Solid Freeform Fabrication 2022: Proceedings of the 33rd Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

Towards Online Monitoring and Data-driven Control: A Study of Segmentation Algorithms for Laser Powder Bed Fusion Processes

Alexander Nettekoven[a], Scott Fish[a], Joseph Beaman[a], Ufuk Topcu[b]

[a]*Department of Mechanical Engineering, The University of Texas at Austin, 204 E. Dean Keeton Street, Stop C2200, Austin, Texas 78712, USA*
[b]*Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, 2617 Wichita Street, C0600, Austin, Texas 78712, USA*

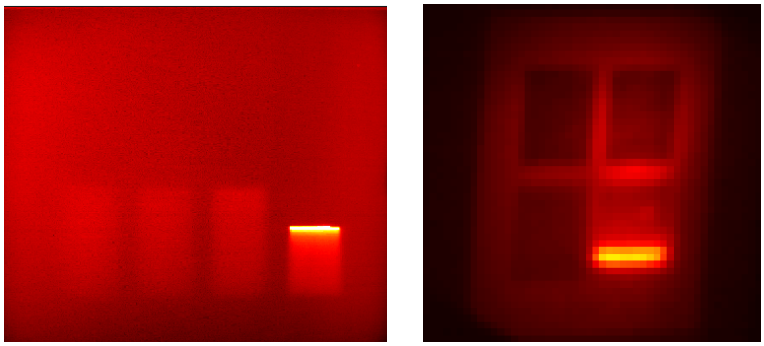
Abstract

An increasing number of laser powder bed fusion machines use off-axis infrared cameras to improve online monitoring and data-driven control capabilities. However, there is still a severe lack of algorithmic solutions to properly process the infrared images from these cameras, which has led to several key limitations: a lack of online monitoring capabilities for the laser tracks, insufficient pre-processing of the infrared images for data-driven methods, and large memory requirements for storing the infrared images. To address these limitations, we study over 30 segmentation algorithms that segment each infrared image into a foreground and background. By evaluating each algorithm based on its segmentation accuracy, computational speed, and spatter detection characteristics, we identify promising algorithmic solutions. The identified algorithms can be readily applied to the laser powder bed fusion machines to address each of the above limitations and thus, significantly improve process control.

Keywords: LPBF, SLS, SLM, laser track, pre-processing, efficient storage

1. Introduction

Numerous applications, such as aircraft and medical devices, increasingly rely on 3D parts produced by laser powder bed fusion (LPBF) [1–4]. Their potential of producing lightweight 3D parts with complex geometries creates a significant advantage for LPBF processes over traditional manufacturing processes [1, 5]. However, limited process control leaves LPBF machines still susceptible to process disturbances that often lead to poor part quality [4–7]. Building high-quality parts and doing so repeatably, especially for safety-critical applications, remain a key challenge for LPBF processes today [3, 4].



(a) SLS off-axis infrared image. (b) SLM off-axis infrared image.

Figure 1: Sample infrared images of build surfaces from two different builds used for this study, one from a selective laser sintering (SLS) machine and one from a selective laser melting (SLM) machine. The laser in both builds scanned four rectangular cross-sections for each layer.

Email addresses: nettekoven@utexas.edu (Alexander Nettekoven), scott.fish@utexas.edu (Scott Fish), jbeaman@me.utexas.edu (Joseph Beaman), utopcu@utexas.edu (Ufuk Topcu)

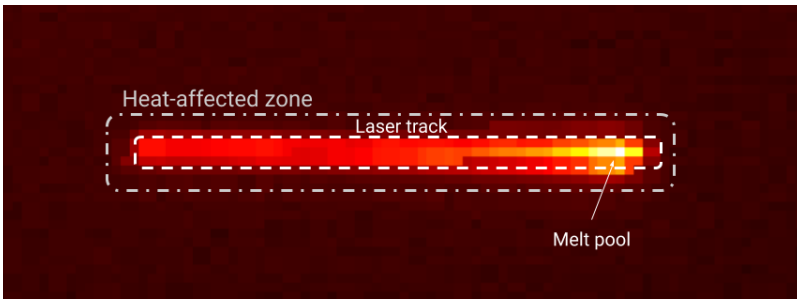


Figure 2: A single laser track scanned by a laser. The approximate locations of several region of interests are marked. The cropped image is taken from the same SLS build as Figure 1a.

To improve process control, extensive research efforts are going into using in situ measurements from off-axis infrared cameras, which continuously record the entire build surface at once [8, 9]. The infrared images produced by these cameras usually consist of 2D arrays of temperature intensity values, where each value represents the measured temperature of the corresponding build surface location. Figure 1 shows two exemplary infrared images for the two type of LPBF processes, selective laser sintering (SLS) and selective laser melting (SLM). In contrast to in situ measurements from other infrared sensors, infrared images from off-axis infrared cameras have the advantage of providing information about multiple regions of interest on the build surface at once. Some of the previously studied regions of interest include the melt pool, the plume, and the laser scans (see Figure 2) [8–19]. Measuring the regions of interest can provide crucial information for process control [8–13, 18]. However, most LPBF processes still lack appropriate algorithms that can automatically extract and use this information from the infrared images of the build surface effectively [8, 20, 21].

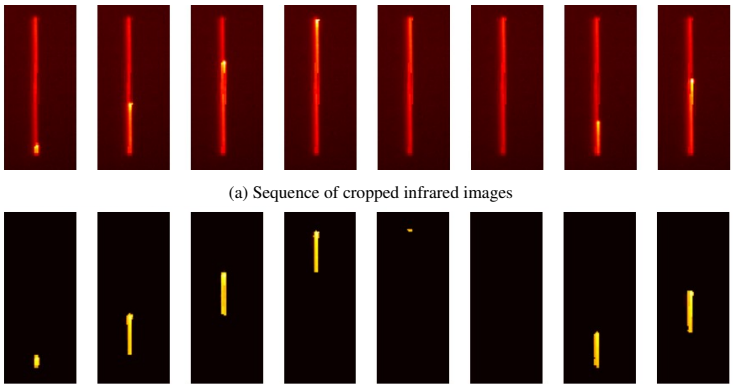
In an effort to reduce the shortcoming of algorithmic solutions for these infrared images, recent studies have focused on developing different online monitoring and data-driven algorithms. [10] applied common computer vision techniques to identify the plume in each image and use the area and the mean intensity of the extracted plume to monitor the process health. In a follow-up study, [11] improved upon this concept by developing statistical descriptors of the plume. [22] developed an image segmentation algorithm that extracts the melt pool, plume, and spatters from the infrared images. The extracted regions were then monitored to study the effect of different process parameters. [23] performed a similar process parameter study by extracting the plume and spatter from the infrared images with a popular image segmentation method called Otsu's method [24]. Some other approaches used the whole image as input for data-driven methods; [25] trained a convolutional neural network to detect splatter and deformation defects. [26] applied a data-driven method that provided human-interpretable information about the cause of strength differences in test specimen.

Though these studies provide promising results for improving process control, there still exist severe limitations that have not been fully investigated or have not been addressed at all.

Monitoring limitation. Recent developments of online monitoring and data-driven algorithms have mostly focused on the melt pool, the plume, the spatters, or the whole build surface. So far, the laser tracks, which are the result of the laser beam continuously traveling along the scan path on the build surface, have not been subject to any significant algorithmic developments. Extracting the laser tracks from the infrared images is difficult because the laser tracks' temperature distribution lies in between the temperature distribution of the melt pool and the temperature distribution of the rest of the powder bed, as indicated by Figure 2. Monitoring a single or multiple laser tracks can provide crucial process information, such as geometrical accuracy of the scanned-cross section, deviation of process parameters, and consistency of each laser track [13, 15–17]. Identifying the laser tracks in the infrared images may also enable monitoring of the heat-affected zone, which encompasses the laser tracks.

Pre-processing limitation. Several data-driven methods use raw infrared images as input without any pre-processing. Raw infrared images are often noisy and high-dimensional, where the informative portion of the data is usually low-dimensional. Examples for the informative portion of the infrared images are the melt pool or the laser tracks, which usually make up only a small region of the infrared images. A lack of pre-processing of the infrared images can cause significant performance degradation of the data-driven methods and also a need for more data [27, 28].

On the other hand, the data-driven methods that do use pre-processing often rely on traditional thresholding methods to separate the informative portion of the image from the uninformative portion. Otsu's method is a popular thresholding method, but usually only works well for images with high contrast and low noise [10, 11, 24, 29]. Both of these conditions depend highly on the process parameters and the material used. Furthermore,



(a) Sequence of cropped infrared images

(b) Manually extracted foregrounds

Figure 3: Exemplary sequence of cropped infrared images and the corresponding foregrounds. The cropped infrared images are taken from the same SLS build as Figure 1a but from a different layer. This layer scans the rectangles vertically.

these methods often cannot distinguish whether the laser is on or off in a given infrared image. Since data-driven methods may only use images that were recorded during scanning, filtering out any other images is crucial. These different limitations can result in restrictions to the end-application and also significantly reduce the efficacy of the data-driven methods.

Storage limitation. In situ measurements produce massive amounts of data and thus, pose a significant challenge for data storage systems [7, 30, 31]. Previous suggestions to overcome this challenge included storing statistical metrics that were calculated from the raw infrared images [7]. However, reducing high-dimensional data, such as infrared images, to scalar values usually connotes significant information loss and often prevents any meaningful learning for the data-driven methods. Thus, storing data in an effective and efficient manner is desperately needed as the need for data to train the data-driven methods continuously increases.

To address the monitoring, pre-processing, and storage limitations, we study different segmentation algorithms that segment the infrared images into foregrounds and backgrounds. By retaining the informative portion of the infrared images, the foregrounds can be used to (a) online monitor the laser tracks, (b) function as pre-processed input for the data-driven methods, and (c) significantly reduce the memory footprint of the off-axis cameras. We evaluate each algorithm based on segmentation accuracy, computational speed, and spatter detection characteristics. The first two criteria are critical for determining accurate and fast segmentation. The latter is critical to account for ejected spatter that might influence segmentation results as some processes have strong spatter characteristics. By discussing the details and implications of the study results, we provide a guide for selecting suitable segmentation algorithms for LPBF machines to significantly improve process control.

The remainder of the paper is structured as follows: In Section 2, we define the segmentation problem and provide more detail on the motivation of this paper. We continue by introducing the relevant segmentation algorithms in Section 3 and defining the study methodology in Section 4. We present our results in Section 5 and discuss how our results can overcome the aforementioned limitations in Section 6. We conclude by summarizing our findings and addressing future research in Section 7.

2. Segmentation definition and motivation

In this section, we define the segmentation problem and detail how finding suitable algorithms for this segmentation problem will overcome the aforementioned limitations. The segmentation problem and motivation will guide the selection and study of algorithms in subsequent sections.

Before defining the segmentation problem, we point out several process characteristics relevant to the segmentation problem. The infrared images from off-axis cameras always have a stationary view of the build surface. In other words, the pixel of any image always represents the same location on the build surface. Furthermore, as the laser scans the powder bed (exemplary sequence in Figure 3a), each infrared image contains a snapshot of the laser's effect on the build surface. Some of these scanning effects are new, while others have been captured by previous infrared images. These effects correspond to physical changes on the build surface, such as a different melt pool location and new additions to the laser tracks.

Segmentation definition. Since, in online monitoring and process control, we are often interested in recent changes of a process variable, we define the foreground of each infrared image as follows: the foreground consists

of the pixels that correspond to the most recent changes of the laser tracks. Here, most recent refers to the time difference between the 'current' image that was just recorded by the infrared camera and the 'previous' image that was recorded directly before the current image. Conversely, the background consists of all the pixels that do not belong to the foreground. For illustration, Figure 3b shows the foregrounds for the sequence of Figure 3a. We consider the cooldown of previously scanned powder as part of the continuously changing background.

This definition of the foreground-background segmentation problem offers several advantages that address the monitoring, pre-processing, and storage limitations.

First, the foregrounds from the segmented output can be used to directly monitor the recently scanned parts of the laser tracks. These parts can be monitored with respect to different properties, such as temperature distribution or geometric dimensions. Furthermore, by creating binary masks from consecutive foregrounds and superimposing the foreground masks, we can extract the pixel values from the current image and monitor all or parts of the previously-scanned laser tracks. This segmentation could provide crucial information about the temperature evolution of the scanned powder. Note that by superimposing foreground masks, we refer to the process of creating one composite mask where a pixel has the value *true* if this pixel was selected as a foreground in at least one of the foregrounds and *false* otherwise.

Second, the foregrounds and the superimposed foreground masks contain most regions of interest, such as the melt pool and the laser tracks. Depending on the process, the foregrounds may also contain the spatter and plume. Since recent data-driven approaches are predominantly focused on monitoring these regions of interest, the data-driven methods can use the foregrounds or modified foregrounds with the superimposed foreground masks as input and eliminate the uninformative portions from the data. This pre-processing can significantly reduce the dimensionality and noise of the data, thus increasing the effectiveness of the data-driven approaches. Furthermore, the segmented outputs also make it easy to eliminate entire images from the input where the laser has not recently scanned the build surface and may not contain any relevant or new information at all. These images will entirely or almost entirely consist of backgrounds.

Finally, based on the flexibility given by the segmented outputs discussed in the previous two paragraphs, the end-user can significantly reduce the memory footprint of the infrared images by storing only the portions of the images that are relevant for the end-application. Since the foregrounds are usually of interest, the LPBF machine can extract the foregrounds from the images and save the raw values for later usage without any information loss. Other portions of the images, such as the previously-scanned powder or possibly the heat-affected zone, can also be stored with the help of the superimposed foreground masks. In the case of the heat-affected zone, the location information of the scanned powder in the infrared images could be used to find the larger heat-affected zone, which directly encompasses the scanned powder.

3. Segmentation algorithms

In order to address the process limitations mentioned in Section 1, suitable algorithms should meet several criteria. First, the segmentation algorithms should possess high segmentation accuracy while still maintaining low computational complexity. Second, the algorithms should perform well without having to undergo an extensive training phase before deployment and should be applicable to various hardware setups and segmentation purposes. Thus, minimizing human input and avoiding restrictions to the end-applications is crucial. The algorithms may require some calibration of individual parameters, but this calibration should ideally occur only once for similar process setups, such as builds with the same powder material.

For the scope of this study, we focus on segmentation algorithms provided by the OpenCV and scikit-image libraries [32, 33]. Using OpenCV and scikit-image has several advantages. First, both libraries possess a breadth of segmentation algorithms that use almost no training data and have high potential to provide accurate segmentation with low computational complexity [32–34]. Most of the recently studied methods for the LPBF processes, such as Otsu's method, KNN, and Li's method, are also part of the implemented algorithms [10, 11, 29]. Second, most of the OpenCV and scikit-image algorithms are well-tested and optimized for speed, which will allow for comparative analysis of the algorithms' performances [33, 34]. Finally, since the algorithms are commonly available and easy to use, suitable algorithms identified in this study can be readily applied to research and commercial machines to address the above process limitations.

Table 1 lists the algorithms investigated in this study. The methods underlying these algorithms range from thresholding methods, such as basic thresholding or adaptive thresholding, to advanced statistical methods, such as mixture of gaussians [32, 33, 35, 36]. We provide further detail on these algorithms in the next paragraphs.

The scikit-image algorithms (see footnotes in Table 1) and several algorithms from OpenCV, specifically the adaptive mean algorithm, the adaptive Gaussian algorithm, and Otsu's method, are thresholding algorithms. These algorithms use either a global or local threshold to determine whether a pixel belongs to the foreground or the

Table 1: Overview of algorithms studied in this paper.

Name	Description
MOG[1]	Mixture of gaussian algorithm [37]
MOG2[1]	Improved MOG algorithm [38]
KNN[1]	K-nearest neighbors algorithm [39]
GMG[1]	Applies bayesian inference pixel-wise [40]
CNT[1]	Developed by OpenCV community [32]
GSOC[1]	Developed by Google Summer of Code community [32]
LSBP[1]	Uses local SVD patterns [41]
Otsu[1]	Global, clustering-based method [24, 42]
Li[2]	Global, entropy-based threshold [42, 43]
isodata[2]	Global, clustering-based threshold [42, 44]
Yen[2]	Global, entropy-based threshold [42, 45]
Triangle[1]	Global threshold using triangle method [46]
Sauvola[2]	Locally adaptive threshold [42, 47]
AdaptMean[1]	Local threshold based on adaptive mean [32]
AdaptGauss[1]	Local threshold based on Gaussian weighting [32]
Thresh(λ)	Global threshold with scalar λ
FD	Frame differencing: Subtracts previous image from current image [48–50]
SubMax(δ)	Thresholds image based on δ and the maximum pixel value of the current image
FD+'Name'	FD, then Thresh(1), then method 'Name'

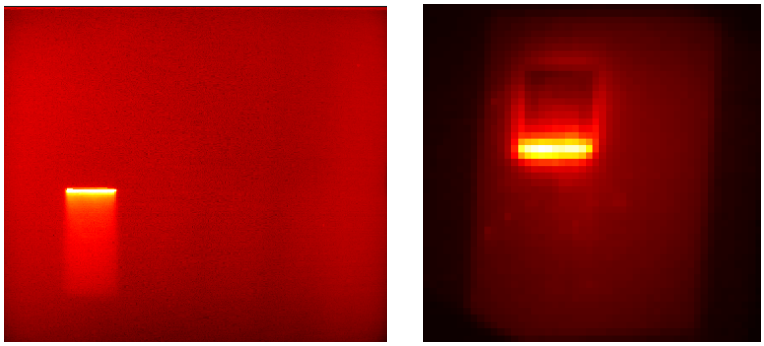
[1] openCV [32]
[2] scikit-image [33]

background [42, 51, 52] Since the laser always increases the powder temperature, and the foreground is directly affected by this increase in temperature, thresholding methods can be an effective way of segmenting an image. Recent work has demonstrated the potential of these segmentation algorithms [10, 11, 29].

Most of the OpenCV algorithms belong to the family of background subtraction algorithms that first model the background of an image and then use background subtraction to find the foreground [35, 36, 53]. OpenCV includes well-established background subtraction algorithms, such as the MOG, MOG2, KNN, and GMG algorithms [54]. The MOG and MOG2 algorithms use mixture of gaussians to model the background [37, 38]. The KNN algorithm uses a k-nearest neighbors method to update its background model, while the GMG algorithm uses Bayesian inference [39, 40]. OpenCV also includes recent algorithms, such as the LSBP algorithm [41], the GSOC, and the CNT algorithms, where the latter two algorithms do not originate from publications. As demonstrated by previous work, the algorithms strike a balance between computational complexity and segmentation accuracy, and thus, make ideal candidates for solving the segmentation problem [35, 55, 56].

In addition to the algorithms provided by OpenCV and scikit-image, we propose several simple algorithms to either supplement the above algorithms or to establish a baseline for the segmentation study. We detail these algorithms in the next paragraphs.

In order to establish a baseline for the segmentation study, we add two algorithms based on process intuition of the LPBF processes. The first algorithm, called *Thresh(λ)*, uses a constant global threshold λ for all images. For each image, the pixels with values above the threshold are selected as foreground, while the remaining pixels are selected as background. The second algorithm, called *SubMax(δ)*, selects a pixel as a foreground pixel if its value is close enough to the maximum pixel value of the corresponding image. Here, we define close enough as the pixel values that are only a constant δ below the maximum pixel value. For instance, if the maximum pixel value for a given image corresponds to 300 $°C$, then only the pixels whose values lay between $(300 - \delta)$ $°C$ and 300 $°C$ will be selected as foregrounds. Both of these algorithms exploit the fact that the laser increases the temperature of the build surface by selecting the pixels with the highest intensity values as foregrounds.



(a) SLS off-axis infrared image.

(b) SLM off-axis infrared image.

Figure 4: Exemplary infrared images recorded during the processing of the first rectangular cross-section for the SLS and SLM builds. The images are from the same layer as Figure 1.

Since the foregrounds represent the most recent laser track changes, we use a common change detection technique, called *frame differencing*, to supplement the above algorithms [48–50]. Basic frame differencing subtracts the pixel values of the previous image from the pixel values of the current image to detect changes in the current image [49]. We denote frame differencing as *FD* and the combination of frame differencing with a subsequent algorithm as *FD+'ID'*, where *'ID'* is the name of the subsequent algorithm (see Table 1).

For the combination of frame differencing with other algorithms, frame differencing first applies the subtraction to the current image and then supplies the subsequent algorithm with the frame differenced image for further processing. However, before the subsequent algorithm uses the frame differenced image, the image is thresholded with Thresh(1) to eliminate pixels with near-zero values caused by the frame differencing step. Eliminating these pixels reduces the noise of the images, which can cause significant segmentation problems for the subsequent algorithm. Combining frame differencing with other segmentation algorithms has the potential to improve the segmentation accuracy while avoiding any meaningful increase in computational complexity [50, 52].

4. Study methodology

In this section, we introduce the study methodology. First, we describe the experimental data and the study procedure used to test the algorithms. Next, we introduce the methods used to evaluate the segmentation accuracy, the computational speed, and the spatter detection. Finally, we provide detail on the computational hardware and describe how each algorithm's parameters are tuned to ensure an adequate comparison between the algorithms.

4.1. Study setup

Two different research machines were used to collect data for this study; one SLS machine and one SLM machine. The SLS machine is located at the University of Texas at Austin, while the SLM machine is located at the Rensselaer Polytechnic Institute. Both machines are equipped with an off-axis infrared camera; the SLS machine uses a FLIR 6701sc camera with a resolution of 640 by 512 at a frame rate of 60 Hz. The SLM machine uses a FLIR A320 camera with an effective resolution of 50 by 50 at a frame rate of 60 Hz.

Both machines produced four rectangular cuboids. The SLS machine used carbon fiber reinforced polyether ether ketone (PEEK), while the SLM machine used cobalt-chrome. We show exemplary infrared images from both builds in Figure 4. These images stem from the same layer as Figure 1. The SLS images have been transformed to account for the non-perpendicular angle of view to the build surface. The SLM images have not been modified.

For this study, we select one batch of images from the SLM dataset and two batches of images from the SLS dataset. Each batch was recorded during the scanning of the first rectangular cross-section. Since some algorithms need a few images to initialize, each batch starts with roughly 50 images before the laser starts scanning. To imitate conditions in online monitoring and process control, we sequentially apply the infrared images from each batch to the algorithms. After the algorithms segment the images, the images are evaluated without any post-processing.

4.2. Segmentation accuracy

In order to evaluate each algorithm's segmentation accuracy, we test the algorithms on the SLS dataset and compare the segmented outputs to the segmentation ground truth. To determine the segmentation ground truth, we can use insights from the process setup of the SLS build. We know that the laser processed the rectangular

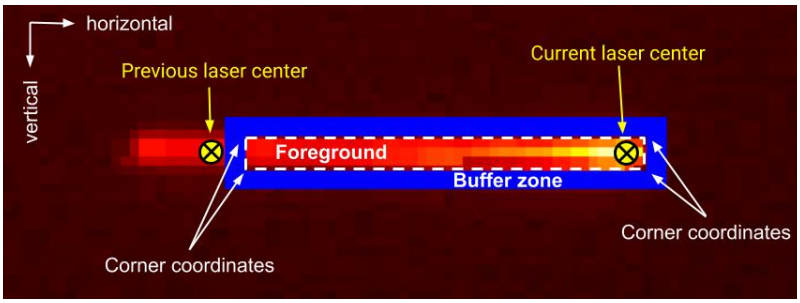


Figure 5: Single laser track annotated with the foreground ground truth and the corresponding buffer zone.

cross-section with 215 parallel and overlapping scan lines, where each scan line was processed with the same process parameters. The processed scan lines show up as horizontal laser tracks on the infrared images. Based on this information, we can use the average width of a laser track and the approximate location of the laser to determine the segmentation ground truth for each image.

Finding the laser's approximate location in each image is straightforward since the highest temperatures on the build surface occur close to where the laser spot is. Thus, the location can be determined by the pixels with the highest intensity values in each image. Finding the average width of the laser tracks is more challenging. First, we need to quantify what belongs to a laser track. Second, we need to determine the average width from multiple single laser tracks.

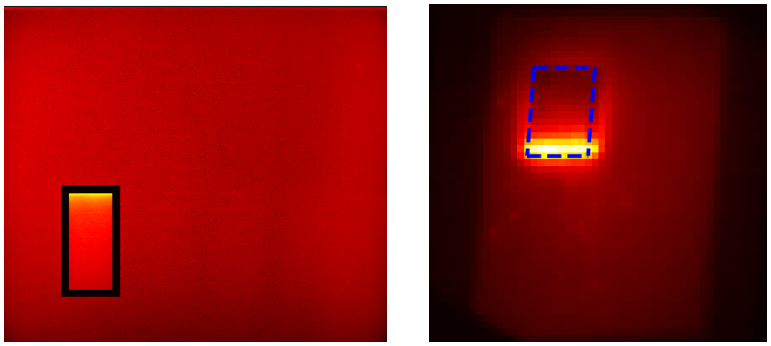
Since the intensity distribution of the laser spot resembles a Gaussian profile, the temperature transition from a single laser track to the surrounding heat-affected zone and unprocessed powder is mostly smooth on the infrared images. To determine which pixels belong to a single laser track, we define a cutoff temperature. Any pixels with intensity values above the cutoff temperature belong to this laser track. This cutoff temperature needs to be sufficiently above the build surface temperature to ensure that the elevated intensity values are only caused by the direct scanning of the laser.

For this study, we define the cutoff temperature as $T_c = 3 \cdot \sigma_{bs} + T_{max}$, where T_{max} is the maximum temperature of the build surface and σ_{bs} is the standard deviation for the temperature distribution of the build surface. σ_{bs} and T_{max} are calculated from images that show the build surface before the laser starts scanning. In the case of the selected batches for the SLS build, the average σ_{bs} is around 6 $°C$, while the average T_{max} is around 277 $°C$. We, thus, set T_c to 295 $°C$. Based on this cutoff temperature, we can determine the average width of a single laser track by counting the average amount of pixels stacked on top of each other along the length of the laser track. Figure 2 shows an exemplary laser track that is used for estimating the width, where the width is in the vertical direction of the image. We use multiple single laser tracks recorded during the beginning of different cross-sections and average over the measured widths.

Given the average laser track width and the laser's approximate location for each image, we can find the most recent laser track changes as follows: We know that each laser track is scanned from left to right along the horizontal axis. Thus, we approximate the most recent laser track changes with a rectangular box along the horizontal axis. This box has the same width in the vertical direction as the average laser track width. Furthermore, the box stretches from the laser location of the current image to the laser location of the previous image. If the current image does not have a laser location, e.g., because the laser is off, the box sits between the previous laser location and the right border of the rectangular cross-section. If the previous image does not have a laser location, then the box sits between the current laser location and the left border of the rectangular cross-section. If neither the current nor the previous image have laser locations, then no box is drawn as no foreground is present in the current image.

To demonstrate how the exact box location for each image is determined, we use the same laser track image from Figure 2 and annotate the image in Figure 5. For the vertical direction, the box coordinates are exactly half the average laser track width above and below the vertical coordinate of the current laser center. Since the laser has a circular beam spot on the build surface, the horizontal coordinates of the right side corners are half the average laser track width to the right of the current laser center. For the left side corners, the horizontal coordinates are to the right of the previous laser center location. However, the horizontal distance between the left side corner coordinates and the previous laser center location is half the average laser track width plus an extra distance due to a buffer zone (blue box in Figure 5).

To eliminate any dependency of the study results on the cutoff temperature, we use a small buffer zone around



(a) SLS off-axis infrared image.

(b) SLM off-axis infrared image.

Figure 6: Annotated infrared images from Figure 4. The buffer zone for the SLS cross-section is marked in black. The contour of the SLM cross-section is marked with dashed lines in blue.

each laser track that will not be part of the segmentation evaluation. This buffer zone eliminates the pixels representing the transition zone between the laser track and the surrounding areas, such as the heat-affected zone and the unprocessed powder. Using a buffer zone allows the end-user to use lower cutoff temperatures if desired. Furthermore, we use an outer buffer zone around the entire rectangular cross-section to eliminate any segmentation uncertainties at the transition of the processed powder to the unprocessed powder (see Figure 6a).

Given the box coordinates, we can determine the segmentation ground truth for each infrared image. The pixels within each box represent the current foreground, while the pixels outside of the box represent the background. Before the segmented outputs are compared to the ground truth, the inner and outer buffer zones eliminate the pixels representing the transition zones from the evaluation. For this study, the border thickness of the inner buffer zone is set to half the average laser track width, while the thickness for the outer buffer zone is set to five times the average laser track width.

We can evaluate the segmentation accuracy with the commonly-applied F_1-score. By comparing the binary mask of each segmentation output pixel-wise to the corresponding ground truth image, we can calculate the true positive rate TP, false positive rate FP, true negative rate TN and false negative rate FN. From these rates, we can calculate the resulting F_1-score by

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

where *Precision* and *Recall* are defined as

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}.$$

4.3. Parameter tuning

Several of the introduced algorithms have adjustable parameters that can significantly differ the segmentation accuracy of each algorithm. To ensure adequate comparison of the different algorithms, we test each algorithm twice; once with default parameters and once with calibrated parameters. We use a small number of images from the other SLS batch that were recorded during the scanning of the first 20 consecutive laser tracks to tune each algorithm's parameters. By applying a random grid search to test hundreds of different sets of parameters on these laser tracks and scoring each set with the F_1-score, we calibrate each algorithm's parameters.

Table A.4 in Appendix A shows the default and calibrated parameters for each algorithm. The default for the Thresh(λ) parameter was set to the calculated T_c of 295, while the default for the FD+Thresh(λ) was set to 3. The default for the SubMax(δ) parameter was set to 20.

4.4. Computational speed

We test the computational speed of each algorithm by timing the sequential processing on a batch of 2000 images from the SLS dataset and calculating the average computation speed for each image. The algorithms are implemented in Python but are often speed-optimized with underlying C or C++ implementations by OpenCV and scikit-image [32–34]. For the algorithms we implemented, e.g., the frame differencing method, we rely on numpy [57]. To ensure a relative comparison between the different algorithms, we use the CPU implementation and the algorithm's calibrated parameters. We perform our computation on a laptop with an Intel i9-9750H six core (2.6 up to 4.5 GHz, 12 MB Cache) and 32 GB DDR4 RAM.

4.5. Spatter detection

Spatter is often present during the processing of metal powder. Spatter can be part of the monitoring variables, as demonstrated by previous work, or can be seen as a disturbance [22, 23]. In the case of this segmentation study, the spatter can cause false positives and create foregrounds that would suggest changes to nonexistent laser tracks. The testing of the segmentation accuracy with the SLS dataset circumvented this problem since PEEK does not create noticeable spatters during the processing of the build surface. However, the segmentation accuracy may change drastically if the algorithms are applied to processes with high amounts of spatter ejection. Thus, we separately investigate the spatter detection for the segmentation algorithms on the SLM dataset. We use the SLM dataset due to its strong spatter ejection characteristics as the spatter was repeatedly ejected across the entire build surface and significantly obscured the laser tracks.

The upper-left rectangular cross-section is the first cross-section scanned for each layer of the SLM build. We use the first 400 images of a sample layer that show the processing of the upper-left cross-section and apply the algorithms to the images. We manually analyze the sequence of images for each algorithm by qualitatively looking for obvious artifacts in the foregrounds. If a given algorithm does not detect spatter in its foregrounds, the foregrounds would only exhibit a single coherent laser track, such as Figure 3b. Otherwise, the foreground will include additional nonzero pixels across the entire image.

To support and illustrate our manual analysis, we use a binary composite image for each algorithm that shows which pixels have been part of a foreground at least once and which pixels were never selected as part of a foreground. The binary composite image is created by first turning the foregrounds of a given algorithm into binary masks, where the foreground pixels have the value *one* and the background pixels the value *zero*. Following, the masks are summed up pixel-wise. The resulting binary composite image has nonzero values for pixels that have been part of a foreground at least once and pixels with zero values that were never selected as part of a foreground. If a given algorithm includes spatter in its foregrounds, the binary composite image would show numerous pixels selected as foreground that lay outside of the upper-left rectangular cross-section.

For visualization purposes, we manually extract the approximate border contour of the upper-left cross-section (Figure 6b) and project the contour onto each binary composite image. The area of the nonzero pixels in the binary composite image should approximately match the area represented by the projected contour. Some nonzero pixels fall slightly outside of the contour since this 'overflow' of nonzero pixels around the contour is caused by the difference in segmentation of the transition zone between the laser tracks and the unprocessed powder.

For the segmentation of the SLM batch, we use the default parameters for the segmentation algorithms. The calibrated parameters from the SLS dataset may not translate to the SLM dataset since the pixel intensity values from the SLS dataset are different from the SLM dataset.

5. Results

5.1. Segmentation accuracy

The simple combination algorithm FD+Thresh(λ) performed best on the segmentation accuracy test (Table 2). Even with default parameters, FD+Thresh(λ) had a better F_1-score than all of the other algorithms. Several algorithms, such as FD+GSOC and FD+Yen, trailed FD+Thresh(λ) but still performed better than the baseline algorithm Thresh(λ). All algorithms that performed better than Thresh(λ) were combination algorithms that used the frame differencing method as the first processing step.

Most of the OpenCV algorithms that were not combination algorithms scored in the midfield of the F_1-score. Most of the thresholding algorithms, on the other hand, scored at the bottom of the F_1-score. Some of these thresholding algorithms, such as the Yen, isodata, and Sauvola algorithms, did not score higher than 0.00 and are grouped as *Others* in Table 2. This group also includes the other baseline algorithm SubMax(δ). For a detailed breakdown of each algorithm's performance that includes the precision and recall, we refer the reader to Table A.5 and Table A.6 in Appendix A.

To demonstrate the segmentation results, we present the results from different algorithms for an exemplary sequence of the SLS batch in Figure 7. The first two rows show the raw infrared images and the corresponding ground truth images. The ground truth images mark the current foreground as red, the current background as yellow, and the current buffer zones as black. The subsequent rows show the binary masks of the segmentation results. The white pixels represent the foreground, while the black pixels represent the background. All of the images are cropped to the area of the first cross-section (see Figure 4a and 6a).

As indicated by the third row of Figure 7, FD+Thresh(λ) successfully detected the desired foreground pixels. FD+Thresh also recognized when there are no foreground pixels present due to the laser not scanning the build surface (see third column). In contrast, Thresh, MOG, and Otsu's method, did not successfully segment the images

Table 2: Segmentation accuracy of tested algorithms on the SLS dataset. Algorithms are ordered by calibrated F_1-score. The bold algorithm is one of the baseline algorithms.

Algorithms	Parameters	F_1-score
FD+Thresh(λ)	calibrated	0.93
	default	0.88
FD+GSOC	calibrated	0.84
	default	0.41
FD+Yen	–	0.78
FD+CNT	calibrated	0.70
	default	0.03
FD+MOG	calibrated	0.67
	default	0.60
Thresh	calibrated	0.62
	default	0.17
FD+isodata	–	0.61
FD+Otsu	–	0.60
FD+Triangle	–	0.60
GSOC	calibrated	0.60
	default	0.13
MOG	calibrated	0.58
	default	0.35
CNT	calibrated	0.56
	default	0.04
FD+Sauvola	calibrated	0.54
	default	0.48
FD+Li	–	0.52
FD+GMG	calibrated	0.38
	default	0.36
FD+MOG2	calibrated	0.35
	default	0.33
GMG	calibrated	0.29
	default	0.29
LSBP	calibrated	0.29
	default	0.26
MOG2	calibrated	0.29
	default	0.17
FD+KNN	calibrated	0.22
	default	0.21
KNN	calibrated	0.14
	default	0.10
Triangle	–	0.02
Yen	–	0.02
FD+LSBP	calibrated	0.01
	default	0.01
Otsu	–	0.01
Others	–	0.00

Table 3: Average computation times of tested algorithms. Algorithms are ranked by time in increasing order.

ID	Time per image [ms]
Thresh(λ)	0.21
FD	0.29
FD+Thresh(λ)	0.47
Triangle	0.69
Otsu	0.72
CNT	0.74
SubMax(δ)	0.96
FD+Triangle	1.09
Yen	1.10
isodata	1.12
FD+Otsu	1.15
AdaptMean	1.21
FD+AdaptMean	1.40
FD+isodata	1.50
FD+Yen	1.51
FD+AdaptGauss	1.66
FD+CNT	1.67
MOG2	2.31
KNN	2.59
FD+MOG2	3.32
MOG	5.17
FD+KNN	5.19
FD+MOG	5.33
FD+Li	7.02
Sauvola	8.36
GMG	8.44
Li	9.21
FD+GMG	9.39
AdaptGauss	15.77
FD+LSBP	16.00
LSBP	16.83
FD+Sauvola	23.74
GSOC	267.37
FD+GSOC	296.03

into foregrounds and backgrounds. All three algorithms exhibit foreground pixels that span the entire horizontal length of the cross-section and include false positives during the time the laser is off.

5.2. *Computational speed*

As indicated by Table 3 and Figure 8, the Thresh(λ), FD, and FD+Thresh(λ) algorithms achieved the fastest average computation time per image due to their low computational complexity. The thresholding algorithms Triangle and Otsu as well as the CNT and SubMax(δ) algorithms also demonstrated fast computation with sub-millisecond times. The remaining algorithms took anywhere between 1 to 25 milliseconds to process each image. The only algorithms that required considerably more time were the GSOC algorithm and its combination algorithm FD+GSOC. The computational speed of these two algorithms was an order of magnitude higher than any other algorithm.

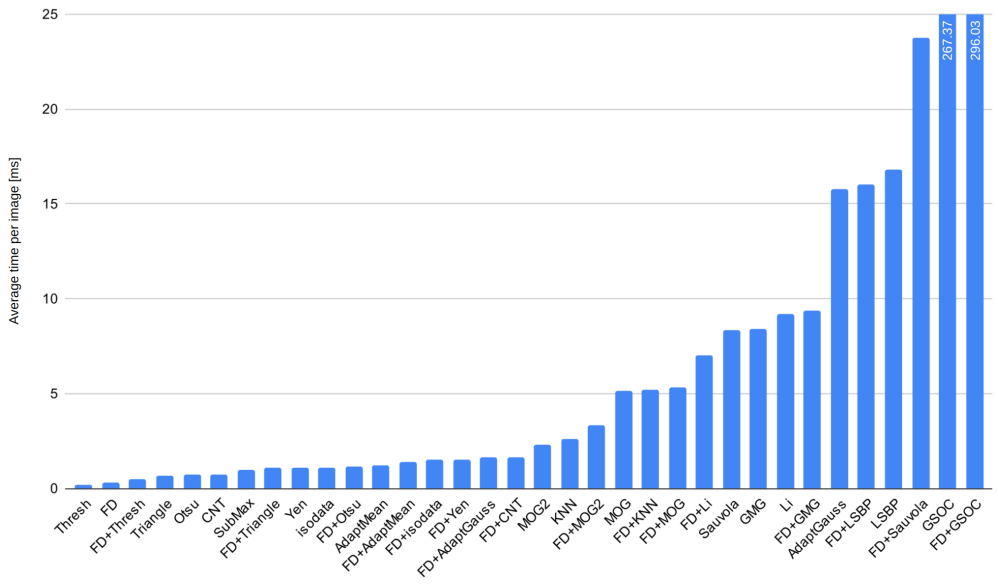


Figure 8: Average computation time of tested algorithms.

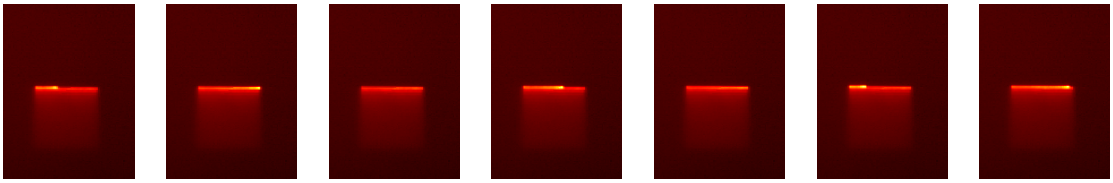
For some of the thresholding algorithms, the speed of the individual thresholding algorithm was higher than the speed of the corresponding combination algorithm. Since the thresholding step in the combination algorithm eliminates some of the nonzero pixels, the decrease in speed is likely due to a smaller number of nonzero pixels present in each image.

5.3. *Spatter detection*

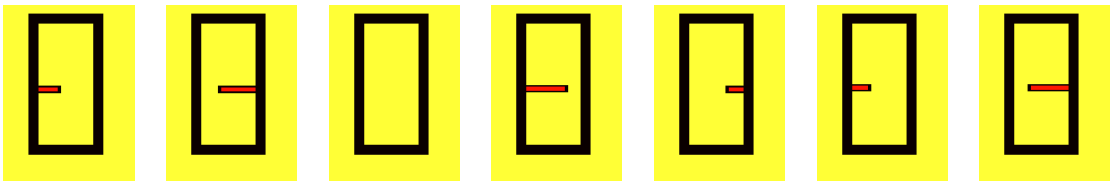
Figure 9 shows the binary composite image for each algorithm. The white colors represent nonzero pixel values, while the black colors represent values of zero. The dashed boxes are the projected contours and indicate the area of the upper-left cross-section from Figure 6b.

For the algorithms of the first row (MOG, MOG2, CNT, GSOC, and Thresh) and the KNN algorithm in the second row of Figure 9, the areas of the nonzero pixels approximately match the areas of the projected contours. The nonzero pixels in the composite images from these algorithms are either inside the projected contours or part of the overflow of nonzero pixels around the contours. Aligned with our manual inspection of the segmented results, these algorithms did not exhibit any artifacts in their foregrounds. Each foreground only featured nonzero pixels that approximately represented the most recent laser track changes on the build surface. Note that KNN has a single isolated nonzero pixel laying outside of the contour. Based on our manual inspection, this detection stems from general segmentation problems of the KNN algorithm due to poor segmentation accuracy (see Section 5.1).

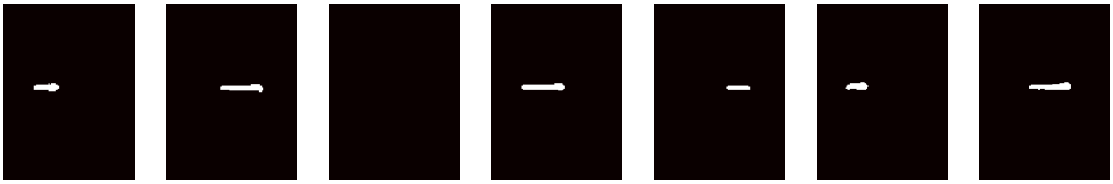
The remaining algorithms were either significantly influenced by the spatter or had general segmentation problems. The segmented results from these algorithms exhibited numerous artifacts in each foreground. These artifacts may stem from the ejected spatter or general segmentation problems of the algorithm. As a result, the



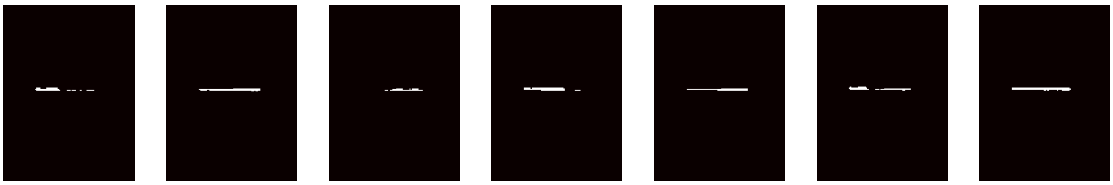
(a) Cropped raw infrared images from the SLS build (images from the same layer as Figure 6a)



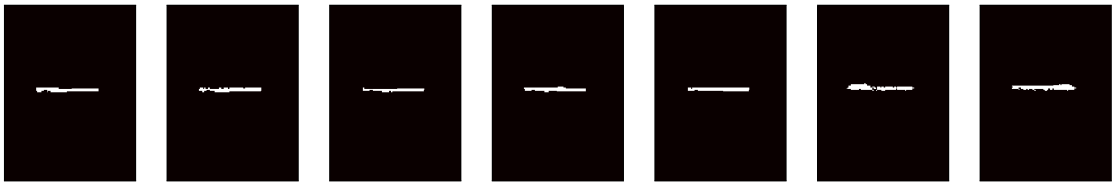
(b) Segmentation ground truth. Black marks the buffer zone, red the foreground, and yellow the background.



(c) FD + Thresh(λ)



(d) Thresh(λ)



(e) MOG

(f) Otsu's method

Figure 7: Segmentation results of different algorithms for an exemplary sequence of infrared images. The images are cropped to the area of the first cross-section.

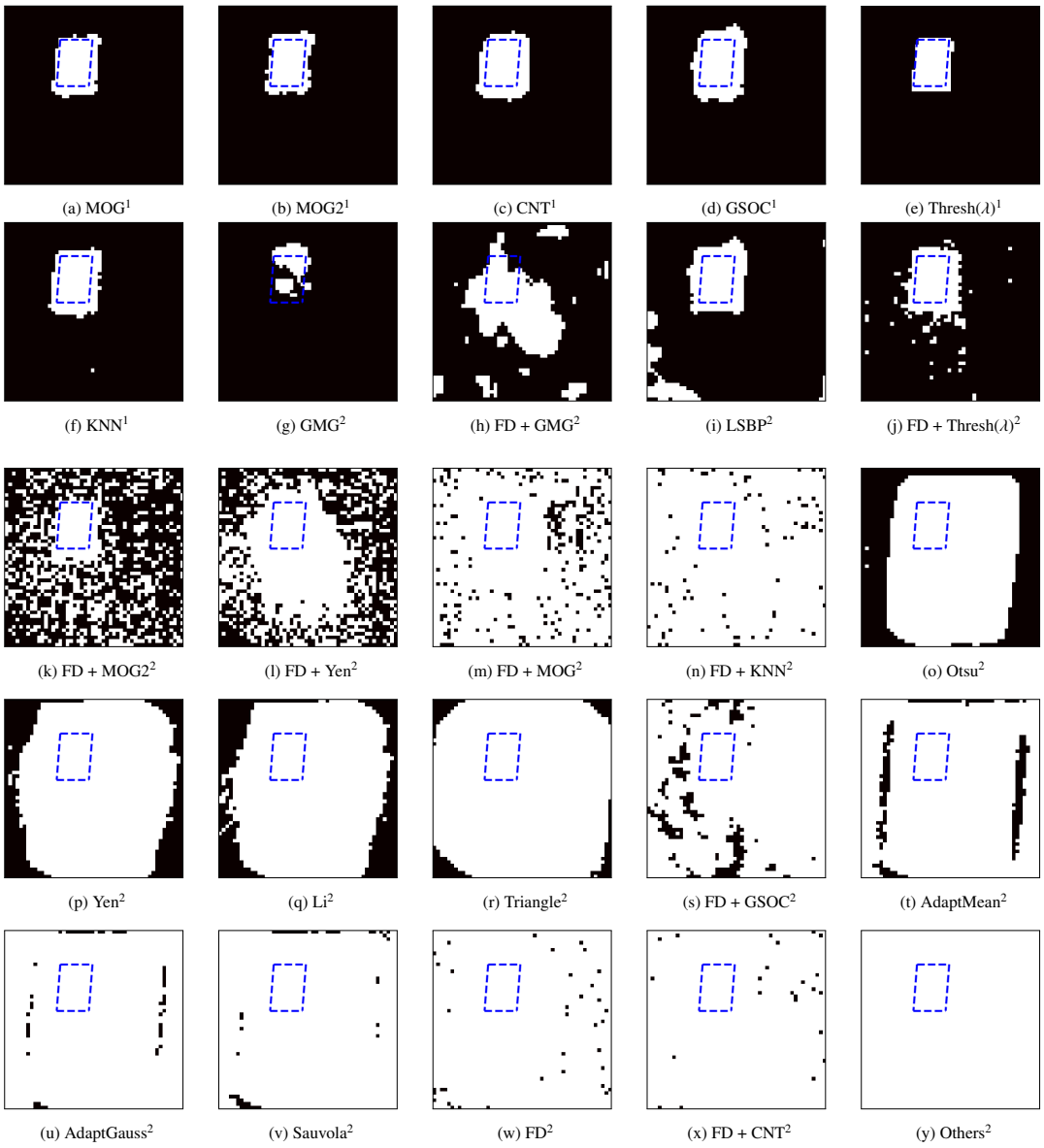


Figure 9: Binary composite images with projected contours for the SLM dataset. Superscript 1 indicates no spatter detection was observed in the foregrounds, superscript 2 indicates that the foregrounds exhibited were impacted by ejected spatter or had general segmentation problems.

composite images for most of these algorithms exhibit numerous nonzero pixels outside of the projected contours, some even consisting almost entirely of nonzero pixels. Only the Thresh(λ) and GMG algorithms did not exhibit this characteristic. Instead, these algorithms missed foreground pixels during scanning.

6. Discussion

We found that the combination of frame differencing with common thresholding and background segmentation algorithms performed best for finding the most recent changes of the laser tracks in each infrared image. Especially the FD+Thresh(λ) algorithm, which combines frame differencing with simple thresholding, performed better than any other segmentation algorithm while also maintaining one of the highest speeds for processing each image. Furthermore, we found that all of the popular thresholding algorithms that were recently used in LPBF studies, such as Otsu's method, performed poorly on the segmentation accuracy test.

The FD+Thresh(λ) algorithm has the advantage that the λ parameter is relatively easy to tune compared to the other segmentation algorithms. If the desired width for the detected laser track should be thinner, λ can be

increased to eliminate more foreground pixels from the border of the laser track; if the desired width should be bigger, then λ can be decreased. However, there might be LPBF process setups where the default λ does not work well, and calibrating the algorithm is not possible. In this case, the parameterless combination algorithm FD+Yen is a promising alternative that has similar speeds as the FD+Thresh(λ) algorithm and still has better performance than the baseline algorithm Thresh(λ).

Algorithms like FD+Thresh(λ) can be readily deployed to address several severe process limitations for LPBF processes. First, the FD+Thresh(λ) algorithm can be used to online monitor the most recent laser track changes with respect to different process signatures, such as temperature distribution and geometrical dimensions. Even the entire laser tracks and the heat-affected zone could be monitored with the help of the segmented outputs, as described in Section 2. Second, the segmented outputs can be used to reduce noise and dimensionality of the data for data-driven methods by eliminating the uninformative portions of the data, such as the backgrounds and the images where the laser is off. This pre-processing of the data can significantly increase the performance of the data-driven methods. Finally, the promise of extracting the informative portions of the images can also be applied to store the images efficiently. Since high-speed infrared cameras produce massive amounts of data, using the segmented outputs to store only the informative portions of the data significantly reduces the memory footprint and addresses current data storage limitations.

The second key finding of this study is that only a few algorithms (MOG, MOG2, CNT, GSOC, Thresh(λ), KNN) successfully excluded spatter from the foregrounds. However, none of these algorithms were among the top-performing algorithms in Section 5.1; these six algorithms only had mediocre to poor segmentation accuracy. Nevertheless, the baseline algorithm Thresh(λ) still approximately detected the changes in the laser tracks and may prove useful for processes with strong spatter characteristics, where an approximate detection of the most recent changes of the laser tracks suffices.

With regards to the qualitative nature of the manual inspection, it is important to note that the manual inspection only sufficed to identify which algorithms do *not* detect spatter in their foregrounds. The manual inspection was not sufficient to differentiate between poor segmentation accuracy and spatter detection for the algorithms with artifacts in the foregrounds. Some of the algorithms that exhibited artifacts in the foregrounds on the SLM dataset may perform differently with non-default parameters. Further investigation is needed to quantify the degree of spatter detection for each algorithm and differentiate between segmentation accuracy and spatter detection. Nevertheless, the current findings may provide important insights into the effect of ejected spatter on the segmentation algorithms.

Overall, based on the two key findings of this study, none of the algorithms currently possess high segmentation accuracy, high computational speed while also being free of foreground artifacts for processes with strong spatter characteristics. Finding segmentation algorithms that combine these characteristics still requires further investigation. Investigating the degree of spatter detection may also be useful for applications where spatter detection is desired.

7. Conclusion

We studied different segmentation algorithms that separate each infrared image into a foreground and background to address key process limitations for the laser powder bed fusion (LPBF) processes. We present a summary of the results in Table A.5 and A.6. By evaluating each algorithm based on segmentation accuracy and computational speed, we found that the combination of frame differencing methods with simple thresholding, called FD+Thresh(λ), works best for the LPBF processes. This algorithm had the best segmentation accuracy and also one of the fastest computational speeds. Due to its simple implementation and intuitive parameter, the FD+Thresh(λ) algorithm can be readily applied to online monitor the most recent laser track changes and extract the informative portions of the infrared images for pre-processing and efficient data storage.

We also qualitatively analyzed the spatter detection for each algorithm as the ejected spatter can significantly influence the image segmentation. We found that only a few algorithms did not include spatter in the foregrounds. However, those algorithms possessed mediocre segmentation accuracy at best. Further investigations are needed to find algorithms that combine high segmentation accuracy and computational speed with robustness against spatter detection.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1646522.

The authors would like to thank Alexander Shkoruta and Dr. Sandipan Mishra at the Rensselaer Polytechnic Institute for providing the Selective Laser Melting data.

The authors would also like to acknowledge the instrumental research support of the US Department of Defense (DoD) Air Force Research Laboratory (AFRL) Project # FA8650-17-C-5716 P00003: Laser Additive Manufacturing Pilot Scale (LAMPS) III: Advanced Process Monitoring and Control (PI: Scott Fish)

References

[1] B. Song, X. Zhao, S. Li, C. Han, Q. Wei, S. Wen, J. Liu, Y. Shi, Differences in microstructure and properties between selective laser melting and traditional manufacturing for fabrication of metal parts: A review, Frontiers of Mechanical Engineering 10 (2015) 111–125. doi:10.1007/s11465-015-0341-2.

[2] J.-P. Kruth, B. Vandenbroucke, J. Vaerenbergh, P. Mercelis, Benchmarking of different sls/slm processes as rapid manufacturing techniques, IEEE Electron Device Letters - IEEE ELECTRON DEV LETT 10 (01 2005).

[3] A. Uriondo, M. Esperon Miguez, S. Perinpanayagam, The present and future of additive manufacturing in the aerospace sector: A review of important aspects, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 229 (01 2015). doi:10.1177/0954410014568797.

[4] S. A. Tofail, E. P. Koumoulos, A. Bandyopadhyay, S. Bose, L. O'Donoghue, C. Charitidis, Additive manufacturing: scientific and technological challenges, market uptake and opportunities, Materials Today 21 (1) (2018) 22 – 37. doi:10.1016/j.mattod.2017.07.001.

[5] T. Pereira, J. V. Kennedy, J. Potgieter, A comparison of traditional manufacturing vs additive manufacturing, the best method for the job, Procedia Manufacturing 30 (2019) 11 – 18, Digital Manufacturing Transforming Industry Towards Sustainable Growth. doi:10.1016/j.promfg.2019.02.003.

[6] G. Tapia, A. Elwany, A Review on Process Monitoring and Control in Metal-Based Additive Manufacturing, Journal of Manufacturing Science and Engineering 136 (6), 060801 (10 2014). doi:10.1115/1.4028540.

[7] T. Spears, S. Gold, In-process sensing in selective laser melting (slm) additive manufacturing, Integrating Materials and Manufacturing Innovation 5 (12 2016). doi:10.1186/s40192-016-0045-4.

[8] M. Grasso, B. Colosimo, Process defects and in-situ monitoring methods in metal powder bed fusion: a review, Measurement Science and Technology 28 (2017) 1–25. doi:10.1088/1361-6501/aa5c4f.

[9] H. Krauss, T. Zeugner, M. F. Zaeh, Layerwise monitoring of the selective laser melting process by thermography, Physics Procedia 56 (2014) 64 – 71, 8th International Conference on Laser Assisted Net Shape Engineering LANE 2014. doi:https://doi.org/10.1016/j.phpro.2014.08.097.

[10] M. Grasso, A. G. Demir, B. Colosimo, In situ monitoring of selective laser melting of zinc powder via infrared imaging of the process plume, Robotics and Computer-Integrated Manufacturing 49 (2018) 229–239. doi:10.1016/j.rcim.2017.07.001.

[11] M. Grasso, B. Colosimo, A statistical learning method for image-based monitoring of the plume signature in laser powder bed fusion, Robotics and Computer-Integrated Manufacturing 57 (2019) 103 – 115. doi:https://doi.org/10.1016/j.rcim.2018.11.007.

[12] H. Krauss, C. Eschey, M. F. Zäh, Thermography for monitoring the selective laser melting process, 2012.

[13] E. Louvis, P. Fox, C. J. Sutcliffe, Selective laser melting of aluminium components, Journal of Materials Processing Technology 211 (2) (2011) 275 – 284. doi:https://doi.org/10.1016/j.jmatprotec.2010.09.019.

[14] Y.-L. Lo, B.-Y. Liu, H.-C. Tran, Optimized hatch space selection in double-scanning track selective laser melting process, The International Journal of Advanced Manufacturing Technology 105 (12 2019). doi:10.1007/s00170-019-04456-w.

[15] I. Yadroitsev, I. Yadroitsava, P. Bertrand, I. Smurov, Factor analysis of selective laser melting process parameters and geometrical characteristics of synthesized single tracks, Rapid Prototyping Journal 18 (2012) 201–208. doi:10.1108/13552541211218117.

[16] X. Shi, S. Ma, C. Liu, Q. Wu, Parameter optimization for ti-47al-2cr-2nb in selective laser melting based on geometric characteristics of single scan tracks, Optics & Laser Technology 90 (2017) 71 – 79. doi:https://doi.org/10.1016/j.optlastec.2016.11.002.

[17] I. Yadroitsev, A. Gusarov, I. Yadroitsava, I. Smurov, Single track formation in selective laser melting of metal powders, Journal of Materials Processing Technology 210 (12) (2010) 1624 – 1631. doi:https://doi.org/10.1016/j.jmatprotec.2010.05.010.

[18] N. Nadammal, S. Cabeza, T. Mishurova, T. Thiede, A. Kromm, C. Seyfert, L. Farahbod, C. Haberland, J. A. Schneider, P. D. Portella, G. Bruno, Effect of hatch length on the development of microstructure, texture and residual stresses in selective laser melted superalloy inconel 718, Materials & Design 134 (2017) 139 – 150. doi:https://doi.org/10.1016/j.matdes.2017.08.049.

[19] J. Robinson, I. Ashton, E. Jones, P. Fox, C. Sutcliffe, The effect of hatch angle rotation on parts manufactured using selective laser melting, Rapid Prototyping Journal 25 (10 2018). doi:10.1108/RPJ-06-2017-0111.

[20] S. Everton, M. Hirsch, P. Stavroulakis, R. Leach, A. Clare, Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing, Materials & Design 95 (2016) 431–445. doi:10.1016/j.matdes.2016.01.099.

[21] P. Yadav, O. Rigo, C. Arvieu, E. Guen, E. Lacoste, In situ monitoring systems of the slm process: On the need to develop machine learning models for data processing, Crystals 10 (2020) 524. doi:10.3390/cryst10060524.

[22] Y. Zhang, J. Fuh, Y. Dongsen, G.-S. Hong, In-situ monitoring of laser-based pbf via off-axis vision and image processing approaches, Additive Manufacturing 25 (10 2018). doi:10.1016/j.addma.2018.10.020.

[23] D. Ye, K. Zhu, J. Y. H. Fuh, Y. Zhang, H. G. Soon, The investigation of plume and spatter signatures on melted states in selective laser melting, Optics & Laser Technology 111 (2019) 395 – 406. doi:https://doi.org/10.1016/j.optlastec.2018.10.019.

[24] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man, and Cybernetics 9 (1) (1979) 62–66.

[25] H. Baumgartl, J. Tomas, R. Buettner, M. Merkel, A deep learning-based model for defect detection in laser-powder bed fusion using in-situ thermographic monitoring (02 2020). doi:10.1007/s40964-019-00108-3.

[26] Z. Xu, A. Nettekoven, A. Julius, U. Topcu, Graph temporal logic inference for classification and identification, 2019, pp. 4761–4768. doi:10.1109/CDC40024.2019.9029181.

[27] N. M. Nawi, W. H. Atomi, M. Rehman, The effect of data pre-processing on optimized training of artificial neural networks, Procedia Technology 11 (2013) 32 – 39, 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013. doi:https://doi.org/10.1016/j.protcy.2013.12.159.

[28] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, Data preprocessing for supervised learning, International Journal of Computer Science 1 (2006) 111–117.

[29] R. Williams, T. Ronneberg, M.-S. Pham, C. Davies, P. Hooper, A. Piglione, C. Jones, In situ thermography for laser powder bed fusion: Effects of layer temperature on porosity, microstructure and mechanical properties 30 (10 2019). doi:10.1016/j.addma.2019.100880.

[30] S. Clijsters, T. Craeghs, S. Buls, K. Kempen, J.-P. Kruth, In situ quality control of the selective laser melting process using a high-speed, real-time melt pool monitoring system, International Journal of Advanced Manufacturing Technology 75 (2014) 1089–1101. doi:10.1007/s00170-014-6214-8.
[31] L. Mazzoleni, A. G. Demir, L. Caprio, M. Pacher, B. Previtali, Real-time observation of melt pool in selective laser melting: Spatial, temporal and wavelength resolution criteria, IEEE Transactions on Instrumentation and Measurement PP (2019) 1–1. doi:10.1109/TIM.2019.2912236.
[32] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools (2000).
[33] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, the scikit-image contributors, scikit-image: image processing in Python, PeerJ 2 (2014) e453. doi:10.7717/peerj.453.
[34] K. Pulli, A. Baksheev, K. Kornyakov, V. Eruhimov, Real-time computer vision with opencv, Communications of the ACM 55 (2012) 61–69. doi:10.1145/2184319.2184337.
[35] T. Bouwmans, Traditional and recent approaches in background modeling for foreground detection: An overview, Computer Science Review 11 (05 2014). doi:10.1016/j.cosrev.2014.04.001.
[36] A. Sobral, A. Vacavant, A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos, Computer Vision and Image Understanding 122 (2014) 4–21. doi:10.1016/j.cviu.2013.12.005.
[37] P. Kaewtrakulpong, R. Bowden, An improved adaptive background mixture model for realtime tracking with shadow detection, Proceedings of 2nd European Workshop on Advanced Video-Based Surveillance Systems; September 4, 2001; London, U.K (05 2002). doi:10.1007/978-1-4615-0913-4_11.
[38] Z. Zivkovic, Improved adaptive gaussian mixture model for background subtraction, Vol. 2, 2004, pp. 28 – 31 Vol.2. doi:10.1109/ICPR.2004.1333992.
[39] Z. Zivkovic, F. van der Heijden, Efficient adaptive density estimation per image pixel for the task of background subtraction, Pattern Recognition Letters 27 (7) (2006) 773 – 780. doi:https://doi.org/10.1016/j.patrec.2005.11.005.
[40] A. Godbehere, A. Matsukawa, K. Goldberg, Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation, 2012, pp. 4305–4312. doi:10.1109/ACC.2012.6315174.
[41] L. Guo, D. Xu, Z. Qiang, Background subtraction using local svd binary pattern, 2016, pp. 1159–1167. doi:10.1109/CVPRW.2016.148.
[42] M. Sezgin, B. Sankur, Survey over image thresholding techniques and quantitative performance evaluation, Journal of Electronic Imaging 13 (2004) 146–168. doi:10.1117/1.1631315.
[43] C. Li, P. Tam, An iterative algorithm for minimum cross entropy thresholding, Pattern Recognition Letters 19 (8) (1998) 771 – 776. doi:https://doi.org/10.1016/S0167-8655(98)00057-9.
[44] T. W. Ridler, Picture thresholding using an iterative selection method., IEEE Transactions on Systems, Man, and Cybernetics 8 (1978) 630–632.
[45] Jui-Cheng Yen, Fu-Juay Chang, Shyang Chang, A new criterion for automatic multilevel thresholding, IEEE Transactions on Image Processing 4 (3) (1995) 370–378.
[46] G. W. Zack, W. E. Rogers, S. Latt, Automatic measurement of sister chromatid exchange frequency., The journal of histochemistry and cytochemistry : official journal of the Histochemistry Society 25 (1977) 741 – 753.
[47] J. Sauvola, M. Pietikäinen, Adaptive document image binarization, Pattern Recognition 33 (2) (2000) 225 – 236. doi:https://doi.org/10.1016/S0031-3203(99)00055-2.
[48] R. Radke, S. Andra, O. Al-Kofahi, B. Roysam, Image change detection algorithms: A systematic survey, IEEE transactions on image processing : a publication of the IEEE Signal Processing Society 14 (2005) 294–307. doi:10.1109/TIP.2004.838698.
[49] D. Lu, P. Mausel, E. Brondízio, E. Moran, Change detection techniques. int j remote sens, International Journal of Remote Sensing 25 (2004) 2365–2401. doi:10.1080/0143116031000139863.
[50] P. Rosin, E. Ioannidis, Evaluation of global image thresholding for change detection, Pattern Recognition Letters 24 (2003) 2345–2356. doi:10.1016/S0167-8655(03)00060-6.
[51] N. Chaki, S. Shaikh, K. Saeed, A Comprehensive Survey on Image Binarization Techniques, 2014, pp. 5–15. doi:10.1007/978-81-322-1907-1_2.
[52] P. Rosin, T. Ellis, Image difference threshold strategies and shadow detection 1 (11 1995). doi:10.5244/C.9.35.
[53] T. Bouwmans, F. Porikli, B. Höferlin, A. Vacavant, Handbook on "Background Modeling and Foreground Detection for Video Surveillance", 2014. doi:10.1201/b17223.
[54] A. Sobral, Bgslibrary: An opencv c++ background subtraction library, 2013. doi:10.13140/2.1.1740.7044.
[55] G. Yao, T. Lei, J. Zhong, P. Jiang, W. Jia, Comparative evaluation of background subtraction algorithms in remote scene videos captured by mwir sensors, Sensors 17 (2017) 1945. doi:10.3390/s17091945.
[56] T. Trnovszky, P. Sykora, R. Hudec, Comparison of background subtraction methods on near infra-red spectrum video sequences, Procedia Engineering 192 (2017) 887–892. doi:10.1016/j.proeng.2017.06.153.
[57] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with NumPy, Nature 585 (2020) 357–362. doi:10.1038/s41586-020-2649-2.

Appendix A. Tables

Table A.4: Parameters for the segmentation algorithms.

Algorithms	Type	Parameters
Thresh(λ)	calibrated	λ: 377.60
	default	λ: 295
FD+Thresh(λ)	calibrated	λ: 4.44
	default	λ: 3
SubMax(δ)	calibrated	δ: 125.81
	default	δ: 20
GSOC	calibrated	'hitsThresh': 5, 'nSamples': 506, 'propRate': 0.76, 'replaceRate': 0.01
	default	OpenCV default
FD+GSOC	calibrated	'hitsThresh': 2, 'nSamples': 596, 'propRate': 0.72, 'replaceRate': 0.058
	default	OpenCV default
CNT	calibrated	'maxStability': 6, 'minStability': 0
	default	OpenCV default
FD+CNT	calibrated	'maxStability': 6, 'minStability': 0
	default	0.03
MOG	calibrated	'backRatio': 0.47, 'history': 14, 'nmixtures': 290
	default	OpenCV default
FD+MOG	calibrated	'backRatio': 0.77, 'history': 29, 'nmixtures': 327
	default	OpenCV default
GMG	calibrated	'thresh': 0.81
	default	'thresh': 0.8
FD+GMG	calibrated	'thresh': 0.03
	default	'thresh': 0.8
MOG2	calibrated	'history': 67, 'thresh': 10.10
	default	OpenCV default
FD+MOG2	calibrated	'history': 27, 'thresh': 1.10
	default	OpenCV default
LSBP	calibrated	'radius': 200, 'samples': 266
	default	OpenCV default
FD+LSBP	calibrated	'radius': 25, 'samples': 700
	default	OpenCV default
KNN	calibrated	'history': 68, 'thresh': 298.81
	default	OpenCV default
FD+KNN	calibrated	'history': 2, 'thresh': 4.40
	default	OpenCV default
AdaptMean	calibrated	'C': 0, 'box': 451
	default	OpenCV default
FD+AdaptMean	calibrated	'C': 66, 'box': 263
	default	OpenCV default
AdaptGauss	calibrated	'C': 2, 'box': 387
	default	OpenCV default
FD+AdaptGauss	calibrated	'C': 78, 'box': 7
	default	OpenCV default
Sauvola	calibrated	'box': 79, 'k': 0.41
	default	scikit-image default
FD+Sauvola	calibrated	'box': 37, 'k': 0.23
	default	scikit-image default

Table A.5: Performance summary of all algorithms (Part 1).

Algorithms		Segmentation accuracy			Computation time	Spatter robust?
		Precision	Recall	F_1-score	[ms]	
Thresh(λ)	default	0.09	0.98	0.17	**0.21**	No
	calibrated	0.61	0.63	0.62		
FD	–	0.00	1.00	0.00	0.29	No
MOG	default	0.21	0.90	0.35	5.17	**Yes**
	calibrated	0.46	0.81	0.58		
FD+MOG	default	0.54	0.68	0.60	5.33	No
	calibrated	0.66	0.67	0.67		
MOG2	default	0.10	0.96	0.17	2.31	**Yes**
	calibrated	0.17	0.87	0.29		
FD+MOG2	default	0.21	0.79	0.33	3.32	No
	calibrated	0.22	0.93	0.35		
KNN	default	0.06	0.97	0.10	2.59	**Yes**
	calibrated	0.08	0.96	0.14		
FD+KNN	default	0.12	0.81	0.21	5.19	No
	calibrated	0.13	0.95	0.22		
GMG	default	0.17	0.89	0.29	8.44	No
	calibrated	0.17	0.89	0.29		
FD+GMG	default	0.23	0.80	0.36	9.39	No
	calibrated	0.24	0.95	0.38		
CNT	default	0.02	0.94	0.04	0.74	**Yes**
	calibrated	0.42	0.81	0.56		
FD+CNT	default	0.02	0.73	0.03	1.67	No
	calibrated	0.67	0.73	0.70		
FD+Thresh(λ)	default	0.85	0.93	**0.88**	0.47	No
	calibrated	0.95	0.90	**0.93**		
SubMax(δ)	default	0.00	0.07	0.00	0.96	No
	calibrated	0.00	0.65	0.01		
GSOC	default	0.07	1.00	0.13	267.37	**Yes**
	calibrated	0.66	0.55	0.60		
FD+GSOC	default	0.26	0.87	0.41	296.03	No
	calibrated	0.91	0.78	0.84		
LSBP	default	0.16	0.73	0.26	16.83	No
	calibrated	0.18	0.68	0.29		
FD+LSBP	default	0.01	0.59	0.01	16.00	No
	calibrated	0.01	0.56	0.01		

Table A.6: Performance summary of all algorithms (Part 2).

Algorithms		Segmentation accuracy			Computation time	Spatter robust?
		Precision	Recall	F_1-score	[ms]	
AdaptMean	default	0.00	0.85	0.00	1.21	No
	calibrated	0.00	1.00	0.00		
AdaptGauss	default	0.00	0.77	0.00	15.77	No
	calibrated	0.00	1.00	0.00		
FD+AdaptMean	default	0.00	0.71	0.00	1.40	No
	calibrated	0.00	1.00	0.00		
FD+AdaptGauss	default	0.00	0.60	0.00	1.66	No
	calibrated	0.00	1.00	0.00		
Otsu	–	0.00	0.96	0.01	0.72	No
Li	–	0.00	1.00	0.00	9.21	No
Sauvola	default	0.00	0.91	0.00	8.36	No
	calibrated	0.00	1.00	0.00		
FD+Sauvola	default	0.35	0.79	0.48	23.74	No
	calibrated	0.39	0.90	0.54		
Triangle	–	0.01	1.00	0.02	0.69	No
Yen	–	0.01	0.91	0.02	1.10	No
isodata	–	0.00	1.00	0.00	1.12	No
FD+Otsu	–	0.62	0.59	0.60	1.15	No
FD+Li	–	0.36	0.94	0.52	7.02	No
FD+Triangle	–	0.44	0.95	0.60	1.09	No
FD+Yen	–	0.73	0.84	0.78	1.51	No
FD+isodata	–	0.62	0.59	0.61	1.50	No