RESEARCH-ARTICLE

Layer-by-Layer Toolpath Simulation for Error Detection and Correction during Live 3D Printing

MILAN SUMEGI, University of Central Lancashire, Preston, Lancashire, U.K.

WEI QUAN, University of Central Lancashire, Preston, Lancashire, U.K.

HADLEY L BROOKS, University of Central Lancashire, Preston, Lancashire, U.K.

LIK KWAN SHARK, University of Central Lancashire, Preston, Lancashire, U.K.

Open Access Support provided by:

University of Central Lancashire

Published: 13 June 2025

ICCMS 2025: 2025 The 17th International Conference on Computer Modeling and Simulation (ICCMS)
June 13 - 15, 2025
Zhuhai, China

ICCMS '25: Proceedings of the 2025 17th International Conference on Computer Modeling and Simulation (June 2025)
https://doi.org/10.1145/3761668.3761696
ISBN: 9798400713156

Layer-by-Layer Toolpath Simulation for Error Detection and Correction during Live 3D Printing

Milan Sumegi
Scool of Engineering and Computing
Univeristy of Lancashire
Preston, United Kingdom
msumegi@lancashire.ac.uk

Wei Quan
Scool of Engineering and Computing
Univeristy of Lancashire
Preston, United Kingdom
wquan@lancashire.ac.uk

Hadley Brooks
Scool of Engineering and Computing
Univeristy of Lancashire
Preston, United Kingdom
hlbrooks1@lancashire.ac.uk

Lik-Kwan Shark
Scool of Engineering and Computing
Univeristy of Lancashire
Preston, United Kingdom
lshark@lancashire.ac.uk

Abstract

This paper presents a methodology for simulating 3D printing processes from G-code, generating accurate layer-by-layer visualizations, and aligning virtual camera views with physical setups for real-time monitoring, error detection, and in-process correction. By parsing G-code, extracting extrusion commands, and simulating toolpath trajectories, a detailed 3D representation of the print process is created. To enhance realism, real image data—such as noise, brightness variations, and contrast adjustments—is incorporated, improving alignment between simulated and captured images. Additionally, camera field-of-view alignment is achieved using rotation matrices and translation vectors, bridging the gap between simulation and real-world observations. The proposed methods enable real-time error detection and correction, streamlining simulation and visualization workflows to enhance additive manufacturing processes.

CCS Concepts

• **Computing methodologies** → Simulation evaluation; Model verification and validation; • **Hardware** → Emerging tools and methodologies.

Keywords

3D printing, 3D printing toolpath simulation, simulation, image processing, quality assessment, 3D printing error correction

ACM Reference Format:
Milan Sumegi, Wei Quan, Hadley Brooks, and Lik-Kwan Shark. 2025. Layer-by-Layer Toolpath Simulation for Error Detection and Correction during Live 3D Printing. In *2025 The 17th International Conference on Computer Modeling and Simulation (ICCMS) (ICCMS 2025), June 13–15, 2025, Zhuhai, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3761668.3761696

ICCMS 2025, Zhuhai, China

ACM ISBN 979-8-4007-1315-6/2025/06
https://doi.org/10.1145/3761668.3761696

1 Introduction

The adoption of 3D printing, particularly Fused Filament Fabrication (FFF), has grown significantly due to its flexibility, cost-effectiveness, and ability to fabricate complex geometries [1]. However, challenges such as print failures and inaccuracies persist, often arising from material deposition errors and misalignment between simulated and actual prints [2]. To address these issues, various error detection methods have been explored, including touch probe, acoustic, and camera-based approaches. While touch probes and acoustic methods have inherent limitations, camera-based techniques have emerged as the most promising. These methods can be broadly classified into comparison-based and pre-trained network-based error detection techniques [3].

This paper focuses on using a visual representation of the object to compare it with the actual object captured by a camera. Some previous studies have utilized existing visualization toolboxes, such as the OctoPrint visualizer toolbox, to capture screenshots for analysis [4]. Simulating individual layers enables monitoring of the outer shape contour and internal structures, such as infill and pockets, as long as the shapes remain 2D projections perpendicular to the Z-axis [5]. However, most existing methods do not incorporate real-time compensation or correction of 3D printing failures during the printing process. Extracting the contour from the simulated G-code and applying it to the actual printed object allows for detecting material displacement. This method enables corrections or process termination when dealing with concave 3D shapes. However, a key limitation is an inherent delay, as corrections cannot be applied to the current layer in real-time [6]. Simulating 3D printing from G-code provides a powerful way to visualize and validate the manufacturing process before execution [7]. However, aligning contours or layer parameters on a 3D object is challenging, and binary comparison can fail to detect errors when the object has concavities. Instead of directly comparing the simulated toolpath to the camera-captured object, this approach analyses structural differences between corresponding layers in the simulated toolpath and the actual printed object. This method enhances accuracy by enabling rapid and precise identification of discrepancies.

This paper proposes a methodology involved in parsing G-code, simulating toolpaths, and reconstructing 3D objects, along with

techniques for aligning simulated views with physical camera observations to improve accuracy. The correction process is applied to errors that can be corrected during printing. The results section presents a comparison between the simulated and actual frames, demonstrating how closely the simulated output aligns with the real-world print. Finally, the conclusion outlines future work, limitations, and key successes.

2 Methodology

The proposed system is a comparison-based error detection and correction system. It utilises the G-code file to generate a 3D simulated view of the corresponding section of the print. Simultaneously, the system captures an image of the actual printed section after each layer. Both the camera image and the simulated image undergo preprocessing steps, including cropping, background subtraction, and gray scaling. Once the images are standardized, a Between-layer Structural Similarity (BLSS) process can be applied. This process includes a structural similarity measure that compares the current frame to the previous frame, identifying differences between the layers [3]. These differences are then analysed in the frequency domain, resulting in a quantified output. This output is used to establish upper and lower boundary limits of acceptable results. If the result falls within the threshold value, the printing continues to the next layer (if applicable). If the result exceeds the threshold, the system alerts the operator to assess whether the print can be salvaged. If the error is deemed irreparable, the print is terminated. If it is repairable, the system evaluates whether the error notification was a false alarm or if the issue can be addressed during post-processing. For errors requiring immediate repair, the printer reprints the current layer using the material's maximum allowable nozzle temperature and reduces the printing speed to 5%. This approach ensures that missing material is deposited accurately. Excess material, if any, can be removed during post-processing. The entire process is summarised in the flowchart shown in Figure 1.

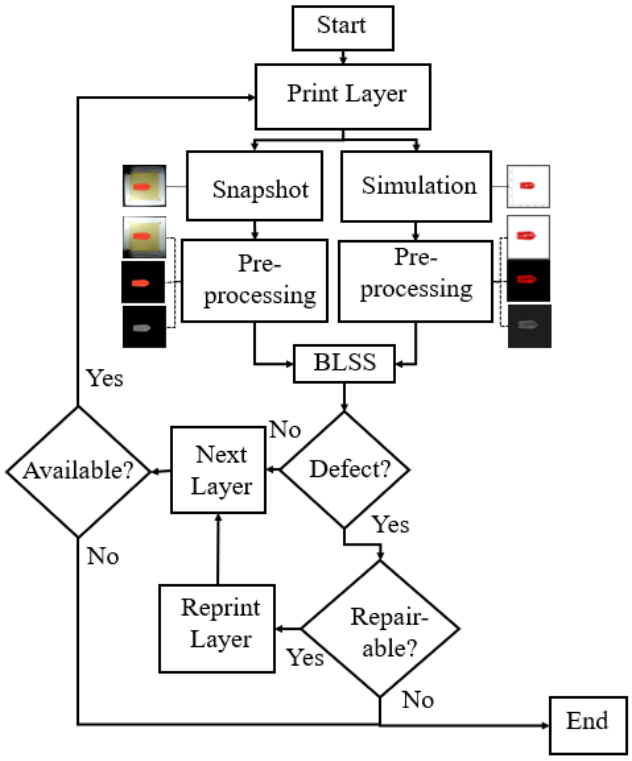

Figure 1: 3D Printing error detection and correction flowchart

3 Art of Simulation

The simulation of a 3D printing toolpath involves extracting data from G-code, interpreting movement commands, visualising the print layer by layer, and generating structured images for analysis. This method ensures an accurate digital reconstruction of the printing process and allows alignment with real-world camera views.

G-code is a standard instruction language that defines printer movements, extrusion commands, and operational settings. The process begins with reading the G-code file and determining the extrusion mode (absolute or relative), which is essential for correctly interpreting extrusion values. The system then identifies layer change markers based on slicer-specific conventions, such as ";BEFORE_LAYER_CHANGE" (PrusaSlicer) or ";LAYER:" (Cura), ensuring compatibility across different slicing software. Once the layer markers are identified, the simulation proceeds iteratively. Each layer's commands are extracted from the G-code and processed sequentially. Movements defined by G0 (non-extrusion) and G1 (extrusion) are analysed to track the toolhead's position in X, Y, and Z coordinates. The parser accumulates movement data to distinguish between travel paths and extrusion paths, ensuring an accurate representation of the deposited material. For each layer, a 3D plot is created, and the extracted toolpath coordinates are plotted dynamically. The system continuously checks extrusion activity, accumulating tool head positions where the material is deposited. Once an entire layer's extrusion paths are processed, plot limits and labels are applied, ensuring that each visualisation aligns with the defined build plate dimensions. After completing a layer, the simulation saves an image (Layer_X_model.png) within an output directory named after the model. The system then determines whether there are more layers left to process. If so, it loops back to extract and simulate the next layer. Otherwise, the process concludes, and the final set of visualisations is generated, the workflow represented in Figure 2.

Visualization of the toolpath is achieved by plotting the extracted coordinates. Figure 3A presents the camera-captured image of the object at Layer 100 of the 3D Benchy. The goal is to reconstruct the model in the best possible way to match the simulation. Extrusion paths are rendered as continuous 3D lines, highlighted in red, while non-extrusion moves are made transparent to avoid affecting feature post-processing. The toolpath for each layer is plotted similarly to Layer 100, as shown in Figure 3B. Stacking these single-colour toolpaths layer by layer forms the overall object; however, only the outer shape remains visible, while internal features are lost. To ensure visibility of internal structures and line connections, it is necessary to add contours to the lines. This is achieved during

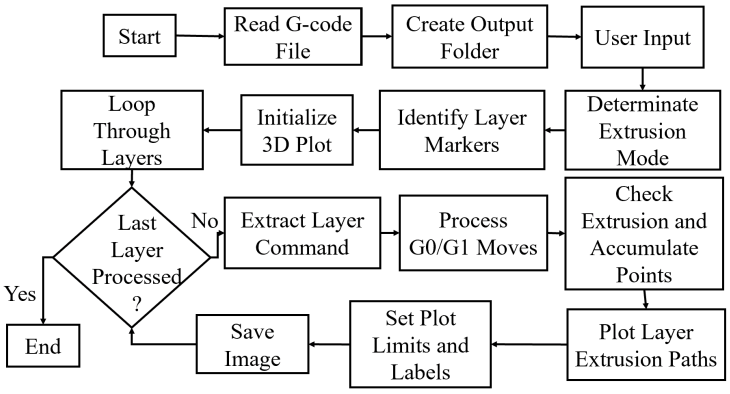

Figure 2: G-code Parsing and Data Integration Workflow

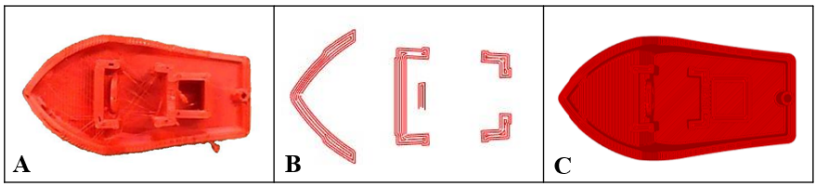

Figure 3: A) Camera image of the 100th layer, B) Simulated 100th layer, C) Simulated model view at layer 100.

toolpath plotting by rendering each layer twice using two different shades of the same colour, with one slightly wider than the other. This approach enhances the visibility of internal details, as demonstrated in the single-layer toolpath view in Figure 3C.

3.1 Camera view scene reconstruction

Camera alignment is essential for validating the simulation against real-world observations. Camera calibration was carried out using the MATLAB Camera Calibration Toolbox [8]. The intrinsic parameters—including focal length, principal point, and distortion coefficients- define the internal characteristics of the camera, while the extrinsic parameters (rotation matrix and translation vector) describe the camera's position and orientation relative to the build plate. See Figure 4 for calibration results.

The proposed methodology was evaluated using various 3D models, ranging from simple geometries like cubes to more complex shapes such as low-poly animal models. In each test, simulated toolpaths were compared against physical prints to assess accuracy. The low-poly fox model demonstrated the system's ability to handle intricate geometries, with high-resolution simulations capturing subtle deviations in layer alignment (Figure 5.).

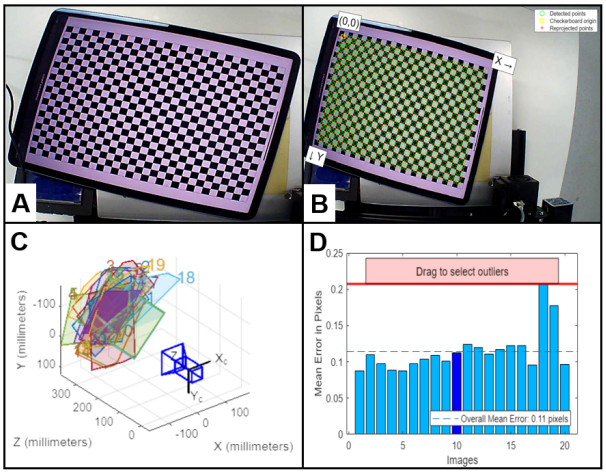

Figure 4: A) Camera calibration using an A4 checkerboard with 10 mm square size and 20 images. C) Pose estimation and reprojection error analysis. D) Mean reprojection error of 0.45 pixels.

3.2 Preprocessing and BLSS

To accurately calculate the similarity between the simulated data and the camera images, both sets of images must undergo a preprocessing stage. If the simulated image is not already in a perpendicular view from one of the axes, it needs to be rotated accordingly. Meanwhile, camera images require distortion correction using the camera calibration data mentioned in the previous section, as well as perspective rotation if necessary. After these corrections, both the simulated and camera images are cropped around the build surface edges to eliminate unnecessary background features, ensuring a consistent comparison framework. Now that both images represent the same section, the next step is to bring them closer in terms of visual properties. Although the 3D view is generated using different contours, the simulated images lack the same characteristics as the camera images. Unlike real images, simulated images have no noise, exhibit perfect contours, and have uniform

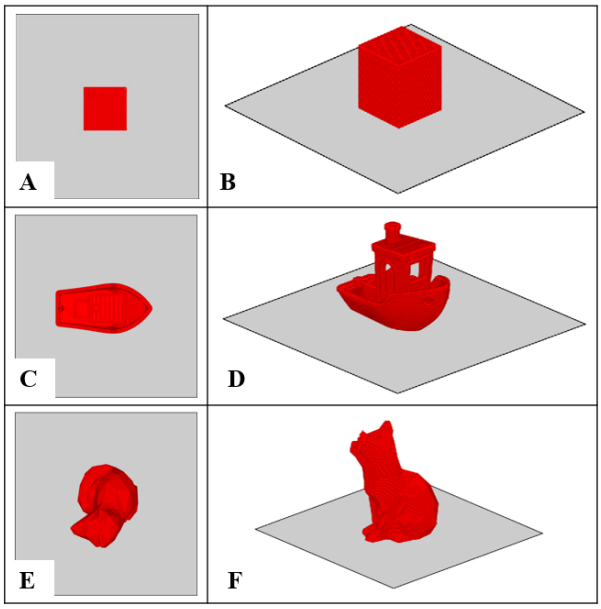

Figure 5: A) Top view of a calibration cube, B) Angle view of a calibration cube, C) Top view of a 3D Benchy, D) Angle view of a 3D Benchy, E) Top view of a low poly fox, F) Angle view of a low poly fox

brightness. To ensure the simulation attains a similar level of image detail, an algorithm extracts key properties from the current camera frame and applies them back to the cropped simulated image. This process standardizes the image properties. Using a preset masking approach on all three colour channels, the build plate is subtracted, leaving only the object visible. The image is then converted to grayscale. Once in grayscale, the object's minimum and maximum values—previously extracted during simulation—are used alongside the build plate's dimensions to calculate the pixel-per-millimetre ratio. With this information, the object's maximum boundary is determined, allowing for precise cropping coordinates. Ensuring both images are of the same size simplifies future processing and comparison.

The grayscale images first undergo Structural Similarity Index Measure (SSIM) [9] analysis, which is applied to subsequent frames within each dataset to generate a difference map over time. After obtaining these difference maps, Fast Fourier Transform (FFT) [10] is applied to transform the images into the frequency domain, allowing for a more detailed comparison of structural variations. Following this, Normalized Cross-Correlation (NCC) [11] is used to quantify the similarity between the transformed difference maps of corresponding frames from the simulated and camera datasets, displayed in Figure 6. [3].

3.3 Simulation Adjustment Using Camera-Derived Parameters

To achieve a closer similarity between the simulated and camera images, it is essential to transfer noise parameters from the original camera image to the simulated images. The process begins with capturing real-world images using a camera and extracting key visual features such as entropy, edge density, noise, contrast, and brightness. To ensure accurate mapping, the camera image is divided into a grid, with each section proportionally matched to the simulated image. Iterative adjustments are then applied to the simulation, aligning its features with the extracted camera data by modifying entropy through histogram equalization [12], enhancing edges using Sobel filtering [13], regulating noise with Gaussian adjustments [14], and normalizing brightness and contrast. A blending factor is incorporated to ensure smooth transitions between modified regions. The pseudocode is displayed in Figure 7.

After adjustment, the simulation is converted back to RGB using the LAB colour model. The final step involves evaluating object similarity by comparing outlines and computing a similarity score to verify alignment. This approach ensures that the simulated image accurately replicates real-world conditions while maintaining structural integrity. In Figure 8, the original simulated SSIM values are represented by the blue line, while the simulated SSIM values with noise parameters are shown in green. The SSIM values from actual camera images, captured from subsequent frames, are depicted in orange. When noise parameters are added to the simulated images, the SSIM values align much more closely with those from the camera images, demonstrating the effectiveness of this approach.

3.4 On-the-fly 3D printing correction

Some errors can be corrected if detected in time during the 3D printing process, particularly material misplacement and extrusion-related issues. When a layer shift is detected, the algorithm reads the previous layer number and modifies the G-code for the incorrectly printed layer.

The adjusted parameters include reducing the print speed to 5% and increasing the nozzle temperature to the maximum the material can handle. With these modifications, the misprinted layer is reprinted, effectively ironing out excess material where it is not needed and depositing new material in areas where it was previously missing. To ensure accuracy, the coordinate system is verified during the homing of the X and Y axes. This approach prevents print failures, facilitates post-processing for excess material removal, and ensures a correct surface, ultimately reducing material costs and manufacturing time.

3.5 Experimental Configuration

All experiments were conducted on a Creality Ender 3 V1 running Marlin firmware, connected via OctoPrint. A 1.0 mm nozzle was used with a 0.5 mm layer height and 1.0 mm line width, and 'thin wall' printing was enabled in the slicer. All 3D models were uniformly scaled to 300% to ensure sufficient resolution and feature visibility. PLA was selected for consistency, and MATLAB handled image acquisition, simulation, and print control via the OctoPrint API, including pause/resume and layer reprint commands. To isolate the computer vision workflow, all tests used a fixed hardware-material setup; broader parameter variations are proposed as future work.

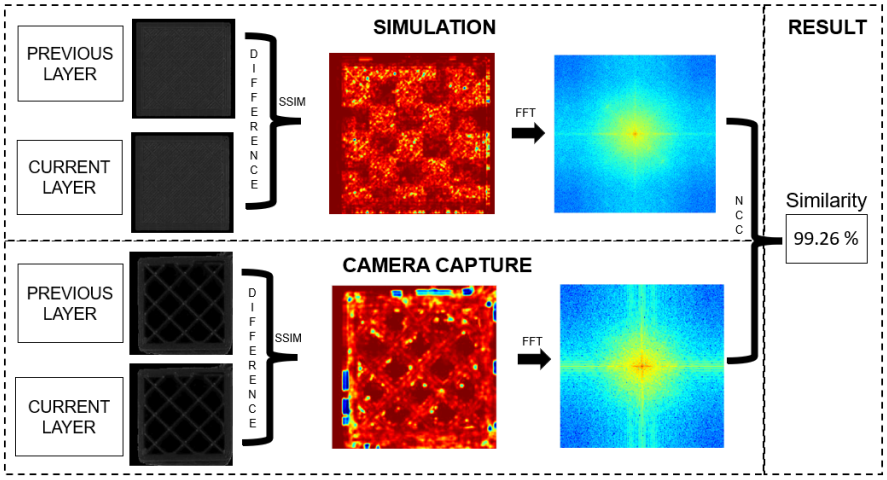

Figure 6: Flowchart of the quantification process between the simulated and the actual camera image difference

```
Input:
    - Latest camera snapshot from folder
    - Folder of simulated images
Output:
    - Simulated images with added noise, blur, and coarseness
        1. Read the latest camera image from the snapshot folder
        2. Convert the image to grayscale and normalize pixel values
        3. Estimate:
            a. Noise level using median filtering difference
            b. Blur using pixel intensity variance
            c. Coarseness using FFT of the noise pattern
        4. For each simulated image:
            a. Convert to grayscale and normalize
            b. Generate low-resolution noise map based on coarseness
            c. Resize noise map to match image size
            d. Smooth the noise with Gaussian blur (blur level from step 3)
            e. Add noise map to simulated image
            f. Clip values to [0, 1] and convert back to 8-bit
            g. Save the enhanced simulated image
```

Figure 7: Noise Feature Extraction and Application to Simulated Images (Pseudocode)

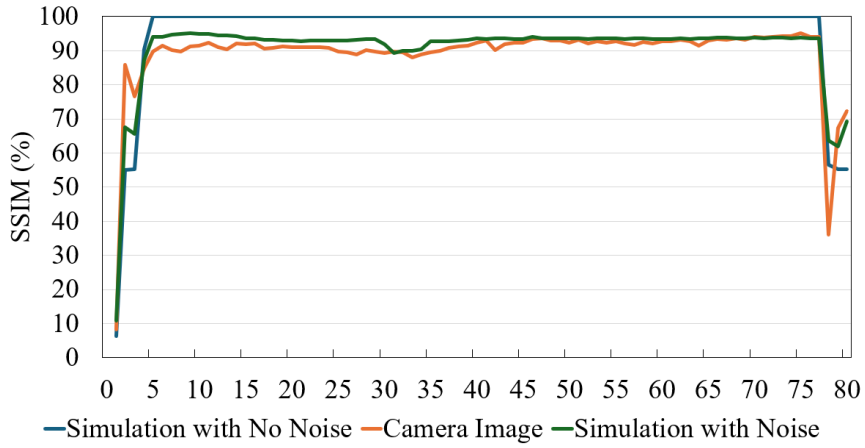

Figure 8: Comparison of structural similarity between camera images and simulations, with and without applied noise.

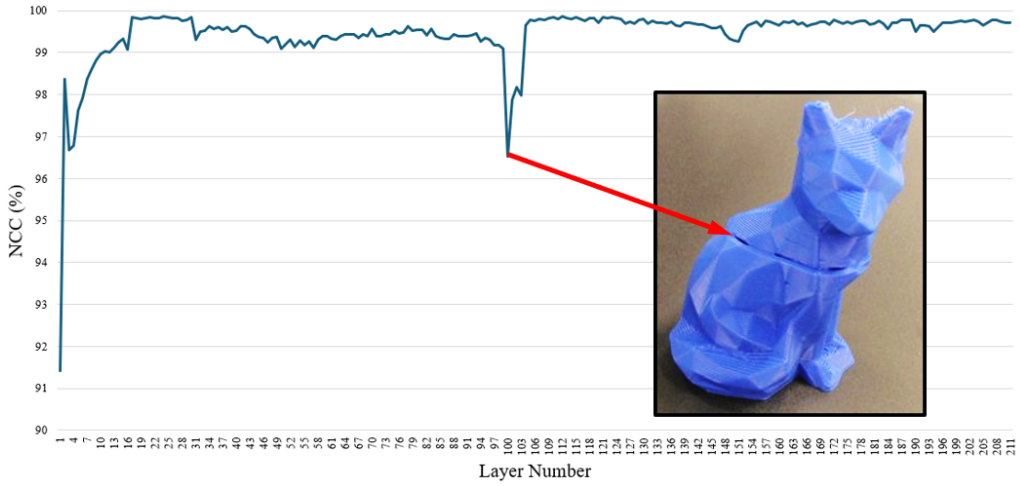

Figure 9: Full print with BLLS generated similarity values throughout the Low Poly Fox layer-shift correction

4 Results

The evaluation of the proposed system begins by assessing the similarity between successive camera images to establish a baseline for detecting variations during printing. Once consistency within the camera images is confirmed, the next step is to compare the simulated and camera images to determine their structural alignment. This analysis is performed using FFT and NCC, allowing for a frequency-domain comparison that highlights differences in spatial features. To validate the system's ability to detect print errors, a controlled experiment was conducted using a fox print, where a deliberate layer shift was introduced at Layer 100. This serves as a benchmark case to assess the detection accuracy of the BLSS process and the subsequent adjustments applied to the simulation. The following subsections present the comparative analysis of real and simulated prints, noise integration effects, and the impact of on-the-fly corrections. To quantify the similarity between the simulated and camera images across layers, NCC was used as a metric. Figure 9. presents the NCC values for each layer, where the vertical axis represents the similarity percentage, and the horizontal axis corresponds to the layer numbers. The graph shows consistently high NCC values across most layers, indicating strong alignment between the simulated and real images. However, a significant drop in similarity is observed at Layer 100, where an artificial layer shift was introduced. This deviation highlights the system's capability to detect print anomalies effectively. The results confirm that the simulation accurately reflects real-world printing conditions while preserving structural consistency. Figure 9. also shows that the system requires approximately five additional frames after the correction to stabilize and return to normal similarity levels. This does not indicate a limitation in continuous monitoring; rather, if a new error occurs, it would result in a much more pronounced drop, ensuring effective real-time detection. Furthermore, the results confirm that this error detection method is not limited to identifying a single error. Once an error is corrected, it will no longer be flagged and any new errors arising during the process will still be detected, ensuring ongoing quality control throughout the print.

A layer shift can ruin a printed part, leading to material waste and increased production time. However, these errors can be corrected during the printing process if detected in time. Once a layer shift is identified, the X and Y axes can be re-homed, and the faulty layer can be reprinted before continuing the print. This corrective approach minimises waste and prevents the need for a full reprint of the part. Figure 10. displays three cases: a perfectly printed fox, a fox with an induced layer shift, and a cube after the layer shift was detected and corrected. The results demonstrate that by modifying the G-code to reprint the misaligned layer with adjusted parameters, the shifted material can be redistributed, ensuring structural integrity.

Although minor surface irregularities may remain, post-processing techniques, such as sanding, can further refine the print, making it functionally accurate while saving material and print time. The post-processed fox and correctly printed fox were both scanned using structured light scanning technology. A surface comparison test was then conducted using GOM Inspection, revealing only slight deviations in the surface, all less than 1 mm, displayed in Figure 11.

5 Conclusion

This paper presents a novel approach for layer-by-layer toolpath simulation in 3D printing, integrating real-time monitoring and error detection using structural similarity measures. By aligning simulated toolpaths with actual printed layers and incorporating real image parameters, the proposed method enhances detection accuracy beyond standard simulation-based approaches. The study demonstrates that adding noise parameters from real images improves similarity analysis and error identification, enabling effective correction of defects such as layer shifts. Experimental results validate the system's ability to detect errors through SSIM, FFT, and NCC analysis, ensuring real-time intervention and correction. The method successfully identifies and corrects layer misalignment, significantly reducing material waste and improving print quality. Additionally, post-correction analysis confirms that errors do

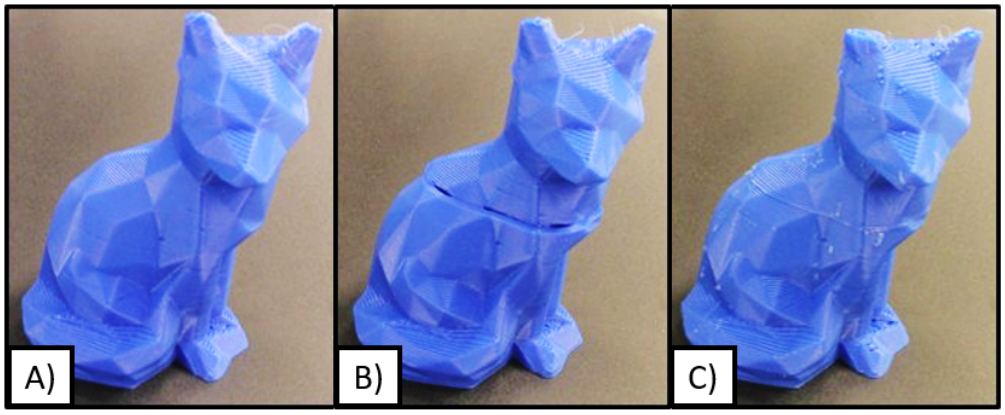

Figure 10: Actual looks of the Low Poly Fox A) Correct Fox B) Layer shift in Fox C) Repaired Fox

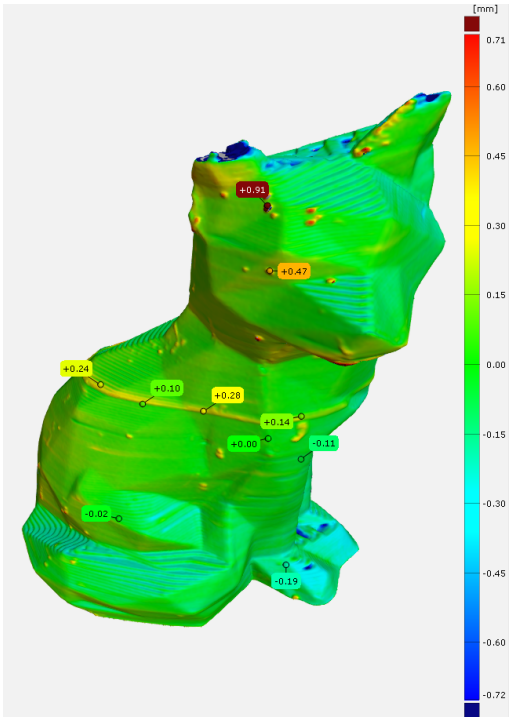

Figure 11: Surface deviation between a correctly printed calibration cube and an altered print of a calibration cube

not propagate once rectified, highlighting the robustness of the approach.

Although the primary experiments were conducted using PLA and a 1.0 mm nozzle, additional prints were also tested with 0.8 mm, 0.6 mm, and 0.4 mm nozzles, as well as with ABS and PETG materials. No significant visual differences in system performance were observed, but further testing is planned to evaluate limitations across varying materials and printer setups.

Future work will also focus on enhancing real-time processing efficiency, exploring multi-camera perspectives for improved accuracy, and integrating machine learning for more advanced defect prediction. By refining these techniques, this research contributes to the development of more autonomous and reliable additive manufacturing processes, reducing human intervention while ensuring consistent print quality.

References

[1] Royal Academy of Engineering (Great Britain), Additive manufacturing: opportunities and constraints: a summary of a roundtable forum held on 23 May 2013 hosted by the Royal Academy of Engineering.
[2] T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Q. Nguyen, and D. Hui, "Additive manufacturing (3D printing): A review of materials, methods, applications and challenges," Jun. 15, 2018, Elsevier Ltd. doi: 10.1016/j.compositesb.2018.02.012.
[3] M. Sumegi, W. Quan, H. L. Brooks, and L. K. Shark, "Real-time Monitoring of 3D Printing Using Between-layer Structural Similarity (BLSS)," in ACM International Conference Proceeding Series, Association for Computing Machinery, Jan. 2024, pp. 278–285. doi: 10.1145/3647649.3647694.
[4] A. Aburaia, C. Holzgethan, C. Ambros, K. Stuja, and B. Katalinic, "Online Vision-based Error Detection for Fused Filament Fabrication," 2021, pp. 193–212. doi: 10.2507/daaam.scibook.2021.16.
[5] Li, W. Quan, L. K. Shark, and H. Laurence Brooks, "A Vision-based Monitoring System for Quality Assessment of Fused Filament Fabrication (FFF) 3D Printing," in ACM International Conference Proceeding Series, Association for Computing Machinery, Jan. 2022, pp. 242–250. doi: 10.1145/3512388.3512424.
[6] L. Petsiuk and J. M. Pearce, "Open source computer vision-based layer-wise 3D printing analysis," Addit Manuf, vol. 36, Dec. 2020, doi: 10.1016/j.addma.2020.101473.
[7] F. Schindler, M. Aburaia, B. Katalinic, M. Lackner, and K. Stuja, "Computer Vision Based Analysis for Fused Filament Fabrication Using a G-Code Visualization Comparison," 2023, pp. 356–371. doi: 10.1007/978-3-031-20875-1_33.
[8] Mathworks, "Camera Calibration," [Online] Available: https://uk.mathworks.com/help/vision/camera-calibration.html. [Accesed: Dec, 2024]
[9] R. Dosselmann and X. D. Yang, "A comprehensive assessment of the structural similarity index," Signal Image Video Process, vol. 5, no. 1, pp. 81–91, Mar. 2011, doi: 10.1007/s11760-009-0144-1.
[10] E. 0 Brigham and R. E. Morrow, "The fast Fourier transform."
[11] F. Zhao, Q. Huang, and W. Gao, "IMAGE MATCHING BY NORMALIZED CROSS-CORRELATION."
[12] K. Okarma and J. Law Fastowicz, "Quality Assessment of Photographed 3D Printed Flat Surfaces Using Hough Transform and Histogram Equalization," Article in JOURNAL OF UNIVERSAL COMPUTER SCIENCE, 2019, doi: 10.3217/jucs-025-06-0701.
[13] N. Kanopulos, N Vasanthavada and R. L. Baker "Design of an Image Edge Detection Filter Using the Sobel Operator." 1988
[14] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical Poissonian-Gaussian noise modelling and fitting for single-image raw-data," IEEE Transactions on Image Processing, vol. 17, no. 10, pp. 1737–1754, 2008, doi: 10.1109/TIP.2008.2001399