Solid Freeform Fabrication 2023: Proceedings of the 34th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

Automated layer identification in large area additive manufacturing (LAAM): A comparison of image thresholding and edge detection techniques

A. Wadidie*, G. M. Studer*, K. Villez**

*University of Maine Advanced Structures and Composites Center, Orono, ME, 04473

**Oak Ridge National Laboratory, Oak Ridge, TN, 37830

Abstract

Our study aims to develop an automated method for identifying layers on images of 3D-printed walls from a LAAM printer, as manual identification is subjective and can be time-consuming. We applied three different image processing methods to identify edges between layers: simple thresholding, Otsu thresholding, and Canny edge detection. Otsu thresholding was found to be the most accurate and required minimal manual intervention. From our study, we propose a new approach by going through essential steps for greater accuracy. This research demonstrates the feasibility of using computer-based methods to automatically identify layers in 3D printing, reducing manual time and effort and improving the strength and quality of 3D-printed parts.

Introduction

The advancements in Large Area Additive Manufacturing (LAAM) have ushered in a new era of efficient and cost-effective production of intricate three-dimensional structures. As LAAM structures are composed of layers of thermoplastic filaments, often planar, a significant factor in how a LAAM structure performs is the quality and consistency of adhesion at the layer interfaces. In most cases, it is very easy to identify the layer's boundaries on the surface of a 3D-printed object with human eyes. However, the precise localization of all layers and boundaries on the surface of LAAM-printed objects is a complex and time-consuming task when performed manually, as a single print may contain hundreds of layers, each potential meters long.

The development of an automated way to identify layers in 3D-printed walls is motivated by the need to quantify layer and boundary consistency while avoiding the time-consuming nature of manual identification. By accurately and automatically identifying layer boundaries, it becomes possible to detect anomalies as well as to analyze and optimize the printing conditions. In addition, the quality of layer adhesion and layer defects affect the overall quality, structural integrity, and many other mechanical properties of the printed object. Manual identification of layers in 3D-printed LAAM walls is performed at the Advanced Structures and Composites Center (ASCC), but the process is subjective, tedious, error-prone, and not scalable for large-scale or time-sensitive projects. By developing an automated method, we aim to streamline the process and eliminate human subjectivity in identification. An automated method could be applied to a large number of images in a shorter time frame, making it more efficient and practical for industrial applications and real-time analysis and optimization.

To develop an automatic layer identification method, our research is currently focused on state-of-the-art image processing techniques. We execute a comparative analysis of image thresholding and edge detection techniques. Three different methods are used to identify edges between layers in images of 3D-printed walls obtained from a LAAM printer. Initial analysis revealed that Otsu thresholding, a well-established image segmentation technique, exhibited the highest accuracy in identifying layer boundaries. Notably, it required minimal manual intervention, making it an attractive choice for practical implementation within LAAM processes. Each of these techniques is integrated into an image-to-layer data pipeline that is composed of a 4-step sequence of data processing steps.

Two relevant developments that have influenced our research on layer identification in Large Area Additive Manufacturing (LAAM) are the *Peregrine* software [1]" and *digital image correlation* [2]." The *Peregrine* software is designed for in-situ quality control in powder bed systems and introduces AI-based software for defect detection in the top layer of a powder bed system, whereas *digital image correlation* addresses in-situ monitoring of residual thermal stresses and warpage in polymer composite printed components using digital image correlation. We have taken inspiration from these studies and developed a new method for the specific purpose of automated identification of layer boundaries in LAAM structures, which has not been addressed before.

Materials and Methods

Data collection

We used rectangular, vertically-printed 3D-printed walls as the primary material for our layer identification work (Figure 1). These walls were produced using a LAAM printer as part of an effort to produce material test specimens from ABS thermoplastic infused with carbon fiber, creating a black appearance, and were composed of 120 layers, each about 5mm high. The images were captured using a high-resolution camera at 8192 x 5464 pixels with overhead, uncontrolled overhead lighting from the facility ceiling. The wall itself was the primary feature in the camera scene and covered about 2/3rds of the image. We sampled from three different images with representative edge identification features. The layers of the walls were generally printed imperfectly and included many variations and inconsistencies commonly encountered in LAAM. In addition, the various walls were printed at different speeds, resulting in further inconsistency.

Other available LAAM materials included wood-fiber composites, which are more reflective but have a less smooth texture - these are considered for future work.

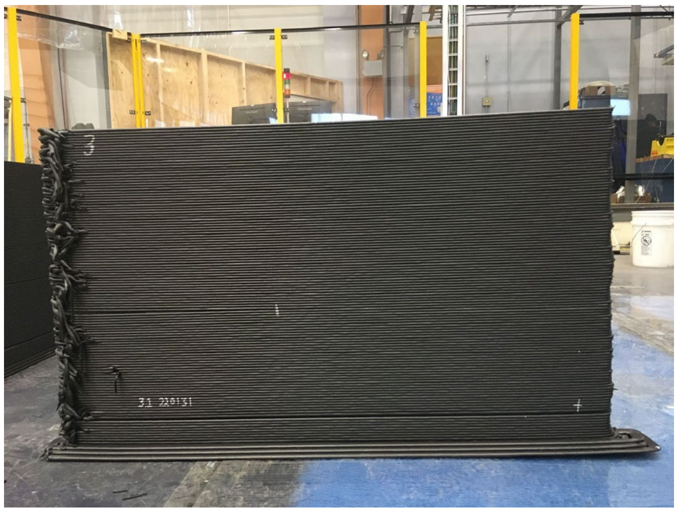

Figure 1: 3D Printed Wall

Data Preprocessing and Tiling

To avoid image segmentation complications and focus on layer boundaries, the first step was to remove the background from our input images. As you can see in Figure 1, there are different types of objects around the wall, and the wall itself contains painted writing and ragged areas. To tackle this, we initiated the process by cropping the images and selecting specific regions that could be accurately labeled, effectively removing the background and the irregular portions on the wall images (Figure 2).

Each image is sliced into 128x128 tiles to be used when proceeding to the automated layer identification for consistency in our metrics.

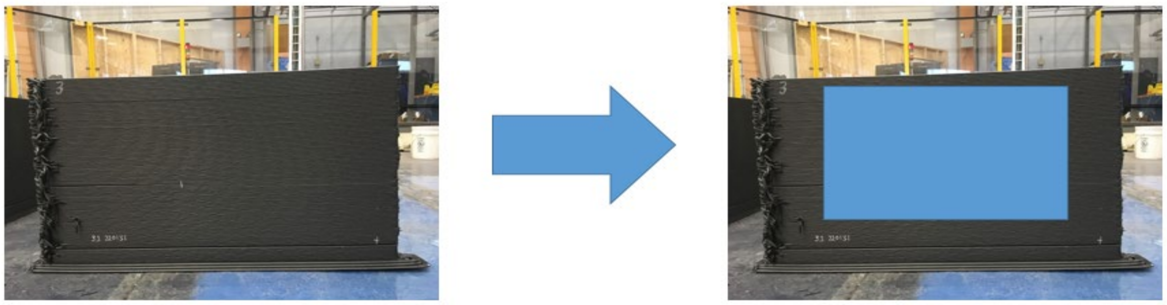

Figure 2: Image cropping

Manual edge identification

After tiling, the images are ready for manual and automatic labeling. Two different image manipulation and editing software tools were used - The first tool was Gimp, an open-source image editing tool [3], which we used to draw white lines along each edge between two layers, as shown in Figure 3. As this was very time-consuming, we started using ImageJ [4], another open-source scientific image tool, which included more dedicated manual edge-detection tools. In ImageJ, we first had to convert the images to an 8-bit color format in order to apply a manual threshold to it. By adjusting the values depending on the intensity variations on the image, contiguous lines are possible. Figure 4 shows the same image manually labeled with ImageJ. After completing the manual labeling for each image, we would similarly tile the labeled masks into 128x128 tiles to match the image tiles.

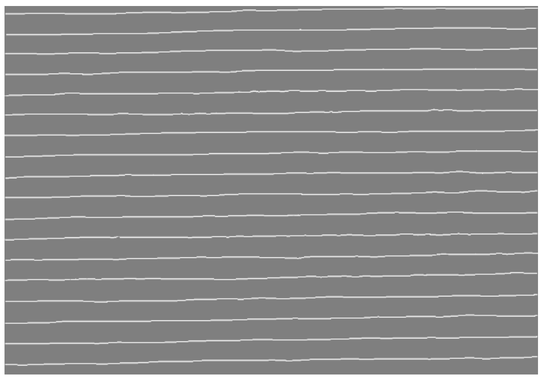

Figure 3: Manual edge labeling with GIMP

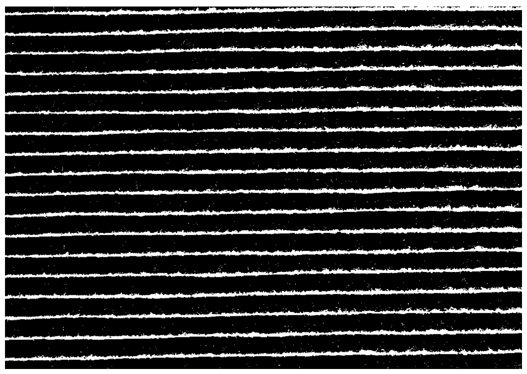

Figure 4: Manual edge labeling with ImageJ

Automated edge identification - Overview

The pipeline in Figure 5 goes through the steps of our layer identification research. The solid arrows are for the manual steps, while the dashed arrows are for the automated steps.

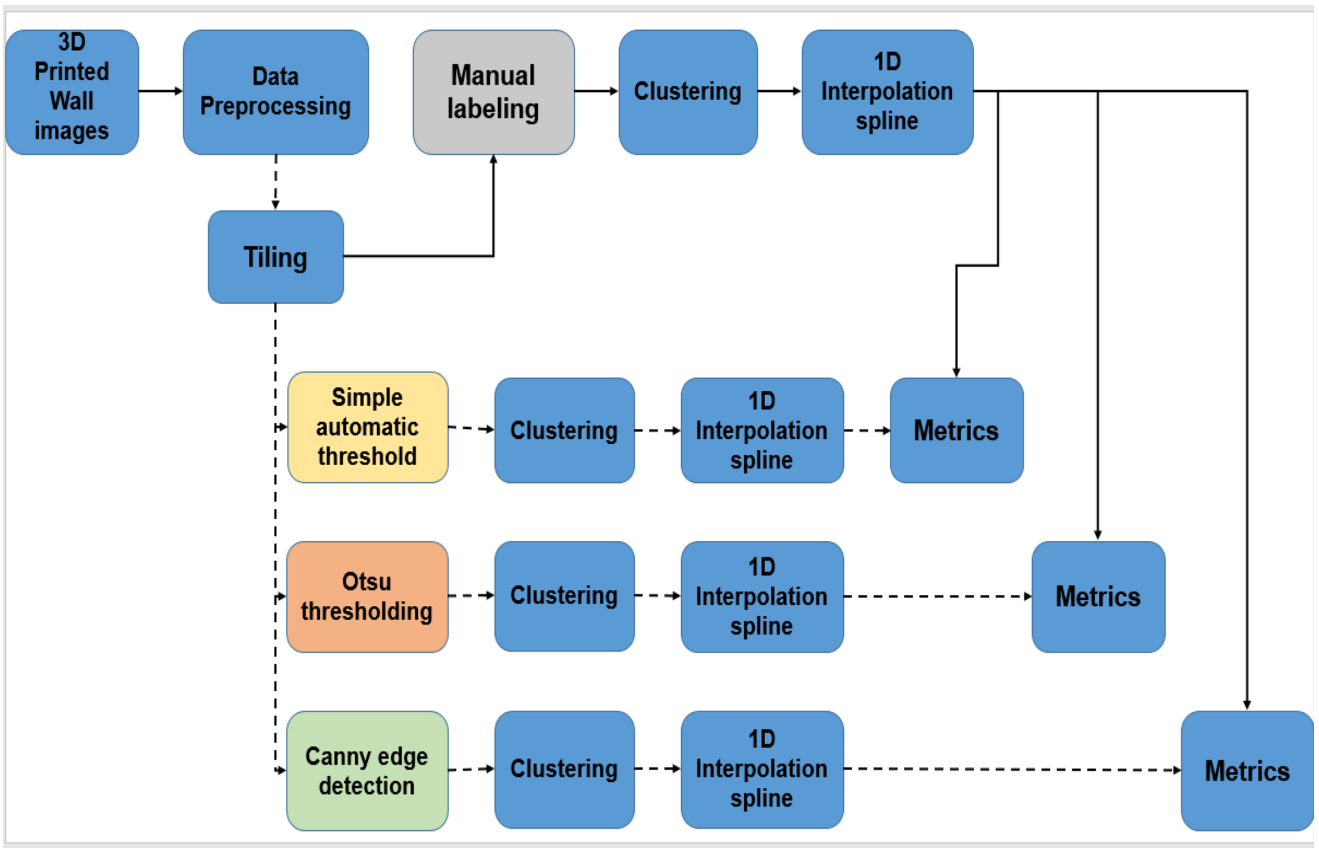

Figure 5: Layer interface identification pipeline

Automated edge identification - Step 1: Thresholding

After manually labeling the sample images, we proceeded with the automatic labeling of the same images. We opted to employ three specific methods:

- Simple thresholding
- Otsu thresholding [5]
- Canny edge detection [6]

First, we use simple image thresholding to identify the layers on the 3D-printed walls. Simple thresholding provided a straightforward and intuitive way to identify layers based on pixel intensities. The simple thresholding method converts the input image into a binary representation. A manually selected threshold value was applied to classify pixels as either edge or non-edge. Figure 6 shows the image sample, and Figure 7 is the resulting binary image. The edge pixels were colored in white, and the black areas were the non-edge pixels.

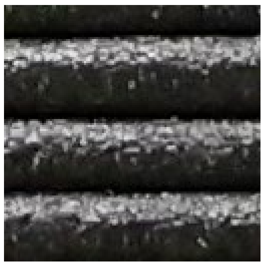

Figure 6: Camera image sample

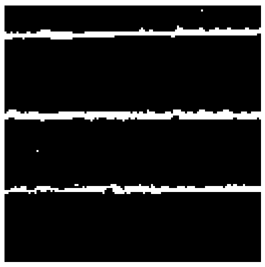

Figure 7: Method 1: Simple thresholding mask. White pixels are considered edge pixels. Black pixels are considered non-edge.

For the second method, we utilized Otsu image thresholding, a well-known automatic technique for determining the optimal threshold value for each layer based on the histogram of pixel intensities (Figure 8). Otsu thresholding separates layer and edge pixels by finding the value that minimizes the variance between light and dark classes. It automatically analyzes the pixel intensity distribution, iterates through possible thresholds, and identifies the one that yields the clearest separation. By incorporating Otsu thresholding, we achieved precise layer identification, enhancing our image analysis and enabling further investigation (Figure 8).

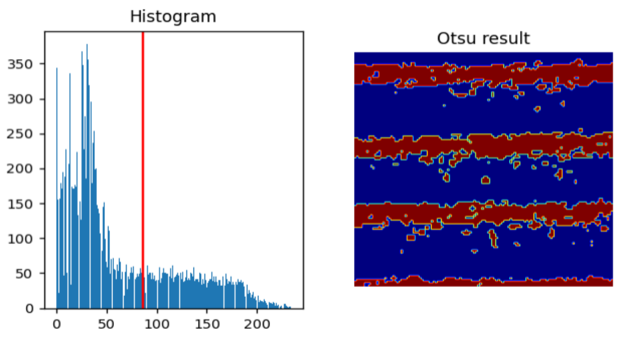

Figure 8: Method 2: Otsu thresholding. Histogram of pixel intensities and typical results. Red pixels are considered edge pixels while blue pixels are considered non-edge pixels.

Lastly, we employed Canny edge detection, a well-established edge detection algorithm developed by John Canny in 1986. Given its popularity and effectiveness in image processing and computer vision tasks, we selected Canny edge detection as our third method for layer identification. Canny edge detection identifies edges in an image based on the gradient of pixel intensities. It includes key steps for identifying edges in an image. It starts by applying a Gaussian filter to reduce noise and enhance edge quality. Then, the gradient of pixel intensities is calculated using the Sobel operator, providing the rate and direction of intensity change. Non-maximum suppression thins out edges by preserving only local maxima in the gradient direction. Finally, hysteresis thresholding connects potential edge pixels to form continuous strong edges. Canny detection is expected to produce accurate, well-defined edges while handling noise. By following these steps, we were able to extract the edges of interest for most of the images (Figure 9), providing valuable information for our analysis.

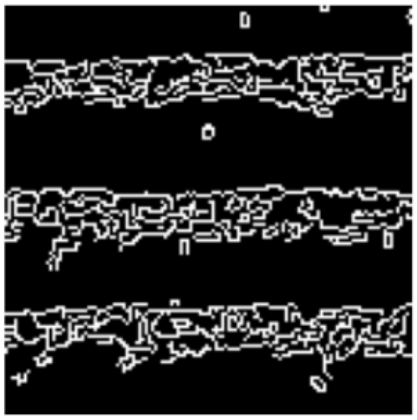

Figure 9: Method 3: Canny edge detection. White pixels correspond to identified edges while black pixels are considered as non-edge pixels.

For each image, the three methods, simple thresholding, Otsu thresholding, and Canny edge detection, are applied in parallel. This approach allowed us to explore different techniques and assess their utility for layer identification purposes.

Automated edge identification - Step 2: Clustering

In order to address irregularities in the identified pixels of the images, the incorporation of clustering techniques was found to be necessary. The pixel identifications in the images exhibit a certain level of noise, as you can see in Figure 10. This is characterized by numerous small discontinuities and outliers.

To tackle this challenge, we employed the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [7] for grouping the identified pixels into contiguous horizontal rows. DBSCAN is a powerful clustering method widely used in machine learning for separating clusters of high density from areas of low density, which, in our case, corresponds to distinguishing edge clusters from non-edge clusters.

One of the key reasons behind our selection of DBSCAN was its robustness in handling noise and outliers. Designed specifically for noisy and sparse datasets, DBSCAN effectively differentiates between clusters and noise points by considering the density of the data points. Any outliers or isolated points that do not belong to any cluster are labeled as noise, enabling DBSCAN to maintain its reliability even in the presence of noisy data.

An additional advantage of DBSCAN is its ability to automatically determine the number of clusters without requiring pre-specification. By dynamically detecting densely populated regions and ensuring their connection, DBSCAN offers a flexible approach for approximating layer boundary contours, which was highly beneficial for our research objectives.

Figure 10 provides an illustrative example showcasing the application of DBSCAN to a sample ground truth image and a sample obtained through our simple thresholding method. The effectiveness of DBSCAN in accurately clustering the pixels and identifying layer boundary contours can be observed.

By utilizing DBSCAN in our analysis pipeline, we were able to overcome the challenges posed by the "messiness" of pixel identifications, enabling us to achieve more accurate and representative results in our research.

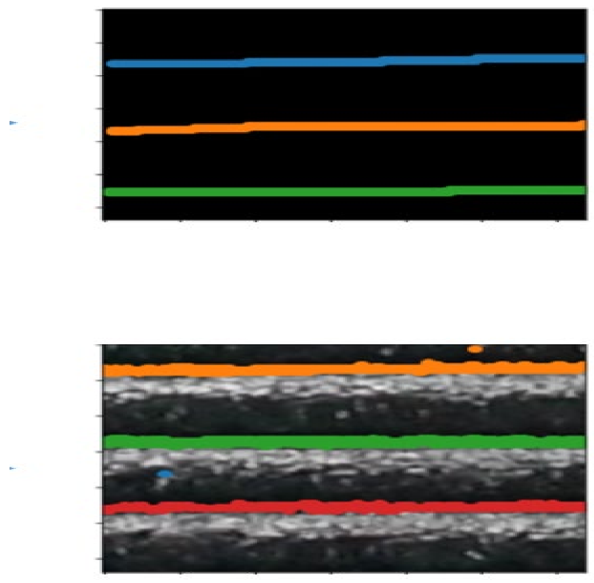

Figure 10: DBSCAN clustering applied to a manually labeled image (top) and automated labeled image (bottom).

Automated edge identification - Step 3: Function fitting

We then implemented an interpolation function that drew a 1-D smoothing spline that fit every group of edge cluster points of every image [6] (Figure 11).

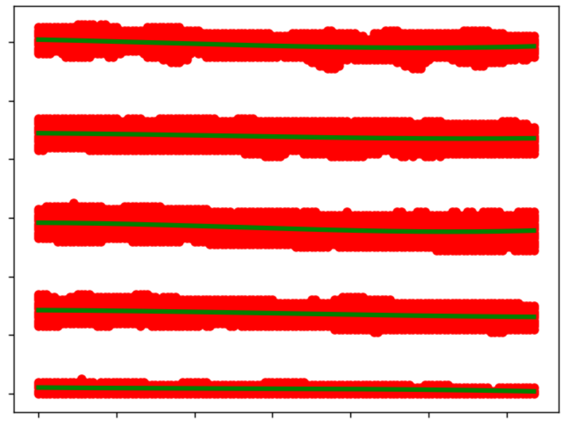

Figure 11: Interpolation function applied to pixel clusters.

Performance evaluation

This step allowed us to compare and evaluate the error between the edges on each image labeled manually and its corresponding image automatically labeled. Performance metrics were computed to assess the effectiveness of each method. To quantify the error between the sets of splines, we defined a function that takes two sets of splines as input. This function calculates the error by comparing the values of the splines at multiple points (in this case, 128 points) along the x-axis. The absolute difference between the spline values at each point is accumulated to compute a running total of the error.

Subsequently, we iterated through pairs of ground truth and predicted sets of splines. For each pair, we calculated the error between the corresponding splines and tracked the total error for each set of splines. To determine the average pixel error for one pair of splines, we divided the total error by the number of pixels.

Next, we calculated the average error for each set of splines (each image) by dividing the total average error by the number of splines in that set (image). Additionally, we computed the average error for all sets of splines by dividing the total average error across all sets (images) by the total number of processed sets of splines.

Finally, we calculated average pixel errors for all sets of splines. In cases where the number of predicted splines differed from the number of ground truth splines, we considered it a failure and reported failures separately.

<u>Results</u>

In this section, we present the results of our study on automated layer identification in 3D-printed walls from a LAAM printer. Figures 12 and 13, respectively, show examples leading to low and high accuracy for all methods. (The specific accuracy for each method depends on the light intensity variations over the image.) One can see the difference between the definition of the clusters for each automated method and our ground truth. For the simple thresholding method, we observed that this method was sensitive to variations in image quality and noise, leading to inaccurate results. Next, we employed Otsu thresholding, which performed better than simple thresholding in terms of accuracy and was less sensitive to noise and image quality variations. However, it was computationally more expensive. Finally, Canny edge detection produced accurate results but was computationally most expensive and could be sensitive to noise.

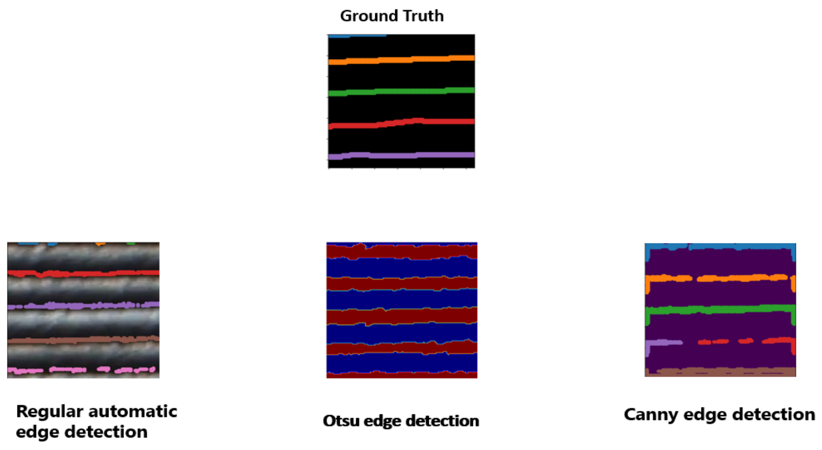

Figure 12: Example of Low accuracy DBSCAN clustering results on manually labeled image and automated labeled images

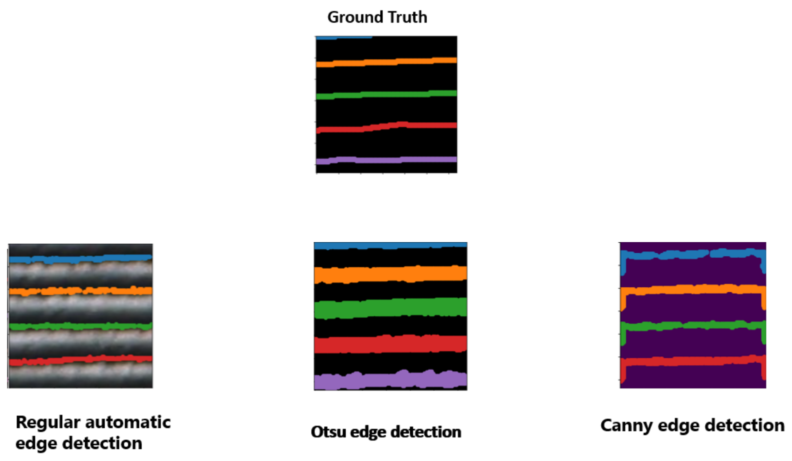

Figure 13: Good Example of High accuracy DBSCAN clustering results on manually labeled image and automated labeled images

For each image, the three methods were applied in parallel, and the resulting layer identification outputs were compared against manual identifications. We achieved highly accurate layer identification in 70% of the images using Otsu, 40% using Canny edge detection, and 30% using our simple thresholding method, with an average error of only 1 pixel per image when comparing the manually labeled and automatically labeled images for our best method. In Figure 14, we have the average pixel errors for all sets of splines. In cases where the number of predicted splines differed from the number of ground truth splines, we considered it a failure and factored it into the evaluation. This is shown in Figure 15.

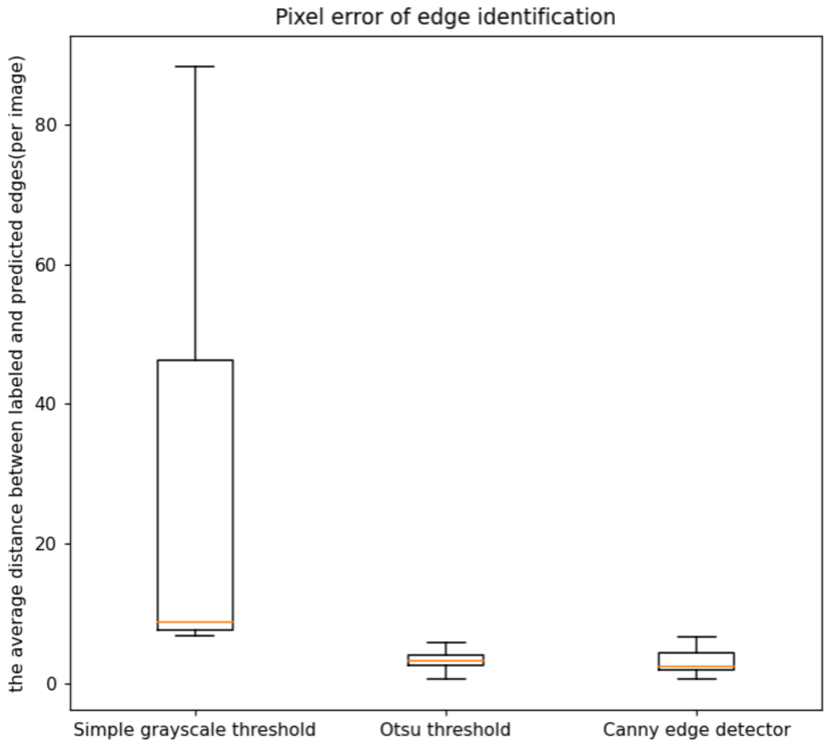

Figure 14: Average pixel error of edge identification

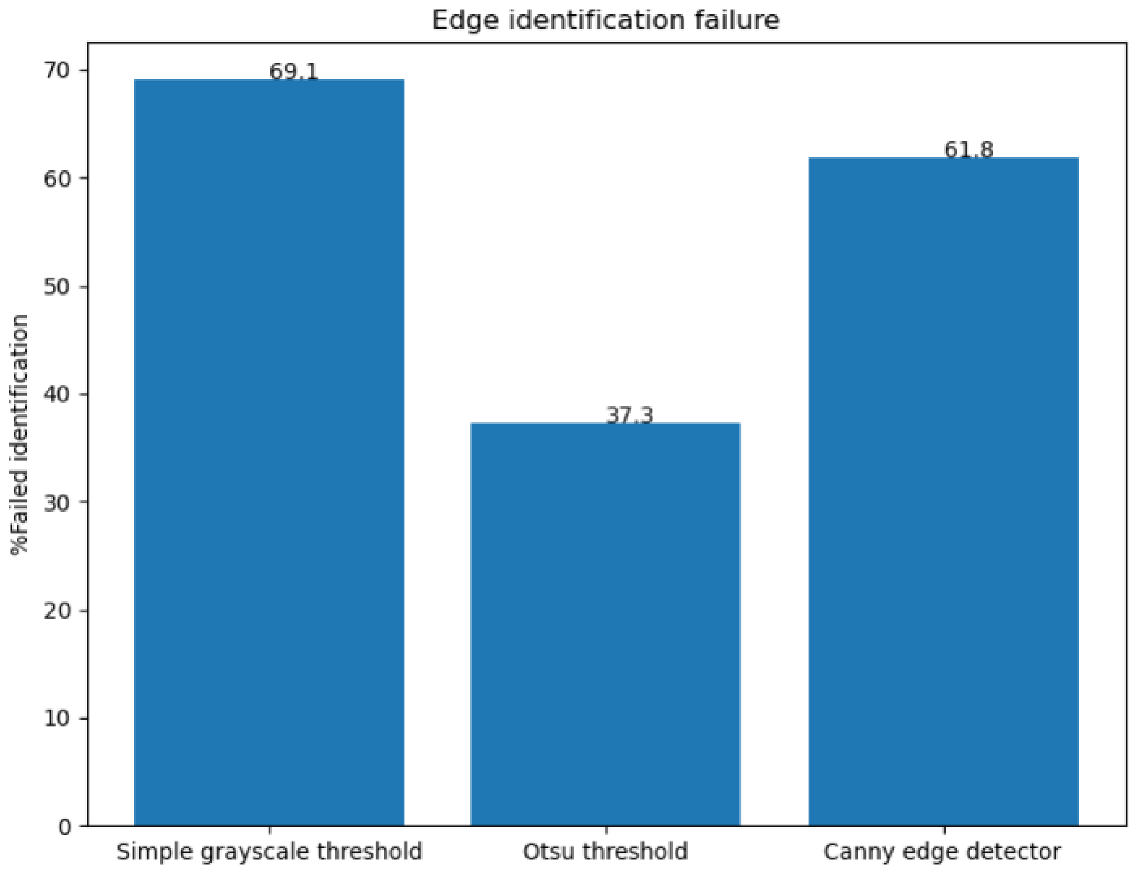

Figure 15: Edge Identification Failure graph

Conclusion and Future Work

Our main goal is to develop an automated method for layer identification in 3D-printed walls produced by a LAAM printer. We tested three image thresholding methods: simple thresholding, Otsu thresholding, and Canny edge detection for layer boundary identification. Otsu thresholding was the most accurate method with minimal manual intervention, handling image quality variations and noise in approximately 70% of our images. Otsu thresholding is a promising approach for automated layer identification, providing high accuracy and reduced manual effort. Canny edge detection is effective but computationally complex. Simple thresholding may be suitable in specific scenarios, despite lower accuracy compared to the other methods.

In the future, we would like to apply our methods to thermal images, to different types of materials, and potentially look into video streams. Finally, we expect to embed our approach to obtain an automatic image-to-layer data pipeline and integrate it into an optimization framework for manufacturing quality assessment and control.

References

1. Paquit, Vincent C, Ferguson II, James S, Goldsby, Desarae J, Halsey, William H, Joslin, Chase B, Rose, Derek C, Scime, Luke R, Siddel, Derek H, and Richardson, Dylan. *Peregrine*. Computer software. Vers. 00. USDOE. 16 Aug. 2019. Web.
2. Spencer, Ryan, et al. "An innovative digital image correlation technique for in-situ process monitoring of composite structures in large scale additive manufacturing." *Composite Structures* 276 (2021): 114545.
3. The GIMP Development Team. (2019). *GIMP*. Retrieved from https://www.gimp.org
4. Rueden, Curtis T., et al. "ImageJ2: ImageJ for the next generation of scientific image data." *BMC bioinformatics* 18 (2017): 1-26.
5. Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." *IEEE transactions on systems, man, and cybernetics* 9.1 (1979): 62-66.
6. Rong, Weibin, et al. "An improved CANNY edge detectiond algorithm." *2014 IEEE international conference on mechatronics and automation*. IEEE, 2014.
7. Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." *kdd*. Vol. 96. No. 34. 1996