

Web-Based STEP AP242 Boss and Pocket Feature Recognition for Automated CNC G-Code Generation

Amirulihsan Mohamad

Zammeri Abd Rahman

zammeri@unisza.edu.my

Universiti Sultan Zainal Abidin - Kampus Gong Badak <https://orcid.org/0000-0001-5582-8823>

Saiful Bahri Mohamed

Roslan Awang

Muhammad Abdulrahim Rabbani Md Md Sharizam

Tengku Mohd Sharir Tengku Sulaiman

Research Article

Keywords: G-Code, STEP AP242, Boss Feature Recognition, Pocket Feature Recognition, Geometric Data Extraction, Web-Based System

Posted Date: April 13th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9253485/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Version of Record: A version of this preprint was published at The International Journal of Advanced Manufacturing Technology on May 27th, 2026. See the published version at <https://doi.org/10.1007/s00170-026-18381-2>.

Web-Based STEP AP242 Boss and Pocket Feature Recognition for Automated CNC G-Code Generation.

Amirulihsan Mohamad², Zammeri Abd Rahman^{1,*}, Saiful Bahri Mohamed¹, Roslan Awang¹, Muhammad Abdulrahim Rabbani Md Sharizam², Tengku Mohd Sharir Tengku Sulaiman².

¹ Design Optimisation and Manufacturing Research Group, Faculty of Innovative Design and Technology, Universiti Sultan Zainal Abidin, Gong Badak Campus, 21300 Kuala Nerus, Terengganu, Malaysia

² Faculty of Innovative Design and Technology, Universiti Sultan Zainal Abidin, Gong Badak Campus, 21300 Kuala Nerus, Terengganu, Malaysia

*Corresponding author: zammeri@unisza.edu.my

Abstract

This paper presents a web-based island–Pocket Feature Recognition System (IPFRS) that recognises islands and blind pockets from STEP AP242 files and automatically generates CNC milling G-code. The system applies geometric data extraction to parse STEP entities and reconstruct feature boundaries using key geometric definitions such as CARTESIAN_POINT, LINE, CIRCLE, and PLANE. Feature classification is performed by comparing Z-level relationships between reference planes and extracted boundary points to distinguish raised islands (boss) from recessed blind pockets. Based on the recognised features, IPFRS generates toolpaths and converts geometric primitives into standard CNC commands, producing linear and circular motions (e.g., G01, G02/G03) while incorporating user-defined machining parameters such as tool diameter and overlap. The software is implemented using a web architecture (PHP, HTML, and JavaScript) with database support, enabling file upload, processing, code visualisation, and direct G-code download. Validation through CNC simulation using different endmill diameters (5 mm and 10 mm) confirms correct machining sequence, valid syntax, and collision-free toolpaths. The proposed system reduces manual feature identification and programming effort, providing an accessible workflow for consistent G-code generation directly from STEP AP242 models.

Keyword:

G-Code, STEP AP242, Boss Feature Recognition, Pocket Feature Recognition, Geometric Data Extraction, Web-Based System

1 Introduction

In CNC machining, a persistent challenge is the accurate identification and handling of embedded machining features in 3D CAD models, particularly as models become geometrically complex [1]. Among the most common yet demanding features are islands and blind pockets, which require careful process planning to prevent tool collisions, incomplete material removal, and dimensional inaccuracies [2]. When these features are identified and programmed manually, the workflow can become time-consuming and error-prone, often leading to inefficiencies, rework, and increased risk of part failure. Consequently, automating feature recognition and converting recognised features into reliable machining instructions (G-code) is essential for improving productivity and reducing machining errors.

To support this automation, recent studies highlight the importance of Geometric Data Extraction (GDE) as a practical approach for interpreting STEP data and reconstructing usable geometric information for downstream manufacturing applications [3-5]. By systematically extracting and analysing geometric entities, GDE enables consistent identification of relevant feature boundaries and parameters, thereby improving recognition accuracy and reducing ambiguity in feature interpretation [6]. This is especially important when the final output is G-code, where small interpretation errors may propagate into incorrect tool motions, poor surface quality, or machining failure. Therefore, a STEP-based pipeline that integrates feature recognition with automatic code generation can reduce human dependency and enhance the reliability of CNC programming.

In response to these needs, this research develops a web-based Island Pocket Feature Recognition System (IPFRS) to recognise islands and blind pockets from STEP AP242 files and generate the corresponding G-code for CNC machining. The system is implemented using PHP, HTML, and JavaScript, providing a browser-accessible interface that enables users to upload STEP files and obtain machining code with minimal setup and technical overhead [7]. A web-based deployment also supports usability and accessibility for practitioners, while allowing smoother adoption within existing production environments and digital workflows [8, 9]. By reducing manual interpretation and repetitive CAM programming tasks, the proposed system aims to shorten process planning time and improve the consistency of the generated machining instructions, supporting more efficient and accurate machining operations [10].

Overall, IPFRS addresses an important gap in current CNC workflow automation by providing an integrated solution for feature recognition and G-code generation, specifically for islands and blind pockets [11-13]. Automating these tasks can streamline machining preparation, reduce programming errors, and improve operational efficiency for machinists and engineers [14]. By leveraging web technologies, the proposed approach offers a scalable, accessible, and implementation-friendly pathway toward more integrated STEP-enabled CNC manufacturing.

2 Methodology

2.1. Software Architecture

Software architecture is the foundation of software. It demonstrates the connection and relationship between the blocks, ensuring that the software functions as designed. Figure 1 below shows the

software architecture for this system. The software is designed using two separates but connected architectures, as shown in Figure 1. It separates the front end and back end, corresponding to their function. The front end is developed using HTML, CSS, and JavaScript. It presents an intuitive user interface that allows users to upload STEP AP242 files into the system, generate G-code, visualise the generated G-code, and then download the completed G-code.

The back end, implemented in PHP, processes uploaded files and executes the GDE algorithms. The system also manages G-code generation. The application follows a client-server model. The client (user interface) communicates with the server (back-end processing) using standard queries to access the database. This architecture not only enhances the application's scalability but also enables easy integration with other web services and tools in the future.

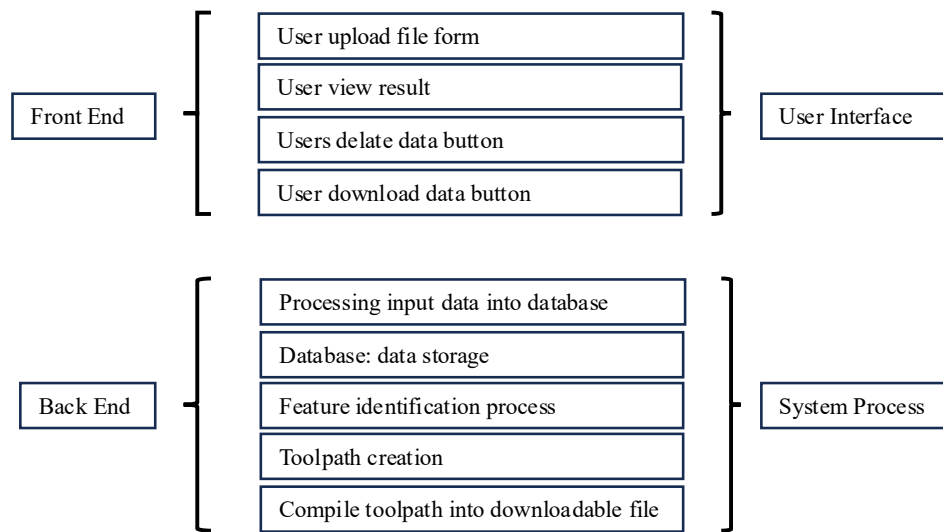


Figure 1. Front End and Back End Software Architecture

2.2. Geometric Data Extraction (GDE)

Geometric Data Extraction (GDE) is a critical process within the software that parses STEP AP242 files to extract relevant geometric information [15]. This process is implemented through a combination of libraries and custom algorithms specifically designed to interpret the hierarchical structure of STEP files. The GDE process is essential for ensuring that the geometric data is accurately captured for subsequent analysis and machining operations.

The first step in GDE is STEP file analysis, which involves reading the STEP AP242 file format. This format is structured to represent various geometric entities. It includes points, lines, surfaces, and circles. The software employs a STEP file analyser that navigates the intricate structure of the file, enabling it to extract the necessary geometric data effectively. This analysis step is fundamental to the overall GDE process, as it lays the groundwork for further data manipulation [16].

Once the geometric data is extracted, it undergoes data representation, which converts it into a suitable format for further processing. This typically involves transforming the data into a format that can be manipulated, such as a graph or a set of mathematical representations of the geometric features. Additionally, the GDE process includes feature identification, which entails analysing the geometric relationships between different entities to determine which features correspond to islands and blind pockets. This comprehensive approach ensures the software accurately identifies and classifies relevant geometric features, facilitating efficient machining operations. For this case, the Cartesian point is being extracted. The process is illustrated in Figure 2 below.

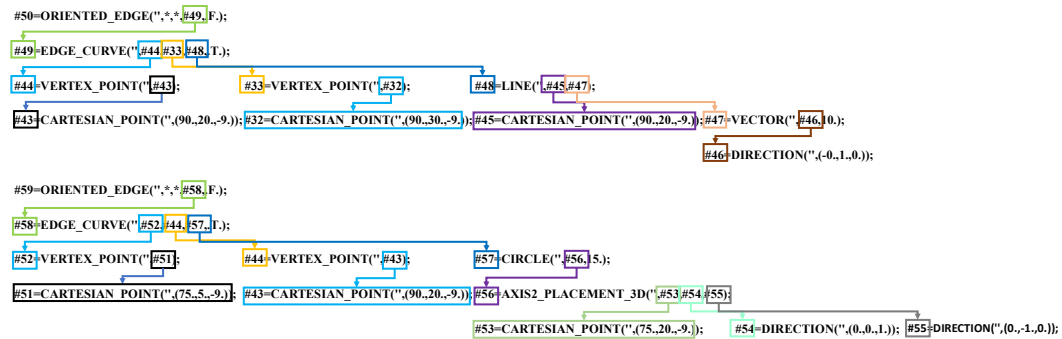


Figure 2. Examples of geometric data extraction

Figure 2 illustrates an example of geometric data extraction. The ORIENTED_EDGE entity originates from the CLOSED_SHELL definition. EDGE_CURVE (#49) defines a line segment (LINE, #48) between two points (VERTEX_POINT, #44 and #33), which are in turn determined by their corresponding CARTESIAN_POINT values. This segment represents a basic edge. It is used to define part boundaries. The curved features are described using the CIRCLE and AXIS2_PLACEMENT_3D entities. It's enabled the recognition of circular or curved regions within the object.

In the given example, the curve's centre is located at coordinates (75, 20, -9). It is 15 mm in radius. The system analyses and reconstructs these geometric entities to create an accurate representation of the part's structure. The extracted data is then processed by the island and pocket recognition algorithms, which classify the features into their respective types. This is followed by the automatic generation of G-code based on the recognised features.

2.3. Island and Pocket Recognition Algorithm

This process begins with the user entering the STEP file into the system via the provided interface. The system automatically processes the STEP file and populates the database with related data. Next, the system will run the geometric analysis. The software evaluates the data to identify potential islands and blind pockets. During this phase, various geometric feature properties are calculated. They include depth, the minimum and maximum dimensions along the X and Y axes, and the distance between the object and the edge of the workpiece

During this process, the Z-axis value is the focus. The z-axis value from the CARTESIAN_POINT and the PLANE is compared. The blind pocket is determined when the z-axis value is smaller than the Z-axis value at the plane, and the value is an island when it's bigger than the Z-axis PLANE. These calculations are essential for accurately determining the characteristics of the feature present in the dataset. This recognition methodology ensures the software can correctly recognise between islands and blind pockets. It is an essential capability to generate accurate, efficient machining commands. By utilising geometric analysis and classification techniques, the system significantly improves the accuracy of toolpath generation, thereby enhancing productivity and quality in manufacturing operations.

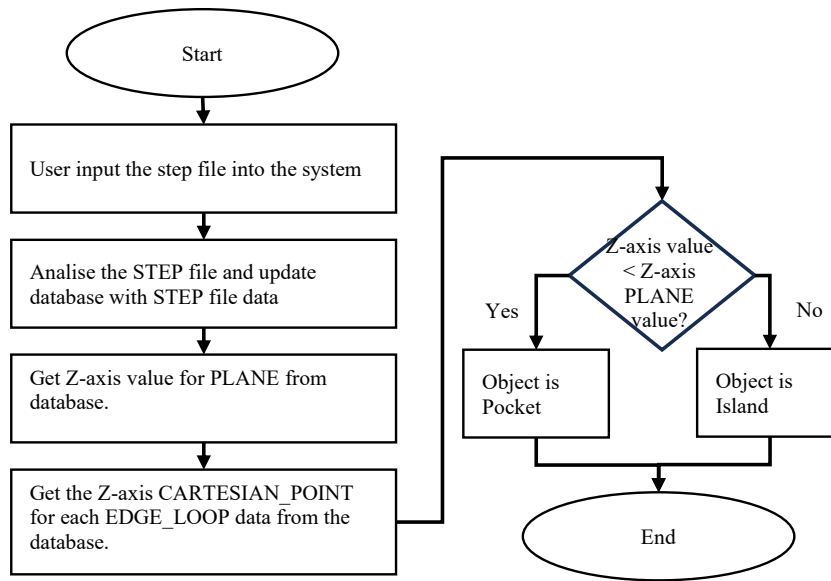


Figure 3 IPFRS Basic Island and Pocket Recognition Algorithm Flowchart

2.4. G-Code Generation

Following the successful identification of islands and blind pockets, the next vital phase involves generating the corresponding G-code for machining. This process involves several necessary steps to ensure both efficiency and precision in machining operations. The first stage is toolpath planning. The software analyses through the recognised features. Using this data, the number of loops for each object needed is calculated. The user is given the option to select between two cutting tools: a 5 mm or 10 mm flat endmill, to perform machining based on their specific requirements. Additionally, spindle speed remains user-configurable, enabling optimisation based on material type and desired surface finish.

The process of determining the number of loops for the machining process required for efficient toolpath generation while following the machining parameters, such as endmill diameter, offset, overlapping percentage, width, and length of the island and blind pocket, and many more. The calculation process is slightly different between the island and the pocket. The pocket is machined

from the offset parameter to the centre of the pocket, while the island extends from the offset parameter to the end of the workpiece. The line for pocket calculation is shown in Equation (1) below. The calculation process involves comparing the length and width. The highest value is chosen and labelled as L, which stands for distance.

$$Na = \frac{\left(\frac{L}{2}\right) - (D)}{P * D} \quad (1)$$

Na: Number of lines (tool paths) for pocket.

Nb: Number of lines (tool paths) for island.

Nl: Number of levels.

L: Total distance or length (e.g., the width or span to cover).

D: Endmill diameter.

Dp: Depth.

P: percentage (as a decimal, e.g., 0.5 for 50% overlap).

Equation (1) calculates the number of tool paths, or "lines," required for machining a given distance. The total distance L is divided by 2 to focus on one side of the object, since the calculation often assumes symmetric coverage from the centre. Next, half the endmill diameter (0.5*D) is subtracted to account for the tool's offset, preventing it from cutting beyond the intended boundaries. The result of this subtraction is then divided by the product of the percentage and the endmill diameter (P*D), which determines the incremental distance between each line based on the desired overlap.

$$Nb = \frac{\left(\frac{L}{2}\right) - (0.5 * D)}{P * D} \quad (2)$$

The island will have the process as shown in equation (2) for getting the number of loops. Firstly, the distance between the object and the edge of the workpiece is calculated from all relevant directions. The result is then compared. The highest value is chosen and labelled as length (L). The next half-endmill is subtracted from the length to determine the offset. The result of this subtraction is then divided by the product of the percentage and the endmill diameter (P*D) for the overlap calculation.

$$Nl = \frac{Dp}{P * D} \quad (3)$$

The number of cutting levels or passes can be calculated using Equation (3), based on depth values extracted from the STEP file analysis for each object. First, the depth (Dp) of an object is obtained. Next, the depth is divided by the percentage endmill diameter (P*D). This equation is working for both the island and the pocket.

Table 1 provides an example of G-code syntax. The example includes rapid movement (G00), straight-line movement (G01), clockwise arc movement (G02), counterclockwise arc movement

(G03), spindle speed (S1000), and feed rate (F400). The values for spindle speed and feed rate depend on several factors, including the material type, endmill diameter, and cutting depth.

Table 1 Example of G-Code Syntax

No	Syntax	Description
1	G00	Rapid movement to the locations
2	G01	Straight line movement
3	G02/G03	Arc movement, G02 clockwise and G03 counter-clockwise
4	S1000	Spindle speed value
5	F400	Feed rate value

The data in Figure 2 provides the information necessary to determine the required syntax for the related Cartesian point. The figure containing the keyword LINE instructs the system to use the G01 syntax as its target. In the CIRCLE, the syntax to be used is either G02 or G03, depending on the previous Cartesian point. The G-code is developed after the type of object is determined, because the process of generating G-code for an island differs from the process of generating a pocket. Still, it uses the same process to generate the G-code syntax. The difference is in the direction of the end mill movement. In the pocket, the movement is toward the centre of an object, whereas on the island, it is outward.

Table 2 Syntax code identification

No	Syntax	Identifier	Value Source
1	G01	LINE	LINE → CARTESIAN_POINT
2	G02/G03	CIRCLE	CIRCLE → AXIS2_PLACEMENT_3D → CARTESIAN_POINT
3	R	2 nd value in CIRCLE	CIRCLE("#56,15).

As shown in Figure 2, and Table 2 syntax code identification to specific movements. CIRCLE syntax corresponds to arc paths, while LINE relates to straight-line trajectories. Coordinates are sourced from CARTESIAN_POINT entries. For instance, #32=CARTESIAN_POINT(",(90.,30.,-9.)) traces back to #49=EDGE_CURVE(",#44,#33,#48,.T.), which includes #48=LINE(",#45,#47). From this, G-code is generated. LINE maps to G01, with CARTESIAN_POINT values yielding X90,Y30,Z-9.

Meanwhile, #43=CARTESIAN_POINT(",(90.,20.,-9.)) links to #58=EDGE_CURVE(",#52,#44,#57,.T.), containing #57=CIRCLE(",#56,15.). CIRCLE maps to G03, and CARTESIAN_POINT values produce X90 Y20 Z-9, with the radius (15) added as R15. The output commands are G01 X90 Y30 Z-9 and G03 X90 Y20 Z-9 R15. This extraction process repeats until all EDGE_LOOP data is processed. Using the island and pocket processes, combined with the number of lines and the required level, generates the complete G-code for each object.

The next step is to sequence the machining order. In this phase, the system prioritises machining islands first, as these raised features must be preserved during material removal, followed by machining blind pockets. Once the sequencing and toolpaths for each object are generated, they are then combined according to the previously determined sequence.

After the final toolpath has been created, the software switches to G-code syntax to generate G-code commands for the planned toolpath. Those commands include instructions for correctly positioning the tool, controlling the spindle speed, and adjusting the feed rate. Each of these elements is important for accurately executing the machining process and ensuring the final output meets the specifications.

2.5. Technology Used

The web-based application development used various technologies. It is designed to enhance functionality, performance, and the overall user experience. In this case, HTML, CSS, and JavaScript were used on the front end to build a responsive, interactive GUI. Bootstrap.js framework provides additional support for layout management and component styling. On the back end, PHP was used as the server-side scripting language to manage file uploads, handle user interactions, and execute core processes. This process includes Geometric Data Extraction (GDE) and feature recognition algorithms. A MySQL database was integrated to store user data and intermediate processing results. This practice is enabling efficient data retrieval and system scalability [17].

The application was hosted using an Apache server. This server facilitated seamless communication between the front-end and back-end components [18, 19]. Sublime Text served as the primary development tool. It provides efficient code management, syntax highlighting, and debugging tools. For testing and deployment, the Opera web browser was primarily used to verify the application's functionality and responsiveness. Its consistent rendering engine ensured that the web-based system behaved as intended across different use cases and environments.

3 Results and Discussion

3.1 Interface Result

As a result, Figure 4 shows the primary interface for the Island Pocket Feature Recognition System (IPFRS). The main interface allows users to upload STEP files into the system, which are then stored in the database. There is also a clear database button to remove the inserted data. Next, the 'Clear Folder' button is used to remove the file from the specific folder.

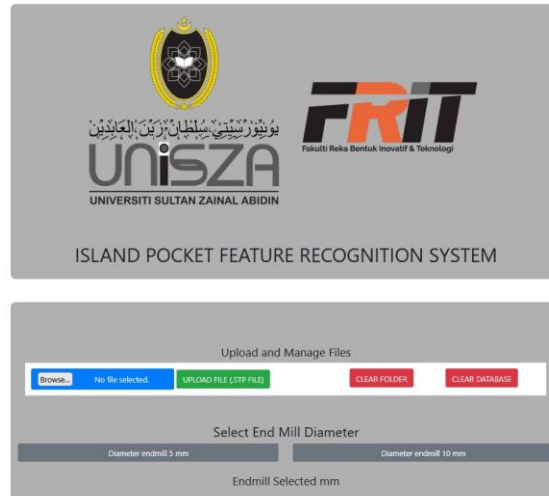


Figure 4 Island Pocket Feature Recognition System (IPFRS)

The figure also shows that the endmill selection for machining purposes is to be chosen from. From there, there are two selections to be made: either an endmill with 5 mm or an endmill with 10 mm. Once the selection has been made, the selected value is displayed to confirm it to the user. The choice of endmill size will affect the number of lines the system will produce. The smaller the endmill, the more lines it produces [20].

WORKPIECE SIZE					
Min-X	Min-Y	Min-Z	Max-X	Max-Y	Max-Z
0	0	-24	100	100	0

DEPTH OF OBJECT	
-9 mm	-18 mm

START G-CODE	

END G-CODE	

[Submit G-Code](#)

Figure 5 Workpiece Size, Depth of Object, Start and End G-Code

Figure 5 illustrates the workpiece size, the object's depth, the start G-code, and the end G-code for the CNC process. Displaying the workpiece size helps users understand the designed object and select appropriate machining operations, supporting the correct identification of the workpiece [21]. Firstly, the workpiece size. It defines the machining area boundaries and specifies the minimum and maximum values for each axis. For example, the x-axis ranges from 0 to 100, the y-axis from 0 to 100, and the z-axis from -24 to 0. This helps the software limit machine movements to prevent errors during production. Next is the object's depth. The example in the figure below contains two values, which means this file contains two objects. By comparing these two values to the workpiece size, the user can determine which is an island and which is a pocket. The lowest depth can be considered the pocket, and the other is the island.

After the in-depth analysis, the application has dedicated text areas for the user-defined G-code start and end sequences, which are vital for customising the machining process. The input slots accommodate user preferences to customise the machine's initialisation and termination routines, as different CNC machines or workflows may require unique commands to control the spindle, perform tool changes, or set safety protocols. This allows for compatibility and flexibility, enabling users to enter specific G-code snippets tailored to their equipment and operational needs, thereby facilitating a smooth transition from design to production

G-CODE RESULT					
G	X	Y	Z	R	
G01	X100	Y100	Z-2.5		
G03	X94	Y100	Z-2.5	R17.5	
G01	X0	Y100	Z-2.5		
G03	X0	Y100	Z-2.5	R17.5	
G01	X0	Y16	Z-2.5		
G03	X0	Y0	Z-2.5	R17.5	
G01	X94	Y0	Z-2.5		
G03	X100	Y16	Z-2.5	R17.5	
G01	X100	Y100	Z-2.5		
G01	X100	Y100	Z-2.5		
G03	X92	Y100	Z-2.5	R17.5	
G01	X0	Y100	Z-2.5		
G03	X0	Y100	Z-2.5	R17.5	
G01	X0	Y18	Z-2.5		

Figure 6 View G-code Result

Figure 6 above shows the view area for the G-code result generated by the system. It contains six columns to accommodate the six markers for G-code production. They are N-number, the G column, the X-axis column, the Y-axis column, the Z-axis column and the R column for radius. The G-code generated by the system is displayed here. This area allows the user to review the generated G-code before downloading

N1496	G01	X47.5	Y20	Z-18	
N1497	G03	X80	Y7.5	Z-18	R12.5
N1498	G01	X80	Y7.5	Z-18	
N1499	G03	X92.5	Y20	Z-18	R12.5
N1500	G01	X92.5	Y30	Z-18	

Download G-CODE

Download G-Code

Figure 7 Download G-code button.

Figure 7 above shows the button to download the G-code. This button allows the user to download the G-code generated by the system. The download button helps users easily access the generated data without having to copy it one by one. This will result in quicker data capture and reduce the likelihood of human error [22].

3.2 G-Code Result

The IPFRS system was tested with two flat endmill diameters, 5 mm and 10 mm, to evaluate its ability to generate accurate, valid G-code for CNC machining operations. A STEP AP242 file was uploaded into the system, and the corresponding toolpath was automatically generated according to the selected tool size. This change aims to assess the system's capacity to adapt to changes in tool diameter and to confirm the accuracy of the machining sequence. The simulation is used to verify the generated G-code.

Table 3 Extracted Portion of G-code Generated by IPFRS for 5 mm Endmill Machining

N	G	X	Y	Z	R
N1	G01	X100	Y100	Z-2.5	
N2	G03	X100	Y100	Z-2.5	R17.5
N3	G01	X0	Y100	Z-2.5	
N4	G03	X0	Y100	Z-2.5	R17.5
N5	G01	X0	Y16	Z-2.5	
N6	G03	X0	Y0	Z-2.5	R17.5
N1411	G01	X54.5	Y12	Z-18	
N1412	G03	X42	Y24.5	Z-18	R12.5
N1413	G01	X58	Y24.5	Z-18	
N1414	G03	X45.5	Y12	Z-18	R12.5
N1415	G01	X45.5	Y38	Z-18	
N1416	G03	X58	Y25.5	Z-18	R12.5
N1417	G01	X42	Y25.5	Z-18	
N1418	G03	X54.5	Y38	Z-18	R12.5
N1419	G01	X54.5	Y12	Z-18	
N1420	G01	X56.5	Y14	Z-18	
N1421	G03	X44	Y26.5	Z-18	R12.5
N1422	G01	X56	Y26.5	Z-18	
N1423	G03	X43.5	Y14	Z-18	R12.5
N1424	G01	X43.5	Y36	Z-18	
N1411	G01	X54.5	Y12	Z-18	
N1412	G03	X42	Y24.5	Z-18	R12.5
N1413	G01	X58	Y24.5	Z-18	
N1414	G03	X45.5	Y12	Z-18	R12.5
N1415	G01	X45.5	Y38	Z-18	
N1416	G03	X58	Y25.5	Z-18	R12.5

Table 4 Extracted Portion of G-code Generated by IPFRS for 10 mm Endmill Machining

N	G	X	Y	Z	R
N1	G01	X100	Y100	Z-5	
N2	G03	X100	Y100	Z-5	R20
N3	G01	X0	Y100	Z-5	
N4	G03	X0	Y100	Z-5	R20
N5	G01	X0	Y18	Z-5	
N6	G03	X0	Y0	Z-5	R20
N392	G01	X66	Y26	Z-18	
N393	G03	X56	Y36	Z-18	R10
N394	G01	X44	Y36	Z-18	
N395	G03	X34	Y26	Z-18	R10
N396	G01	X34	Y24	Z-18	
N397	G03	X44	Y14	Z-18	R10

N398	G01	X56	Y14	Z-18	
N399	G03	X66	Y24	Z-18	R10
N400	G01	X66	Y26	Z-18	
N401	G01	X70	Y30	Z-18	
N402	G03	X60	Y40	Z-18	R10
N403	G01	X40	Y40	Z-18	
N404	G03	X30	Y30	Z-18	R10
N405	G01	X30	Y20	Z-18	
N406	G03	X40	Y10	Z-18	R10
N407	G01	X60	Y10	Z-18	
N408	G03	X70	Y20	Z-18	R10
N409	G01	X70	Y30	Z-18	

The table above shows the results of the IPFRS system. Table 3 presents the results for the 5 mm endmill, while Table 4 presents the results for the 10 mm endmill. According to the G-code, the machining process begins with producing the island, followed by the pocket. The visualisation is shown in Figures 7 and 8.

In both tables, the N1-N6 lines show the part of the G-code produced for machining the island. The table also indicates that the machining process starts from the outside and moves inward toward the island object. The radius of the G-code is also changed to compensate for the radius of the endmill. The initial radius value in the G-code is R15. The radius change was increased to 2.5 mm from the initial object for a 5 mm endmill, producing R17.5, while 5mm is for a 10 mm endmill. This will end up as R20. Adding a radius ensures the final island meets the specified 15 mm diameter. The machining process of the island will continue until line number N1410.

N1411 through N1416 are for the G-code machining pocket for a 5 mm endmill, while N392 through N409 show the part G-code for a 10 mm endmill. The process of machining a pocket begins at the centre of the object and expands outward until the specified dimensions are met. By design, the pocket radius is 15mm. The G-code was adjusted based on the end mill's radius. The 5 mm endmill was reduced to 12.5, and the R10 for the 10 mm endmill. The increase and decrease in R value for G-code are used to compensate for the endmill's offset, producing an accurately machined object. The final simulated machining result is shown in Figures 8 and 9.

3.3 G-Code Validation

The generated G-code was validated using both simulated and actual machining to ensure accuracy, reliability, and compatibility with CNC hardware. Simulation testing involves using software such as Fusion 360 or Cimco Edit to verify the G-code. This software visualises the tool path on virtual workpieces. Several parameters were analysed, indicating that the G-code is following the machining strategy programmed for this research. Specifically, the island is machined first, followed by the pocket.

The G-code follows the specific geometry and dimensions specified in the object design, including depth, radius, diameter, and length. There must be no syntax errors or collisions observed during the simulation, and all generated toolpaths must be successfully generated. The generated G-code

was validated using a simulated process to ensure accuracy, reliability, and compatibility with CNC hardware, thereby eliminating the need for real machining and reducing material and tool costs.

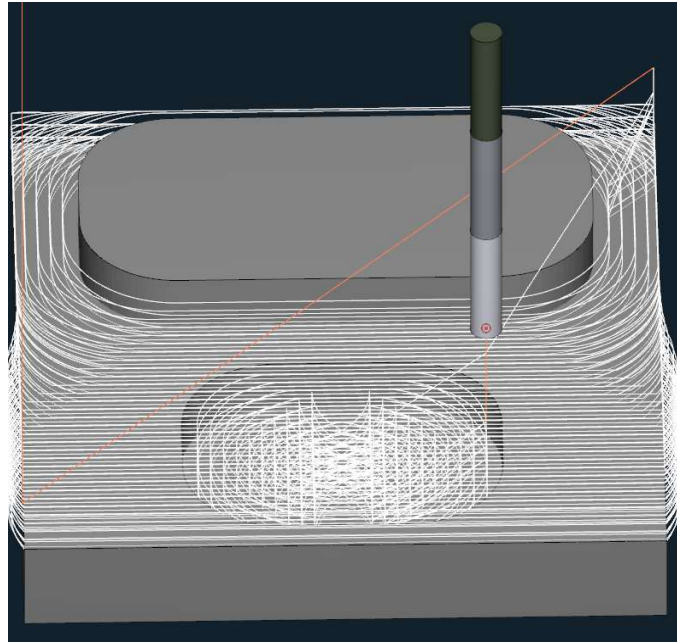


Figure 8 Simulation for Toolpath Using 5 Mm Endmill

Figures 8 and 9 show simulation results for 5 mm and 10 mm flat-end mills produced by the IPFRS. From the simulation, no collisions are detected, and no errors are displayed. This means that the IPFRS has generated toolpaths with a 100 percent success rate with 0 collision. The simulation also shows that the number of lines generated corresponds to the flat-end mill's size. A smaller flat-end mill size and a higher number of lines will be used to complete the machining of an object.

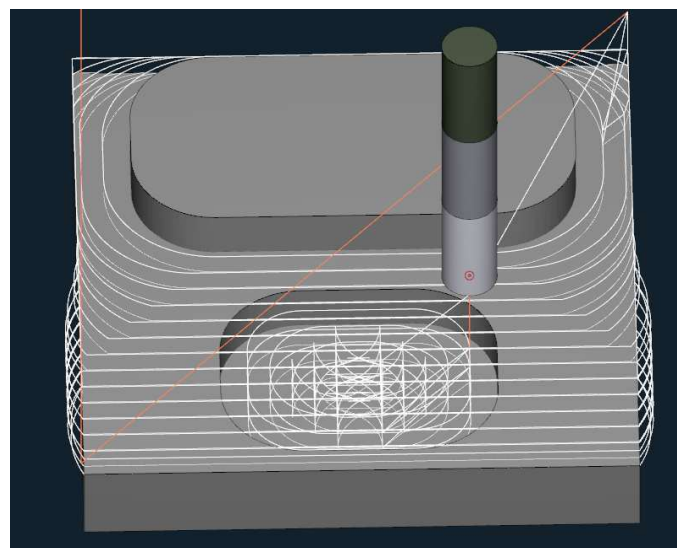


Figure 9 Simulation for Toolpath Using 10 Mm Endmill

4 Conclusion

In conclusion, the findings show that the web-based Island Pocket Feature Recognition System (IPFRS) can identify islands and blind pockets on a workpiece using the STEP file AP242. The system can also generate toolpaths (G-code) for machining using the GDE technique, with a user-friendly, responsive graphical interface. The system streamlines and accelerates G-code generation for CNC machining. The system demonstrates high accuracy in feature detection and flexibility across a range of tool diameters. The system delivers reliable G-code output, validated through simulation trials with zero simulation errors. These findings demonstrate the feasibility of integrating feature recognition with web-based CNC programming, enhancing productivity and minimising human error.

Acknowledgements/ Funding

The authors gratefully acknowledge the support, facilities, and research environment provided by Universiti Sultan Zainal Abidin (UniSZA) throughout this research. The authors also thank all individuals who contributed to the successful completion of the research. This work was funded by the UniSZA industry grant UniSZA/2023/PPL/WENTEC (031).

Authors' Contributions

The authors confirm their contributions to this work as follows: study conception and design: Amirulihسان Mohamad, Zammeri Abd Rahman; data collection and system implementation: Amirulihسان Mohamad; analysis and interpretation of results: Amirulihسان Mohamad, Zammeri Abd Rahman, Saiful Bahri Mohamed; software development and simulation validation: Amirulihسان Mohamad; methodology refinement: Amirulihسان Mohamad, Zammeri Abd Rahman, Muhammad Abdulrahim Rabbani Md Sharizam , Tengku Mohd Sharir Tengku Sulaiman; draft manuscript preparation: Amirulihسان Mohamad, Zammeri Abd Rahman, Saiful Bahri Mohamed, Roslan Awang. All authors reviewed the results, contributed to the manuscript's revision, and approved the final version of the paper.

Declaration

Conflict of Interest

The authors agree that this research was conducted in the absence of any self-benefit, commercial, or financial conflicts and declare that there are no conflicts of interest with the funders.

Availability of Data and Materials

The data and materials that support the findings of this study are available from the corresponding author upon reasonable request.

References

- [1] Yeo C, Kim BC, Cheon S, Lee J, Mun D (2021) Machining feature recognition based on deep neural networks to support tight integration with 3D CAD systems. *Scientific Reports* 11:1–20. <https://doi.org/10.1038/s41598-021-01313-3>
- [2] Nguyen TK, Phung LX, Bui N-T (2020) Novel Integration of CAPP in a G-Code Generation Module Using Macro Programming for CNC Application. *Machines* 8:61. <https://doi.org/10.3390/machines8040061>
- [3] Abd Rahman Z, Mohamed SB, Minhat M, Mohd A (2022) Integrated Interface System of STEP Data Models for the Tool Path Generation. *International Journal of Integrated Engineering* 14:160–170. <https://doi.org/10.30880/ijie.2022.14.04.012>
- [4] Ibrahim Z, Mohamed SB, Minhat M, Mohamed AS, Musanih MR, Rahman ZA (2016) Reason Maintenance in Product Modelling via Open Source CAD System. *International Journal on Advanced Science, Engineering and Information Technology* 6:990–996. <https://doi.org/10.18517/ijaseit.6.6.953>
- [5] Rahman ZA, Mohamed SB, Minhat M, Mohd A, Rahman ZA, Awang R (2023) G-code converter using interface system for a STEP file (ISO 10303). *Indonesian Journal of Electrical Engineering and Computer Science* 31:1362–1371. <https://doi.org/10.11591/ijeecs.v31.i3.pp1362-1371>
- [6] Koura M, Koura O, Sayed T (2024) GEOMETRIC DATA EXTRACTION METHODOLOGY FROM STEP FILE FOR CYLINDRICAL PARTS. *Journal of Al-Azhar University Engineering Sector* 19:698–708. <https://doi.org/10.21608/auej.2024.259608.1580>
- [7] Apiag C, Cadiz E, Lincopinis D (2023). A Review on PHP Programming Language.
- [8] Bi T, Xia X, Lo D, Grundy J, Zimmermann T, Ford D (2022) Accessibility in Software Practice: A Practitioner’s Perspective. *ACM Transactions on Software Engineering and Methodology* 31:1–26. <https://doi.org/10.1145/3503508>
- [9] Ahmad Z, Rahmani K, D’Souza RM (2010) Applications of genetic algorithms in process planning: tool sequence selection for 2.5-axis pocket machining. *Journal of Intelligent Manufacturing* 21:461–470. <https://doi.org/10.1007/s10845-008-0201-6>
- [10] Pajaziti A, Tafilaj O, Gjellaj A, Berisha B (2025) Optimization of Toolpath Planning and CNC Machine Performance in Time-Efficient Machining. *Machines* 13:65. <https://doi.org/10.3390/machines13010065>
- [11] Li P, Chen M, Ji C, Zhou Z, Lin X, Yu D (2024) An Agent-Based Method for Feature Recognition and Path Optimization of CNC Machining Trajectories <https://doi.org/10.20944/PREPRINTS202408.0453.V1>.

- [12] Xu Z, Selvaraj V, Min S (2024) Intelligent G-code-based power prediction of ultra-precision CNC machine tools through 1DCNN-LSTM-Attention model. *Journal of Intelligent Manufacturing* 36:1237–1260. <https://doi.org/10.1007/s10845-023-02293-z>
- [13] Zhang Y, Sang S, Bei Y (2023) Image to G-Code Conversion using JavaScript for CNC Machine Control. *Academic Journal of Science and Technology* 6:62–66. <https://doi.org/10.54097/ajst.v6i3.10328>
- [14] Tzotzis A, Manavis A, Efkolidis N, Kyratsis P (2021) CAD-BASED AUTOMATED G-CODE GENERATION FOR DRILLING OPERATIONS. *International Journal of Modern Manufacturing Technologies* 13:177–184. <https://doi.org/10.54684/ijmmt.2021.13.3.177>
- [15] Fan S, Yu Y, Dai S, et al (2017) STEP AP242 Format Expression and Development of MBD Model. Atlantis Press <https://doi.org/10.2991/msmee-17.2017.170>.
- [16] Rahman ZA, Mohamed SB, Minhat M, Rahman ZA, Kasim MS, Mohamad WNF (2021) Design and fabrication of a PC-based 3 axis CNC milling machine, *international Journal of Engineering Trends and Technology* 69:1–13 <https://doi.org/10.14445/22315381/IJETT-V69I9P201>.
- [17] Hosen MS, Islam R, Naeem Z, Folorunso EO, Chu TS, Al Mamun MA, Orunbon NO (2024) Data-Driven Decision Making: Advanced Database Systems for Business Intelligence. *Nanotechnology Perceptions* 20:687–704. <https://doi.org/10.62441/nano-ntp.v20is3.51>
- [18] Gurusamy A, Mohamed IA (2020) The Evolution of Full Stack Development: Trends and Technologies Shaping the Future. *Journal of Knowledge Learning and Science Technology* ISSN: 2959-6386 (online) 1:100–108. <https://doi.org/10.60087/jklst.vol1.n1.p108>
- [19] Yunrui Q (2018) Front-End and Back-End Separation for Warehouse Management System. *institute of electrical electronics engineers*, pp 204–208
- [20] Bickel S, Sauer C, Schleich B, Wartzack S (2021) Comparing CAD part models for geometrical similarity: A concept using machine learning algorithms. *Procedia CIRP* 96:133–138. <https://doi.org/10.1016/j.procir.2021.01.065>
- [21] Prasetya F, Fajri BR, Ranuharja F (2020) Development Design Augmented Reality-Based Jobsheet in CNC Programming Subjects INVOTEK: *Jurnal Inovasi Vokasional dan Teknologi* 10: 50–67.
- [22] Rosid UA (2023) Penerapan Aplikasi Web Upload Download menggunakan PHP pada Laboratorium Komputer LP3I Tasikmalaya. *Jurnal Sistem Informasi Galuh* 1:8–14. <https://doi.org/10.25157/jsig.v1i1.2916>

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [SUPPLEMENTARYMATERIALSApril2026.docx](#)