Digital Manufacturing Technology
https://ojs.wiserpub.com/index.php/DMT/

Research Article

Feature Segmentation of Multifeatured Freeform Geometries for Incremental Sheet Forming

Amita Sahu, Aniket Nagargoje*, Puneet Tandon

deLOGIC Lab, Mechanical Engineering Discipline, Indian Institute of Information Technology Design and Manufacturing, Jabalpur, India
E-mail: aniket@iiitdmj.ac.in

Received: 25 August 2022; **Revised:** 11 October 2022; **Accepted:** 18 October 2022

Abstract: Incremental sheet forming (ISF) of sheets is a flexible forming process that forms a geometry from a clamped sheet using local deformation of the sheet by moving a forming tool along a predefined path. Many factors are responsible for the accessibility and accuracy of forming in the ISF; some of them are forming force, sheet thickness, clamping force, working temperature, and toolpath. One of the important factors which affect the ISF process is the toolpath. However, conventional toolpath generation strategies are designed for the machining process and are not suitable for forming multifeature parts in the case of ISF. A feature segmentation method, which would segregate geometry into multiple features and generate the optimum toolpath with the correct forming sequence, is required to successfully form multifeatured parts through ISF. This paper presents a simple, but an innovative method to segment a component into multiple geometrical features. The proposed method uses the internal characteristics of the STL part file and the number of sliced contours on the parts for feature segmentation.

***Keywords*:** incremental sheet forming, multifeatured parts, geometrical features, feature segmentation, STL model

1. Introduction

Several developments have been done to address the problem of geometry segmentation in various fields. Attene et al. [1] and Agathos et al. [2] reviewed various methods of CAD mesh segmentations developed for various applications of computer graphics. However, these methodologies cannot be used directly for the application in incremental sheet forming (ISF). Thus, a problem-specific methodology needs to be developed.

Recently, ISF has become a cost-effective method for producing quickly tailored parts and sheet metal components with complex geometries in small batches. The ISF process uses a forming tool to locally deform the clamped sheet by moving the tool in a predefined path to get the desired shape of the sheet without using geometry specific die for the product. The ISF has two popular variants, Single Point Incremental Forming (SPIF) and Double-Sided Incremental Forming (DSIF) [3]. Jeswiet et al. [4] described the steps involved in the complete SPIF process, while Meier et al. [5] and Malhotra et al. [6] investigated the importance of using two tools in the ISF.

Toolpath in the ISF process affects many factors like surface finish, geometrical accuracy, forming time, etc. Therefore, toolpath generation is one of the important steps in the ISF process. Many toolpath strategies are investigated by different researchers (Loney et al. [7], Skjoedt et al. [8], Zhu et al. [9], Attanasio et al. [10]) to enhance the surface finish and geometrical accuracy. Currently, toolpath strategies for the surface machining process are being used for the ISF process, which does not result in optimum solutions for the multifeatured parts.

Lu et al. [11] introduced feature-based toolpath generation for SPIF, where equipotential lines based on boundary edges, irrespective of Z height, have been generated and used for toolpath generation. Lingam et al. [12] developed automatic feature recognition and toolpath strategies for DSIF where feature separation and recognition steps were performed using silhouette curves on the surfaces, followed by sequencing of features and toolpath generation. Ndip-Agbor et al. [13] developed a new toolpath strategy for DSIF to generate a toolpath for multifeatured parts, where different features were separated by a contour relationship map, and the forming strategy was developed using a hierarchical tree structure. Zhu et al. [14] used a feature recognition technique for the adjustment of sheet metal posture during ISF. Nagargoje et al. [15-17] developed a machine learning-based geometrical feature extraction approach for ISF.

The current toolpath generation strategies of the ISF process have certain limitations and drawbacks. Some of these limitations are high forming time; difficulties in continuous spiral toolpath generation, when multiple features are present on the same horizontal plane; complicated feature separation methods for feature-based toolpath generation, etc.

Conventional toolpath strategies slice the complete geometry from one end to the other irrespective of the number of features available in each slice. The toolpaths developed using conventional toolpath strategies may not be used to form the multifeatured geometries using the ISF process. This may lead to the collision of the tool on the walls of the formed component if the contours on a particular slice are more than one. Thus, there is a need for geometrical feature extraction to develop the feature-based toolpaths to successfully form multifeatured geometries.

In a feature-based toolpath, the tool moves from one feature to another. Therefore, the features on a part need to be separated so that the tool can form one feature at a time, without breaking tool-sheet contact. Unlike conventional strategies, in feature-wise toolpaths, different features of the geometry are first separated into multiple subparts. This process is called the segmentation of geometry. The objective of the present work is to develop a simple and effective geometric feature separation methodology that can segregate features of a multifeature part.

The STL file format has been employed in this study since it is simple to grasp and approximate the geometrical characteristics of the component in terms of triangles, making computation easier.

2. Methodology

A feature is defined as a segment of the geometry for multifeatured freeform geometries. If sliced, the toolpath developed using the same can be used to incrementally form the segment without losing the tool-sheet contact. To separate features in a multifeatured part, boundaries, which separate the feature, need to be identified. Boundaries should be detected in such a manner that they can be used to define a feature. Therefore, feature boundary detection is the first step for feature separation. In this work, the feature segmentation process consists of two main steps:

1) Feature boundary detection;
2) Extraction of feature facets between the boundaries.

These steps are described briefly in Sections 2.1 and 2.2, respectively.

2.1 *Feature boundary detection using STL model*

In this work, features have been separated using two different methods. The first method is based on the internal properties of the STL file, where internal characteristics have been used to detect feature edges, and the second method is based on the number of contours on each consecutive slice. Both methods are described below in detail.

2.1.1 *STL property-based feature boundary detection*

The STL model has many internal characteristics, like the size of triangles, a sudden change of angles of facet normal, and unpaired edges in the surface boundary. These characteristics can be used to detect edges on the boundary and detected edges are further used to form the feature boundary.

In the STL models, surfaces, whether planer surfaces of the sheet metal or the curved parts of the geometries like fillets, are approximated using triangles. Each edge of a facet in the STL file has a pair edge except boundary edges, and every edge shares exactly two facets. For every edge, one can find facets that share a common edge. The angle between the outward normal vectors of these facets is called the dihedral angle (DA) for each pair of facets. This angle is very useful to detect feature edges. The DA between the facets has been calculated with the help of equation 1,

$$\mathrm{DA} = \frac{n_\mathrm{i} \cdot n_\mathrm{j}}{|n_\mathrm{i}||n_\mathrm{j}|} \tag{1}$$

Here, n_i = outward normal vector of the facet i, and n_j = outward normal vector of facet j, where i and j are the two neighbouring facets.

The features of any component change either abruptly or smoothly. When a feature changes abruptly, it will be called a sharp feature change. On the other hand, when a feature changes smoothly, it is called a smooth feature change. The feature boundary edges for sharp feature changes are called sharp feature edges, while the feature boundary edges for smooth feature changes are called non-sharp feature edges. Normally, DA between facets is less and it changes very smoothly and gradually, however, when there is a sharp feature change, then the DA changes significantly and is higher than the DA of their neighbour facet pairs. Most of the freeform geometry used in sheet metal forming work has DA less than 5°. Therefore, for sharp feature boundaries, we started detecting feature boundary edges by varying the limit from 5° to 15° and found that most of the sharp boundary edges are detected by using a limit of 10° to 12°. Therefore, 15° has been chosen as a safe limit to detect all sharp feature boundary edges.

In the case of smooth feature change, DA does not change by a significant amount. In such cases, the area ratio (AR) of facets is used to detect the feature boundary. AR is the ratio of the area of facets sharing a common edge and it changes significantly when the feature changes smoothly. If AR is higher than a given limit, then the edge can be detected as a non-sharp feature edge. Based on these characteristics, an edge is called a feature edge, if DA or AR corresponding to that edge is beyond a given limit. For the boundary edge, all the edges are searched for their pair edge, and if the pair edge is not there, then that edge is considered the boundary edge of geometry, and the contour made from these boundary edges will be a feature boundary of the outermost feature.

For non-sharp feature edge, due to high AR in the portion where feature changes, AR of the facets sharing a common edge has been used. Selection of an edge as a feature edge is based on Average Area Ratio (AAR), because the range of area ratio of facets varies as per the shape, size, and type (i.e., revolved, planer, freeform, etc.) of the geometry. Furthermore, the limit of AR is given as per the range of AAR because for smaller AAR, the limit of AR would be smaller, and on the other hand, higher AAR requires a larger AR limit to select an edge as a feature edge. Here, based on the investigations carried out for multiple geometries, the limit of AAR for the geometries having AR less than 1.5 has been taken as 1.7 times the AAR, while for the geometries having AR greater than 1.5, the limit is taken as a square of AAR. For facets, AR and AAR have been calculated using equations 2 and 3, respectively,

$$\mathrm{AR} = \frac{\max(A_1, A_2)}{\min(A_1, A_2)} \tag{2}$$

$$\mathrm{AAR} = \frac{\sum_{\mathrm{j}=1}^{n} \mathrm{AR}}{n} \tag{3}$$

Here, A_i = area of facet i, j = index for facet pair, and n = number of adjacent facet pairs whose area ratio has been calculated.

All sharp, non-sharp, and boundary edges are used to form contours, which in turn form feature boundaries. To make contours from detected edges, the Head-to-Tail approach has been used. In this approach, the contour starts from an edge and the second coordinate of the edge is matched with any of the coordinates of other edges and the matched edge will come next on the contour. This process will continue until the contour is closed or all edges are used in contour formation.

A few undesirable boundary edges may be found by the algorithm; these edges must be eliminated from the collection of boundary edges. As demonstrated in Figure 1, edges are deemed undesirable and eliminated if they only serve as a short tail between the contour rather than leading to the closure of the contour.

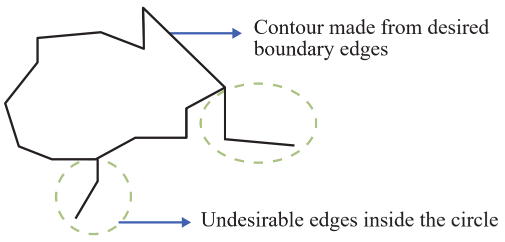
Contour made from desired boundary edges
Undesirable edges inside the circle

Figure 1. Schematic of undesirable boundary edges

Figure 2 describes the algorithm for feature boundary detection methodology based on the properties of the STL file format.

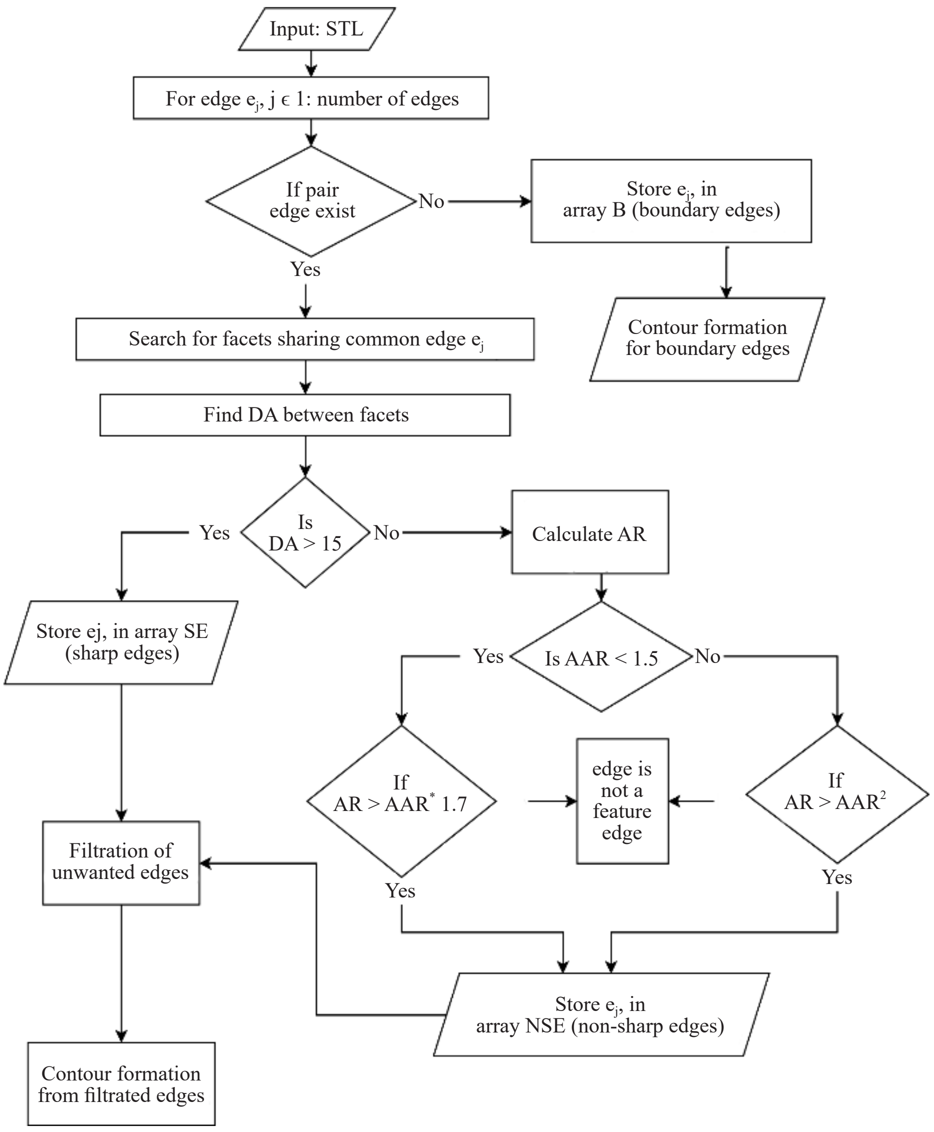
Input: STL
For edge e_j, j ϵ 1: number of edges
If pair edge exist
No
Store e_j, in array B (boundary edges)
Contour formation for boundary edges
Yes
Search for facets sharing common edge e_j
Find DA between facets
Is DA > 15
Yes
No
Calculate AR
Store ej, in array SE (sharp edges)
Is AAR < 1.5
Yes
No
If AR > AAR* 1.7
edge is not a feature edge
If AR > AAR²
Yes
Yes
Filtration of unwanted edges
Store e_j, in array NSE (non-sharp edges)
Contour formation from filtrated edges

Figure 2. Algorithm for feature edge detection

2.1.2 *Number of contour-based feature boundary detection*

If the geometry of any multifeature part is freeform in nature, then the feature changes very smoothly and its boundary edges cannot be detected using AR or DA between the facets, as described in the previous section. For such parts, feature boundaries have been detected using sliced contours on the part. It has been observed that, if a part is sliced through a cutting plane parallel to the base plane, then the number of contours on a slicing plane is an important parameter to detect feature boundary. For every part, there is always a contour of the geometry's boundary, as described in the previous section. If there is no feature boundary other than the geometry's outer boundary, then the part is considered a freeform part, and the method described here can be used to identify the boundaries of a freeform geometry.

In this method, a part is sliced from its minimum Z height to the maximum Z height and the number of contours on each height is stored. When the number of contours changes, then the corresponding contours on a particular Z height are stored as feature boundaries. As, it has been observed that for a single feature, the number of contours remains the same, but as the feature changes from one height to another, then the number of contours on consecutive slices would change as shown in Figure 3. The algorithm for this method is described with the help of Figure 4. To use the number of contour-based feature boundary detection method, the geometry must be laid on the XY plane and the slicing of contour must be in the positive z-direction.

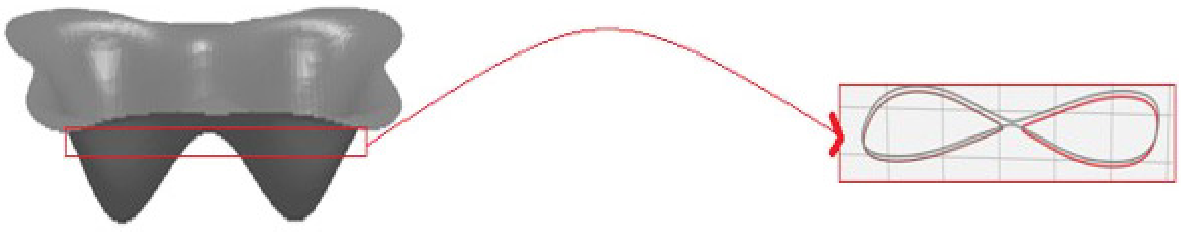
Figure 3. Contours identified on two successive slices when the feature changes

For any geometry, the applicability of the boundary detection method is based on geometry. The part is first checked for feature boundaries using the first method (STL property-based feature boundary detection method) and if there is no feature boundary other than geometry's outer boundary, then it is considered as freeform geometry and checked for feature boundary using the second method (number of contour-based feature boundary detection method). This is described with the help of Figure 5.

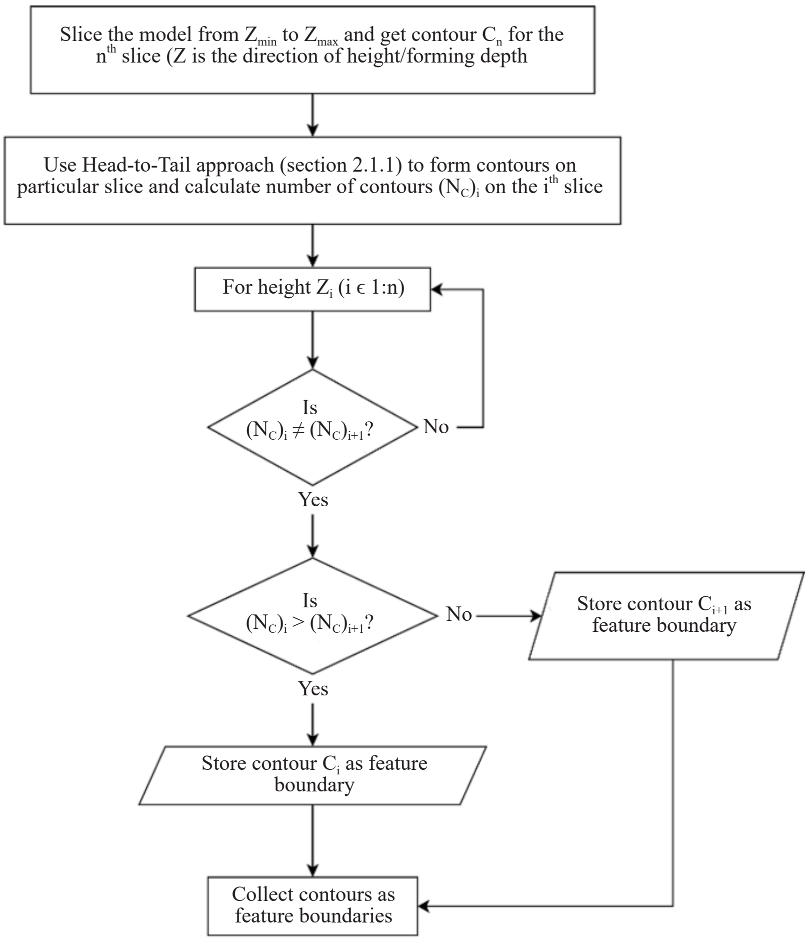
Slice the model from Z_{min} to Z_{max} and get contour C_n for the nth slice (Z is the direction of height/forming depth
Use Head-to-Tail approach (section 2.1.1) to form contours on particular slice and calculate number of contours $(N_C)_i$ on the ith slice
For height Z_i (i ϵ 1:n)
Is $(N_C)_i \neq (N_C)_{i+1}$?
No
Yes
Is $(N_C)_i > (N_C)_{i+1}$?
No
Store contour C_{i+1} as feature boundary
Yes
Store contour C_i as feature boundary
Collect contours as feature boundaries

Figure 4. Feature boundary detection method for freeform surfaces

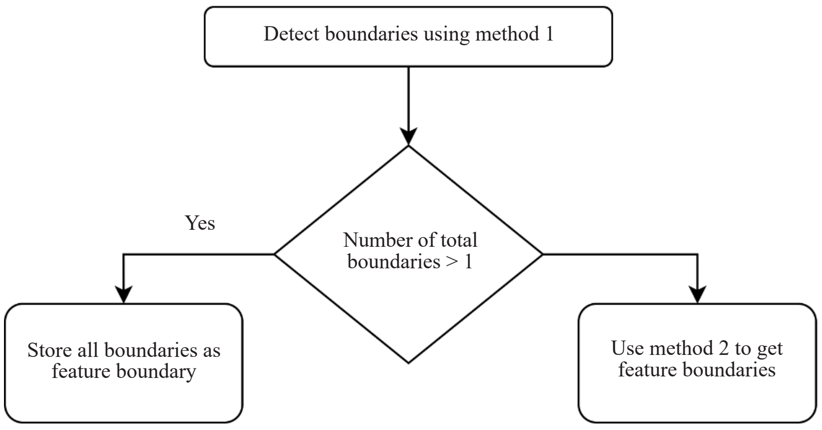
Detect boundaries using method 1
Yes
Number of total boundaries > 1
Store all boundaries as feature boundary
Use method 2 to get feature boundaries

Figure 5. Criterion for selection of method

2.2 *Extraction of feature facets*

Identification of correct feature boundaries is essential for correct feature segmentation as these are further used to get feature surface data. These feature boundaries play a vital role in segmentation. Edges in the contours are used as breakpoints for feature facet data propagation. Feature facets data is extracted with the help of a method similar to the region-growing method for image segmentation [18]. Using the region growing method, the facets inside the boundaries are extracted, and these facets inside the boundaries are collectively called feature data. Out of the extracted facets, a facet is selected randomly as a seed facet and every neighbour facet is added to the region if any of its edges is not a part of the feature boundary edge. This propagation of neighbourhood facets goes on until no neighbourhood facet exists. The algorithm for the region growing method for feature facet extraction is presented with the help of Figure 6. After detecting boundary facets, these facets will be stored in a separate array or matrix. To implement the region-growing method, the original array which stores all facets data will be used again together with the array containing boundary facets.

This region growing method is analogous to the region growing method of image segmentation where a range of similar pixel intensity is grouped for segmentation. In the region-growing method used here, a randomly chosen seed facet will propagate to all neighbour facets having a common edge to that facet, then these all newly added facets will propagate to their neighbour and the chain of propagation will be continued until all the newly added facets will find any boundary facet as neighbour facet, and collectively these facets will be called as feature data of a feature.

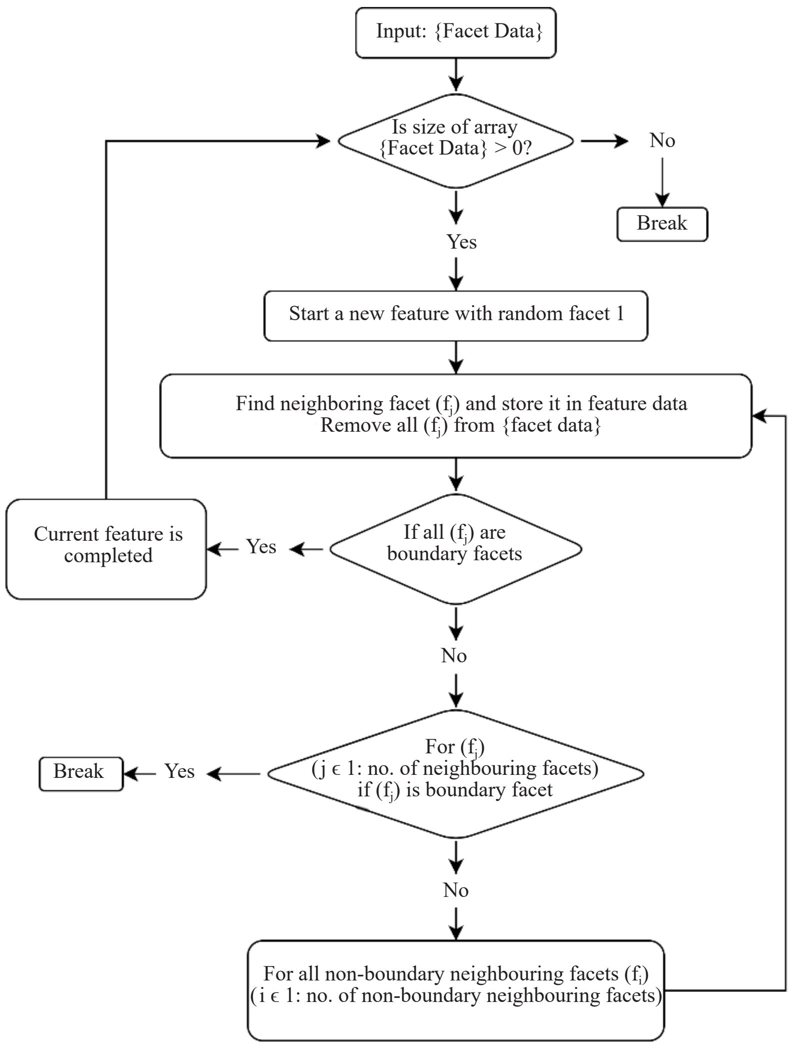
Input: {Facet Data}
Is size of array {Facet Data} > 0?
No
Break
Yes
Start a new feature with random facet 1
Find neighboring facet (f_j) and store it in feature data
Remove all (f_j) from {facet data}
Current feature is completed
Yes
If all (f_j) are boundary facets
No
Break
Yes
For (f_j) (j ϵ 1: no. of neighbouring facets) if (f_j) is boundary facet
No
For all non-boundary neighbouring facets (f_i) (i ϵ 1: no. of non-boundary neighbouring facets)

Figure 6. Region growing methodology

2.3 *Implementation*

To implement the proposed feature segmentation methodology for the components which can be formed through ISF, a Python code has been developed in PYTHON 3 environment using PYCHARM community edition IDE. The component to be incrementally formed is first modelled in the SOLIDWORKS modelling software and the STL file has been generated. The STL model is taken as an input to the developed code. However, many challenges have been faced during its implementation. Some of the challenges include high performance time due to the processing of a large number of facets in the STL format, problems associated with STL file repair before feeding as an input to the developed code, etc. Figure 7 shows the algorithm for the procedure employed for complete feature segregation.

3. Results and discussion

The given methodology has been implemented for some test geometries; the corresponding results are presented in this section. In the algorithm's initial training phase, we used simpler components with only plane features, as illustrated in Figure 8 and Figure 9. Following that, the technique was applied to freeform components.

A non-freeform component having a combination of planer and curved features, as shown in Figure 10 (a, b), has been implemented. The outermost feature of the component, feature 1, as shown in Figure 10 (d), is the planer and other features are curved. The feature boundaries of this component have been detected using both the concepts of DA and AR as described in Section 2.1.1.

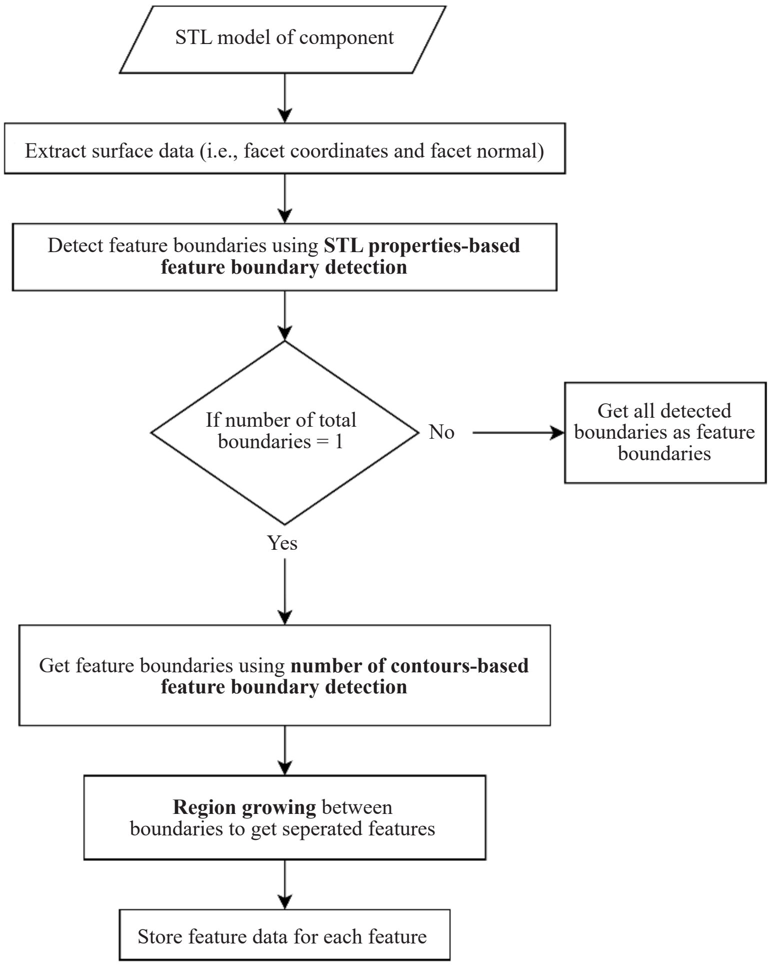
STL model of component
Extract surface data (i.e., facet coordinates and facet normal)
Detect feature boundaries using **STL properties-based feature boundary detection**
If number of total boundaries = 1
No
Get all detected boundaries as feature boundaries
Yes
Get feature boundaries using **number of contours-based feature boundary detection**
Region growing between boundaries to get seperated features
Store feature data for each feature

Figure 7. Algorithm for feature segmentation

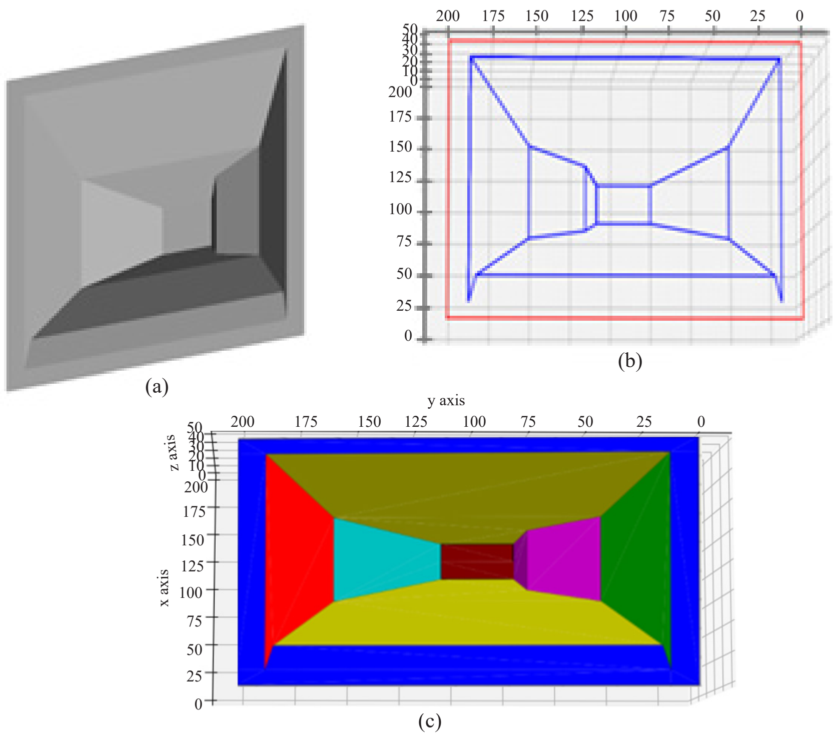

Figure 8. Feature segmentation of a geometry with planer features. (a) CAD model, (b) Component's feature boundaries, (c) Segmented features in different colors

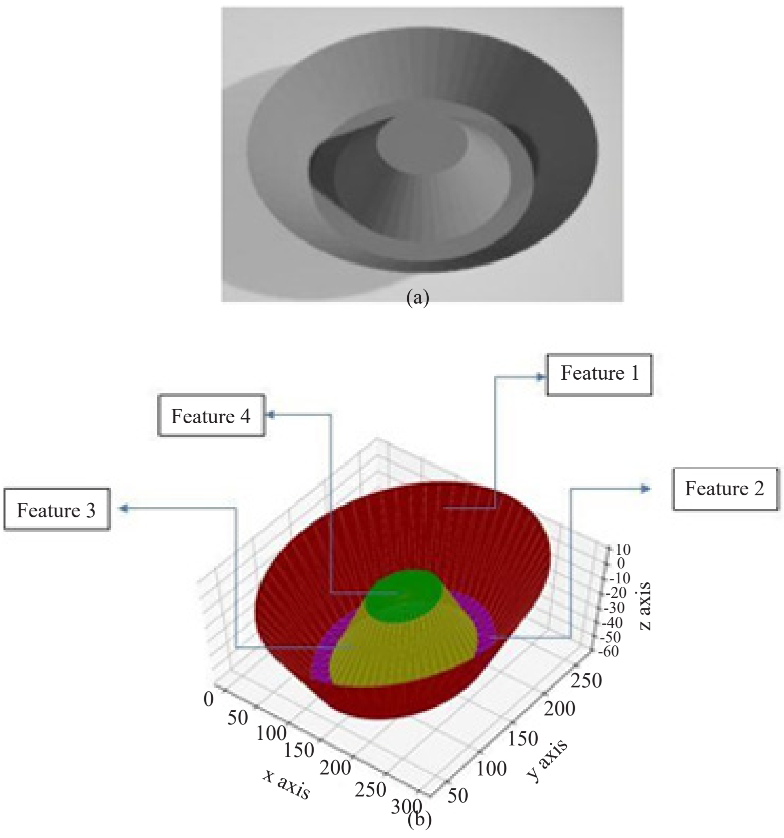

Figure 9. Feature segmentation of a geometry. (a) CAD model, (b) Segmented features in different colors

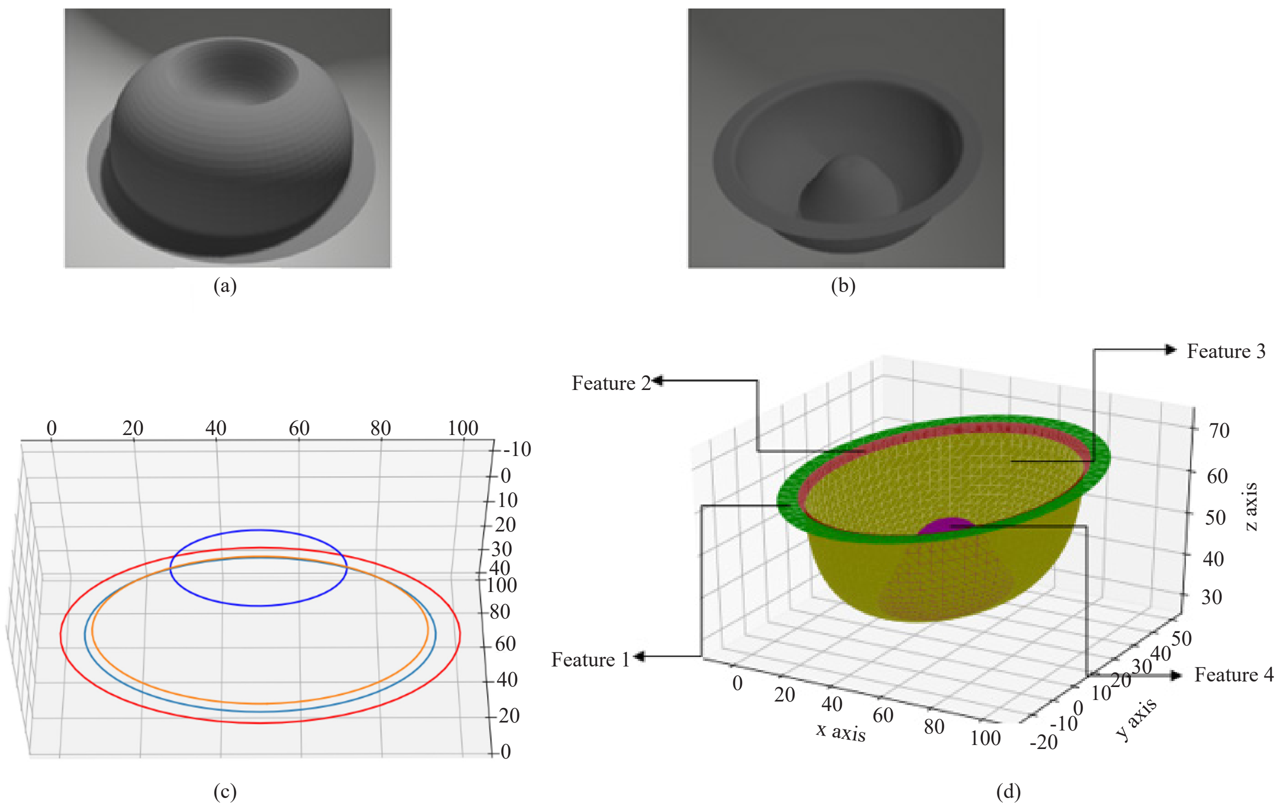

Figure 10. Feature segmentation of a geometry with curved features. (a) Bottom view of the CAD model, (b) Top view of the CAD model, (c) Component's feature boundaries, (d) Segmented features in different colors

A complete freeform part, as shown in Figure 11 (a, b), has also been considered to validate the developed algorithms. There are many curvatures in this component, and all of these are very smooth. When the STL property-based feature boundary detection method is used for this component, no useful boundary is detected as the features are very smooth. In this case, contour-based feature boundary detection would be useful as the base plane of the part to be formed is parallel to the horizontal plane. The results are shown in Figure 11.

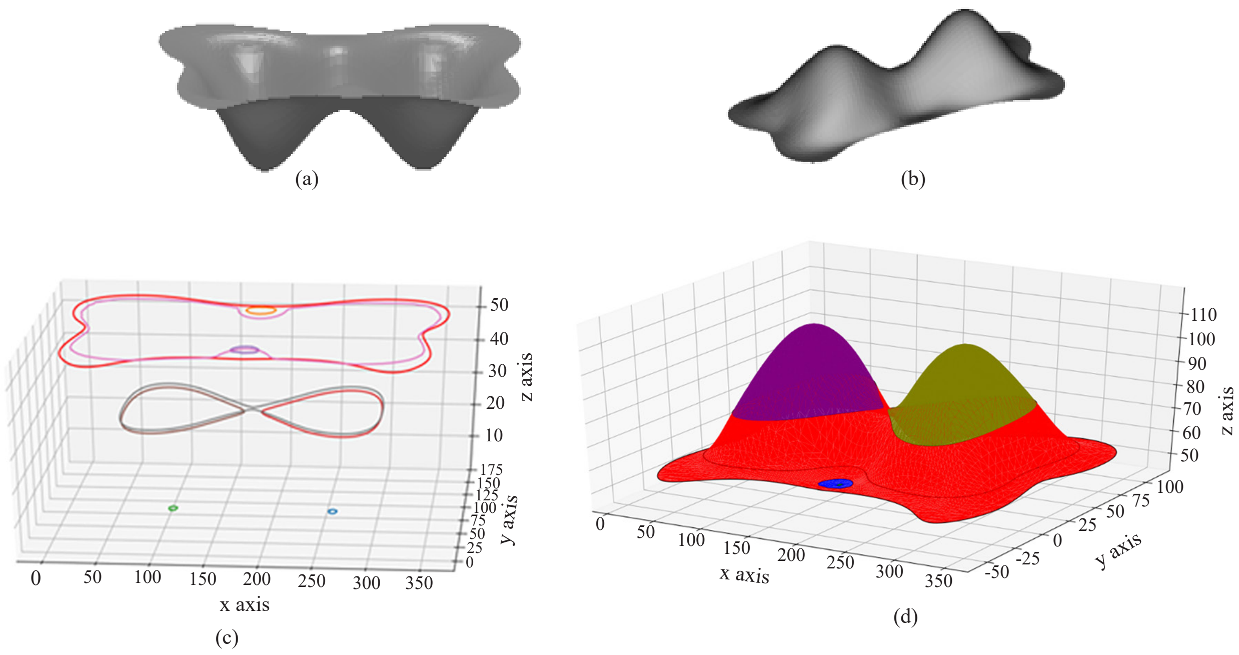

Figure 11. Feature segmentation of a freeform geometry. (a) Bottom view of the CAD model, (b) Top view of the CAD model, (c) Feature boundaries of component, (d) Segmented features of the component

4. Conclusions and future scope

In this work, a novel feature segmentation methodology has been developed and implemented for various types of multifeature parts. The methodology takes the STL file of the CAD model to be formed as an input and delivers segmented features with surface data as output. The following conclusion can be drawn from this work:

- Internal properties of the STL file of a part like DA, uneven distribution of facets, and different shapes of facets are very useful for getting useful information about the features on a surface. This information can be used for the detection of feature boundaries.
- STL property-based feature boundary detection method is useful and provides accurate results for the parts where the feature changes abruptly and where geometrical properties like facet normal and area of nearby facets vary by a considerable amount. On the other hand, the feature boundary detection method based on the number of contours works accurately for the parts having a smooth change in the feature. In this case, the base plane of the component should be horizontal. However, if it is not horizontal, one can rotate the component by an appropriate angle to make its base horizontal, so that on horizontal slicing, it would give a closed contour.
- The proposed methodology that combines STL-based feature boundary detection and contour-based feature boundary detection works effectively for multifeatured parts.
- Separate features can be further used to generate a feature-wise toolpath for ISF. The methodology can also be adapted for feature-based additive manufacturing and machining processes.
- The region-growing method for feature data extraction is a simple and fast method to record facets between feature boundaries.
- To develop the feature-based toolpaths, the extracted features, rather than the complete geometry, can be sliced one by one in a certain sequence to avoid the collision of the tool with the wall of the formed component. Furthermore, the features can be divided based on the features growing towards the negative z-direction and features growing towards the positive z-direction to develop double-sided feature-based toolpaths. The feature sequencing algorithms sequence the features to avoid accidents and decrease the forming time for various variants of the ISF process.

Funding

This research was financially supported by Impacting Research, Innovation and Technology (IMPRINT) INDIA through a project of the Government of India (Project Number: 5506). The authors wish to acknowledge the funding agencies.

Conflict of interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

[1] Attene M, Patané G, Katz S, Mortara M, Patane G, Spagnuolo M, Tal A. Mesh segmentation - A comparative study. In: *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*. New Jersey: IEEE; 2006. p.7. https://doi.org/10.1109/SMI.2006.24

[2] Agathos A, Pratikakis I, Perantonis S, Sapidis N, Azariadis P. 3D mesh segmentation methodologies for CAD applications. *Computer-Aided Design and Applications*. 2007; 4(6): 827-841. https://doi.org/10.1080/16864360.2007.10738515

[3] Nagargoje A, Kankar PK, Jain PK, Tandon P. Application of artificial intelligence techniques in incremental forming: A state-of-the-art review. *Journal of Intelligent Manufacturing*. 2021. https://doi.org/10.1007/s10845-021-01868-y

[4] Jeswiet J, Micari F, Hirt G, Bramley A, Duflou J, Allwood J. Asymmetric single point incremental forming of sheet metal. *CIRP Annals*. 2005; 54(2): 88-114. https://doi.org/10.1016/s0007-8506(07)60021-3

[5] Meier H, Magnus C, Smukala V. Impact of superimposed pressure on dieless incremental sheet metal forming with two moving tools. *CIRP Annals*. 2011; 60(1): 327-330. https://doi.org/10.1016/J.CIRP.2011.03.134

[6] Malhotra R, Cao J, Ren F, Kiridena V, Cia C, Reddy NV. Improvement of geometric accuracy in incremental forming by using a squeezing toolpath strategy with two forming tools. *Journal of Manufacturing Science and Engineering*. 2011; 133(6): 061019. https://doi.org/10.1115/1.4005179

[7] Loney GC, Ozsoy TM. NC machining of free form surfaces. *Computer-Aided Design*. 1987; 19(2): 85-90. https://doi.org/10.1016/S0010-4485(87)80050-7

[8] Skjoedt M, Hancock MH, Bay N. Creating helical tool paths for single point incremental forming. *Key Engineering Materials*. 2007; 344: 583-590. https://doi.org/10.4028/www.scientific.net/KEM.344.583

[9] Zhu H, Liu Z, Fu J. Spiral tool-path generation with constant scallop height for sheet metal CNC incremental forming. *The International Journal of Advanced Manufacturing Technology*. 2011; 54: 911-919. https://doi.org/10.1007/s00170-010-2996-5

[10] Attanasio A, Ceretti E, Giardini C. Optimization of tool path in two points incremental forming. *Journal of Materials Processing Technology*. 2006; 177(1-3): 409-412. https://doi.org/10.1016/j.jmatprotec.2006.04.047

[11] Lu B, Chen J, Ou H, Cao J. Feature-based tool path generation approach for incremental sheet forming process. *Journal of Materials Processing Technology*. 2013; 213(7): 1221-1233. https://doi.org/10.1016/j.jmatprotec.2013.01.023

[12] Lingam R, Prakash O, Belk JH, Reddy NV. Automatic feature recognition and tool path strategies for enhancing accuracy in double sided incremental forming. *The International Journal of Advanced Manufacturing Technology*. 2017; 88: 1639-1655. https://doi.org/10.1007/s00170-016-8880-1

[13] Ndip-Agbor E, Ehmann K, Cao J. Automated flexible forming strategy for geometries with multiple features in double-sided incremental forming. *Journal of Manufacturing Science and Engineering*. 2018; 140(3): 031004. https://doi.org/10.1115/1.4038511

[14] Zhu H, Li J. Research on the CNC incremental forming based on multidirectional real-time adjustment of the sheet posture. *The International Journal of Advanced Manufacturing Technology*. 2020; 110: 1339-1350. https://doi.org/10.1007/s00170-020-05918-2

[15] Nagargoje A, Kankar PK, Jain PK, Tandon P. Development of the geometrical feature extraction tool using DBSCAN clustering for toolpath generation in incremental forming. Research Square [Preprint] 2021. https://doi.org/10.21203/RS.3.RS-340927/V1

[16] Nagargoje A, Kankar PK, Jain PK, Tandon P. Performance evaluation of the data clustering techniques and cluster validity indices for efficient toolpath development for incremental sheet forming. *Journal of Computing and Information Science in Engineering*. 2021; 21(3): 031001. https://doi.org/10.1115/1.4048914

[17] Nagargoje A, Kankar PK, Jain PK, Tandon P. Comparison of clustering techniques for feature-based toolpath generation in dieless manufacturing. In: *ASME 2021 International Mechanical Engineering Congress and Exposition. Volume 2B: Advanced Manufacturing*. ASME; 2021. https://doi.org/10.1115/IMECE2021-70255

[18] Adams R, Bischof L. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1994; 16(6): 641-647. https://doi.org/10.1109/34.295913