Nutchapol Winmoon 6231320321

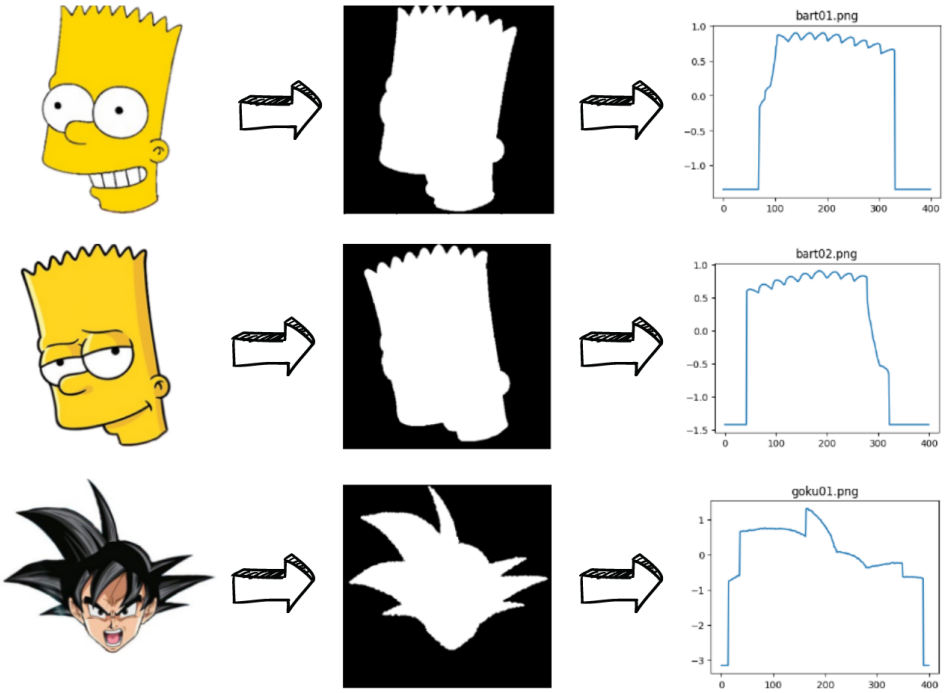
Midterm Project Report:
Animation Character Image to Time-series

2110430 Time Series Mining and Knowledge Discovery

Introduction

Animation character image classification is a challenging problem in the field of computer vision and multimedia analysis. The identification and classification of characters in animation require an accurate and efficient approach due to the high variability and complexity of animation character images. Converting images into time series data enables the application of traditional time series analysis techniques to image classification problems, reducing the need for large amounts of labeled data.

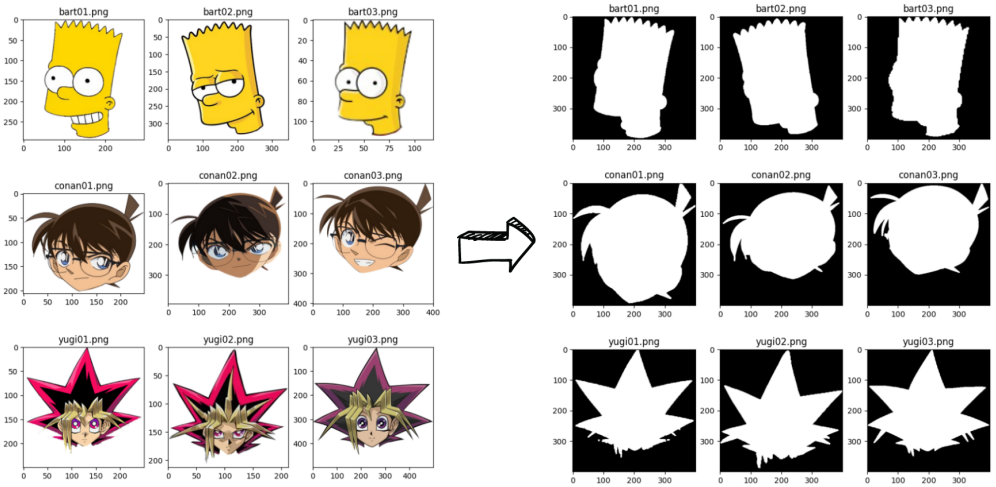
This project aims to explore the use of time series data for animation character image classification using OpenCV and Dynamic Time Warping (DTW) algorithm. We will investigate the use of OpenCV, an open-source computer vision library, to convert animation character images into time series data. We will also use DTW, a widely used algorithm for measuring the similarity between two time series, to compare the similarity of the time series data for character classification.



Nutchapol Winmoon 6231320321

Methodology

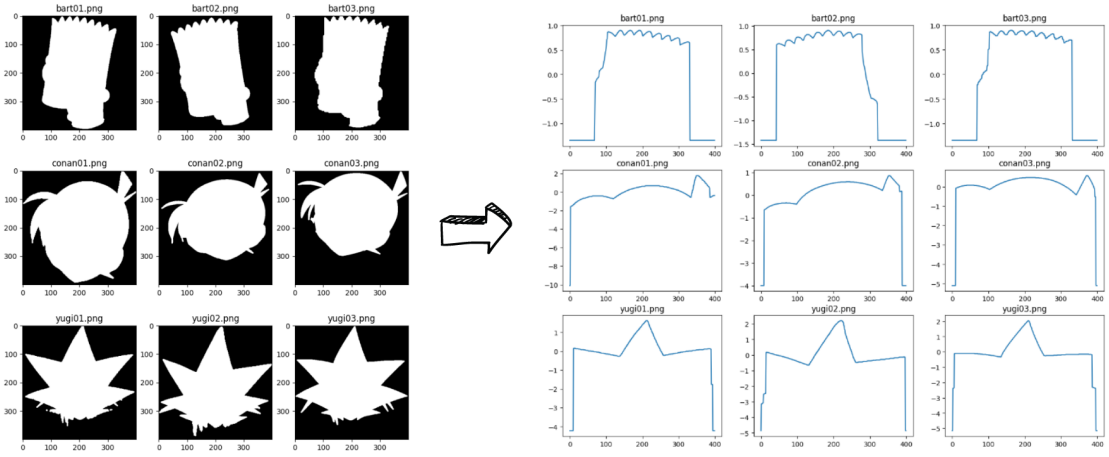
1. **Preprocessing:** converting images into binary format involves assigning pixel values of the character area to one and pixel values of the background to zero. To accomplish this task, we need to do the following steps:
    - Thresholding: converting an image into a binary format by selecting a threshold value and assigning all pixels with intensity values above the threshold to one (white) and those below the threshold to zero (black).
    - Fill contours: filling in the interior of closed curves or shapes in an image.
    - Inverted binarization: inverting a binary image to make the background of an image black (zero) and the foreground is white (one).



2. **Converting into time-series**
    - Given input of two-dimensional binary array, we explored two algorithms to convert them into time-series data including:

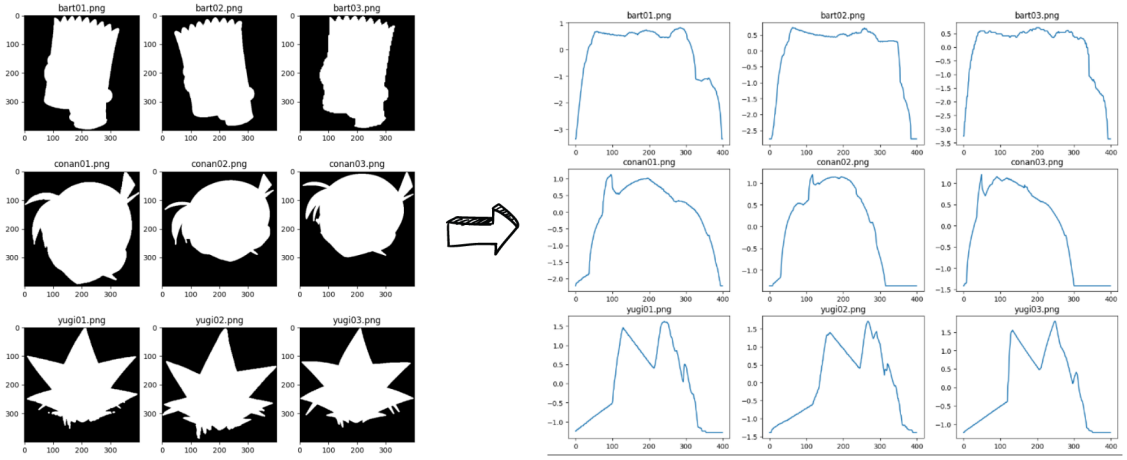
    1. Finding first pixel vertically

        This algorithm involves scanning each column of the input array from top to bottom until the first non-zero (or non-black) pixel is found. The position of this pixel in each column is recorded as a time-series data point.



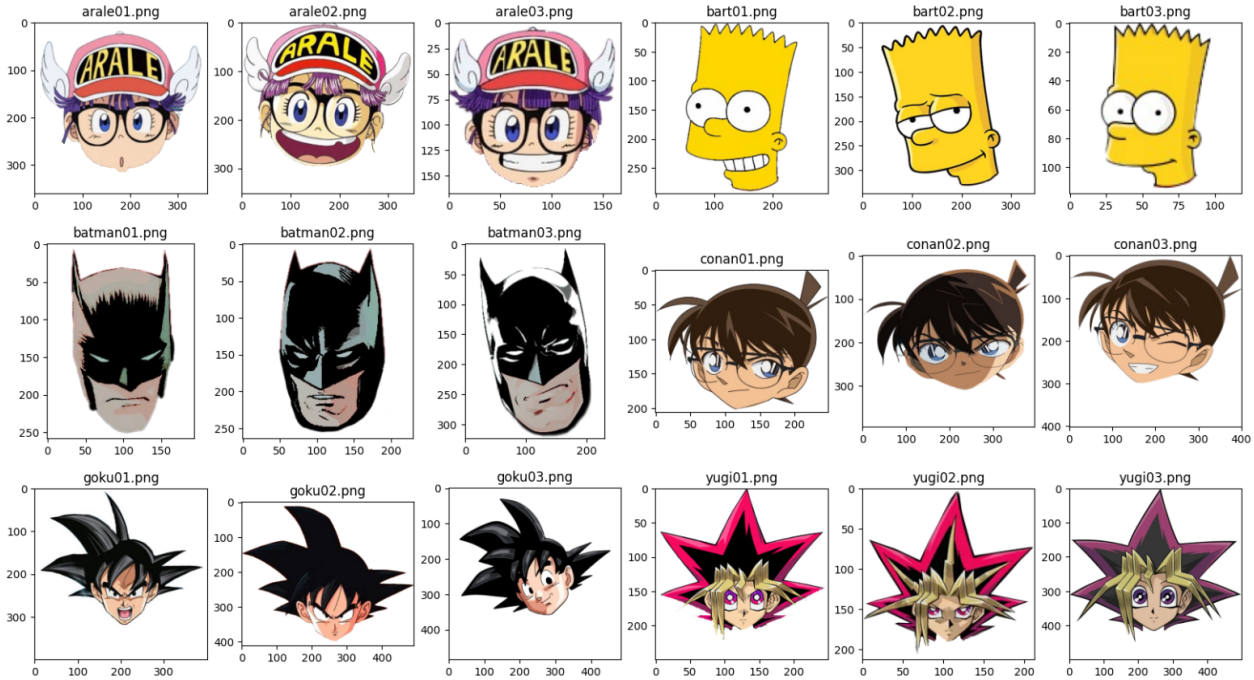
2. Summing the rows

This algorithm involves summing the values in each row of the input array and recording the resulting value as a time-series data point.



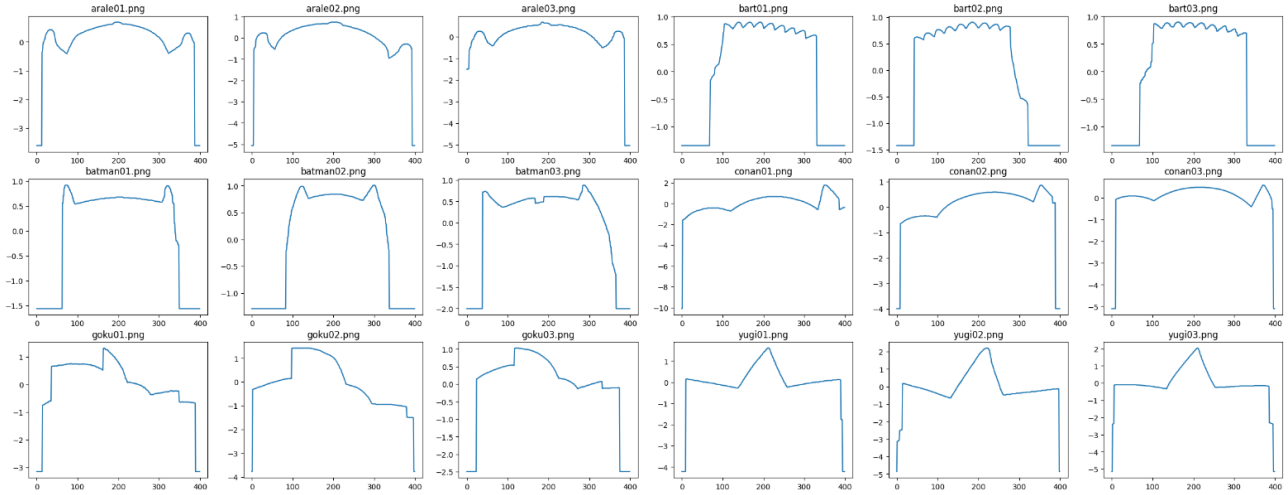
Results

Sample input images:

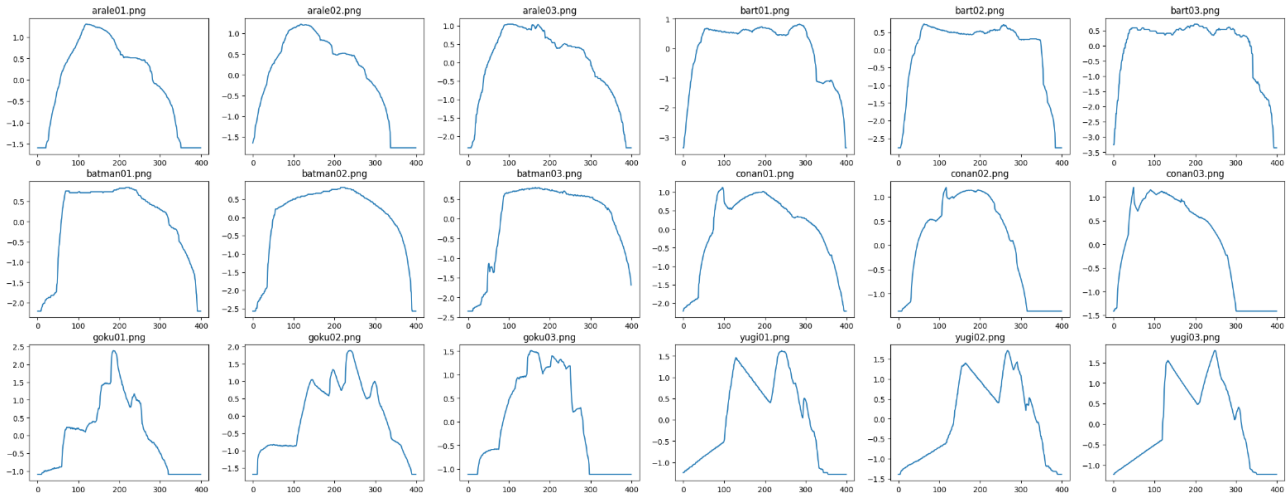


Nutchapol Winmoon 6231320321

Converting into Time-series by finding first pixel vertically:



Converting into Time-series by summing the rows:



To demonstrate that the resulting time series data are potentially useful as they can be used as representatives of the original data. The results from from KNN-classification algorithm (K=1) using Dynamic Time Warping (DTW) algorithms for distance measuring are shown below:

```
classify(data=row_sum, labels=labels)
✓ 3m 20.6s
arale01 is closest to arale02   | Correct!
arale02 is closest to arale01   | Correct!
arale03 is closest to conan01   | Wrong!
bart01 is closest to bart02     | Correct!
bart02 is closest to bart01     | Correct!
bart03 is closest to bart01     | Correct!
batman01 is closest to batman03 | Correct!
batman02 is closest to batman01 | Correct!
batman03 is closest to batman01 | Correct!
conan01 is closest to batman01  | Wrong!
conan02 is closest to conan03   | Correct!
conan03 is closest to conan02   | Correct!
goku01 is closest to goku03     | Correct!
goku02 is closest to yugi02     | Wrong!
goku03 is closest to goku01     | Correct!
yugi01 is closest to yugi02     | Correct!
yugi02 is closest to yugi01     | Correct!
yugi03 is closest to yugi01     | Correct!
```

```
classify(data=first_col, labels=labels)
✓ 3m 24.6s
arale01 is closest to arale03   | Correct!
arale02 is closest to arale03   | Correct!
arale03 is closest to arale02   | Correct!
bart01 is closest to bart03     | Correct!
bart02 is closest to batman01   | Wrong!
bart03 is closest to bart01     | Correct!
batman01 is closest to bart02   | Wrong!
batman02 is closest to bart01   | Wrong!
batman03 is closest to batman01 | Correct!
conan01 is closest to conan02   | Correct!
conan02 is closest to arale01   | Wrong!
conan03 is closest to arale02   | Wrong!
goku01 is closest to goku03     | Correct!
goku02 is closest to yugi03     | Wrong!
goku03 is closest to batman03   | Wrong!
yugi01 is closest to yugi03     | Correct!
yugi02 is closest to yugi03     | Correct!
yugi03 is closest to yugi02     | Correct!
```

For more information about the implementation: explore this notebook.