Making Internal Corrosion Dynamic Segmentation based on In-line Inspection for OnshorePipeline

Kreshna Mukti Wibawa

July – August 2021

Root cause & Objective

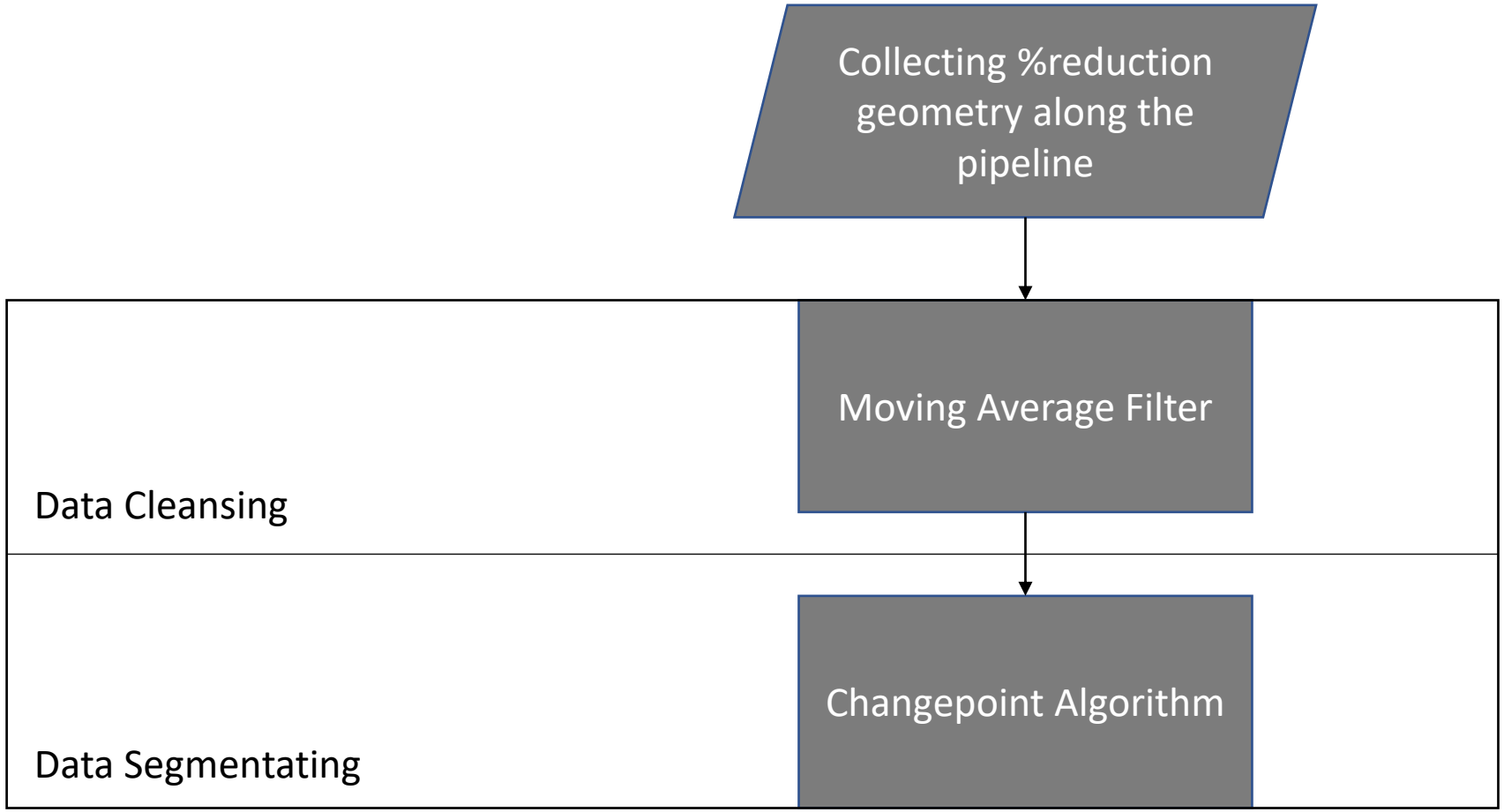
- Root Cause:

Corrosion handling is usually treated along the pipeline. This problem leads to uncertainty maintenance cost.

- Objective:

Making segmentation (range-based) to automatically detect change points through the minimization of the cost function.

Methodology



Collecting Data

- Data is collected from offline database. Data is obtained in the form of %reduction of depth, length, and width along the In-line Inspection of the pipeline

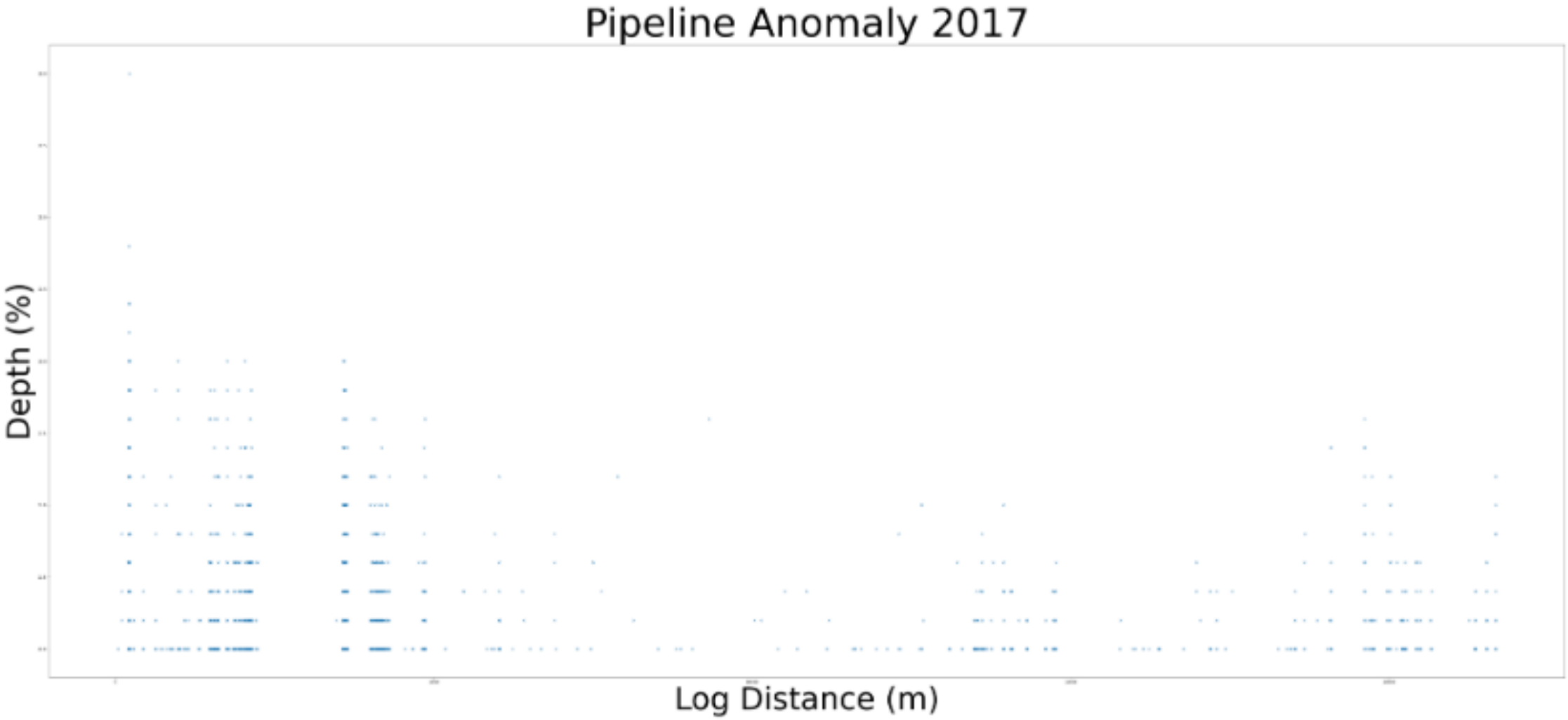
Libraries & Input:

```
import pandas as pd
from matplotlib import pyplot as plt
import math
import statistics
import ruptures as rpt

data = pd.read_csv('ILI2017.csv')
```

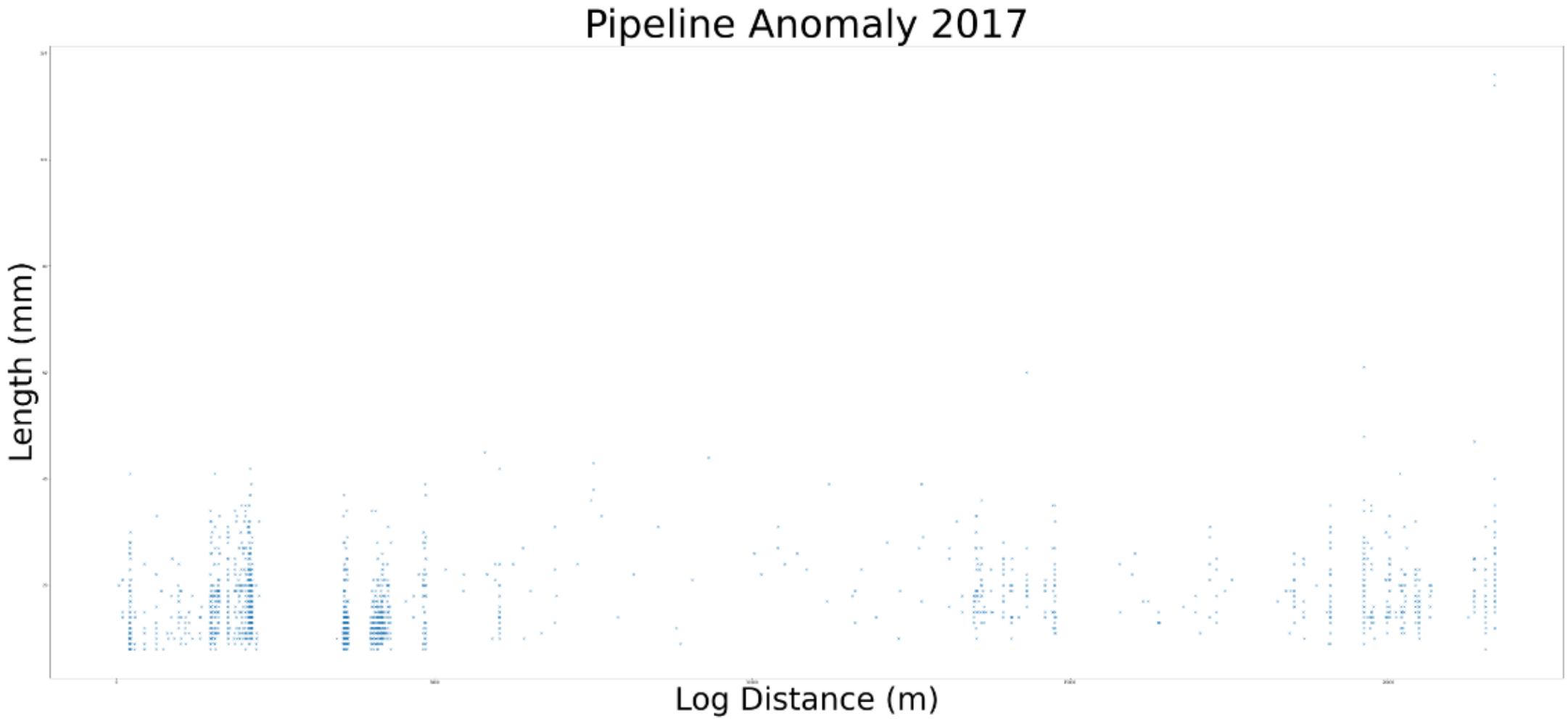
Output:

	distance	depth	length	width
0	44.20	10	20	15
1	97.85	14	14	16
2	98.25	12	15	16
3	98.82	11	21	15
4	202.60	13	10	15
...	...	...	...	...
1783	21673.72	11	22	15
1784	21673.79	10	27	15
1785	21673.80	10	23	57
1786	21673.86	11	24	15
1787	21673.95	12	35	21

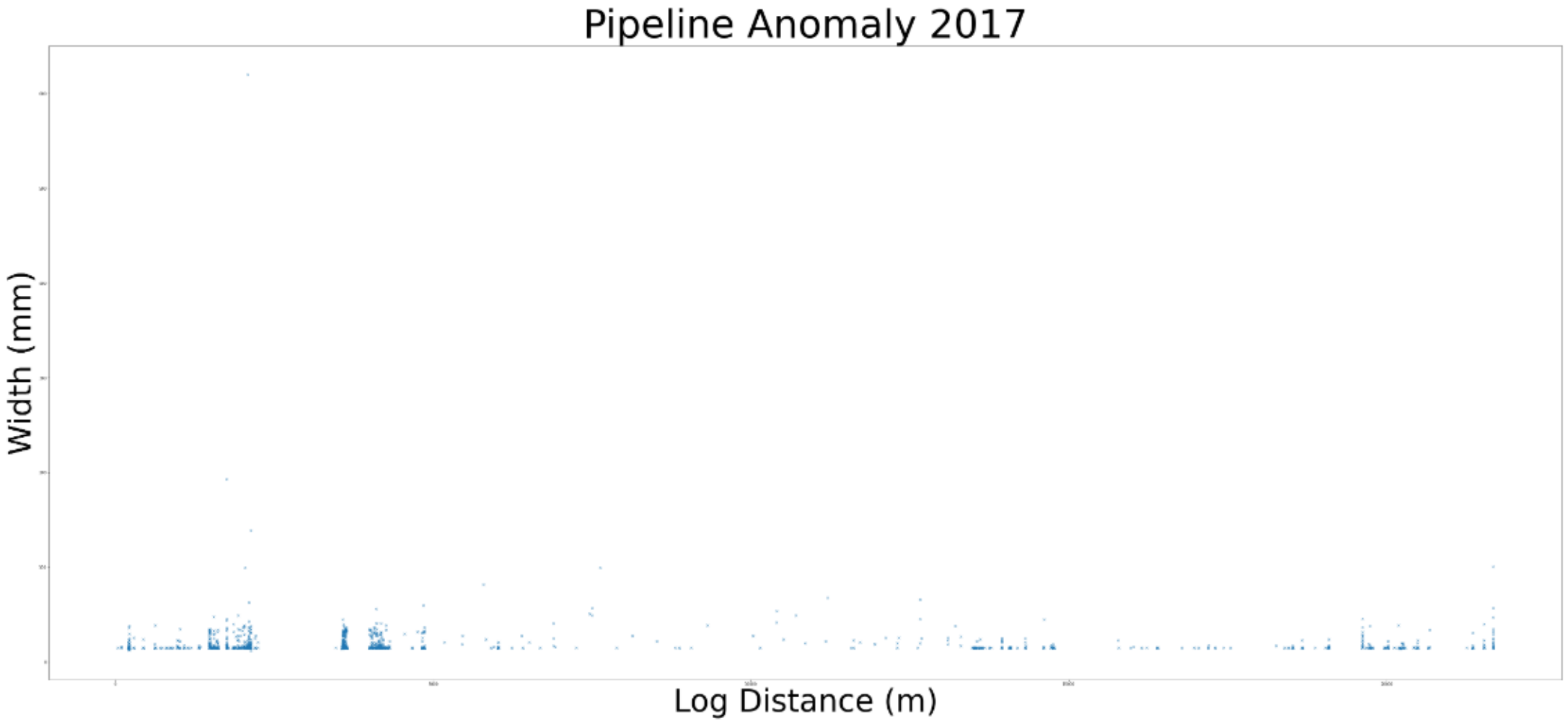
Collecting Data (%reduction of Depth)



Collecting Data (%reduction of Length)



Collecting Data (%reduction of Width)



Data Cleansing

- Data is cleansed by using Moving Average Filter (MAF).

```
Lraw = round(data.distance.shape[0]/100)
```

```
if (Lraw%2==0):
    Lindex = Lraw -1
else:
    Lindex = Lraw
```

```
Mraw = max(31, Lindex)
```

```
Mindex = int((Mraw+1)/2)
```

```
Lraw = round(data.distance.shape[0]/100)
Mindex = int((Mraw+1)/2)

if (Lraw%2==0):
    Lindex = Lraw -1
else:
    Lindex = Lraw
    Mraw = max(31, Lindex)
```

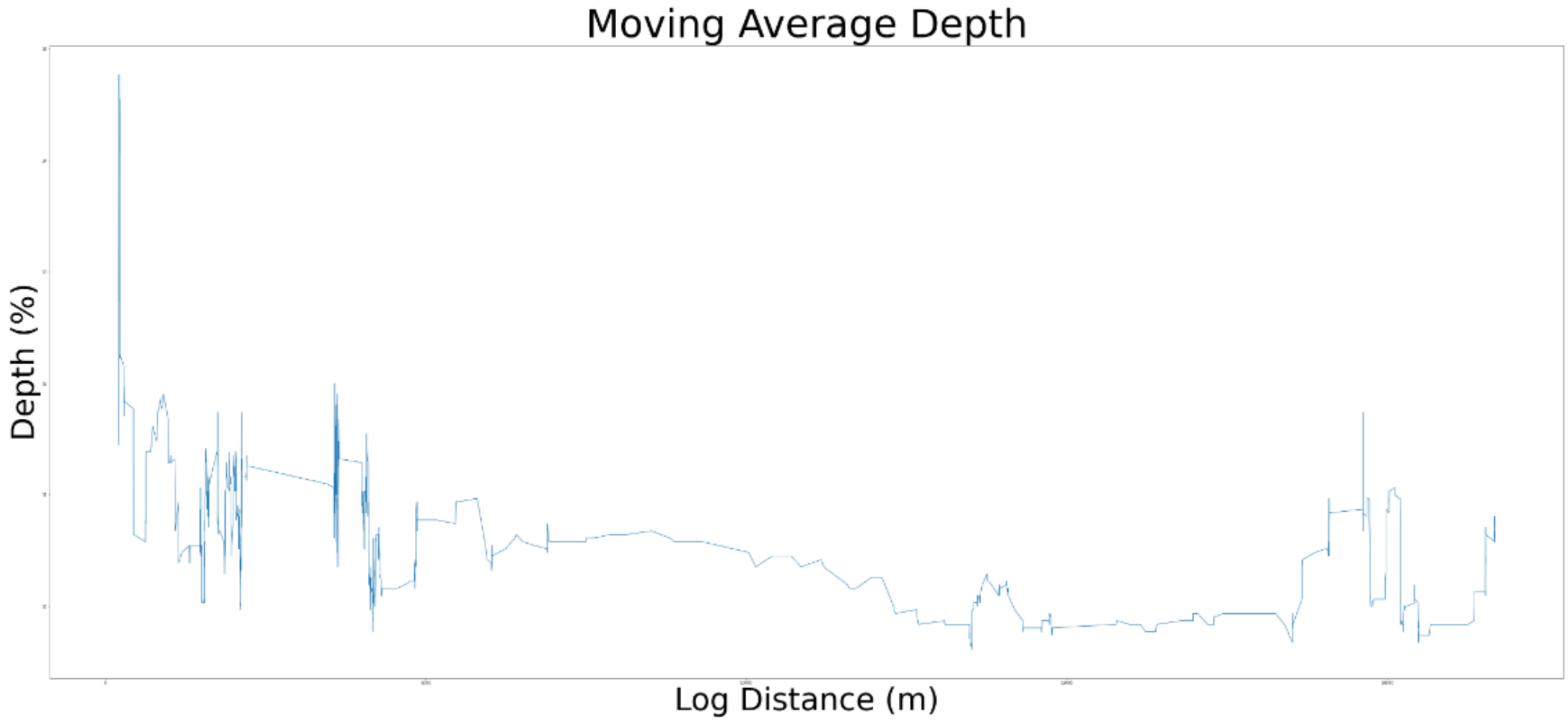
Example for %reduction of depth:

```
list_movdepth = []
for i in range (1, data.distance.shape[0]-Mraw+2):
    a = i-1
    b = Mraw + i - 3
    sumdepth = sum(data.depth[a:b])
    movdepth = sumdepth/Mraw
    list_movdepth.append(movdepth)
movingdepth = pd.DataFrame({'movdepth' : (list_movdepth)})
```

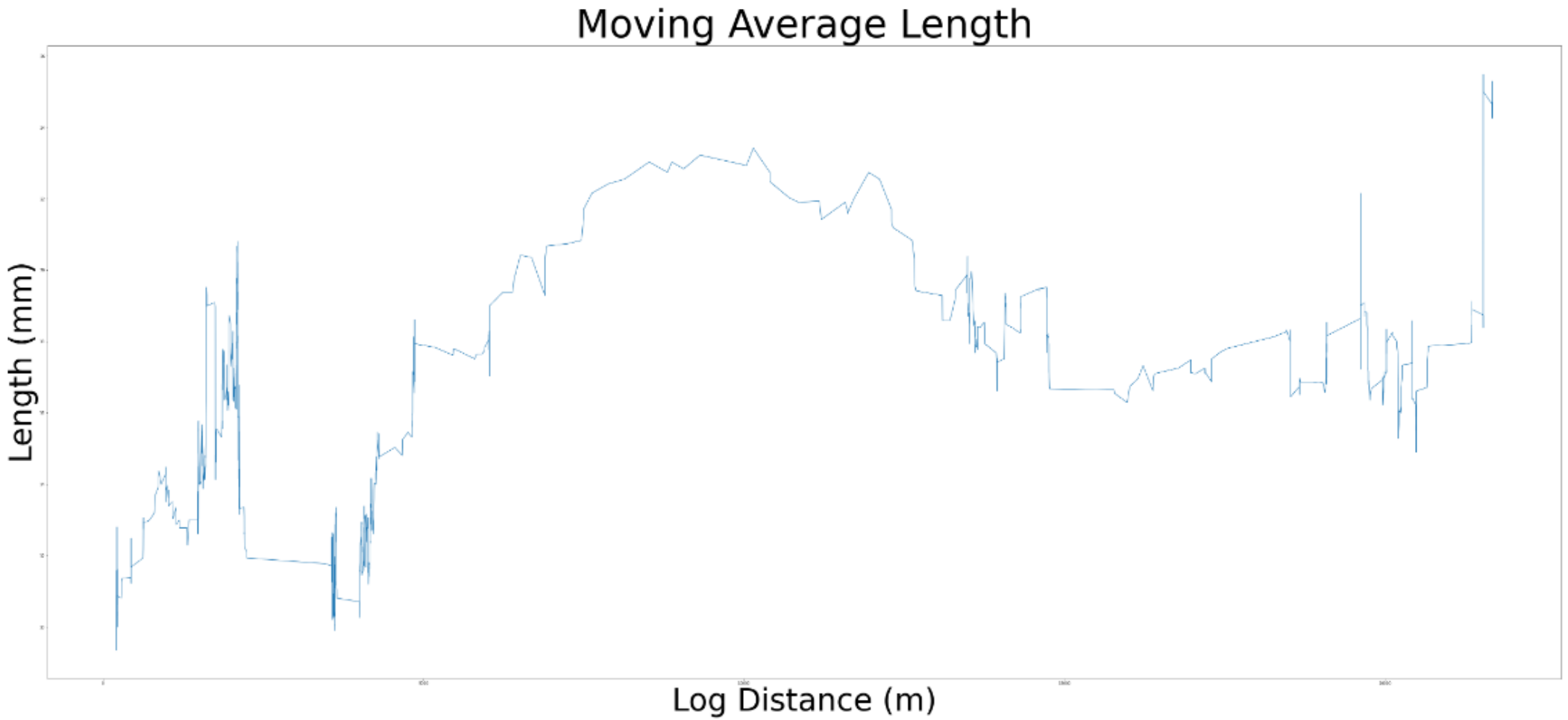
Output:

	mdepth	mlength	mwidth
0	11.451613	10.870968	14.096774
1	11.516129	10.516129	14.096774
2	11.483871	10.387097	14.064516
3	11.451613	10.225806	14.032258
4	11.483871	9.870968	14.032258
...	...	...	...
1753	10.806452	24.258065	20.612903
1754	10.741935	24.419355	20.612903
1755	10.774194	24.870968	20.612903
1756	10.741935	25.000000	20.612903
1757	10.709677	25.290323	21.967742

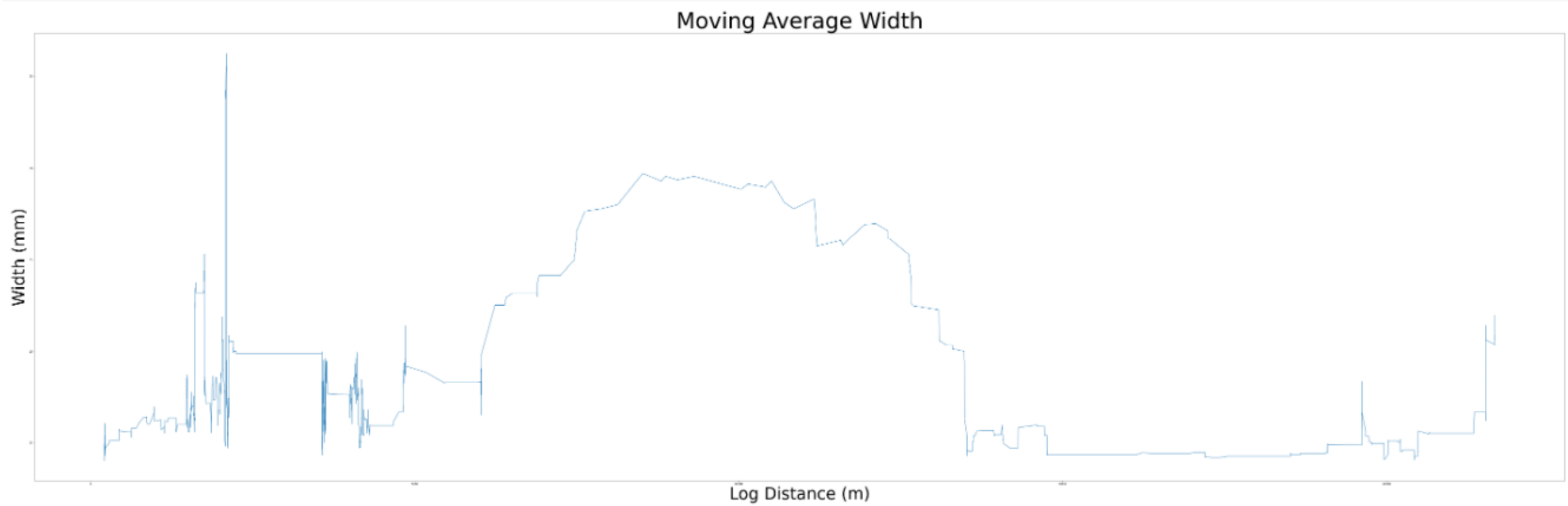
Data Cleansing (%reduction of depth)



Data Cleansing (%reduction of length)



Data Cleansing (%reduction of width)



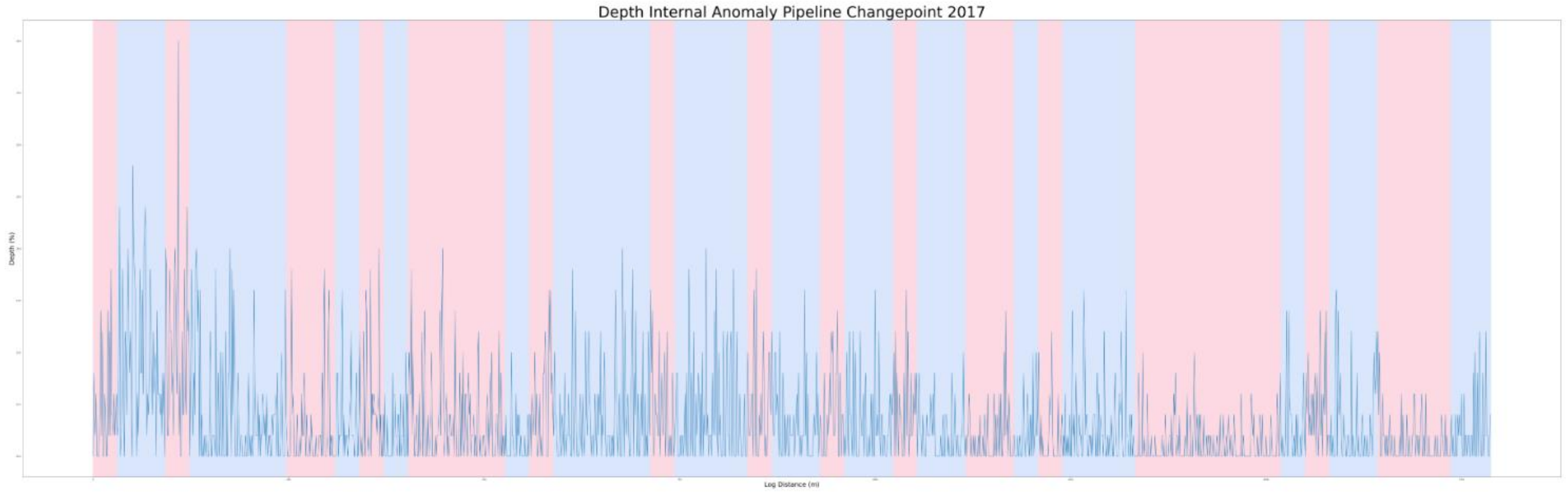
Data Segmentating

- Data is segmented by using Changepoint Algorithm.

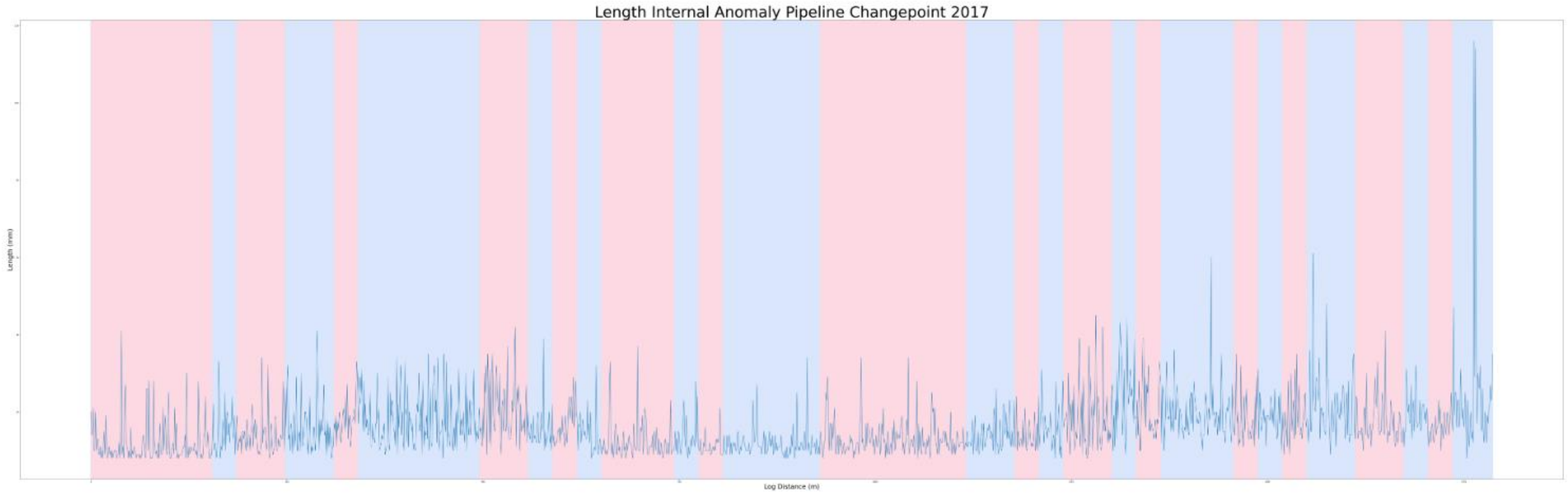
Example for %reduction of depth:

```
# change point detection
model = "l1"  # "l2", "rbf"
algo = rpt.Pelt(model=model, min_size=3, jump=31).fit(data.depth.values)
my_bkps = algo.predict(pen=0)
my_bkps.insert(0, 0)
df_changepoint = pd.DataFrame({'changepoint' : (my_bkps)})

list_defect_difference = []
for i in range (1, df_changepoint.changepoint.shape[0]):
    for j in range (1, df_changepoint.changepoint.shape[0]):
        defect_difference = df_changepoint.changepoint[j] - df_changepoint.changepoint[j-1]
        list_defect_difference.append(defect_difference)
    if min(list_defect_difference) < 30 :
        my_bkps = algo.predict(pen=i)
        my_bkps.insert(0, 0)
        df_changepoint = pd.DataFrame({'changepoint' : (my_bkps)})
        list_defect_difference = []
    else:
        my_bkps = algo.predict(pen=i-1)
        my_bkps.insert(0, 0)
        df_changepoint = pd.DataFrame({'changepoint' : (my_bkps)})
        list_defect_difference = []
        break

# show results
rpt.display(data.depth, my_bkps, figsize=(100, 30))
plt.title('Depth Internal Anomaly Pipeline Changepoint 2017', fontsize = 70)
plt.xlabel('Log Distance (m)', fontsize = 30)
plt.ylabel('Depth (%)', fontsize= 30)
plt.show()
```

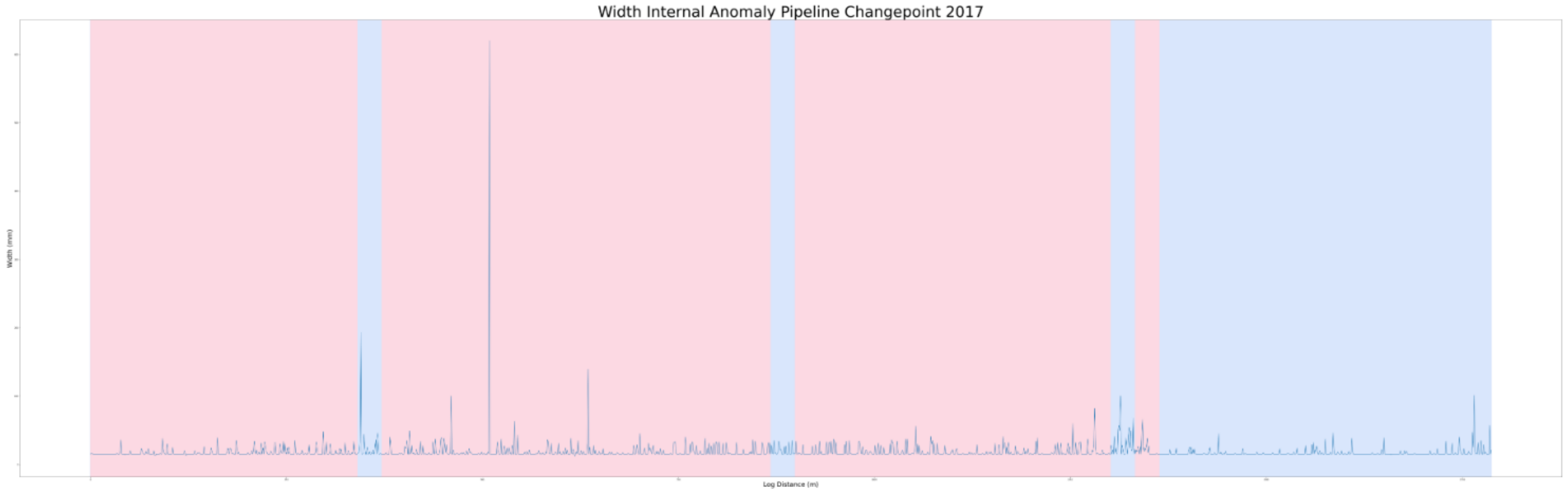
Data Segmentating (%reduction of depth)



Data Segmentating (%reduction of length)



Data Segmentating (%reduction of width)



Results

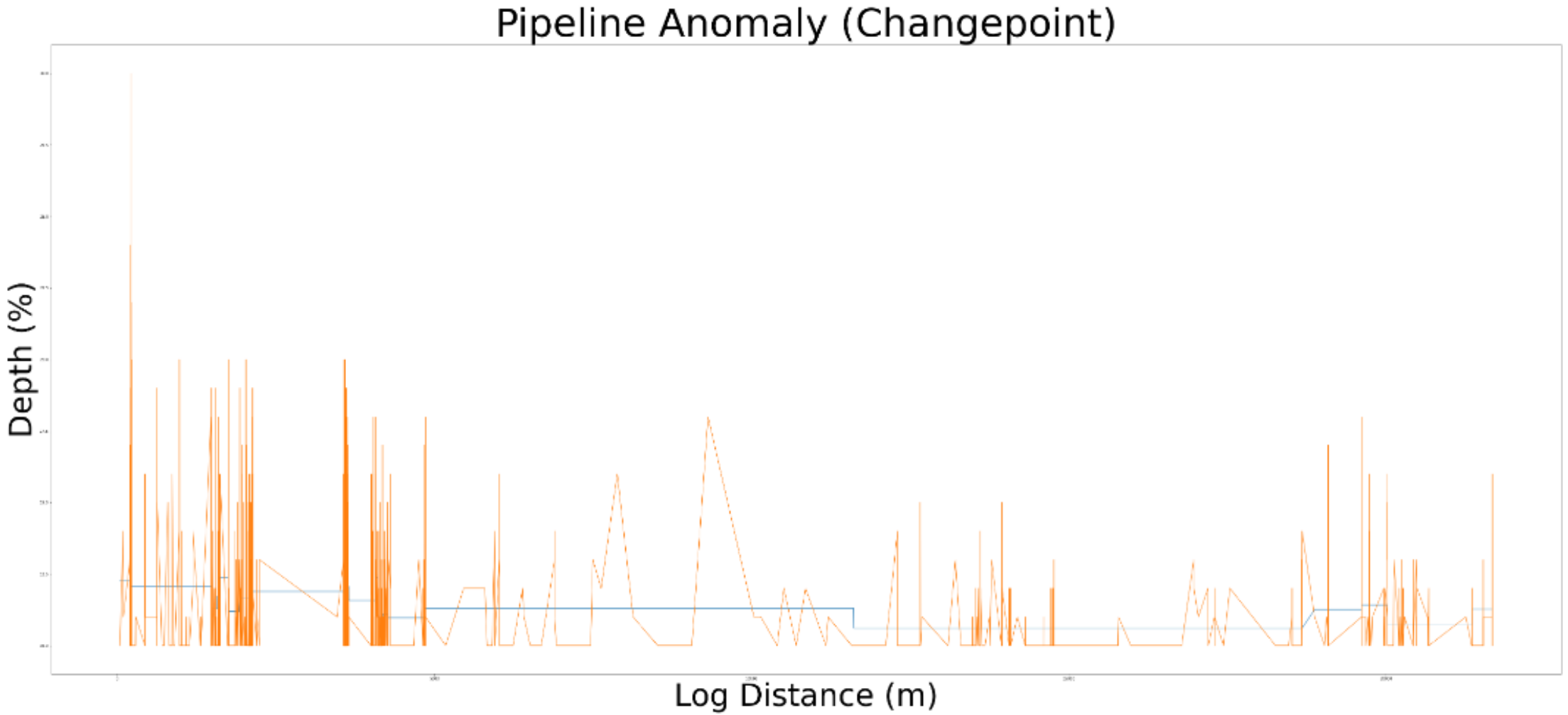
- From Changepoint Algorithm, It can be visualized by comparing the raw data between the segmented data.

For example of %reduction of depth:

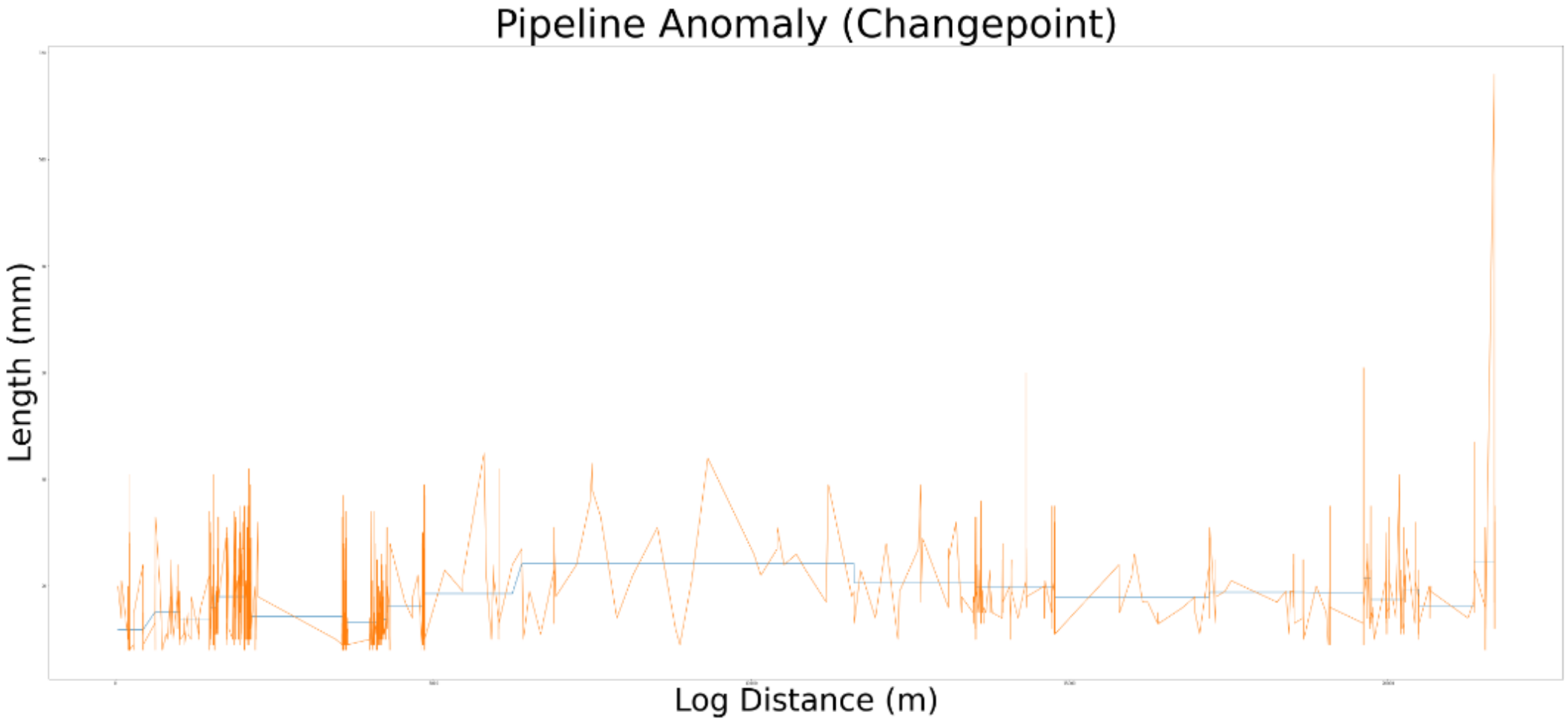
```
list_segment = []
for i in range (1, df_changepoint.changepoint.shape[0]):
    for j in range (1, df_changepoint.changepoint[i]-df_changepoint.changepoint[i-1]+1):
        a = 0
        b = statistics.mean(data.depth[df_changepoint.changepoint[i-1]: df_changepoint.changepoint[i]])
        x = i
        y = (a*i +b)
        list_segment.append(y)
segmentav = pd.DataFrame({'segmentav' : (list_segment)})
```

```
plt.figure(figsize=[70,30])
plt.plot(data.distance, list_segment)
plt.title('Pipeline Anomaly (Changepoint)', fontsize = 100)
plt.xlabel('Log Distance (m)', fontsize = 80)
plt.ylabel('Depth (%)', fontsize = 80)
plt.plot(data.distance, data.depth)
plt.xlabel ('Log Distance (m)', fontsize=80)
plt.ylabel ('Depth (%)', fontsize=80)
plt.show()
```

Result (%reduction of depth)



Result (%reduction of length)



Result (%reduction of width)