DIGITAL MANUFACTURING

Group Name: Team 7

Group Members:

- Harshal B Varpe
- Kedar A Sudhalkar
- Shubham Thorat

CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

MAY 4, 2022

SEMESTER PROJECT MILESTONE 5

Index

Table of Figures

Abstract

Technology has been advancing very rapidly in recent years. Advent in technology has changed many industries from the ground up. The manufacturing industry is one such industry. In the recent decade, we have seen how the manufacturing industry, which focused on manual labor, has slowly moved towards automation and keeping humans away from Dull, Dangerous, and Demanding jobs. One of the ways the manufacturing industry is changing is the novel manufacturing process.

Powder-based fusion additive manufacturing is one such novel process. In this process, shapes of complex geometries are formed with the help of a laser beam that moves in a 2D plane. Advanced power electronics have made it possible to control the intensity of laser to such a degree that different material powers can be used to form materials with different structural properties.

In this project, we tried to solve one of the problems with this process. First, we start with a brief introduction of the project in the introduction section. The methodology section covers our approach to the solution. In the next section, Results, we talk about our results. In the last section, we discuss if we have managed to cover the project's scope and if the goals have been met. Moreover, we add a few comments on how the project could be made better and suggestions for next year's students.

Introduction

Problem Statement: Melt-pool size prediction for Powder-Bed Fusion Additive Manufacturing (PBF AM)

Why:

- Melt Pool Characteristics are highly correlated to as-built part structure and properties.
- Melt pool measurements provide a valid process signature for both process monitoring and control.
- Establishing the relationship between the process parameters is a key for melt pool-based process monitoring and control.

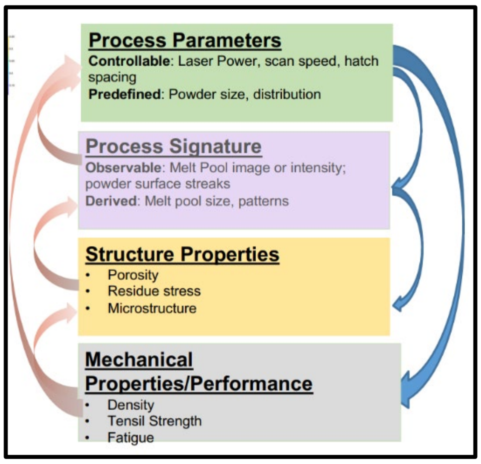

Process Parameters
Controllable: Laser Power, scan speed, hatch spacing
Predefined: Powder size, distribution
Process Signature
Observable: Melt Pool image or intensity; powder surface streaks
Derived: Melt pool size, patterns
Structure Properties
Porosity
Residue stress
Microstructure
Mechanical Properties/Performance
Density
Tensil Strength
Fatigue

Figure 1. Relationship between various parameters

- Data analysis for AM has the potential to enhance process understanding, process optimization, real-real feedback process variation control and quality assurance and certification with reduced time and cost.
- It is necessary to improve the controls of the process because it is a relatively new manufacturing process that requires a high level of precision in the control. Currently the process is subject to errors and faults in the process (splatter, melt pool size). It is necessary to dive into the data and extract what relationships and features exist in the current control process so they can be accounted for, and the process improved.
- Additive manufacturing is used to create custom parts with complex geometry, with a homogeneous material structure throughout the part. Currently there is no way to produce such parts with current subtractive manufacturing technologies. The homogeneous material structure is necessary for certain applications, so improving control strategies is vital to making defect free parts. Once the control process is well understood, the technology can begin to become more mainstream.

Goal of the project:

- Predicting the melt-pool size (area) based on build commands, to avoid the time and cost associated with experiments for process planning.
- This will be achieved by developing sets of data analytics tools, predictive models, and process control and optimization algorithms for PBF processes.
- Spatial alignment of the data to a reasonable degree

- Accurately identifying relationships between build commands and melt pool size and quality
- Identifying relationships between build commands and melt pool characteristics and ranking them based on significance
- Developing an accurate and improved control strategy that can be implemented

Measurable goals:

- Accurately finding the melt pool and melt pool size (95% accuracy for melt pool size) from image analysis within
- Correctly identifying melt pool splatter size and location
    - 95% accuracy on splatter size, correctly identifying all major splatters

Scope of work:

- Spatial aligning of MPM data before conducting correlation.
- Feature extraction from the multi rate, multi scale AM in-process data.
- Determining the relationship between build commands and in-process measurements.
- Developing real-time control, or layerwise control strategies based on in-process measurements for an AM process.

Project timeline:

In this section, we have given a brief overview of how we broke down the problem into several tasks and distributed responsibilities. We also mention the proposed timeline vs actual timeline for the completion of the broken-down tasks. Wherever we took more time than we anticipated we have explained the adversities faced while dealing with the problems. All the members performed their duties as required and scheduled.

In the later part of this section a cohesive run-down of the entire solution process is given.

- **Task 1:** Getting a clear idea of every process parameter like laser power, scan speed, hatch spacing, powder size and distribution. Getting a general idea of how the data is collected and stored, especially with regards to the different frequencies of data collection for 2 different data types (build commands @100kHz and co-axial images 20kHz). **Original Timeline – 2/22/2022**
    - **Challenges and Task Update -** We underestimated the sheer amount of data there was to explore and how the datasets were structured. This task took longer than expected due to the complexity of the datasets. We now have a good understanding of the dataset and what is involved in processing it. Some of the things we have learned are that the co-axial images were not taken at a rate of 20 kHz but were instead taken at a rate of 10 kHz. There is also only one Trigger value (2). In one of the reports given to us, we learned that the Trigger value is associated with a channel on which the image was taken. We have also learned that only 1 channel was connected in creation of the dataset, which is why the only trigger value observed is 2.
    - **Updated Timeline** – Completed 3/17/2022
    - **Completion Date** – Completed 3/17/2022
    - **Responsible Members** – all members

- **Task 2:** Undertaking the first challenge of spatial alignment of data. Relate where the build commands are taking place on the layer and where the melt pool is in relation to the layer. Relate where each image is taken in relation to the build commands and location on the layer. **Original Timeline – 3/8/2022**

- **Challenges and Task Update -** This process is ongoing, and we are behind our goal for completion of this task. As we work to complete this task, we are also learning and understanding the data. We understand how this task should be completed and are in the process of implementing this. To complete this task, we will assign the coordinate values of the build command with a trigger value to the associated image. Essentially, if an image was taken at a given build command, we will assign the XY coordinates to that image file. This task is taking longer than expected due to us not having the python skills or knowledge to fully implement this until recently, however we are confident that we can now tackle this task in a timely manner.
- **Updated Timeline** – Complete this task by March 22nd
- **Completion Date** – Completed 04/01/2022
- **Responsible Members** – Shubham and Kedar

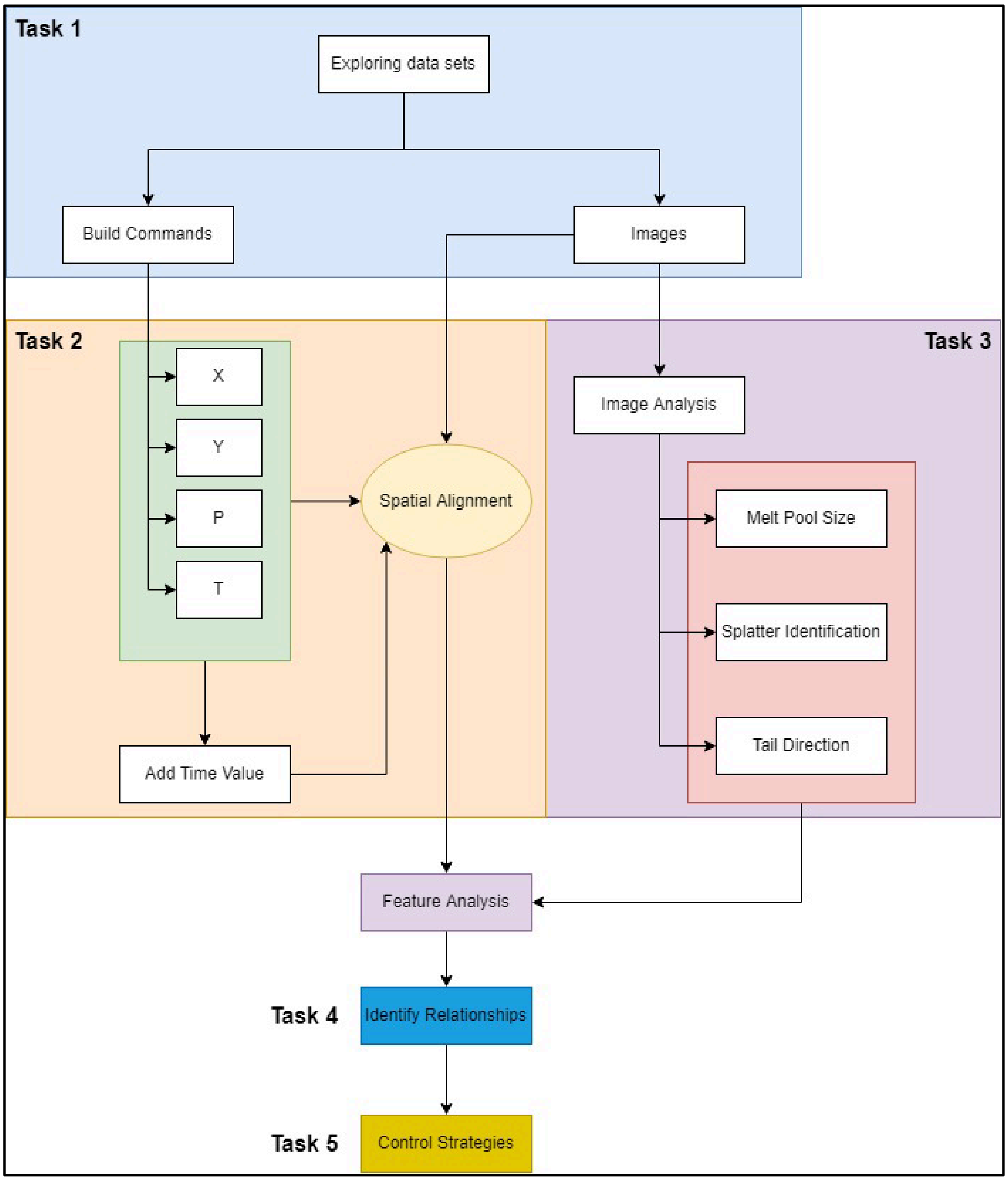
- **Task 3:** Feature extraction from build commands and co-axial images. This involves the image analysis of the melt pool to obtain melt pool size, splatter identification, melt pool tail length and direction relative to the build commands. **Original Timeline – 3/24/2022**
  - **Challenges and Task Update -** Being behind on task 2 pushes the deadline for task 3 back slightly. To account for this delay, we will work on task 3 in parallel with task 2 over the next 2 weeks. To get a head start on this task, we will work on image analysis in parallel with task 2. To start, we will start with melt pool identification and size, and move on to splatter identification. As task 2 progresses, we will be able to assign those images and their analysis to build commands and locations on the layer.
  - **Updated Timeline –** Targeted completion by March 29th, first class date after spring break. To adjust for our delays, we will use spring break to gain ground on multiple tasks.
  - **Completion Date** – Completed 04/01/2022
  - **Responsible Members** – Harshal

- **Task 4:** Define and analyze relationships between the build commands and in-process measurements, as well as how build commands affect melt pool size and splatter development. **Original Timeline – 4/7/2022**
  - **Challenges and Task Update -** This task is dependent on the completion of task 3. Since our timeline is delayed, the completion of this task will be pushed back slightly. However, the complexity of this task is now less daunting than it may have originally seemed. Having done feature extraction through class assignments, we have learned how to do this, and an outline of what to do has been completed. The main goals are to establish relationships between laser power and melt pool size, laser power and splatter, laser speed and melt pool size (and tail length), and laser speed and splatter. We will also be checking for multicollinearity between laser power, Speed, and splatter. Splatter will be coded as a categorical variable (true or false) for each image.
  - **Updated Timeline –** April 12th
  - **Completion Date** – Completed 04/20/2022
  - **Responsible Members** – Each member will be responsible for at least 1 relationship

- **Task 5**: Develop a real time control strategy, or a layer-based control strategy, or process optimization based on in process measurements, feature extraction, and the relationship analysis from previous tasks. This involves managing the laser intensity and Speed of the laser to manage melt pool size, limit splatter, and controlling for optimal tail length of the melt pool. **Original Timeline – 5/3/2022**

- **Challenges and Task Update -** This task remains unchanged in both scope and timeline. Once relationships are identified and correlations observed, we can dictate a control strategy to avoid splatter and accurately control melt pool size by controlling laser speed and power.
- **Updated Timeline** - Incomplete
- **Responsible Members** – Each member should contribute to the control strategy using the same relationships identified in the previous task.

Flowchart of Tasks



Because out project had no physical aspects, we did not require any hardware. Our requirements or resources were of intangible nature. Some of the required resourced are listed below.

Required resources:

- Advice and details of the dataset and process itself from Dr. Krugh
- Advice on how to implement the coding required
- Knowledge of sensor fusion and how to extract meaningful data and relationships from the raw data given to the group
- Advice on implementing OpenCV to help with image analysis, and how to code image analysis in python
- How to store the results of our analysis and feature extraction and how to display our results

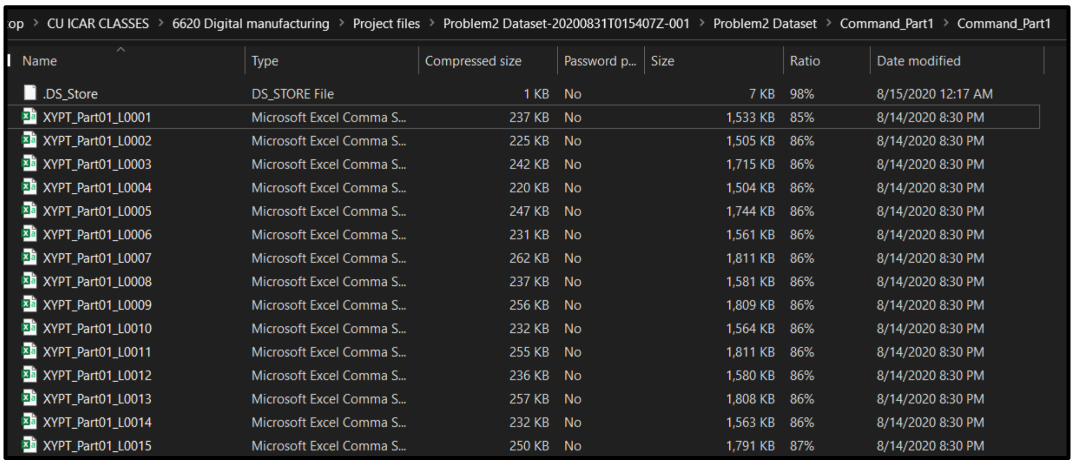
Methodology

Approach to Solution

In this section we would like discuss approach taken to solve this problem.

We had a dataset that was mainly divided into two parts. One part being CSV files and other part being images. For each layer there was a CSV file containing information about the movement of laser in 2D plane, laser intensity during the movement and trigger value to indicate if the picture was taken during certain movement. For each CSV file, there is also a folder containing all the images taken during the execution of those commands.

- The following screenshot shows some of the build command CSV files in dataset 001. There is a build command file for each layer, from layer 1-250.

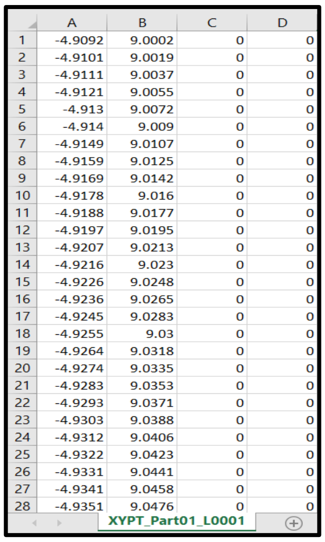


op > CU ICAR CLASSES > 6620 Digital manufacturing > Project files > Problem2 Dataset-20200831T015407Z-001 > Problem2 Dataset > Command_Part1 > Command_Part1

Name	Type	Compressed size	Password p...	Size	Ratio	Date modified
.DS_Store	DS_STORE File	1 KB	No	7 KB	98%	8/15/2020 12:17 AM
XYPT_Part01_L0001	Microsoft Excel Comma S...	237 KB	No	1,533 KB	85%	8/14/2020 8:30 PM
XYPT_Part01_L0002	Microsoft Excel Comma S...	225 KB	No	1,505 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0003	Microsoft Excel Comma S...	242 KB	No	1,715 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0004	Microsoft Excel Comma S...	220 KB	No	1,504 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0005	Microsoft Excel Comma S...	247 KB	No	1,744 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0006	Microsoft Excel Comma S...	231 KB	No	1,561 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0007	Microsoft Excel Comma S...	262 KB	No	1,811 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0008	Microsoft Excel Comma S...	237 KB	No	1,581 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0009	Microsoft Excel Comma S...	256 KB	No	1,809 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0010	Microsoft Excel Comma S...	232 KB	No	1,564 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0011	Microsoft Excel Comma S...	255 KB	No	1,811 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0012	Microsoft Excel Comma S...	236 KB	No	1,580 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0013	Microsoft Excel Comma S...	257 KB	No	1,808 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0014	Microsoft Excel Comma S...	232 KB	No	1,563 KB	86%	8/14/2020 8:30 PM
XYPT_Part01_L0015	Microsoft Excel Comma S...	250 KB	No	1,791 KB	87%	8/14/2020 8:30 PM

Figure 2 Screenshot of layerwise CSV File

- The next screenshot shows the inside of the build command file for Layer 1 (L0001). The data is structured in 4 columns. The columns represent X-coordinate (mm), Y-coordinate (mm), Laser Power (kW), and Trigger. Trigger is non-dimensional and represents the channel that an image was sent to. 0 means no picture taken, and 2 means an image was sent to channel 2. Since only channel 2 was connected, 2 represents the fact that an image was taken. This screenshot only shows an area where the laser was not active, so there were no images taken.

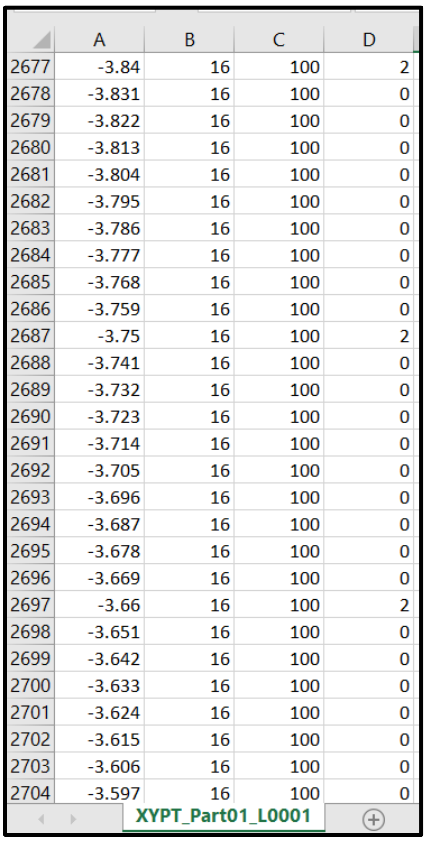


	A	B	C	D
1	-4.9092	9.0002	0	0
2	-4.9101	9.0019	0	0
3	-4.9111	9.0037	0	0
4	-4.9121	9.0055	0	0
5	-4.913	9.0072	0	0
6	-4.914	9.009	0	0
7	-4.9149	9.0107	0	0
8	-4.9159	9.0125	0	0
9	-4.9169	9.0142	0	0
10	-4.9178	9.016	0	0
11	-4.9188	9.0177	0	0
12	-4.9197	9.0195	0	0
13	-4.9207	9.0213	0	0
14	-4.9216	9.023	0	0
15	-4.9226	9.0248	0	0
16	-4.9236	9.0265	0	0
17	-4.9245	9.0283	0	0
18	-4.9255	9.03	0	0
19	-4.9264	9.0318	0	0
20	-4.9274	9.0335	0	0
21	-4.9283	9.0353	0	0
22	-4.9293	9.0371	0	0
23	-4.9303	9.0388	0	0
24	-4.9312	9.0406	0	0
25	-4.9322	9.0423	0	0
26	-4.9331	9.0441	0	0
27	-4.9341	9.0458	0	0
28	-4.9351	9.0476	0	0

XYPT_Part01_L0001

Figure 3 Structure of CSV file

- The next screenshot is from the same layer, but in an area where the laser was active, as seen in the power column. Here we can see the rate at which images were taken (every 10 columns). The build commands are sampled at a rate of 100 kHz, so the data confirms that the images were taken at a rate of 10 kHz, not 20 kHz as stated in the introduction document for the project.

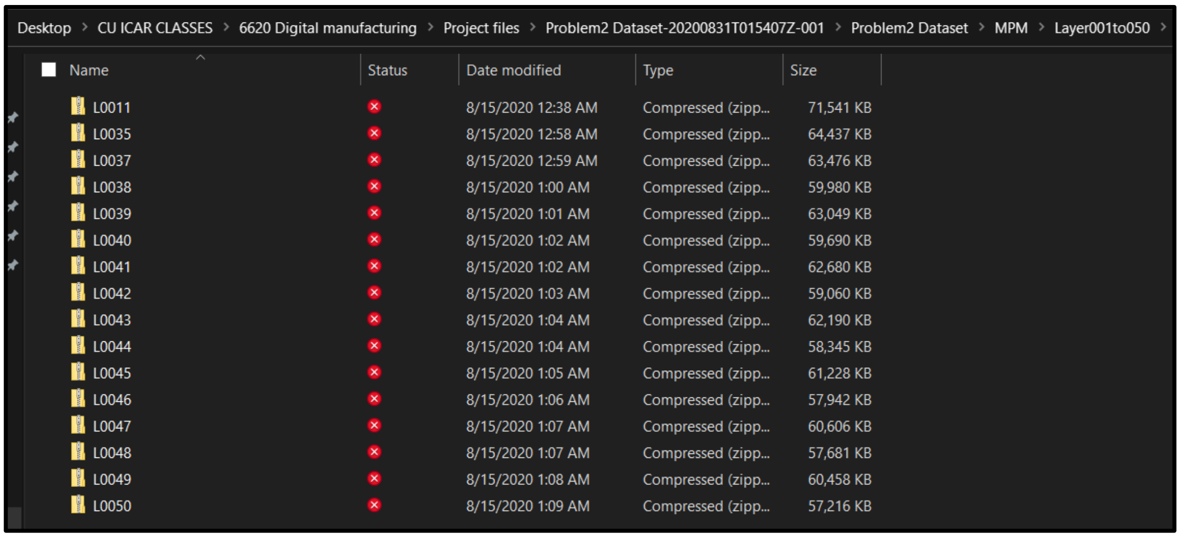


	A	B	C	D
2677	-3.84	16	100	2
2678	-3.831	16	100	0
2679	-3.822	16	100	0
2680	-3.813	16	100	0
2681	-3.804	16	100	0
2682	-3.795	16	100	0
2683	-3.786	16	100	0
2684	-3.777	16	100	0
2685	-3.768	16	100	0
2686	-3.759	16	100	0
2687	-3.75	16	100	2
2688	-3.741	16	100	0
2689	-3.732	16	100	0
2690	-3.723	16	100	0
2691	-3.714	16	100	0
2692	-3.705	16	100	0
2693	-3.696	16	100	0
2694	-3.687	16	100	0
2695	-3.678	16	100	0
2696	-3.669	16	100	0
2697	-3.66	16	100	2
2698	-3.651	16	100	0
2699	-3.642	16	100	0
2700	-3.633	16	100	0
2701	-3.624	16	100	0
2702	-3.615	16	100	0
2703	-3.606	16	100	0
2704	-3.597	16	100	0

XYPT_Part01_L0001

Figure 4 CSV file with triggers

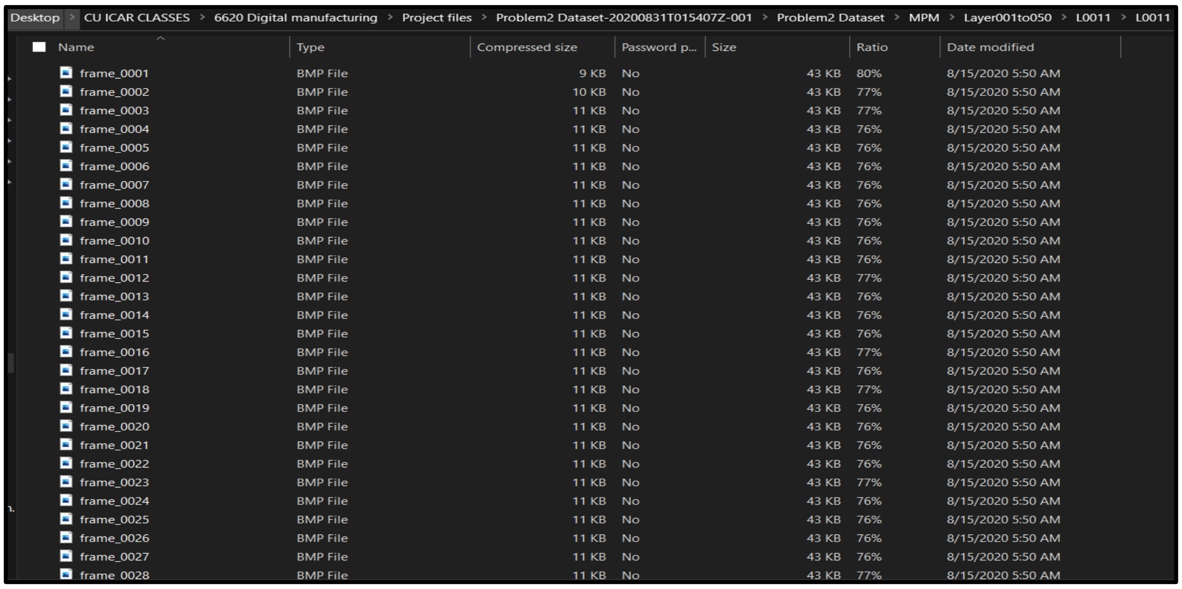
- The next screenshot shows the folders for the image files in dataset 001. Here we can see that there are only images for some of the layers.



Desktop > CU ICAR CLASSES > 6620 Digital manufacturing > Project files > Problem2 Dataset-20200831T015407Z-001 > Problem2 Dataset > MPM > Layer001to050 >

Name	Status	Date modified	Type	Size
L0011	✖	8/15/2020 12:38 AM	Compressed (zipp...	71,541 KB
L0035	✖	8/15/2020 12:58 AM	Compressed (zipp...	64,437 KB
L0037	✖	8/15/2020 12:59 AM	Compressed (zipp...	63,476 KB
L0038	✖	8/15/2020 1:00 AM	Compressed (zipp...	59,980 KB
L0039	✖	8/15/2020 1:01 AM	Compressed (zipp...	63,049 KB
L0040	✖	8/15/2020 1:02 AM	Compressed (zipp...	59,690 KB
L0041	✖	8/15/2020 1:02 AM	Compressed (zipp...	62,680 KB
L0042	✖	8/15/2020 1:03 AM	Compressed (zipp...	59,060 KB
L0043	✖	8/15/2020 1:04 AM	Compressed (zipp...	62,190 KB
L0044	✖	8/15/2020 1:04 AM	Compressed (zipp...	58,345 KB
L0045	✖	8/15/2020 1:05 AM	Compressed (zipp...	61,228 KB
L0046	✖	8/15/2020 1:06 AM	Compressed (zipp...	57,942 KB
L0047	✖	8/15/2020 1:07 AM	Compressed (zipp...	60,606 KB
L0048	✖	8/15/2020 1:07 AM	Compressed (zipp...	57,681 KB
L0049	✖	8/15/2020 1:08 AM	Compressed (zipp...	60,458 KB
L0050	✖	8/15/2020 1:09 AM	Compressed (zipp...	57,216 KB

Figure 5 Folder with images in dataset



Desktop > CU ICAR CLASSES > 6620 Digital manufacturing > Project files > Problem2 Dataset-20200831T015407Z-001 > Problem2 Dataset > MPM > Layer001to050 > L0011 > L0011

Name	Type	Compressed size	Password p...	Size	Ratio	Date modified
frame_0001	BMP File	9 KB	No	43 KB	80%	8/15/2020 5:50 AM
frame_0002	BMP File	10 KB	No	43 KB	77%	8/15/2020 5:50 AM
frame_0003	BMP File	11 KB	No	43 KB	77%	8/15/2020 5:50 AM
frame_0004	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0005	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0006	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0007	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0008	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0009	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0010	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0011	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0012	BMP File	11 KB	No	43 KB	77%	8/15/2020 5:50 AM
frame_0013	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0014	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0015	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0016	BMP File	11 KB	No	43 KB	77%	8/15/2020 5:50 AM
frame_0017	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0018	BMP File	11 KB	No	43 KB	77%	8/15/2020 5:50 AM
frame_0019	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0020	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0021	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0022	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0023	BMP File	11 KB	No	43 KB	77%	8/15/2020 5:50 AM
frame_0024	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0025	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0026	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0027	BMP File	11 KB	No	43 KB	76%	8/15/2020 5:50 AM
frame_0028	BMP File	11 KB	No	43 KB	77%	8/15/2020 5:50 AM

Figure 6 Inside the image folder

- The inside of the folder for Layer 11 (L0011) is shown next. Here we can see the image files are numbered in order which they were taken. To complete task 2, we will have to program our code to assign each frame from the respective layers to the corresponding coordinate from the build command file for that layer. We will also assign a time value to the image in case the layer happens to go over the same coordinate point multiple times.

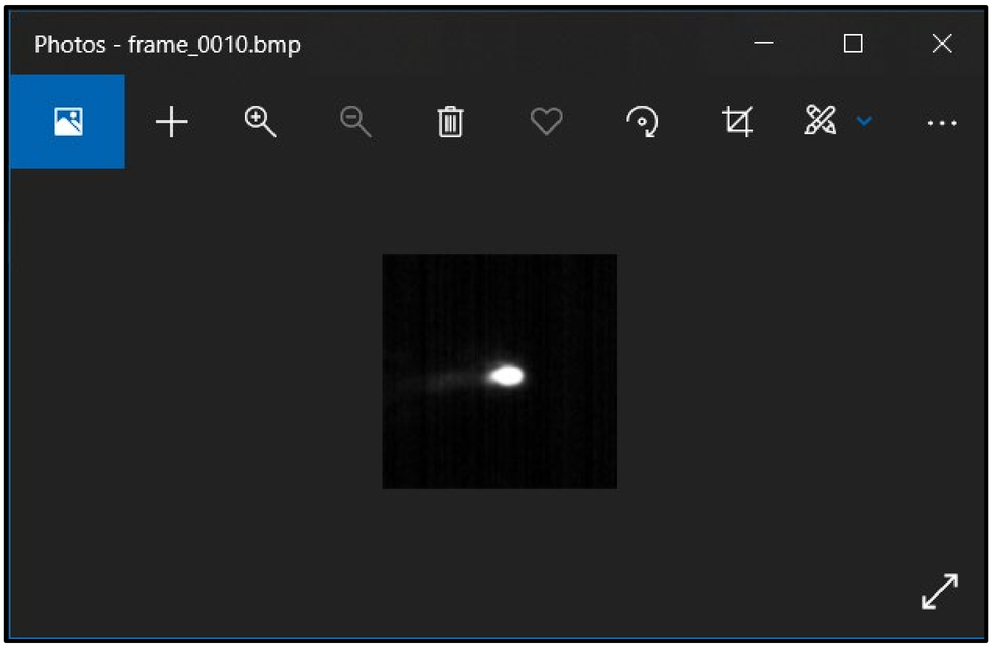
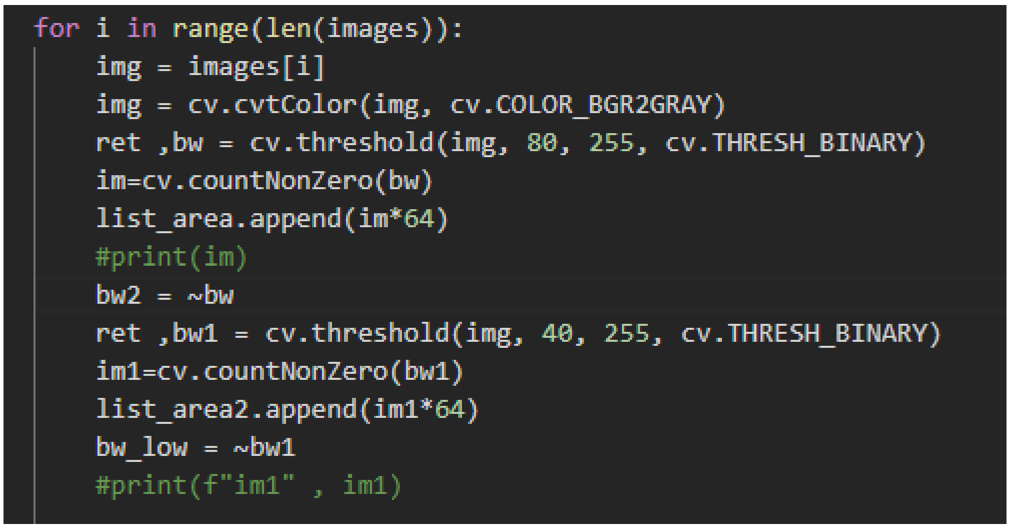


Figure 7 Example image

- Finally, we have a screenshot example of one of the image files for frame_0010 from Layer 11 (L0011) in dataset 001. The images are all 120x120 pixels, and each pixel represents an area of 64 µm^2.

We had broken down the approach into two subtasks. One was the subtasks was to analyze the images for melt-pool area, splatter identification, and tail length calculation. For the completion of this task, we relied on OpenCV library in Python. In the dataset instructions we had been given the area corresponding to each pixel in an image. Hence, we first converted the images from three channel RGB images to grayscale images which were then converted into binary images with threshold of 80. Values higher than 80 gave us melt-pool area bigger than actual area, where lower values had the opposite effect. Binary images have only one channel and each pixel has values either 0 or 1, we calculated the number of pixels with non-zero values which gave us the pixel count.

Each pixel has an area of 64 micro-meters. That is how we calculated the melt-pool area. The code snippet below how the image was converted from original form to binary, and melt-pool area calculation.



```
for i in range(len(images)):
    img = images[i]
    img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    ret ,bw = cv.threshold(img, 80, 255, cv.THRESH_BINARY)
    im=cv.countNonZero(bw)
    list_area.append(im*64)
    #print(im)
    bw2 = ~bw
    ret ,bw1 = cv.threshold(img, 40, 255, cv.THRESH_BINARY)
    im1=cv.countNonZero(bw1)
    list_area2.append(im1*64)
    bw_low = ~bw1
    #print(f"im1" , im1)
```

Figure 8 Code snippet – Binary conversion

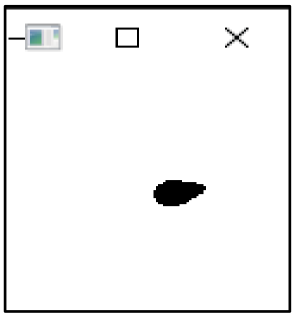
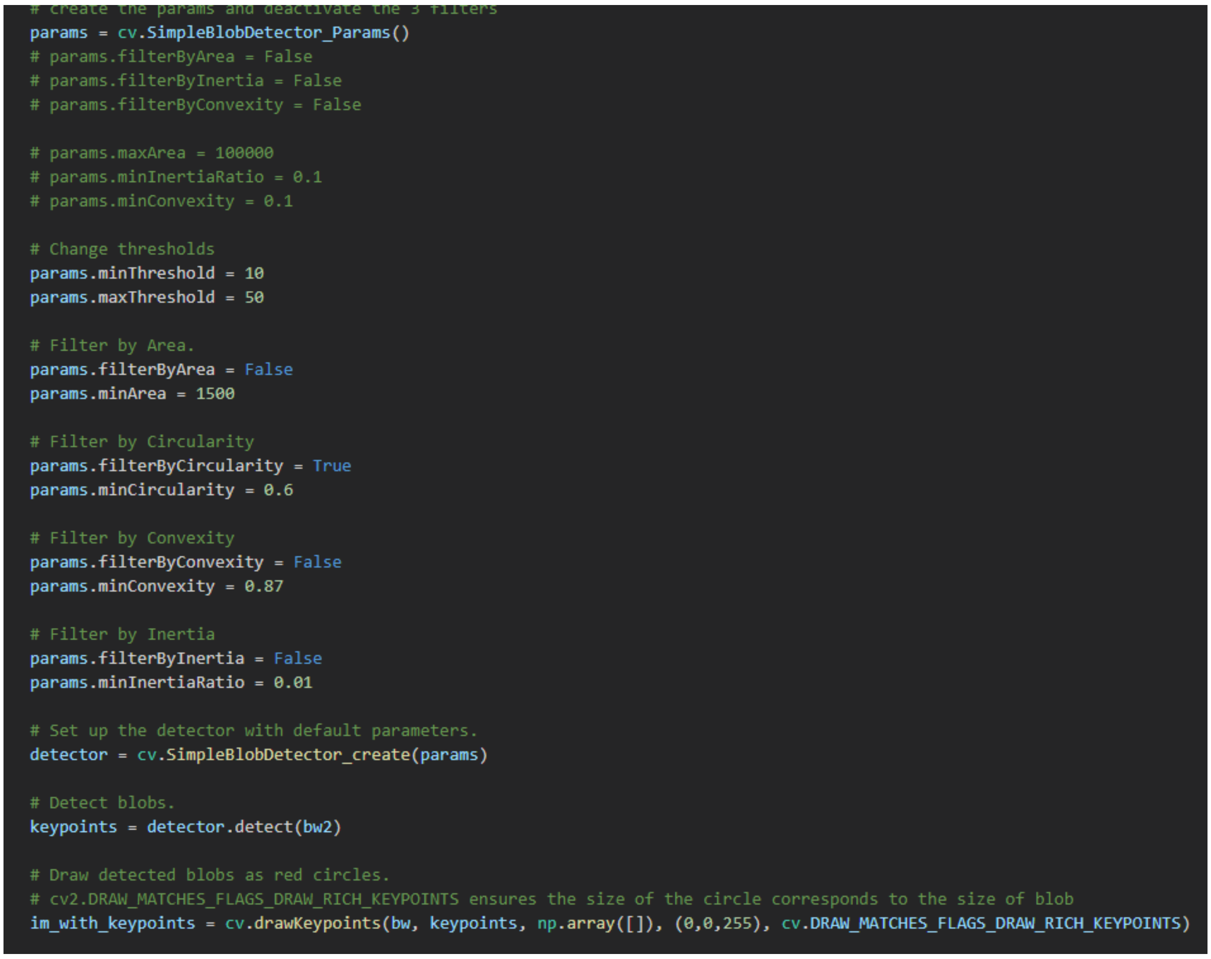


Figure 9 - Inverted Binary Image

For the splatter identification, we decided to use SimpleBlobDetector, another OpenCV tool. While working with this tool, we noticed that it works well with the dark or gray blobs rather than white blobs. Therefore, we had to convert the binary images to inverse binary images. Once the image was inverted then we applied SimpleBlobDetector with certain parameters. The following code snipped shows how it was achieved.



```python
# create the params and deactivate the 3 filters
params = cv.SimpleBlobDetector_Params()
# params.filterByArea = False
# params.filterByInertia = False
# params.filterByConvexity = False

# params.maxArea = 100000
# params.minInertiaRatio = 0.1
# params.minConvexity = 0.1

# Change thresholds
params.minThreshold = 10
params.maxThreshold = 50

# Filter by Area.
params.filterByArea = False
params.minArea = 1500

# Filter by Circularity
params.filterByCircularity = True
params.minCircularity = 0.6

# Filter by Convexity
params.filterByConvexity = False
params.minConvexity = 0.87

# Filter by Inertia
params.filterByInertia = False
params.minInertiaRatio = 0.01

# Set up the detector with default parameters.
detector = cv.SimpleBlobDetector_create(params)

# Detect blobs.
keypoints = detector.detect(bw2)

# Draw detected blobs as red circles.
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS ensures the size of the circle corresponds to the size of blob
im_with_keypoints = cv.drawKeypoints(bw, keypoints, np.array([]), (0,0,255), cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

Figure 10 – Code snippet SimpleBlobDetector

We decided to filter out the blob with only the circularity parameter. The result is shown in the image below. By changing the threshold values, we were able to detect other blobs in the image that were separated from the melt pool. We identified these as splatter.

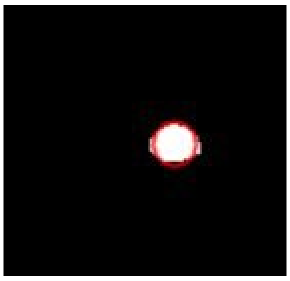


Figure 11 - Binary Image with blob

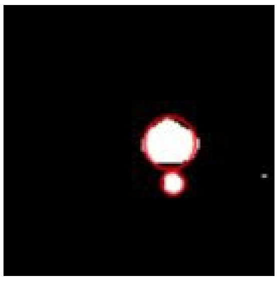
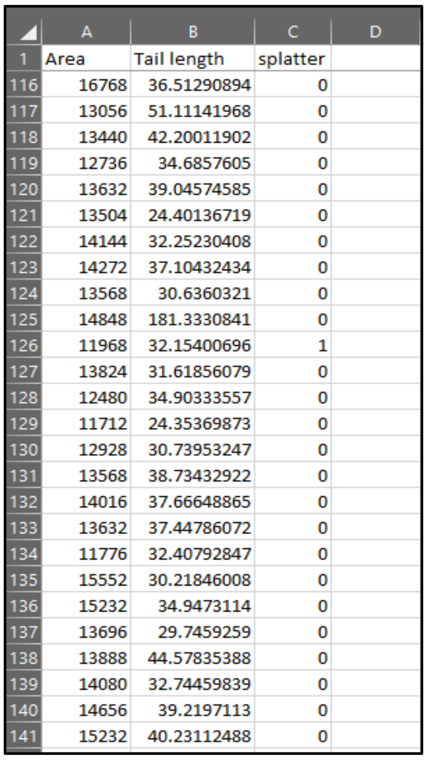


Figure 12 -Blob detection with different threshold

For the task of tail-length calculation, we change the circularity parameter along with the threshold parameters. We then subtracted the diameter of the small blob from the bigger blob to obtain the tail-length value.

Once the image analysis was done, we then created a DataFrame with the melt-pool area, tail length, and splatter in Boolean format. Since Dataframe is computationally heavy, we then converted it into a CSV file. The following image shows a snippet of the CSV file.

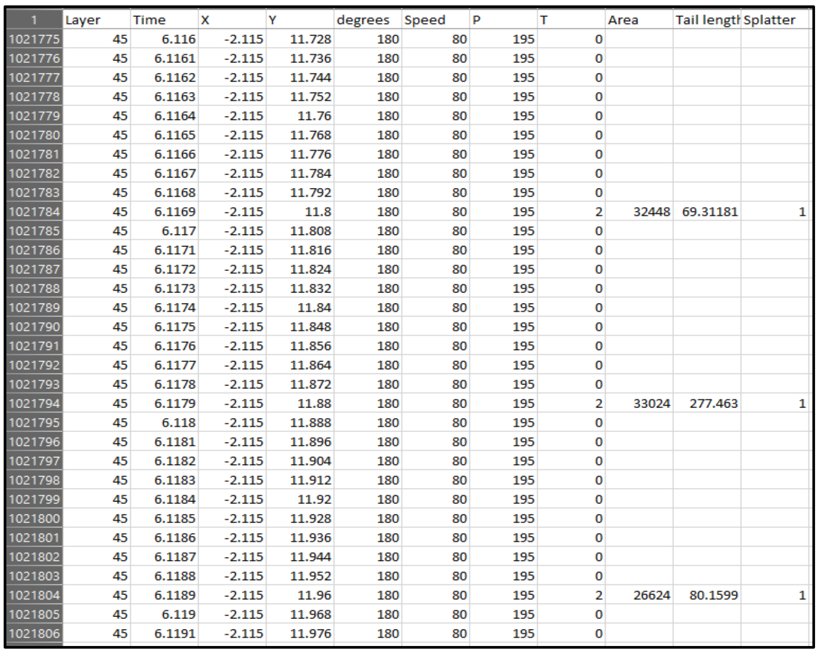


	A	B	C	D
1	Area	Tail length	splatter	
116	16768	36.51290894	0	
117	13056	51.11141968	0	
118	13440	42.20011902	0	
119	12736	34.6857605	0	
120	13632	39.04574585	0	
121	13504	24.40136719	0	
122	14144	32.25230408	0	
123	14272	37.10432434	0	
124	13568	30.6360321	0	
125	14848	181.3330841	0	
126	11968	32.15400696	1	
127	13824	31.61856079	0	
128	12480	34.90333557	0	
129	11712	24.35369873	0	
130	12928	30.73953247	0	
131	13568	38.73432922	0	
132	14016	37.66648865	0	
133	13632	37.44786072	0	
134	11776	32.40792847	0	
135	15552	30.21846008	0	
136	15232	34.9473114	0	
137	13696	29.7459259	0	
138	13888	44.57835388	0	
139	14080	32.74459839	0	
140	14656	39.2197113	0	
141	15232	40.23112488	0	

Figure 13 Dataframe to CSV of image analysis

With these three tasks, we concluded the image analysis part of the solution. During the image analysis phase, we were also working on the spatial alignment of the data in CSV file. For this subtask, we decided to put all the data available, including data obtained through image analysis in the Data frame with the Pandas library.

In this phase, we fused the CSV files with XYPT values, image analysis CSV and extracted values such as Speed, degrees, and time. The following image shows a section of fused CSV files or spatially aligned data. Since the images are captured at 10kHz and build commands are sampled at 100 kHz, melt-pool area, tail length, and the splatter are calculated every 10th instance.

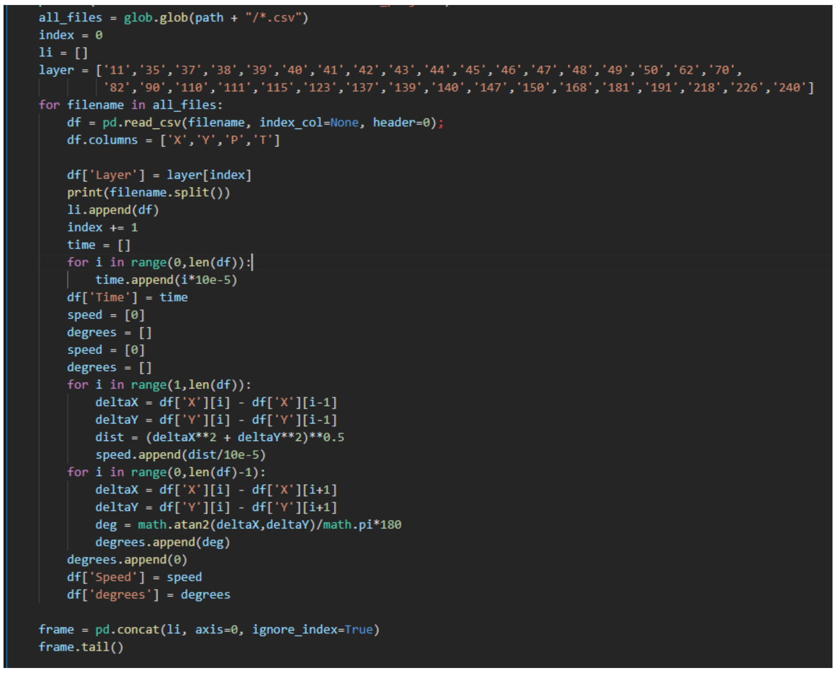


1	Layer	Time	X	Y	degrees	Speed	P	T	Area	Tail length	Splatter
1021775	45	6.116	-2.115	11.728	180	80	195	0			
1021776	45	6.1161	-2.115	11.736	180	80	195	0			
1021777	45	6.1162	-2.115	11.744	180	80	195	0			
1021778	45	6.1163	-2.115	11.752	180	80	195	0			
1021779	45	6.1164	-2.115	11.76	180	80	195	0			
1021780	45	6.1165	-2.115	11.768	180	80	195	0			
1021781	45	6.1166	-2.115	11.776	180	80	195	0			
1021782	45	6.1167	-2.115	11.784	180	80	195	0			
1021783	45	6.1168	-2.115	11.792	180	80	195	0			
1021784	45	6.1169	-2.115	11.8	180	80	195	2	32448	69.31181	1
1021785	45	6.117	-2.115	11.808	180	80	195	0			
1021786	45	6.1171	-2.115	11.816	180	80	195	0			
1021787	45	6.1172	-2.115	11.824	180	80	195	0			
1021788	45	6.1173	-2.115	11.832	180	80	195	0			
1021789	45	6.1174	-2.115	11.84	180	80	195	0			
1021790	45	6.1175	-2.115	11.848	180	80	195	0			
1021791	45	6.1176	-2.115	11.856	180	80	195	0			
1021792	45	6.1177	-2.115	11.864	180	80	195	0			
1021793	45	6.1178	-2.115	11.872	180	80	195	0			
1021794	45	6.1179	-2.115	11.88	180	80	195	2	33024	277.463	1
1021795	45	6.118	-2.115	11.888	180	80	195	0			
1021796	45	6.1181	-2.115	11.896	180	80	195	0			
1021797	45	6.1182	-2.115	11.904	180	80	195	0			
1021798	45	6.1183	-2.115	11.912	180	80	195	0			
1021799	45	6.1184	-2.115	11.92	180	80	195	0			
1021800	45	6.1185	-2.115	11.928	180	80	195	0			
1021801	45	6.1186	-2.115	11.936	180	80	195	0			
1021802	45	6.1187	-2.115	11.944	180	80	195	0			
1021803	45	6.1188	-2.115	11.952	180	80	195	0			
1021804	45	6.1189	-2.115	11.96	180	80	195	2	26624	80.1599	1
1021805	45	6.119	-2.115	11.968	180	80	195	0			
1021806	45	6.1191	-2.115	11.976	180	80	195	0			

Figure 14 CSV file with feature extraction

Our approach to creating a Dataframe with spatially aligned data is described below. The code snippets below show exactly how we managed to extract the Speed, time, and degrees from the data given. We used the distance formula to calculate the distance traveled by the laser between two data points. Then we divided the calculated distance by the time difference between the said data points. Similarly, we used the tan inverse of the Y coordinate divided by the X coordinate to get the degrees/ direction of the laser.

The second snippet of the code below shows how we added the area, tail length, and splatter calculated during the image analysis phase of the project.

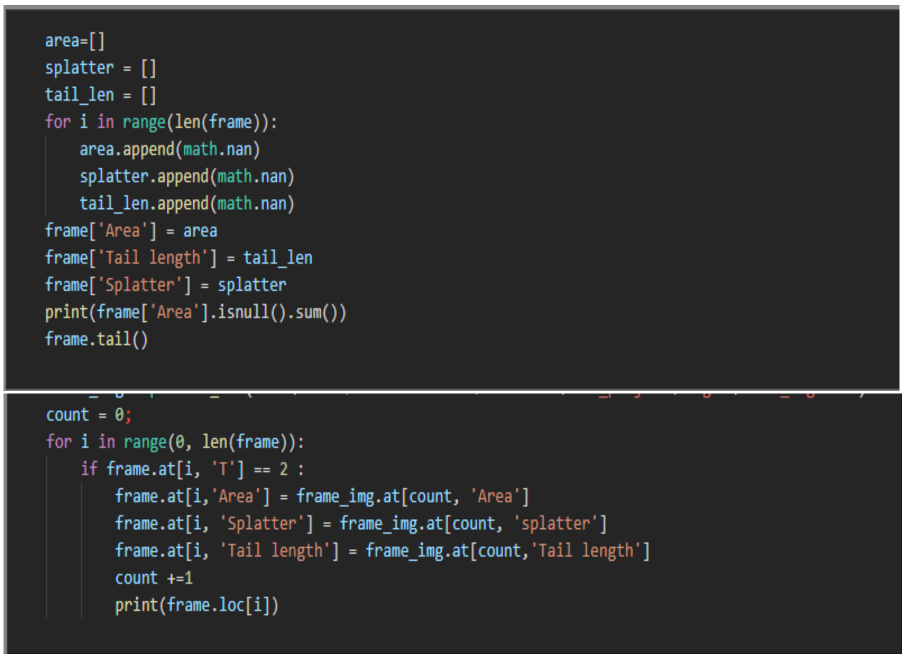


```
all_files = glob.glob(path + "/*.csv")
index = 0
li = []
layer = ['11','35','37','38','39','40','41','42','43','44','45','46','47','48','49','50','62','70',
         '82','90','110','111','115','123','137','139','140','147','150','168','181','191','218','226','240']
for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0);
    df.columns = ['X','Y','P','T']

    df['Layer'] = layer[index]
    print(filename.split())
    li.append(df)
    index += 1
    time = []
    for i in range(0,len(df)):
        time.append(i*10e-5)
    df['Time'] = time
    speed = [0]
    degrees = []
    speed = [0]
    degrees = []
    for i in range(1,len(df)):
        deltaX = df['X'][i] - df['X'][i-1]
        deltaY = df['Y'][i] - df['Y'][i-1]
        dist = (deltaX**2 + deltaY**2)**0.5
        speed.append(dist/10e-5)
    for i in range(0,len(df)-1):
        deltaX = df['X'][i] - df['X'][i+1]
        deltaY = df['Y'][i] - df['Y'][i+1]
        deg = math.atan2(deltaX,deltaY)/math.pi*180
        degrees.append(deg)
    degrees.append(0)
    df['Speed'] = speed
    df['degrees'] = degrees

frame = pd.concat(li, axis=0, ignore_index=True)
frame.tail()
```

Figure 15 Code Snippet – Calculating Speed, direction and time from the given data



```
area=[]
splatter = []
tail_len = []
for i in range(len(frame)):
    area.append(math.nan)
    splatter.append(math.nan)
    tail_len.append(math.nan)
frame['Area'] = area
frame['Tail length'] = tail_len
frame['Splatter'] = splatter
print(frame['Area'].isnull().sum())
frame.tail()
```

```
count = 0;
for i in range(0, len(frame)):
    if frame.at[i, 'T'] == 2 :
        frame.at[i,'Area'] = frame_img.at[count, 'Area']
        frame.at[i, 'Splatter'] = frame_img.at[count, 'splatter']
        frame.at[i, 'Tail length'] = frame_img.at[count,'Tail length']
        count +=1
        print(frame.loc[i])
```

Figure 16 Code snippet- Addition of image analysis parameters to the original features

Results

In this section, we talk about if we met the entire scope of the project, if we achieved a solution to the problem statement, and if we were able to meet our goals.

In the project's scope, we had mentioned that we would be doing the spatial alignment of the data along with feature extraction. Moreover, one of the aspects was determining the empirical relationships between the build commands and the extracted features. And lastly, we also had planned to develop a real-time control strategy for the AM process. Apart from developing the control strategy, we met every other aspect of our scope.

Our problem statement was to predict the melt-pool area of powder-based fusion additive manufacturing. In this project, we have written a shortcode for image analysis that calculates the image's melt-poll area, tail length, and splatter. Therefore, the answer to the question if we have achieved the solution to your problem statement is positive. We have successfully found a solution to the given problem. Instead of a dataset, if we were to feed this live algorithm data with some modifications, it would give out the melt-pool areal for the process.

At the beginning of the project, we had set a list of goals that we wanted to meet. Although we had mentioned the goals at the beginning of this report, we would like to reiterate them here. In our case, the scope of the project aligned with the scope of our project perfectly. One of our goals was to predict the melt-pool size based on the build commands and to create an accurate and improved control strategy. This goal of the project was unfinished. One of the reasons why this goal was unfinished is that we lacked the technical expertise to interpret the available data in such a way that a valid working control strategy could be devised.

We successfully managed to spatially align the data we had and perform image analysis. Through the spatial alignment of data, we were able to extract features of the dataset as well. Furthermore, we were able to identify a relationship between the build commands and melt-pool size.

Most of the tasks went well with our project. However, we underestimated the amount of data that we had to wade through. This set us back a bit during the initial phase of the project. It is also worth mentioning that we began the image analysis a bit later than we had planned. The reason for the delay could be attributed to the fact that we had to spend some time getting to know the available tools and how to use them.

Discussion / Future Work

In this section of the report, we would like to discuss the future scope of our project along with a few changes in approach if we had to do the project over. Moreover, we also talked about what we would improve if we had more time to complete the project.

We were a bit confused when it came to the image analysis part. All of us had an idea of what to do but little knowledge of how to do it. If we had to do this project all over again, we would start by going through the dataset thoroughly at the early stages of the project. At the same time, we would start looking at various tools and techniques available for data and image analysis. This time we only explored the tools/techniques that we were aware of.

We would like to mention that there were a few more goals that we wanted to achieve, such as creating a robust control strategy for the process by establishing an empirical relationship between the various parameters. However, we had to cut these goals short in lieu of time. If given more time, we would certainly improve upon the feature extraction from the given dataset to such a degree that it would allow us to create a robust control strategy.

We would like to give next year's students a few suggestions from our learnings. One of the main suggestions would be to go through the dataset as early as possible. This would give you more time to experiment in the latter stages of the project. Another suggestion would be to utilize the Palmetto cluster for faster computation of the results. There are more than 3.1 million data points, and it is time-consuming to process them without proper computational power. For the image analysis, there are other tools than OpenCV. Try and explore them too.

As we conclude this discussion section, we would like to comment on how next year's class semester project could be improved. It would be better if we covered the data analysis/feature extraction lecture a bit earlier in the semester. This would also give students more time to deal with problems with large datasets.