Industrial Component Classification and Damage Marking.

Part 1: Impeller for Submersible Pumps

- Binary classification of the component.
- Keras Sequential Model
- Optimizer: Adam
- Performance Metrics: Precision, Recall, f1-score

Part 2: Steel Slab Damage Marking

- Manufacturing-damage marking of Multiclass classified instances.
- OpenCV
- Run-length Encoding

Data Information

Data from Severstal Steel Company.

The Component #1 data was a well cleaned out (but unbalanced) data while the Component #2 data was unbalance but rich and multiclassed.

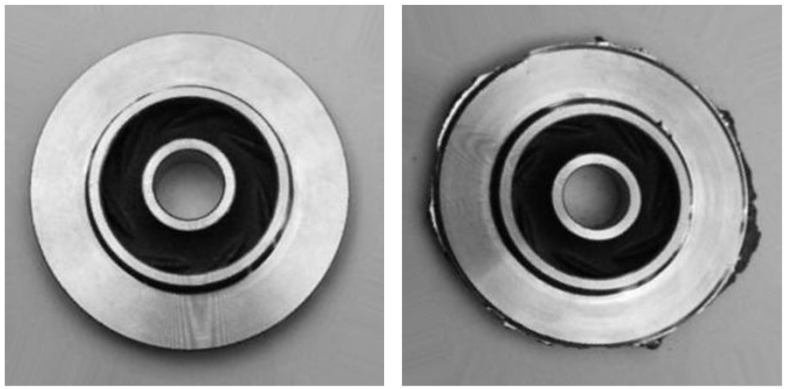
Impeller for Submersible Pumps

- A submersible water pump has the same function as a standard water pump of draining water.

- However, it comes with an added advantage of the ability to be placed underwater and still function properly.

- Often used for drainage in floods, emptying ponds or even as pond filters.

Impeller for Submersible Pumps



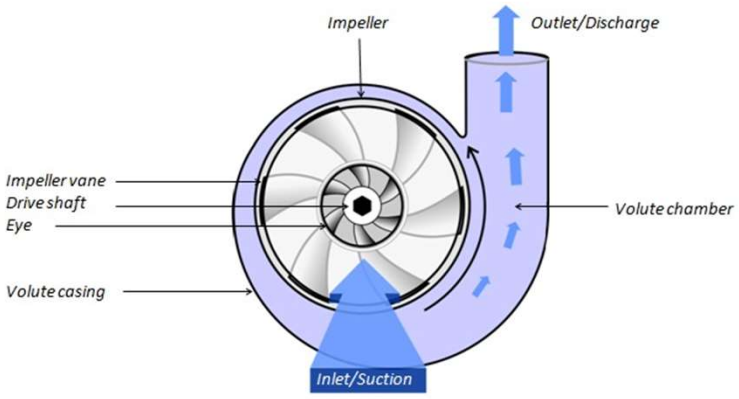


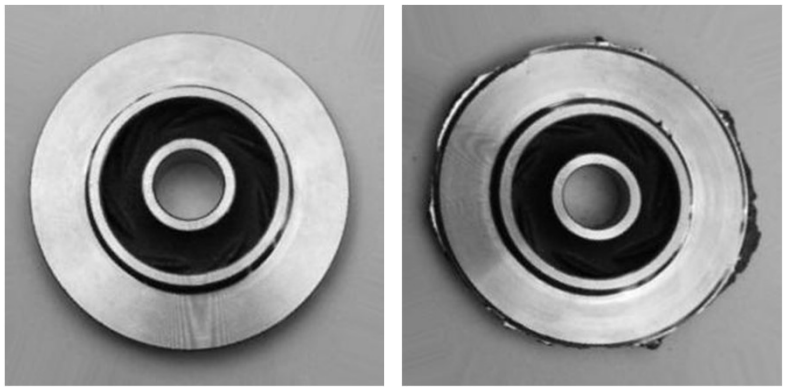
Figure 2. Volute case design

- The action of the impeller increases the fluid's velocity and pressure and directs it towards the pump outlet.

- Different types of Impellers: open impeller, semi-open impeller, closed impeller, vortex impeller, cutter impeller.

- In this research, the closed Impeller is studied.

Training and Testing



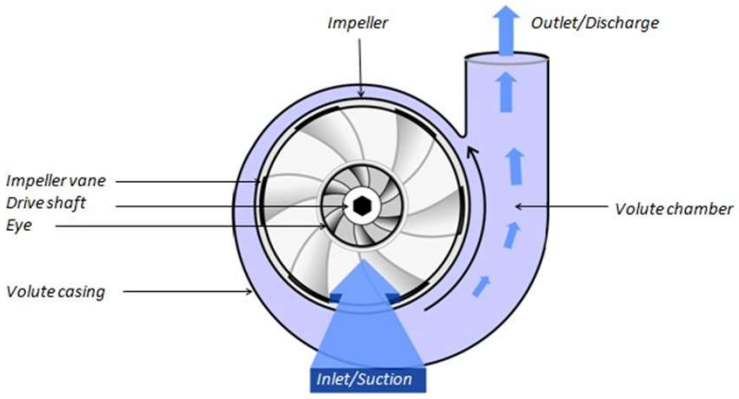


Figure 2. Volute case design

Training: 3,758 defective samples

2,875 ok samples

Total training: 6,633

Testing: 453 defective samples

262 ok samples

Total testing: 715

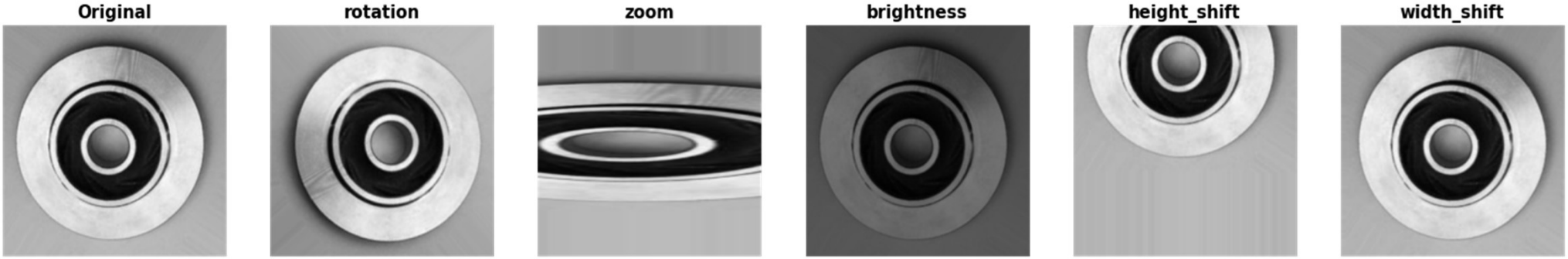
Total: 7,348 samples

Keras Sequential Model

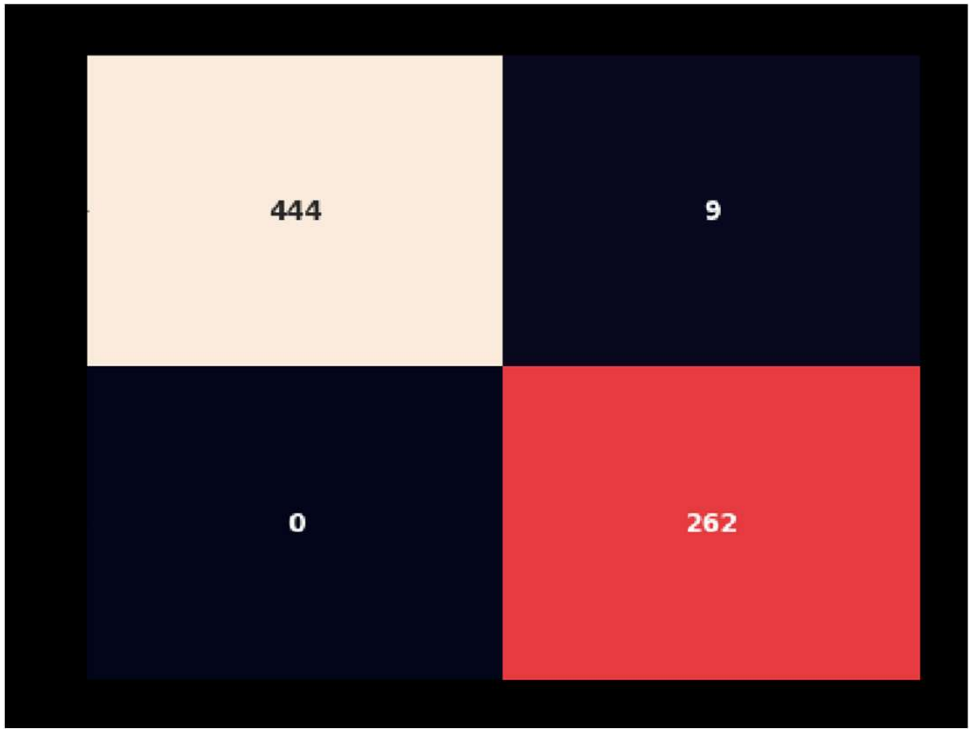
Image Data Augmentation: Different variations of same component for future proofing.

Good practice to expand the size of a training dataset by creating modified versions of images in the dataset.

The model is trained more efficiently for the future occurrences of such modified or distorted samples.



Metrics: Confusion Matrix



715 test cases.

706 correctly classified.

9 incorrectly classified.

(with n_epochs = 12)

Training Time: 1.7 hours

Prediction for Random OK component

Prediction for Random DEFECTIVE component

Metrics: Precision, Recall and f1-score

```
715 test cases.

453 DEF samples

262 OK samples
```

Confusion matrix and accuracy is not a good metric when the dataset is unbalanced.

The dataset in focus is unbalanced; comprises of more DEF samples.

Metrics: Precision, Recall and f1-score

Precision = $$\frac{(True\ Positive)}{(True\ Positive + False\ Positive)}$$

Recall = $$\frac{(True\ Positive)}{(True\ Positive + False\ Negative)}$$

f1 Score = $$2 \cdot \left(\frac{(Precision \cdot Recall)}{(Precision + Recall)} \right)$$

Metrics: Precision, Recall and f1-score for DEF samples

precision	recall	f1-score	support
0.995	0.817	0.897	453

Precision = 0.995

Recall = 0.817

f1 - score = $2 \cdot \frac{(0.995 \cdot 0.817)}{(0.995 + 0.817)}$ = 0.89725

Metrics: Precision, Recall and f1-score for OK samples

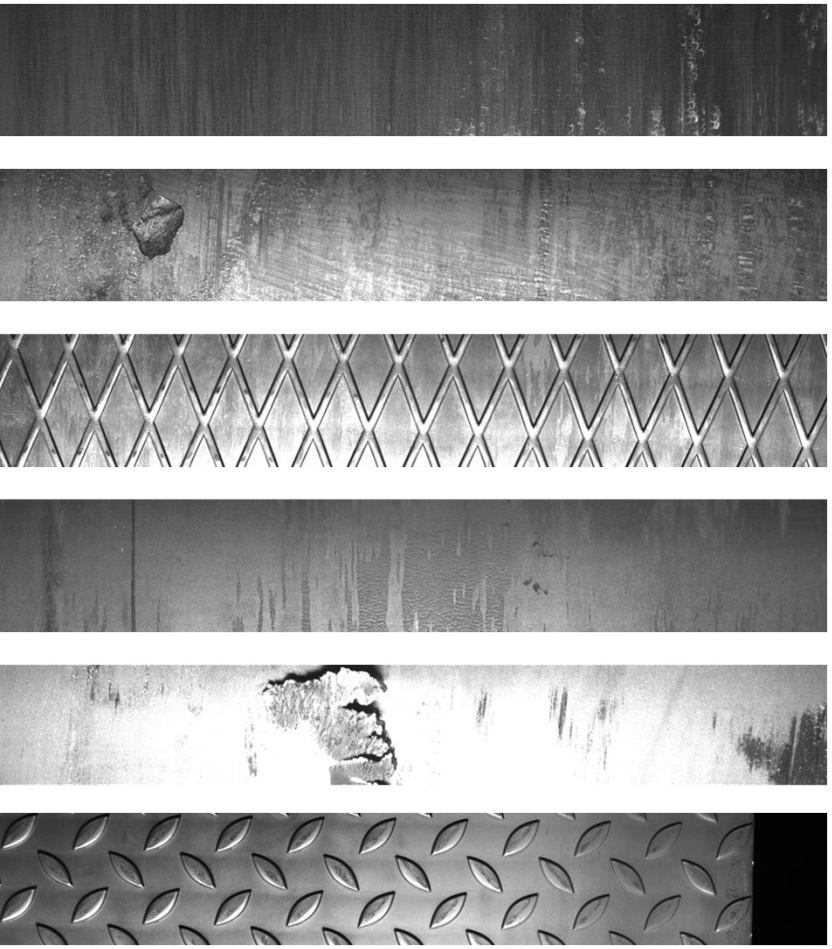
precision	recall	f1-score	support
0.758	0.992	0.860	262

Precision = 0.758

Recall = 0.992

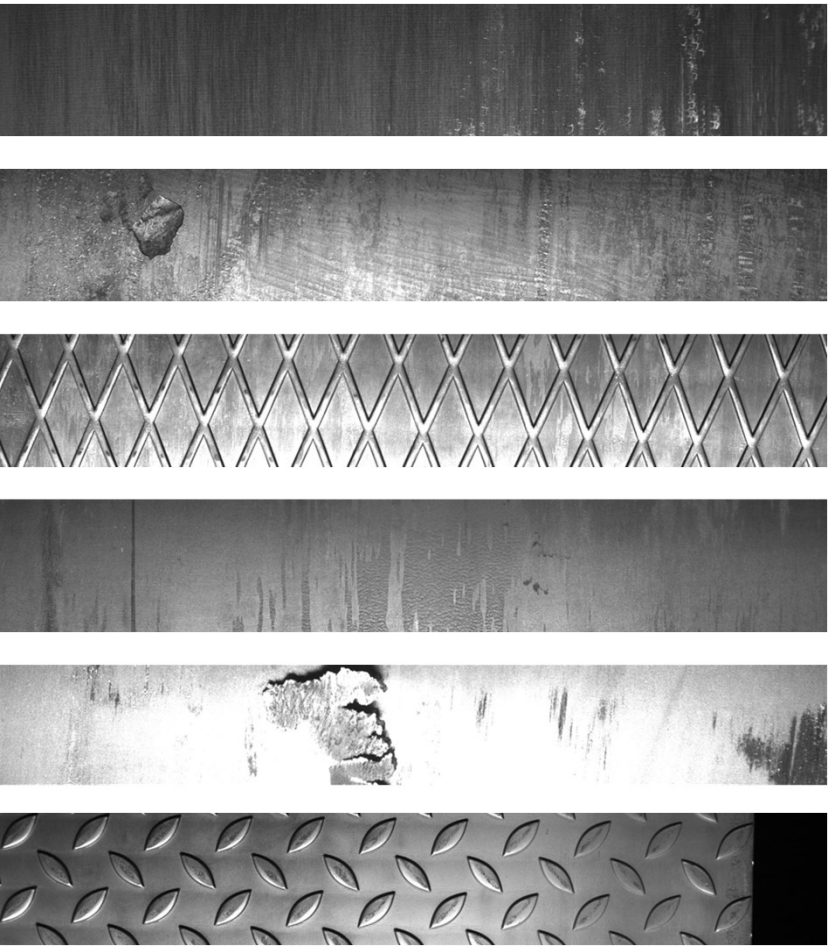
f1 - score = $2 \cdot \frac{(0.758 \cdot 0.992)}{(0.758 + 0.992)}$ = 0.85935

Steel Slab Damage Marking



- Manufacturing-damage marking of Multiclass classified instances.
- OpenCV
- Run-length Encoding

Steel Slab Damage Marking



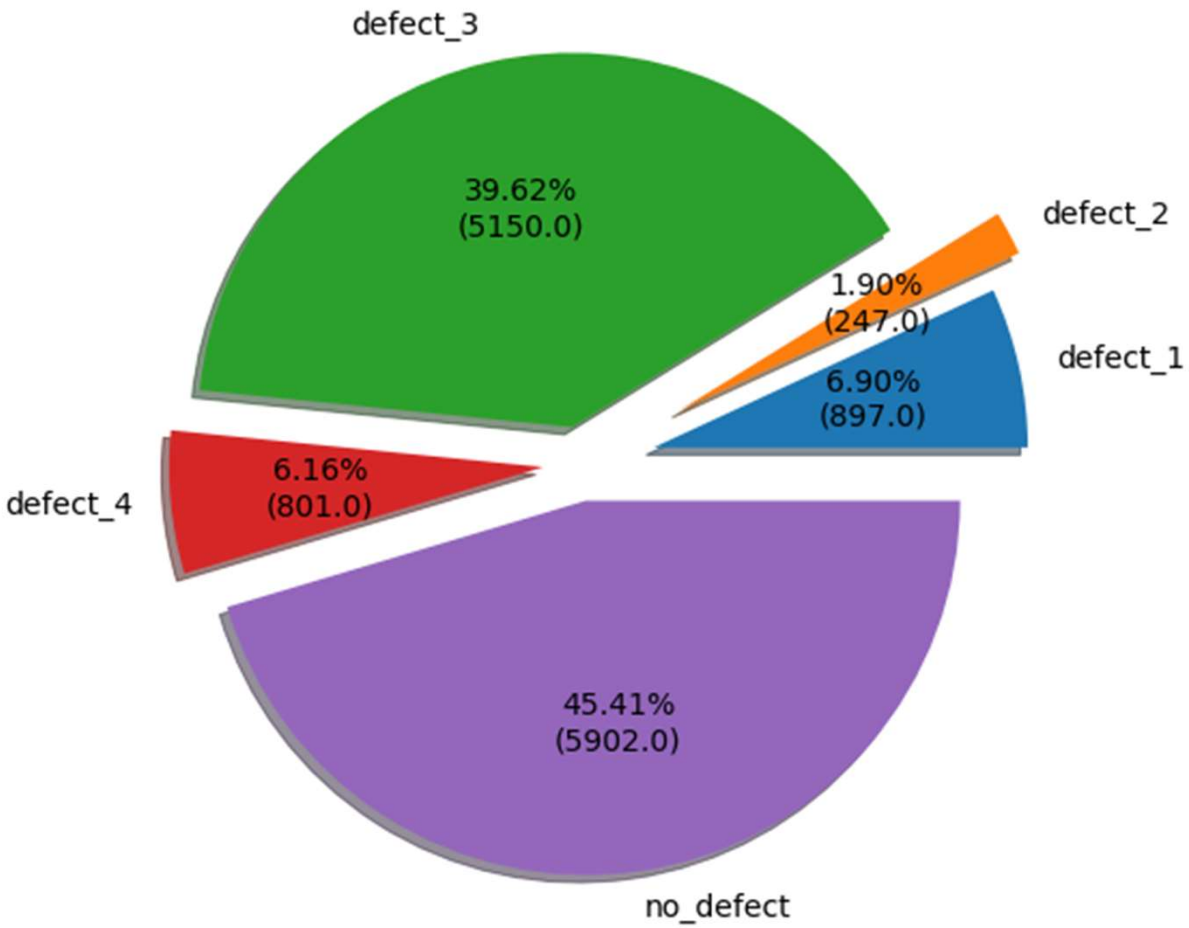
- Total categories: No_defect, defect_1, defect_2, defect_3, defect_4

- Classes in focus: defect_1, defect_2, defect_3, defect_4

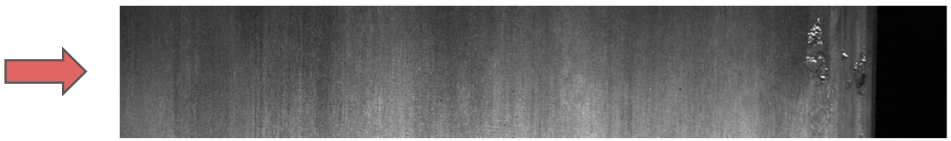
- Total: 18,074 images and corresponding damage

Encoded pixels.

Dataset classification chart.



Defect - Pixel Encoding



RGB image is (256,1600,3) 256*1600 = 409,600 locations

Example image *01661826d.jpg*

```
339049 1 339291 8 339305 3 339541 25 339789 34 340038 43 340290
53 340544 60 340798 63 341052 67 341306 70 341559 74 341811 79
342065 82 342319 85 342574 87 342828 90 343082 92 343337 93
343591 95 343846 95 344100 97 344355 98 344609 100 344864 101
345118 103 345372 104 345626 106
```

First number is starting pixel and next number is count from that starting pixel.

Defect - Pixel Encoding

Example image *01661826d.jpg* 

`339049 1 339291 8 339305 3`

`339049 1` – 1 pixel starting from 339049

`339291 8` – 8 pixels starting from 339291

`339305 3` – 3 pixels starting from 339305

Defect - Pixel Encoding

Example image *01661826d.jpg*

`339049 1` – 1 pixel starting from 339049

`339291 8` – 8 pixels starting from 339291

`339305 3` – 3 pixels starting from 339305

Signifies:

339049	339291	339292	339293	339294	339295	339296	339297	339298	339305	339306	339307

Defective Pixel Encoding

Example image *01661826d.jpg*

```
339049 1 339291 8 339305 3 339541 25 339789 34 340038 43 340290 53
340544 60 340798 63 341052 67 341306 70 341559 74 341811 79 342065
82 342319 85 342574 87 342828 90 343082 92 343337 93 343591 95
343846 95 344100 97 344355 98 344609 100 344864 101 345118 103
345372 104 345626 106
```

Splitting the data frame into 2 dataframes; start_pixel and pixel_count

```
start_pixel = ['339049', '339291', '339305', '339541', '339789']
```

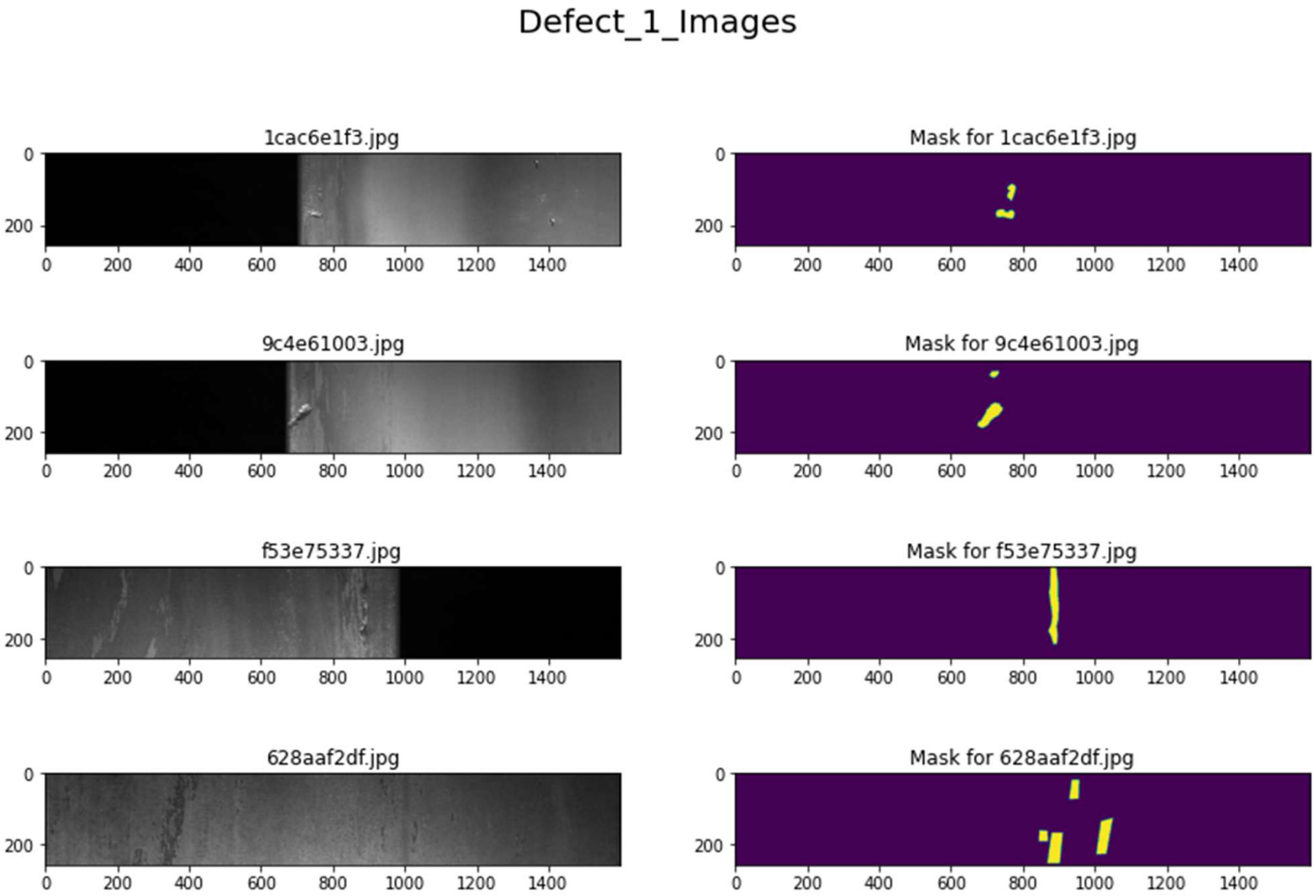
```
pixel_count = [ '1', '8', '3', '25', '34', '43', '53']
```

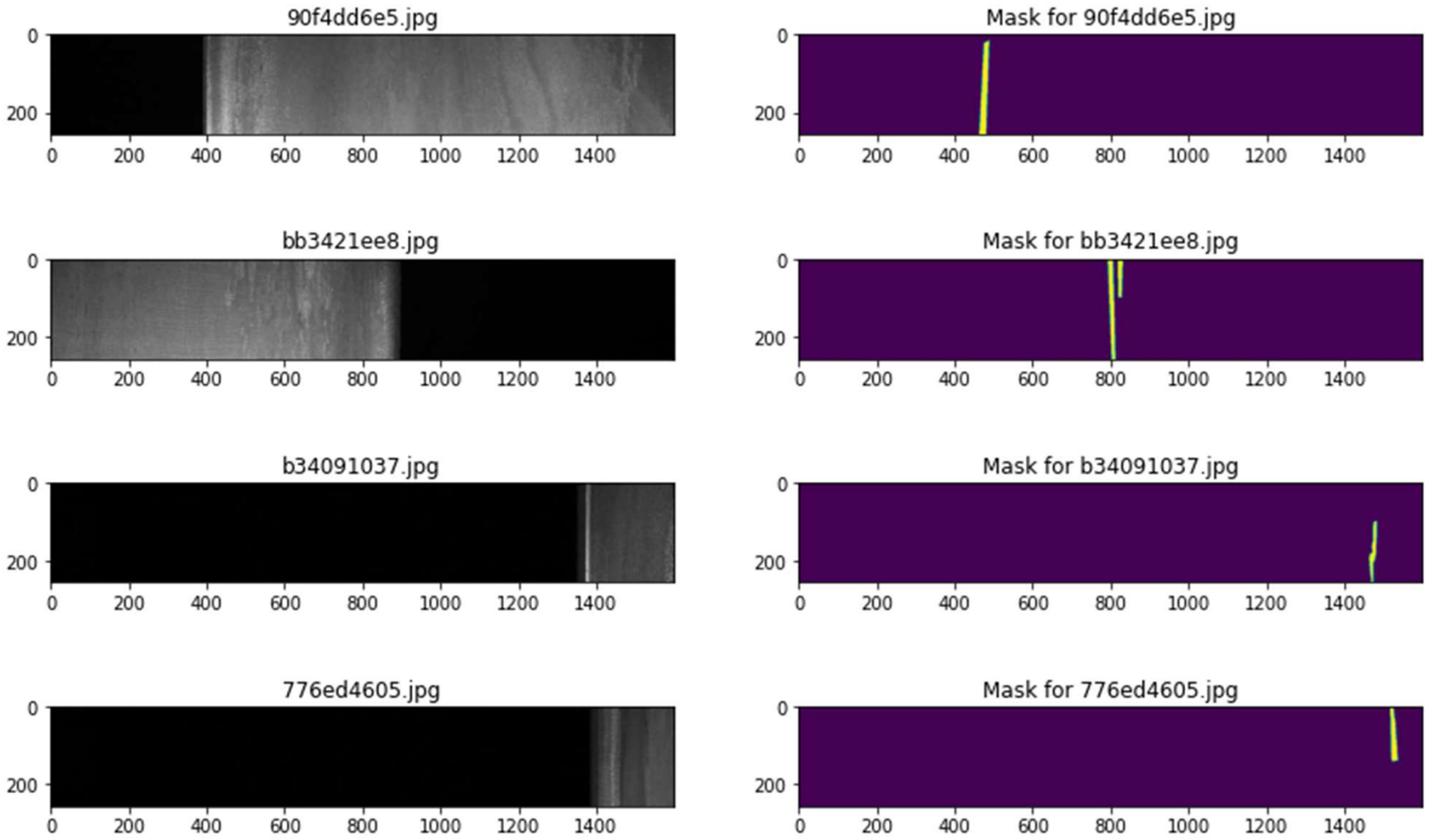
Defect - Pixel Encoding

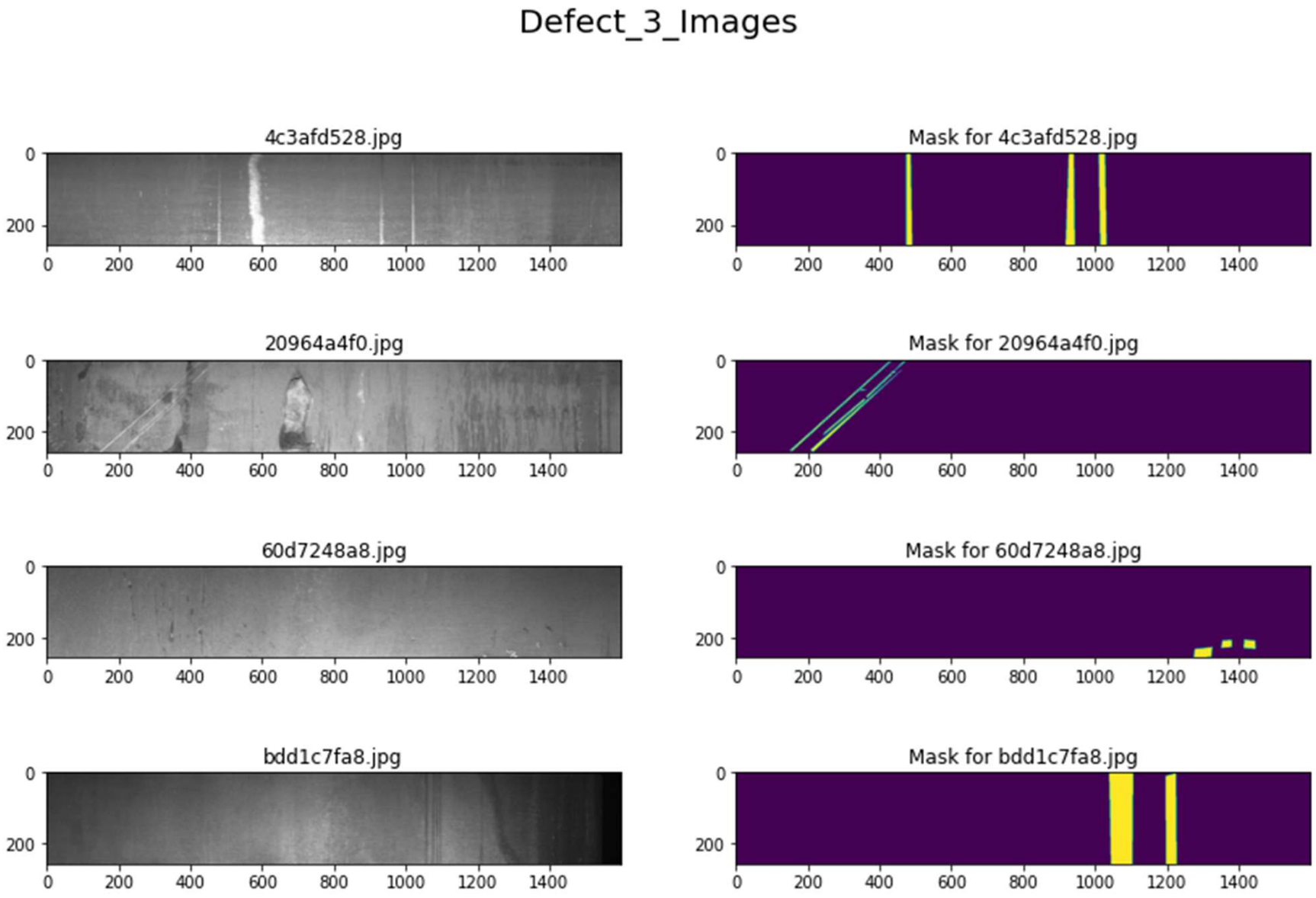
Example image *01661826d.jpg*

```
start_pixel = ['339049', '339291', '339305', '339541', '339789']
pixel_count = [ '1', '8', '3', '25', '34', '43', '53']

range(start_pixel[i],start_pixel[i] + pixel_count[i])


['339049', '339291', '339292', '339293', '339294', '339295', '339296',
'339297', '339298',....., '339841']
```

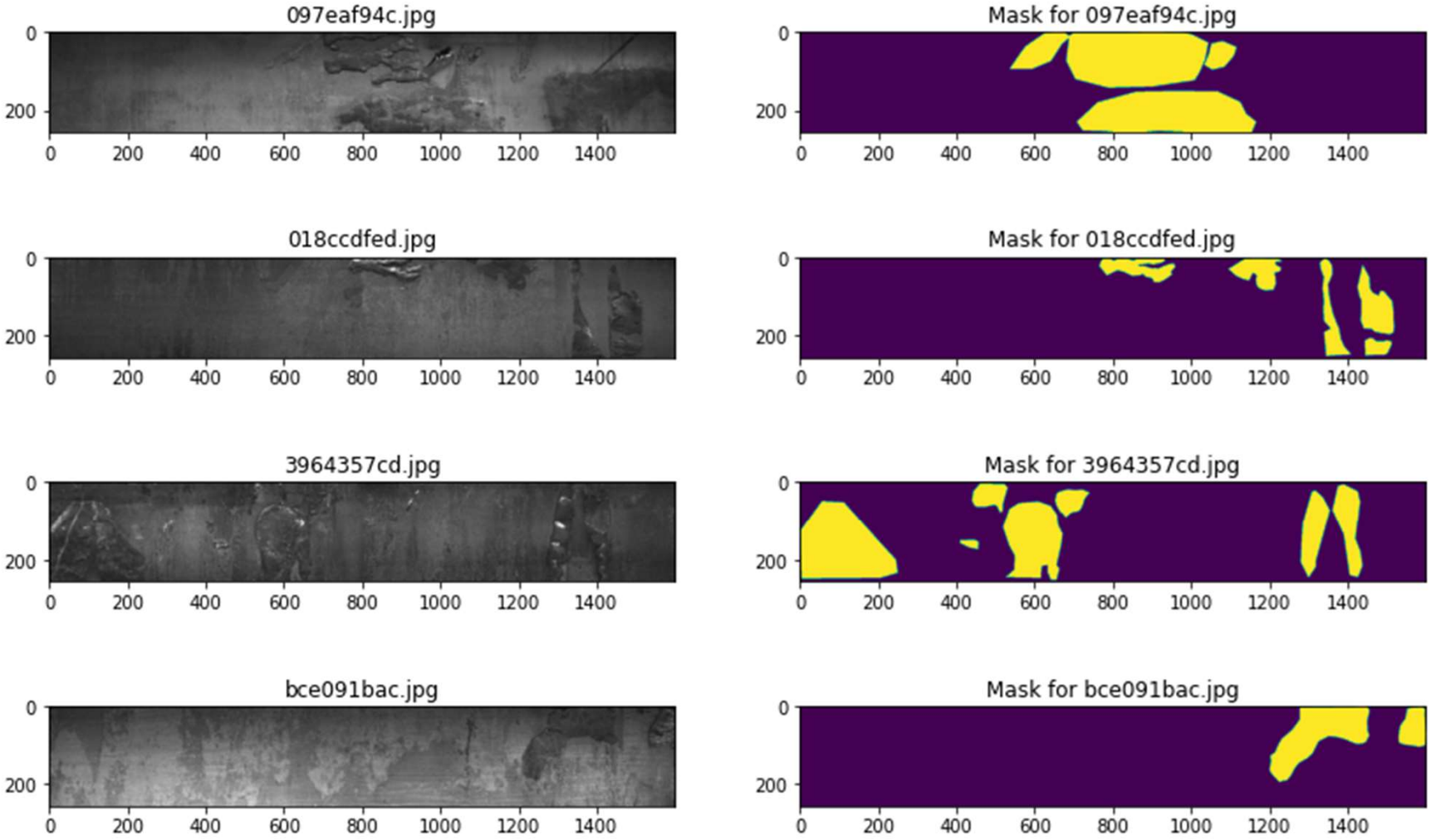
OpenCV for highlighting coordinates.


Defect_1_Images
1cac6e1f3.jpg
Mask for 1cac6e1f3.jpg
9c4e61003.jpg
Mask for 9c4e61003.jpg
f53e75337.jpg
Mask for f53e75337.jpg
628aaf2df.jpg
Mask for 628aaf2df.jpg
0
200
0
200
400
600
800
1000
1200
1400

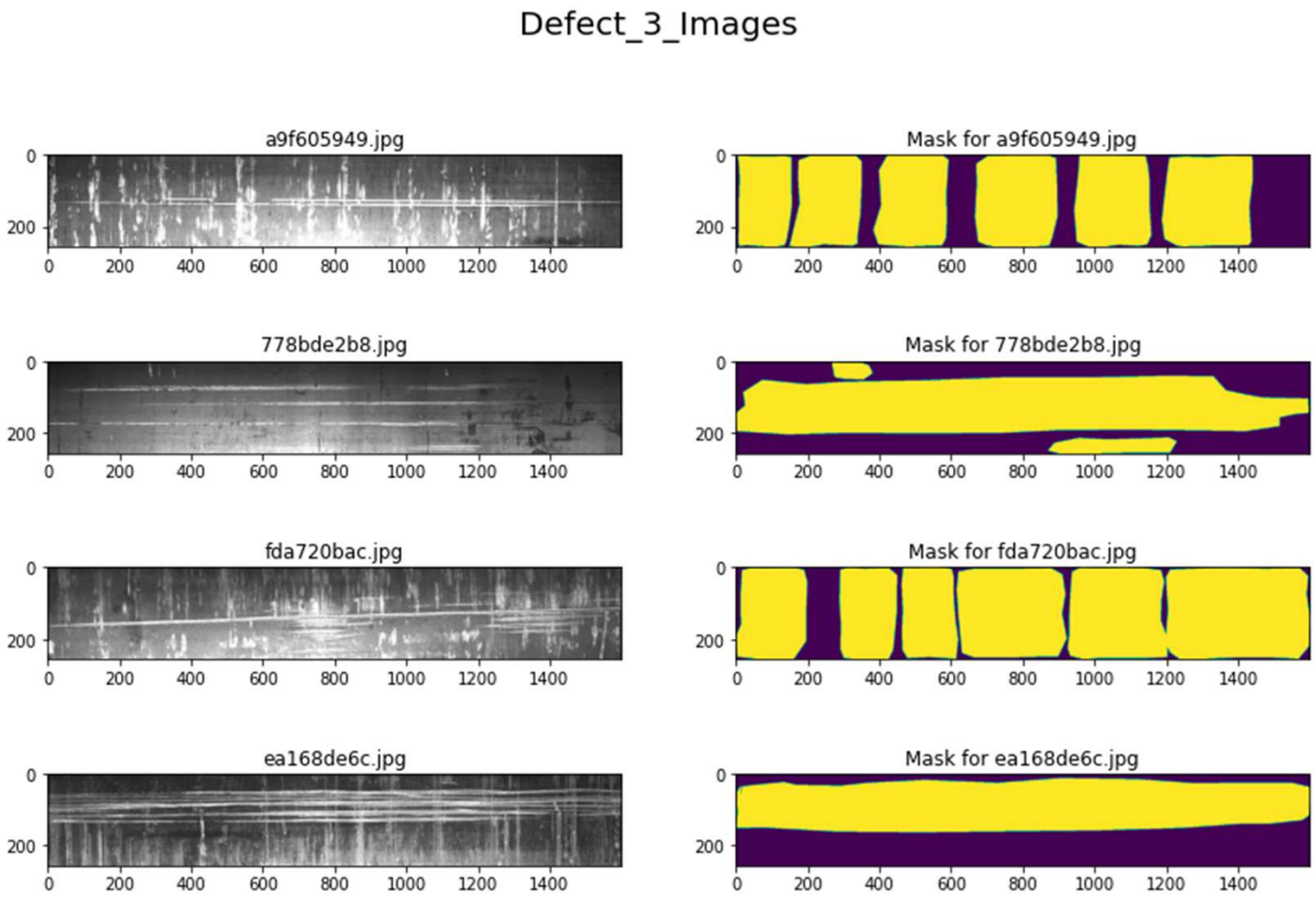

Defect_2_Images
90f4dd6e5.jpg
Mask for 90f4dd6e5.jpg
bb3421ee8.jpg
Mask for bb3421ee8.jpg
b34091037.jpg
Mask for b34091037.jpg
776ed4605.jpg
Mask for 776ed4605.jpg
0
200
0
200
400
600
800
1000
1200
1400


Defect_3_Images
4c3afd528.jpg
Mask for 4c3afd528.jpg
20964a4f0.jpg
Mask for 20964a4f0.jpg
60d7248a8.jpg
Mask for 60d7248a8.jpg
bdd1c7fa8.jpg
Mask for bdd1c7fa8.jpg
0
200
0
200
400
600
800
1000
1200
1400

Defect_4_Images



Problem area: Images with large mask areas

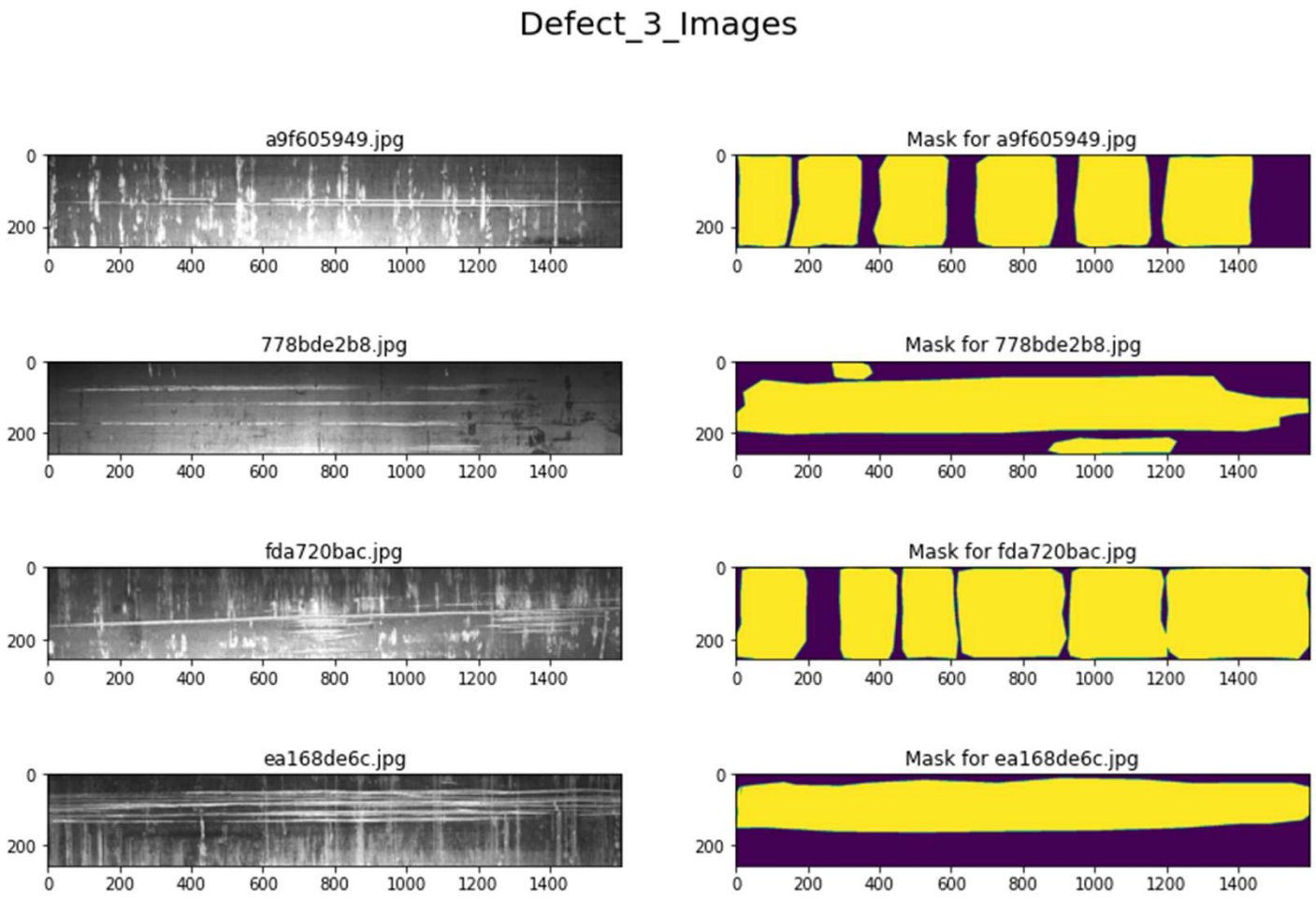


409,600 locations.

rle_Masks > 200,000

Masks covered more than 50% of the image.

Problem area: Images with large mask areas



Few samples with large mask areas were originally labeled as ‘Defect_3’.

Visually and by mask_size, they belong to the class ‘Defect_4’.

By industry standards they belong to ‘Defect_3’