Development of automated feature extraction and convolutional neural network optimization for real-time warping monitoring in 3D printing

Jiarui Xie[a], Aditya Saluja[b], Amirmohammad Rahimizadeh[c], Kazem Fayazbakhsh[a]*

[a] *Department of Aerospace Engineering, Ryerson University, Toronto, Ontario M5B2K3, Canada*

[b]*The Irving K. Barber School of Arts and Sciences, University of British Columbia – Okanagan Campus, Kelowna, BC Canada, V1V 1V7*

[c] *Department of Mechanical Engineering, McGill University, Montreal, QC H3A0C3, Canada*

**Corresponding author: kazem@ryerson.ca; Tel: (+1) 416-979-5000 ext. 6414; fax: (+1) 416-979-5056*

https://orcid.org/0000-0003-3963-8282

Abstract

Defects such as warping, which are introduced during additive manufacturing, severely compromise the quality of parts and could even damage the 3D printer. This paper proposes automated feature extraction and hyperparameter optimization in a closed-loop in-process system to monitor warping in fused filament fabrication (FFF). The feature extraction is based on G-code analysis, and map matching between the build platform and the captured image. This allows the warping detection algorithm to be applied to different camera angles, part locations, and corner geometries. Bayesian optimization is adopted to determine the best hyperparameters for the classification model. This model, based on a convolutional neural network (CNN), is executed in a Raspberry Pi pre-configured with OctoPrint, with plugins coordinating and controlling the camera, 3D printer, and microcomputer. Based on 16 tests carried out, the warping monitoring system was determined to be 99.2% accurate.

Keywords: Convolutional Neural Network (CNN); Fused Filament Fabrication (FFF); Warping detection; Automated feature extraction; Bayesian optimization

International Journal of Computer Integrated Manufacturing

https://doi.org/10.1080/0951192X.2022.2025621

1. Introduction

Over the past two decades, additive manufacturing (AM) has emerged as an efficient and rapid manufacturing technique that offers numerous advantages over traditional manufacturing methods (Gardan 2016; Frazier 2014; Zhang and Zhao 2021). Apart from rapid prototyping, the ability of AM to handle complex geometries has significantly increased its application in diverse industries, ranging from those manufacturing sports and leisure products to critical components in the aerospace industries. Low wastage, diverse feedstock materials, and a convenient manufacturing process are three key factors that led to the introduction of fused filament fabrication (FFF) as one of the most widely used AM processes in many application sectors (Popescu et al. 2018; Rahimizadeh et al. 2019). It uses thermoplastic filament extrusion through a nozzle to fabricate a component via a layer-upon-layer method known as 3D printing.

Despite their reliability, components manufactured using FFF often exhibit defects that adversely impact their functionality and performance, thereby leading to significant wastage of material and time. Although it would be advantageous to monitor the fabrication process consistently to detect defective parts, the inherent vulnerabilities of 3D printing could cause the defect detection process to become inaccurate and expensive. This could result in an enormous difference between the properties of as-manufactured and as-designed parts (Wu et al. 2016). Furthermore, monitoring the build process is generally integrated with high-speed scanners and analysis, which can be expensive, complicated, and time-consuming (Gobert et al. 2018). Other typical methods pursued to perform quality checks at critical stages of the 3D printing process include techniques that involve a multi-camera system. They enable image processing to inspect the final fabricated product and perform a pixel-by-pixel comparison with the target object image. For instance, Yi et al. (2017) used a statistical approach to extract cutouts of images obtained from single layers of a 3D printed part to inspect defects. This machine vision method can accurately detect geometrical deviations greater than 0.5 mm from its original design with low computational complexity. However, it becomes ineffective when varying cross-sections and various light conditions are concerned. Rao et al. (2016) deployed a FaroArm laser scanner above the build platform to acquire the point clouds of the parts. They utilized spectral graph theory to construct a dimensional integrity classification tool to classify FFF printed parts of different qualities with high statistical significance (p-value < 0.01). Arguably, dimensional deviation monitoring with a 3D scanner and geometric approximation can achieve high accuracy and avoid image quality challenges. Nevertheless, a camera as monitoring equipment for FFF has gained widespread popularity (Kim et al. 2020). For instance, OctoPrint is a popular web interface with a camera interface to allow real-time monitoring for FFF 3D printers (Rankin 2015). Therefore, image analysis systems can be more easily integrated into existing 3D printing environments without significant cost and effort. Additionally, geometric approximation cannot fulfill high flexibility and intelligence requirements to perform defect detection for FFF 3D printing across various standards and industries. Instead, machine learning (ML) algorithms can improve through data accumulation and flexibly adapt to different production environments and standards.

The many strategies that have lately been developed to overcome manufacturing defect detection problems mainly entail applying ML algorithms (Çaydaş and Ekici 2012; Salahshoor, Kordestani, and Khoshro 2010; Ribeiro 2005; Wuest et al. 2016; Zhang et al. 2019). The ability of ML to process a large quantity of data that contain the least amount of sparse knowledge has resulted in it being employed as one of the most commonly used techniques in different manufacturing applications, namely defect detection. Additionally, ML allows the 3D printing process to be monitored in real-time, a capability that could be exploited to correct defects during the fabrication process. Delli and Chang (2018) used an algorithm based on a support vector machine to classify 3D printed parts as either defective or defect-free based on criteria at particular checkpoints. In this investigation, images of the fabricated parts were taken at different stages throughout the process to obtain training data and define the ideal print job. Wu et al. (2016) resorted to ML and image classification to identify malicious infill defects in 3D printed parts. Images of defective and non-defective parts were used as input for a naïve Bayes classifier and J48 decision tree algorithms, for which, respectively, the accuracy reached 85.26% and 95.51% for predicting unseen data. The realization of ML methods can also be extended to other AM techniques to address the defect detection problem. For example, a supervised ML method was developed and implemented to detect defects in a powder bed fusion process (Gobert et al. 2018). The technique obtained 3D CT scans from parts featuring distinctive flaws, e.g., cracks, porosities, and inclusions. Then, the actual location of the defects was transferred into a layer-wise image domain using an affine transformation. Once the binary classifiers were appropriately trained by way of layer-wise imaging, in-situ defect detection of which the accuracy exceeded 80% was reported for cross-validation experiments. The accurate recognition of image features via conventional ML involves a tight intertwinement between effective features learning and feature vector extraction. Despite benefiting from statistical classifiers to locate different features in an image, the performance of ML algorithms is highly sensitive to the features selected by the user. Deep learning algorithms employ artificial neural networks (ANN) structures with large depth, automatically extracting representative features that better characterize the target subject from images (Bengio, Goodfellow, and Courville 2017). Haghighi and Li (2020) built a vision-based filament bonding condition monitoring pipeline that is more than 94% accurate on average. They constructed an ANN model that predicts the geometric deviation of layer thickness based on variables extracted from build parameters and images to support this pipeline. This model learned from 270 images and reached 90.6% and 90.3% R-squared values for the validation and test sets, respectively. Convolutional neural network (CNN) adopts similar structures as ANN and utilized parameter sharing to account for sequential relationships among data points (Bengio, Goodfellow, and Courville 2017). Compared with other ML algorithms, CNN has gained unparalleled popularity in image analysis due to the ability to explore the spatial relationships among pixels (Wang et al. 2018). In research work by Park et al. (2016), a CNN was successfully deployed to inspect surface defects and irregularities of parts manufactured using various techniques. The accuracy for predicting the features of samples that were not included in the set of training samples was 98%. Khan et al. (2021) built a CNN model to classify FFF printed parts with and without defects. A digital camera was mounted on the top of a RAISE3D printer to capture 1695 top view images of the parts. This model achieved an accuracy of 84% after 50 epochs, but both the test accuracy and test loss curves fluctuated significantly, highlighting stability and reproducibility problems. Jin et al. (2021) implemented a semantic segmentation model to locate defect sections of FFF 3D printed parts using the you only look once (YOLO) algorithm based on deep CNN. A total of 1400 images captured by a camera mounted on the extruder were labeled with “good quality,” “over-extrusion,” and “under-extrusion” to train this vision-based defect detection model. This model can simultaneously highlight regions with over-extrusion and under-extrusion on an image and reach an accuracy of 93.9%.

Warping is a typical defect in FFF 3D printed parts, where the residual thermal stresses within a part cause distortion. Many studies have exploited different approaches to analyze and detect warping in 3D printed parts. Most of these schemes are associated with computational and accuracy limitations, but a recent study by Saluja, Xie, and Fayazbakhsh (2020) led to the development of a closed-loop in-process method to detect warping during 3D printing using a CNN classification model. The proposed system enables in-process detection such that once warp distortion is detected, the printing process is aborted. Despite its advantages, image cropping and the selection of hyperparameters were performed manually, which decreased the system's efficiency in terms of the time and computational complexity and limited its practicality.

None of the vision-based autonomous in-process defect detection systems applied to FFF AM to date has been developed with efficient automatic feature extraction. The algorithm developed by Delli and Chang (2018), based on a support vector machine, accepts the entire top view of the part as the input. Straub (2015) designed a quality assessment system that captured and analyzed images of the entire printing process with cameras at different angles. Liu et al. (2019) used a camera to capture images of FFF 3D printed parts. They used these images for textural analysis after manually cropping the region of interest. Baumann and Roller (2016) attached six optical markers to the 3D printer and cropped the area enclosed by them, which helped reduce the background noise and lower the computational burden. They also tried to extract the object's contour based on the hue-saturation-value (HSV) color model, but their approach does not support defect detections with high accuracy. Holzmond and Li (2017) developed a three-dimensional digital image correlation (3D-DIC) system that captured the image of a 3D printed part with two cameras and transferred the images to a point cloud that can be compared with the computer-aided design (CAD) model. This involved extracting features from the original images, which were then converted into a data type that a computer can process. However, the intensive data transformation and simulation were computationally inefficient and challenging to use in real-time. Bisheh, Chang, and Lei (2021) used ML algorithms to segregate pixels of the part from the background, including ANN, support vector machine (SVM), and gradient boosting classifier. Although NN and GBC models are above 90% accurate, the prediction result cannot support precise geometric deviation detection.

ML models contain two types of parameters: hyperparameters and model parameters. Hyperparameters are specified by the user before training and are not learned by the models. Because the performance of the CNN classification model changes considerably with the data type and size, an automated approach is required to select hyperparameters to improve the practicality of the system. Several strategies have been proposed for optimizing the hyperparameter search space (Feurer and Hutter 2019; Wu et al. 2019; Hinz et al. 2018; Murugan 2017). A common optimization algorithm is the grid search method. This method involves training a model over a range of manually selected sub-set of hyperparameters; the values that give the best performance on the validation dataset are chosen. While easier to code, this method is time-consuming, ineffective, and often suffers from the curse of dimensionality. An alternative to this strategy is the random search method, where the hyperparameter search space is randomly sampled. Like the previous method, random search is often time-consuming when dealing with many hyperparameters in the search space. Although an improvement to manual tuning, Grid and random search algorithms are inefficient as they are uninformed by past evaluations; in contrast, a Bayesian-based optimizer keeps track of past evaluations to form a surrogate model. This surrogate model is then used to find the next set of hyperparameters by evaluating the results from the previous iteration. To the best of the author's knowledge, despite being an efficient technique, Bayesian optimization has not been explored extensively to develop AM defect detection architecture.

This paper first describes the proposed automated feature extraction and data collection system, which uses the correlation between the build platform and captured images. Then, the use of Bayesian optimization for the automated selection of hyperparameters for the CNN classification model is explained. Subsequently, the system pipeline constructed by OctoPrint plugins is presented. Finally, the real-time printing experiments are discussed to investigate the accuracy of the closed-loop warping monitoring system by considering different camera angles, corner positions, and corner geometries.

2. Methodology

This section first presents the experimental setup and the information flow of the closed-loop monitoring system. Then, the automated techniques to achieve feature extraction and hyperparameter optimization are introduced. Finally, the logic that guides the OctoPrint plugins to form a closed-loop system for warping monitoring is revealed.

2.1. Experimental Setup

Figure 1 shows the experimental setup of the system for warping monitoring. A Prusa i3 MK2S FFF 3D printer was selected because it represents a wide range of FFF 3D printers with a build platform moving in the Y-direction and an extruder moving in the X- and Z-directions. The monitoring system developed for this printer can be used in conjunction with many other FFF 3D printers. A Raspberry Pi 3 Model B pre-configured with OctoPi is connected to the printer and the camera by two USB cables. It has a Quad-Core ARMv8 processor with 1GB RAM and runs on Linux, and has sufficient computational power for the classification model, G-code analysis, and feature extraction. The training and testing images were captured from sample specimens printed with white and gray polylactic acid (PLA) filaments in the experiments. The reflection effect was mitigated by using blue or green non-reflective painter's tape.

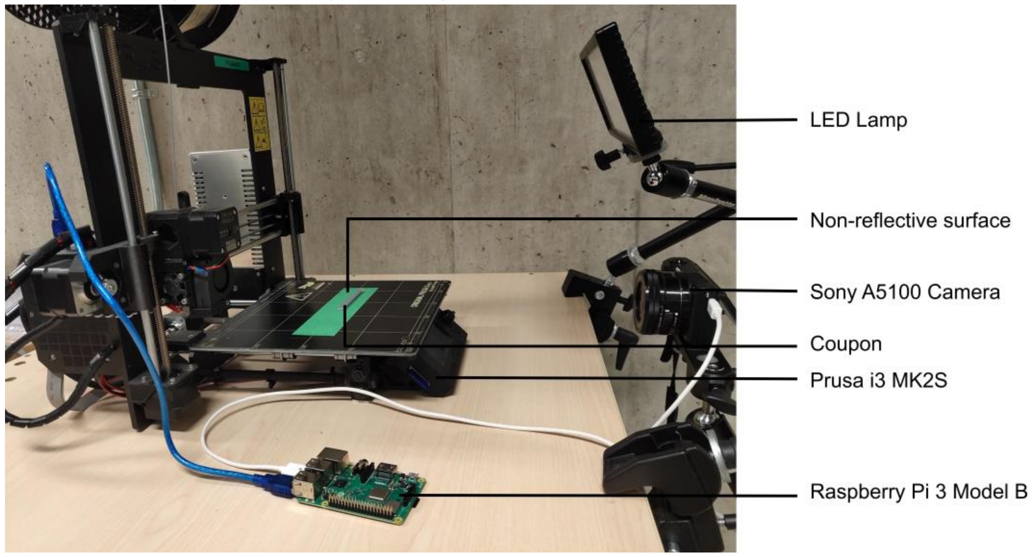


Figure 1. Experimental setup of the closed-loop in-process system for warping monitoring.

2.2. OctoPrint interface

OctoPrint is an open-source web server and an interface for 3D printing written in Python. This interface allows users to download and create plugins to add more functionality and customize the printing operations. Figure 2 illustrates the flow of information between the components of the closed-loop system by depicting the general procedures. The computer first uploads a G-code script to the OctoPrint server. Then, the Raspberry Pi downloads the G-code script via wireless connection and analyzes it with the OctoPrint plugins. In short, the plugins of the system automatically locate the layer changes, retrieve the real-time position of the corner from the G-codes, and then pause the printing process for 4 seconds.

Controlled by the plugins into which gphoto libraries were imported, the camera captures an image when the printer pauses and stores it in the Raspberry Pi. Gphoto is a group of software applications that supports remote image capture and camera configuration adjustment.

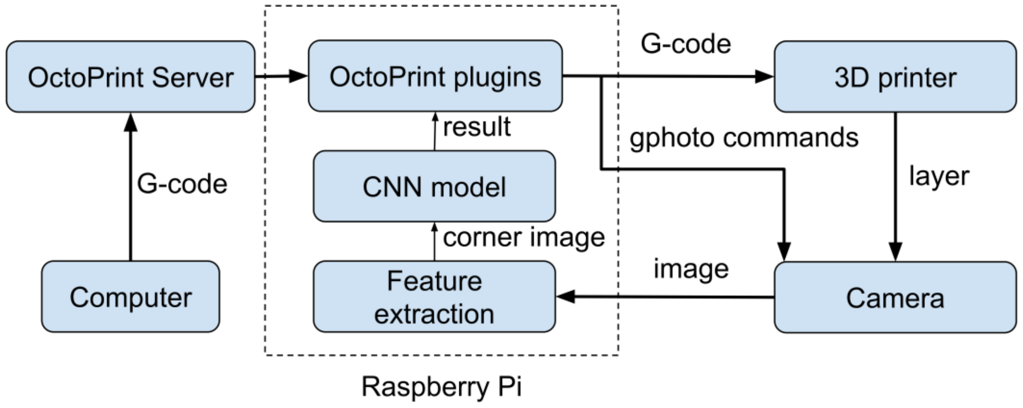


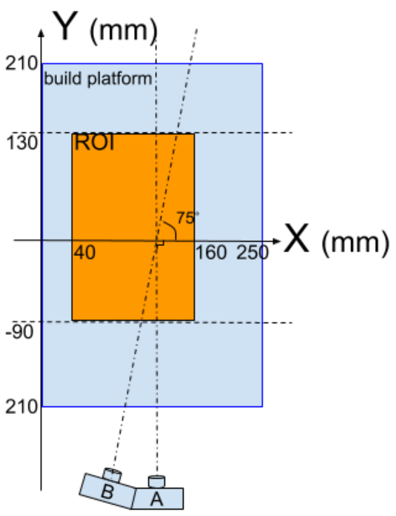
Figure 2. Flow chart of the closed-loop warping detection procedures.

To reduce the computational cost, the image needs to be cropped such that only the corner remains. The cropped image is then used as input for the CNN classification model. Based on the classification result, the OctoPrint plugins decide to either continue or terminate the print, thereby completing the closed-loop system.

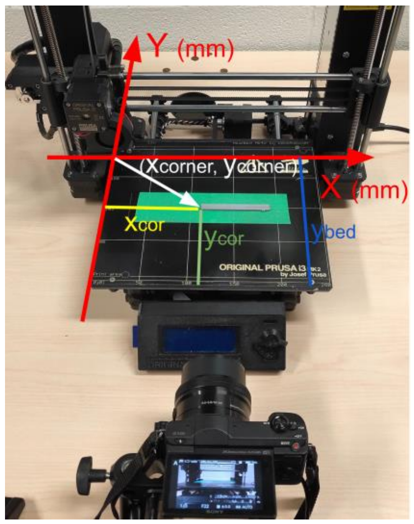
2.3. Automated feature extraction system

A Sony A5100 camera is placed in front of the 3D printer and at the same height as the build platform to capture an image of the side view of a part. The blue region in Figure 3a indicates the area covered when the build platform (250 mm×210 mm) moves from the front-most to the rearmost location along the Y-axis. The orange zone is the region of interest (ROI), where the left corner of the part exists during experiments based on manual inspection of the G-codes. The data collection and correlation process description, presented in Section 2.3.1, focuses on the ROI. In the experiments, the real-time image analysis is primarily performed on the printed part's left corner. To simulate imperfect setup in a production environment, the camera is placed at orientation A, perpendicular to the X-axis, and B, at an angle of 75° with respect to the X-axis. Figure 3b shows the coordinate system of the build platform and certain variables that are utilized to locate the real-time position of the left corner. Further, x_{cor} is the distance between the left corner of the part and the left edge of the build platform, and y_{cor} is the distance between the left corner of the part and the front edge of the build platform; y_{bed} is the distance between the X-axis and the front edge of the build platform; x_{cor}, y_{cor}, and y_{bed} are acquired from the G-codes to analyze the position of the left corner of the part relative to the origin, indicated by the vector, (x_{corner}, y_{corner}).

To capture clear images of the corners within the ROI, the F-number of the camera needs to be increased to obtain the largest depth of field such that the camera remains focused over a wide area of the build platform. Because the brightness is reduced due to less light passing through the lens, an LED lamp is deployed right above the camera to increase the light intensity. The F-number and mounting angle of the LED lamp were carefully adjusted to enable clear images of the corners within the ROI to be captured with no significant flare. The optimal F-number and lamp angle are F18 and 70 degrees, respectively.



(a)



(b)

Figure 3. Build platform coordinate system: (a) region of interest (ROI); and (b) definition of the corner position in the build platform coordinate system.

Figure 4 shows the coordinate system of a captured image, and the desired region to be cropped and used as input into the classification model. The red box indicates the corner to be analyzed, and the feature extraction system generates the cropped image. The feature extraction tool is a correlation system that correlates the corner position in the image coordinate system, x'_{image}, with its position in the build platform coordinate system, x_{corner} and y_{corner}. When the camera is placed at the same height as the build platform, the variation in the vertical position of the part corner on the image (y'_{image}) can be assumed to be insignificant; therefore, it can be considered a constant number.

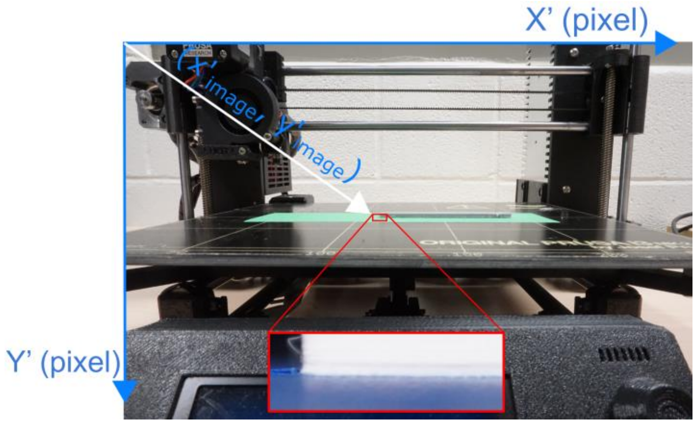


Figure 4. Image captured by the camera and the desired region to be cropped.

Analysis of the G-code commands allows the location of the corner in the build platform coordinate system (x_{corner} and y_{corner}) to be obtained. The location of the corner in the image coordinate system (x'_{image} and y'_{image}) is required to crop the image. The purpose of the correlation is to estimate the location of the corner on the image using its location on the build platform. Therefore, map matching is performed between the two coordinate systems.

2.3.1. Automated data collection

Two 3D printed parts were created to aid the development of the automated map-matching process between the two coordinate systems (Figure 5). The upper part, which contains a rectangular notch, is mounted on the extruder block using the circular hole. The lower part, which is mobile and is placed on the build platform, has a rectangular block on top that can slide into the upper part and a triangular platform pointing in the direction of the extruder. The tip of the triangular platform, to which a red marker is attached, is positioned directly beneath the extruder head, and represents the location of the reference point. The triangular platform has a lower height than the block to leave sufficient clearance between the extruder head and the part. When the rectangular block of the lower part slides into the notch in the upper part, the lower part moves with the extruder, and when they are detached, the lower part moves with the build platform. A G-code script is prepared to guide the lower part to each reference point automatically. The OctoPrint plugins autonomously capture an image when the lower block arrives at each point. After the data collection process, the two parts are detached from the 3D printer.

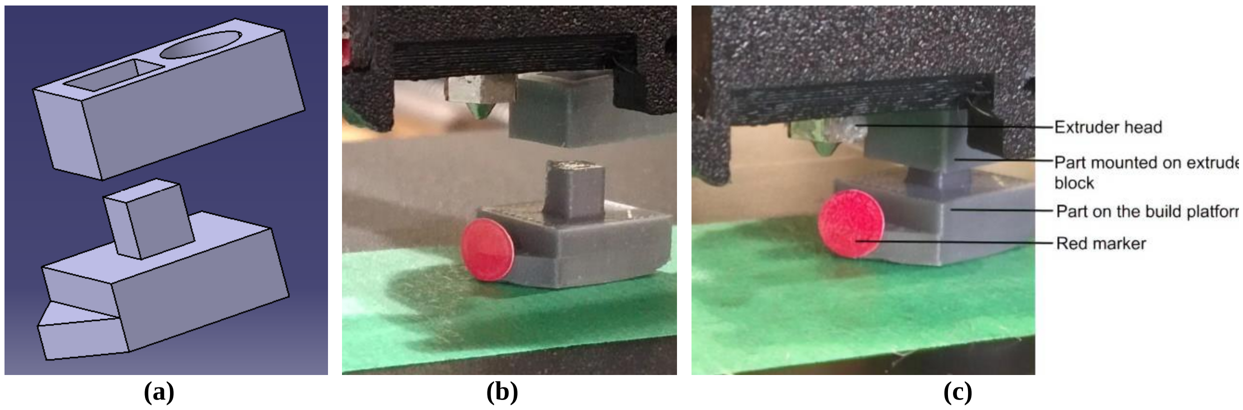
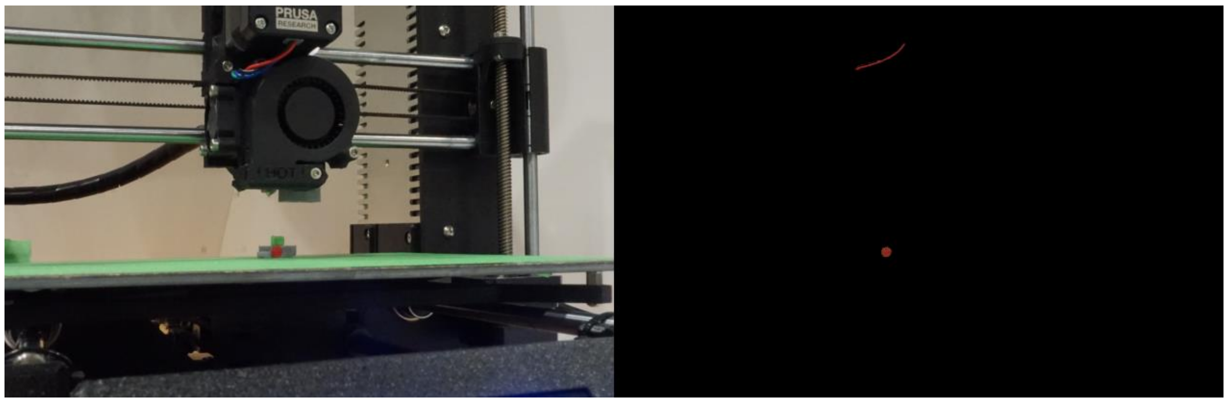


Figure 5. Tools aiding automatic correlation: (a) CAD models; the two parts (b) detached from and (c) attached to each other.

Color recognition technology is used to determine the location of the red marker. The Red-Green-Blue (RGB) color model is utilized to define the threshold for "red" (Table 1). The original image that was captured, shown in Figure 6a, is filtered using a color mask such that only the pixels with an RGB color difference between the upper and lower bounds are retained. As shown in Figure 6b, the pixels of the red marker are kept along with a red stripe at the top, which is the red wire from the 3D printing head. The red pixel represents the corner position on the image with a known location on the build platform (x_{corner} and y_{corner}). Its horizontal location on the image, x'_{image}, is defined by the horizontal location of that pixel. The procedure described above can be repeated for the 112 reference points on the build platform and the correlation between the two systems can be obtained.

Table 1. Thresholds defined by setting upper and lower bounds for red.

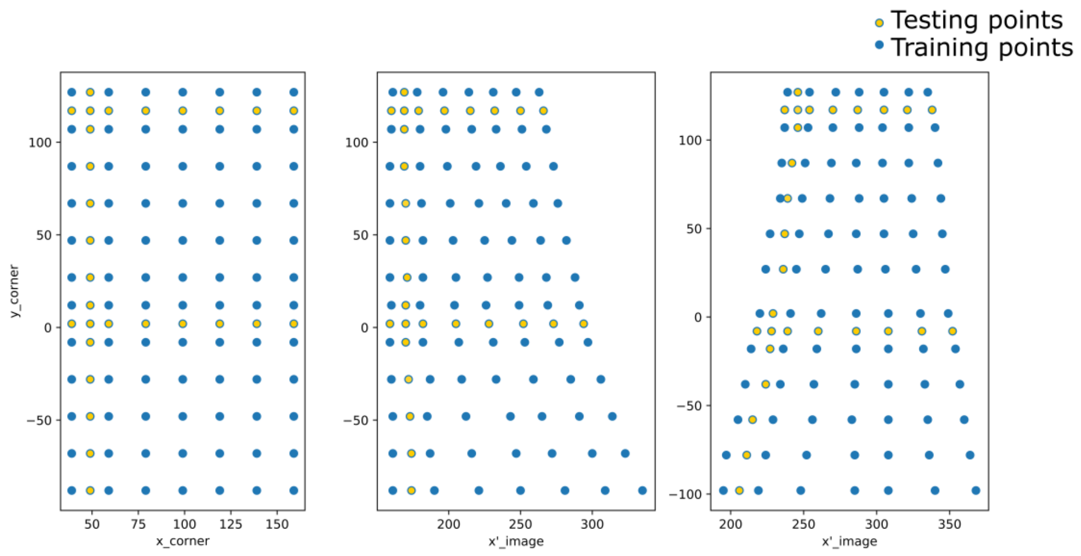
	R	G	B
Lower bound	17	15	100
Upper bound	50	56	200



(a) (b)

Figure 6. Red marker used to define the position of the reference point on the image: (a) original image; and (b) image acquired using the color mask for red.

As shown in Figure 7a, 112 reference points are created within the ROI. The reference points are equally spaced except for two rows inserted at y_{corner}=0 and y_{corner}=120, and one column added at x_{corner}=50. These three lines of reference points marked with yellow dots are not used to create the interpolation model; instead, they are treated as testing points to validate the correlation. Figures 7b and c show the position of the reference points in camera views of 75° and 90°, respectively, where the x'$_{image}$ value of each reference point is correlated to a pair of x_{corner} and y_{corner}. They indicate that the images are skewed in the camera view due to the lens's radial distortion. The relation between the position of any point within the ROI and its location on the image can be determined using interpolation. Because the images are distorted, linear interpolation cannot be used, and spline interpolation is chosen.



(a) (b) (c)

Figure 7. Actual and distorted positions of the reference points: (a) actual reference points; reference points viewed by the camera at (b) 75˚ and (c) 90˚.

2.3.2. Bivariate Spline Interpolation

Spline interpolation is a numerical approximation method that uses a piecewise polynomial named spline. Taking three points A(x_{i-1}, y_{i-1}), B(x_i, y_i), and C(x_{i+1}, y_{i+1}), where $x_{i-1} < x_i < x_{i+1}$, two polynomial curves, described by functions L_1 and L_2, were created to connect them smoothly. Point B is the intercept of L_1 and L_2.

$$L_1(x_{i-1}) = y_{i-1} \tag{1}$$
$$L_1(x_i) = y_i \tag{2}$$
$$L_2(x_i) = y_i \tag{3}$$
$$L_2(x_{i+1}) = y_{i+1} \tag{4}$$

The classical approach involves the use of a cubic spline (Ahlberg, Nilson, and Walsh 1967), which should be smooth at the joint such that the two curves have the same slope and concavity at point B:

$$L_1'(x_i) = L_2'(x_i) \tag{5}$$
$$L_1''(x_i) = L_2''(x_i) \tag{6}$$

It is necessary to assume boundary conditions to solve the equations of the two curves. Natural spline boundary conditions are commonly adopted to generate a natural and smooth interpolation, where the second derivative of the two ends A and B are equal to zero:

$$L_1''(x_{i-1}) = 0 \tag{7}$$
$$L_2''(x_{i+1}) = 0 \tag{8}$$

Thus, two cubic polynomial curves can be acquired to form a univariate spline. In this correlation, two input variables, x_{corner} and y_{corner}, are used. To obtain an interpolation result with two variables, a smoothing bivariate spline approximation tool from SciPy is introduced to the correlation procedure. SciPy is an open-source software tool written in Python to solve various mathematical problems. Additionally, the degree of the bivariate smoothing spline was set to 3 (cubic spline for both variables) in the experiments.

The x'_{image} positions of the testing points are compared with the interpolated results to validate the interpolation result. The mean absolute error (MAE) is used in the validation, where a small MAE value indicates an accurate interpolation model.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left(\left|x'_{image,j} - \hat{x}'_{image,j}\right|\right) \quad , \tag{9}$$

where n is the number of testing points (n=28), x'_{image} is the dataset containing the real coordinates of the testing points, and $\hat{x}'_{image}$ is the dataset containing the imputed coordinates of the testing points with bivariate spline interpolation. MAE is a statistical measure that indicates the error between the observed and predicted values. In the correlation, this value physically represents the average difference in pixels between the observed and interpolated values of x'_{image} of the testing points.

2.4. Automated CNN construction

This section compares the automated approach, based on the construction of a CNN, with the manual approach the authors previously reported (Saluja, Xie, and Fayazbakhsh 2020). The input dataset, hyperparameter optimization, and optimized CNN architecture are described.

2.4.1. Dataset Preparation

Fifty-two 30 mm $\times$15mm $\times$ 5mm cuboids were printed (individually) and recorded layer-by-layer to collect images of unwarped corners. The corners of these cuboids were then peeled from the build platform to imitate warp deformation. CNN models are sensitive to the quality of input images. For example, features are less prominent in darker images and can have an adverse effect on classification accuracy. This can be addressed by training the underlying model with images captured under different ambient lighting conditions. The dataset proposed for this study consisted of 550 color images (6000 $\times$ 4000 pixel) divided (equally) between two classes, warped and unwarped. In each class, the first half contained images taken in an ideal environment, i.e., the lighting conditions were closely monitored to ensure clear images. The other half consisted of images captured by altering the ambient conditions (including the lighting conditions and camera positions) to obtain grainy images.

Training a classification algorithm with this dataset would require optimizing 3.96×10^{10} nodes ($550 \times 6000 \times 4000 \times 3$), a computationally expensive process. To improve the computational efficiency, the region containing the corner (Figure 8) was extracted using the feature extraction method described in Section 2.3. The figures were converted to grayscale and resized to 100×100 pixels by performing nearest-neighbor interpolation. The processed dataset was then shuffled and split into training and validation sets in the ratio 80/10, and the remaining 10% was used for testing. Figure 8 shows three training samples for each class from both datasets under different ambient lighting conditions.

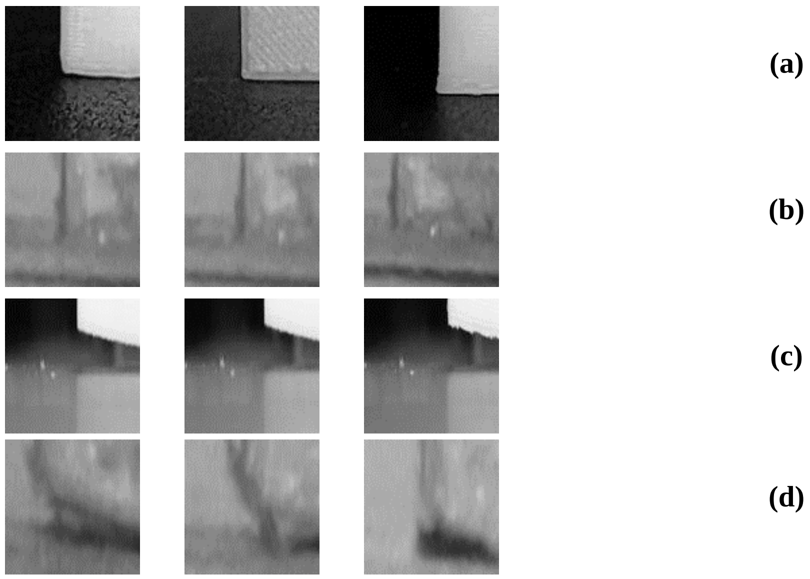


Figure 8. Samples from the training set: unwarped samples taken (a) in an ideal lighting environment, (b) under varying lighting conditions; warped samples taken (c) in an ideal lighting environment, (d) under varying lighting conditions.

2.4.2. Automated Hyperparameter Optimization

A CNN can typically be constructed manually or autonomously. Although easier to code, the former necessitates a trial-and-error approach and is often time-consuming and inaccurate. In contrast, an automated approach is suitable for an agile manufacturing environment; however, it is computationally expensive and requires a good understanding of black-box optimization techniques. The manual approach was covered extensively by Saluja, Xie, and Fayazbakhsh (2020). Since then, a Bayesian-based probabilistic model was developed to automate optimizing the hyperparameters for the underlying CNN classification model. Figure 9 outlines the high-level algorithm utilized to optimize the hyperparameter values.

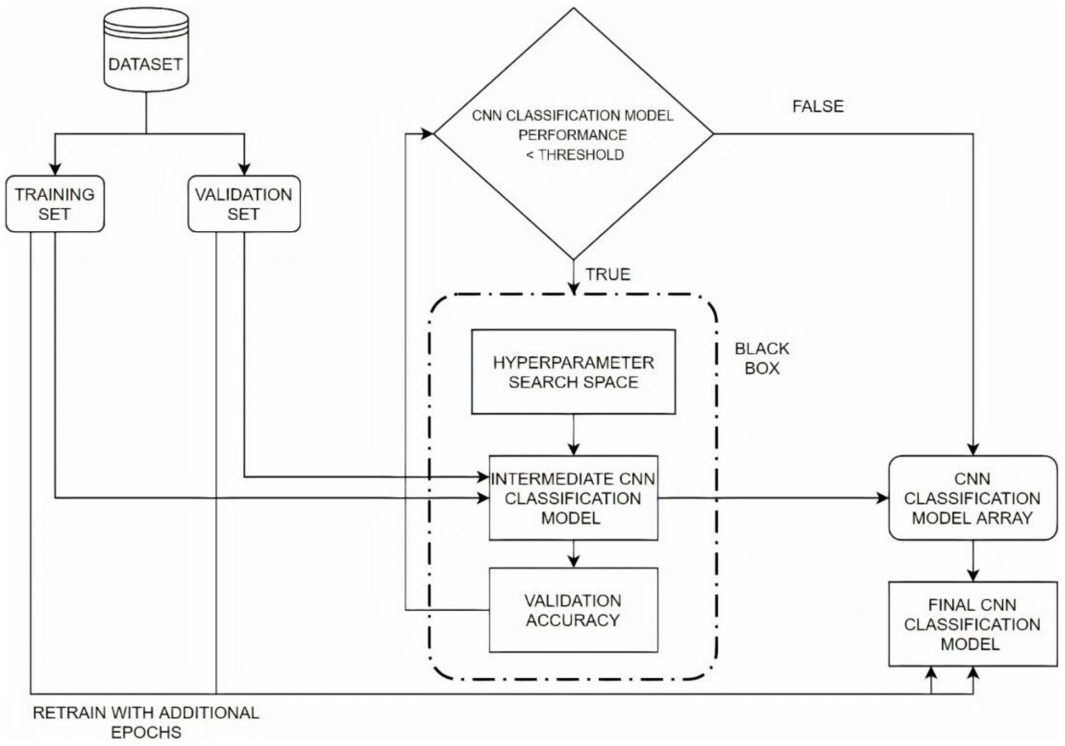

DATASET
TRAINING SET
VALIDATION SET
CNN CLASSIFICATION MODEL PERFORMANCE < THRESHOLD
FALSE
TRUE
BLACK BOX
HYPERPARAMETER SEARCH SPACE
INTERMEDIATE CNN CLASSIFICATION MODEL
VALIDATION ACCURACY
CNN CLASSIFICATION MODEL ARRAY
FINAL CNN CLASSIFICATION MODEL
RETRAIN WITH ADDITIONAL EPOCHS

Figure 9. Simplified representation of the algorithm underlying the automated approach.

The algorithm divides the dataset into training and validation sets and then repeats an optimization loop until a certain threshold is met. Hyperparameters are selected from a search space defined by the user during each iterative cycle. The search space contains the total number of convolutional and pooling layers (including the kernel size and stride length), the number feature maps, the neurons and activation function for each layer, and the optimal dropout and learning rate. The intermediate classification model is then created and trained on the training set using the selected hyperparameters. This model is evaluated on the validation set wherein the validation accuracy was stored after each iterative cycle to estimate the values of hyperparameters for the subsequent iteration. The intermediate classification model and its validation accuracy are stored in an array. The best model is then selected and retrained on additional epochs to improve the validation accuracy. The number of epochs is limited by implementing early termination to prevent the model from training once the specified threshold for validation accuracy is reached.

The manual approach implemented by Saluja, Xie, and Fayazbakhsh (2020) performed well, yielding a mean validation and training accuracy of 0.98; however, it took more than three hours to optimize and train the model. In contrast, the automated approach constructed and retrained the model with additional epochs in less than 15 minutes. The architecture of the optimized CNN classification model is presented in Table 2.

Table 2. Serialized architecture of the underlying CNN classification model

Layer	Operator	Kernel Size	Stride	Number of Filters/Nodes/%	Method/Activation
LY1 – C1	Convolution	2 × 2	2	18	ReLU
LY2 – P1	Pooling	2 × 2	2	18	Max Pooling
LY3 – C2	Convolution	2 × 2	2	14	ReLU

LY4 – C3	Convolution	2 × 2	2	26	ReLU
LY5 – C4	Convolution	2 × 2	2	16	ReLU
LY6 – F1	Fully Connected	Flattened to a Vector			
LY7 – DP1	Dropout	80% of Nodes Retained			
LY8 – FC2	Fully Connected	-	-	2	SoftMax

2.5. OctoPrint Plugins

OctoPrint plugins are the skeleton of the system and are responsible for multiple tasks, such as G-code analysis, feature extraction, and the CNN classification model. (Figure 10). The G-code script of a part is downloaded to the Raspberry Pi 3 once the user activates the print job on the OctoPrint website. The G-codes are first queued in the Raspberry Pi before they are sent to the printer to guide the movements of the extruder and the build platform. The 3D printer receives five G-code commands from the Raspberry Pi at each turn and requests another five commands when the previous commands are being executed. The OctoPrint plugins allow the G-code commands that are in the queue to be edited. G-code commands can help locate the coordinate of the left corner of the part (x_{corner}, y_{corner}) in the build platform coordinate system during the printing process. Commands "G0" and "G1" are rapid and controlled motions, respectively. "X" and "Y" usually follow these two commands to indicate where the extruder should move on the build platform. For example, the command in Eq. 10 mandates the extruder to rapidly move to the point on the build platform where it is 20 mm and 40 mm away from the left and front edges, respectively.

$$G0\ X20\ Y40 \tag{10}$$

The left corner of the part on the build platform (x_{cor} and y_{cor} shown in Figure 3b) can be located by analyzing the G-codes with OctoPrint plugins to find the track of the extruder, shown in Algorithm 1.

Algorithm 1: Pseudocode to find x_{cor} and y_{cor} to locate the position of the left corner of the part on the build platform.

```
Input: G-code commands in sequential order
FOR each command in G-code queue
        IF contains 'G0' OR contains "G1" THEN
                append numbers following X and Y to arrays $x_{coordinate}$ and $y_{coordinate}$, respectively
        END IF
END FOR
$x_{cor}$, minIndex = min($x_{coordinate}$)
$y_{cor}$ = min($y_{coordinate}$[minIndex])
return ($x_{cor}$, $y_{cor}$)
```

This way, the system can detect the corner coordinate of multiple geometries such as rectangular, rounded, and triangular corners. Additionally, the position of the build platform must be acquired to find the real-time location of the corner when an image is captured. G-code command can also provide the real-time position of the build platform, y_{bed}. Once a layer change is detected, the plugins insert a pause command (G4) to pause the print before the layer change. While the printer is paused, the location of the build platform y_{bed} is retrieved from the G-codes and used to compute the real-time coordinate of the corner. The input to the correlation (Section 2.3.1) can be obtained as:

$$x_{corner} = x_{cor} \tag{11}$$

$$y_{corner} = y_{cor} - y_{bed} \quad (12)$$

Then, the plugins send a command to the camera to capture an image when the printer pauses. Based on x_{corner} and y_{corner}, the correlation model provides x'_{image}, allowing the algorithm to locate and crop the image. The cropped image only contains the important features, which is the corner here. Afterward, the cropped image is sent to the CNN classification model for image analysis, and the output is the probability of an unwarped corner. A threshold of 0.3 is chosen, which means if the result is less than 0.3, the corner is considered a warped corner. The threshold was set to 0.3 instead of 0.5 because analysis of the classification results of the training and validating dataset indicated that a warped corner usually has a result of less than 0.01. Therefore, the threshold was adjusted to a value smaller than 0.5. The Raspberry Pi either continues or aborts the print depending on the classification result.

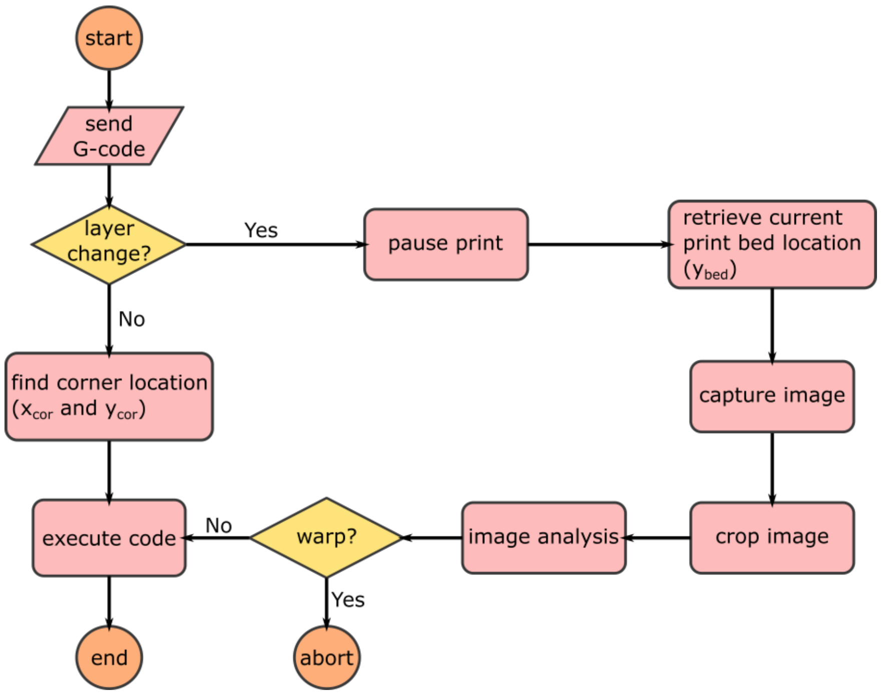


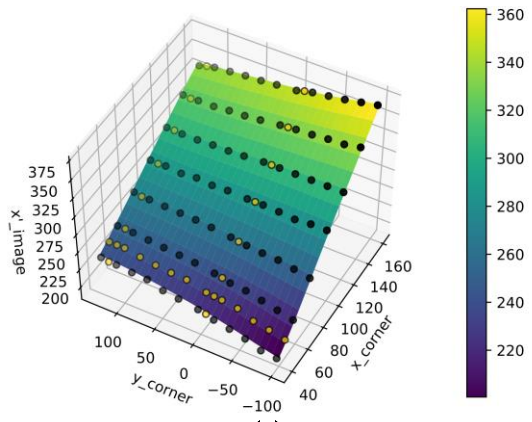
Figure 10. The pipeline of the OctoPrint plugins.

3. Results and Discussion

This section presents the correlation results of the coordinates of the image-build platform, hyperparameter optimization, and the closed-loop in-process system for warping monitoring.

3.1. Correlation Results

Figure 11 shows the correlation surface between the position of any point (x_{corner} and y_{corner}) and its location in the image (x'_{image}) for camera views of 75° and 90°. The figure also includes the original reference points used for the creation and testing of the bivariate spline interpolation. As discussed in Section 2.3.2, the mean absolute error was used to evaluate the accuracy of the interpolation results. The MAEs for the testing points (yellow dots) were 0.65 and 0.59 for camera views of 90° and 75°, respectively. The MAE values show that the average difference in pixels between the actual values and the correlation results for the testing points is acceptable. The cropped image of 100×100 pixels successfully retains the important features of the corner of the parts with these small shifts from the center. Therefore, the correlation surface can estimate the location of any point on the build platform in the image captured by the camera. This information is used to automatically crop images captured during a part 3D printing to retain only the corner of the part.



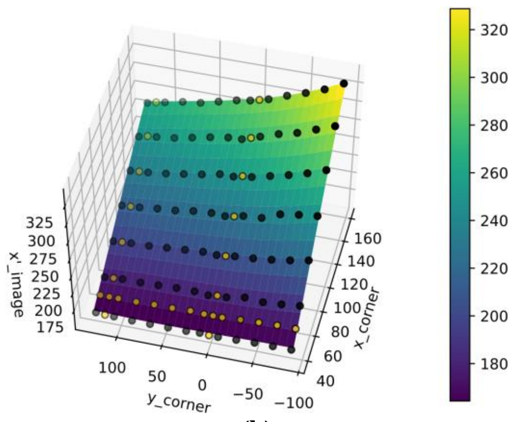


Figure 11. Correlation surfaces and scatter plots of original reference points: (a) 90° and (b) 75°.

3.2. Closed-loop System for Warping Monitoring

As discussed in Sections 2.1 and 2.2, the closed-loop in-process system for warping monitoring employs CNNs for automated feature extraction and hyperparameter optimization. The test started with rectangular specimens with a length of 80 mm, a width of 8 mm, and a height of 3 mm. The manufacturing and design parameters are listed in Table 3.

Table 3. 3D printing process parameters for the tests.

Manufacturing parameter	Value	Manufacturing parameter	Value
Print direction	XYZ	Nozzle diameter (mm)	0.4
Material	PLA	Nozzle temperature (°C)	200
Raster angle	0	Cooling	No fan cooling
Layer height (mm)	0.15	Infill (%)	30
Bed temperature (°C)	60	Filament diameter (mm)	1.75
Print speed (mm/min)	2400	Number of layers	20

The system first allows the print to be built up to approximately 1 mm. Upon completing the sixth layer, the first image is captured and analyzed. Based on the result from the classification model, the OctoPrint plugins either continue (Probability, $P(X = \text{unwarped corner}) \geq 0.3$) or stop ($P(X = \text{unwarped corner}) < 0.3$) the printing process. A test was conducted by positioning a rectangular specimen at the top left of the build platform. A camera angle of 75° was selected to demonstrate the system's effectiveness. The results are summarized in Table 4 with the accompanying plots in Figure 12. The first 13 layers that were analyzed (layer #6 to #18) were classified as corners without warpage because the results were greater than 0.3. The 19th layer resulted in 0.000324; thus, it was categorized as a warped corner.

Table 4. Warping detection results for a specimen printed in the top left corner of the build platform and a camera angle of 75°.

Automated Approach	
Layer #	**P (X = unwarped corner)**
6	0.689
7	0.525
8	0.892
9	0.765
10	0.814
11	0.803
12	0.732
13	0.665
14	0.844
15	0.753
16	0.790
17	0.709
18	0.690
19	0.000324

The classification results in Table 4 correspond precisely with the actual print shown in Figure 12. Once the warpage was detected, the Raspberry Pi aborted the print. This test demonstrated that the closed-loop system for warping monitoring operates as intended.

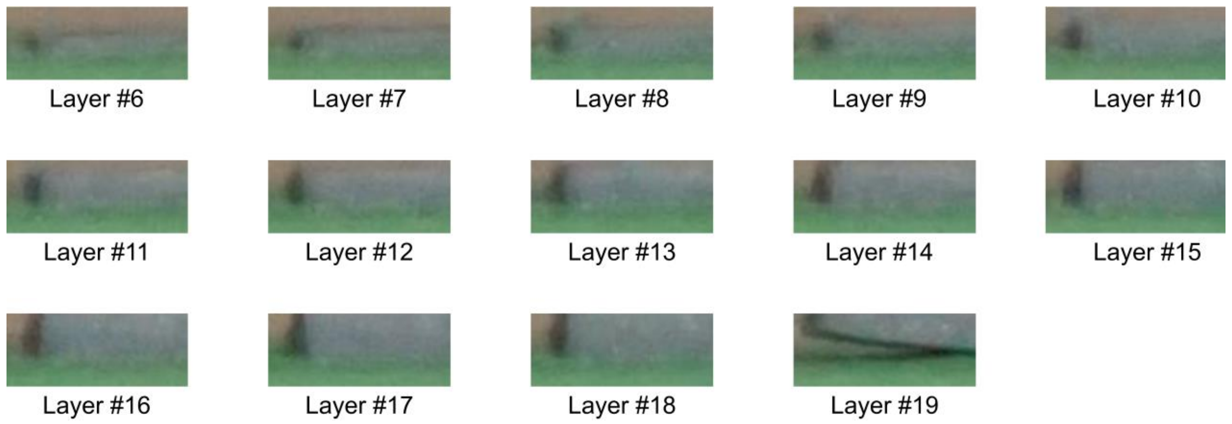


Figure 12. Cropped images of up-left position and camera at 75 degrees from layer #6 to #19.

To evaluate the accuracy of the system, sixteen tests were performed that included different camera views, part corners placed at various locations on the build platform, and corners with different geometries. Initially, the camera was set perpendicular to the X-axis (orientation A). The rectangular specimens were printed in different locations on the build platform (Figure 13a): upper left, center, and lower right. The 3D printing was repeated for each location, yielding six specimens. Then, the camera was placed at 75° with respect to the X-axis (orientation B) and another six rectangular specimens were printed. Finally, the camera was restored to orientation A to investigate the impact of different corner geometries on the accuracy of the system. Rectangular specimens with rounded corners with a radius of 4 mm and sharp corners with a height of 15 mm were investigated (Figure 13b).

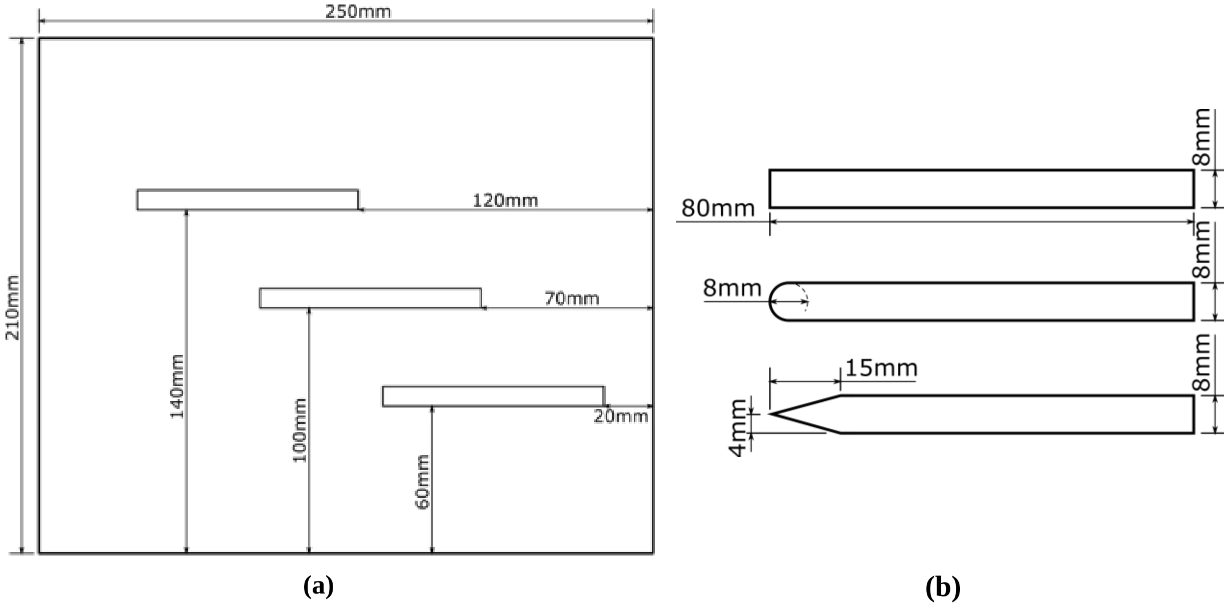


Figure 13. Specimens for testing the system for warping monitoring: (a) different specimen positions; and (b) different corner geometries.

A total of 16 specimens were printed, and 128 images were captured, cropped, and analyzed by the closed-loop system for warping monitoring (Table 5). Altogether 127 images were correctly classified, resulting in an accuracy of 99.2%. In addition, all the cropped images contained the specimen corners, indicating an image cropping success of 100%. The results also showed that the system worked effectively for both camera angles, all three positions, and all three corner geometries with high accuracy. The only image that was not correctly identified was that from Test 8 in which the corner warped in layer #6. However, the system did not recognize the warpage until layer #7 and the print was terminated at layer #7 instead of layer #6.

To investigate the reasons behind the incorrectly classified image, all images of warped corners at the center of the build platform acquired from the experiments are shown in Figure 14. Figure 14(c) was incorrectly recognized as a corner without warping, while other images were correctly classified as warped corners. At the 6th layer of Test 8, the part has marginally detached from the build platform, with minimal shadow and curvature beneath the part. After one more layer has been deposited (Figure 14(d)), thermal stress further built up and pulled the part away from the build platform, resulting in the clearer and thicker shadow under the part. Similarly, other images in Figure 14 also present obvious curvature and shadow. According to Saluja, Xie, and Fayazbakhsh (2020), features such as curvature and shadow extracted by CNN filters are essential evidence of warping. Nonetheless, images characterizing marginal warping like Figure 14(c) are not present in the training set, where most warped corners are away from the build platform. Thus, Figure 14(c) lands at a region where the training example is sparse and accurate prediction cannot be obtained with the current training set.

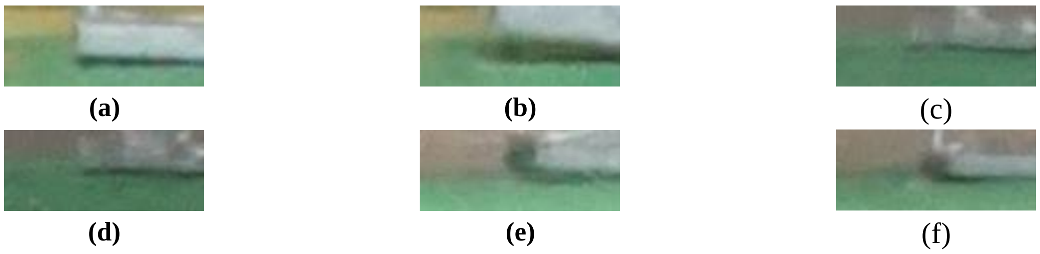


Figure 14. Images of warped corners at the center of the build platform: (a) Test 3 at layer #15; (b) Test 4 at layer #8; (c) Test 8 at layer #6 (incorrectly classified); (d) Test 8 at layer #7; (e) Test 13 at layer #11; and (f) Test 14 at layer #10.

Table 5. CNN classification results for the tests.

Camera angle	Specimen position	Test number	Total number of analyzed layers	Number of unwarped layers	Number of warped layers	Layers correctly classified	Corners correctly cropped
90 degrees	Downright	1	11	10	1	11	11
		2	9	8	1	9	9
	Center	3	10	9	1	10	10
		4	3	2	1	3	3
	Top-left	5	13	13	0	13	13
		6	5	4	1	5	5
	Triangular corner	7	14	14	0	14	14
		8	2	0	2	1	2
	Rounded corner	9	11	10	1	11	11
		10	4	3	1	4	4
75 degrees	Downright	11	5	4	1	5	5
		12	8	7	1	8	8
	Center	13	6	5	1	6	6
		14	5	4	1	5	5
	Top-left	15	14	13	1	14	14
		16	7	6	1	7	7

As shown in Figure 12, the cropped images of the corners had a blurry appearance. This was attributed to the fact that the system captured an image of the corner at which the part being printed was paused, but the build platform was not delivered close to the camera, making an optimized focus distance unachievable. Nonetheless, the system was still highly accurate despite the blurriness of the input images. Compared with the original CNN classification model (Saluja, Xie, and Fayazbakhsh 2020), the current version of the model is shallower in that it contains eight hidden layers instead of 21. Hence, the automated approach utilizing Bayesian optimization reduces the time required to train the model from approximately 3 hours to 15 minutes, and the in-process execution time per layer from 50 seconds to 20 seconds with Raspberry Pi

3. Additionally, the number of feature maps in each layer was increased, allowing the model to learn several low-level features from the input image while retaining its computational efficiency. Furthermore, by downsampling the input image from 525 × 100 pixels to 100 × 100 pixels, the current version of the model was invariant to shift and distortion, as is evident from the results. Theoretically, deeper networks are more effective at abstracting features, resulting in a more comprehensive understanding of the input image. However, the redundant pooling layers and lack of kernels prevented the model developed by using the manual approach from learning low-level features, as indicated by the results. The new approach yielded a much shallower yet efficient model in less than 15 minutes, significantly reducing the time required to train the model and the in-process execution time.

The initial assumption that y_{image} can be constant was valid as the corners were precisely cropped for all images with a constant value assigned to y_{image}. Using the camera for a high- or low-angle shot, in which case y_{image} cannot be treated as a constant, would enable the same correlation methodology described in Section 2.3 to be used to estimate its value. Moreover, this technique is not restricted to 3D printing and could be applied to other manufacturing processes. For example, the automated feature extraction system could be used in CNC machining to extract the important regions of an image using the proposed cropping technique since G-codes also run CNC machines.

By detecting the warpage and stopping the manufacturing process, this real-time monitoring system can significantly reduce materials wastage, and prevent any damage to the nozzle and the 3D printing head. Although this system is precise, it still requires more enhancements to be implemented in a practical production environment. Currently deployed on a Raspberry Pi, the system might run into scalability issues when more defect detection features are added.

3.3. Computational complexity analysis

This section aims to illustrate the enhancements brought by Bayesian optimization to the computational efficiency of the FFF warping detection model. Compared with the model presented in Saluja, Xie, and Fayazbakhsh (2020), where the hyperparameters were determined by grid search, Bayesian optimization helps reduce both the hyperparameter optimization time and the time complexity of the model. The time complexities of the forward pass of the old and new CNN models are approximated and compared with another CNN model for geometric unconformity detection presented by Khan et al. (2021). Note that the shapes of the input images and kernels are squares that have equal length and width. According to He and Sun (2015), the time complexity of a convolutional layer is:

$$T_{conv} = O(n_{in} k^2 n_{out} m_{out}^2) \,, \tag{13}$$

where k is the width of the kernel, m_{out} is the width of the output feature map, n_{in} and n_{out} are the numbers of channels of the input layer and output feature map. Both the time complexities of max pooling and ReLU activation can be approximated by:

$$T_{pooling/activation} = O(n_{out} m_{out}^2) \tag{14}$$

Freire et al. (2021) approximate the complexity of dense layers as:

$$T_{dense} = O(l_{in} l_{out}) \,, \tag{15}$$

where l_{in} and l_{out} are the numbers of neurons in the last and current layers, respectively. The representations above are simplified, but often provide satisfactory accuracy for comparison purposes. Table 6 compares the forward pass time complexities of reported algorithms that detect warping. Although the algorithm built by Khan et al. (2021) aims to classify the overall geometric conformity of a part without pointing out the defect types, it can also be utilized to detect warping. It can be observed that the new model with hyperparameter optimization substantially reduces the time complexity of CNN-based warping detection. Bayesian optimization helps construct the model that can achieve optimal accuracy with less depth and number of filters than the previous model. The model presented by Yi et al. (2017) evaluates geometric unconformity by performing a pixel-by-pixel comparison between the current print and a standard one. Petsiuk and Pearce (2020) proposed an open-source defect detection framework for FFF that utilizes both geometric approximation and machine learning algorithms to detect various defects. Although warping detection is not included in this framework, this function can be compared to its side view height validation algorithm. Arguably, a much smaller time complexity can be obtained with geometric approximation than CNN models. Nonetheless, CNNs are more flexible and intelligent to deal with numerous challenges concerning defect detection of additive manufacturing. For instance, varying light conditions seriously compromise the performance of image-based geometric approximation algorithms (Petsiuk and Pearce 2020; Yi et al. 2017). The new CNN model has been developed with training examples captured under different lighting conditions to enable warping detection at different corner positions and camera angles. Further, the input image size has been reduced to n=100 such that the complexity difference between $O(46n^2)$ and $O(n^2)$ is not significant regarding modern GPU and CPU.

Table 6. Comparison of the forward pass time complexities among reported algorithms that can detect warping for FFF 3D printing.

Algorithm	**#1**	**#2**	**#3**	**#4**	**#5**
Source	This paper	Saluja, Xie, and Fayazbakhsh (2020)	Khan et al. (2021)	Yi et al. (2017)	Petsiuk and Pearce (2020)
Type	CNN	CNN	CNN	Geometric approximation	Geometric approximation
Time complexity	$O(46n^2)$	$O(288n^2)$	$O(3480n^2)$	$O(n^2)$	$O(n^2)$

4. Conclusions

This study led to the design of a closed-loop in-process system for warping monitoring for FFF 3D printing. The system comprises a 3D printer, camera, and microcomputer operating in coordination. A feature extraction functionality was created to autonomously locate and extract the corner from the original image, which mitigated the environmental noise and reduced computational complexity. An automated correlation process was designed to perform map matching between the build platform and the image coordinate systems to aid high-precision feature extraction. Bayesian optimization was adopted to automatedly select the optimized hyperparameters for the CNN classification model based on the training and testing datasets. Finally, the OctoPrint plugins served as an interface between the software and hardware, performing real-time G-code analysis and information transmission. A total of 16 specimens were printed, and 128 images were captured, cropped, and analyzed. All the cropped images contained the specimen corners, and 127 images were correctly classified, indicating that the system is 99.2% accurate.

The dataset for this study was relatively small and allowed the models to be trained in less than 15 minutes. However, in an industrial environment, datasets are complex and require a much longer training time for an automated approach. The increasing availability of hardware accelerators means the computational costs have become inconsequential compared to the time saved and the accuracy achieved using black-box optimization algorithms. The automated feature extraction methodology can be flexibly applied to a system that operates at different camera angles. Thus, the system potentially allows cameras to be deployed around the build platform to inspect more types of defects. Moreover, this technique is not restricted to 3D printing and can monitor other manufacturing techniques, e.g., CNC machining.

In the future, efforts will be made to address random variations in images' brightness or color information (image noise). In addition, the deployment of the defect detection architecture in a real production system will be investigated. The focus will be on implementing software development and information technology operations (DevOps) principles to ML, especially on the continuous delivery and automation of ML pipelines. At a higher level, to improve the current system's scalability and account for incoming data streams, the system would be hosted on a cloud-based service accessible via application programming interface (API) requests. Additionally, workflows will be set up to identify any bottlenecks or root causes to improve the generalizability and performance of the model.

Authors Contributions

Jiarui Xie: Methodology, Software, Writing – Original Draft. **Aditya Saluja**: Methodology, Software, Writing – Original Draft. **Amirmohammad Rahimizadeh**: Writing – Original Draft, Writing – Review & Editing. **Kazem Fayazbakhsh**: Resources, Writing - review & editing, Supervision, Project administration.

Declaration of Competing Interest

The authors declare that there are no conflicts of interest.

Data availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Funding

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN-2018-04144 and Fall/Winter 2019-2020 Career Boost International by Ryerson University.

References

Ahlberg, J Harold, Edwin Norman Nilson, and Joseph Leonard Walsh. 1967. "The theory of splines and their applications." *Mathematics in science and engineering*.

Baumann, Felix, and Dieter Roller. 2016. "Vision based error detection for 3D printing processes." MATEC web of conferences.

Bengio, Yoshua, Ian Goodfellow, and Aaron Courville. 2017. *Deep learning*. Vol. 1: MIT press Massachusetts, USA:.

Bisheh, Mohammad Najjartabar, Shing I Chang, and Shuting Lei. 2021. "A layer-by-layer quality monitoring framework for 3D printing." *Computers & Industrial Engineering* 157:107314.

Çaydaş, Ulaş, and Sami Ekici. 2012. "Support vector machines models for surface roughness prediction in CNC turning of AISI 304 austenitic stainless steel." *Journal of intelligent Manufacturing* 23 (3):639-650.

Delli, Ugandhar, and Shing Chang. 2018. "Automated process monitoring in 3D printing using supervised machine learning." *Procedia Manufacturing* 26:865-870.

Feurer, Matthias, and Frank Hutter. 2019. "Hyperparameter optimization." In *Automated machine learning*, 3-33. Springer, Cham.

Frazier, William E. 2014. "Metal additive manufacturing: a review." *Journal of Materials Engineering and performance* 23 (6):1917-1928.

Freire, Pedro J, Yevhenii Osadchuk, Bernhard Spinnler, Antonio Napoli, Wolfgang Schairer, Nelson Costa, Jaroslaw E Prilepsky, and Sergei K Turitsyn. 2021. "Performance versus complexity study of neural network equalizers in coherent optical systems." *arXiv preprint arXiv:2103.08212*.

Gardan, Julien. 2016. "Additive manufacturing technologies: state of the art and trends." *International Journal of Production Research* 54 (10):3118-3132.

Gobert, Christian, Edward W Reutzel, Jan Petrich, Abdalla R Nassar, and Shashi Phoha. 2018. "Application of supervised machine learning for defect detection during metallic powder bed fusion additive manufacturing using high resolution imaging." *Additive Manufacturing* 21:517-528.

Haghighi, Azadeh, and Lin Li. 2020. "A hybrid physics-based and data-driven approach for characterizing porosity variation and filament bonding in extrusion-based additive manufacturing." *Additive Manufacturing* 36:101399.

He, Kaiming, and Jian Sun. 2015. "Convolutional neural networks at constrained time cost." Proceedings of the IEEE conference on computer vision and pattern recognition.

Hinz, Tobias, Nicolás Navarro-Guerrero, Sven Magg, and Stefan Wermter. 2018. "Speeding up the hyperparameter optimization of deep convolutional neural networks." *International Journal of Computational Intelligence and Applications* 17 (02):1850008.

Holzmond, Oliver, and Xiaodong Li. 2017. "In situ real time defect detection of 3D printed parts." *Additive Manufacturing* 17:135-142.

Jin, Zeqing, Zhizhou Zhang, Joshua Ott, and Grace X Gu. 2021. "Precise localization and semantic segmentation detection of printing conditions in fused filament fabrication technologies using machine learning." *Additive Manufacturing* 37:101696.

Khan, Mohammad Farhan, Aftaab Alam, Mohammad Ateeb Siddiqui, Mohammad Saad Alam, Yasser Rafat, Nehal Salik, and Ibrahim Al-Saidan. 2021. "Real-time defect detection in 3D printing using machine learning." *Materials Today: Proceedings* 42:521-528.

Kim, Hyungjung, Hyunsu Lee, Ji-Soo Kim, and Sung-Hoon Ahn. 2020. "Image-based failure detection for material extrusion process using a convolutional neural network." *The International Journal of Advanced Manufacturing Technology* 111 (5):1291-1302.

Liu, Chenang, Andrew Chung Chee Law, David Roberson, and Zhenyu James Kong. 2019. "Image analysis-based closed loop quality control for additive manufacturing with fused filament fabrication." *Journal of Manufacturing Systems* 51:75-86.

Murugan, Pushparaja. 2017. "Hyperparameters optimization in deep convolutional neural network/bayesian approach with gaussian process prior." *arXiv preprint arXiv:1712.07233*.

Park, Je-Kang, Bae-Keun Kwon, Jun-Hyub Park, and Dong-Joong Kang. 2016. "Machine learning-based imaging system for surface defect inspection." *International Journal of Precision Engineering and Manufacturing-Green Technology* 3 (3):303-310.

Petsiuk, Aliaksei L, and Joshua M Pearce. 2020. "Open source computer vision-based layer-wise 3D printing analysis." *Additive Manufacturing* 36:101473.

Popescu, Diana, Aurelian Zapciu, Catalin Amza, Florin Baciu, and Rodica Marinescu. 2018. "FDM process parameters influence over the mechanical properties of polymer specimens: A review." *Polymer Testing* 69:157-166.

Rahimizadeh, Amirmohammad, Jordan Kalman, Rodolphe Henri, Kazem Fayazbakhsh, and Larry Lessard. 2019. "Recycled glass fiber composites from wind turbine waste for 3D printing feedstock: effects of fiber content and interface on mechanical performance." *Materials* 12 (23):3929.

Rankin, Kyle. 2015. "Hack and: what's new in 3D printing, Part IV: OctoPrint." *Linux Journal* 2015 (257):5.

Rao, Prahalad K, Zhenyu Kong, Chad E Duty, Rachel J Smith, Vlastimil Kunc, and Lonnie J Love. 2016. "Assessment of dimensional integrity and spatial defect localization in additive manufacturing using spectral graph theory." *Journal of Manufacturing Science and Engineering* 138 (5).

Ribeiro, Bernardete. 2005. "Support vector machines for quality monitoring in a plastic injection molding process." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35 (3):401-410.

Salahshoor, Karim, Mojtaba Kordestani, and Majid S Khoshro. 2010. "Fault detection and diagnosis of an industrial steam turbine using fusion of SVM (support vector machine) and ANFIS (adaptive neuro-fuzzy inference system) classifiers." *Energy* 35 (12):5472-5482.

Saluja, Aditya, Jiarui Xie, and Kazem Fayazbakhsh. 2020. "A closed-loop in-process warping detection system for fused filament fabrication using convolutional neural networks." *Journal of Manufacturing Processes* 58:407-415.

Straub, Jeremy. 2015. "Initial work on the characterization of additive manufacturing (3D printing) using software image analysis." *Machines* 3 (2):55-71.

Wang, Jinjiang, Yulin Ma, Laibin Zhang, Robert X Gao, and Dazhong Wu. 2018. "Deep learning for smart manufacturing: Methods and applications." *Journal of manufacturing systems* 48:144-156.

Wu, Jia, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. 2019. "Hyperparameter optimization for machine learning models based on Bayesian optimization." *Journal of Electronic Science and Technology* 17 (1):26-40.

Wu, Mingtao, Vir V Phoha, Young B Moon, and Amith K Belman. 2016. "Detecting malicious defects in 3d printing process using machine learning and image classification." ASME International Mechanical Engineering Congress and Exposition.

Wuest, Thorsten, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. 2016. "Machine learning in manufacturing: advantages, challenges, and applications." *Production & Manufacturing Research* 4 (1):23-45.

Yi, Wu, He Ketai, Zhou Xiaomin, and Ding Wenying. 2017. "Machine vision based statistical process control in fused deposition modeling." 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA).

Zhang, Ying, Guoying Dong, Sheng Yang, and Yaoyao Fiona Zhao. 2019. "Machine Learning Assisted Prediction of the Manufacturability of Laser-Based Powder Bed Fusion Process." International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

Zhang, Ying, and Yaoyao Fiona Zhao. 2021. "Hybrid sparse convolutional neural networks for predicting manufacturability of visual defects of laser powder bed fusion processes." *Journal of Manufacturing Systems*.