Detecting Attacks in CyberManufacturing Systems: Additive Manufacturing Example

Mingtao Wu[1], Heguang Zhou[1], Longwang Lucas Lin[1], Bruno Silva[2], Zhengyi Song[1], Jackie Cheung[1] and Young Moon[1]

[1]*263 Link Hall, Department of Mechanical and Aerospace Engineering, Syracuse University, Syracuse NY 13244, USA*
[2]*15 Francisco Alexandre Almeida - Vila Westin, UNIFAE, São João da Boa Vista SP 13870-377, Brasil*

Abstract. CyberManufacturing System is a vision for future manufacturing where physical components are fully integrated with computational processes in a connected environment. However, realizing the vision requires that its security be adequately ensured. This paper presents a vision-based system to detect intentional attacks on additive manufacturing processes, utilizing machine learning techniques. Particularly, additive manufacturing systems have unique vulnerabilities to malicious attacks, which can result in defective infills but without affecting the exterior. In order to detect such infill defects, the research uses simulated 3D printing process images as well as actual 3D printing process images to compare accuracies of machine learning algorithms in classifying, clustering and detecting anomalies on different types of infills. Three algorithms - (i) random forest, (ii) k nearest neighbor, and (iii) anomaly detection - have been adopted in the research and shown to be effective in detecting such defects.

1 Introduction

CyberManufacturing System is a vision for future manufacturing where physical components are fully integrated with computational processes in a connected environment. CyberManufacturing system utilizes recent developments in Internet of Things, Cloud Computing, Fog Computing, Service-Oriented Technologies, Modeling and Simulation, Virtual Reality, Embedded Systems, Sensor Networks, Wireless Communications, Machine Learning, Data Analytics, and Advanced Manufacturing Processes. Manufacturing resources and abilities can be sensed and connected to each other directly or through the Internet, thus enabling intelligent behaviors of a manufacturing system or networked manufacturing systems such as self-awareness, self-prediction, self-organization, and self-configuration [1]. The concept can also extend to a network of shared CyberManufacturing resources over the Internet.

However, realizing the vision requires that its security be adequately ensured. A recent study by IBM reports that manufacturing is the second most frequently targeted industry in 2015 in terms of the number of cyber-attacks [2]. The consequence of cyber-attack on manufacturing systems could be defected parts, the cost of poor quality, damaged equipment, and loss of customer loyalty or even create customer safety issues [3].

Particularly, additive manufacturing systems have unique vulnerabilities to malicious attacks, which can result in defective infills but without affecting the exterior. This can lead to the production of malicious defective parts without any warning. Those defects can also result in a reduction in yield load, reduction in strain at failure, change of natural frequency, etc. that eventually affect product's safety and functionality. Image classification using machine learning techniques is adopted to detect such defects caused by deliberate attacks, from normal parts as well as common defects such as weak infill, gaps in thin walls, inconsistent extrusion, layer separation and splitting, and bed drop.

The previous research by Wu et al. [3] defined five different infill defect patterns: Seam, Irregular Polygon, Circle, Rectangle, and Triangle. The research used Naive Bayes Classifier and J48 Decision Trees classification algorithms to differentiate the defective and non-defective parts. The images were simulated from MakerBot software with the setting of Honeycomb infills. The result showed Naive Bayes Classifier has an accuracy of 85.26% and J48 Decision Trees has an accuracy of 95.51% for correct image classification.

This work builds on the previous research [3], but extends further by:

1) using real images collected by an Arducam OV2640 2 Megapixels Camera. Relevant changes are made to address the differences between the simulated images and real images.
2) using unsupervised machine learning methods in addition to supervised marching learning methods, thus covering not only classification but also clustering and anomaly detection mechanisms;
3) testing four additional types of infills (Diamond, Linear, Star and Catfill) in addition to a single

type of infill (Honeycomb), to test the infill type factor on accuracy of machine learning methods.

2 Image collection

Two methods have been used in collecting images: (i) by generating software simulated images, and (ii) by capturing real images using a real-time camera system that was designed and installed in a 3D printer to collect images under real environment.

2.1 Images from simulation

Simulated images have been generated from 3D printing software, MakerBot Desktop 3.9.1 preview function. For the feature extraction stage, a fixed size of images at 512 x 512 pixels was used. In total 3887 images have been generated for training classifier and testing. 532 images of non-defect parts have been generated and labeled as group A. Within group A, 34 images are from parts with honeycomb infill, 60 images are from parts with linear infill, 151 images are from parts with star infill, 140 images are from parts with catfill infill, and 147 images are from parts with diamond infill. Fig 1. shows five types of non-defective part infill. 3355 images of defective parts are captured and labeled as group B. Within group B, 102 images are from parts with honeycomb infill, 349 images are from parts with linear infill, 956 images are from parts with star infill, 946 images are from parts with catfill infill, 1002 images are from parts with diamond infill. Both the non-defective group A images as well as the defective group B images have been captured every 3-5 layers during the printing process, with infill density varying from 8%-12% to increase the diversity of training image.

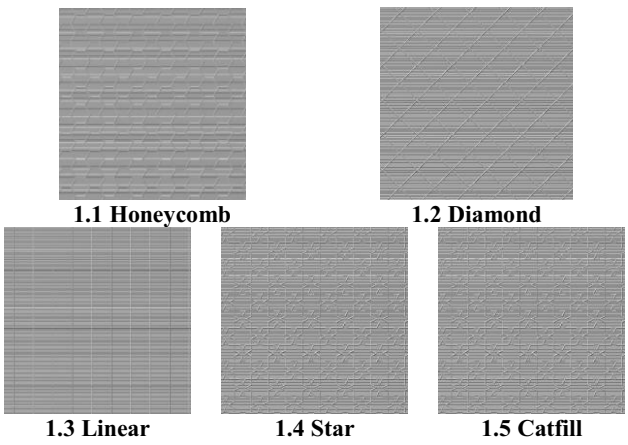


Figure 1. Five types of part infill.

2.2 Images from camera

To test and verify the image classification method in real environment, a camera-based vision detection system has been designed and installed on two different brands of 3D printers: MakerBot Replicator™2 and CubePro Duo. The camera used in the experiment is an Arducam Mini Module Camera Shield with OV2640 2 Megapixels Lens, compatible with Arduino UNO Mega2560 Board. The camera unit's dimensions are 3 x 2 x 1 inches, connected to the Arduino UNO via extended jumper wires. With programming in Arducam software, it can produce images any size scaling down from SXGA to 40×30 in jpeg format. As a result, the feature extraction code for previous 512x512 size images needed to be altered. Moreover, because the camera provides a color image instead of grayscale image from simulation, additional image preprocessing was necessary.

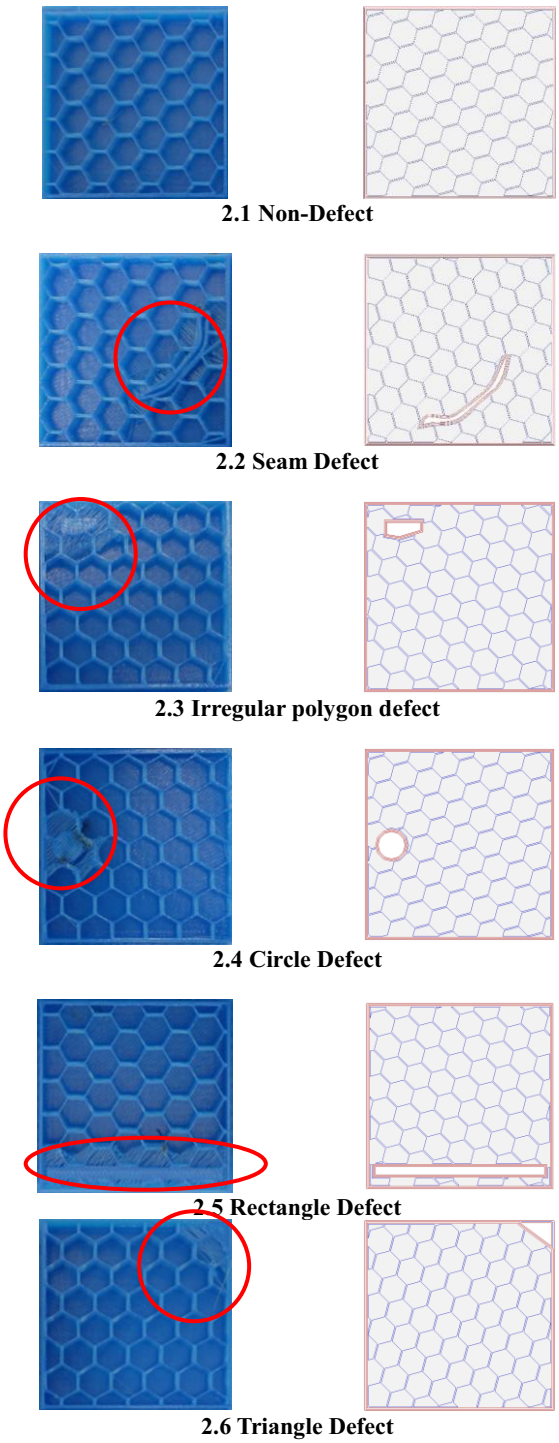


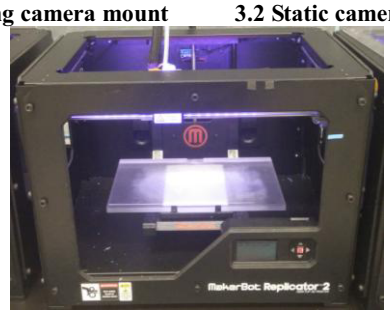
Figure 2. Five Types of Infill Defect Patterns Camera & Simulation View.

MakerBot Replicator™2 has the build envelope of 11.2 x 6.0 x 6.1" and can print at 100 µm per layer.

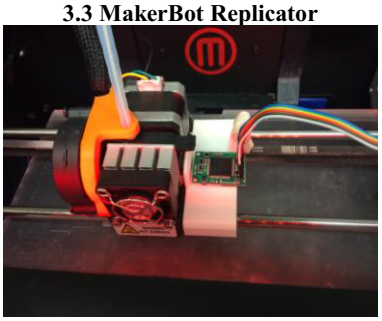
Installation of the camera on a MakerBot Replicator 2 is shown in Fig 3. There are two ways to install on MakerBot Replicator 2. One is mounting the camera right next to the intruder and move along with it, called 'moving camera.' The other is mounting the camera on the frame of the 3D printer, called 'static camera.' The CubePro Duo is capable of printing models up to 9 x 10.75 x 9.5" at the resolution of as fine as 70 µm. The camera is mounted on a cooling fan frame of CubePro Duo located next to the first intruder.

3.1 Moving camera mount

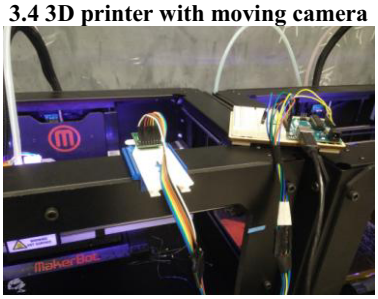
3.2 Static camera mount



3.3 MakerBot Replicator



3.4 3D printer with moving camera



3.5 3D printer with static camera

Figure 3. MakerBot Replicator with moving and static camera

3 Machine learning methods

To improve the accuracy for malicious defect detection, this work extends from supervised machine learning methods to unsupervised machine learning methods. The reason is that even if a large number of sample experiments are used as training set, it is impossible to predict unseen malicious attack patterns. By implementing random forest, k-nearest neighbors (kNN) and anomaly detection machine learning algorithms, the research attempts to explore the optimal methods for 3D printing vision detection system.

Random Forest: A random forest classifier consists of a number of trees. The leaf nodes of each tree are labeled by estimating the posterior distribution over the image classes. Each internal node contains a test that best classifies the data. An image is classified by sending it down each tree and aggregating the reached leaf distributions [4]. In this research, three decision trees are used and each of them has five leaf nodes to classify. Compared to J48 decision tree algorithm, the random forest classifier with three decision trees was able to achieve a high accuracy with relatively shorter time to run the classifier.

k-Nearest Neighbors (kNN): kNN classifier is used to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine [5]. K value is a positive integer, typically small, but normally larger than one. In this research, a heuristic rule that is k is calculated as the square root of "n" number of features, which is very likely to achieve the highest accuracy.

Anomaly detection: Anomaly detection techniques have been used to detect any changes in an image over time (motion detection) or in regions which appear abnormal on the static image [6]. Images from moving camera are often not so clear. Therefore, in this research, the anomaly detection method is combined with the random forest method to increase the accuracy. By embedding the anomaly detection method in the random forest method, detection accuracy was increased almost by 5% for the moving camera cases as shown in Table 2.

4 Result analysis

The analysis for the research focuses on three main issues: i) accuracy of machine learning methods on both simulated and real images, ii) whether a certain infill type has particular influence on accuracy of detection, and iii) whether the vision detection system can maintain acceptable accuracy levels in real environment and can be adopted for real-time detection applications. Accuracy is defined by the equation (1). Where *TruePositive* means images in class A that are predicted as class A, and *TrueNegative* stands for images in class B predicted as class B.

$$Accuracy = \frac{TurePositive + TureNegative}{Total} \quad (1)$$

According to (1), the accuracy for the Simulated Linear Infill Random Forest classification example in Table 1 is calculated as:

$$Accuracy = (57+330)/(57+3+19+330) = 94.6\%$$

Table 1. Confusion Matrix for Simulated Linear Infill Random Forest Classification Results.

Class		Predicted Class	
		A	B
Actual Class	A	57	3
	B	19	330

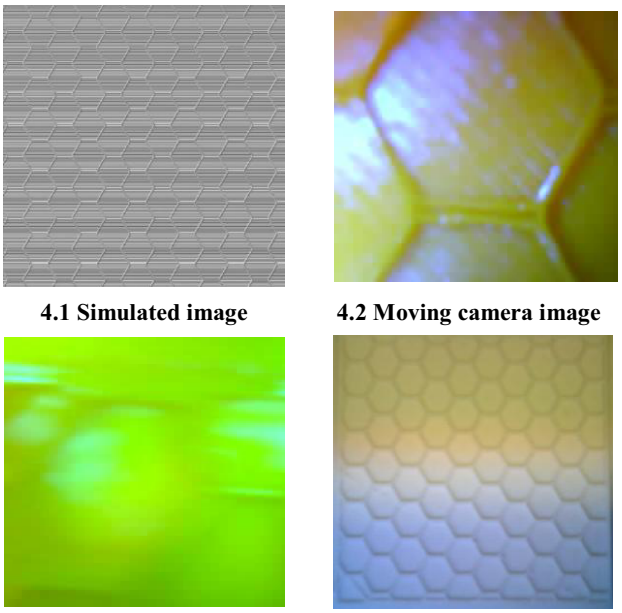
Table 2. Accuracy results (HC-Honeycomb, DM-Diamond, LN-Linear, ST-Star, CA-Catfill)

Accuracy		Machine Learning Method		
		Random Forest (%)	kNN (%)	Anomaly Detection (%)
Image from Simulation	HC	88.4	81.3	100.0
	DM	100.0	85.0	100.0
	LN	94.6	92.5	100.0
	ST	97.8	100	99.8
	CA	91.5	100	100.0
Image from Moving Camera	HC	68.4	68.75	72.5
Image from Static Camera		95.5	87.5	96.1

As shown in Table 2, images from simulation show that different type of infill have some influences on accuracy of kNN, but still within an acceptable range. The reason behind 100% accuracies by the anomaly detection method could be the simulated images are captured as clean image without any distortion. For the simulated data sets, the anomaly detection method still achieved the best accuracy. However, the images from 'moving camera' result in less accuracy compared to images from 'static camera.' This is because the machine learning system was trained based on simulated images that were captured from a full view of the object, as shown in **Fig. 4.1**, while the moving camera images contain only a part of the object as shown in **Fig. 4.2**. Also, the static view images are clearer because the moving camera creates blur images as illustrated in **Fig. 4.3** that decreases accuracy. The images captured by static camera as shown in **Fig. 4.4** has resulted an accuracy of 96.1%.

5 Conclusion

This work aims to detect malicious infill structure during 3D printing processes using machine learning techniques. This work is based on Wu et al. [3] but extends it with i) additional machine learning methods, ii) more types of infills and iii) use of a camera in collecting and processing real images from 3D printing machines, and iv) application of machine learning to detect malicious defect in real images. The steps of installation of a camera, image processing and detection results shows the potential for implementing this 3D printing vision-based detection system in a real CyberManufacturing Systems. The following can be achieved in future: i) research on the influence on the detection accuracy by different colors and types of materials, ii) implementation of the system in differentiating natural defects in 3D printing process rather than malicious defects, and iii) additional systems and methodologies to adopt for ensuring security in CyberManufacturing systems.



4.1 Simulated image **4.2 Moving camera image**

4.3 Blur Moving camera image **4.4 Static camera image**

Figure 4. Sample images.

References

1. Z. Song and Y.B. Moon, *Proceedings of the IFIP 13th International Conference on Product Lifecycle Management*, (in press).
2. IBM, *IBM X-Force Research Report*, 2016.
3. M. Wu, V. V. Phoha, Y. B. Moon, and A. K. Belman, *Proceedings of the ASME 2016 International Mechanical Engineering Congress and Exposition*, (in press).
4. A. Bosch, A. Zisserman, and X. Munoz, *2007 IEEE 11th International Conference on Computer Vision*, 1–8, (2007)
5. L. Peterson, *K-nearest neighbor*, Scholarpedia, 2009. [Online]. Available:http://www.scholarpedia.org/article/K-nearest_neighbor. [Accessed: 23-Jun-2016].
6. V. Chandola, A. Banerjee, and V. Kumar, ACM Comput. Surv., **41-3**, 1–58, (2009)