Chinese Character Writing Trajectory Optimization

James Guo
University of Washington
Department of Electrical Engineering
Seattle, WA, USA
jamesguo@uw.edu

I. Introduction

The *Autonomous Helicopter Aerobatics through Apprenticeship Learning Paper* used the apprenticeship learning method to generate autonomous stunt flying control based on data from human 'expert' trials. [1]

Particularly, the trajectory learning algorithm of the paper is a unique iterative process that generates a hidden 'true' trajectory based on the data/sensory readings of the multiple human 'expert' trials. An example of this is shown in Figure . above, where a double-loop stunt trajectory is generated from the multiple trials in the background.

Inspired by the trajectory learning algorithm, I decided to try and modify the iterative process of the algorithm to generate a 'true' user input trajectory (i.e. hand gestures, handwriting etc.) from the user's repeated demonstration.

For ease of implementation and to work on a relatively unique subject/dataset, I generated two-dimensional trajectories of writing Chinese characters, where data of me writing specific Chinese characters around 20 times per character were recorded and used.

As a result, trajectories consistent with the Chinese characters were able to be generated by this method, with varying degrees of character complexity, similar to the generated stunt trajectories of the paper.

II. Background

Chinese character writing at first glance can seem random for an unfamiliar individual. However, there are actual rules for writing of individual lines known as strokes, which dictates the order of the strokes and determines the overall form of a Chinese character. [2] An example of this is shown in Figure 1 below, where seemingly similar strokes in a character differ drastically and the writing order is unintuitive at first glance. I believe that these properties make for an interesting dataset - if implemented correctly many small quirks of the individual strokes and writing order can be reflected through the optimized 'hidden' trajectory generated.

Figure 1. An example of varying strokes and unintuitive writing orders of a Chinese character.

III. Implementation

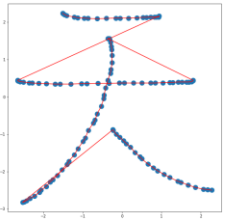

Figure 2. An example trajectory recording of obtained from the handwriting recorder

A. Handwriting Recorder in Unity engine

To generate my own handwriting data, I used the Unity game engine to build a custom writing recorder, with the software interface shown in Figure 3 below. The recorder would record the strokes on mouse click, render the drawn trajectories and parse and save the data of the individual completed trajectories. An example of one trajectory trial is shown in Figure 2 above, where the data is in the form of sequential (x,y) coordinates, which can be used to trace the stroke/writing trajectory of the character, which is outlined through the connected line in Figure 2.

Figure 3. The interface of the custom handwriting recorder.

Several optimizations for the software were implemented to reduce artificial noise/useless information collected by the software.

Firstly, the coordinates are only being recorded when actual writing occurs (where the user is holding down and dragging around the pen/mouse), essentially parsing the individual strokes together without recording the unnecessary time points when the pen leaves the surface to start a different stroke. In the example figure above, this can be observed from the trace of the trajectory, where the trace jumps immediately from the end of one stroke to the next in a straight line.

Secondly, only actual significant changes in the coordinates are recorded in the trajectory - if the user is holding down the mouse at a single pixel point through multiple time points, the extra identical coordinates are not recorded. The major benefit of this optimization is that the writing coordinates are smoothed out, which benefits both the

rendering of the writing in the software and the smoothness of the trajectory data recorded. The software is able to listen and execute code every 'rendering cycle' - which highly varies based on the computer used. Without this optimization, a poorly optimized rendering would likely produce repeated points that do not reflect the actual trajectory and use extra unnecessary resources from the computer to render.

B. Trajectory Learning Algorithm

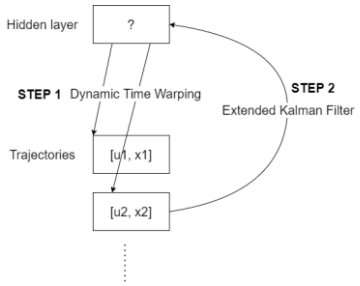

Figure 4. Iterative process flow chart of the trajectory learning algorithm. [1]

After gathering the data, I imported the data into a Python environment to process and implement the modified trajectory learning algorithm. A simple, abbreviated iterative process of the learning algorithm is shown in Figure 4 above.. Notably the iterative steps are:

1) Align the hidden trajectory and user trial using dynamic time warping

2) Generate new hidden trajectory through filtering methods (Extended Kalman(EK) filter)

3) Replace the hidden trajectory

4) Repeat for next user trial until convergence.

For my modified algorithm, the steps are identical except step 2, which I generated the new hidden trajectory by simply obtaining the element-wise mean of each time point. The EK filter was necessary to model non-linear dynamics needed for the highly variable state of helicopter stunts but unnecessary for simple two-dimensional trajectories of Chinese characters.

IV. RESULTS

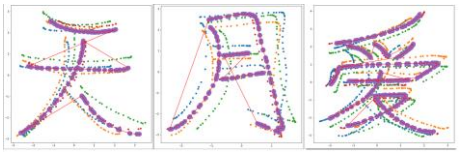

Figure 5. Generated trajectories over several individual trajectories of characters in ascending complexity – 'sky', 'moon', 'love'.

Figure 6. comparison of trajectory generated from the paper and my results. [1]

By utilizing the techniques shown above to collect and process my handwriting data, I was able to generate trajectories with the unique form and stroke of Chinese characters based on my handwriting. The method also proved effective for characters in varying complexities, as shown in Figrue 5 above.

Notability, like the method shown in the paper (with example in Figure 6 below) [1], the modified trajectory algorithm that I used was also noise resistant to individual trial trajectories, as shown in the comparison of one noisy trial of the 'moon' character compared to the generated trajectory in Figure 6 below.

Further information could be extracted from the trajectories. For example by taking a first order derivative, we can extract the pen velocity during specific strokes/parts of the trajectory. Extra information can be utilized to further establish a unique trajectory profile specific to individual users. These user-specific trajectories and information can potentially be used to optimize and build better decoding/recognition algorithms specific to the user for better performance.

REFERENCES

[1] Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous Helicopter Aerobatics through Apprenticeship Learning. The International Journal of Robotics Research, 29(13), 1608–1639. https://doi.org/10.1177/0278364910371999

[2] Yeromiyan, T. (2021, December 21). The Fundamentals of Chinese Stroke Order. The Chinese Language Institute. https://studycli.org/chinese-characters/chinese-stroke-order/