Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

STL_Process: A .STL-based preprocessor for robot path planning in manufacturing and quality control processes

Abanti Shama Afroz [a], Francesco Inglese [a], Cesare Stefanini [a,b], Mario Milazzo [a,*]

[a] *The BioRobotics Institute, Scuola Superiore Sant'Anna, Viale Rinaldo Piaggio 34, 56025 Pontedera, PI, Italy*
[b] *The Healthcare Engineering Innovation Center (HEIC), Khalifa University, Abu Dhabi, PO Box, 127788, United Arab Emirates*

A R T I C L E   I N F O

Article history:
Received 11 December 2020
Received in revised form 24 May 2021
Accepted 25 May 2021

Keywords:
Computer-aided manufacturing
Trajectory planning
Feature extraction
Human–machine interface

A B S T R A C T

One of the most commonly used file formats in manufacturing industry is .STL. We developed a semi-automated C++ -based preprocessor with a command line interface and an OpenGL-based visual module to extract topological features from simplified 3D volumes presented in .STL file format. The preprocessor evaluates input 3D models in five major steps: (i) edge identification, (ii) 2D regional grouping, (iii) region characterization, (iv) 3D sub-volume extraction and (v) virtual cross-section generation. It has been designed to aid online robot programming by offering an intuitive and user-friendly system to robot operators for planning additive and subtractive mechanical operations, as well as improve non-destructive quality control activities.

Code metadata

Current Code version	*v1*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00094
Legal code license	*MIT*
Code versioning system used	*Git*
Software code language used	*C++*
Compilation requirements, Operating environments & dependencies	*Microsoft Visual Studio 2015 in Windows 2010*
If available link to developer documentation/manual	https://github.com/AbantiS/STL_process/blob/master/README.md
Support email for questions	afroz.abantishama@gmail.com, abanti2007@gmail.com

Software metadata

Current software version	*v1*
Permanent link to executables of this version	https://github.com/AbantiS/STL_process/tree/master
Legal software license	*MIT*
Computing platform/Operating system	*Windows 10.*
Installation requirements & dependencies	*Requires Eigen and OpenGL library for C++*
If available link to user manual - if formally published include a reference to the publication in the reference list	*Not published yet*
Support email for questions	afroz.abantishama@gmail.com, abanti2007@gmail.com

1. Introduction

The recent advancements in automation and robotics have greatly improved industrial processes in terms of efficiency and productivity. Computer-aided design (CAD) and manufacturing (CAM) have played a significant role to simplify and optimize

* Corresponding author.
E-mail address: mario.milazzo@santannapisa.it (Mario Milazzo).

https://doi.org/10.1016/j.softx.2021.100725

the developmental stages of complex products. The outputs of such procedures are 3D virtual models that are used as inputs to manufacturing machines or robotic assemblies for the actual fabrication and assembly. Many solutions have been developed as an extension to commercial CAD drawing packages [1,2], and as plugins for robot processing software programs [3].

The current procedures, based on a CAD-file dependency, are limited by two main factors: 1) they require original CAD files with all the information related to the fabrication of structural features. In practice, such files are often either unavailable or partially edited on site without following the references of the original CAD files [4–6]. 2) In case of robot-guided inspections with 2D laser scanners [7–9], it is beneficial to have reference templates [10,11]. Conventionally this task is completed by directly comparing volumetric topologies [9,12,13], and require a large computation power and additional tools such as Geomagic [13] or Atos [11].

As an alternative to complex CAD files, researchers have also developed simplified file formats. The most popular is the .STL, from the word "stereolithography" since this format was developed for the so-called additive manufacturing technology [14, 15]. The difference of this format with respect to the traditional CAD file is the absence of detailed topological information (e.g., edges, surfaces) of the design approaches used to create features (e.g., holes, protrusions) and the 3D structures are represented by a collection of connected triangles. Most of the published works, showing the extraction of topological features from .STL models, have been carried out with commercial platforms (e.g., MATLAB) [16–18]. However, they are application-dependent and do not provide any general solution for extracting topological features from .STL models [19–23].

Based on this premise, our work describes an open source C++ based pre-processor for feature extraction of 3D structures from a .STL file to support robot path planning of toolpath generation and product quality inspection.

2. Problems and background

In .STL files, 3D volumes appear as a collection of triangles (T_i), interconnected through their vertices (v_{ij}) and sides (L_{ij}). Each topological feature is mathematically expressed by a matrix as follows:

$$v_{ij} = [x_j \; y_j \; z_j] \qquad T_i = [v_{i1} \; v_{i2} \; v_{i3}] \qquad L_{ij} = [v_{i1} \; v_{i2}] \tag{1}$$

in which x_j, y_j, z_j are the coordinates of the vertices of the triangles [19,24] (Fig. 1).

This highly-simplified .STL format has been exploited in both additive [19,25] and subtractive [26] technologies, as well as in reverse engineering processes [27,28]. The extraction of features from a virtual geometry, simplified with the .STL format, has been the objective of a large number of studies [16,19,22,23]. An efficient procedure allows, indeed, the accurate generation of tool paths both for internal and external features [21,29–31]. Concerning the specific case of additive manufacturing, this approach helps defining the amount of support material required to frame the actual structures [21,32] by slicing the volume along the so-called build direction [20]. Additional applications include the automation of manufacturing and assembly processes (e.g., coupling parts [33,34] or welding structures [35–37]), to develop efficient multiphysics and probabilistic simulation of complex systems [38–40], path planning for robotic spray painting [41], as well as quality control inspection [9,17,42–44]. Moreover, the automatic estimation of the tool paths from a .STL file may be crucial in manufacturing processes that are performed in different geographic locations [45] and in geo-scattered, small manufacturing units, a concept which turned out very promising during the present pandemic situation [46].

However, there is the absence of a general automatic solution that can detect volumetric features for processes not limited to direct additive manufacturing. There is also room for improving non-destructive process quality control procedures in terms of computational memory consumption. This can be achieved by using data, extracted from .STL models, and interpolated to match cross-sections from laser profilers for a comparison.

The software here described aims at overcoming the limitations of the state-of-the-art systems through a C++ -based application able to extract geometric features semi-automatically from the components whose .STL file is available. Our platform is driven by a command line interface and an OpenGL-based visual module to present the results of an estimated maneuver for a later employment in the manufacturing chain. In order to validate the approach, three models were used as case studies in which the features propagate only along the thickness (Fig. 2). Model 1 is a block with six randomly positioned different blind holes. Model 2 possesses two similar semi-open grooves, while Model 3 consists of a circular plate with seven open homogeneous curved grooves. These models were selected since in manufacturing industry, similar designs are used for applications like welding, gluing, or inserting. The most common of these forms were utilized to generate these sample blocks. We designed each model with Solidworks 2016 (Dassault Systèmes, Johnston, RI, USA), using cutting/extrusion design features along the z-axis, as it occurs in subtractive and additive manufacturing processes [20], and eventually exported them to the .STL format.

3. Software framework

The developed approach consists in processing an input volume with features propagating along a specific direction, given in the .STL format, and extrapolating its mechanical features to reconstruct the topological operations originated in the CAD file for a later employment as tool paths in additive or subtractive operations. The 3D volume (V_M) is composed of a number of volumetric features (V_{F_i}) immersed in a surrounding region or bounding surface (V_S), so that

$$V_M = V_S + \sum_i V_{F\_i} \tag{2}$$

The software procedural approach was illustratively explained using one of two 3D hypothetical volume samples reported in Fig. A.1 of Appendix. The first volume in Fig. A.1 has a cylindrical extrusion and a path symmetric intrusion and the second volume possesses two cylindrical volumetric intrusions. The algorithm processes only one volumetric feature at a time. The algorithm starts by calculating gross mechanical features followed by identifying the feature plane (Π_i^*), parallel to the feature's propagation, where the major geometric variations are recognized. This is followed by the evaluation of the feature edges given by the collection of triangle sides (L_i^*) related to the geometric feature contours. A sub-algorithm works on the group of L_i^*, defining in a new matrix the sequence of L_i^* that forms the planar feature Π_i^*. Contours are then cataloged and associated to the related sub-volume before using a new section plane Π_{i+1}^*, parallel to Π_i^*, before repeating the procedure. The envelope of the features along the direction of propagation gives the description of the topological feature and the tool path for later uses.

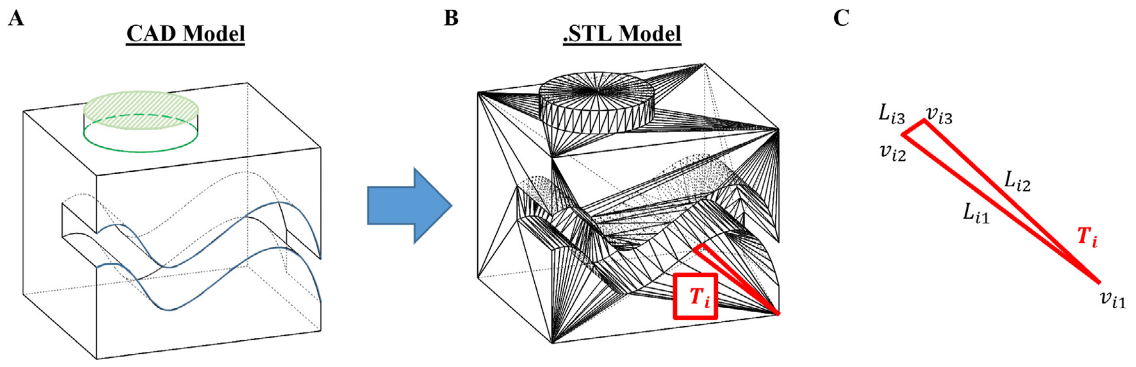

Fig. 1. From CAD model (A) to .STL model (B). Panel C reports the mathematical features (vertices - v_{ij}, and sides L_{ij}) of a generic triangle (T_i) composing the .STL model.

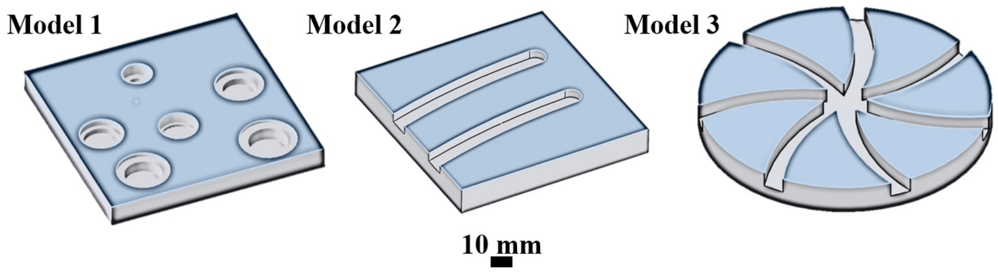

Fig. 2. Three models used as case studies. Model 1 is a block with blind holes randomly positioned in the volume. Model 2 is a block with two off-set grooves. Model 3 presents seven open grooves distributed along the circumference.

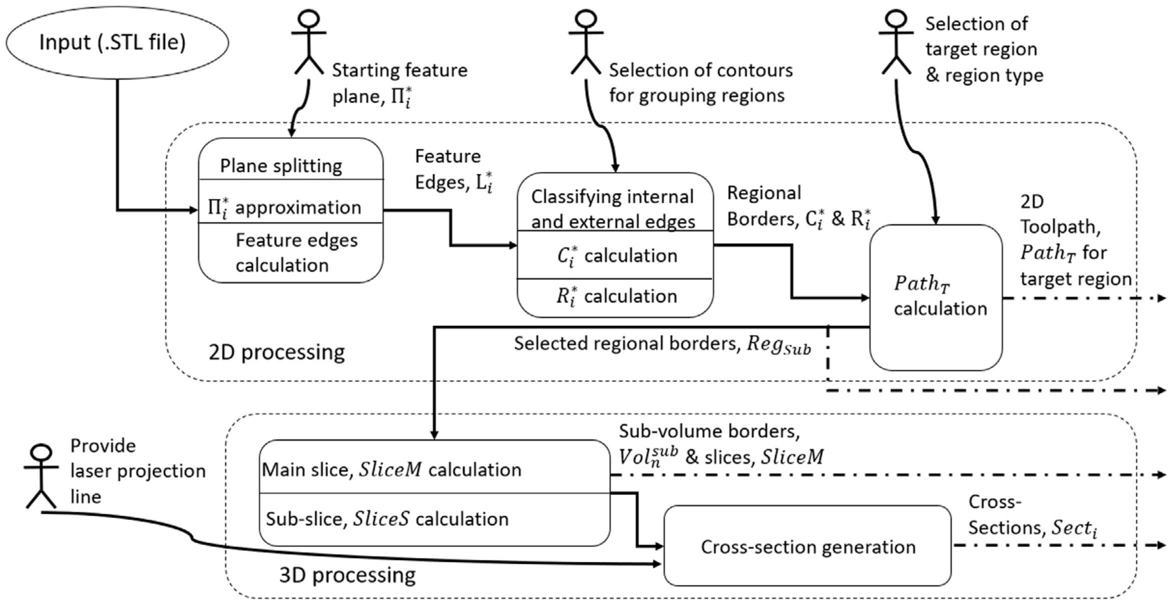
Input (.STL file)
Starting feature plane, Π_i^*
Selection of contours for grouping regions
Selection of target region & region type
Plane splitting
Π_i^* approximation
Feature edges calculation
Feature Edges, L_i^*
Classifying internal and external edges
C_i^* calculation
R_i^* calculation
Regional Borders, C_i^* & R_i^*
$Path_T$ calculation
2D Toolpath, $Path_T$ for target region
2D processing
Selected regional borders, Reg_{Sub}
Provide laser projection line
Main slice, $SliceM$ calculation
Sub-slice, $SliceS$ calculation
Sub-volume borders, Vol_n^{sub} & slices, $SliceM$
Cross-section generation
Cross-Sections, $Sect_i$
3D processing

Fig. 3. A schematic of the first .STL pre-processor where black angular arrows indicate data flow from modules, black spline arrows indicate input from user, black dashed arrows indicate usable outputs. Black boxes indicate primary modules while dashed boxes indicates 2D and 3D analysis phases.

3.1. Software architecture

A schematic representation of the software procedure for the C++ -based pre-processor is depicted the in Fig. 3. The inter-relationship of the pre-processor modules and their user input requirements were summarized in this schematic picture and described in the following sections.

3.2. Software functionalities

The code functionalities were organized into five major modules: (i) feature edge identification, (ii) contour generation and regional grouping, (iii) 3D region characterization, (iv) 3D sub-volume extraction and (v) virtual cross-section generation. The action of these modules can be grouped into two phases: 2D and 3D analysis phases. The detailed functionalities are described in the following section.

3.2.1. *Feature plane identification*

We first slice V_M along the build direction with a finite number of equidistant planes (P_i) [47]. We define as working plane Π_i, the geometry achieved with the Boolean operation:

$$\Pi_i = V_M \cap P_i \tag{3}$$

that contains the following sets of topological features (TF): vertices - v^{Π_i}, triangles - T^{Π_i}, lines - L^{Π_i}. Note that for simple/small models v^{Π_i} can be void. Feature planes, Π_i^* are defined as those planes that have a significantly large size of v^{Π_i} with respect to other Π_i planes. Panel B of Fig. A.2 (Appendix) depicts the evaluation of Π_i^* and Π_i (the four red planes intersecting the geometric features are feature planes Π_i^* against blue Π_i planes). Panel C of the same figure depicts the 2D planar view of the topmost feature plane, where v^{Π_i} components are marked in red.

3.2.2. *Identification of contours*

We use the automatic procedure developed in [21] to detect the TF for each Π_i^*. We define as feature edges ($L^{\Pi_i^*}$) those segments that belong to triangles with only two vertices on Π_i^*, and, thus, form border boundary features. From these, those that have both the vertices on the external periphery are referred to as external $L^{\Pi_i^*}$ and the rest as internal $L^{\Pi_i^*}$. Panel B of Fig. A.3 (Appendix) reports, as an example, in green and in blue colors any internal and external $L^{\Pi_i^*}$, respectively. This step is followed by extraction of internal contours ($C^{\Pi_i^*}$), which are internal contours connected to each other by a common shared vertex. Note that depending on the selected contour type, contours are merged to bind a single region (open slot type) or used to define a single region (semi-open and closed slot type). Thus, a sub-region is defined by Eq. (4).

$$Reg_{Sub} = \sum C^{\Pi_i^*} \tag{4}$$

3.2.3. *2D tool-path generation*

Once the planar feature is recognized and mathematically described by $C^{\Pi_i^*}$ and Reg_{Sub}, the next step of the procedure consists in defining the generic 2D tool-path to create the sub-volumetric feature by robotic tool.

In the cases of open/semi-open/closed grooves/protrusions, the 2D tool-path can be efficiently defined by the construction symmetry line between contours or by boundary derived lines (Panel A of Fig. A.4). Here, a dedicated algorithm localizes the contours that bound the groove/protrusion region and segment by segment of $C^{\Pi_i^*}$, calculates the center point of the normal lines that connect correspondent vertices. The envelope of such center points is thus reduced to a centerline and stored for a later use as a tool path, $Path_T$. For the second case, i.e., the specific case of blind/through holes, the path is simply a straight line along the building direction (Panel B of Fig. A.4).

3.2.4. *Sub-volume border extraction*

Every separated 2D boundary, Reg_{Sub}, is adjacent to its own sub-volume, Vol_{Sub}. From a topological standpoint, any sub-volume is composed of multiple layers, containing at least one upper and one lower layer, and these layers are defined as slices of a sub-volume. A slice of a sub-volume (V_{F_n}) is a 2D geometry generated where a plane (P_m) intersects the sub-volume. It is defined as a main slice ($SliceM$) if its boundary points originally exist in the .STL file, or as a sub-slice ($SliceS$) if the boundary points are interpolated within the sub-volume. Main and sub-slices of a sub-volume are depicted in panel B Fig. A.5. Therefore, a sub-volume can also be represented as a set of main slices in form of Eq. (5).

$$Vol_n^{sub} = \sum_i SliceM_i \tag{5}$$

The process starts with generating an intermediate 3D model from its parent Reg_{Sub}. The model initially considers all the vertices connected directly with the vertices of the parent Reg_{Sub} and thus the intermediate model includes all triangles having common vertices in the selected sub-region. The 2D sub-region is considered as the first slice, $SliceM_1$. For each iteration, the algorithm looks to see if there exists another groove like a boundary for the targeted sub-region along the build direction of the volume or in the same plane. The pseudo-code is following –

Pseudo-code for the sub-volume, Vol_n^{sub} extraction

```
1: Take input from parent 2D regional border, $Reg_{Sub}$
2: The input 2D sub-region is considered as the first slice, $SliceM_1$
3: Consider all the vertices connected directly with the vertices of the parent $Reg_{Sub}$.
4: Keep only the points which are along the negative build direction and fall under a dilated shadow zone,
5: Apply contour extraction algorithm, check if there exists another connected boundary either in the lower or in the same plane.
5: Iterate through step 2–5, till the last layer arrives, where no more connected vertices are not found.
```

Fig. A.5 shows this sub-algorithm step in details where sub-volume (V_{F_2}) has been extracted and contains two main slices ($SliceM_1$ & $SliceM_2$).

3.2.5. *3D fine segment template generation*

When the toolpath and sub-volume of a corresponding sub-region are available, virtual templates can be generated for integration with non-destructive inspection systems like laser scanners.

Pseudo-code for the cross-section generation for path symmetric regions

```
1: Take input from the main slices and 2D tool path.
2: For each point ($i$) of the center symmetric toolpath ($Path_T$) provide a line for the laser projection
3: For each projection line Interpolate the external border from the slices along the laser projection line and compute $Sect_i$.
```

Fig. A.6 depicts a sub-region boundary with its path descriptor are shown with red and blue colors, respectively and the user defines the projection line position, marked in green dashed line, over the surface volume (V_S) (Panel A). The determined intrusive sub-volume can be seen in Panel B and the virtual cross-sections, marked with various combinations of black color and dashed patterns in Panel C. Such templates can be easily compared with profile scans obtained with other hardware resources like non-destructive test scanners.

4. Implementation and results

We have implemented a practical approach to identify sub-volumetric features in three-dimensional components starting from a .STL file without relying on original CAD sketches. Our procedure may serve as an intuitive and user-friendly system for operators processing mechanical components with either additive or subtractive manufacturing procedures. We use three different case studies to validate our approach, determining the performance of the system with different topological conditions, namely closed, semi-open, and open volumetric features.

The preprocessor was written with C++ language and was implemented with Microsoft Community Visual Studio Environment 2015 and contains a text-based interface. The user can visualize the end-results through an intuitive OpenGL-based interface specifically developed for this purpose. The software allows the simultaneous analysis of the 3D view, 2D top view along the building direction with depiction of the detected contours, and a zoomed view of the selected region of interest. Fig. 4 reports the graphic user interface developed in the framework of this study. It

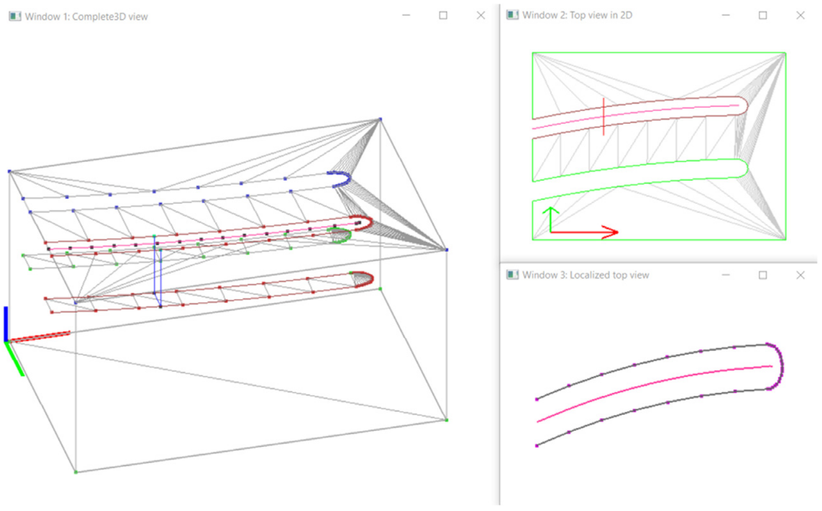

Fig. 4. Graphical user interface to study a .STL virtual mode. Left window shows the 3D view of the model while the right windows depict the 2D top view along the building direction and a zoomed view of the selected region of interest. The colored lines refer to the topological features recognized by the system.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

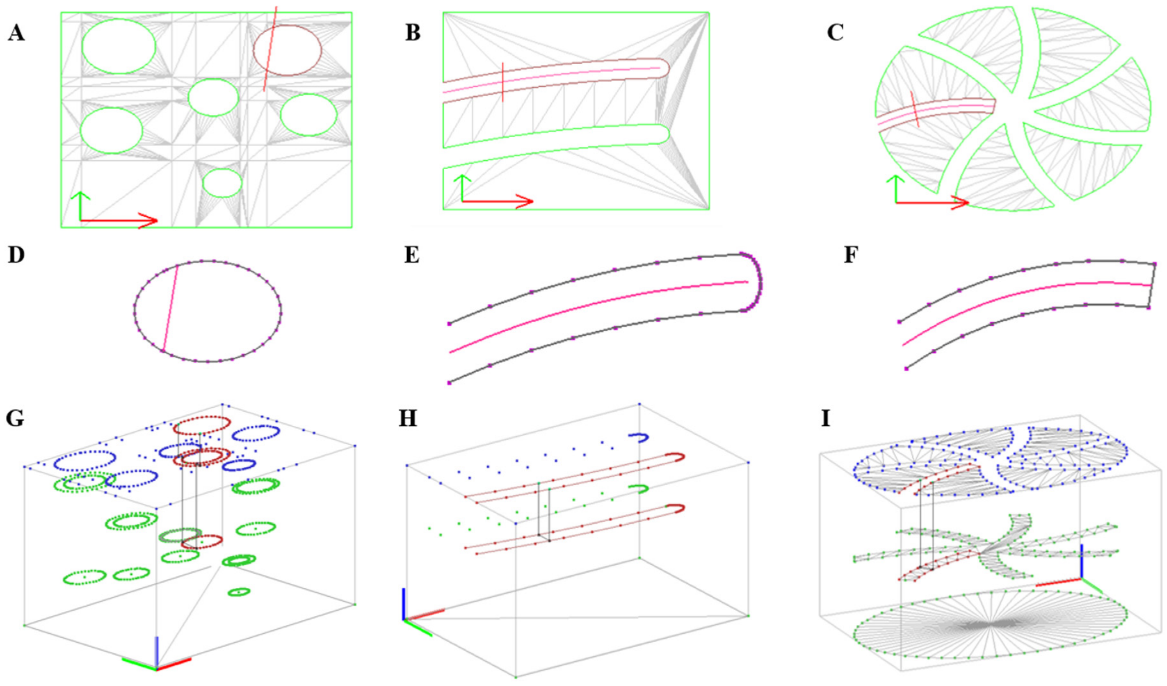

Fig. 5. Analysis of the three case studies. Panels A, B and C show the top views of block 1, 2 and 3 where green line marker depicts the feature lines, maroon color is used to highlight the borders of the selected region. Panel D, E and F separately show the selected regions of block 1, 2 and 3 with gray lines and purple markers, magenta lines indicate either the path descriptor line (Panels E, F) or user defined scanning pose (Panel D). Panel G, H, I show the 3D views of the case studies: dark blue markers depict the vertices on the first feature plane, main slices are depicted in red, and one projection of the template is marked with solid black lines.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is important to note here that, in the 3D view panel, the building direction (+z) view is magnified against the x- or y-axis view to emphasize the topological changes along the building direction.

Three test geometries with multiple sub-volumes have been used to assess the developed pre-processor. The comparative outcomes are reported in Fig. 5.

For this implementation it was observed that feature-vertex identification, projection line calculation and virtual template calculation were achieved in a reduced amount of time in the order of milli seconds. In contrast, feature-edge connectivity detection and sub-slice calculation required much more time. This is related to the fact that the algorithm evaluates all the considered triangles and connectivity data in the mentioned steps. However, although these steps require a processing time in the order of seconds, this procedure ensures an increased efficiency over pure online robot path planning approach. Table A.1 of Appendix summarizes the computation time required for each step. Additionally, we analyzed the primary memory allocation while running the software. We observed that data processing before visualization takes a maximum of 3 MB. While OpenGL visualization modules causes a jump in of around 17 MB. These results indicate that developed system can be implemented in embedded visualization systems as well. Numerical findings were summarized in Table A.2 of Appendix.

For each complete analysis of a model, the user has to provide multiple input selection values. In view of this, we selected three

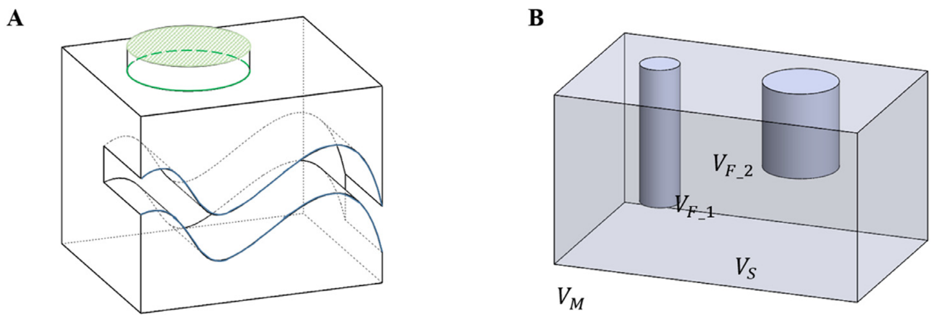

Fig. A.1. From .STL model to planar feature identification. Panel A: Sample volume (V_M) composed of a cylindrical extrusion (V_{F_1}) and a path symmetric intrusion (V_{F_1}) within the surface volume (V_S). Panel B: Sample volume (V_M) with two sub-volumetric features (V_{F_1} and V_{F_2}) merged within the surface volume (V_S).

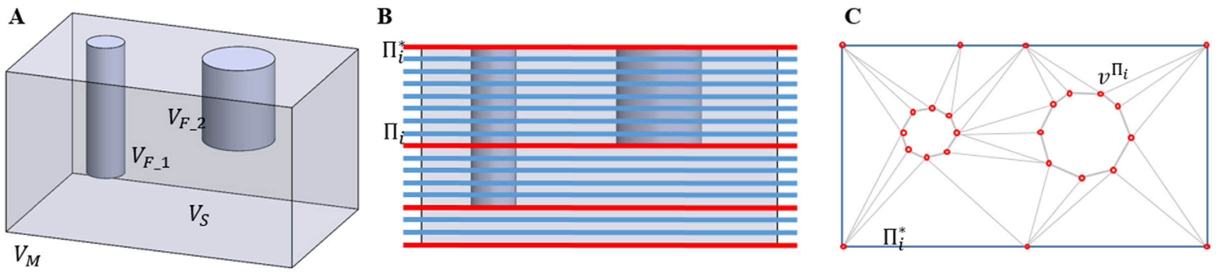

Fig. A.2. From .STL model to planar feature identification. Panel A: a block (V_M) composed of two sub-volumetric features (V_{F_i} and V_S). For clarity, the .STL model is shown as a CAD model. Panel B: volume slicing and identification of Π_i^*, planes with planar geometrical features. Panel C: visualization of feature vertices (v^{Π_i}) on the top most feature plane in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

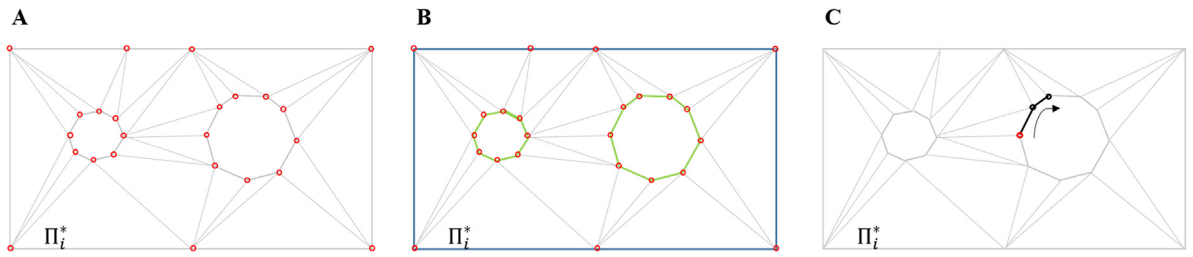

Fig. A.3. Panel A: Visualization of vertex components (v^{Π_i}) of the feature plane in red. Panel B: Identification of the sides (L^{Π_i}) composing of regular edges (in gray), internal feature edges (in green), external feature edges (in blue), and planar features (in green). Panel C: Contour identification from a starting point v_{in} (in red — selected by the user) progressive contour $C^{\Pi_i^*}$ identified on Π_i^* in black. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

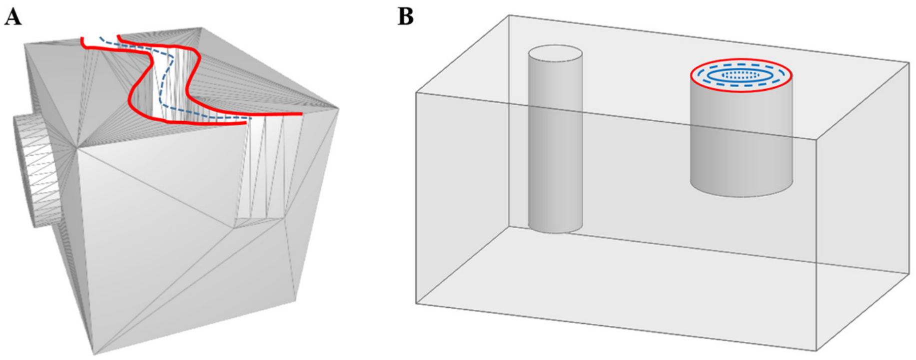

Fig. A.4. Tool path generation from symmetry-based and boundary-based samples. Panel A. A 3D volume sample with a cylindrical extrusion from the left-side plane, and a path-based intrusion from the top plane, which is the singulated sub-region and therefore marked in red. The curvilinear description of the path is reported with a blue dashed line. Panel B. A 3D volume with two cylindrical intrusions from the top plane. Here, the right-hand cylindrical volume is the targeted sub-volume for the analysis. Since the sub-region has a cylindrical border, marked in red, it is considered as the parent path descriptor. Such path descriptors can be utilized with additional algorithms to generate either cutter location source data [1] for subtractive operations or to estimate the extrusion path [3] for additive operations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

different scenarios for each of the three case studies, and repeated each of the nine scenarios ten times. Note that, as a convention, we define as a scenario a specific set of inputs related to a

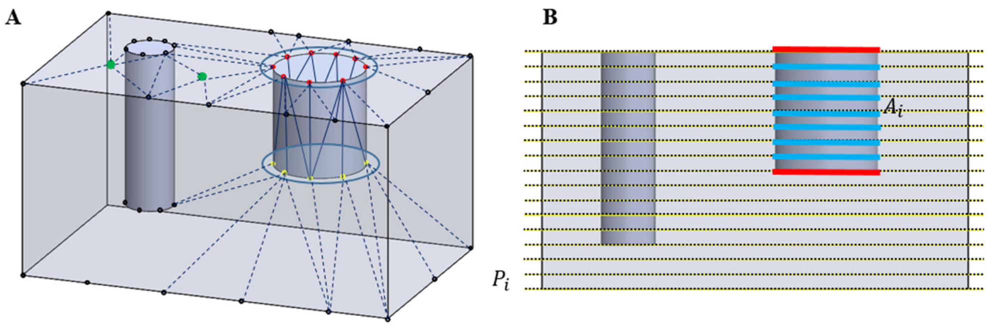

Fig. A.5. Extraction of sub-volumes. Panel A. The sub-region of interest has vertices that are marked in red and forms the first main slice. For the next main slice extraction, all vertices connected with the red ones are initially taken into account, However, only the vertices that fall within the shadow region, shown with blue circles, are selected as the sub-volume components and forms the next main slice. Panel B. Side view of the two main slices in red (containing red and yellow vertices from Panel A). By interpolating these two main slices the blue sub-slices are generated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

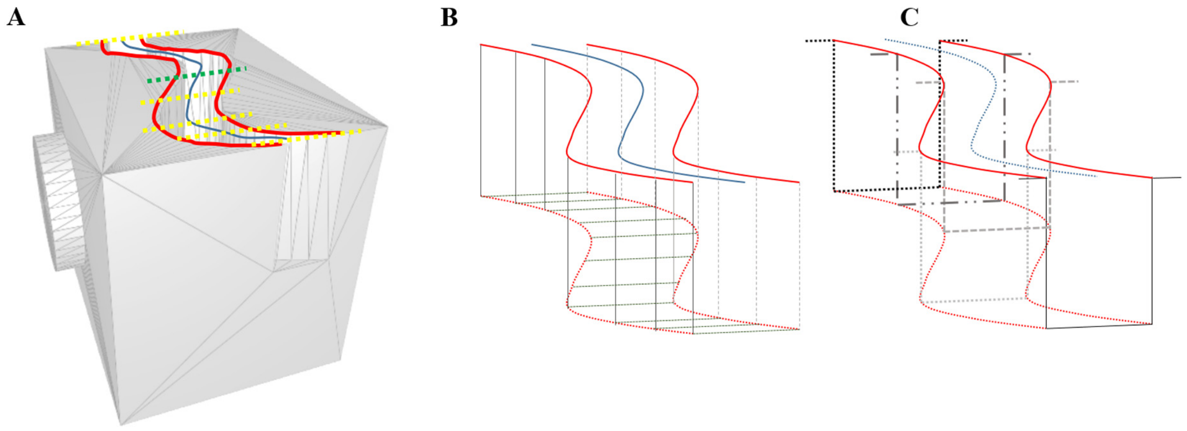

Fig. A.6. Template generation in 2D from 3D model: Panel A: Singulated sub-region borders are marked in red. The path descriptor can be seen in blue dashed line and emulated usable scanner projection lines are depicted with yellow dashes while the projection line is marked with green dashes. Panel B: Singulated, intrusive sub-volume with red boundary lines, blue path descriptor and interconnected borders in gray, Panel C: Hypothetical 2D templates along projection lines in multiple shades of black. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table A.1
Summary of computational time.

Case studies	Average time for feature plane vertices detection (ms)	Average time for feature edges detection (s)	Average time for extended projection line calculation (ms)	Average time for sub-volume calculation (s)	Average time for template generation (ms)
Case 1	4.44 ± 0.08	2.18 ± 0.8	3.35 ± 0.99	8.88 ± 7.17	87.54 ± 36.18
Case 2	0.83 ± 0.05	1.4 ± 0.48	3.63 ± 1.46	10.46 ± 6.47	49.02 ± 14.18
Case 3	0.83 ± 0.05	1.14 ± 0.1	2.46 ± 0.02	10.85 ± 2.29	44.88 ± 21.45

model. Concerning the accuracy of sub-region detection and sub-volume detection, we obtained a value of 100% for case studies 2 and 3. In contrast, we found reduced values for region and sub-volume extractions related to case 1. This feature can be improved by updating the functionalities the second module, i.e., regional contour and border generation module. As for the 2D template generation, we achieved accuracy values of 80% and 92% for cases 2 and 3. However, for closed loops (case 1), values reduce to 50%. This is due to the predicted scanning projection lines that should be physically usable, and should be above the sub-volume so that any structural variation falls under its projection plane.

5. Conclusions

We present a semi-automated procedure and a visual interface to identify volumetric features on .STL virtual models and the associated virtual operations (i.e., extrusion, cut) for their eventual fabrication or quality inspection. This software was developed for manufacturing industries that use subtractive or non-conventional additive processes, as well as joining procedures such as gluing or welding. As a matter of fact, talking with the experts who use robot programming, online programming is today highly time consuming, especially for complex structures. Therefore, having a generic estimation of robot tool path, as our software does, would improve efficiency in manufacturing industry. In view of this, our software overcomes several limitations of state-of-the-art approaches: we are able to reference mechanical components and identify their fabrication processes from .STL files without commercial softwares (e.g., SolidWorks [34,45] or MATLAB [16,17,21]). Thus, we are to reconstruct the Boolean operations that lead to the final topology, without using as inputs complex CAD files and processors with high computational power. Additionally, our procedure shows high flexibility, being able to identify and reference not only closed concave features, as reported in [21], but also more complex topologies such as semi-open or fully open grooves. Finally, we are able to bridge the gap that still persists for ongoing processes such as building

Table A.2
Summary of primary heap memory usage.

Case studies	Feature plane vertices detection (MB)	Feature edges detection (MB)	Extended projection line calculation (MB)	Sub-volume calculation (MB)	OpenGL visualization (MB)
Case 1	0.672	0.179	0.193	0.07	17.48
Case 2	0.051	0.08	0.048	0.015	17.065
Case 3	0.598	0.005	0.005	0.005	17.452

up of geometric signatures [48], in which an online comparison to reference data is not available until after finishing the product fabrication process [9,11,13].

Future releases of the software will provide an improvement of the edge extraction module to identify a broader range of feature edges not limited to those with high curvature, and the integration of with a Graphical User Interface (GUI) extension (i.e., Qt) to enhance a real-time interaction feature and the development of a plugin-like feature for 3D scanning applications.

Declaration of competing interest

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

Acknowledgments

This work was possible thanks to the fruitful collaboration between Scuola Superiore Sant'Anna, The BioRobotics Institute, and Baker Hughes, a GE Company (BHGE), who have strongly believed in the collaboration between industrial companies and research centers. The authors would also like to thank Mr. Marco Miraglia and Mr. Godfried Jansen Van Vuuren for their support in the study.

Appendix

See Figs. A.1–A.6.
See Tables A.1–A.2.

References

[1] Neto P, Mendes N. Robot Auton Syst 2013;61:896–910.
[2] Baizid K, Ćuković S, Iqbal J, Yousnadj A, Chellali R, Meddahi A, Devedžić G, Ghionea I. Robot Comput Integr Manuf 2016;42:121–34.
[3] Morozov M, Pierce SG, MacLeod CN, Mineo C, Summan R. Meas J Int Meas Confed 2018;122:284–90.
[4] Dorai C, Wang G, Jain AK, Mercer C. IEEE Trans Pattern Anal Mach Intell 1998;20:83–9.
[5] Thompson WB, Owen JC, Germain HJDe St, Stark SR, Henderson TC. IEEE Trans Robot Autom 1999;15:57–66.
[6] Kuş A. Sensors 2009;9:1967–79.
[7] Zhao H, Kruth JP, Van Gestel N, Boeckmans B, Bleys P. Meas J Int Meas Confed 2012;45:1057–66.
[8] Chen W, Xiong W, Cheng J, Gu Y, Li Y. 2018 IEEE/ACIS 17th int. conf. comput. inf. sci.. 2018, p. 246–51.
[9] Phan NDM, Quinsat Y, Lavernhe S, Lartigue C. Int J Adv Manuf Technol 2018;1–15.
[10] Deneke C, Schlosser C, Mehler S, Schüppstuhl T. 7th int. symp. NDT aerosp.. 2015, p. 1–8.
[11] Burghardt A, Kurc K, Szybicki D, Muszyńska M, Szczęch T. Teh Vjesn 2017;24:345–8.
[12] Marani R, Nitti M, Cicirelli G, D'Orazio T, Stella E. Adv Mech Eng 2013;2013.
[13] Wu Q, Lu J, Zou W, Xu D. 2015 IEEE int. conf. mechatronics autom. ICMA 2015. 2015, p. 2284–9.
[14] Milazzo M, Contessi Negrini N, Scialla S, Marelli B, Farè S, Danti S, Buehler MJ. Adv Funct Mater 2019;29:1903055.
[15] Morouço P, Azimi B, Milazzo M, Mokhtari F, Fernandes C, Reis D, Danti S. Appl Sci 2020;10:9143.
[16] Ding D, Shen C, Pan Z, Cuiuri D, Li H, Larkin N, Van Duin S. CAD Comput Aided Des 2016;73:66–75.
[17] Mineo C, Pierce SG, Nicholson PI, Cooper I. Robot Comput Integr Manuf 2016;37:1–12.
[18] Milazzo M, Spezzaneve A, Persichetti A, Tomasi M, Peselli V, Messina A, Gambineri F, Aringhieri G, Roccella S. Int J Adv Manuf Technol 2020;109:385–95.
[19] Brown AC, De Beer D. IEEE AFRICON conf.. 2013.
[20] Ding D, Pan Z, Cuiuri D, Li H, Larkin N. J Clean Prod 2016;133:942–52.
[21] Ding D, Pan Z, Cuiuri D, Li H, Larkin N, Van Duin S. Robot Comput Integr Manuf 2016;37:139–50.
[22] Thuault A, Deveaux E, Béhin P, Cam CAD. Dent Mater 2017;33:477–85.
[23] Zheng H, Cong M, Dong H, Liu Y, Liu D. Int J Adv Manuf Technol 2017;92:3605–14.
[24] Nagata F, Okada Y, Sakamoto T, Kusano T, Habib MK, Watanabe K. IOP conf. ser. earth environ. sci., Vol. 69. 2017, 012115.
[25] Jin Y, He Y, Fu G, Zhang A, Du J. Robot Comput Integr Manuf 2017;48:132–44.
[26] Manogharan GP, Wysk R, Harrysson OLA. Int J Comput Integr Manuf 2016;29:473–88.
[27] Toth T, Rajtukova V, Zivcak J. CINTI 2013-14th IEEE int. symp. comput. intell. informatics, Proc. IEEE. 2013, p. 79–82.
[28] Wongwaen N, Sinthanayothin C. ICEIE 2010-2010 int. conf. electron. inf. eng. proc. 1. 2010, p. V1–277–V1–280.
[29] Rebaioli L, Magnoni P, Fassi I, Pedrocchi N, Molinari Tosatti L. Robot Comput Integr Manuf 2019;55:55–64.
[30] Nagata F, Takeshita K, Horie N. 2016 IEEE int. symp. robot. intell. sensors. 2016, p. 86–91.
[31] Xu J, Hou W, Sun Y, Lee YS. Robot Comput Integr Manuf 2018;49:1–12.
[32] Coupek D, Friedrich J, Battran D, Riedel O. Procedia CIRP 2018;67:221–6.
[33] Mahr A, Mayr A, Jung T, Franke J. Procedia Manuf 2019;38:866–75.
[34] Hasan B, Wikander J. Adv. dr. conf. comput. electr. ind. syst. DoCEIS 2017. Cham: Springer; 2017, p. 144–53.
[35] Chen W, Du J, Xiong W, Wang Y, Chia S, Liu B, Cheng J, Gu Y. IEEE Trans Autom Sci Eng 2018;15:251–63.
[36] Chang D, Son D, Lee J, Lee D, Kim TW, Lee KY, Kim J. Robot Comput Integr Manuf 2012;28:1–13.
[37] Chen X, Dharmawan AG, Foong S, Soh GS. Robot Comput Integr Manuf 2018;50:242–55.
[38] Yi Y, Yan Y, Liu X, Ni Z, Feng J, Liu J. J Manuf Syst 2020.
[39] Liu M, Fang S, Dong H, Xu C. J Manuf Syst 2020.
[40] Cai Y, Wang Y, Burnett M. J Manuf Syst 2020;56:598–604.
[41] Andulkar MV, Chiddarwar SS, Marathe AS. J Manuf Syst 2015;37:201–16.
[42] Kah P, Shrestha M, Hiltunen E, Martikainen J. Int J Mech Mater Eng 2015;10:13.
[43] Sharifzadeh S, Biro I, Lohse N, Kinnell P. Mechatronics 2018;51:59–74.
[44] Liu Y, Zhao W, Sun R, Yue X. J Manuf Syst 2020;56:84–92.
[45] Swain AK, Sen D, Gurumoorthy B. Robot Comput Integr Manuf 2014;30:527–45.
[46] Nazir A, Azhar A, Nazir U, Liu Y-F, Qureshi WS, Chen J-E, Alanazi E. J Manuf Syst 2020.
[47] Choi SH, Kwok KT. Rapid Prototyp J 2002;8:161–79.
[48] Li Z, Liu X, Wen S, He P, Zhong K, Wei Q, Shi Y, Liu S. Sensors (Switz) 2018;18.