orientation. The B-Rep delivers constant time expenses $O(k)$ in the following interrogations:

(a) EDGES incident on a VERTEX, (b) FACEs incident on an EDGE, (c) FACEs neighboring a FACE, (d) ordered LOOPs of a SHELL border, (e) SHELL owner of a FACE, (f) EDGEs of a LOOP, (g) LOOPs of a FACE. Figures 2.5 and 2.6 present plain-sight visible effects of building the B-Rep of two input triangle sets M. Those visible effects are the consistent orientation of the FACEs in a SHELL and the identifications of the SHELL BORDER.



(a) Raw triangle set M. Disoriented triangles become transparent.

(b) B-Rep woven from the raw triangle set M. No disoriented triangles are present.

Figure 2.5: Input triangle set M with inconsistent FACE orientation. Output B-Rep of M. (**a**) and (**b**) displayed with culling based on the triangle orientations.

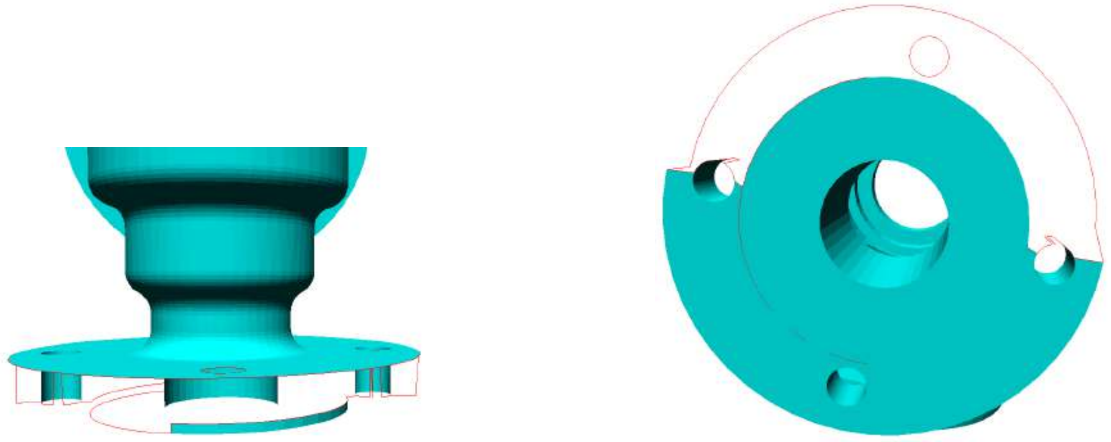

(a) (b)

Figure 2.6: Result of border identification in the triangle set. (**a**) and (**b**) present the identified border of the triangle set after weaving the B-Rep.

2.3.2 Primitive Fitting from Triangle Set

This section addresses the determination of the parameters of a given analytic form (cylinder, cone, sphere) of type $\mathbb{T}$, which the user wants to fit to a triangle set S. The strategy used in our algorithm is to locally *probe* the triangle set, progressively determining parameters of the form (radius, center, axis, apex, etc.). This strategy was chosen instead of *statistical* fitting, which requires statistically meaningful (dense, large, homogeneous, isotropic) triangle set inputs. Our implementation does use quadratic form fitting for the purpose of determining minimal and maximal curvature directions on specific and small neighborhoods of the triangle set S.

The fitting of the $g()$ analytic form to set $S \subset M$ is indifferent to the manner in which triangle subset S is determined. The description of the fitting follows.

Input:

1 S: triangle support set ($S \subset M$) to fit the analytic form $g()$.

2 $\mathbb{T}$: primitive type, selected by the user from the set {Cylinder, Sphere, Cone}

Output:

1 $g()$: The analytic form of the chosen primitive, fit to subset S.

For each analytic form, we implement an approach for the identification of its parameters.

Cylinder Fitting

The geometrical parameters to define the cylinder are:

1 p_0: axis pivot point.

2 $\hat{v}$: axis direction.

3 R: radius.

Our algorithm takes advantage of the fact that the input triangle set provides information on the vectors locally normal to the cylinder. This information enables the calculation of p_0 and $\hat{v}$.

Axis point p_0

The axis pivot point is calculated as follows:

1 Create a set of lines $L_n = \{\ell_1, \ell_2, ..., \ell_n\}$, where ℓ_i is a line generated by the baricenter b_i and the normal n_i of the triangle t_i.

2 Obtain C_p: as the set containing the points of the minimum distance between each pair of lines from L_n. Every line ℓ_i is approximately perpendicular to, and nearly intersects, the cylinder axis $[p_0, \hat{v}]$. Therefore, C_p is a point sample of the axis $[p_0, \hat{v}]$.

3 Compute p_0 as the geometric center of gravity of C_p.

Figure 2.7 presents an axial view of a cylindrical triangle set. Lines $[b_i, n_i]$ ideally intersect, and are perpendicular to, the cylinder axis $[p_0, \hat{v}]$. Figure 2.8 displays an actual example set C_p and cylinder axis identification.

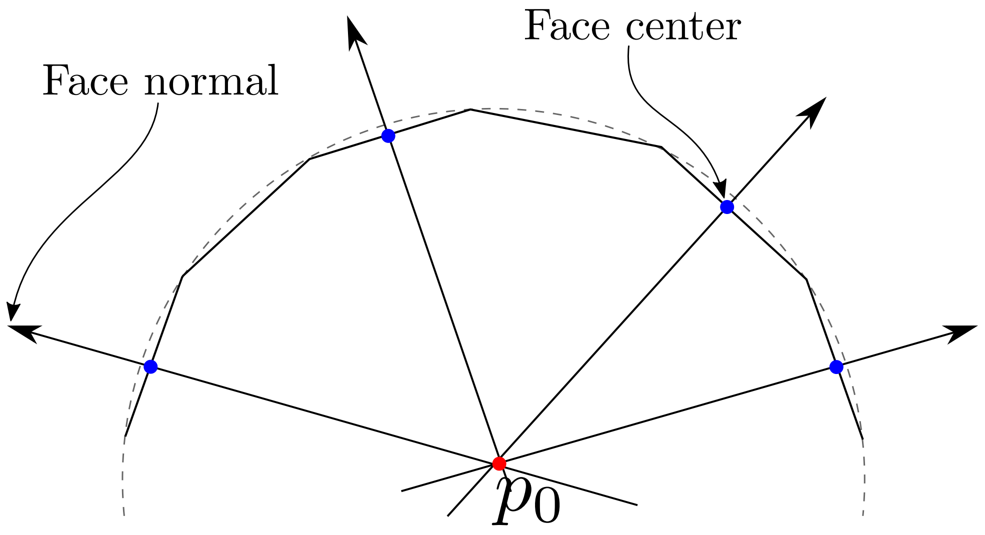


Figure 2.7: p_0 calculation with the intersection of lines $[b_i, n_i]$ whose directions are triangle normal vectors n_i and pass through triangle baricenters b_i.

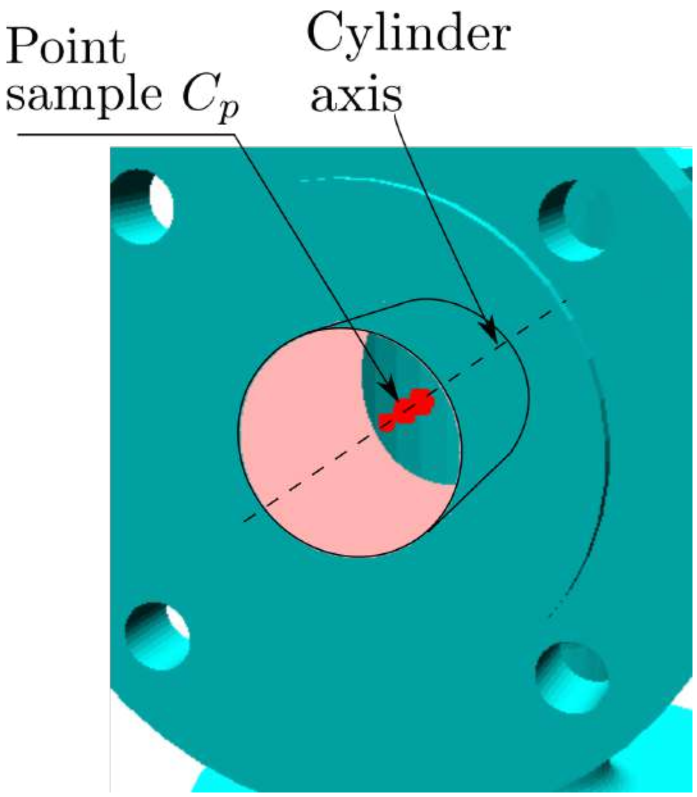


Figure 2.8: Set of points C_p approximating the cylinder axis.

Cylinder Axis direction $\hat{v}$

To calculate the cylinder axis, the method relies on the ability to:

1 Translate the unitary vectors n_i normal to the triangles from their supposed pivot point (triangle baricenter b_i) to the origin.

2 Collect the tips of the origin-based vectors resulting from step (1). These tip points form a tilted flat disk d centered at the origin (Figure 2.9).

3 Obtain (via Principal Component Analysis (PCA)) the cylinder axis direction $\hat{v}$ as the one of smallest dispersions of the points of disk d (Figure 2.9).

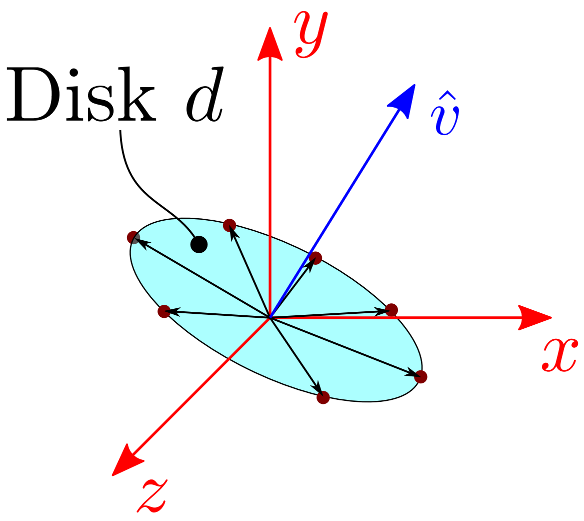


Figure 2.9: Cylinder fitting. Coplanar circular set of tips of cylinder normal unitary vectors, applied at origin. $\hat{v}$: direction of minimal dispersion of disk d, cylinder axis direction.

Cylinder Radius R

Knowing the cylinder axis $[p_0, \hat{v}]$, the radius R of the cylinder is determined as follows:

1 Rigidly move the triangle set S so its axis $[p_0, \hat{v}]$ maps to the Y_w World axis and the geometric center of S maps to the origin O_w.

2 Project the point set resulting from (1) onto the $X_w Z_w$ plane, resulting in a flat circular point set.

3 Compute the radius of such a circular point set.

Cone Fitting

The geometrical parameters of the cone are:

1 p_0: cone apex.

2 $\hat{v}$: axis direction.

3 γ: opening angle.

Cone axis direction $\hat{v}$

As in the cylinder case (Section 2.3.2), the vectors normal to the cone triangles allow us to estimate the direction of the cone axis.

The unitary vectors n_i normal to the cone triangles are translated to the origin (Figure 2.10). Their tips form a planar circle which is the base of a dual cone whose apex is at the origin. Observe that this dual cone *is not* the one contained in triangle input S. This *dual* cone in Figure 2.10 is obtained by using the normal vectors in the triangle set S as cone generatrices, but it has the same axis direction vector as the sough cone $g()$.

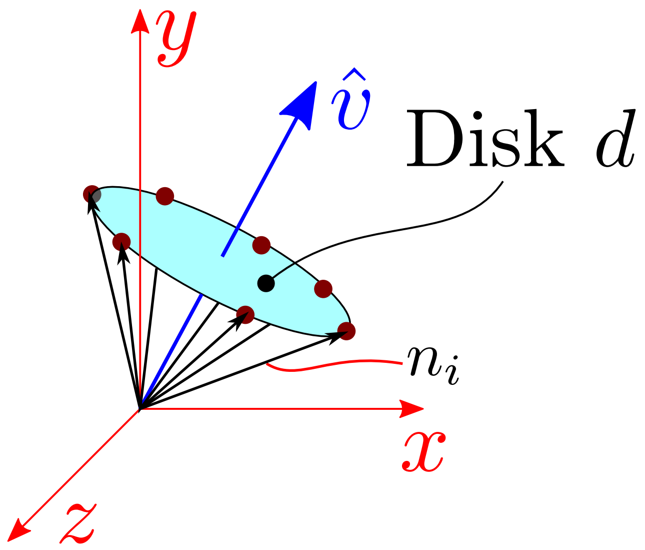


Figure 2.10: Cone fitting. Dual cone formed with Origin as apex. Unitary vectors normal to S are generatrices for the dual cone. Origin-located tails of n_i vectors normal to triangles t_i at Origin. Normal vector tips contained in a plane form a flat disk d. Minimal dispersion direction (principal component analysis) of disk d is the direction of cone axis $\hat{v}$.

Cone apex p_0

The straight line $[b_i, \hat{e_i}]$ (Figure 2.11) tangent to triangle t_i passes through the triangle baricenter b_i and has direction $\hat{e_i}$ (i.e., direction of the cone generatrix at b_i). The point at which the lines $[b_i, \hat{e_i}]$ are nearest to (or intersect) each other are the elements of C_p. The cone apex p_0 is the center of gravity of C_p.

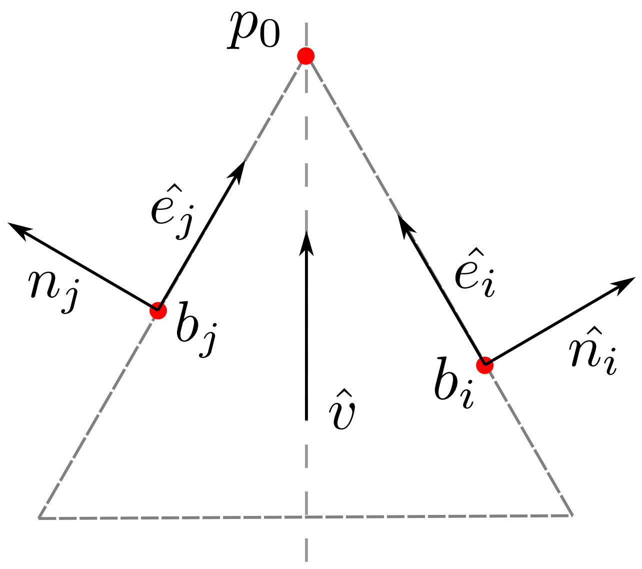


Figure 2.11: Cone 2D example. Vectors e_i point to p_v and are perpendicular to n_i.

Each set of vectors $[n_i, \hat{e}_i, \hat{v}]$ lies in a plane that intersects the cone and contains the apex p_0. We calculate a temporal vector *temp* which is normal to such a plane as the cross product between n_i and $\hat{v}$. The cross product between the *temp* vector and n_i guarantees that $\hat{e}_i$ is perpendicular to n_i and lies in the aforementioned plane. Equation (2.2) summarizes the process.

$$\begin{aligned} temp &= n_i \times \hat{v} \\ e_i &= temp \times n_i \end{aligned} \tag{2.2}$$

Opening Angle γ

The opening angle of the cone is computed as the average of the angles γ_i (Equation (2.3)) between the cone axis direction $\hat{v}$ and the calculated vectors $\hat{e}_i$.

$$\gamma = \frac{1}{m} \sum_i \gamma_i. \tag{2.3}$$

Sphere Fitting

The geometrical parameters to define the sphere are:

1 p_0: sphere center.

2 R: radius.

Sphere center p_0

We consider a sphere which is faceted by a manifold set S of triangles t. The straight line $[b_1, n_i]$ normal to triangle t_i passes through the triangle baricenter b_i. This line has direction n_i. The points at which all lines $[b_i, n_i]$ are nearest to (or intersect) each other are the elements of set C_p (Figure 2.7) and probabilistically lie at the sphere center p_0 . The center of the sphere p_0 is estimated to be the center of gravity of C_p.

Sphere Radius *R*

The radius of the sphere is calculated by computing the mean distance from p_0 to all the vertices of the triangle subset.

$$R = \frac{1}{m}\sum_{i}^{m} ||p_i - p_0||. \tag{2.4}$$

Conclusions of Fitting Section

This section presents methods based on *progressive probing* rather than *statistical fitting* to estimate the $g()$ parameters of the analytic cones, cylinders, and spheres which has been faceted with manifold sets S of irregular, anisotropic, heterogeneous triangles. The extraction of the $g()$ support triangle subset $S \subset M$ follows.

2.3.3 Extraction of Subset S Based on Dihedral Angle

This method for extracting from M the triangle subset S ($\subset M$) which contains the seed triangle t_s, is applicable when the EDGE border between S and $M - S$ holds only C^0- continuity (i.e., ∂S is a sharp border). In this case, (a) S is indifferent to the $\mathbb{T}$ type of analytic form sought and (b) the extraction of S precedes and is independent from the $g()$ fitting process.

In this manuscript, we say that two EDGE-sharing triangles $t_i, t_j \in M$ hold only C^0-continuity if their dihedral angle is above a threshold α_{max}. The dihedral angle $\angle(t_i, t_j)$ between triangles t_i and t_j is the one between their normal vectors (under a uniform LOOP traversal direction). We measure the dihedral angle in the interval $[0, 180)$ degrees. Notice that $\angle(n_1, n_2) = 180$ implies that M would be non-manifold.

The first method for extraction of the triangle subset S is based on the dihedral angle between the triangles of M. This method is employed when the target segment of the triangulation holds strict C^0-continuity with respect to the other segments of M. This extraction process is indifferent to the type $\mathbb{T}$ of the analytic form and occurs separately from the fitting process as shown in Figure 2.2 (red).

Input:

1 $M(\mathcal{T}, \mathbf{P})$: triangle (two-manifold) set with geometry $\mathbf{P}$ and topology $\mathcal{T}$.

2 t_s: selected seed triangle from $\mathcal{T}$.

3 α_{max}: maximum angle value for dihedral angle-based extraction.

4 $\mathbf{E}_\alpha()$: dihedral angle equivalence relation defined on M.

Output:

1 S: the equivalence class (or block) containing t_s, of M under the equivalence relation $\mathbf{E}_\alpha()$.

Definition. Equivalence Relation $\mathbf{E}_\alpha()$

The triangles are equivalent under $\mathbf{E}_\alpha()$ if there is a triangle sequence (Figure 2.12) $L = [t_0, t_1, ..., t_f]$ such as:

1 $L \subset M$

2 Any two consecutive triangles of L, t_i and t_{i+1} share an EDGE.

3 $\angle(t_i, t_{i+1}) \ < \alpha_{max}$

The equivalence character of relation $\mathbf{E}_\alpha()$ is not proved here. However, it is easy to verify that $\mathbf{E}_\alpha()$ is symmetric, reflexive and transitive. The reason we show this small digression here is that the $g()$ support subset $S \subset M$ containing the seed triangle t_s is computable (when S and $M - S$ hold only C^0-continuity at ∂S) by using well known algorithms for transitive closure.

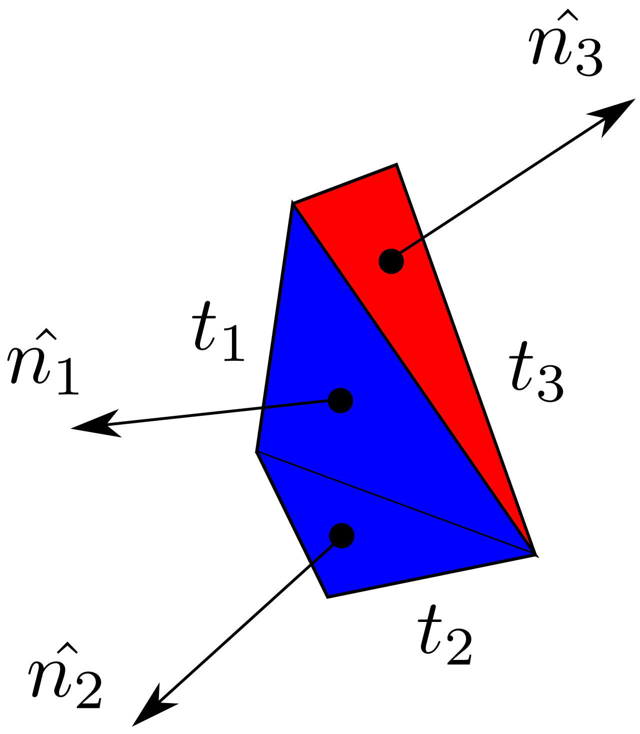


Figure 2.12: Dihedral angle equivalence $\mathbf{E}_\alpha()$. $\angle(n_1, n_2) \ \to 0$ and $\mathbf{E}_\alpha(t_1, t_2)$ (blue equivalence class). $\angle(n_1, n_3) \ > \alpha_{max}$ and t_3 (red) is outside the equivalence class (blue) of t_1 and t_2.

The boundary representation of input triangle set M supports FACE, VERTEX, EDGE neighborhood interrogations in constant time. Therefore, the dihedral angle-based extraction process of the $g()$ support subset S presents low computational cost and fast execution times.

2.3.4 Extraction of Subset S Based on Fitting Quality

The dihedral angle-based extraction presents limitations when the input triangle set presents C^1 or higher continuity at border ∂S. The extraction of the $g()$ support subset S depends on the quality of the $g()$ fitting and on the type $\mathbb{T}$ of the analytical form as follows.

Input:

1 $M(\mathcal{T}, \mathbf{P})$: Triangle (two-manifold) set with geometry $\mathbf{P}$ and topology $\mathcal{T}$.

2 t_s: selected seed triangle from $\mathcal{T}$.

3 $\mathbb{T}$: type of analytic form (selected by the user from the options cylinder, sphere, cone).

Output:

1 S: Largest connected subset of M, formed by triangles from M, containing t_s and fitting the analytical form of type $\mathbb{T}$.

2 $g()$: The parameters to define the analytic form that fits S.

Figure 2.13 presents a diagram with the general structure of the fitting-based extraction algorithm. The quality of $g()$ and the extent of the S support subset impact on each other at every iteration. The algorithm in Figure 2.13 assumes the existence of an initial set S, its border triangles $B(S)$ (in $M - S$) and an initial estimation for the analytic form $g()$. This basic S is formed by the seed triangle (marked by the user) and its neighbor triangles in M. The initial form $g()$ is estimated as per Section 2.3.2. $B(S)$ contains the triangles in $M - S$ that encircle S.

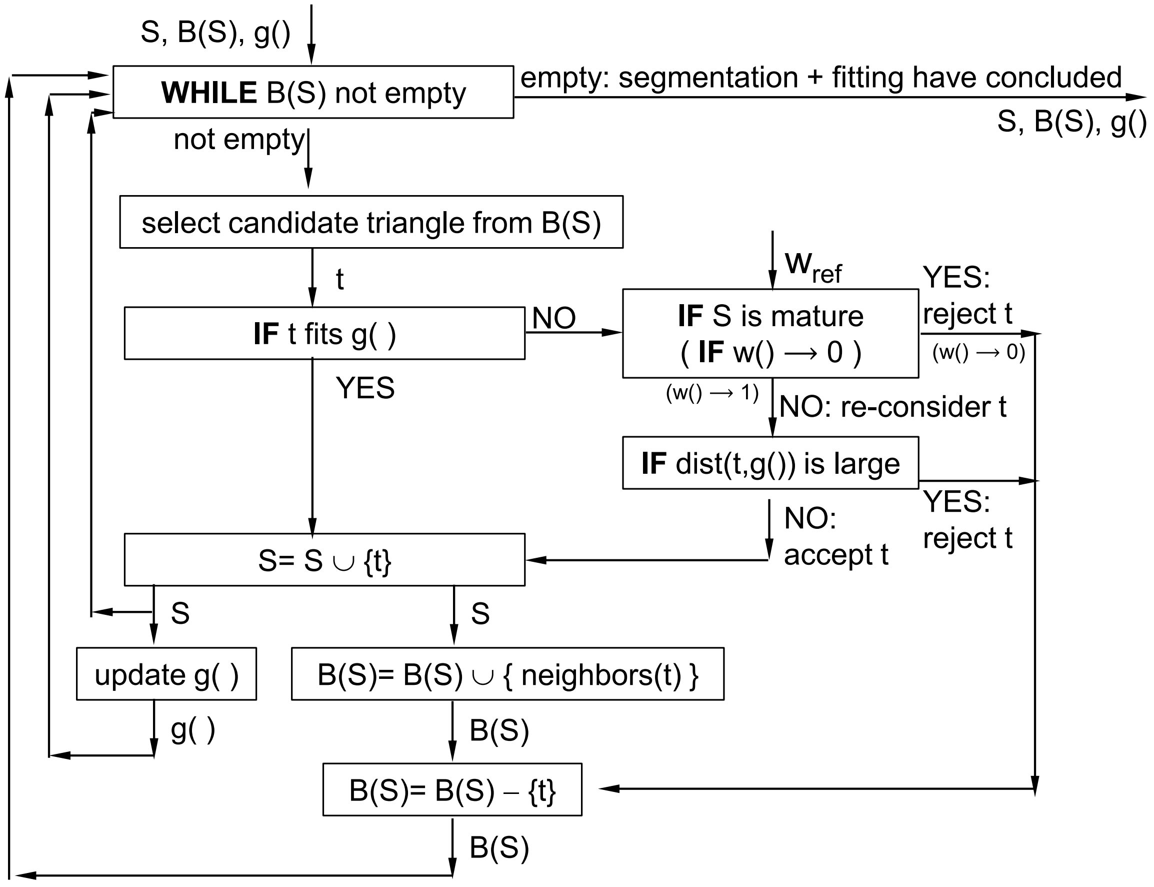


Figure 2.13: Algorithm for simultaneous fitting of $g()$ and extraction of $g()$ support subset S.

In Figure 2.13, the process/box "select candidate triangle from $B(S)$" selects a triangle in $M-S$ adjacent to the border triangles of current S to be screened for its belonging to the currently fitted analytic form $g()$. Figure 2.14 addresses this task. Its discussion appears after explaining the main algorithm of Figure 2.13.

In Figure 2.13, $B(S)$ denotes a set of triangles which encircle S, in $M - S$, and represent an opportunity to expand S (i.e., they do not differ from $g()$). The algorithm acts as long as $B(S)$ is not empty. A triangle t from the $B(S)$ border set is chosen (Figure 2.14, discussed later). If t strongly fits $g()$, it is accepted in S. If t does not fit $g()$ in a clear manner, the validity of S is examined. If S is a "mature" set ($w() \to 0$, Figure 2.15, discussed below), $g()$ it is supposed to be reliable and t is rejected. If the set S is "immature" ($w() \to 1$, t is re-assessed: if its distance to $g()$ is large, it is rejected. If it is near $g()$, it is accepted in S. If t is accepted in S, then it is removed from $B(S)$ and its neighbors in $M - S$ are included in $B(S)$. Additionally, $g()$ is re-computed against the new, augmented S.

If a triangle t is rejected, it is removed from $B(S)$. In this case, obviously, its neighbors in $M-S$ will not be considered to expand S.

We use a $w() : R^+ \to [0, 1]$ heuristic function (Figure 2.15) as a measure of the immaturity of

S. Let $u = N_S/N_T$ be the ratio between the instant size of S and a threshold number of triangles N_T, which is considered the size of a "mature" S. The function $w(u)$ satisfies:

i $w(0) = 1$ (S is immature, has few triangles).

ii $w(u) = 0$ for all $u \geq 1.0$ (S is mature, has many triangles).

iii $w(u)$ is monotonically decreasing in $\mathbb{R}^+$

Figure 2.14 presents an expansion of the process/box "select candidate triangle from $B(S)$ " in Figure 2.13, as follows: (1) A triangle t of the triangle border of S ($B(S)$) is selected. (2) The neighbors of triangle t outside S are determined, forming set N_t. (3) From the set N_t, triangles are eliminated whose dihedral angle with t is large (i.e., form sharp edges with t). (4) From the remaining eligible neighbors of triangle t, N_t, the direction u_{max} of largest curvature is determined. (5) The triangle t_s, neighbor of t, is selected, which materializes this large curvature (i.e., direction u_{max} from t). This algorithm has an implicit bias of considering that expansion of the support triangle subset S in the direction of largest smooth curvature will speedily lead to a meaningful fitting of $g()$. Thus, the evolving of $g()$ would be most efficient. Eventually, all feasible triangles around t are included in the expansion and tested accordingly. It is important to emphasize that the segmentation/fitting algorithm (Figure 2.13), both support subset S and analytic primitive $g()$ results.

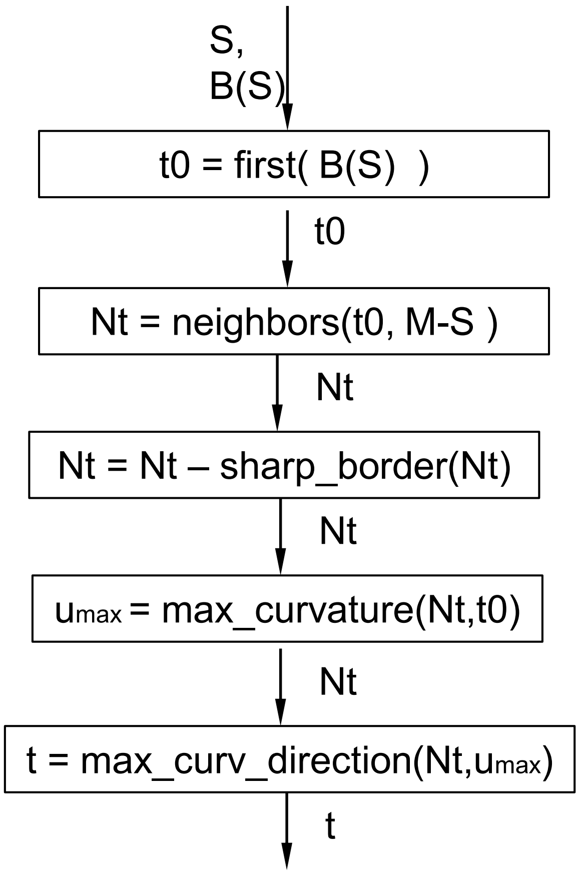

S,
B(S)
t0 = first(B(S))
t0
Nt = neighbors(t0, M-S)
Nt
Nt = Nt – sharp_border(Nt)
Nt
umax = max_curvature(Nt,t0)
Nt
t = max_curv_direction(Nt,umax)
t

Figure 2.14: Expansion of box/process "select candidate triangle from $B(S)$" from Figure 2.13.

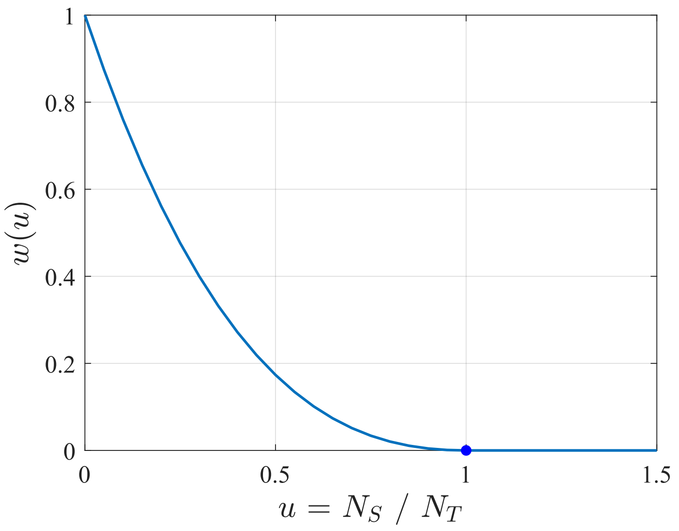


Figure 2.15: S immaturity function $w()$. $w \to 1$: S is immature set, with few triangles ($N_S \to 0$). $w \to 0$: S is mature set, with many triangles ($N_S > N_T$). N_T = threshold in number of triangles to consider $g()$ as stable analytic form.

This codependent fitting-segmentation process probes and expands the $g()$ support subset $S \subset M$, authorized by the quality of the $g()$ fitting (Section 2.3.2). Due to this codependency, this process is slower than the dihedral angle-based identification of S.

Fitting-Based Extraction of Cylinders and Cones

Cones and cylinders are ruled surfaces. As a consequence (Figure 2.16), at each point of them, there is a direction u_{min} of $K_{min} = 0$ minimal curvature (i.e., generatrix direction). As always, the direction of K_{max} maximal curvature u_{max} is normal to the u_{min} and both span the plane tangent to the surface at such a point.

Our algorithm aims to fit a reliable $g()$ analytic form to the support triangle subset S as early as possible. Triangles in the direction of maximal curvature u_{max} (obviously) provide information on the cone or cylinder radii in addition to generatrices. Due to this reason, S primarily grows in the direction u_{max} (box "select candidate from $B(S)$" in Figures 2.13 and 2.14). However, it is important to notice that S must eventually encompass the whole set of triangles supporting $g()$.

The process to calculate the local curvature at a given seed triangle t_s (with normal vector n and baricenter b) is described below.

1 Build a homogeneous coordinate frame S_i:

$$S_i = \begin{bmatrix} e_1 & e_2 & \hat{n} & b \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

with e_1, e_2 spanning the plane tangent to surface M at b and making $[e_1, e_2, n]$ a Special Orthogonal SO(3) (3×3) submatrix.

2 Apply a coordinate transformation T to rigidly move t_s and support subset S to the World Coordinate System S_w with n and b mapped to direction z and point $[0,0,0]'$, respectively (Figure 2.16).

$$\begin{aligned} T\,S_i &= S_w, \\ T &= S_i\,S_w^{-1}. \end{aligned} \tag{2.5}$$

3 Fit a quadratic surface (Equation (2.6)) to $T * S$ by multi-variable linear regression.

$$f(x,y) \;= a + bx + cy + dxy + ex^2 + fy^2. \tag{2.6}$$

4 Find the minimum and maximum curvature directions v_{max}, v_{min} with the Hessian eigenvalues and eigenvectors.

$$\begin{aligned} H(f) \;&= \begin{bmatrix} 2e & d \\ d & 2f \end{bmatrix}, \\ H[v_{max}\,v_{min}] &= [v_{max}\,v_{min}] \begin{bmatrix} K_{max} & 0 \\ 0 & K_{min} \end{bmatrix}. \end{aligned} \tag{2.7}$$

5 Convert direction v_{max} v_{min} back to global coordinates, with:

$$[u_{max}\,u_{min}] = T^{-1}[v_{max}\,v_{min}], \tag{2.8}$$

where, for the sake of simplified notation, we use the same symbols for 2D, 3D and 3D homogeneous vectors. Notice that T^-1 does not involve inverting a matrix. Instead, as T is an SO(3) rotation plus translation, the *transpose* of the rotation is its inverse.

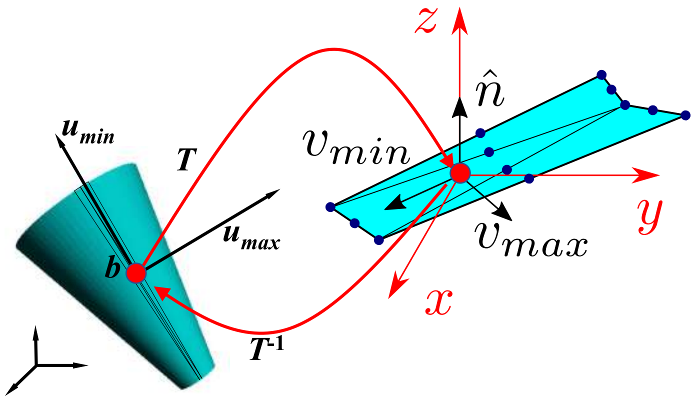


Figure 2.16: Rigid mapping M of local tangent plane of a ruled surface S onto the XY plane for the purpose of computing maximal and minimal curvature directions on S.

Fitting-Based Extraction of Spheres

This section corresponds to the cases in which the primitive $g()$ and triangle support subset S to extract from M correspond to a sphere making a smooth blend with $M - S$. The fitting and extraction algorithm appears in Figure 2.13 and is commented on in Section 2.3.4; at this point, we only add some particularities of the box "select candidate triangle from $B(S)$" from Figure 2.13.

This section uses the concept of a *topological circle* $C(t, k)$ and *topological ball* $B(t, k)$ defined on a triangular mesh M. $C(t, k)$ consists of the triangles of M that have distance k with respect to a "center" triangle t. A distance k between any triangles t_a and t_b of M corresponds to the minimal number of neighboring triangles in M to be traversed to travel from triangle t_a to triangle t_b, or vice versa. For this definition, two triangles are considered to be a distance of 1 apart if they share an EDGE or a VERTEX. The topological ball $B(t, k)$ includes all triangles of M whose topological distance to triangle t is less than or equal to k. Figure 2.17 presents a graphical example of the topological circle $C(t_s, 2)$ and topological ball $B(t_s, 2)$.

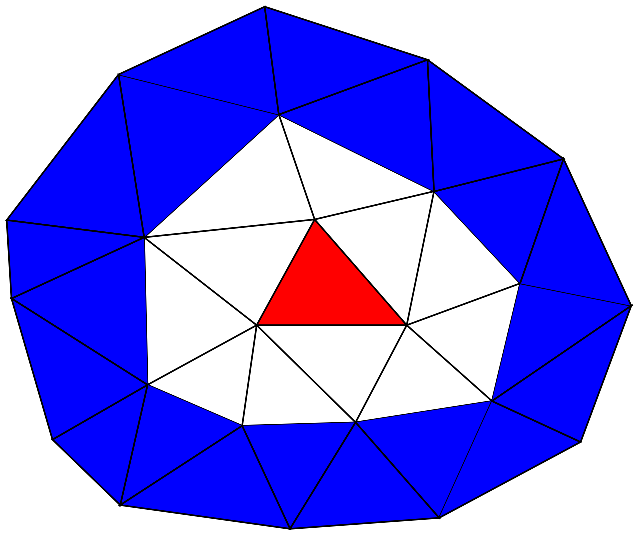


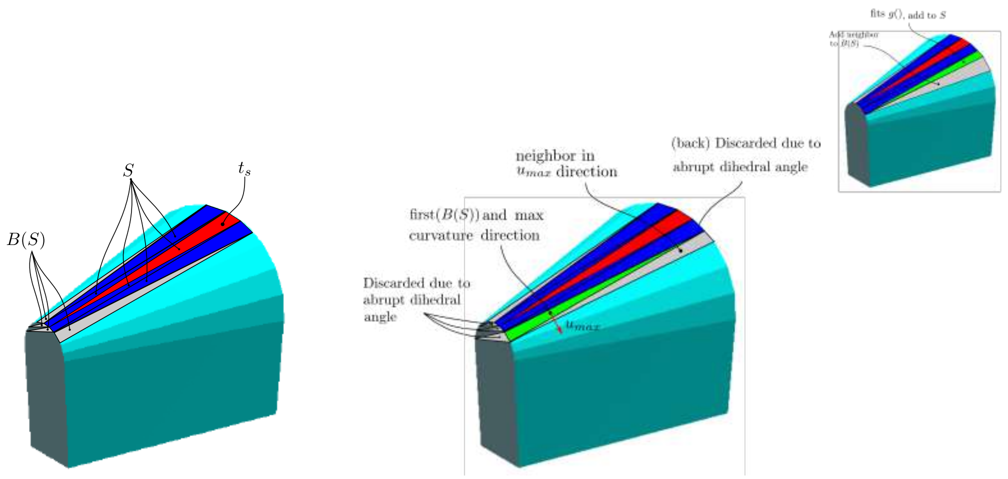
Figure 2.17: Red: user input seed triangle ($C(t_s, 0)$). Blue: topological circle $C(t_s, 2)$. White: topological circle $C(t, 1)$. Red + White + Blue: topological ball $B(t_s, 2)$. The initial guess for the analytic form $g()$ is computed with $C(ts, 0) \cup C(ts, 2)$.

2.4 Results

Sections 2.4.1–2.4.3 present support subset extraction and fitting for forms cylinder, cone and sphere, respectively. In each section, (1) dihedral angle-based and (2) fitting-based support subset S extractions are presented, along with the $g()$ analytic forms fitted to S. Section 2.4.5 presents a complexity analysis comparison of our approach vs. competitor approaches. Section 2.4.6 discusses the real-time application of our implementation, achieved via spatial hashing and boundary representation preprocessing.

Figure 2.18 presents an execution of the fitting-extraction algorithm of Figures 2.13 and 2.14. The execution fits and extracts a cone S from a triangle set M in which the cone sector S keeps

C^1-continuity with the remaining $M - S$ subset, and therefore the dihedral angle criteria are not able to identify S. Figure 2.18a displays the current S triangle set containing the seed triangle t_s (red) and already accepted neighboring triangles (blue). Figure 2.18b shows the grey boundary of S, $B(S)$. The bases of the cone contain triangles of $B(S)$ are not considered to be part of the cone due to their abrupt dihedral angle at the border of S, ∂S. The green triangle is detected in the direction of maximal curvature u_{max}. This green triangle fits into the current $g()$ estimation. Figure 2.18c shows that the green triangle is added to the support subset S, extracted from the boundary ($B(S)$) and its neighbors (grey) are added to $B(S)$, ready for the next iteration. The final result of the S extraction and $g()$ estimation is shown in Section 2.4.2.


S
t_s
$B(S)$
neighbor in u_{max} direction
(back) Discarded due to abrupt dihedral angle
first($B(S)$) and max curvature direction
Discarded due to abrupt dihedral angle
u_{max}
fits $g()$, add to S
Add neighbor to $B(S)$

(a) Initial support set S and $B(S)$.

(b) S expansion in u_{max} direction.

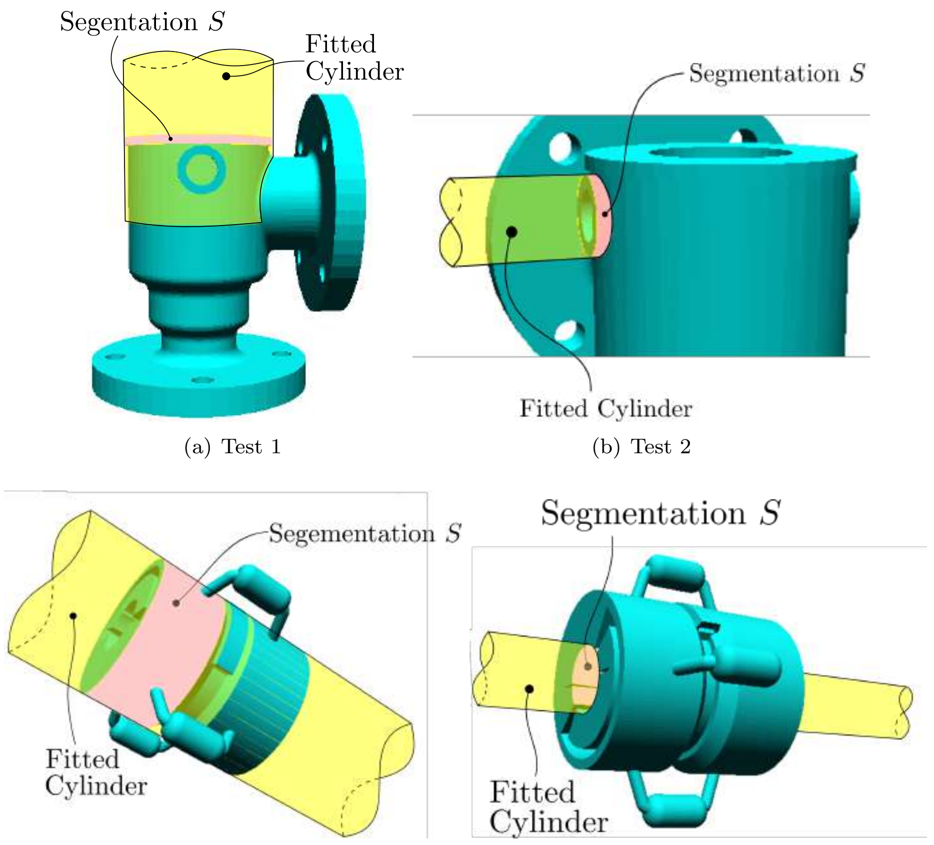
(c) Inclusion of accepted triangle in S and its neighbor in $B(S)$.

Figure 2.18: Step-wise execution of the S extraction $g()$ fitting algorithm, applied to a cone which does not accept S extraction based on dihedral angles.

2.4.1 Cylinder Fitting

Cylinder. Dihedral Angle-based Support Subset S Extraction

Figure 2.19 presents results for $g()$ cylinder fitting when the $g()$ support subset S has only C^0-continuity with the rest $(M - S)$ of the input set M. S is extracted at sharp dihedral angle EDGEs. As $g()$ support subset S extraction is independent of $g()$, the extraction and fitting are real-time processes (taking less than 1 s).



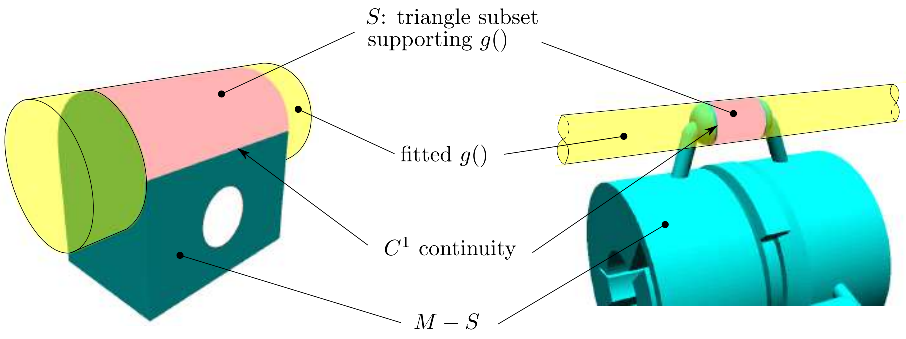
(a) Test 1 (b) Test 2

(c) Test 3 (d) Test 4

Figure 2.19: Cylinder fit. S extraction based on dihedral angles. Pink: extracted S support subset. Yellow: fitted $g()$.

Cylinder. Fitting-based Support Subset S Extraction

Figure 2.20 displays cases in which the support subset S for $g()$ holds C^1-continuity or higher with $M - S$. In these cases, $g()$ fitting and $g()$ support subset S extraction take place simultaneously.



(a) Test 5 (b) Test 6

Figure 2.20: Cylinder fit. Fitting-based S extraction. Pink: extracted S support subset. Yellow: fitted $g()$.

Cylinder. Fitting performance

Table 2.6 presents the numerical results for the test on cylinder extraction, comparing the preset parameters against the parameters obtained by the fitting. Subindex fit indicates the variable obtained via fitting, as opposed to the experimental preset value.

An interesting result is that for the more difficult cases (fitting-based extraction), the estimated $g()$ is more accurate than the one achieved in the simpler cases (C^0-continuity between S and $M - S$). This result shows that the algorithm discussed in Figures 2.13 and 2.14, albeit more complex, renders a better accuracy.

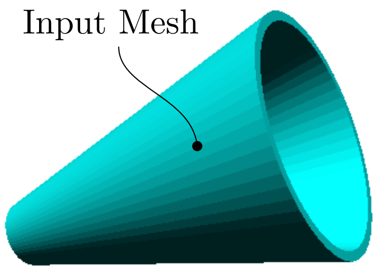
Dihedral angle-based extraction cases (1–4) perform in real time, while fitting-based extraction cases (5–6) do not. The former are faster not only because extraction and fitting are independent from each other, but also because $g()$ is fitted to small triangle subsets of S. These triangle subsets are scattered to make reasonable even though fast estimations.

Table 2.6: Cylinder fit. Set vs. found parameters. Tests 1–4: S extraction based on dihedral angles. Tests 5–6: Fitting-based S extraction.

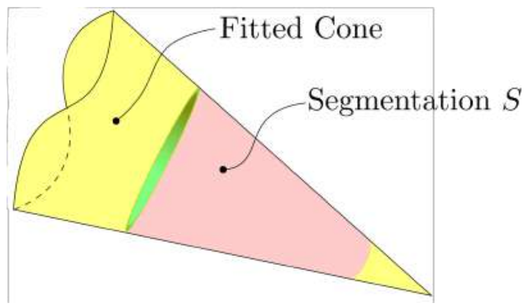
	$\boldsymbol{R}$	$\boldsymbol{R_{fit}}$	$\hat{\boldsymbol{v}}$	$\hat{\boldsymbol{v}}_{\boldsymbol{fit}}$	$\angle(\hat{\boldsymbol{v}}, \hat{\boldsymbol{v}}_{\boldsymbol{fit}})$ **Degrees**
Test #1	47	46.9999	$\begin{bmatrix} -0.6104 \\ 0.6169 \\ 0.4967 \end{bmatrix}$	$\begin{bmatrix} -0.6104 \\ 0.6169 \\ 0.4968 \end{bmatrix}$	$0°$
Test #2	14	13.9972	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.9999 \\ 0.0008 \\ 0.0003 \end{bmatrix}$	$0.0497°$
Test #3	13	13.00004	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0.00003 \\ -0.0002 \\ 0.9999 \end{bmatrix}$	$0.0136°$
Test #4	5	5.0037	$\begin{bmatrix} 0.4968 \\ -0.1904 \\ 0.8467 \end{bmatrix}$	$\begin{bmatrix} 0.4967 \\ -0.1903 \\ 0.8468 \end{bmatrix}$	$0.0051°$
Test #5	20	20.0000	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$0°$
Test #6	2.5	2.4999	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$0°$

2.4.2 Cone Fitting

Figures 2.21 and 2.22 illustrate the results for cone fitting with dihedral angle-based and fitting-based extraction, respectively. Table 2.7 presents numerical results comparing the actual vs. fitted $g()$ parameters.

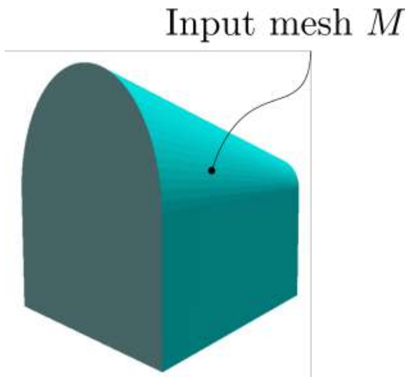


(a)

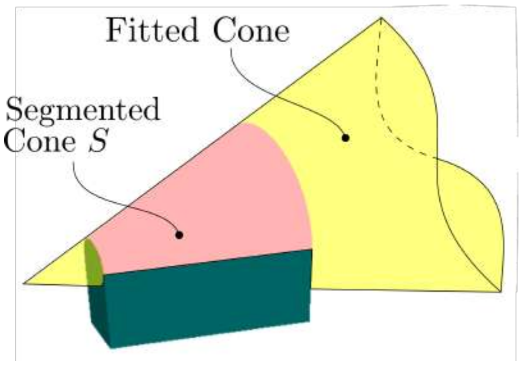


(b)

Figure 2.21: Test 1. Cone fitting test using dihedral angle-based extraction. Pink: segmentation. Yellow: cone fitting.



(a)



(b)

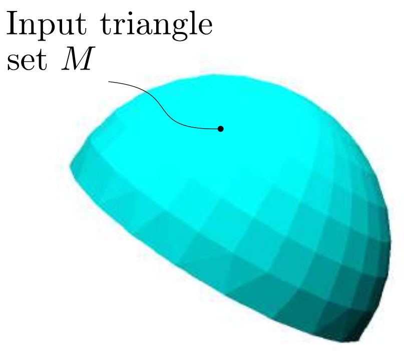
Figure 2.22: Cone Fit. Fitting-based S extraction. Pink: extracted S support subset. Yellow: fitted $g()$.

Table 2.7: Cone fit. Set vs. found parameters. Test 1: S extraction based on dihedral angles. Test 2: fitting-based S extraction.

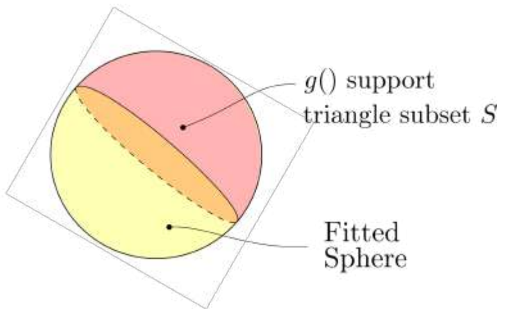
	$\boldsymbol{A_p}$	$\boldsymbol{A_{p\,fit}}$	γ	γ_{fit}	$\hat{\boldsymbol{v}}$	$\hat{\boldsymbol{v}}_{fit}$	$\angle(\hat{\boldsymbol{v}}, \hat{\boldsymbol{v}}_{fit})$ **degrees**
Test #1	$\begin{bmatrix}0\\0\\144\end{bmatrix}$	$\begin{bmatrix}0.0010\\0.0063\\144.1274\end{bmatrix}$	13.87	13.8852	$\begin{bmatrix}0\\0\\1\end{bmatrix}$	$\begin{bmatrix}0.0000\\0.0000\\0.9999\end{bmatrix}$	0.0033°
Test #2	$\begin{bmatrix}0\\0\\72.27\end{bmatrix}$	$\begin{bmatrix}-0.1258\\-3.0241\\72.2727\end{bmatrix}$	19	18.9716	$\begin{bmatrix}0\\0\\1\end{bmatrix}$	$\begin{bmatrix}-4.57\times10^{-11}\\-5.96\times10^{-9}\\1\end{bmatrix}$	0.0033°

2.4.3 Sphere Fitting

For the sphere fitting, the dihedral angle-based extraction does not produce a segmentation of the input mesh M. Therefore, $S = M$ and the fitting occurs over M. Figure 2.23 presents the results for that particular case. However, the parameters obtained by the fitting are accurate (see Table 2.8).



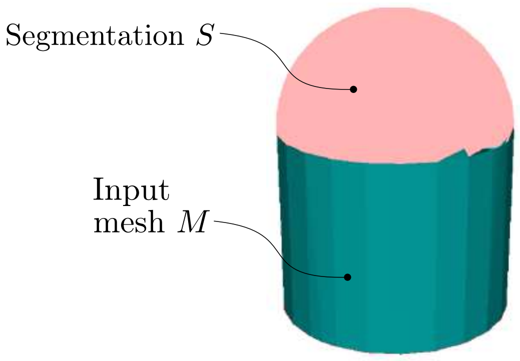
(a) Sphere sample $r = 35$



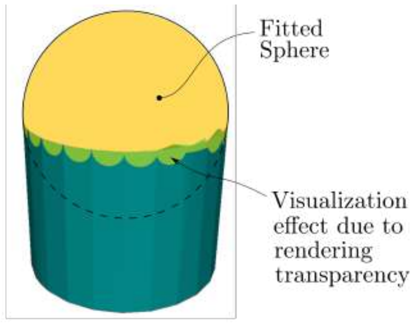
(b) Sphere fit $r_fit = 2.99$

Figure 2.23: Sphere fitting using dihedral angle-based extraction. Pink: segmented sphere. Yellow: fitted sphere. Notice that $S = M$.

Figure 2.24 illustrates the results for a fitting-based extraction in a sphere coupled with a cylinder and presenting C^1-continuity at the cylinder–sphere border.



(a) Input mesh. Sphere



(b) Segmentation and sphere detection.

Figure 2.24: Sphere fit. Fitting-based S extraction. Pink: extracted S support subset. Yellow: fitted $g()$. The process leaves out of S few sphere triangles at the C^1 border with $M - S$. However, it succeeds in rejecting inclusion in S of triangles actually belonging to the cylinder.

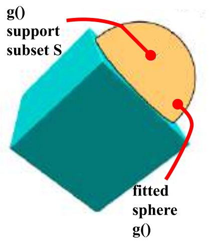
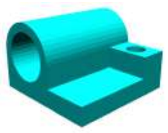
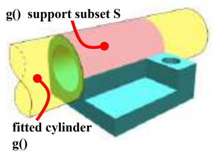
Table 2.8: SPHERE Fit. Set vs. found parameters. Test 1: S extraction based on dihedral angles. Test 2: fitting-based S extraction.

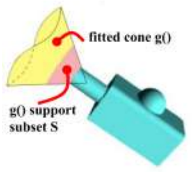
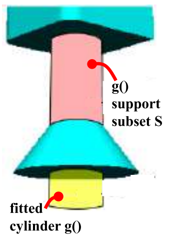
	$\boldsymbol{R}$	$\boldsymbol{R_{fit}}$	$\boldsymbol{p_0}$	$\boldsymbol{p_{0fit}}$
Test #1	3	2.9752	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}-0.0118\\-0.0074\\-0.0399\end{bmatrix}$
Test #2	25	25.0216	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0.0357\\0.1242\\-0.0333\end{bmatrix}$

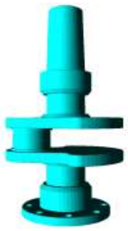
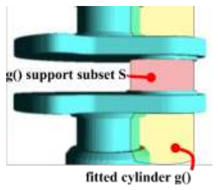
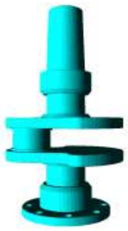
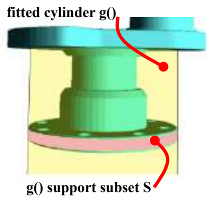
2.4.4 Comparison with Competitor Approaches

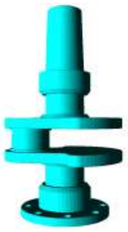
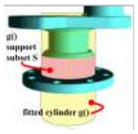
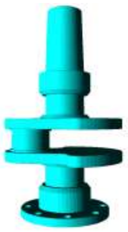
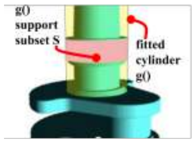
The datasets, or code, used by competitor approaches are not publicly available. In searching benchmark datasets for analytic form fitting, we found that they consist of point clouds. In contrast, the input for our algorithm should be (poor quality) triangle meshes. In consequence, we resorted to (a) executing CAD models or (b) downloading triangular meshes presenting object similarity and the same challenges as the ones in those references. The results of the comparison are presented in Table 2.9.

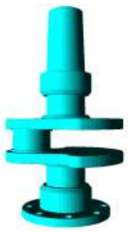
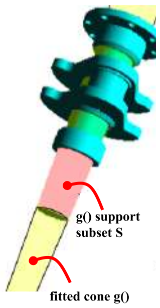
Table 2.9: Test with reference datasets of competitor approaches.

Test	**Reference + Dataset**	**Input Triangular Set** M	**Primitive + Segmentation**	**Our Result**
1	Ref. [10]. Dataset: Sphere-in-Cube Ref. [10] fails to include all triangles in S Ref. [10] does not inform execution times. Requires dense datasets. Our execution is correct, both in S and $g()$.		Sphere + Dihedral Segmentation	
2	Ref. [18]. Dataset: C^1-cylinder Ref. [18] requires dense datasets. Ref. [18] does not give execution times. Our execution is correct, both in S and $g()$.		Cylinder + Fitting Segmentation	

Test	Reference + Dataset	Input Triangular Set M	Primitive + Segmentation	Our Result
3	Ref. [17]. Dataset: Lamp Post Ref. [17] uses triangle aspect ratios ≈ 1. Ref. [17] does not inform exc. Times. Our execution is correct, both in S and $g()$.		Cone + Dihedral Segmentation	
4			Cylinder + Dihedral Segmentation	
Test	**Reference + Dataset**	**Input Triangular Set** M	**Primitive + Segmentation**	**Our Result**

5	Ref. [8]. Dataset: Crankshaft Ref. [8] uses dense, good quality triangle set ≈ 1. Ref. [8] gives global execution times. Our execution is correct, both in S and $g()$.		Cylinder + Dihedral Segmentation	
6			Cylinder + Fitting Segmentation	

7		Cylinder + Dihedral Segmentation	
8		Cylinder + Fitting Segmentation	

9	Ref. [8] does not cover cone extraction. Our execution is correct, both in S and $g()$.		Cone + Dihedral Segmentation	

Our algorithm does not fail in any one of the datasets presented by competitor approaches. In row 9, our algorithm is able to fit a correct $g()$ conical analytic form, while Ref. [8] does not attempt this fit. These competitor approaches ([8,10,17,18]) either do not publish execution times at all or publish lumped times spent for fitting the full set of primitives sought in the input triangle set M.

2.4.5 Complexity

Table 2.10 exhibits the complexities for distinguishable existing approaches in the literature. The complexity is calculated based on the standard costs of the diverse subprocesses required for each method.

Table 2.10: Complexities of existing approaches and of our algorithm (n is the number of points or triangles of the input set). * Obtaining initial guess of $g()$. ** Applying optimization to $g()$. † Segmentation and fitting are codependent.

Approach	**Preprocessing**	**B-Rep**	**Extraction of S**	**Fitting**	**Extraction + Fitting †**	**Overall**
[8] Yan et al.	$O(n^2)$	-	$O(n^2)$	$O(n^2)$	$O(n^4)$	$O(n^4)$
[7] Attene et al.	$O(n^2)$	-	-	-	$O(n^3)$	$O(n^3)$
[4] Ruiz et al.	$O(n^3)^*$	-	-	$O(n^2)^{**}$	-	$O(n^3)$
[13] Li et al.	$O(n^2)$	-	$O(n)$	$O(n^3)$	$O(n^4)$	$O(n^4)$
[5] Hulik et al.	$O(n)$	-	-	$O(n^3)$	-	$O(n^3)$
Our approach	$O(n)$	$O(n^2)$	$O(n)$	$O(n^2)$	$O(n^3)$	$O(n^3)$

Notice that the methods (rows) analyzed in Table 2.10 have diverse goals. Therefore, their overall complexity is not comparable on an equal basis. The standard cost for extracting the support set S (i.e., computation of neighboring triangles) is $O(n^2)$. However, we achieve complexity $O(n)$ when profiting off the boundary representation for input triangle set M. This B-Rep construction holds a complexity $O(n^2)$, but the hashing preprocessing reduces the search set (n) to, at most, three voxels. Section 2.4.6 presents the time reduction due to the hashing execution.

2.4.6 Reduction in Computing Time by Using Hashing

The preprocessing present in our algorithm contains the sequence: 1. spatial hashing and 2. boundary representation. The B-Rep makes the neighborhood relations (among VERTEXs, EDGESs and FACEs) in the input triangle set M explicit. At the same time, it is a diagnosis of non-manifold conditions and guarantees uniform orientation of the input triangle set M.

Although the B-Rep is the visible database for interrogation of M, it is the spatial hashing that enables the efficient construction of B-Rep and subsequent extraction of support subset S and fitting of $g()$ to S. Due to this reason, this section focuses on the acceleration of B-Rep construction due to the spatial hashing.

The spatial hashing reduces the search set space for interrogations of VERTEX, EDGE and FACE neighborhoods. Figure 2.25 presents the reduction in execution time of the B-Rep construction for different input sizes before and after applying the spatial hashing. The figure shows a reduction factor of about 1000 in the execution time. It is also relevant to mention that the B-Rep + hashing cost represents a very faithful logarithmic reduction in the bare B-Rep expenses.

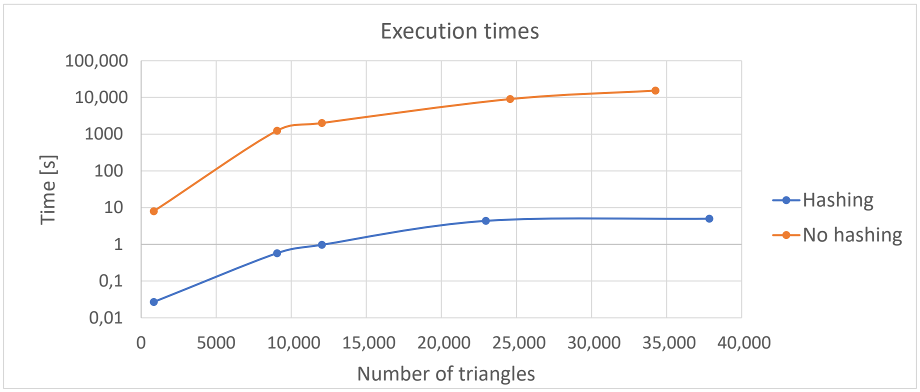


Figure 2.25: Execution times for B-Rep construction with and without previous spatial hashing.

2.5 Conclusions and Future Work

This manuscript presents the implementation of an algorithm to identify user-requested analytic forms (cone, cylinder, sphere) and their triangle support subsets S, in a connected manifold triangle set M.

The statistical quality of M is low as it is a heterogeneous and anisotropic, with a poor aspect ratio triangulation. This characteristic aligns with specific industrial applications in which low-polygon sets are purposely preferred at the expense of mesh quality.

Our algorithm favors a one-by-one progressive estimation of the parameters of the $g()$ form, based on local probing. Our approach does not favor the application of generic statistical multi-parameter fittings, which renounce the exploitation of $g()$-specific properties and require a dense, good quality input M.

Three different types of primitives (cylinder, cone, and sphere) were fitted. Two methods were implemented for the extraction of $g()$ support subset S, depending on the C-continuity level at the border ∂S with $M - S$. The results are consistent and accurate in obtaining $g()$ for statistically

poor triangulations. The precision in the parameters of each analytical form was sufficient for our industrial application, presenting (for example) a maximal error of ≈ 0.1 % in the opening angle of the cone.
As expected, fitting-based extraction presents higher execution times than a dihedral angle-based one. The iterative process of calculating the primitive parameters introduces an additional cost to the identification of the $g()$ support subset S. In contrast, real-time execution was achieved for all cases of dihedral angle-based S extraction. Both methods are accelerated by the connectivity information provided by the B-Rep.
The spatial hashing is central in accelerating the B-Rep construction, and thus the S support subset extraction and $g()$ primitive fitting. Our approach takes advantage of the B-Rep benefit of constant-time access to VERTEX, EDGE and FACE neighborhoods. Spatial hashing reduces time complexity but adds storage complexity. Therefore, it is not a theoretical advantage. In practice, however, spatial hashing dramatically speeds up B-Rep construction because it delivers bounded neighborhood search times as it (B-Rep construction) acts in a constant search space. Future work is required on the acceleration of fitting-based extraction. A natural upgrade of our method is envisioned for cases in which the primitive sought type $\mathbb{T}$ is not specified but instead is one out of a given finite set.

Abbreviations

The following abbreviations are used in this manuscript:

Term	Description
$M(\mathcal{T}, \mathbf{P})$	Triangle (2-manifold) set with geometry $\mathbf{P}$ and topology $\mathcal{T}$.
$\mathcal{T} = \{t_1, t_2, ...\}$	Set of triangles of the input mesh M.
$\mathbf{P} = \{t_1, t_2, ...\}$	Set of vertices of the input mesh M.
t_s	Seed triangle.
$\mathbb{T}$	User-specified type of the analytic form {Cylinder, Sphere, Cone}.
$g()$	Analytic form denoting one of the cone, cylinder or spheric primitives. $g()$ equally expresses parametric or implicit forms, as well as the set of parameters determining a primitive.
E_t	User-specified method for the analytic form fitting.
S	Support set of triangles that fit a particular analytic form.
$C(t, k)$	Topological circle on a triangular mesh M, with center triangle t and distance k.
p_0	Distinguished point of cylinder, sphere or cone definition.
R	Cylinder or sphere radius.
$\hat{v}$	Axis vector for cylinder or cone.
γ	Opening angle for cone.
$\delta_x, \delta_y, \delta_z$	Voxel side length in the x, y, z directions, respectively.
Ω	Rectangular prism orthogonally oriented w.r.t. the coordinate axes, containing a $N_v \times N_v \times N_v$ grid of rectangular voxels v_{ijk}.
N_v	Number of voxels per side of Ω

h	Hashing function: $h : P \rightarrow N_v \times N_v \times N_v$, which maps each vertex of the input triangle set M into the (i,j,k) indices of the voxel $v_{ijk} \subset \Omega$, which contains such a vertex.
H	$\Omega \rightarrow \mathcal{T}$, which maps each voxel v_ijk of Omega into the set of triangles of M which contain a vertex inside voxel v_ijk.
C_p	Set of auxiliary points to calculate the axis of the analytic forms (center for sphere case).
$\hat{n_i}, n$	Vector normal to triangle i.
b_i, b	Baricenter of triangle i.
K_{min}, K_{max}	Maximal and minimal surface curvatures.
u_{min}, u_{max}	Directions of maximum and minimal curvatures.
T	Homogeneous rigid transformation mapping the point b of mesh M onto the $[0, 0, 0]'$ of $\mathbb{R}^3$, and the plane tangent to M at p onto the plane XY in $\mathbb{R}^3$.
$L_n = \{\ell_1, \ell_2, ...\}$	Set of lines generated by the baricenter b_i and normal $\hat{n}_i$.
N_S	Number of triangles in triangle subset S.
N_T	Threshold in number of triangles needed to obtain a stable approximation of an analytic form.
$w(\|S\|)$	Maturity measure of the set S. $1 \rightarrow 1$ if $\|S\|$ is small (S immature) $w \rightarrow 0$ if $\|S\|$ is large (S mature).
u	Relation between the number of triangles in subset N_S and the threshold N_T.

3
Singularity Correction in Boundary Element Method for Steady Incompressible Viscous Flow

Samuel Velez-Sanin[1], Cristian Rendon-Cardona[1], Juan Diego-Jaramillo[2], Nicolas Guarín-Zapata[2] and Oscar Ruiz-Salguero[1]

[1] CAD CAM CAE Laboratory - EAFIT University, Colombia
[2] Applied Mechanics Laboratory - EAFIT University, Colombia

Context

Samuel Velez-Sanin, Cristian Rendon-Cardona, Juan Diego-Jaramillo, Nicolás Guarín Zapata, Oscar Ruiz Salguero. Computation, ISSN: 2079-3197, Journal webpage: https://www.mdpi.com/journal/computation/about. Sumbitted to Journal

Abstract

In the domain of Boundary Element Methods, the problem of Boundary and Surface Integral singularities implied in computing the effects of a loaded node upon itself, is relevant because it

introduces discontinuities and estimation errors in the solution. Existing solutions for this problem are: (a) sub-segmentations of the integration domain, (b) integration over a distorted but non-singular domain (element) and correction. Limitations of the mentioned solutions are: high computing expenses (strategies (a)) and increased complexity of the integration domain (strategies (b)). To partially overcome these limitations, this manuscript implements a method to compute the boundary and surface integrals for the singular Green functions when the positions of the source and field elements coincide. The implemented method evaluates the analytic boundary integral displacing the source node of the singular element. In addition, we compute the analytic surface integral for the singular cell. Both of this methods are implemented in the field of fluid dynamics. Our results (in the area of fluid dynamics) correctly predict laminar flow in immersed profiles which present low deflections in the flow direction. Our implementation presents poor performance when the boundary normal vectors significantly differ within a boundary neighborhood. Future work is required in improving the performance when boundary normal vectors significantly differ within a boundary neighborhood, increasing the order of the elements, and the extension to other scientific fields.

Glossary

Ω	Boundary Element Problem domain. Closed set. May be semi-infinite or finite.
B	Boundary of Ω. This boundary presents singularities in the evaluation of the boundary integrals. Thus, it is the focus of our contribution. $B \subset \Omega$.
Γ_0	External LOOP of B. $\Gamma_0 \subset B$.
Γ_i	Internal LOOP of B. $\Gamma_i \subset B$, $i = 1, 2, ..., n$.
R	Interior of Ω. Flow region for which the velocity vector field is calculated by the BEM integral equation. $R = int(\Omega)$ with $int(\Omega) = \Omega - B$.
S	$S \subset R$. Region in which the non-linear convective acceleration effects are significant. Singularities occur in the integration of $\frac{\partial G_{ij}}{\partial x_k}$ for this region. Scenario of our contribution.
x_i	[m] Spatial Cartesian coordinate $\in \Omega$. $i = 0, 1$.
u_i	[m] Absolute velocity vector in the i direction. $i = 0, 1$.
v_i	[m] Velocity perturbation vector in the i direction. $i = 0, 1$.
V_i	[m] Free flow velocity vector in the i direction. $i = 0, 1$.
f_i	$[\frac{N}{m^2}]$ Traction vector in the i direction. $i = 0, 1$.
$\vec{f_i}$	[N] Body force vector in the i direction. $i = 0, 1$.
ρ	[kg/m^3] Density.
μ	[Ns/m^2] Dynamic viscosity.
X	[m] Field element position vector.

ξ	[m] Source element position vector.
I_aG	Analytic integration coefficient of the Green function in the canonical coordinate system.
${}^oI_aG$	Analytic integration coefficient of the Green function in the $x\,y$ cartesian coordinate system.
I_nG	Numerical integration coefficient of the Green function by Gaussian quadrature.
M	Total number of boundary elements.
L	Total number of surface elements.
m	Boundary element identifier.
l	Surface element identifier.
$\hat{n}$	Normal vector of a boundary element.
r	Position vector from source element (ξ) to field element (X).
δ	Kronecker's delta.
$\mathbf{t}$	Nodal traction vector.
$\mathbf{v}$	Nodal velocity perturbation vector.
$\mathbf{t^o}$	Nodal convective traction vector.
$\sigma^{\mathbf{o}}$	Convective stress nodal tensor.
$\mathbf{G}$	Matrix with coefficients from the G_{ij} boundary integrals.
$\mathbf{F}$	Matrix with coefficients from the F_{ij} boundary integrals and the c_{ij} values.
$\mathbf{D}$	Matrix with coefficients from the $\frac{\partial G_{ij}}{\partial x_k}$ surface integrals.
P	Total number of functional nodes.
Q	$\sum_{m=1}^{M} A_m$.
A_m	Number of functional nodes in element m.
Convective acceleration	Change of speed produced by changes in spatial position.
$\mathbb{I}$	Identity matrix.
Ω_r	Mesh triangle element.
Ω_c	Canonical triangle element.

3.1 Introduction

3.1.1 Problem Specification

The field of fluid dynamics have always been of high interest in engineering. The complex phenomena of flow and its behaviour introduces the need of different manners to predict it. The Boundary Element Method (BEM) is one of many of these solutions to the problem. However, the method presents singularities in its integrals that must be sorted in order to implement it. This manuscript proposes the implementation of the displacement of the source node method [22, 23] as a solution to this problem for a specific case of flow. This is the 2-dimensional Steady Incompressible Viscous flow.

Input

1 BEP: Boundary Element Problem for the 2-dimensional Steady Incompressible Viscous flow. Contains boundary and surface singularities.

Output

1 $\widetilde{BEP}$: Boundary Element Problem approximation with boundary and surface integral singularities solved.

2 A: An algorithm which approximates the solution of the BEP by solving the $\widetilde{BEP}$.

Observations on Problem Specification
The next observations must be taken into account. (1) We are not presenting a software that transforms a problem into an algorithm. (2) The goal is to exhibit the procedure of the problem specification. (3) We are not mathematically proving the transformation of problems. (4) The algorithm does not solve the BEP under a distance norm between the problems. (5) We do not provide a proof of $A: BEP \to \widetilde{BEP}$ with a defined error. (6) The proposed solution is an intuitive approach.

This manuscript is organized as follows: Section 3.2 discusses the relevant literature. Section 3.3 presents the proposed method to sort the singularities in the Boundary Integrals. Section 3.4 discusses the results of various simulations. finally, section 3.5 discuses the conclusions and introduces to future work.

3.2 Literature Review

The current literature for sorting the singularity presented in the boundary element method is divided into two main categories. (1) Sub-segmentations of the integration domain, (2) distortion of the singular boundary/surface element.

Element sub-segmentation:
The integration domain (singular element) is subdivided into smaller domains around the singular position. A numerical integration (quadrature scheme) is computed in the resulting non-singular domains. Depending on the type of singularity, special quadrature schemes may be utilized for the remaining singular part. In some cases, the singular part is computed in the Cauchy Principle Value sense.

Singular Element Distortion:
The singular element is distorted (lumped) in order to separate the source node from the the boundary. In consequence, the previous singular kernels can be integrated analytically and then the limit is computed for shrinking the boundary to its original state.

[24] presents a machine learning-based framework, constructing a prediction model by supervised machine learning algorithms. The authors intend to construct the mapping relationship between integral elements and singular integral results. This approach is not of the concern of the authors of the current article because of the statistical nature of machine learning and the approach presented by [24].

3.2.1 Conclusions of Literature Review

Table 3.2: Different approaches and our contribution

Approach	**Refs.**	**Advantages**	**Disadvantages**
Sub-segmentation of the singular boundary/surface element (numerical evaluation)	[25–27]	(1) Versatility for the type of element used.	(1) Computational cost for the evaluation of each segment
Distortion of the singular boundary/surface element	[28–33]	(1) *Reduced computational effort because of the evaluation of a single equation.	(1) Increased complexity of the integration domain.
Displacement of the source node in the singular canonical boundary element (our approach)	[22, 23]	(1) Simple computation of integrals with pre-calculated formulae. (2) No modification of the integration domain, resulting in less complex integrals.	(1) Complex analytic integration.

3.3 Methodology

The Boundary Element Method formulation presented in this section is strongly based on the formulation made by [34] for Steady Incompressible Thermoviscous flow. It is our simplification for the flow characteristics specified in section 3.1. The simplifications are (1) The flow is adiabatic and isotherm leading to a simplified system of equations, (2) low order elements are used for the discretization of the boundary and surface (S), (3) the singular integrals (boundary and surface) are approached in an analytic way (scenario of our contribution).

3.3.1 Domain Layout

The general domain layout for this approach is presented on Fig. 3.1. Complete domain $\Omega = R \cup B$. With B being the boundary and R the interior region. The boundary B is conformed by $\Gamma_0 \cup \Gamma_1$, the exterior and interior boundaries, respectively.

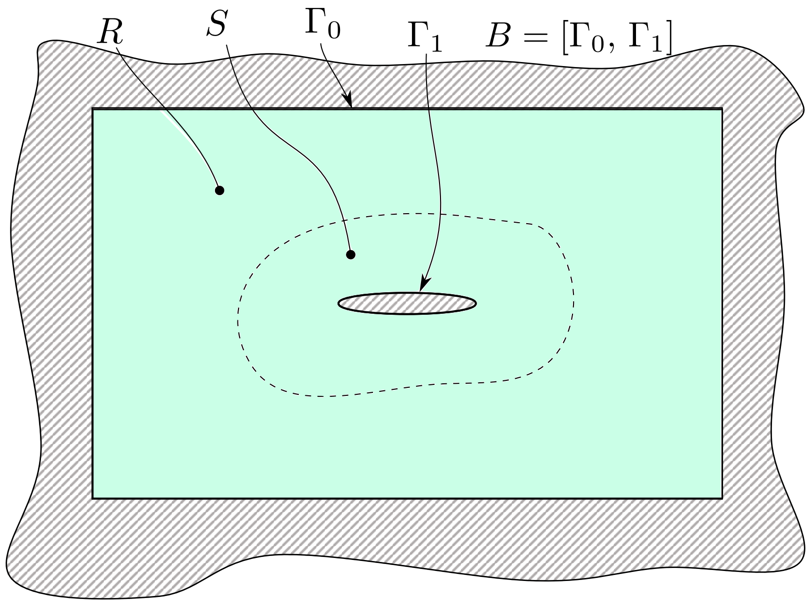


Figure 3.1: BEP domain layout. $\Omega = R \cup B$.

3.3.2 Governing Equations

We use the governing equations for Steady Incompressible Thermoviscous Flow presented in [34] for the implemantation of the BEM in fluid dynamics.

$$\text{Mass conservation} \quad \frac{\partial u_j}{\partial x_j} = 0, \tag{3.1}$$

$$\text{Momentum} \quad \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j} - \frac{\partial p}{\partial x_i} - \rho u_j \frac{\partial u_i}{\partial x_j} + \vec{f}_i = 0, \tag{3.2}$$

$$\text{Energy} \quad k \frac{\partial^2 \theta}{\partial x_j \partial x_j} - \rho c_\varepsilon u_j \frac{\partial \theta}{\partial x_j} + \Upsilon + \Psi = 0. \tag{3.3}$$

For purposes of this work, some assumptions are introduced to the Eqs. (3.1), (3.2) and (3.3). These assumptions are described below:

1 Constant temperature.

$$\frac{\partial \theta}{\partial x_i} = 0$$

2 There aren't any heat sources nor viscous dissipation is considered.

$$\Upsilon = 0 \quad \Psi = 0$$

Because of statements 1 and 2, Eq. (3.3) is not considered.

3 There aren't any body forces.

$$\vec{f}_i = 0$$

After applying the assumptions, the resulting governing equations are:

$$\text{Mass conservation} \quad \frac{\partial u_j}{\partial x_j} = 0 \tag{3.4}$$

$$\text{Momentum conservation} \quad \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j} - \frac{\partial p}{\partial x_i} + f_i = 0 \tag{3.5}$$

Where

$$f_i = -\rho u_j \frac{\partial u_i}{\partial x_j} + \vec{f}_i$$

Not taking into account body forces, $\vec{f}_i = 0$.

3.3.3 Continuous Formulation of the BEM Integral Equation

With all the assumptions presented in section 3.3.2, the Eq. (16a) from [34] is rewritten in Eq. (3.6) for the continuous integral formulation of the Boundary Element Method to be solved.

$$\begin{aligned} c_{ij}(\xi)v_i(\xi) = &\int_B [G_{ij}(X-\xi)t_i(X) - F_{ij}(X-\xi)v_i(X) \\ &- G_{ij}(X-\xi)\rho(U_k(X)+v_k(X))n_k(X)v_i(X)]dB(X) \\ &- \int_S \left[\frac{\partial G_{ij}(X-\xi)}{\partial x_k}\rho(U_k(X)+v_k(X))v_i(X)\right] dS(X). \end{aligned} \tag{3.6}$$

Green Functions

Eqs. (3.7 - 3.9) present the Green Functions utilized in Eq. (3.6). Singularities occur when $X_i = \xi_i$, making $r^2 = 0$.

$$G_{ij} = \frac{1}{4\pi\mu}\left(\frac{y_i y_j}{r^2} - \delta_{ij}\ln r\right). \tag{3.7}$$

$$F_{ij} = -\frac{1}{2\pi r}\left(\frac{2y_i y_j y_k n_k}{r^3}\right). \tag{3.8}$$

$$\frac{\partial G_{ij}}{\partial x_k} = \frac{1}{4\pi\mu r}\left(\frac{\delta_{jk}y_i}{r} + \frac{\delta_{ik}y_j}{r} - \frac{\delta_{ij}y_k}{r} - \frac{2y_i y_j y_k}{r^3}\right) \tag{3.9}$$

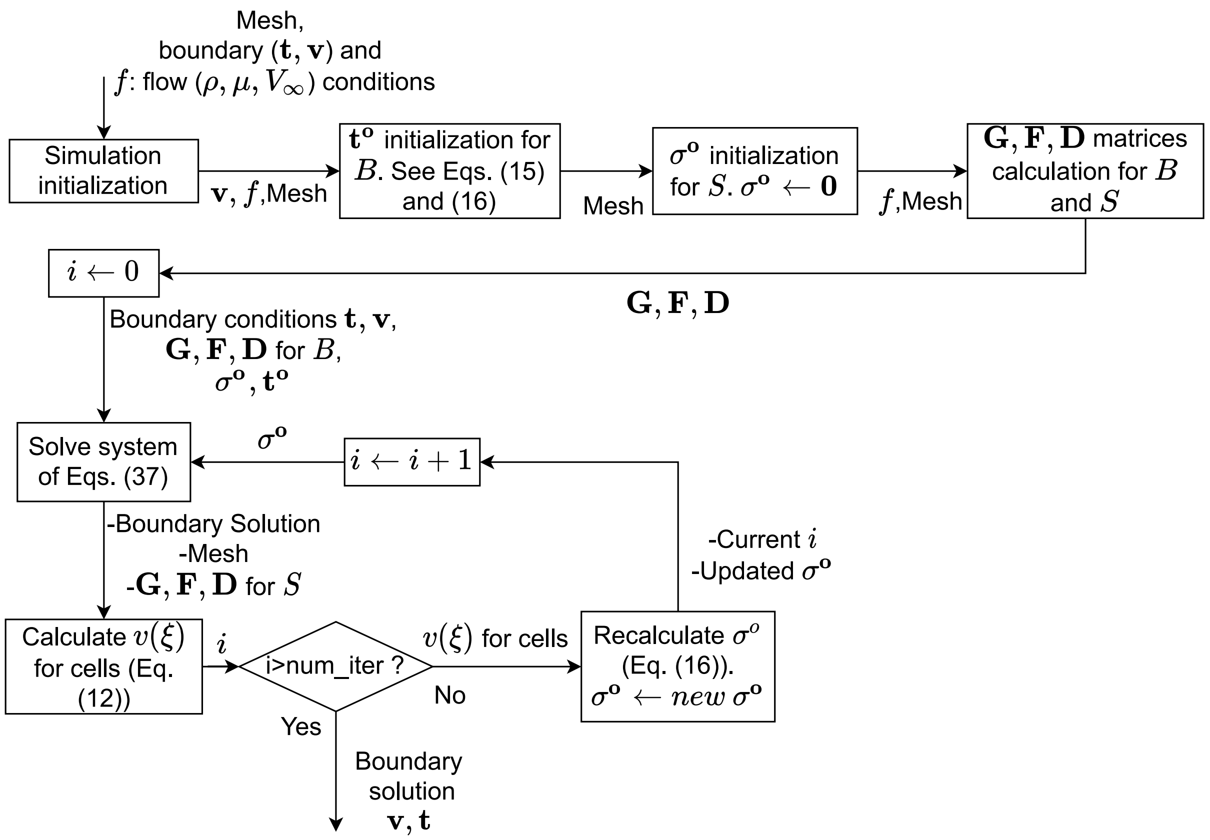
$$y_i = X_i - \xi_i. \tag{3.10}$$

$$r^2 = y_i y_i. \tag{3.11}$$

3.3.4 Numerical Implementation

Since no analytic solution can be found for the integral BEM equation (Eq. 3.6), a numerical approximation is sufficient. The numerical solution scheme and the discretization for the BEP is discussed in this section.

The scheme is represented in the following block diagram. It is the process by which an approximated numerical solution to $\widetilde{BEP}$ is found.



Mesh, boundary ($\mathbf{t}, \mathbf{v}$) and f: flow (ρ, μ, V_∞) conditions
Simulation initialization
$\mathbf{v}, f$,Mesh
$\mathbf{t^o}$ initialization for B. See Eqs. (15) and (16)
Mesh
$\sigma^\mathbf{o}$ initialization for S. $\sigma^\mathbf{o} \leftarrow \mathbf{0}$
f,Mesh
$\mathbf{G}, \mathbf{F}, \mathbf{D}$ matrices calculation for B and S
$\mathbf{G}, \mathbf{F}, \mathbf{D}$
$i \leftarrow 0$
Boundary conditions $\mathbf{t}, \mathbf{v}$, $\mathbf{G}, \mathbf{F}, \mathbf{D}$ for B, $\sigma^\mathbf{o}, \mathbf{t^o}$
Solve system of Eqs. (37)
$\sigma^\mathbf{o}$
$i \leftarrow i + 1$
-Current i
-Updated $\sigma^\mathbf{o}$
-Boundary Solution
-Mesh
-$\mathbf{G}, \mathbf{F}, \mathbf{D}$ for S
Calculate $v(\xi)$ for cells (Eq. (12))
i
i>num_iter ?
$v(\xi)$ for cells
No
Yes
Recalculate σ^o (Eq. (16)). $\sigma^\mathbf{o} \leftarrow new\, \sigma^\mathbf{o}$
Boundary solution $\mathbf{v}, \mathbf{t}$

Figure 3.2: Block diagram for the general process to achieve a numerical solution for $\widetilde{BEP}$. f: Flow conditions variables.

Spatial Discretization

The domain Ω for the BEP has to be discretized since the analytic continuous equation that represents it cannot be determined. For this, the boundary B and surface S elements are discretized as follows:

- Boundary Elements: 1-dimensional low order constant elements (see Fig. (3.3)). Each element is defined by two geometrical nodes and one functional node.

- Surface Cells: 2-dimensional low order constant triangular cell elements (see Fig. (3.3)). Each cell is defined by 3 geometrical nodes and one functional node.

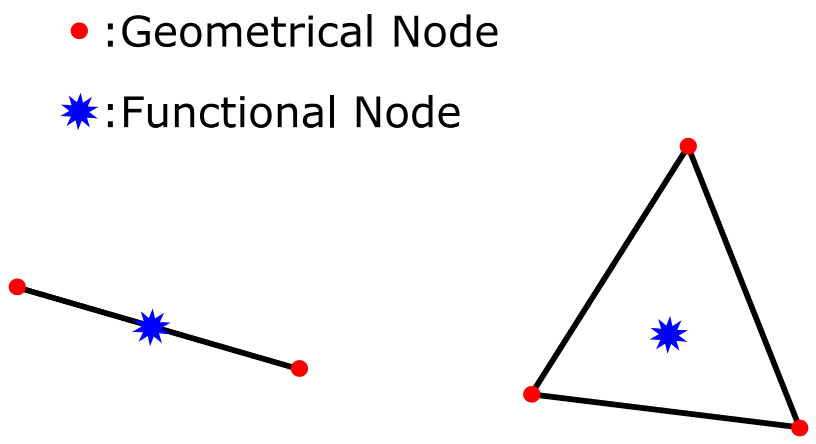


Figure 3.3: Boundary (left) and surface (right) elements.

Discretized BEM Equation

Taking into account the spatial discretization defined in section 3.3.4, the Eq. (3.6) is reformulated in Eq. (3.12).

$$
\begin{aligned}
c_{ij} v_i(\xi) = \sum_{m=1}^{M} \left(t_i(Xm) \int_{B_m} G_{ij}(r)\mathrm{d}B - v_i(Xm) \int_{B_m} F_{ij}(r)\mathrm{d}B - t_i^o(Xm) \int_{B_m} G_{ij}(r)\mathrm{d}B \right) \\
+ \sum_{l=1}^{L} \sigma_{ki}^o(Xl) \int_{S_l} \frac{\partial G_{ij}(r)}{\partial x_k} \mathrm{d}S
\end{aligned}
\tag{3.12}
$$

, with M being the total number of boundary elements and L the total number of surface S cells. Descriptions for some terms of equation 3.12 can be seen in table 3.3.

Table 3.3: Terms description for the Discretized BEM equation. T: Time units, L: Distance units, M: Mass units.

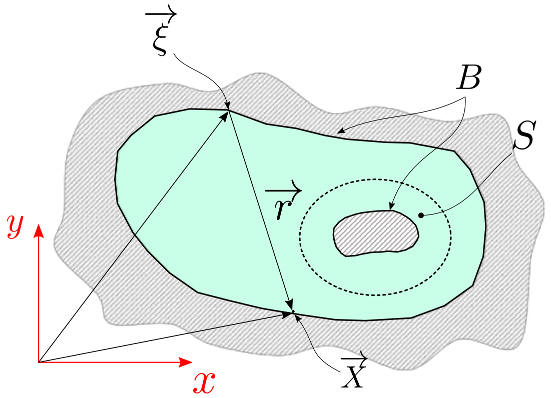
Term	**Description**	**Dims.**
$B = \sum_{m=1}^{M} B_m$	Boundary discrete composition	-
$S = \sum_{l=1}^{L} S_l$	Non-linear convective region discrete composition	-
$t_i^o = \sigma_{ki}^o n_k$	Non-linear convective traction vector	$\frac{M}{LT^2}$
$\sigma_{ki}^o = \rho(U_k + v_k)v_i$	Non-linear convective traction tensor	$\frac{M}{LT^2}$
$v_i = u_i - V_i$	Velocity perturbation vector	$\frac{L}{T}$
t_i	Traction vector	$\frac{M}{LT^2}$
$c = 0.5\mathbb{I}$	Diagonal matrix of values 0.5	-

3.3.5 Integral Approaches

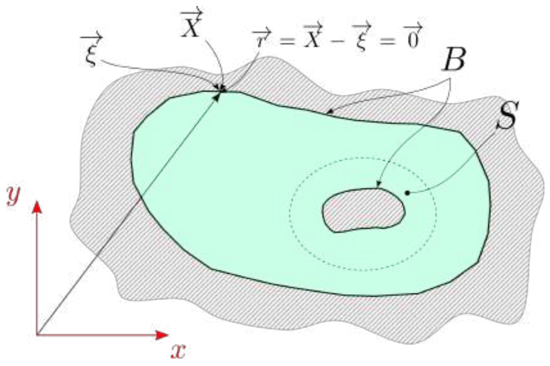
The integrals contained in Eq. 3.12 are evaluated to obtain a linear system of equations in terms of unknown boundary conditions. They are evaluated for the Green functions over each boundary and surface elements respectively. For this, the current section presents an implementation of the method proposed by [22, 23] for the evaluation of the singular boundary integrals. In addition, a direct analytic evaluation is also discussed for the singular surface integrals as part of our contribution in the BEM solution for Non-linear Steady Incompressible Viscous 2-dimensional flow.

Boundary and Surface Elements Non-singular and Singular Situations

The boundary and surface integrals are solved for all combinations of $\vec{\xi}$ and $\vec{X}$, both positioned in boundary and surface elements. The positioning of $\vec{\xi}$ in the boundary elements (Fig. 3.4) corresponds to the integration process needed to compute the missing boundary conditions. In addition, positioning $\vec{\xi}$ in the surface cells (Fig. 3.5) is performed for integrals needed for recalculation of σ^o in each iteration.

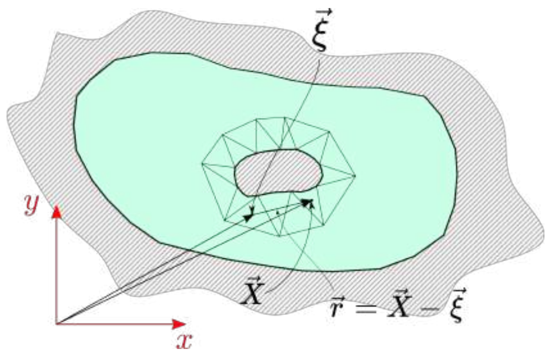


(a) Non singular situation. ξ and X are not coincident. $\vec{r} = X - \xi \neq \mathbf{0}$. Numerical integration is applied for the mesh elements.

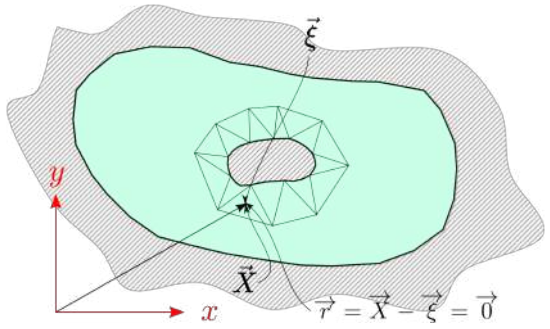


(b) Singular situation. ξ and X are coincident. $\vec{r} = X - \xi = \mathbf{0}$. Situation for which our contribution is made. Analytic integration scheme is implement over a canonical element.

Figure 3.4: Boundary (S) ξ and X elements positions for boundary integrals. Surface (S) is the relevant non-linear convection zone.



(a) Non singular situation. ξ and X are not coincident. $\vec{r} = X - \xi \neq \mathbf{0}$. Numerical integration is applied for the mesh element.



(b) Singular situation. ξ and X are coincident. $\vec{r} = X - \xi = \mathbf{0}$. Situation for which our contribution is made. Direct analytic integration is performed over a canonical element.

Figure 3.5: Surface (S) ξ and X elements positions for surface integrals. Integral evaluation cases for the calculation of σ^o between iterations.

Non-singular Boundary and Surface Integrals

For the non-singular cases $X \neq \xi$ (see Figs. (3.4. a) and (3.5. a)), the integration over the boundary and surface is computed numerically. For the boundary integrals the evaluation is performed using a 3-point Gaussian Quadrature. For surface integrals the evaluation is performed using a 1-point Gaussian Quadrature. These numerical integrals are not treated or explained in detail since they are not the focus of this article.

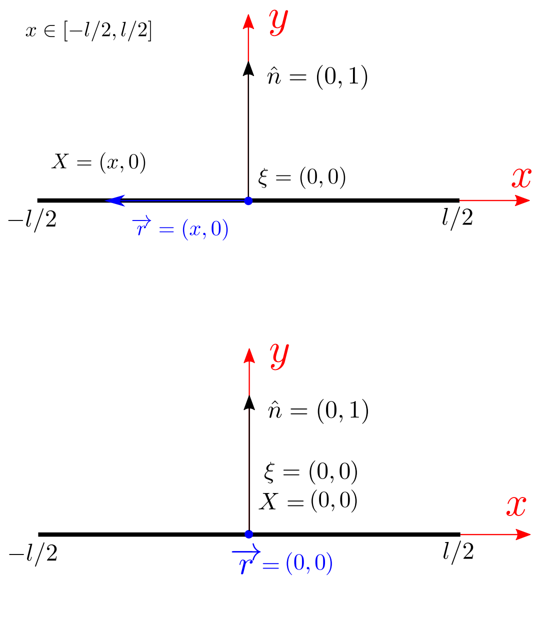
Singular Boundary Integrals

For the singular boundary integrals ($\vec{\xi} = \vec{X}$), the computation is performed with a modified Green function as the integrand over a canonical element. The method consists in displacing the source node ξ a distance D (Fig. 3.6). After displacement, the singular boundary integrals can be evaluated analytically and the limit taken when $D \to 0^+$ (since D is a distance) as per Eqs. (3.13) and (3.14).

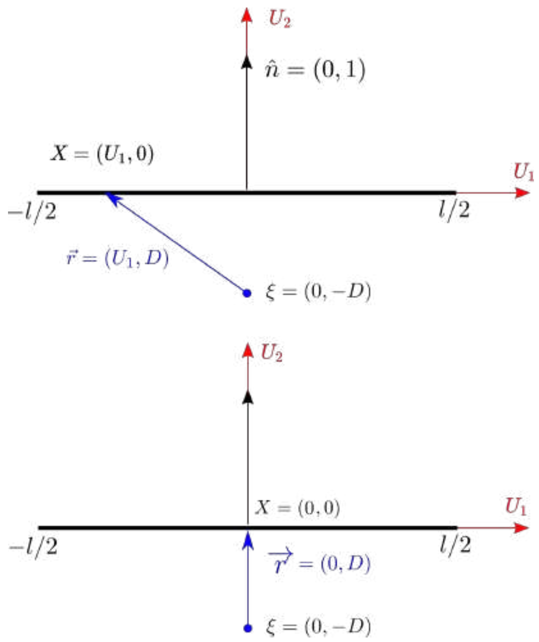
$$I_a G_{ij}(X_m - \xi_m) = \lim_{D \to 0^+} \int_{B_m} G_{ij}(X_m - (\xi_m - D))dB \tag{3.13}$$

$$I_a F_{ij}(X_m - \xi_m) = \lim_{D \to 0^+} \int_{B_m} F_{ij}(X_m - (\xi_m - D))dB \tag{3.14}$$

This method of displacement of the source node is presented by [22, 23] for the elasticity and fracture fields respectively. Our contribution consists in its implementation applied in the field of fluid dynamics described in section 1. In addition, a contribution is proposed as the direct analytic solution of the singular surface integrals (see section 3.3.5).



(a) Singular case occurs when $x = 0$. X and ξ coincide (bottom).



(b) Displacement of source node (ξ) for the correction of the singular case. $\forall x, \vec{r} \neq \mathbf{0}$.

Figure 3.6: Analytic integration over the singular element m (see Fig. 3.4 (b)) $\xi = X_m$. Canonical coordinate system. Integrating for $U_1 \in [-l/2 \rightarrow l/2]$ in local canonical coordinates. Scenario (subFig. (a)) of our contribution.

The mathematical procedure for avoiding the singularity (Fig. 3.6 (a)) and the analytic integral evaluation is now discussed.

The modified variables for the Green functions (Eqs.3.10 and 3.11) are presented in Eqs. (3.15-3.17). These are then replaced in each Green functions G_{ij} and F_{ij} as shown in Eq. (3.18) for G_{11}.

$$y_1 = X_1 - \xi_1 = U_1 - 0 = U_1 \tag{3.15}$$

$$y_2 = X_2 - \xi_2 = 0 - (-D) = D \tag{3.16}$$

$$r = \sqrt{U_1^2 + D^2} \tag{3.17}$$

$$G_{11}(X - \xi, D) = \frac{1}{4\pi\mu}\left(\frac{U_1^2}{U_1^2 + D^2} - \ln\left(\sqrt{U_1^2 + D^2}\right)\right) \tag{3.18}$$

The result of integrating this modified function is shown in Eq. (3.19). In addition, following the same procedure, the results for the integrands G_{12}, G_{21} and $G22$ are as shown in Eqs. (3.20-3.22)

$$\lim_{D \to 0^+} \frac{1}{4\pi\mu} \int_{-l/2}^{l/2} \frac{U_1^2}{U_1^2 + D^2} - \ln\left(\sqrt{U_1^2 + D^2}\right) dU_1 = \frac{l\left(-\log\left(l^2\right) + \log(4) + 4\right)}{8\mu\pi} \tag{3.19}$$

$$\lim_{D\to 0^+}\int_{-l/2}^{l/2} G_{12}(X-\xi,D)dU_1 = 0 \tag{3.20}$$

$$\lim_{D\to 0^+}\int_{-l/2}^{l/2} G_{21}(X-\xi,D)dU_1 = 0 \tag{3.21}$$

$$\lim_{D\to 0^+}\int_{-l/2}^{l/2} G_{22}(X-\xi,D)dU_1 = \frac{l\left(-\log\left(l^2\right)+\log(4)+2\right)}{8\mu\pi} \tag{3.22}$$

Furthermore, to solve $I_aF_{ij}(X_m-\xi_m)$, one has that the normal vector $\hat{n}$ of the canonical element is,

$$n_1 = 0 \quad n_2 = 1 \tag{3.23}$$

Now, taking into account the normal vector, the variables in Eqs. (3.15-3.17) are replaced for the integrands F_{ij} as seen in Eq. (3.8) for F_{11}. The integration is then performed as expressed in Eq. (3.14). This is shown in Eqs. (3.25-3.28)

$$F_{11}(X-\xi,D,\hat{n}) = \frac{1}{2\pi}\left(\frac{2U_1^2D}{(U_1^2+D^2)^2}\right) \tag{3.24}$$

$$\lim_{D\to 0^+}\frac{2}{2\pi}\int_0^{l/2}\frac{2U_1^2D}{(U_1^2+D^2)^2}dU_1 = \frac{1}{2} \tag{3.25}$$

$$\lim_{D\to 0^+} 2\int_0^{l/2} F_{22}(X-\xi,D,\hat{n})dU_1 = \frac{1}{2} \tag{3.26}$$

$$\lim_{D\to 0^+}\int_{-l/2}^{l/2} F_{12}(X-\xi,D,\hat{n})dU_1 = 0 \tag{3.27}$$

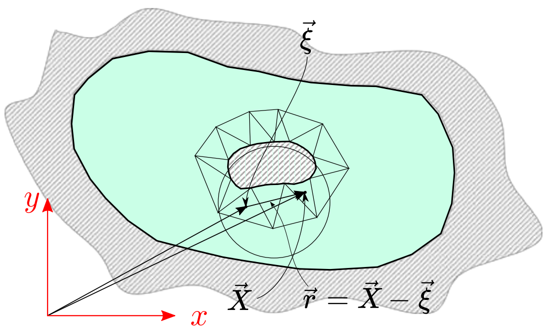
$$\lim_{D\to 0^+}\int_{-l/2}^{l/2} F_{21}(X-\xi,D,\hat{n})dU_1 = 0 \tag{3.28}$$

For the integrals of F_{12} and F_{21}, the result is nule since this functions are odd with respect to 0.

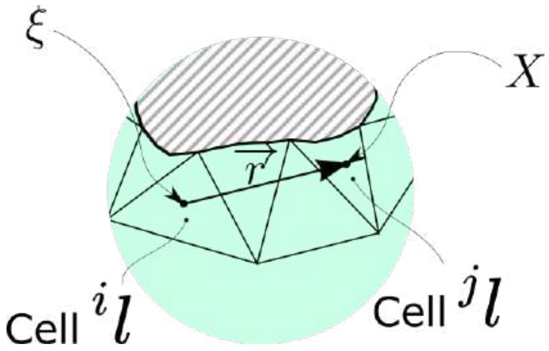
The results obtained in Eqs. (3.19-3.22) and (3.25-3.28) are transformed (see section 3.3.5) and stored in matrices I_aG^m and I_aF^m. These matrices compose the diagonal band of the influence matrices **G** and **F** (see section 3.3.5).

Singular Surface Integrals

As expressed in Fig. 3.5, the surface integral evaluation presents a singularity when $\xi = X$. This occurs only for the integration need to recalculate σ^o in each new iteration (see subsection 3.3.7). Even though, the singularity is present in the Green function $\frac{\partial G_{ij}}{\partial x_k}$, the integral exists and is finite. Consequently, a direct analytic integration is performed over a canonical element (Fig. 3.8) and then transformed for the real element.



(a) General surface integration case. Selected cells ${}^j l$ (X) and ${}^i l$ (ξ).



(b) Selected region in subFig. (a).

Figure 3.7: General integration case for region S. Singularity occurs when $i = j$ (triangles ${}^i l$ and ${}^j l$ coincide) and points ξ and X are coincident ($\vec{r} = \mathbf{0}$).

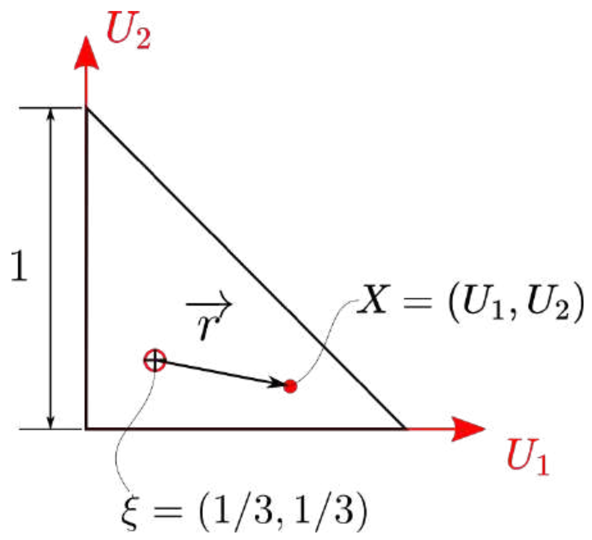


Figure 3.8: Canonical element (axis U_1, U_2). Domain for the analytic integration of $\frac{\partial G_{ij}(X-\xi)}{\partial U_k}$. Scenario of our contribution.

Since the function $\frac{\partial G_{ij}(X-\xi)}{\partial U_k}$ has a reflection with respect to the singularity point $\xi = X$ in the canonical element, the integral of such function exists and has a finite value. The integration is performed in SymPy [1] with the integrate() method. The results of such integration are given in the table 3.4. Since $\int\limits_{S_m} \frac{\partial G_{12}}{\partial x_k} dS_m = \int\limits_{S_m} \frac{\partial G_{21}}{\partial x_k} dS_m$, the result for the integrands $\frac{\partial G_{12}}{\partial x_k}$ are the only ones shown.

Table 3.4: Results for singular surface integrals of $\frac{\partial G_{ij}}{\partial U_k}$. Results for canonical element (see Fig. 3.8). SymPy [1] used for integral evaluation.

Analytic Integral Equation	**Result**	**Observations**
$I_a \frac{\partial G_{11}}{\partial U_1} = \int_0^1 \int_0^{1-U_1} \frac{\partial G_{11}(X-\xi)}{\partial U_1} dU_2 dU_1$	$-\frac{0.017265928194611}{\mu\pi}$	Results are transformed (see Eq. 3.33) and assembled in ${}^oI_a\partial G$ (see Fig. 3.12).
$I_a \frac{\partial G_{12}}{\partial U_1} = \int_0^1 \int_0^{1-U_1} \frac{\partial G_{12}(X-\xi)}{\partial U_1} dU_2 dU_1$	$\frac{0.017265928194611}{\mu\pi}$	
$I_a \frac{\partial G_{22}}{\partial U_1} = \int_0^1 \int_0^{1-U_1} \frac{\partial G_{22}(X-\xi)}{\partial U_1} dU_2 dU_1$	$\frac{0.0481585520039788}{\mu\pi}$	
$I_a \frac{\partial G_{11}}{\partial U_2} = \int_0^1 \int_0^{1-U_1} \frac{\partial G_{11}(X-\xi)}{\partial U_2} dU_2 dU_1$	$\frac{0.0481585520039788}{\mu\pi}$	
$I_a \frac{\partial G_{12}}{\partial U_2} = \int_0^1 \int_0^{1-U_1} \frac{\partial G_{12}(X-\xi)}{\partial U_2} dU_2 dU_1$	$\frac{0.017265928194611}{\mu\pi}$	
$I_a \frac{\partial G_{22}}{\partial U_2} = \int_0^1 \int_0^{1-U_1} \frac{\partial G_{22}(X-\xi)}{\partial U_2} dU_2 dU_1$	$-\frac{0.017265928194611}{\mu\pi}$	

Transformations to the Real Elements for Boundary and Surface Analytic Integrals

The results of the singular integrals, both for boundary and surface elements, are obtained for canonical elements. Therefore, these results must be mapped from a canonical domain to the real domain of the boundary element problem. The transformations that perform such mapping are discussed next for each element type (boundary-surface).

Boundary Elements Transformation

For the boundary analytic integrals, a tensor transformation for the results of Eqs. (3.13) and (3.14) is performed. This transformation is computed with a transformation matrix N, which is calculated for each element with its normal vector $\hat{n}$ (Eq. 3.29.

$$N = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} = \begin{bmatrix} n_2 & -n_1 \\ n_1 & n_2 \end{bmatrix} \tag{3.29}$$

The transformation in Eq. (3.31) represents the tensor I_aG_{ij} in the $x\,y$ coordinate system and is calculated as follows.

$${}^oI_aG_{ij} = N^T I_a G_{ij} N \tag{3.30}$$

$${}^oI_aF_{ij} = N^T I_a F_{ij} N \tag{3.31}$$

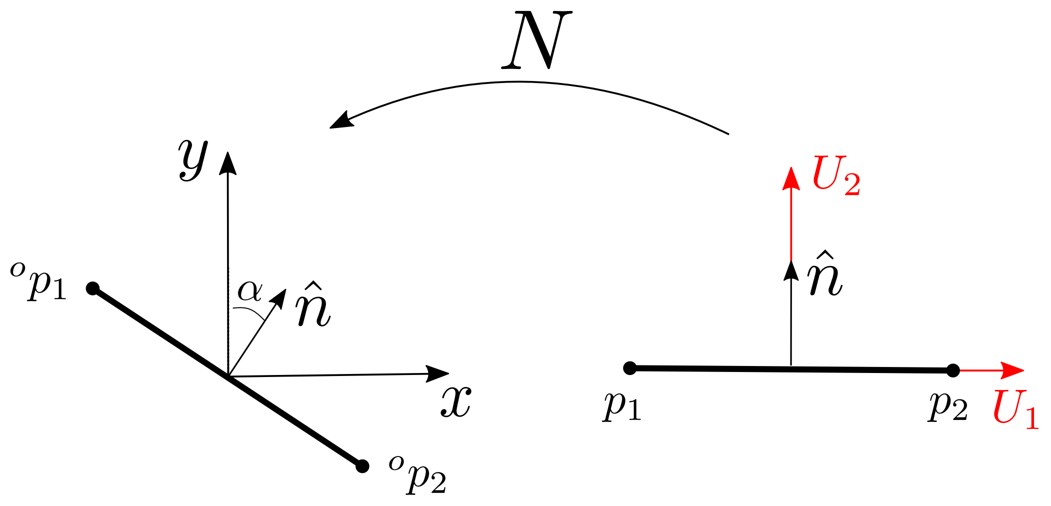


Figure 3.9: Boundary element in canonical and real orientation. Canonical ($U_1\, U_2$) and real ($x\, y$) coordinate systems.

Surface Elements Transformation

The analytic integration is executed over a canonical triangular element, as shown in Fig 3.8. The value of the integral for the surface element is transformed from the canonical to the real domain. This transforamation $T : \Omega_c \to \Omega_r$ is affine. Therefor, the Jacobian is constant for each element.

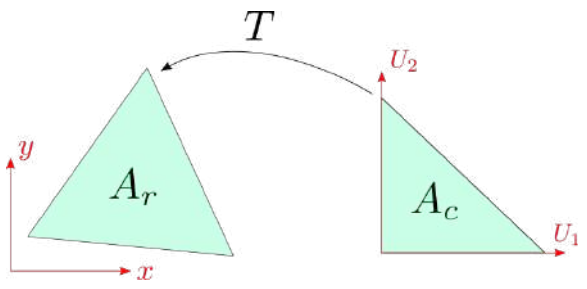


Figure 3.10: Transformation from canonical element (right) to mesh element (left).

Since the Jacobian is constant for each element, the determinant is the same at each point, an can be calculated as the ratio of the areas.

$$|J| = \frac{A_r}{A_c} \tag{3.32}$$

Eq. (3.33) presents an example on how the transformation of the integral is applied. The determinant of the Jacobian provides the relation between the differentials from each domain.

$$^oI_a \frac{\partial G_{ij}}{\partial x_k} = |J| I_a \frac{\partial G_{ij}}{\partial U_k} \tag{3.33}$$

Influence Matrices Assembly

Influence matrices $\mathbf{G}, \mathbf{F}, \mathbf{D}$ are assembled in such a manner so that the system of equations can be written in matrix form. The matrices are represented as block matrices for ease of understanding. Each row i contains all results of integrals for ξ_i w.r.t each X_j, $j = 1, 2, ..., M$.
Influence matrices are needed in two processes. For each process, they are assembled with different coefficients. For calculation of boundary conditions, the coefficients correspond to the results of integrals with ξ placed in B and X in B ($\mathbf{G}, \mathbf{F}$) or in S ($\mathbf{D}$). For recalculation of σ^o in S, the coefficients correspond to the results of integrals with ξ placed in S and X in B ($\mathbf{G}, \mathbf{F}$) or in S ($\mathbf{D}$). The coefficients in the diagonal could be the results of singular integrals but not necessarily for cases in which ξ and X are placed in different topological entities (B and S).

The composition of the influence matrix $\mathbf{G}$ is shown in Fig. 3.11. The positions in the diagonal ($i = j$) correspond to the transformed results of the singular integrals (Eq. 3.31) of the Green functions $G_{ij}(X - \xi)$. Additionally, the non-diagonal ($i \neq j$) correspond to the non-singular integrals (See section 3.3.5).

$$\mathbf{G} = \begin{array}{c} \\ \xi_1 \\ \xi_2 \\ \vdots \\ \xi_M \end{array} \begin{array}{c} \begin{array}{cccc} X_1 & X_2 & \dots & X_M \end{array} \\ \begin{bmatrix} {}^oI_aG^1 & & & \\ & {}^oI_aG^2 & & \\ & & \ddots & \\ & & & {}^oI_aG^M \end{bmatrix}_{2M \times 2M} \end{array}$$

Figure 3.11: Matrix $\mathbf{G}$ composition. Block matrix representation. Results from boundary integration of $G(X - \xi)$. ξ and X placed in B.

The composition of the influence matrix $\mathbf{F}$ is identical to matrix $\mathbf{G}$ (see figure 3.11), except with the results from the integrals of $F_{ij}(\xi - X_m, \hat{n})$ and the contribution of the diagonal c matrix (see table 3.3). For influence matrices $\mathbf{G}$ and $\mathbf{F}$, the sub-matrices ${}^oI_aG^m$ and ${}^oI_aF^m$ are of size 2×2.

The assembly of the influence matrix $\mathbf{D}$ is similar to both of $\mathbf{G}$ and $\mathbf{F}$. The only difference relies in the assembly of the sub-matrices ${}^oI_a\partial G$ shown in Fig. 3.12 (b).

$$\mathbf{D} = \begin{array}{c} \\ \xi_1 \\ \xi_2 \\ \vdots \\ \xi_L \end{array} \begin{array}{c} \begin{array}{cccc} X_1 & X_2 & \dots & X_L \end{array} \\ \begin{bmatrix} {}^oI_a\partial G^1 & & & \\ & {}^oI_a\partial G^2 & & \\ & & \ddots & \\ & & & {}^oI_a\partial G^L \end{bmatrix}_{2L \times 4L} \end{array}$$

(a) Influence matrix **D**

$${}^oI_a\partial G^l = \begin{bmatrix} {}^oI_a \dfrac{\partial G_{11}}{\partial x_1}^l & {}^oI_a \dfrac{\partial G_{12}}{\partial x_1}^l & {}^oI_a \dfrac{\partial G_{21}}{\partial x_1}^l & {}^oI_a \dfrac{\partial G_{22}}{\partial x_1}^l \\ \\ {}^oI_a \dfrac{\partial G_{11}}{\partial x_2}^l & {}^oI_a \dfrac{\partial G_{12}}{\partial x_2}^l & {}^oI_a \dfrac{\partial G_{21}}{\partial x_2}^l & {}^oI_a \dfrac{\partial G_{22}}{\partial x_2}^l \end{bmatrix}$$

(b) Sub-matrix ${}^oI_a\partial G$ composition for cell m

Figure 3.12: Matrix **D** composition. Block matrix representation. Results from surface integration of $\frac{\partial G(X-\xi)}{\partial x}$. ξ and X placed in S.

3.3.6 System of Equations

The $\widetilde{BEP}$ is written as a linear system of equations so that the unknown boundary conditions can be found. Consequently, the evaluation of the Eq. (3.12) for all elements in the boundary ($\xi_i,\ i = 1, 2, ..., M$), produces this system. It is written in matrix form as follows.

$$\mathbf{Gt} - \mathbf{Fv} - \mathbf{Gt^o} + \mathbf{D}\sigma^\mathbf{o} = \mathbf{0} \tag{3.34}$$

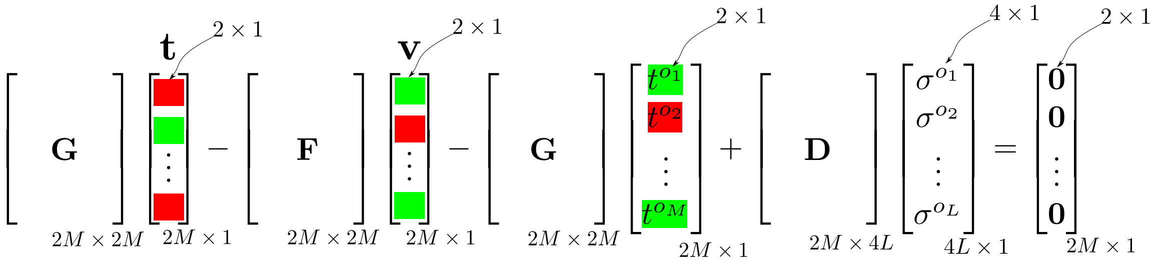


Figure 3.13: System of equations (Eq. 3.34). Known Boundary Conditions (green). Unknown Boundary Conditions (red). M: Total number of boundary B elements. L: Total number of surface S cells.

Eq. 3.34 can be reorganized and rewritten in terms of a vector of unknowns $\mathbf{x}$ and a vector of known boundary conditions $\mathbf{y}$,

$$\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{D}\sigma^{\mathbf{o}} + \mathbf{G}\mathbf{t}^{\mathbf{o}} - \mathbf{B}\mathbf{y} = \mathbf{0} \tag{3.35}$$

, for which matrix $\mathbf{A}(\mathbf{G}, \mathbf{F}, \mathbf{t}, \mathbf{v})$ corresponds to *unknown* boundary conditions. Matrix $\mathbf{B}(\mathbf{G}, \mathbf{F}, \mathbf{t}, \mathbf{v})$ corresponds to *known* boundary conditions.

3.3.7 Iterative Process

An iterative process is necessary due to the assumption that f_i is known for the linearization of Eq. (3.5). This linearization allows to formulate the BEM integral equation and converge the unknown boundary values, iterating over σ^o_{ki}.

The iterative process initializes σ^o_{ki}. Then, at each iteration i, the boundary values are calculated and σ^o_{ki} is updated with these new results. The iterative process is terminated once there is convergence observed in the solution. This happens when the fluctuations between states of iteration i and $i-1$ are insignificant.

1 To converge the boundary values due to the assumption of the term f_i in Eq. (3.5) performed by [34], an iterative cycle over this term must be executed.

2 The term σ^o_{ki} in Eq. (3.12) is the one to be iterated over.

3.4 Results

3.4.1 Numerical Examples

The numerical examples discussed next are solved using the boundary element method formulated previously. This examples are used to test the approximation with low order elements and analytic solutions of the singular integrals. The results are obtained with an implementation in python and compared with results obtained with ANSYS.

Domain Discretization

Discretization of the boundary B and the region S, for each example, is exhibited next. Elements used for discretization of the boundary and surface are discussed in section 3.3.4.

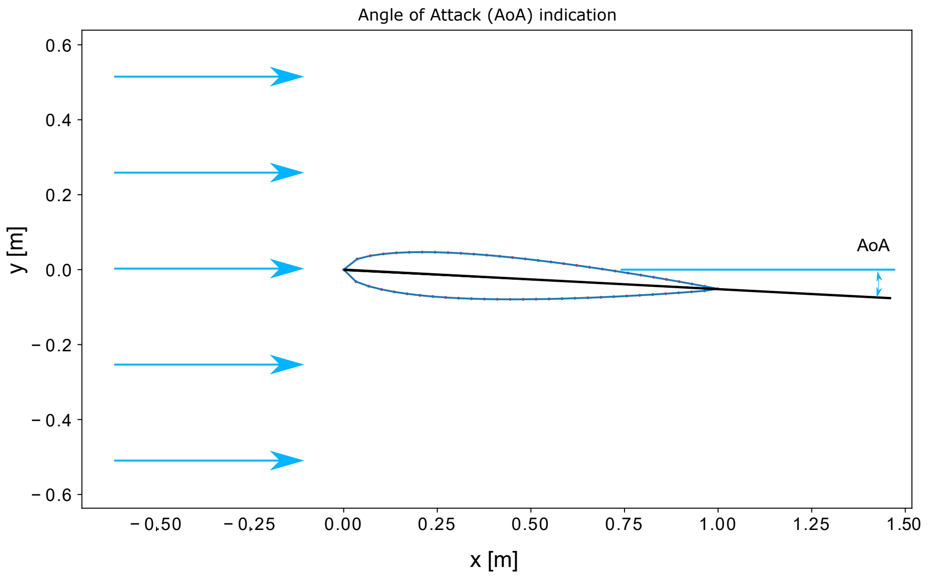


Figure 3.14: Indication of the angle of attack for the airfoils.

For the examples in Figs. 3.15- 3.19, the domains (Ω's) are semi-infinite. External boundary Γ_0 to Ω does not exist. For these, only a region around S is evaluated in order to observe the flow's behaviour around the submerged object.

For the example in Fig. 3.20, Ω is finite with $B = \Gamma_0$. A submerged object Γ_1 does not exist. The objective is to observe the bending of the flow's direction because of the 90^o angle elbow.

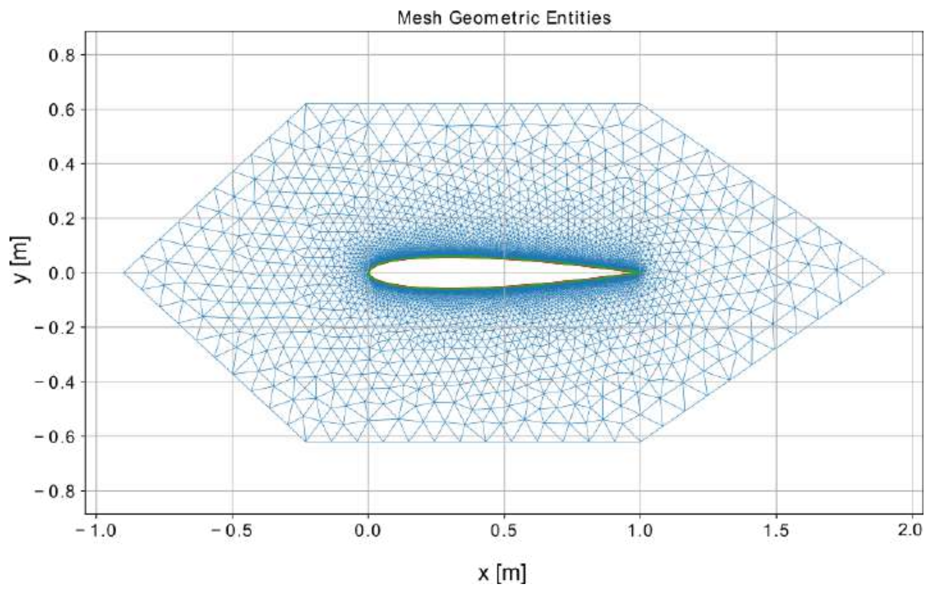


Figure 3.15: Mesh of airfoil at **0**° angle of attack.

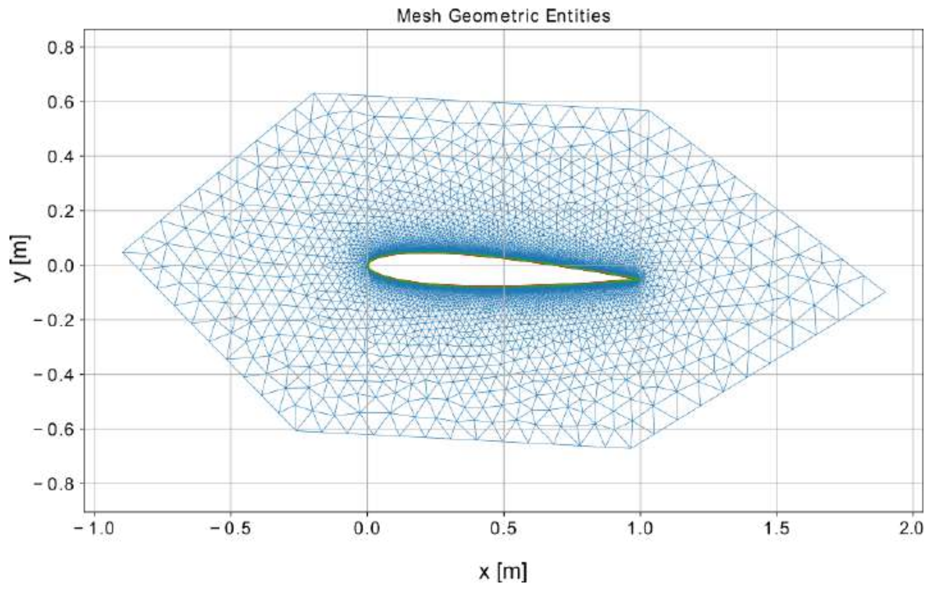


Figure 3.16: Mesh of airfoil at **3**° angle of attack.

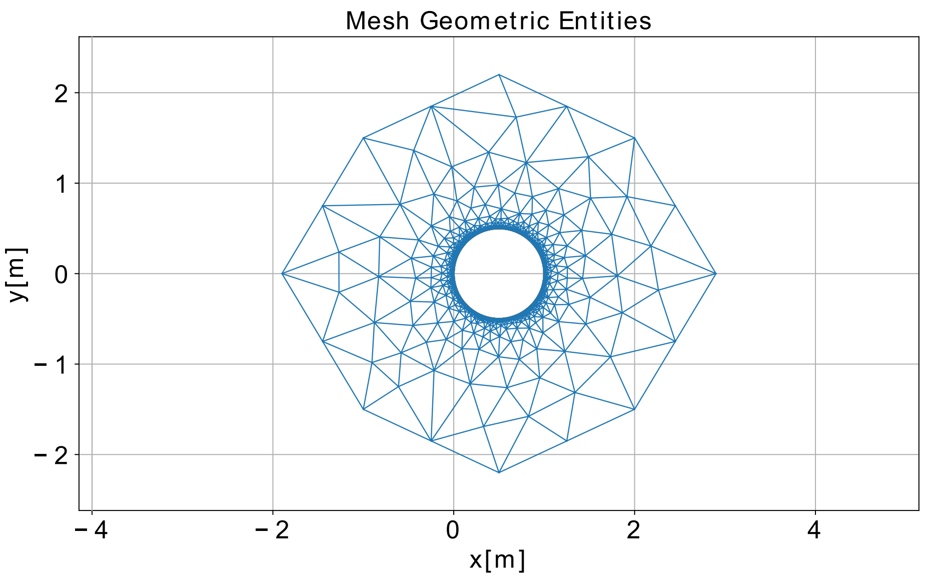


Figure 3.17: Mesh of **0.5m** radius circle. 2000 boundary elements. 7866 cell elements.

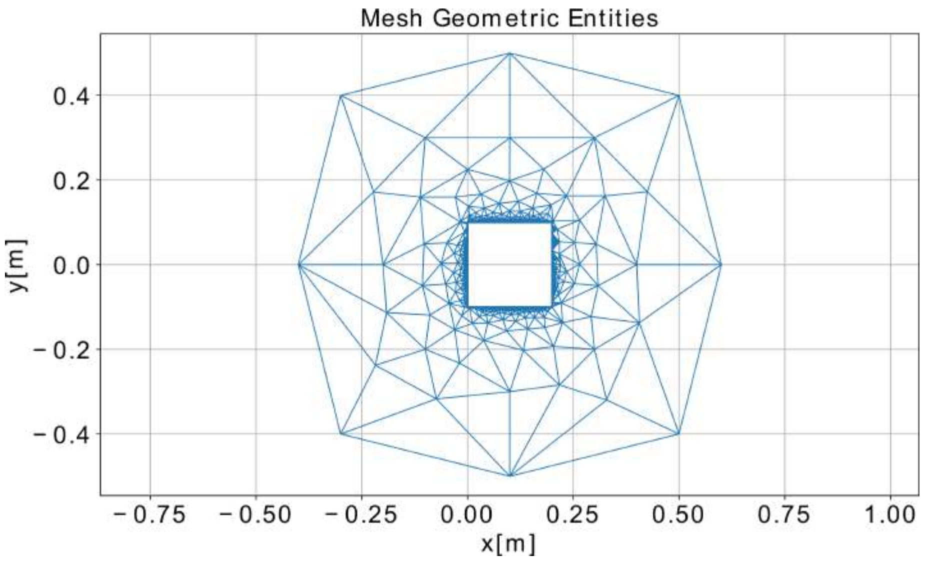


Figure 3.18: Mesh of **0.2m** side square. 2001 boundary elements. 4445 cell elements.

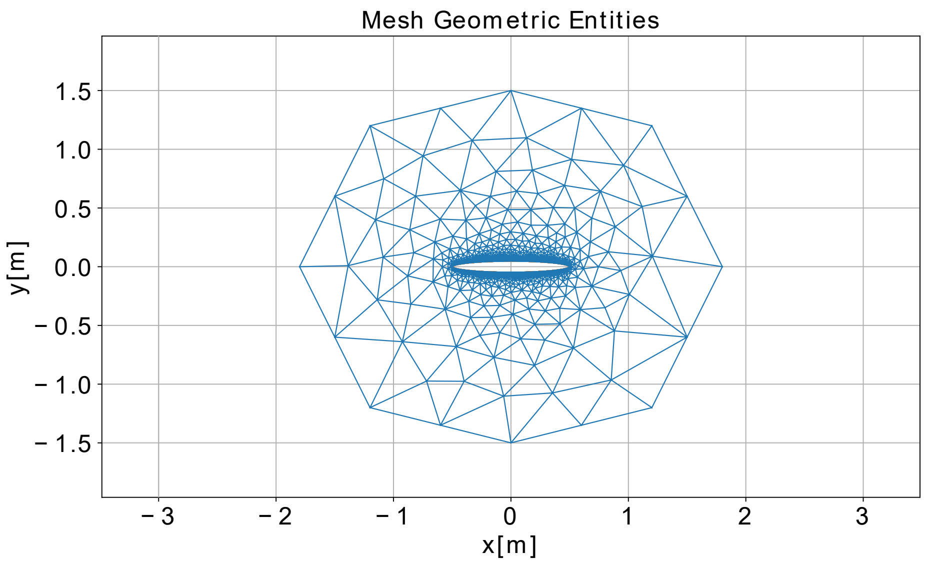


Figure 3.19: Mesh of ellipse. 2000 boundary elements. 11976 cell elements.

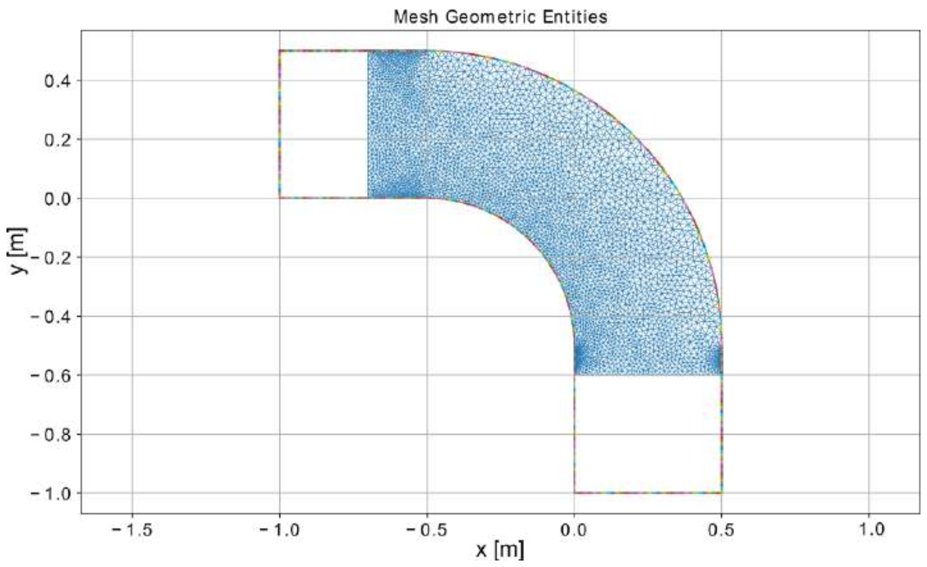


Figure 3.20: Elbow flow mesh. 448 boundary elements. 7732 cell elements.

Flow Conditions/Fluid Properties.
The flow conditions and fluid properties used for the simulation of the numerical examples are presented in Table 3.5.

Table 3.5: Flow conditions. Fluid properties. V_1: free flow velocity in x direction. V_2: free flow velocity in y direction.

$T[C^o]$	$\rho[\frac{kg}{m^3}]$	$\mu[\frac{m^2}{s}]$	$\nu[\frac{kg}{ms}]$	Re	$V_1[\frac{m}{s}]$	$V_2[\frac{m}{s}]$
15	1.225	1.81e-5	1.48e-5	2	$V_1(Re, \nu, c)$	0

The free flow velocity $V_1(Re, \nu, c)$ is different for each example. It depends on the characteristic length c of each domain.

Boundary conditions.
The boundary conditions are assigned solely to the boundaries Γ_i. For every example's Γ_i, a No-slip condition is defined as a Dirichlet boundary condition $u_i(\Gamma) = 0$. Consequently, the traction $t(\Gamma)$ is unknown.

3.4.2 Integral Coefficients Results for Green Functions $G_{ij}(X - \xi)$ and $F_{ij}(X - \xi)$.

The integral coefficients for Green Functions $G_{ij}(X-\xi)$ and $F_{ij}(X-\xi)$ for the example of the airfoil at **0**° Angle of Attack are presented. These results are given in order to observe the behaviour of the numeric and analytic integrals along the boundary.

Integral coefficients shown correspond to the numerical evaluation (3 point Gauss quadrature) for $X \neq \xi$ and the analytic scheme evaluation for $X = \xi$ in the boundary. Even though C^1 discontinuities can be seen at strong geometric changes ($\hat{n}_m \cdot \hat{n}_{m+1} << 1$), it is not our intention to solve this discontinuities but to take them into account in the behaviour of the solutions.

For Figs. (3.22) and (3.23), integral values at extremes $X = 0$ and $X = 1000$ which do not correspond to an analytic value, are a product of numerical evaluation. No special attention should be focused on these values.

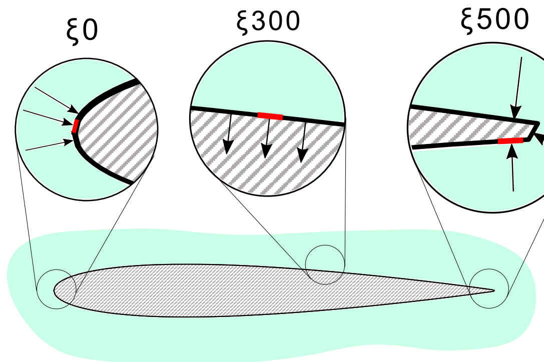
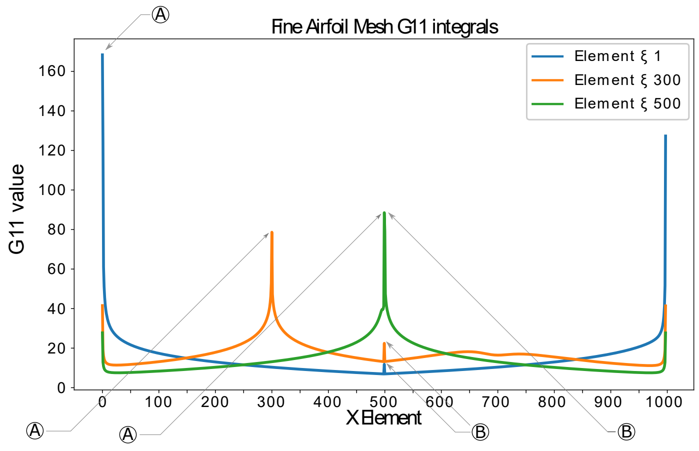
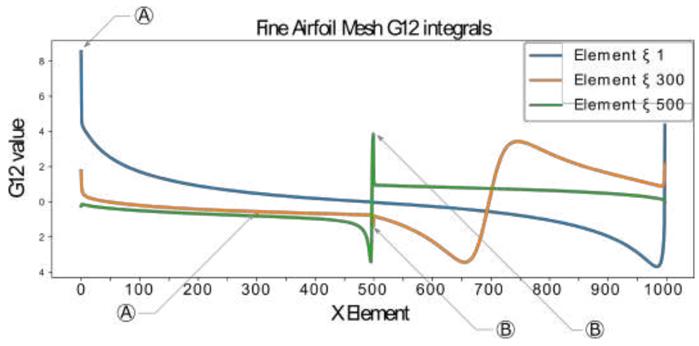


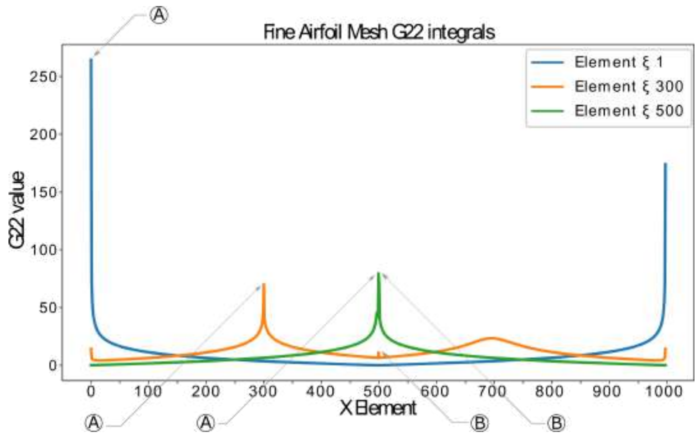
Figure 3.21: Selected neighbourhoods normals for Figs. 3.22 and 3.23.



(a) $G_{11}(X - \xi)$

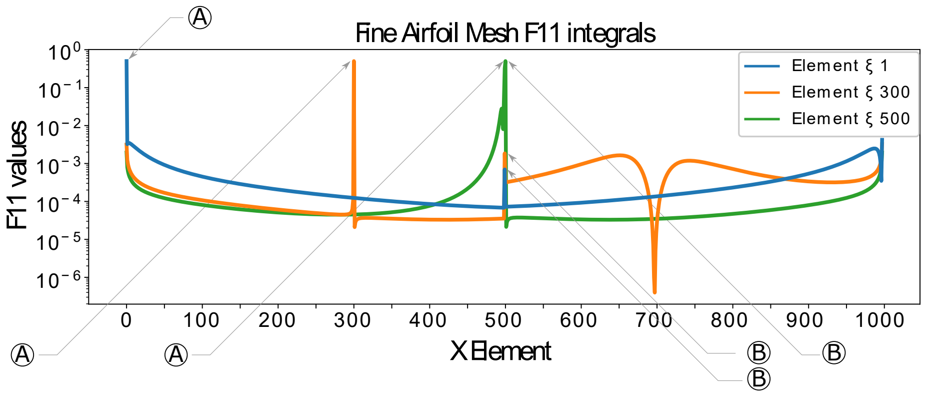


(b) $G_{12}(X - \xi)$

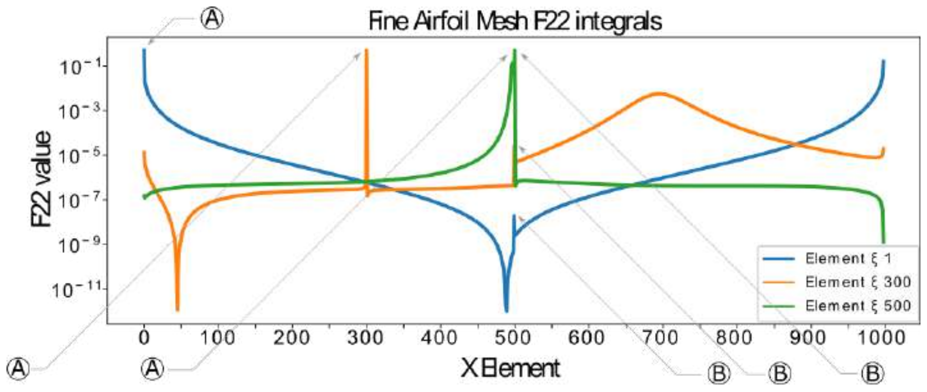


(c) $G_{22}(X - \xi)$

Figure 3.22: Analytic and numerical results for the integrals of Green function G. Airfoil at **0°** angle of attack (mesh in fig 3.15). A: Result of analytic integration for singular case in the boundary. B: C^0 Discontinuity produced by neighbourhood with strong normal changes.



(a) $F_{11}(X - \xi, \hat{n})$



(b) $F_{22}(X - \xi, \hat{n})$

Figure 3.23: Analytic and numerical results for the integrals of Green function F. Airfoil at **0**° angle of attack (mesh in fig 3.15). A: Result of analytic integration for singular case in the boundary. B: C^0 Discontinuity produced by neighbourhood with strong normal changes.

3.5 Conclusions and Future Work

This manuscript presents the simplification and evaluation of a fluid dynamics problem using the Boundary Element Method. The solution of the boundary element integrals is performed via the source node displacement method presented by [22, 23]. To the best of the authors knowledge, this singularity sorting method has not been implemented for the evaluation of this integrals in the specific fluid dynamics problem. In addition, low order elements are used for the discretization and numerical solution of the boundary element problem. This allows for a simpler and more understandable approach. Our implementation and algorithm predicts the flow characteristics to be expected around a submerged object and through changes in directions. Difficulties can be found in the precision of the velocity vectors. This may occur due to a not sufficient surface discretization, great variations in the normal vectors of boundary neighborhoods and low number of elements used. Future work is required in improving the performance when boundary normal vectors significantly differ within a boundary neighborhood, increasing the order of the elements, and the extension of the implemented methods for sorting the singularity to other scientific fields.

4
Conclusions

This work presents a compilation of manuscripts in applications of Computational Mechanics in which Computational Geometry and Numerical Simulation have a central role. Likewise, tools from mathematics, data structures, algorithms and programming fundamentals are essential for the developed approaches.

This compendium shows a novel method for rapid segmentation of 3D meshes with statistically poor triangulation. The results show a successful fitting of three different primitives (sphere, cylinder, cone) together with their support set, with a maximal error of ≈ 0.1 %. Therefore, it also shows that the precision is sufficient for the proposed industrial application. The proposed approach presents real-time execution for cases with continuity C-0. In contrast, C-1 or higher continuity introduces higher execution times.

This document also contributes to the area of numerical simulations for fluid dynamics. The method sorts the singularity in the boundary integral by applying a displacement to the source node. The results were compared with the commercial software ANSYS finding that the implementation predicts flow characteristics to be expected around a submerged object.

Finally, the different contributions presented here can be further extended. Therefore, future research can be focused on: (1) automatically detecting the type of primitive to fit given the input triangle and its neighborhood for our tool of analytic form fitting. (2) Increasing the order of the boundary elements, improving the performance when boundary normal vectors signicantly differ within a neighborhood.

5

Bibliography

[1] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[2] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data. *Computer Graphics Forum*, 38(1):167–196, 2019.

[3] Laurent Busé, André Galligo, and Jiajun Zhang. Extraction of cylinders and cones from minimal point sets. *Graphical Models*, 86:1–12, 2016.

[4] Oscar Ruiz, Santiago Arroyave, and Diego Acosta. Fitting of Analytic Surfaces to Noisy Point Clouds. *American Journal of Computational Mathematics*, 03(01):18–26, 2013.

[5] Rostislav Hulik, Michal Spanel, Pavel Smrz, and Zdenek Materna. Continuous plane detection in point-cloud data based on 3D Hough Transform. *Journal of Visual Communication and Image Representation*, 25(1):86–97, 2014.

[6] Tahir Rabbani and Frank Van Den Heuvel. Efficient Hough Transform for Automatic Detection of Cylinders in Point Clouds. *ISPRS Workshop on Laser Scanning*, 3:60–65, 2005.

[7] Marco Attene and Giuseppe Patanè. Hierarchical structure recovery of point-sampled surfaces. *Computer Graphics Forum*, 29(6):1905–1920, 2010.

[8] Dong Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. Variational mesh segmentation via quadric surface fitting. *CAD Computer Aided Design*, 44(11):1072–1082, 2012.

[9] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an RGBD Camera. *ACM Transactions on Graphics*, 31(6):1–12, 2012.

[10] Natasha Gelfand and Leonidas J. Guibas. Shape segmentation using local slippage analysis. *ACM International Conference Proceeding Series*, 71:214–223, 2004.

[11] Timur Bagautdinov, François Fleuret, and Pascal Fua. Probability occupancy maps for occluded depth images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:2829–2837, 2015.

[12] Jin Liu. An adaptive process of reverse engineering from point clouds to CAD models. *International Journal of Computer Integrated Manufacturing*, 33(9):840–858, 2020.

[13] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4), 2011.

[14] Oliver J. Woodford, Minh Tri Pham, Atsuto Maki, Frank Perbet, and Björn Stenger. Demisting the hough transform for 3D shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341, 2014.

[15] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, 78(2-3):223–236, 2008.

[16] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation. *International Journal of Computer Vision*, 99(1):69–85, 2012.

[17] Roseline Bénière, Gérard Subsol, Gilles Gesquière, François Le Breton, and William Puech. Recovering primitives in 3D CAD meshes. *Three-Dimensional Imaging, Interaction, and Measurement*, 7864(January 2011):78640R, 2011.

[18] Truc Le and Ye Duan. A primitive-based 3D segmentation algorithm for mechanical CAD models. *Computer Aided Geometric Design*, 52-53:231–246, 2017.

[19] Sylvain Lefebvre and Hugues Hoppe. Perfect spatial hashing. *ACM SIGGRAPH 2006 Papers, SIGGRAPH '06*, 1(212):579–588, 2006.

[20] Daniel Mejia-parra, Juan Lalinde-pulido, Jairo R. Sánchez, Oscar Ruiz-salguero, Jorge Posada, C A M Cae, and Universidad Eafit. Perfect Spatial Hashing for Point-cloud-to-mesh Registration. *Conferencia Española de Informática Gráfica (CEIG'19)*, pages 1–9, 2019.

[21] M. Mantyla. *An Introduction to Solid Modeling.* Computer Science Press, Rockville, Maryland, USA, 1988. ISBN-13: 978-0881751086.

[22] C Balakrishna, LJ Gray, and JH Kane. Efficient analytical integration of symmetric galerkin boundary integrals over curved elements; elasticity formulation. *Computer methods in applied mechanics and engineering*, 117(1-2):157–179, 1994.

[23] LJ Gray, Luiz F Martha, and AR Ingraffea. Hypersingular integrals in boundary element fracture analysis. *International journal for numerical methods in engineering*, 29(6):1135–1158, 1990.

[24] Li Yuan, Mao Wentao, Wang Gangsheng, Liu Jing, and Wang Shixun. A general-purpose machine learning framework for predicting singular integrals in boundary element method. *Engineering Analysis with Boundary Elements*, 117:41–56, 2020.

[25] Xiao-Wei Gao, Jin-Bo Zhang, Bao-Jing Zheng, and Ch Zhang. Element-subdivision method for evaluation of singular integrals over narrow strip boundary elements of super thin and slender structures. *Engineering Analysis with Boundary Elements*, 66:145–154, 2016.

[26] JC Lachat and JO Watson. Effective numerical treatment of boundary integral equations: a formulation for three-dimensional elastostatics. *International Journal for Numerical Methods in Engineering*, 10(5):991–1005, 1976.

[27] Guizhong Xie, Yudong Zhong, Fenglin Zhou, Wenliao Du, Hao Li, and Dehai Zhang. Singularity cancellation method for time-domain boundary element formulation of elastodynamics: A direct approach. *Applied Mathematical Modelling*, 80:647–667, 2020.

[28] Abdel-Magid Abdel-Latif Abdalla. Solution of viscous flow problems by using the boundary element method. 1992.

[29] RGR Camacho and JR Barbosa. The boundary element method applied to incompressible viscous fluid flow. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 27(4):456–462, 2005.

[30] Xiao-Wei Gao. An effective method for numerical evaluation of general 2d and 3d high order singular boundary integrals. *Computer methods in applied mechanics and engineering*, 199(45-48):2856–2864, 2010.

[31] YJ Liu. On the simple-solution method and non-singular nature of the bie/bem-a review and some new results. *Engineering analysis with boundary elements*, 24(10):789–795, 2000.

[32] JJ Pérez-Gavilán. Introducción a los elementos de frontera. *México, Consejo Nacional de Ciencia y Tecnologıa*, 2006.

[33] JCF Telles. A self-adaptive co-ordinate transformation for efficient numerical evaluation of general boundary element integrals. *International journal for numerical methods in engineering*, 24(5):959–973, 1987.

[34] Prasanta Kumar Banerjee and Luigi Morino. *Boundary Element Methods in Nonlinear Fluid Dynamics: Developments in boundary element methods-6*. CRC Press, 1990.