RECOGNIZING DEFECTS IN FUSED FILAMENT FABRICATED PARTS: A COMPUTER VISION BASED APPROACH

Joel Chuah Wei Ern, Kok Ven Jyn
Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia
43600 UKM Bangi, Selangor Malaysia.

joelcwe@gmail.com, vj.kok@ukm.edu.my

ABSTRACT

Fused filament fabrication (FFF) is becoming an increasingly popular additive manufacturing process that has found its place in industrial and home settings alike. However, FFF involves multiple process parameters that influence the final quality of the part being fabricated. Defects such as missed layers, shifted layers as well as overfill and underfill can occur if these parameters are not optimized. Thus, it is important to detect such defects as early as possible to prevent potential wastage. Current defect detection algorithm falls into two broad categories, sensor-based and vision-based defect detection. These methods are either complex, cost-prohibitive or a combination of both with the solution often costing many times more than the actual machine. Therefore the aim of this research is to propose a cost-effective and computationally efficient algorithm in defect detection of fused filament fabricated parts. This research follows an experimental approach. A vision-based defect detection framework is presented employing an integrated approach in the detection of error conditions. The first part utilizes a structural approach to detect missed layers and shifted layers defects. The second approach employs machine learning specifically a convolutional neural network to detect normal, underfill and overfill conditions. These two approaches are evaluated using a synthetic dataset generated by manipulating the process parameters of the FFF machine with promising results. Additionally, computational efficiency and cost-effectiveness of the algorithm are validated and serves to provide a practical dimension to this research.

Keywords: Fused Filament Fabrication, FFF, Fused Deposition Modelling, FDM, vision-based defect detection, Convolutional Neural Networks, CNN

I. INTRODUCTION

Fused filament fabrication (FFF) or more commonly known via its trademarked name, Fused Deposition Modelling (FDM) is a form of additive manufacturing. Additive manufacturing is defined as a process of building 3-dimensional (3D) objects from a digital description of the model, layer by layer (Beyer 2014). Specifically in FFF, a 3D object is built by selective deposition of molten material one layer at a time (Beyer 2014; Gibson et al. 2010). Usually, commodity thermoplastic materials are used such as Polylactic Acid (PLA) and Acrylonitrile butadiene styrene (ABS). However, more exotic materials that have a higher performance in factors such as tensile strength are also available suggesting that FFF usage has moved beyond rapid prototyping and into the production of a viable final product.

FFF is arguably the most widely adopted form of additive manufacturing by both hobbyists and businesses alike due to its low cost of entry and ease of maintenance. However, FFF manufacturing technology involves multiple parameters each affecting the final part quality. Common defects as investigated by Peng & Xiao are warping, missing finer details,

and poor surface quality and dimensional errors (Peng & Xiao 2012). The occurrence of such defects is often catastrophic resulting in a product that is not usable either due to poor internal strength or poor surface quality when cosmetic appearance is important. Compounding the problem is the long production or printing time of each part which is an inherent trait of FFF. Thus, early intervention is required if a defect is found to reduce wastage and increase cost and time efficiency.

The importance of early intervention including sensing, control, and process innovations in additive manufacturing is also echoed by Huang et al. (2015) in examining the current gaps in the field. Thus, it is important to detect such defects automatically and inform the machine operator for corrective actions. This ties in with the main objective of this project; which is to develop a FFF defect detection framework. In particular, computer vision-based techniques will be applied in the diagnosis of the defects. Additionally, an attempt is made at making the defect detection algorithm cost-effective and capable to run on a low powered embedded system, the Raspberry Pi 4.

II. RELATED WORK

A. Types of FFF Defects

Peng & Xiao (2012) are one of the first to investigate and categorize defects in FFF. In general, 3 main defect categories are proposed which are shape errors, dimensional errors, and surface quality errors (Peng & Xiao 2012). Defects can also manifest in terms of internal stresses within the fabricated part. Separately, Peng (2012) and Agarwala et al. (1996) describe them as stresses or structural defects. A more recent defect classification can also be found via online resources due to the rapid rise in popularity of FFF machines such as the "Print Quality Guide" provided by Simplify3D ("Print Quality Guide" n.d.). This resource provides a more accessible but targeted description of the common defects affecting FFF machines. However, the defects can still be classified within the general categories proposed by Peng & Xiao as well as Agarwala et al. This research focuses on 4 of these defects which are missed layer, shifted layer, overfill and underfill. Missed layer, shifted layer and underfill are classified as structural defects whereas overfill is classified as shape errors.

B. FFF Defect Detection Algorithms

Rao et al. (2015) are among the first in introducing a real-time monitoring system to detect errors in the additive manufacturing process through the use of sensors. The system proposed employs many sensors which act to detect anomalies or drifts within the manufacturing process (Rao et al. 2015). Since then, there have been several additional sensor-based techniques employed by researchers (Domingo-Espin et al. 2014; Wu et al. 2015). For example, Wu et al. (2015) use acoustic emissions to measure extruder health. This is done by measuring the acoustic signatures of the extruder for each condition; blocked, semi-blocked, normal, run-out. A support vector machine (SVM) classifier is then trained based on this data (Wu et al. 2015).

An alternative to sensor-based approaches are vision based which is within the scope of this research. In general, vision based approaches can be split into 3 categories. The first approach involves the use of a reconstructed 3D model and is compared against the printed object. Holzmond & Li (2017) employs this approach via the use of a stereoscopic camera to generated a 3D model of the current layer being printed. A global comparison is then made with the reconstructed model. Similarly, Nuchitprasitchai et al (2017) applies this approach using a side perspective instead. This allows a comparison on the entire fabricated part to be

done detecting accumulative errors. However, both works follow a global comparison approach where defect classification is not attempted.

The second category involves the use of image processing techniques. Baumann & Roller utilizes this approach using a combination of thresholding and blob detection (2016). The thresholding operation is done in the HSV colour space manually. Blob detection is then carried out on the thresholded image to detect defects such as missed layer, shifted layer and layer detachment. Lyngby et al. (2017) also employs an improved algorithm using image processing techniques. Instead of relying purely on blob detection, a geometric representation of the part is retrieved from its 3D model which the image is compared against.

The final category involves the use of supervised learning. Liu et al. (2019) proposed a closed-loop quality control approach that retains defect type classification. This approach is carried out using two microscopes that inspect the surface of the current part layer that is being deposited. To classify defects, a large number of images are generated (both normal and defect conditions) by varying the machine parameters. However, due to the setup where the microscopes are situated very close to the extruder that deposits the material, the detection can only happen on the XY surface plane. A supervised learning model is also employed by Paraskevoudis et al. (2020) via the use of the convolutional neural network (CNN) model, Single Shot Detector (SSD) to detect and locate stringing defects. However, results are not promising with a precision of 0.44, recall of 0.69 at Intersection over Union (IoU) of 0.4 (Paraskevoudis et al. 2020) largely due to the use a small dataset.

Although the results from Paraskevoudis et al. are not encouraging, the benefits of CNNs as an end-to-end classification approach should not be discounted. Results by Zhang et al. in the domain of additive metal sintering shows promise where they obtain a defect classification acccuracy of 82% using a patch-based network (2019). Furthermore, the success of deep classification neural networks (DCNN) in single-label image classification problems, surpassing human-level performance in the MNIST and ImageNet datasets also enforces this idea (Rawat & Wang 2017).

In summary, the review on related work shows that defect detection approaches utilizes algorithms that focuses on a specific category of defect. For example, Baumann & Roller employs an algorithm based on image processing techniques in detecting structural layer defects such as layer shift and layer misses whereas Lui et al. employs a supervised learning approach in classifying infill defects. Secondly, each approach employed by past researchers have their own strengths and weaknesses. Therefore, this research will focus on proposing an integrated approach building upon the algorithms employed by past researchers.

III. RESEARCH QUESTIONS

The main objective of this research is to propose an effective defect detection framework capable of detecting 4 types of common defects which are missed layer, shifted layer, overfill and underfill. The proposed algorithm must be cost effective and efficient allowing it to run on a low powered embedded system, the Raspberry Pi 4. Against this backdrop, the research will focus on answering the following research questions;

1. Is a single camera solution capable of detecting defects in FFF parts accurately and reliably?
2. What are the factors that affect the classification accuracy of FFF parts defects?

IV. METHODOLOGY

A. Overview of Proposed Framework

The proposed framework is shown in Figure 1 and mainly consists of 2 major components; a region of interest (ROI) extraction followed by the defect detection component. The defect detection component employs an integrated approach combining multiple algorithms that each target specific types of defects. The flow of the framework is summarized below:

1. An image is taken after each layer is fabricated.
2. ROI extraction is then carried out to extract the current layer from the image background.
3. Defect detection is then done sequentially with missed layer detection done first.
4. If no missed layers are detected, shifted layer detection is then carried out.
5. If neither missed layer or shifted layers are detected, overfill and underfill detection is finally carried out.

Figure 1 Proposed Framework

B. Region of Interest Extraction

The region of interest extraction process is summarized in Figure 2. The region of extraction process hinges on two main concepts. Firstly, it leverages on extracting the 3D description of the model being printed by pre-processing the GCODE file. This results in a point cloud containing the outer contours or walls of each layer. Secondly, an accurate pose of the printed object is estimated via the process of camera calibration and pose estimation. The camera calibration utilizes the checkerboard multiplane calibration process which is based on work done by Bouguet and Zhang (Bouguet n.d.; Z. Zhang 2000). The pose estimation step is then done via the use of a specialized holder printed onto the machine bed before a single ArUco marker is placed upon it. In particular, this is achieved by finding the 2D-3D correspondence of the marker. This operation is more commonly known as Perspective-n-point which is based on the pinhole camera model. This enables an accurate 3D-2D mapping allowing the point cloud to be projected accurate onto the image as shown in Figure 3. Finally a perspective correction is done on the extracted ROI. This step is only applicable for the underfill and overfill detection and is done to allow the model to generalize better.

Figure 2 Region of Interest Extraction

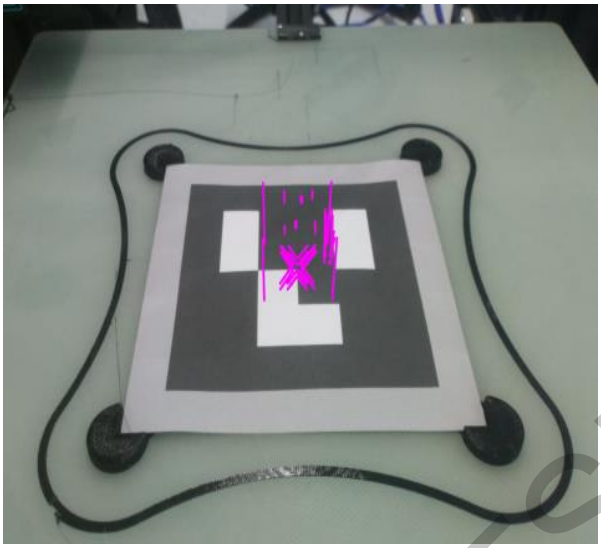

Figure 3 ArUco marker and projection of XYZ cube

C. Missed Layer Detection

The missed layer detection algorithm consists of a thresholding operation followed by structural analysis using the concept of image moments. In order to ensure that thresholding works reliably across different polymer colours and without manual intervention, a combination of histogram backprojection followed by Otsu's thresholding is employed. Histogram backprojection is based on work done by Swain & Ballard (1992). In particular, this algorithm is employed to calculate the probability of a pixel intensity belonging to the ROI. The output of this operation is an image matrix containing the normalized probability distribution allowing a more accurate thresholding process via Otsu's method.

Contours are then found from this thresholded image with the largest used for centroid calculation. The centroid calculation is based on moments with the two dimensional discrete form applicable for images as shown below:

$$m_{p,q} = \sum_{x=0}^{w} \sum_{y=0}^{h} x^p y^q f(x,y)$$

Where the centroid coordinate is simply the first-order moments, $m_{1,0}$ and $m_{0,1}$ divided by the zero-order moment, $m_{0,0}$ which also happens to denote the area of the contour:

$$x = \frac{m_{1,0}}{m_{0,0}}, y = \frac{m_{0,1}}{m_{0,0}}$$

A comparison of centroids between the contour extracted from the point cloud representation and the contour extracted from the thresholded image is then compared. If there are no defects, the 2 centroids will overlap with a small margin of error.

D. Shifted Layer Detection

The shifted layer detection follows the same algorithm as that of missed layer detection with a few key differences. Firstly, the contour search space is dilated by a factor of 1.1x using the following formula:

$$x_d = \lceil (x_i - x_c) c_{dilation} + x_d \rceil,$$
$$y_d = \lceil (y_i - y_c) c_{dilation} + y_d \rceil$$

Where (x_d, y_d) are the new dilated points, (x_i, y_i) the original points, $c_{dilation}$ the dilation factor and (x_c, y_c) the centroid coordinates. The error threshold is also adjusted to account for the increased sensitivity to the size of the fabricated part due to the dilation operation where the constants m and c are found via linear approximation:

$$error\ threshold = m\left(\frac{expected\ contour\ area}{image\ size}\right) + c$$

E. Overfill and Underfill Detection

Three efficient CNN models were picked and evaluated based on the paper by Bianco et al. (2018). In particular, models that are memory efficient and have good inference timing on a low powered system while maintaining top-1 accuracy of more than 60% on the ImageNet dataset. The chosen models are Xception, MobileNet-v2 and ShuffleNet-v2. The training phase is done offline on Google Colab using the Keras API and Tensorflow framework.

3. RESULTS AND DISCUSSION

A. Experimental Setup

Two different sets of experiments are carried out to evaluate the effectiveness of the proposed algorithms. The first set targets the missed layer and shifted layer detection whereas the second set targets overfill and underfill detection. In particular, the algorithms are evaluated for accuracy and timing. Each of these experiments are evaluated using synthetic datasets generated by either manipulating the GCODE file or the process parameters within the slicing software, Cura. In particular, the dataset generated for shifted layer detection involves the addition of G92 commands to introduce shifts in the x and y directions as shown in Figure 4. The dataset generated for overfill and underfill detection is done by changing the extruder flow rate on Cura using the rates shown in Table 1. For the shifted layer detection, 3 models (Elliptic Polygon, Octagonal, Irregular Polygon) printed in 3 colours (Orange, Blue, Black) were each varied to contain shifts in the x, y and 45° xy directions at varying heights. For overfill and underfill detection, a total of 3271 images were collected with 1271 images categorized as normal, 906 as overfill and 1049 as underfill. Timing experiments were done both on the Raspberry Pi 4 and a desktop computer with a Core i7-4770 processor and 16GB of memory.

```
9120 ;MESH:NONMESH
9121 G0 F600 X114.522 Y138.016 Z4.2
9122 G0 F9000 X109.916 Y138.577
9123 G0 X109.915 Y138.898
9124 G92 X114.915
9125 ;TIME_ELAPSED:2675.830129
9126 ;LAYER:20
9127 ;TYPE:WALL-INNER
9128 ;MESH:elliptic_polygon.STL
9129 G1 F1500 X106.056 Y138.748 E2611.3462
```

Figure 4 Example G92 offset of 5mm in the positive x direction at layer 20

Table 1 Extruder flow rate

Defect Class	Flow rate (%)
Normal	100
Overfill	200
Underfill	40

B. Results

Table 2 and Table 3 indicate the accuracy and timing results respectively for the missed layer detection algorithm. The average F1-Score recorded is 0.872. However, recall values varies between 0.7 and 1.0 with the Octagonal model showing the worst results. Low recall values indicate that more images are categorized as false negatives compared to true positives. In terms of timing, the average time to process a single image is around 0.3 seconds which is acceptable on the target system, the Raspberry Pi 4. Comparatively, the desktop computer is on average 4 times faster signifying the need for an efficient algorithm.

Table 2 Missed Layer Detection Accuracy Results

Model	Colour	Defect location	Precision	Recall	F1-Score
Elliptic Polygon	Orange	Layer 11	0.971	0.846	0.904
	Blue	Layer 21	0.957	0.759	0.846
	Black	Layer 31	0.783	0.947	0.857
Octagonal	Orange	Layer 31	0.967	0.744	0.842
	Blue	Layer 31	0.971	0.708	0.819
	Black	Layer 31	0.903	0.933	0.918
Irregular polygon	Orange	Layer 21	0.913	0.724	0.808
	Blue	Layer 10	0.971	0.872	0.919
	Black	Layer 10	0.927	0.974	0.95

Table 3 Missed Layer Detection Timing Results

Model	Colour	Defect Location	Average Timing Raspberry Pi 4 (s)	Average Timing Desktop Computer (s)
Elliptic Polygon	Orange	Layer 10	0.34	0.08
	Blue	Layer 21	0.36	0.11
	Black	Layer 31	0.35	0.09
Octagonal	Orange	Layer 31	0.34	0.08
	Blue	Layer 31	0.34	0.08
	Black	Layer 31	0.33	0.08
Irregular polygon	Orange	Layer 21	0.31	0.07
	Blue	Layer 11	0.31	0.07
	Black	Layer 11	0.31	0.07

Table 4 and Table 5 indicate the accuracy and timing results respectively for the shifted layer detection algorithm. The average F1-Score recorded is 0.75 which is lower than missed layer detection. In particular, layer shifts involving the y-axis which is the axis parallel to the perspective view of the camera show the worst results regardless of model shape and colour. Timing efficiency is similar to that of missed layer detection, at 10% worst, on average.

Table 4 Shifted Layer Detection Accuracy Results

Model	Colour	Defect location	Precision	Recall	F1-Score
Elliptic Polygon	Orange	5mm x-axis shift at layer 31	0.895	0.895	0.895
		5mm y-axis shift at layer 11	0.919	0.872	0.895
		-5mm 45° xy shift at layer 31	0.917	0.579	0.710
	Blue	5mm x-axis shift at layer 21	0.931	0.931	0.931
		5mm y-axis shift at layer 21	0.927	0.974	0.95
		5mm 45° xy shift at layer 11	1.0	0.795	0.886
	Black	5mm x-axis shift at layer 21	1.0	0.862	0.926
		5mm y-axis shift at layer 21	All cases predicted negative.		
		-5mm 45° xy shift at layer 21	All cases predicted negative.		
Octagonal	Orange	5mm x-axis shift at layer 21	0.931	0.9	0.915
		5mm y-axis shift at layer 21	All cases predicted negative.		
		-5mm 45° xy shift at layer 31	0.824	0.424	0.56
	Blue	5mm x-axis shift at layer 11	1.0	0.818	0.9
		5mm y-axis shift at layer 11	0.906	0.935	0.920
		5mm 45° xy shift at layer 11	0.914	0.97	0.941
	Black	5mm x-axis shift at layer 31	1.0	0.844	0.915
		5mm y-axis shift at layer 31	0.688	0.939	0.969
		5mm 45° xy shift at layer 31	0.667	0.970	0.790
Irregular polygon	Orange	5mm x-axis shift at layer 11	0.919	0.872	0.895
		5mm y-axis shift at layer 11	0.2	0.03	0.05
		5mm 45° xy shift at layer 11	0.872	0.872	0.872
	Blue	5mm x-axis shift at layer 21	0.96	0.828	0.889
		5mm y-axis shift at layer 21	0.96	0.828	0.889
		5mm 45° xy shift at layer 21	1.0	0.828	0.906
	Black	5mm x-axis shift at layer 11	1.0	0.949	0.974
		5mm y-axis shift at layer 11	1.0	0.744	0.853
		5mm 45° xy shift at layer 11	1.0	0.846	0.917

Table 5 Shifted Layer Detection Timing Results

Model	Colour	Defect Location	Average Timing Raspberry Pi 4 (s)	Average Timing Desktop Computer (s)
Elliptic Polygon	Orange	5mm x-axis shift at layer 31	0.38	0.09
	Blue	5mm y-axis shift at layer 21	0.35	0.09
	Black	-5mm 45° xy shift at layer 21	0.34	0.08
Octagonal	Orange	5mm x-axis shift at layer 21	0.36	0.08
	Blue	5mm y-axis shift at layer 11	0.37	0.08
	Black	5mm 45° xy shift at layer 31	0.37	0.08
Irregular polygon	Orange	5mm x-axis shift at layer 11	0.35	0.08
	Blue	5mm y-axis shift at layer 21	0.33	0.08
	Black	5mm 45° xy shift at layer 11	0.34	0.08

Table 6, Table 7 and Table 8 indicate the total learning hours and accuracy across validation and test sets for different CNN model configurations. In particular, the Xception model with fine tuning over the entire model gave the highest validation and test set accuracy at 0.9767 and 0.894 respectively. However, training times are also the longest at 6.45 hours. Similarly, the MobileNet-v2 model with fine tuning over the entire model yields the best results

with validation and test set accuracy at 0.9627 and 0.8641 respectively. However, training times are much lower at only 1.17 hours. On the other hand, ShuffleNet-v2 shows respectable results with the validation and test set accuracy at 0.941 and 0.8478 respectively considering only random weights were used due to it not being available as a pre-trained model on the Keras API. Training times falls in the middle at 3.19 hours for ShuffleNet-v2.

Table 6 Xception Model Results

Xception Configuration	Total training time (hours)	Validation Set Accuracy	Test Set Accuracy
Xception + Random weights	4.46	0.8929	0.8125
Xception + ImageNet weights	4.2	0.9037	0.8424
Xception + Fine tuning entire model	6.45	0.9767	0.8940
Xception + Fine tuning from layer 100	3.7	0.9177	0.8832
Xception + Fine tuning from layer 50	4.36	0.9177	0.8832

Table 7 MobileNet-v2 Results

MobileNet-v2 Configuration	Total training time (hours)	Validation Set Accuracy	Test Set Accuracy
MobileNet-v2 + Random weights	1.06	0.9379	0.8288
MobileNet-v2 + ImageNet weights	1.46	0.9425	0.8207
MobileNet-v2 + Fine tuning entire model	1.17	0.9627	0.8641
MobileNet-v2+ Fine tuning from layer 100	0.69	0.9674	0.8560
MobileNet-v2 + Fine tuning from layer 50	0.71	0.9689	0.8478

Table 8 ShuffleNet-v2 Results

Shufflenet-v2 Configuration	Total training time (hours)	Validation Set Accuracy (%)	Test Set Accuracy (%)
ShuffleNet-v2+ Random weights	3.19	0.9410	0.8478

Table 9 shows the average inference timing of a single image for each CNN model in scope. The fastest inference is achieved with the MobileNet-v2 architecture at 0.112 seconds whereas the slowest is Xception at 1.308 seconds on the Raspberry Pi 4 platform. The MobileNet-v2 model was chosen as the underfill and overfill model due to its fast inference timing. Also, the accuracy of this model is similar to that achieved using the Xception model.

Table 9 Average Inference Timing

CNN Model	Average Timing Raspberry Pi 4 (s)	Average Timing Desktop Computer (s)
Xception	1.308	0.244
MobileNet-V2	0.112	0.053
ShuffleNet-V2	0.193	0.037

C. Factors Affecting Defection Detection Framework Accuracy

The experimental results show 3 key factors that affect the accuracy of the defect detection framework. The first factor pertains to algorithmic limitations specifically the missed layer and shifted layer detection. Both of these algorithms utilizes thresholding techniques via a combination of histogram backprojection and Otsu's method and requires the thresholding operation to be as accurate as possible. However, differing lighting conditions affect this operation. A defective thresholding operation will consequently affect the centroid calculation

as shown in Figure 5. The expected contour and centroid is shown in green whereas the defective thresholding and the resulting centroid is shown in pink and red respectively. Secondly, the performance of the shifted layer detection is also influenced by the position of the camera. In particular shifts along the perspective view of the camera are not detected accurately due to the influence of previous layers within the search space. This is evident in Figure 6 where the expected contour is again shown in green and the contour found with a dilated search space from the image is shown in pink.

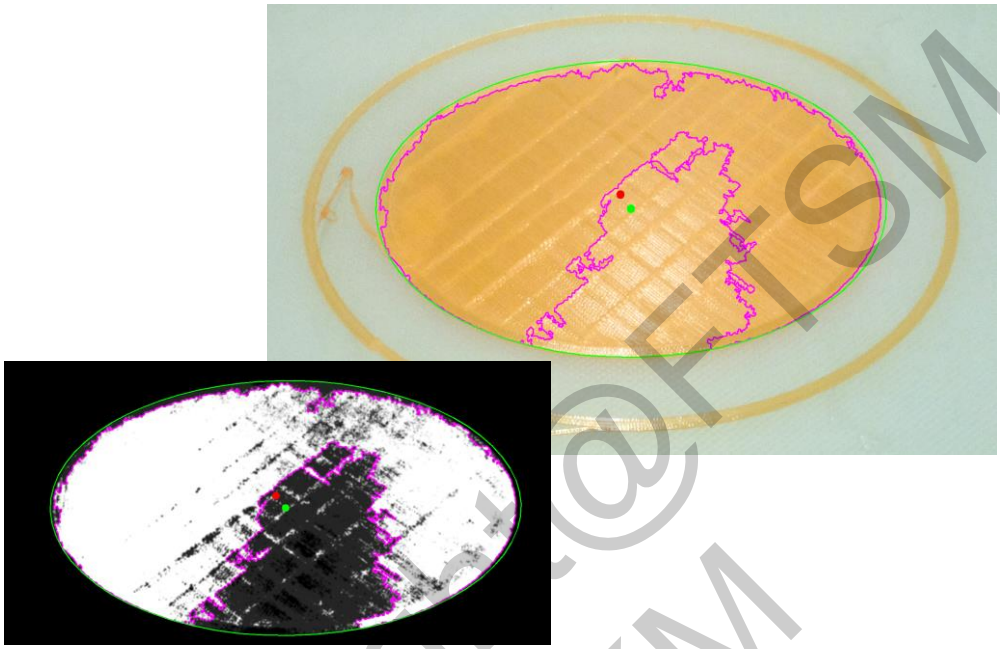

Figure 5 False Positive - Defective Thresholding Operation

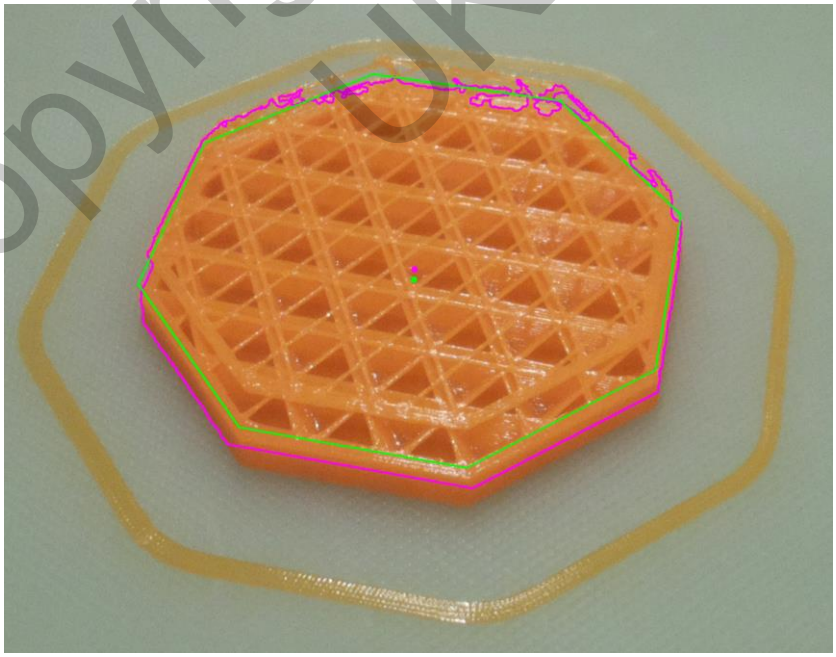

Figure 6 False Negative - Influence of previous layers

Secondly, there is also the issue of size and variation of the training set. The data collection process is inherently time consuming due to the nature of the FFF process. Thus, only about 1000 images per class of normal, overfill and underfill were generated. The accuracy of the evaluated models can be improved up to a certain point if more data can be generated as investigated by Foody & Arora in their work on factors affecting artificial neural network accuracy (1997). Also, the data generated only covers a specific infill design, "cubic" and will likely fail for new unseen infill designs.

Lastly, transfer learning has been found to be a factor in the accuracy of the model used for overfill and underfill detection. There is a marked improvement in the accuracy for both Xception and MobileNet-v2 when ImageNet weights were used together with a round of fine tuning on the entire model. This results are also in line with that from Yosinski et al. in their work on examining generability vs specificity of transferring features (2014). They denote that initializing a network with transfer features improve generalization performance even with fine-tuning.

D. Cost Analysis on Proposed Defect Detection Framework

An objective of this research is to propose a defect detection framework that is also cost-effective and appropriate in relation to the cost of the FFF machine. Popular printers across different price ranges are shown in Table 10.The total cost of the framework is evaluated against FFF machines within the price bracket of under $300 as it is the most popular among hobbyists and first time adopters of FFF. Specifically, the total cost should not exceed 50% of this price bracket which equates to $150 (RM615). This target was achieved successfully with the use of a single camera coupled with the Raspberry Pi 4 as the printer control platform. A breakdown of the costs involved is shown in Table 11.

Table 10 Cost of Popular FFF Machines

FFF Machine	Price Bracket ($USD)	Market Price ($USD)
Dremel DigiLab 3D45	< 2000	1,899
Original Prusa i3 MK3S	< 1000	906
Artillery Sidewinder X1 V4	< 500	479
Creality Ender 3 V2	< 300	262

Table 11 Defect Detection Cost Breakdown

Item	Cost ($USD)
Raspberry Pi 4 4GB	65
Raspberry Pi Camera Module V2	25
Raspberry Pi 4 4GB Power Supply	7.95
Total	97.95

4. CONCLUSION

In this research, a defect detection framework for FFF fabricated parts based on a single camera is proposed capable of detecting 4 different types of defects; missed layers, shifted layers, overfill and underfill. Factors affecting the accuracy of the proposed defect detection framework are also analysed where issues such as differing lighting conditions and the position of the camera affect the defect detection approaches based on image processing techniques. On

the other hand, issues such as the size and variation of the training set and the use of transfer learning affects the approach based on supervised learning.

REFERENCES

Agarwala, M. K., Jamalabad, V. R., Langrana, N. A., Safari, A., Whalen, P. J. & Danforth, S. C. 1996. Structural quality of parts processed by fused deposition. *Rapid Prototyping Journal* 2(4): 4–19.

Baumann, F. & Roller, D. 2016. Vision based error detection for 3D printing processes. (H. F. Abdul Amir, A. M. Korsunsky, & Z. Guo, Eds.)*MATEC Web of Conferences* 59: 06003.

Beyer, C. 2014. Strategic Implications of Current Trends in Additive Manufacturing. *Journal of Manufacturing Science and Engineering, Transactions of the ASME* 136(6).

Bianco, S., Cadene, R., Celona, L. & Napoletano, P. 2018. Benchmark analysis of representative deep neural network architectures. *IEEE Access* 6: 64270–64277.

Bouguet, J.-Y. (n.d.). Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/ [5 July 2020].

Domingo-Espin, M., Borros, S., Agullo, N., Garcia-Granada, A. A. & Reyes, G. 2014. Influence of Building Parameters on the Dynamic Mechanical Properties of Polycarbonate Fused Deposition Modeling Parts. *3D Printing and Additive Manufacturing* 1(2): 70–77.

Foody, G. M. & Arora, M. K. 1997. An evaluation of some factors affecting the accuracy of classification by an artificial neural network. *International Journal of Remote Sensing* 18(4): 799–810.

Gibson, I., Rosen, D. W. & Stucker, B. 2010. Additive manufacturing technologies: Rapid prototyping to direct digital manufacturing. *Additive Manufacturing Technologies: Rapid Prototyping to Direct Digital Manufacturing*. Springer US.

Holzmond, O. & Li, X. 2017. In situ real time defect detection of 3D printed parts. *Additive Manufacturing* 17: 135–142.

Huang, Y., Leu, M. C., Mazumder, J. & Donmez, A. 2015. Additive Manufacturing: Current State, Future Potential, Gaps and Needs, and Recommendations. *Journal of Manufacturing Science and Engineering* 137(1).

Liu, C., Law, A. C. C., Roberson, D. & Kong, Z. (James). 2019. Image analysis-based closed loop quality control for additive manufacturing with fused filament fabrication. *Journal of Manufacturing Systems* 51: 75–86.

Lyngby, R. A., Wilm, J., Eiríksson, E. R., Nielsen, J. B., Jensen, J. N., Aanæs, H. & Pedersen, D. B. 2017. In-line 3D print failure detection using computer vision.

Nuchitprasitchai, S., Roggemann, M. & Pearce, J. M. 2017. Factors effecting real-time optical monitoring of fused filament 3D printing. *Progress in Additive Manufacturing* 2(3): 133–149.

Paraskevoudis, K., Karayannis, P. & Koumoulos, E. P. 2020. Real-Time 3D Printing Remote Defect Detection (Stringing) with Computer Vision and Artificial Intelligence. *Processes* 8(11): 1464.

Peng, Anhua. 2012. Research on the interlayer stress and warpage deformation in FDM. *Advanced Materials Research* 538–541: 1564–1567.

Peng, AnHua & Xiao, X. 2012. Investigation on Reasons Inducing Error and Measures Improving Accuracy in Fused Deposition Modeling. *INTERNATIONAL JOURNAL ON Advances in*

Information Sciences and Service Sciences 4(5): 149–157.

Print Quality Guide. (n.d.). https://www.simplify3d.com/support/print-quality-troubleshooting/ [6 December 2019].

Rao, P. K., Liu, J., Roberson, D., Kong, Z. & Williams, C. 2015. Online Real-Time Quality Monitoring in Additive Manufacturing Processes Using Heterogeneous Sensors. *Journal of Manufacturing Science and Engineering, Transactions of the ASME* 137(6).

Rawat, W. & Wang, Z. 2017. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation* 29(9): 2352–2449.

Swain, M. J. & Ballard, D. H. 1992. Indexing via Color Histograms. *Active Perception and Robot Vision*, hlm. 261–273. Berlin, Heidelberg: Springer Berlin Heidelberg.

Wu, H., Wang, Y. & Yu, Z. 2015. In situ monitoring of FDM machine condition via acoustic emission. *The International Journal of Advanced Manufacturing Technology* 84(5–8): 1483–1495.

Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. 2014. *How transferable are features in deep neural networks?*

Zhang, B., Jaiswal, P., Rai, R., Guerrier, P. & Baggs, G. 2019. Convolutional neural network-based inspection of metal additive manufacturing parts. *Rapid Prototyping Journal* 25(3): 530–540.

Zhang, Z. 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(11): 1330–1334.