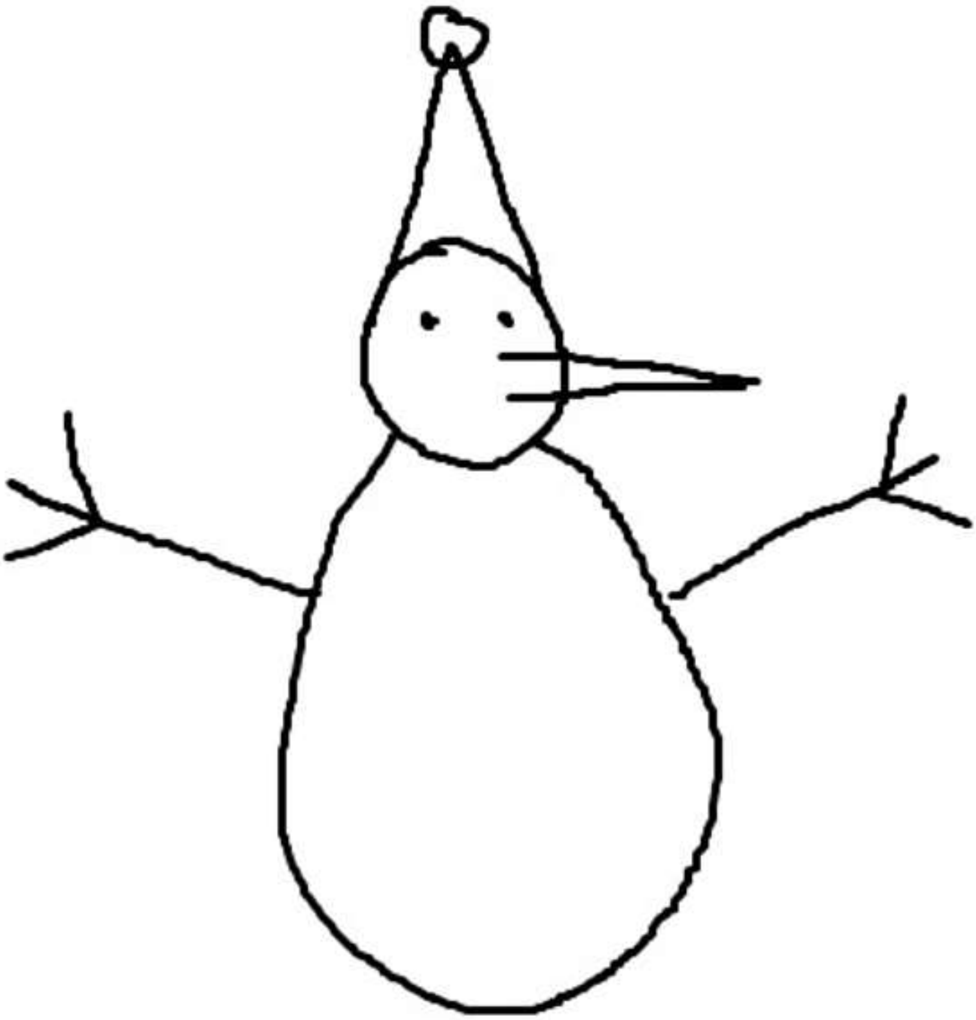
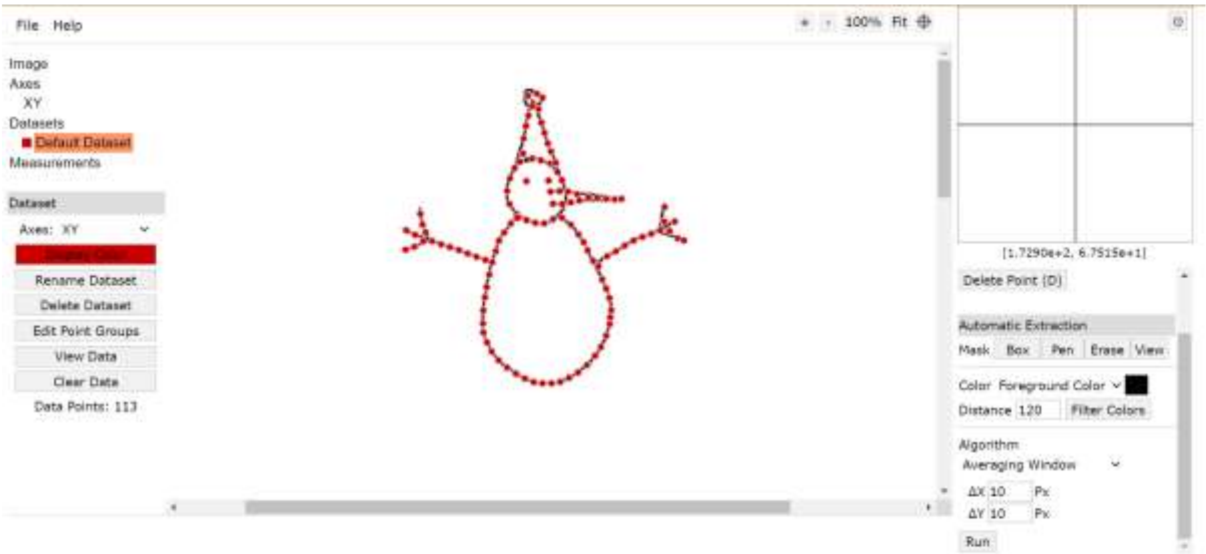
Drawing



Acquisition



X'	Y'
15.59208	44.44141
16.19278	50.30011
18.36453	45.33247
18.83013	48.95239
20.79677	55.33012
21.11841	47.00785
21.2427	52.07108
22.08974	49.72505
23.90608	47.05889
26.56695	46.10101
29.23388	45.24337
31.89879	44.35231
34.55295	43.28304
37.21046	42.26947
39.87148	41.31381
38.3267	25.95386
37.9653	22.01279
37.92314	19.2663
39.06253	29.95619
39.77008	33.08045
38.8516	16.22167
40.79594	36.83774
40.16785	13.45972
41.94771	40.61419
43.24564	43.98475
42.10853	10.80628
44.42796	47.19478
46.18454	49.68194
44.56907	8.576413
47.81233	62.32194
48.31438	58.35462
48.35808	52.42333
48.90533	66.09964
50.52488	69.24141
47.17819	6.760887
50.3548	55.30611
50.52873	54.09266
52.26257	71.22053
53.4147	73.7027
49.80412	5.223815
53.84135	78.26998
53.20876	53.45222
55.96957	91.25232
53.92391	65.30326
54.81324	82.08853

55.74508	85.24342
57.05327	88.03643
56.06801	72.16439
52.45895	4.165687
55.8777	52.62799
58.74035	91.87487
58.97849	87.22029
58.75443	71.62976
55.15545	3.798126
59.57013	83.09932
58.5863	52.46092
60.39741	90.66494
60.37021	79.97505
61.58719	70.68473
61.16659	76.78953
60.73047	65.46662
61.14626	62.11774
57.87413	3.798126
61.66592	58.44213
61.38228	53.74181
62.00734	74.33913
63.28324	70.90856
63.62061	68.08039
63.97778	62.2811
60.61969	4.243654
64.46025	58.33055
64.14841	54.18689
64.81588	65.58914
66.48003	62.10753
63.41298	5.479994
67.1268	58.83197
66.71882	52.07108
69.25469	61.32878
66.23114	7.128447
69.83094	59.41004
68.8884	49.11528
70.35091	46.7292
69.08156	9.311535
72.65865	60.39768
71.43732	44.25429
72.87146	41.10552
71.7256	12.17034
75.36579	60.20642
74.44471	39.38449
73.74067	14.84536
75.35227	35.40834
78.06834	59.93911
77.26064	42.45882

75.55506	18.29051
76.41798	32.59037
77.23654	29.77241
76.85483	21.80859
77.55552	26.04779
77.12881	23.89144
80.77667	59.76758
80.06737	43.91792
82.76263	59.91237
82.8936	45.70003
85.70369	47.21483
88.48891	48.31751
91.26674	49.29767
94.06576	50.31406
95.55569	53.02061
95.83373	57.40071
96.91596	50.15832
98.65821	53.0104
99.33876	47.90539
101.1536	47.26216

Code for loading, cleaning, transformation and visualizing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the CSV file into a pandas dataframe
df = pd.read_csv('Snowman.csv')

# cleansing to remove any invalid data points
df = df[(df["X'"] >= 0) & (df["X'"] <= 100) & (df["Y'"] >= 0) & (df["Y'"] <= 100)]

# Discretize the data to a 1000x1000 boolean matrix
matrix_size = 1000
sparse_matrix = np.zeros((matrix_size, matrix_size), dtype=bool)

# Transform the coordinates to fit into the 1000x1000 matrix
for _, row in df.iterrows():
    x, y = int(row["X'"] * (matrix_size / 100)), int(row["Y'"] * (matrix_size / 100))
    sparse_matrix[y, x] = True

# Rotate the matrix by 90 degrees clockwise (Transpose and then reverse columns)
rotated_matrix = np.transpose(sparse_matrix)  # Transpose the matrix
```

```python
# Flip the original matrix horizontally (Reverse the columns)
flipped_matrix = sparse_matrix.copy()
for i in range(matrix_size):
    flipped_matrix[:, i] = sparse_matrix[:, matrix_size - 1 - i]
# Extract the coordinates from the rotated sparse matrix
rotated_coords = np.column_stack(np.where(rotated_matrix))
# Extract the coordinates from the flipped sparse matrix
flipped_coords = np.column_stack(np.where(flipped_matrix))
# Plot the scatter plot for the original, rotated, and flipped images
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
# Original image
original_coords = np.column_stack(np.where(sparse_matrix))
ax[0].scatter(original_coords[:, 1], matrix_size - original_coords[:, 0], s=1)
ax[0].set_title('Original Image')
ax[0].invert_yaxis()

# Rotated image
ax[1].scatter(rotated_coords[:, 1], matrix_size - rotated_coords[:, 0], s=1)
ax[1].set_title('Rotated Image')
ax[1].invert_yaxis()

# Flipped image
ax[2].scatter(flipped_coords[:, 1], matrix_size - flipped_coords[:, 0], s=1)
ax[2].set_title('Flipped Image')
ax[2].invert_yaxis()

plt.show()
```

```python
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt


# Load the CSV file into a pandas dataframe

df = pd.read_csv('Snowman.csv')


# cleansing to remove any invalid data points

df = df[(df["X'"] >= 0) & (df["X'"] <= 100) & (df["Y'"] >= 0) & (df["Y'"] <= 100)]


# Discretize the data to a 1000x1000 boolean matrix

matrix_size = 1000

sparse_matrix = np.zeros((matrix_size, matrix_size), dtype=bool)


# Transform the coordinates to fit into the 1000x1000 matrix
```

```
for _, row in df.iterrows():
    x, y = int(row["X'"] * (matrix_size / 100)), int(row["Y'"] * (matrix_size / 100))
    sparse_matrix[y, x] = True

# Rotate the matrix by 90 degrees clockwise (Transpose and then reverse columns)
rotated_matrix = np.transpose(sparse_matrix)  # Transpose the matrix

# Reverse each column to rotate 90 degrees clockwise
for i in range(matrix_size):
    rotated_matrix[:, i] = rotated_matrix[:, i][::-1]

# Flip the original matrix horizontally (Reverse the columns)
flipped_matrix = sparse_matrix.copy()
for i in range(matrix_size):
    flipped_matrix[:, i] = sparse_matrix[:, matrix_size - 1 - i]
# Extract the coordinates from the rotated sparse matrix
rotated_coords = np.column_stack(np.where(rotated_matrix))
# Extract the coordinates from the flipped sparse matrix
flipped_coords = np.column_stack(np.where(flipped_matrix))
# Plot the scatter plot for the original, rotated, and flipped images
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
# Original image
original_coords = np.column_stack(np.where(sparse_matrix))
ax[0].scatter(original_coords[:, 1], matrix_size - original_coords[:, 0], s=1)
ax[0].set_title('Original Image')
ax[0].invert_yaxis()

# Rotated image
ax[1].scatter(rotated_coords[:, 1], matrix_size - rotated_coords[:, 0], s=1)
ax[1].set_title('Rotated Image')
ax[1].invert_yaxis()
```

```
# Flipped image

ax[2].scatter(flipped_coords[:, 1], matrix_size - flipped_coords[:, 0], s=1)

ax[2].set_title('Flipped Image')

ax[2].invert_yaxis()


plt.show()
```

Results