Texas A&M University
Department of Computer Science and Engineering

Rubine Feature Implementation

Author:
Millennium Bismay
UIN: 633002191

Supervisor:
Dr. Tracy Hammond

CSCE624: Sketch Recognition
Assignment - 1(a)
September 9, 2022

Contents

1 Motivation

A sketch is a free flow of hand-drawn figures on any surface. It is highly intuitive, easy to accomplish the desired result, and faster than any other input mechanism. The sketch information is also easy to understand for another human. However, the way a computer perceives a sketch is as a collection of points at some time interval. For a human, it is almost impossible to understand what an actual sketch is from the coordinates of points against the time axis. Once a computer intakes the points of a sketch, its very important to distinguish between dissimilar sketches and establish a relation between similar sketches. There are multiple features that can be extracted from a sketch. Feature Extraction can help in identifying similar sketches and distinguishing dissimilar sketches. Rubine features, published by Dean Rubine, has a list of exhaustive features list which can be used to identify sketches. Out of these, a list of 13 features are very important and can be used to identify key features of any sketch.

2 Problem Statement

The dataset consists of 20 sketch-information for each alphabet, consisting of a total of 520 sketches (26*20). Each sketch-information contains data in the form (x, y, t) where (x,y) represent the coordinates of each point plotted against time (t). The task was to implement the 13 Rubine Features that are used for stroke-based recognition in sketches.

3 Approach

Python is the programming language in which all the code was written. *Jupyter* Notebook was used to implement the code for finding the 13 Rubine Features. *Pandas* was used to read the dataframe of each sketch containing the coordinates (x,y) against time t. *Numpy* was used to perform the calculations. Python in-built functions such as *math* was used to perform mathematical operations and textitos was used to access the dataset.

4 Solution

The 13 Rubine features are mathematical operations taking into account the traced points during drawing a sketch. The features are derived from the coordinates (x, y) and their respective time (t). The dataset was presented as tuples of (x, y, t). The following are the 13 Rubine Features:

- **Initial Angle:**

  Initial angle is defined as the angle between the first and the third points of a sketch.Let, (x_0, y_0) be the starting point and (x_2, y_2) be the third point of a sketch.
  Then,
  $\Delta y = y_2 - y_0$
  $\Delta x = x_2 - x_0$
  and hypotenuse (hyp) = $\sqrt{(}\Delta y^2 + \Delta x^2)$

Let, θ be the starting angle
Then,

$$Rf_{01} = cos\theta = \Delta x/hyp \tag{1}$$

$$Rf_{02} = sin\theta = \Delta y/hyp \tag{2}$$

Precaution: To avoid any situation such as divide by zero, in the case that Δy and Δx both are 0, i.e. the sketcher traces back the third point of the sketch to the first point or the first 3 points are at the same point, $numpy.divide()$ is used which returns nan when we get a case of divide by zero instead of throwing an error. *However, I never faced any such error during the assignment.*

- **Bounding Box Diagonal and Angle:**

  Bounding box is defined as the the smallest rectangle with vertical and horizontal sides that completely surrounds an image or an object.
  Let, (x_{min}, y_{max}) and (x_{max}, y_{min}) be the endpoints of the diagonal of the bounding box.
  Then,
  $\Delta y = y_{max} - y_{min}$
  $\Delta x = x_{max} - x_{min}$
  So,

$$Rf_{03} = \sqrt{(\Delta x^2 + \Delta y^2)} \tag{3}$$

  Bounding box angle is the angle between the bounding box diagonal and the x-axis. Let, θ be the bounding box angle
  So,

$$Rf_{04} = \theta = arctan(\Delta y/\Delta x) \tag{4}$$

  Precaution: Finding the angle involves finding the $arctan$ of the angle between the x-axis and the bounding box diagonal. However, the $arctan$ is defined only within the 1st and 4th quadrant, i.e. $(-\pi/2, \pi/2)$. So, to make it defined in all the quadrant, i.e. $(-\pi, \pi)$, we have to use $atan2$, which is taken from the python in-built $math$ library. *However, I never faced any such error during the assignment.*

- **Distance and Angle between endpoints:**
  Let, (x_0, y_0) be the starting point and (x_{n-1}, y_{n-1}) be the end point of a sketch
  Then,
  $\Delta y = y_{n-1} - y_0$
  $\Delta x = x_{n-1} - x_0$
  So,

$$Rf_{05} = \sqrt{(\Delta y^2 + \Delta x^2)} \tag{5}$$

  Let, θ be the angle between the endpoints
  So,

$$Rf_{06} = cos\theta = \Delta x/Rf_{05} \tag{6}$$

$$Rf_{07} = sin\theta = \Delta y/Rf_{05} \tag{7}$$

Precaution: To avoid any situation such as divide by zero, in the case that Δy and Δx both are 0, i.e. the sketcher traces back the last point of the sketch to the first point or all the points are at the same point, $numpy.divide()$ is used which returns nan when we get a case of divide by zero instead of throwing an error. *However, I never faced any such error during the assignment.*

- **Total Stroke Length:**
  Stroke length is defined as the distance of a single stroke. Every sketch contains multiple strokes. The sum of all the strokes in a sketch is the Total Stroke Length

  Let's say, the points of a sketch are (x_0, y_0), (x_1, y_1), ... , (x_{n-1}, y_{n-1})
  Stroke length is the distance between any adjacent points.

  $\Delta x_i = x_i - x_{i-1}$
  $\Delta y_i = y_i - y_{i-1}$

  Stroke Length (s_i) = $\sqrt{(\Delta y_i^2 + \Delta x_i^2)}$
  So,

$$Rf_{08} = \sum_{i=1}^{n-1} s_i \tag{8}$$

- **Total Relative Rotation, Total Absolute Rotation and Total Squared Rotation:**

  Total Relative Rotation is the sum of all angles between each stroke as respect to its previous stroke. It is also understood as the total angle traversed.
  Let, θ_i be the angle between i^{th} and $i-1^{th}$ stroke
  Let, the i^{th} stroke be from (x_{i-1}, y_{i-1}) and (x_i, y_i)
  and the $i-1^{th}$ stroke be from (x_{i-2}, y_{i-2}) and (x_{i-1}, y_{i-1})
  Then,
  $\Delta y_i = y_i - y_{i-1}$
  $\Delta x_i = x_i - x_{i-1}$

  So, $\theta_i = arctan((\Delta x_i * \Delta y_{i-1}$ - $\Delta y_i * \Delta x_{i-1})/(\Delta x_i * \Delta x_{i-i} + \Delta y_i * \Delta y_{i-1}))$

  Then,

$$Rf_{09} = \sum_{i=2}^{n-1} \theta_i \tag{9}$$

  Total Absolute Rotation is the sum of all absolute angles between strokes. It can also be understood as the total movement done during a sketch. Similarly,

$$Rf_{10} = \sum_{i=2}^{n-1} |\theta_i| \tag{10}$$

  Total Squared Rotation is the sum of square of all absolute angles between strokes. It can also be understood as the sharpness of a sketch. It amplifies the sudden change in angle as it squares the angle. So if a shape has sharp corners, the total value of Total Squared Rotation will be higher than a shape with smooth corners.

Hence similarly,

$$Rf_{11} = \sum_{i=2}^{n-1} |\theta_i|^2 \tag{11}$$

Precaution: Finding the angle involves finding the *arctan* of the angle between the x-axis and the bounding box diagonal. However, the *arctan* is defined only within the 1st and 4th quadrant, i.e. $(-\pi/2, \pi/2)$. When consecutive coordinates have the same (x, y) coordinates, *arctan* faces issue. So, to make it defined in all the quadrant, i.e. $(-\pi, \pi)$, we have to use *atan*2, which is taken from the python in-built *math* library. *However, I never faced any such error during the assignment.*

- **Maximum Speed Squared:**
  Speed is the distance travelled in a defined time interval.
  Let's say, the points of a sketch are (x_0, y_0), (x_1, y_1), ... , (x_{n-1}, y_{n-1}) taken at t_0, t_1, ... , t_{n-1} timestamps respectively
  Then let,
  $\Delta y_i = y_i - y_{i-1}$
  $\Delta x_i = x_i - x_{i-1}$
  $\Delta t_i = t_i - t_{i-1}$

  Then, the speed between (x_{i-1}, y_{i-1}) and (x_i, y_i) is given by
  $v_i = \sqrt{(\Delta y_i^2 + \Delta x_i^2)}/\Delta t_i$
  So,

$$Rf_{12} = \max_{i=1}^{n-1} v_i^2 \tag{12}$$

  Precaution: The calculation of speed involves the difference in time in the denominator. With very high sampling rate, we can face that two points could be on the same time stamp. For such case, we need to ignore the point and move ahead with the next point. *In the given assignment, I faced this issue multiple times and have ignored such points where the timestamp was repeating.*

- **Total Time Taken:**
  Total time taken is simply the total time taken to draw the sketch</b> Let, the points of a sketch are (x_0, y_0), (x_1, y_1), ... , (x_{n-1}, y_{n-1}) taken at t_0, t_1, ... , t_{n-1} timestamps respectively

  So,

$$Rf_{13} = t_{n-1} - t_0 \tag{13}$$

5 Code Usage

- Install python==3.9
- Install pandas==1.4.2
- Install numpy==1.21.5
- data/, source/ and results/ folder to be kept under the root directory

- Run all the cells of RubineFeatures.ipynb. The features.csv will be created in the results/ folder.
- The code has been well documented in the jupyter notebook and every function has been explained.

6 Conclusion

The 13 Rubine Features are core to the features of a sketch recognition and many modern day technologies are either using Rubine features directly or have generated features manipulating the original Rubine Features. The implementation was done with caution to avoid any errors such as divide by zero, similar timestamp due to high sampling rate, taking care of domain of arctanθ etc. The given dataset was clean of any issue with respect to divide by zero and arctanθ errors, however there were similar timestamps for multiple coordinates for each sketch. The precaution has been taken for all type of commonly occurring errors.

Please find the link to Github.