Original Research Paper

# Defect Detection on 3D Print Products and in Concrete Structures Using Image Processing and Convolution Neural Network

**Selorm Garfo, M.A. Muktadir and Sun Yi**

*Department of Mechanical Engineering, North Carolina A&T State University, Greensboro, USA*

*Article history*
Received: 06-04-2020
Revised: 30-04-2020
Accepted: 22-05-2020

Corresponding Author:
Sun Yi
Department of Mechanical Engineering, North Carolina A&T State University, Greensboro, USA
Email: syi@ncat.edu

**Abstract:** This paper explores the automated detection of surface defects on 3-D printed products and concrete structures. They are the main factors to evaluate their quality in addition to dimension and roughness. Traditional detection by human inspectors is far from satisfactory. Manual inspection is time-consuming, error-prone and often leads to loss of resources. For this purpose, image processing and deep learning-based object detection adopted by Google Cloud Machine Learning (ML) Engine is used to detect surface defects. In the case of image processing, two approaches are presented in this paper. In both cases, pixels are being considered to differentiate a smooth or rough surface from a picture taken by a USB camera. For the deep learning- based solution, MobileNet -a base convolution neural network treated as an image feature extractor in combination with Single Shot MultiBox Detector (SSD) as an object detector hence MobileNet-SSD. The model was successfully trained on the Google Cloud ML Engine with the dataset of 20000+ images. The review of the results confirms that with the help of MobileNet-SSD can automatically detect surface defects more accurately and rapidly than conventional deep learning methods.

**Keywords:** Image Processing, Machine Learning, Tensor Flow, 3-D Printing, Additive Manufacturing

## Introduction

There is a rise in the necessity for object detection in civil infrastructure (Jahanshahi and Masri, 2012) and the manufacturing sector (Delli and Chang, 2018) in recent years. For example, according to (Jia *et al*., 2004) in the aerospace and automotive industries, materials with defects in manufacturing processes are rejected because a minor defect in a manufactured part might result in a tragedy at a later stage. 3D printing has become a popular and practical opportunity for production, in other words, it tolerates additional manufacturing decisions because it can create complex geometrical shapes with the help of Computer-Aided Design (CAD) software (Thomas Campbell *et al*., 2011). While 3D printing has already contributed extensively to several technological advances in medicine and is emerging in many other fields like fashion design and architecture (White) new computer vision solutions, such as image processing and deep learning, has caught attention in different research areas (Lecun *et al*., 2015).

Detecting defects by performing quality monitoring at various (critical) stages of the printing process not only helps in assuring corrective measures but also eliminates the waste of printing bad parts. An automatic quality check is particularly important for 3D printing machines used in mass production of the same part (Delli and Chang, 2018).

Jovančević *et al*. used a similarity measure to compare projected geometric features from CAD models with detected ones in actual images. Present similarity measure for segments is modified to ellipses. This comparison enables the detection and data association processes for navigation and inspection tasks on aircraft parts (Jovančević *et al*., 2016). Roberson *et al*. (2013) developed a decision making and ranking model for selecting an appropriate 3D printer using Deng's Similarity approach based on accuracy, printing time and product surface smoothness. Wang *et al*. (2016) proposed a segmentation method based on printing direction to improve surface quality as well as reduce printing error and time. Wu *et al*. widely presented a machine learning and image classification method to detect the infill defects in the 3D printing process. The method explored feature extraction and implementation of Naive Bayes Classifier and J48 Decision Trees algorithms (Wu *et al*., 2016).

Civil infrastructure system asset represents one of the major fractions in the United States assets according to Wu *et al.*, is estimated to be worth $20 trillion. These systems are subject to deterioration at an alarming rate (Zhishen *et al*., n.d.). Thus, effective methods for routine inspections and evaluation of the structures are needed to prolong their service life.

Visual inspection is the main approach for the inspection of almost all infrastructure systems. It is an intuitive process that depends on an inspector's training and focus, making it immensely prone to human error and sometimes raises safety issues for the inspectors. The evolution of the automated system can reduce these inadequacies (Arabi *et al*., 2020). Benning *et al*. introduced an automatic crack detection process based on photogrammetry to compute the defects of reinforced concrete surface structure. Photogrammetry is a non-contact measuring method that was utilized with an evaluation software to determine deformations in 2D. The crack pattern was extracted and improved using a finite element method (Benning *et al*., 2003). Chilukuri *et al.* (2019) used computer vision to help the elderly mobile users in virtual screening of directions using GPS, detection of uneven sidewalks, identifying the traffic signals and signboards. Abdel-Qader *et al.* (2003) analyzed the effectiveness of different edge detection methods for crack recognition and classification in concrete pavements of bridges. They deduced that the Fast Haar Transform (FHT) has the most accurate crack detection potential contrary to Fast Fourier transform, Sobel and Canny edge detection operators (Bachmann *et al*., 2000). Tsao *et al.* (1994) designed image analysis and expert system modulus to detect spalling and transverse cracks in pavements. Another example, (Jóźwik *et al*., 1999) generated a visual system that could detect cracks in ferrites. Based on top-hat transform (Salembier, 1990), detections are obtained through unbalanced changes of brightness.

Cracks can be classified from other defects such as grooves using k-Nearest Neighbors classifier (Duda *et al*., 2001). This method is comprehensively considered robust notwithstanding the existence of noise, as opposed to other edge detection operators used for crack extraction. Although almost all the mention machine learning endeavors focused on providing solutions regarding object detection, the computing time was disregarded.

Surface defect detections solve a lot of additive manufacturing problems. For example, in the tiling industry, pattern recognition and image processing algorithms have been used to detect surface defects (Karimi and Asemani, 2014). A basic for detecting the image defect is the edge which is regarded as a boundary between two dissimilar regions in an image and easy to find (Elbehiery *et al*., 2005). To detect wood surface defects, another study proposed an image segmentation method called fuzzy min-max neural network for image segmentation. This method grows boxes from a set of pixels, to find the minimum bounded rectangle for each defect present in the image. The rate of defect detection is 95% and the area recognition rate is 94.4% which is the measure of the quality of segmentation. Both results are for a specific data set (Ruz *et al*., 2005).

Quality control plays an ever-increasing role in mass production and preventing structure failure. Defects on the surface are main factors to evaluate the quality of 3-D printed products and concrete structures, therefore it requires automated analysis to improve their quality (Gordeev *et al*., 2018). Traditional visual surface detection is still carried out by human operators whose performance is generally inadequate, subjective and variable (Zheng *et al*., 2002). According to some studies, a human visual inspection can only catch around 60% to 75% of the significant defects (Schicktanz, 1993). Therefore, developing automatic surface inspection systems is in significant demand.

Based on considerable literature review, it is clear that the performance for most of the studies in surface defect detection needs to be further validated and optimized. In efforts to train for object detection using a local Computer Processing Unit (CPU) computer, we found that the computational loads are substantially heavy which takes CPU days to train. (Training refers to the computer memory task in preparation for object recognition). The project requirements include an active subscription with Google Cloud, TensorFlow (Abadi *et al*., 1983), Python and OpenCV. The experiment was implemented using transfer learning of a pre-trained model called SSD with MobileNet (Haridas and Sandhiya, 2018). By enhancing the convolution neural network framework and parameters with the aid of Google Cloud ML Engine. The reason behind using Tensor Processing Unit (TPU) is due to the Computer Processing Unit not being powerful enough. This method can reach the requirements of accuracy and accelerated training time in deployment. This process was validated with the results showing the classifying and tracking of surface defects.

## Materials and Methods

As a 3-D printer, in this experiment, a MakerBot 3D printer and a USB camera (13MP**,** FOV 75Degree Autofocus USB Camera with Non-Distortion Lens) have been used. Figure 1 shows the experimental setup.

### *Edge-Detection Method*

To find out the printed model defects, two techniques have been used whose basics are implemented through image processing. A flow chart (Fig. 2) shows the working principle of these techniques. In both cases, pictures are taken manually after the printing of the model. In first technique (experiment 1, 2, and 3) in Fig. 3 to 5, the algorithm breaks the cropped picture into four parts and calculate the number of pixels of each part and compared for identifying surface smoothness whereas in the second technique (experiment 4, 5, and 6) in Fig. 6 to 8, the algorithm breaks the RGB image into Grayscale image for calculating white and black pixels.

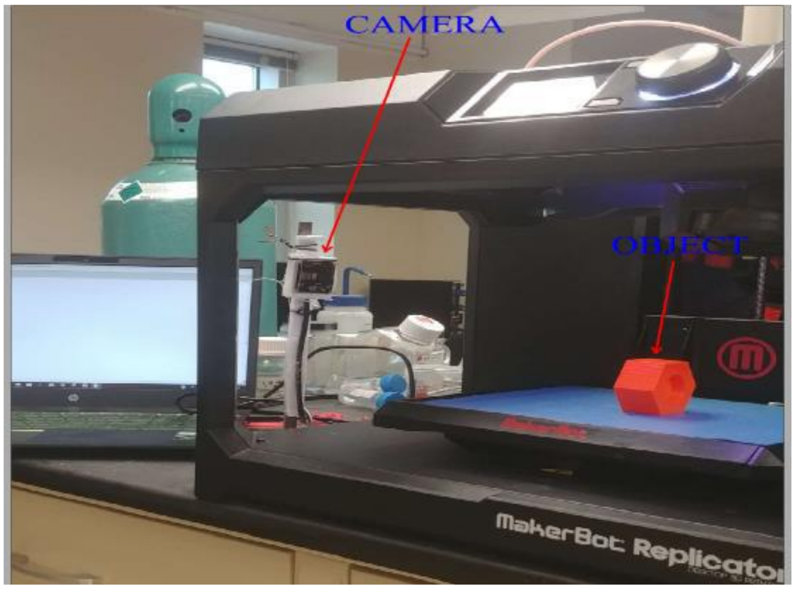


**Fig. 1:** Experimental setup

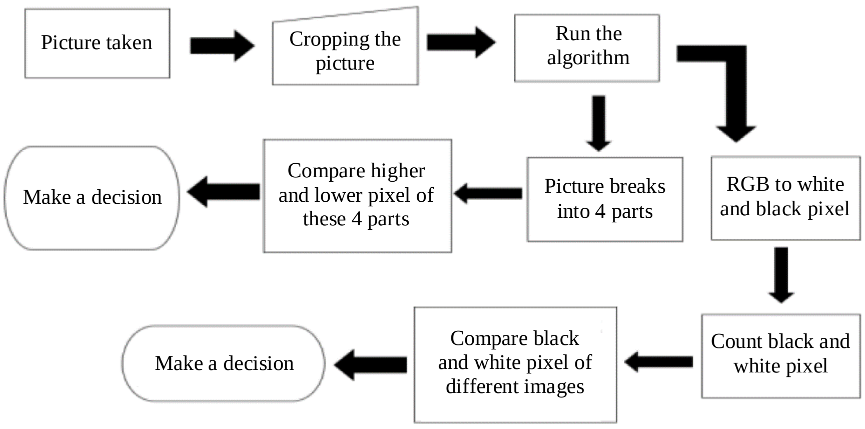

Picture taken
Cropping the picture
Run the algorithm
Picture breaks into 4 parts
RGB to white and black pixel
Compare higher and lower pixel of these 4 parts
Make a decision
Count black and white pixel
Compare black and white pixel of different images
Make a decision

**Fig. 2:** Flow chart of different stages of image analysis

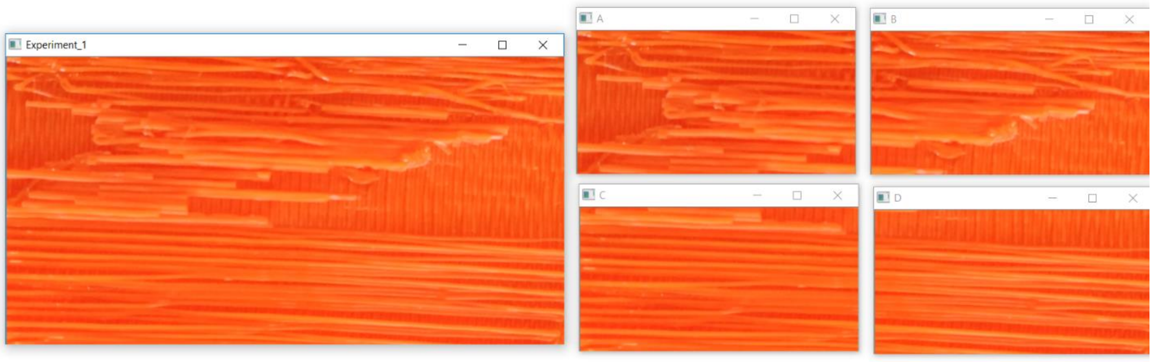


**Fig. 3:** Experiment 1 (Surface roughness is higher compared to experiment 2 (Fig. 4) and 3 (Fig. 5))

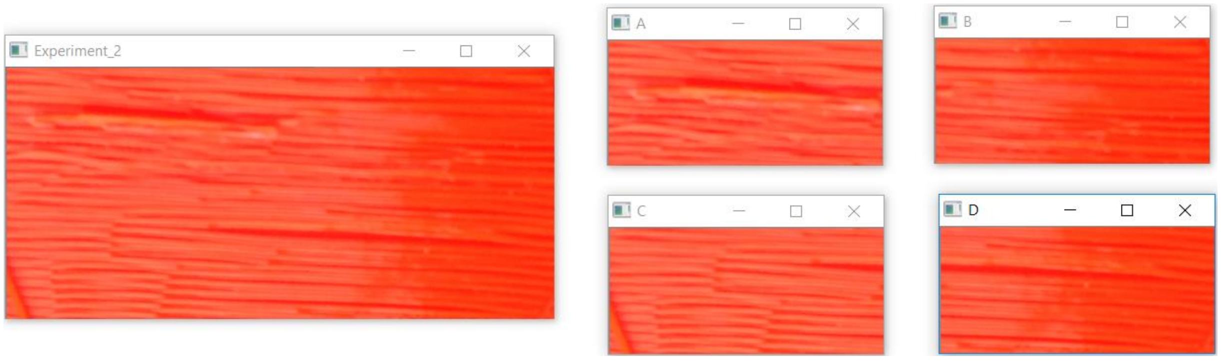


**Fig. 4:** Experiment 2 (Surface roughness is moderate compared to experiment 1 (Fig. 3) and 3 (Fig. 5))

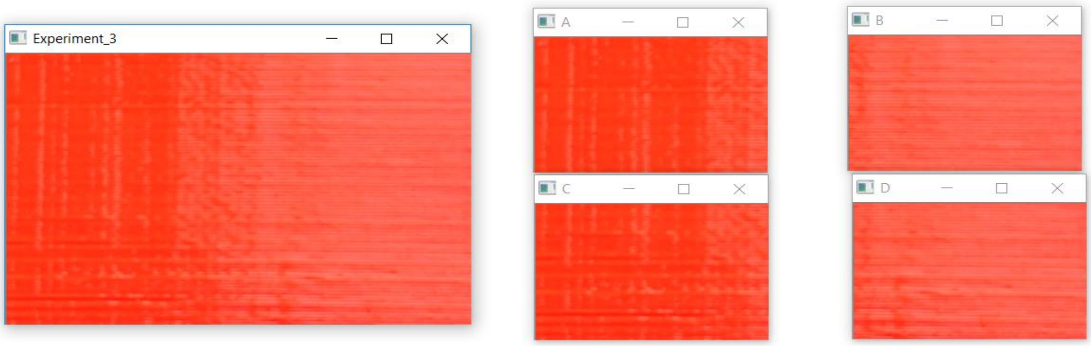


**Fig. 5:** Experiment 3 (Surface roughness is lower compared to experiment 1 (Fig. 3) and 2 (Fig. 4))

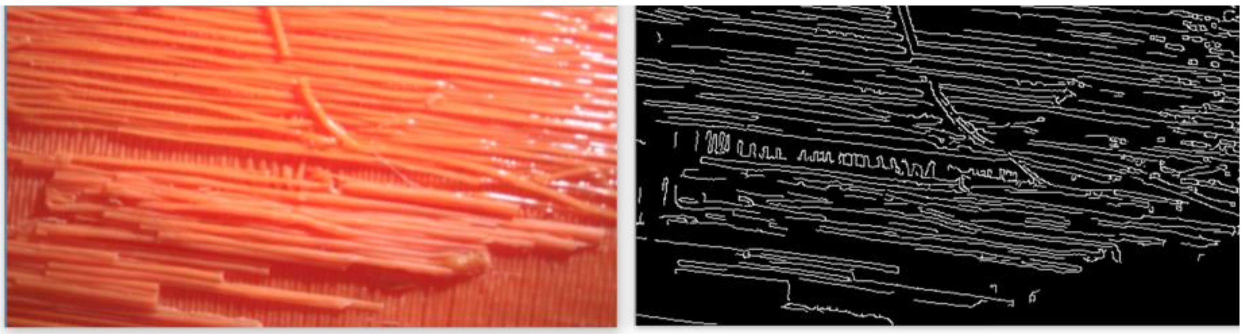


**Fig. 6:** Experiment 4 (Surface roughness is higher compared to experiment 5 (Fig. 7) and 6 (Fig. 8))

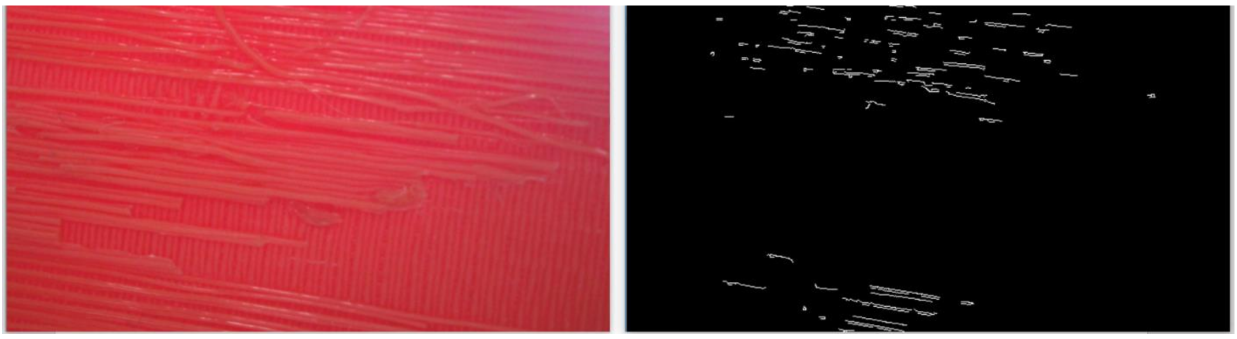


**Fig. 7:** Experiment 5 (Surface roughness is moderate compared to experiment 4 (Fig. 6) and 6 (Fig. 8))

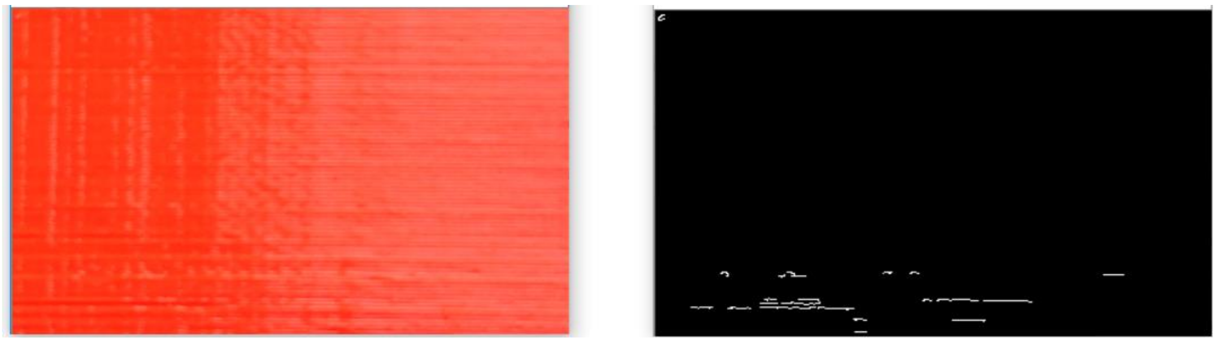


**Fig. 8:** Experiment 6 (Surface roughness is lower compared to experiment 4 (Fig. 6) and 5 (Fig. 7))

The number of white and black pixels between different images decides whether the surface is smooth or not. In both cases, python and OpenCV have been used.

### *Machine Learning Method*

### *Model Development*

The process flowchart is illustrated in Fig. 9. This section describes setting up the dataset (images and annotations) as described in Dataset Preparation (Images and Annotations) section, Feature extraction and object detection using MobileNet-SSD were outlined in MobileNet Feature Extraction sections and SSD Structure sections respectively. In Surface Defects Detection using Mobilenet-SSD Detection Model section, the framework of the model was briefly described.

### *Dataset Preparation (Images and Annotations)*

The 3-D prints images were obtained from taking photos with a USB camera and a smartphone. The images are saved as a jpeg format into a folder. A total of 20000+ images were taken. These images were converted to 300×300 pixels as inputs for training. After visual inspection of these images and approval of the quality, they are annotated.

LabelImg was the software used for labelling defects in the dataset. LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. It operates by drawing a bottom box around each object in each image and repeating the process for all images. LabelImg saves a .xml file containing the label data for each image.

These annotation files include the bounding box coordinates where the specific wall crack or 3-D defect is located in the image. They are converted into TFRecords.

### *MobileNet Feature Extraction*

MobileNet model was developed to optimize deep learning which requires real-time performance due to lower hardware specifications. This network was introduced to reduce the number of parameters while optimizing accuracy. A default single filter is applied to each neural input channel to begin feature extraction in the MobileNet model. Figure 9 shows the basic convolution block of MobileNet.

The model's basic convolution block is a depth-wise separable convolution structure that was initially proposed by (Sifre, 2014). For MobileNets, every input channel is subjected to a single filter in the depthwise convolution. The pointwise convolution then uses a 1×1 convolution to combine the information of the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. It is composed of Depthwise layers (Dw) and Pointwise layers (Pw). It is a form of factorized convolutions have in which the standard convolution is factored in the Dw and Pw. The Dw are deep convolutional layers using 3×3 kernels, while the Pw are common convolutional layers using 1×1 kernels.

Each result is treated by default mechanisms such as Batch normalization (Bn) algorithm and the activation function Rectified Linear Unit (ReLU) (Fig. 10). In this study, the activation function is replaced as ReLU6. The feature map output for standard convolution is calculated as:

$$G_{k,l,n} = \sum_{i,j,p} K_{i,j,p,n} \times F_{k+i-1,l+j-1,p}$$

where, $G$ is the output feature map, $K$ is the standard convolutional kernel and $F$ is the feature map. The reasons behind the increased training speed and a reduced amount of calculation are as follows.

For standard convolution, when the size and number of input channels are $D_F^2$ and $M$ respectively, it is required to have $N$ filters with $M$ channels and the size $D_K^2$ before output in and feature images of size $D_K^2$. The computation cost is $D_K^2 \times M \times N \times D_F^2$ where $D_K$ is kernel size, $D_F$ is square input feature map spatial size, $M$ and $N$ is the number of input channels and number of output channels respectively.

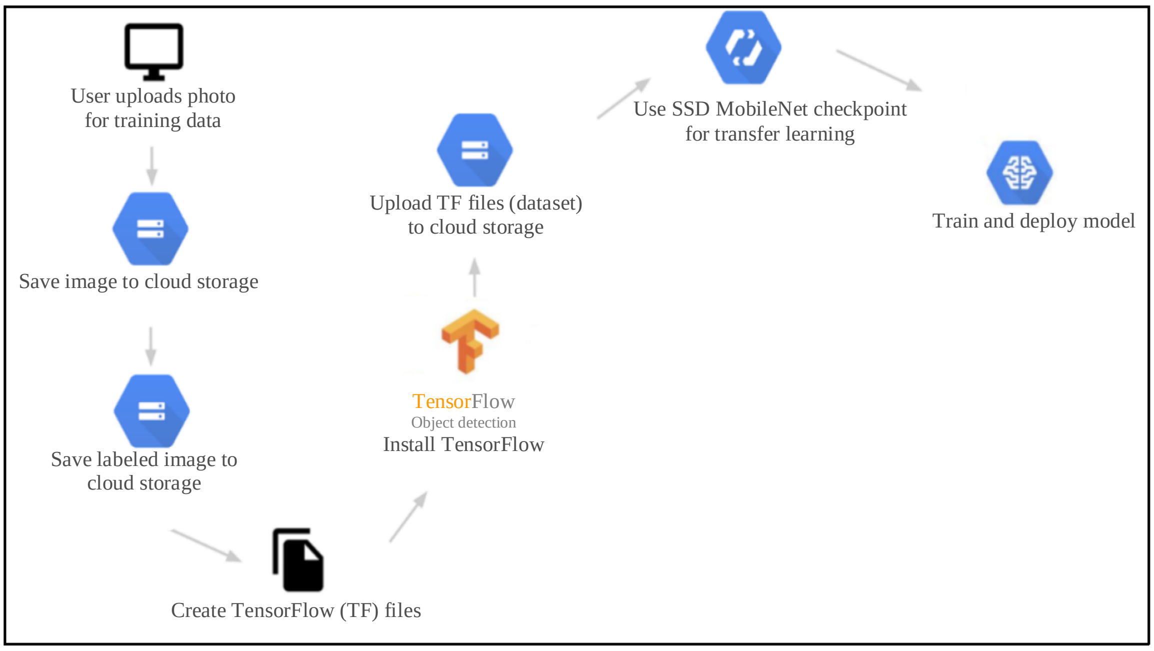

User uploads photo for training data
Save image to cloud storage
Save labeled image to cloud storage
Create TensorFlow (TF) files
TensorFlow
Object detection
Install TensorFlow
Upload TF files (dataset) to cloud storage
Use SSD MobileNet checkpoint for transfer learning
Train and deploy model

**Fig. 9:** The flowchart of model

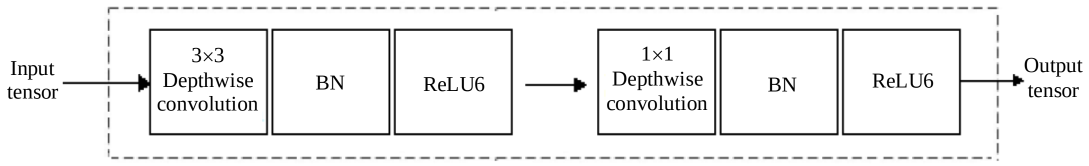


**Fig. 10:** Depthwise Convolution Block with depthwise and Pointwise layers followed by batchnorm and ReLU

Contrarily, stepwise separable convolution requires $M$ filters with one channel and the size of $D_F^2$. The pointwise phase needs $KN$ filters with $M$ channels and the size of 1×1. Depthwise convolution with one filter per input channel (input depth) can be written as:

$$\hat{G}_{k,l,p} = \sum_{i,j} \hat{K}_{i,j,p} \times F_{k+i-1,l+j-1,p}$$

where, $\hat{K}$ is the depthwise convolutional kernel of size $D_K^2 \times M$ where the $p_{th}$ filter in $\hat{K}$ is subjected to the $p_{th}$ channel in $F$ to produce the *pth* channel of the filtered output feature map $\hat{G}$ (Howard *et al*., 2017). The computation cost of depthwise separable convolution is $D_K^2 \times M \times D_F^2 + M \times N \times D_F^2$. It can be shown that stepwise is reduced by of standard convolution $\frac{1}{N} + \frac{1}{D_K^2}$ of standard convolution (Li *et al*., 2018).

*SSD Structure*

The Single Shot Multibox detector is a regression model, which utilizes features of various convolution layers to produce classification regression and boundary box regression. With each selected, we associate a set of default bounding boxes. The default boxes tile the feature Maps of different convolution layers. Each bounding box predicts $c$ class scores and 4 offsets relative so the original default bounding box shape yielding $(c + 4)kmn$ outputs. Where $c$ is number of classes, $k$ is Number of default bounding boxes, $mn$ is the feature map size. For each feature map, the scale of the default boxes is expressed as:

$$S_K = S_{\min} + \frac{S_{\max} - S_{\min}}{m-1}(k-1)\{k \in [1,m]\} \quad (1)$$

where $m$ is the number of feature maps, $S_{\min}(0.2)$ and $S_{\max}(0.9)$ are scale for the highest and lowest feature map respectively, $k$ is the number of defaults bounding boxes.

The 5 different types of width to height (aspect) ratios are expressed as:

$$a_r \in \left\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\right\}$$

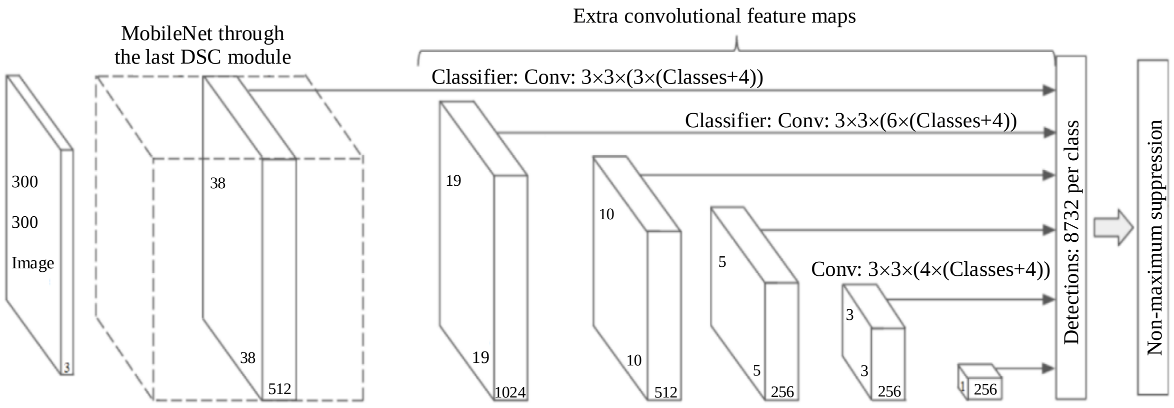

Extra convolutional feature maps
MobileNet through the last DSC module
Classifier: Conv: 3×3×(3×(Classes+4))
Classifier: Conv: 3×3×(6×(Classes+4))
Conv: 3×3×(4×(Classes+4))
300
300
Image
3
38
38
512
19
19
1024
10
10
512
5
5
256
3
3
256
1
256
Detections: 8732 per class
Non-maximum suppression

**Fig. 11:** MobileNet- SSD structure (Driaba *et al*., 2019)

And the width $\left(w_k^a = S_K\sqrt{a_r}\right)$ and height $\left(h_k^a = S_K\sqrt{a_r}\right)$ for each default box.

Next, a default box $S_k' = \sqrt{S_K\ S_{K+1}}$ should be added when the aspect ratio is 1 result in 6 default boxes per feature map location. The center of each default box is set to $\left(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|}\right)$, where |*f*| is the size of the *k*th feature units, *i*,*j* ϵ [0,|*f*$_k$|].

The Jaccard overlap (intersection over union, IoU) between area *X* and *Y* can be calculated as:

$$Jaccard\ Overlap\ (IoU) = \frac{Area\ of\ intersection}{Area\ of\ union} = \frac{Area(X)\cap Area(Y)}{Area(X)\cup Area(Y)}$$

If the IoU of default box and the ground truth box (calibration box) is 0.5 or greater, it means the boxes match in that category. The overall objective loss function of the training is the weighted sum of confidence losses $L_{conf}(s,c)$ of the classification regression and the localization loss of the bounding box regression $L_{loc}(r,l,g)$:

$$L(s,r,c,l,g) = \frac{1}{N}\left[L_{conf}(s,c) + \alpha L_{loc}(r,l,g)\right]$$

where, $\alpha$ is the parameter to balance the confidence loss and localization loss; *s* and *r* are the eigenvectors of confidence loss and localization loss respectively; *c* is the classification, *l* is the offset of predicted box, including the translational offset of the center coordinate and scaling offset of the height and width; *g* is the calibration box of the target actual position; and *N* is the number of default boxes that match the calibration boxes of this category.

### *Surface Defects Detection using Mobilenet-SSD Detection Model*

The complete model contains four (4) parts; the input layer for importing converting the target image into an array of pixels, the MobileNet based network employed to extract significant image features and SSD which utilizes MobileNet feature map outputs to produce classification and bounded box regression and the output layer for exportation of detection results. Figure 11 illustrates the structure of MobileNet-SSD.

## Results and Discussion

From the results of experiments 1,2, and 3 (Table1), as the surface becomes smoother, the difference between the maximum and minimum values of pixels decreased. Conversely, the results of experiments 4,5, and 6 (Table 2) shows that the percentage of white pixels decreases while the roughness of images (defects) decreases.

In deep learning model evaluation metrics, the percentage of the images used in the current training batch labeled with the correct class establishes training accuracy. The evacuation accuracy is the accuracy of a different group of images selected at random. The major difference is that the training accuracy is based on images that the network has been able to learn from so the network can overfit to the noise in the training data. A true measure of the performance of the network is to measure its performance on a data set not contained in the training data -- this is measured by the evacuation accuracy (Donahue *et al*., 2014).

The network is overfitting if the training accuracy is high, but the evacuation accuracy remains low. The training's objective is to make the loss as small as possible (Abadi *et al*., 1983).

Training and evaluation completed in approximately 33 min (Fig. 12).

The images set consists of 20000+ images of 2 different objects. The image set was tested by randomly sampling a number of images to train the object recognition and using the remaining images for testing. After training, the model achieved 80% mean average precision as illustrated in Fig. 13.

TensorBoard is a tool for providing common machine learning metrics used to visualize the graph and statistics, such as how the weights or accuracy varied during training Fig. 13 and 14.

Mean Average Precision measures the model's percentage of correct predictions for all labels. Intersection-over-Union (IoU) is specific to object detection models. This measures the overlap between the bounding box generated by the model and the GroundTruth bounding box represented as a percentage. The model achieved an 80% mean average position after training. On the Images tab in TensorBoard the model's predictions for this image are on the left and on the right the correct GroundTruth box.

The machine learning method presents a system architecture, implemented model and analysis for Google-Cloud based object detection. Object detection is performed in the cloud using Google's Cloud ML Engine. We incorporated MobileNet-SSD from TensorFlow (Abadi *et al.*, 1983) - a pre-trained model for transfer learning (Haridas and Sandhiya, 2018). Its checkpoints from the already trained models are used and then applied in our custom object detection task. The advantage of it is instead of building the model from scratch, a model trained for a similar problem can be used as a starting point for training the network. While we are encouraged by initial results i.e., Fig. 15a and 15b, much remains to be done to allow such a system to be scaled up for many other datasets.

Deep learning models have two discrete computational components: Training and inference (Benning *et al.*, 2003). Because Cloud TPUs are used to accelerate this training, the config file must relate specifically to TPU training.

The training process was carried out on the following hardware: GeForce TITAN X graphics processing unit (GPU), Intel Core i7 3.2 GHz, 16 GB memory processor. TensorFlow-GPU 1.12, CUDA 10.0, Anaconda 3 were the software used.

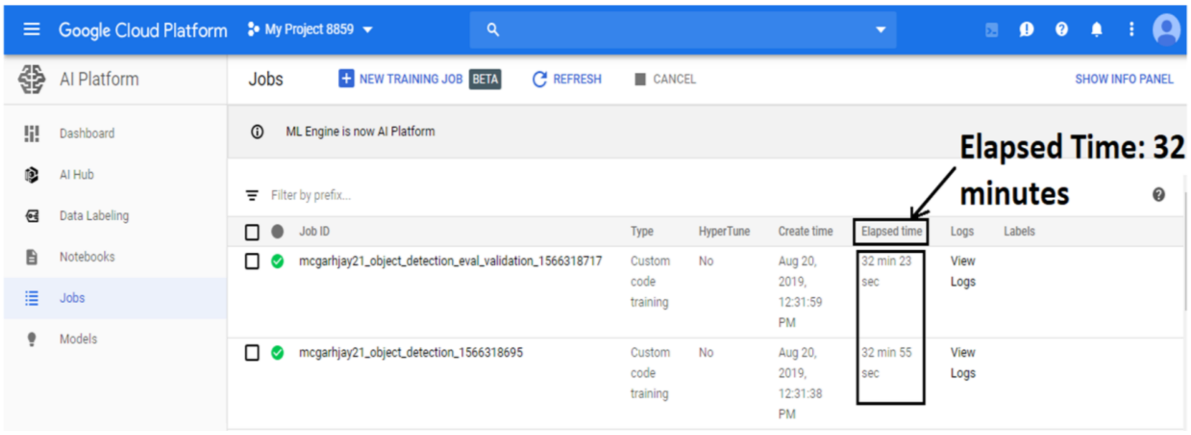
Image processing and machine learning methods can be used in different situations, for example for a small model which takes a short time to print a 3D model we can use image processing but for the bigger model where it takes a longer time we can use machine learning to save resources. Also, image processing is better for more varying images and thus can be deployed in other projects in contrast machine learning.

**Table 1:** Output of experiments 1,2, and 3

| Experiment | Pixels (Part A) | Pixels (Part B) | Pixels (Part C) | Pixels (Part D) | Diff. between max. and min |
|---|---|---|---|---|---|
| 1 | 161,352 | 161,850 | 162,324 | 162,825 | 1,473 |
| 2 | 68,907 | 68,907 | 69,576 | 69,576 | 669 |
| 3 | 73,875 | 73,875 | 74,466 | 74,466 | 591 |

**Table 2:** Output of experiments 4,5, and 6

| Experiment | No. of black pixels (0,0,0) | No. of white pixels (255,255,255) | Percentage of white pixels (%) |
|---|---|---|---|
| 4 | 133,166 | 18,805 | 12.37 |
| 5 | 232,824 | 2,423 | 1.02 |
| 6 | 98,516 | 378 | 0.38 |



| Job ID | Type | HyperTune | Create time | Elapsed time | Logs | Labels |
|---|---|---|---|---|---|---|
| mcgarhjay21_object_detection_eval_validation_1566318717 | Custom code training | No | Aug 20, 2019, 12:31:59 PM | 32 min 23 sec | View Logs | |
| mcgarhjay21_object_detection_1566318695 | Custom code training | No | Aug 20, 2019, 12:31:38 PM | 32 min 55 sec | View Logs | |

**Fig. 12:** Compute time of training

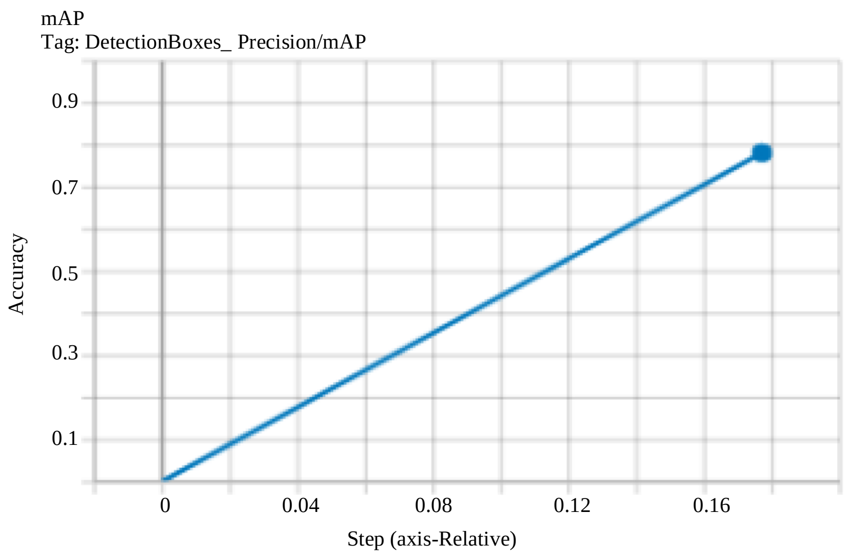


**Fig. 13:** Model accuracy vs steps

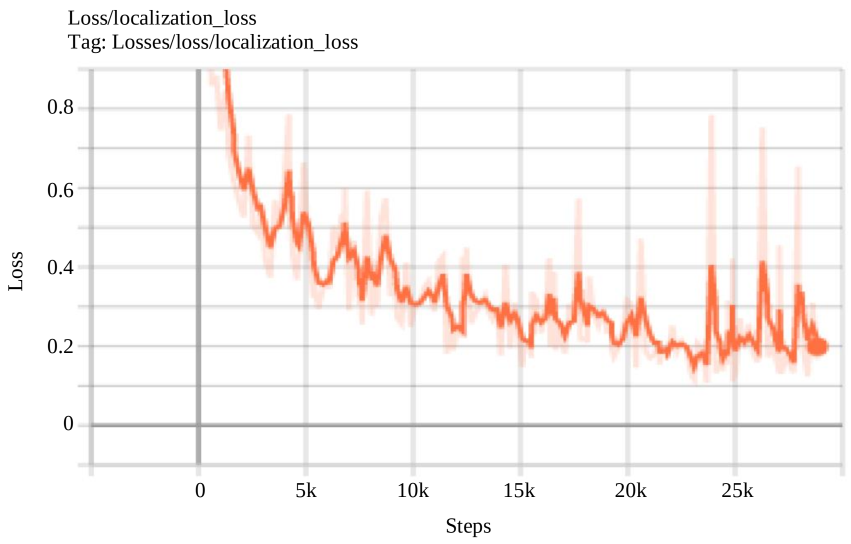
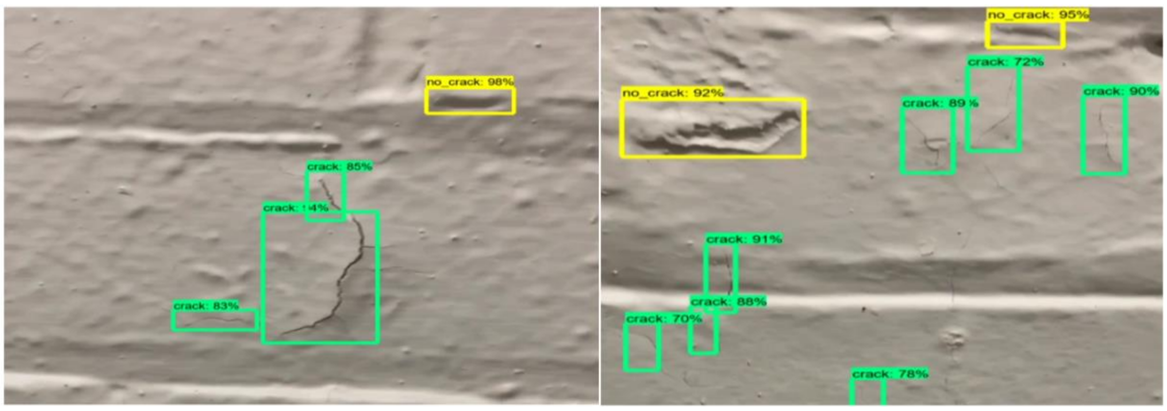
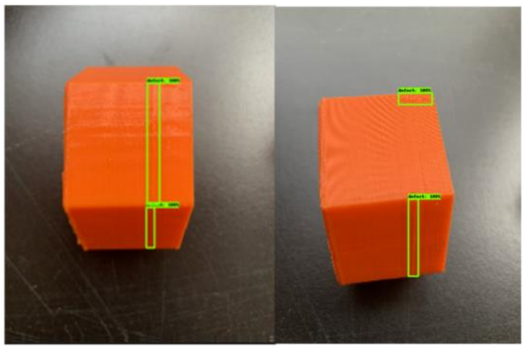


**Fig. 14:** Localization loss at ≈ 29×10$^3$ step



(a)



(b)

**Fig. 15:** (a) Defective areas(cracks) are marked out by a bounding box; (b): Bounding boxes of 3-D defects

## Conclusion and Future Work

While results of edge detector experiments may vary depending on many factors like the position of the camera, light intensity and the color of the 3D objects, pixel intensity comparisons assist in identifying surface discontinuities and defects.

The deep learning model was successfully trained on the Google Cloud ML Engine with the dataset of 20000+ images. With the help of MobileNet-SSD, the outputs, each one showing high training accuracy and evaluation accuracy. The use of robust machine learning algorithms along with image processing techniques to automatically detect defects (surface and cracks) in 3-D printed products and concrete structures has a great potential to affect how quality control is maintained. It was discovered that the use of Google Cloud ML Engine and MobileNet-SSD was able to significantly improve the accuracy and training time of the model. When the model was deployed, the model was able to perform real-time defect detection from video footage with the highest accuracy. Clearly, the key to improving image processing and object detection algorithms is to create ensembles from multiple diverse models. More training images, especially of minute 3D surface defects and wall cracks, could be used to further train and enhance the models. A larger dataset would also allow for the detection of a greater number of defect categories, permitting the identification of unique brands of the defect. Furthermore, a higher-resolution camera would allow for a more accurate and faster analysis of the footage. If the research of this project is continued, the defect detection system could be implemented into real-world applications such as mass product inspection. With the installation of an Inertial Measurement Unit (IMU) and depth camera, robots and drones could report the precise location of early-stage wall cracks to prevent structural failure. Additionally, drones could be programmed with autonomous flight features to easily navigate around and find 3D surface defects without relying on traditional detection by human inspectors.

In our future work, we will seek to improve each aspect of the system. For image recognition, more details of the surface defects will be pursued in addition to incorporate more sophisticated analysis of these defects as they are data specific.

## Acknowledgement

Thanks to Dr. Salil Desai and Santosh Kumar Parupelli, who assisted in printing 3D objects.

## Funding Information

The authors gratefully acknowledge the support of the Department of Energy/National Nuclear Security Administration (DOE/NNSA) for funding this project (Award number: DE-NA0003686), and NSF Engineering Research Center.

## Author’s Contributions

All authors equally contributed to this work.

## Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

## References

Abadi, M., A. Agarwal, E.B.P. Barham, A.D.Z. Chen and C. Citro *et al*., 1983. TensorFlow: Large-scale machine learning on heterogeneous systems. Methods Enzymol.

Abdel-Qader, I., O. Abudayyeh and M.E. Kelly, 2003. Analysis of edge-detection techniques for crack identification in bridges. J. Comput. Civil Eng.

Arabi, S., A. Haghighat and A. Sharma, 2020. A deep-learning-based computer vision solution for construction vehicle detection. Comput. Aided Civil Infrastructure Eng.

Bachmann, G., L. Narici and E. Beckenstein, 2000. Fourier and Wavelet Analysis. 1st Edn., Springer, New York, ISBN-13: 978-1-4612-6793-5.

Benning, W., S. Görtz, J. Lange, R. Schwermann and R. Chudoba, 2003. Development of an algorithm for automatic analysis of deformation of reinforced concrete structures using photogrammetry. VDI Berichte, 1757: 411-418.

Chilukuri, D., S. Yi and Y. Seong, 2019. Computer vision for vulnerable road users using machine learning. J. Mech. Robot., 3: 33-41.
DOI: 10.3844/JMRSP.2019.33.41

Delli, U. and S. Chang, 2018. Automated process monitoring in 3d printing using supervised machine learning. Proc. Manufactur., 26: 865-70.
DOI: 10.1016/J.PROMFG.2018.07.111

Donahue, J., Y. Jia, O. Vinyals, J. Hoffman and N. Zhang *et al*., 2014. DeCAF: A deep convolutional activation feature for generic visual recognition. Proceedings of the 31st International Conference on Machine Learning, (CML' 14), PMLR.

Driaba, A., A. Gordeev and V. Klyachin, 2019. Recognition of various objects from a certain categorical set in real time using deep convolutional neural networks. CEUR Work. Proc., 2500: 4-9.

Duda, R.O., P.E. Hart and D.G. Stork, 2001. Pattern Classification. 2nd Edn., Wiley, NY,
ISBN-13: 978-0-471-05669-0.

Elbehiery, H., A. Hefnawy and M. Elewa, 2005. Surface defects detection for ceramic tiles using image processing and morphological techniques. Proceedings of the 3rd World Enformatika Conference, Apr. 27-29, Istanbul, Turkey, 158-62.
DOI: 10.5281/ZENODO.1084534

Gordeev, E.G., A.S. Galushko and V.P. Ananikov, 2018. Improvement of quality of 3D printed objects by elimination of microscopic structural defects in fused deposition modeling. PLoS ONE, 13: e0198370-e0198370.
DOI: 10.1371/JOURNAL.PONE.0198370

Haridas, N. and S. Sandhiya, 2018. Traffic light detection using the TensorFlow object detection API.

Howard, A.G., M. Zhu, B. Chen, D. Kalenichenko and W. Wang *et al*., 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications.

Jahanshahi, M.R. and S.F. Masri, 2012. Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures. Automat. Constr., 22: 567-576.
DOI: 10.1016/J.AUTCON.2011.11.018

Jia, H., Y.L. Murphey, J. Shi and T.S. Chang, 2004. An intelligent real-time vision system for surface defect detection. Proceedings of the International Conference on Pattern Recognition, Aug. 26-26, IEEE Xplore Press, UK. DOI: 10.1109/ICPR.2004.1334512

Jovančević, I., I. Viana, J.J. Orteu, T. Sentenac and S. Larnier, 2016. Matching CAD model and image features for robot navigation and inspection of an aircraft. Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods, (RAM' 16), Rome, Italy.

Jóźwik, A., M. Nieniewski, L. Chmielewski and M. Skłodowski, 1999. Morphological detection and feature-based classification of cracked regions in ferrites. Mach. Graph. Vis., 8: 699-712

Karimi, M.H. and D. Asemani, 2014. Surface defect detection in tiling industries using digital image processing methods: Analysis and evaluation. ISA Trans., 53: 834-844.
DOI: 10.1016/J.ISATRA.2013.11.015

Lecun, Y., Y. Bengio and G. Hinton, 2015. Deep learning. Nature, 521: 436-444.
DOI: 10.1038/NATURE14539

Li, Y., H. Huang, Q. Xie, L. Yao and Q. Chen, 2018. Research on a surface defect detection algorithm based on MobileNet- SSD. Applied Sci., 8: 1678-1678. DOI: 0.3390/APP8091678

Roberson, D.A., D. Espalin and R.B. Wicker, 2013. 3D printer selection: A decision-making evaluation and ranking model. Virtual Phys. Prototyp., 8: 201-212.
DOI: 10.1080/17452759.2013.830939

Ruz, G.A., P.A. Estévez and C.A. Perez, 2005. A neurofuzzy color image segmentation method for wood surface defect detection. Forest Products J., 55: 52-58.

Salembier, P., 1990. Comparison of some morphological segmentation algorithms based on contrast enhancement: Application to automatic defect detection.

Schicktanz, K., 1993. Automatic fault detection possibilities on nonwoven fabrics. Melliand Textilberichte.

SIfre, L., 2014. Rigid-motion scattering for image classification. PhD Thesis.

Thomas Campbell, C.W., O. Ivanova and B. Garrett, 2011. Could 3D printing change the world? Atlantic Council.

Tsao, S., N. Kehtarnavaz, P. Chan and R. Lytton, 1994. Image-based expert-system approach to distress detection on CRC pavement. J. Trans. Eng.

Wang, W.M., C. Zanni and L. Kobbelt, 2016. Improved surface quality in 3D printing by optimizing the printing direction. Comput. Graph. Forum., 35: 59-70. DOI: 10.1111/CGF.12811.

White, G., The pros and cons of 3D printing. https://www.manufacturingglobal.com/

Wu, M., V.V. Phoha, Y.B. Moon and A.K. Belman, 2016. Detecting malicious defects in 3D printing process using machine learning and image classification. ASME Int.

Zheng, H., L.X. Kong and S. Nahavandi, 2002. Automatic inspection of metallic surface defects using genetic algorithms. J. Mater. Proc. Technol., 125: 427-433.
DOI: 10.1016/S0924-0136(02)00294-7

Zhishen, W., J. Zhang and M. Noori, 2018. Fiber- Optic Sensors for Infrastructure Health Monitoring. 1st Edn., Momentum Press, ISBN-10: 194561224X, pp: 186.