Name – Sapna Mishra

Roll no. – MA23C042

Report on Image Transformation and Visualization

Introduction

This report outlines the process of transforming a line sketch of planetary objects using matrix operations. The objectives were to rotate the sketch by 90 degrees clockwise and flip it horizontally, achieved programmatically without using traditional image processing libraries like OpenCV. The process included data acquisition, cleansing, transformation, and visualization, as implemented in the provided Python code.

Description of the Original Image

The original image is a line sketch featuring three planetary objects:

- **Object 1**: A planet with horizontal wavy lines, resembling a gas giant.
- **Object 2**: A planet with landmasses and water bodies, resembling Earth.
- **Object 3**: A larger planet with a prominent ring system, resembling Saturn.

Data Acquisition

The line sketch was converted into a 2D dataset of x-y coordinates using an online tool:

1. **Image Upload**: The sketch was uploaded to Starry Data.
2. **Fiducial Points Selection**: Fiducial points {X1, X2, Y1, Y2} were defined with X = [1, 100] and Y = [1, 100].
3. **Colour Matching**: The colour of the lines was matched to ensure accurate data extraction.
4. **Data Extraction**: Red dots along the lines of the sketch were extracted and saved as a CSV file for further processing.

Data Cleansing & Loading

The CSV file was processed using pandas and NumPy and also I have included my codes.

1. **Loading Data**: The CSV file was loaded into a pandas Data Frame.

```
df = pd.read_csv("Planet.csv")
```

2. **Data Cleaning**: Columns were renamed, and coordinates were rounded:

```
df.columns = ['x', 'y']

df['x'] = df['x'].round(2)

df['y'] = df['y'].round(2)
```

- The DataFrame was visualized using scatter plots to check the data distribution:

sns.scatterplot(data=df, x='y', y='x')

3. **Data Conversion**: The DataFrame was converted into a sparse matrix and then into a boolean matrix:

sparse_matrix = csr_matrix(df)

arr = sparse_matrix.toarray()

   - The boolean matrix was created by discretizing the x and y coordinates:

discretized_x = np.clip(((arr[:, 0] - min_x) / (max_x - min_x) * (datapoints - 1)).astype(int), 0, datapoints - 1)

discretized_y = np.clip(((arr[:, 1] - min_y) / (max_y - min_y) * (datapoints - 1)).astype(int), 0, datapoints - 1)

boolean_m = np.zeros((datapoints, datapoints), dtype=bool)

for x, y in zip(discretized_x, discretized_y):

  boolean_m[x, y] = True

sparse_matrix1 = csr_matrix(boolean_m)

5. Transformation

Matrix operations were used to transform the image:

1. **Rotation by 90 Degrees**:

python

rotate_df1 = np.rot90(num_1, k=-1)

This operation rotated the boolean matrix 90 degrees clockwise.

2. **Horizontal Flip**:

python

rotate_df2 = np.rot90(num_1, k=-2)

This operation flipped the matrix horizontally by rotating it 180 degrees.

3. **Conversion for Visualization**:

   - The transformed matrices were converted into x-y coordinates for visualization.

6. Visualization

The transformed images were visualized using scatter plots:

1. **Rotated Image**:

python

rows, cols = np.nonzero(rotate_df1)

```
plt.figure(figsize=(10, 10))

ax1 = plt.subplot()

ax1.scatter(rows, cols, s=30)

plt.grid(True)

plt.show()
```

2. **Horizontally Flipped Image**:

```
python

rows, cols = np.nonzero(rotate_df2)

plt.figure(figsize=(8, 8))

ax1 = plt.subplot()

ax1.scatter(rows, cols, s=70)

plt.grid(True)

plt.show()
```

Conclusion

The task effectively demonstrated how to perform image transformations programmatically using matrix operations. By converting a line sketch into a coordinate dataset, cleansing the data, applying matrix transformations, and visualizing the results, this approach highlighted the utility of NumPy and pandas for image manipulation without relying on traditional image processing libraries.