Assignment 2 - Segmentation

Li-Hen Chen 928003907

Department of Computer Science & Engineering, Texas A&M Universit

Abstract – Freehand sketching is a natural and crucial part of everyday human interaction. In this assignment, we are required to perform corner finding algorithm in order to divide strokes into straight-line substrokes. At the beginning, I used the idea from Sezgin's paper, using the combination of curvature and speed to find the which point is the corner point. However, due to the sparse of sampling points, it turned out that speed makes no sense. As a result, I just used curvature to find out the segmentation points.

a. Explain your algorithm and discuss the intuition behind it.

a1.

At the beginning, I used the algorithm Sezgin developed. He used combinations of 3 indexes including direction, curvature and speed. In order to reduce the work and simplify the code, I only took 2 features including curvature and speed. They can be denoted as below formula.

$$\text{Let } \Delta x_p = x_{p+1} - x_p \text{ and Let } \Delta y_p = y_{p+1} - y_p$$

$$curvature = \theta_p = \arctan \frac{\Delta x_p \Delta y_{p-1} - \Delta x_{p-1} \Delta y_p}{\Delta x_p \Delta x_{p-1} + \Delta y_p \Delta y_{p-1}}$$

$$speed = \frac{\sqrt{(x_p - x_{p-1})^2 + (y_p - y_{p-1})^2}}{t_p - t_{p-1}}$$

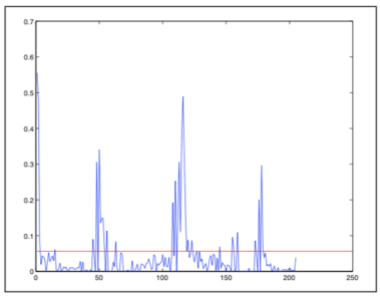
Obviously, using only curvature is not enough for gesture recognition Noise is one problem, but variety in angle changes is another. The algorithm solved this problem by incorporating speed data into our decision as an independent source of guidance. The intuitive using curvature and speed to determine whether a point is a corner is that when coming to the corner the curvature would suddenly increase and the speed would decrease relatively. However, setting threshold remains a significant issue in

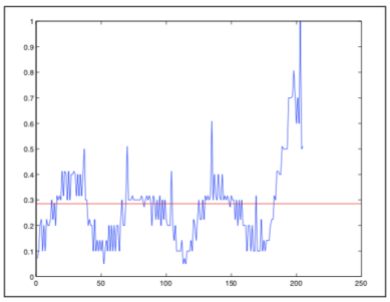
this case.

For the threshold for the curvature, the paper adopted average based filtering for curvature, which can automatically set a threshold without setting the threshold manually. Average based filtering performs better than simply comparing the curvature data against a hard coded thresh- old, it is still clearly not free of empirical constants. However, I don't think average based filter can really deal with the problem so I decided to find out the best curvature threshold manually.

For the threshold for the speed, I decided to adopt the idea presented in the paper setting the threshold of speed to 90% of the mean of speed.

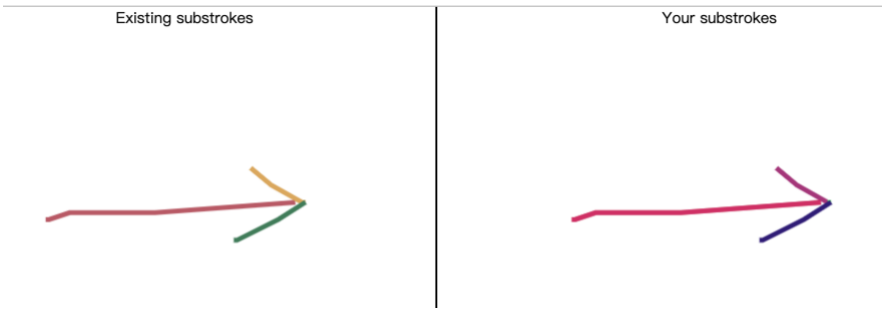
I would find the peak curvature where speed is under the threshold in each range. For example, I would take x-label 50 approximately as the segmentation point.

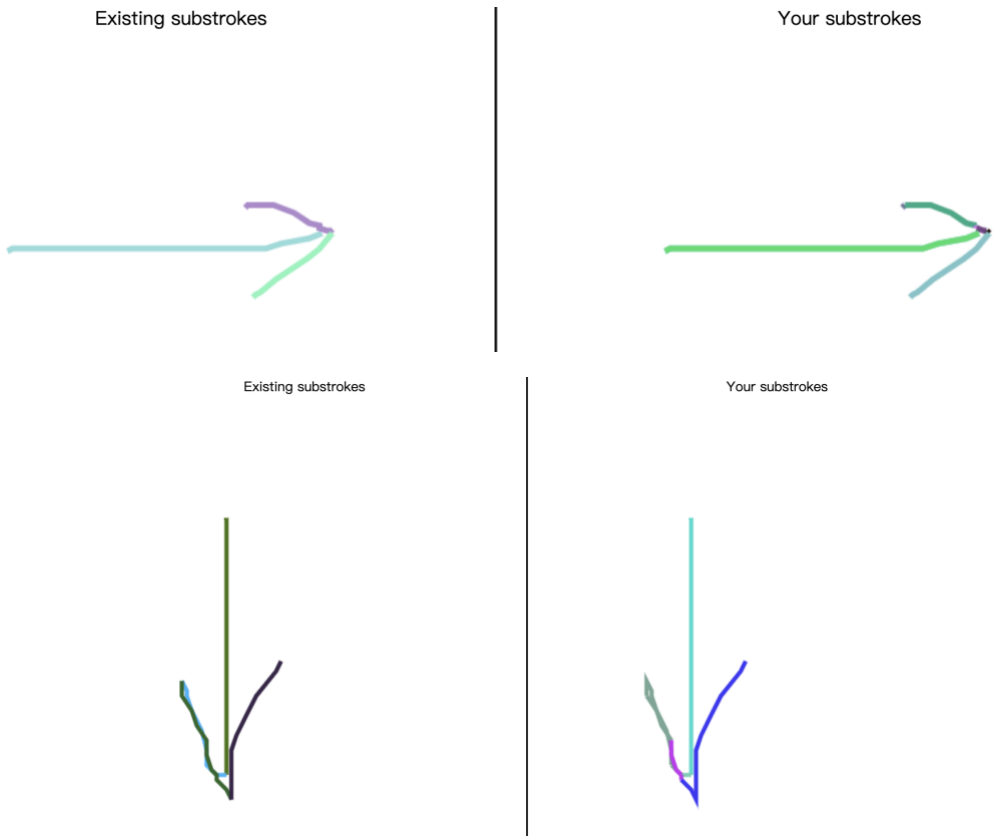




a2.

As mentioned above, due to the sparse of sampling points, it turned out that speed makes no sense. I decided to only use curvature and set a proper threshold manually. It turned out that results seemed pretty well. But in some dense cases, the algorithm would separate the line into too many segmentations. If I would like to improve the performance, I could try Kim's paper.





b. How might you improve your algorithm?

Unlike the paper I set the threshold manually and tuned it by trial and error. It may work well on this dataset because of trial and error but if we wish to apply the algorithm on another dataset, we better use auto threshold. However, setting threshold has been a critical issue for a long time, I think setting the threshold by the mean of data is not a proper way to decide threshold. If I would like to improve my algorithm, I have to work on threshold issue, which I think Otsu's method may help. Otsu's method is originally used to perform automatic image thresholding. In the simplest form, the algorithm returns a single intensity threshold that separate pixels into two classes, foreground and background. This threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance. In this case we can also use the same idea on our algorithm.

c. Compare and contrast the ShortStraw algorithm with Sezgin's method. Give a scenario for when you might choose one over the other

As the case above, Sezgin's method requires machine to record the points really dense, otherwise it would cause the speed of the points make no sense. On the contrary, ShortStraw algorithm resample the input data. Due to the fact, I would choose ShortStraw algorithm when the input points recorded by the machine are sparse.

On the other hand, when input points are sample densely, which means the input data can represent the sketch performed by the human well. In this case I would choose Sezgin's method because it may have higher accuracy which is the straightforward result caused by its robust method.