

Touchless Writer: Object Tracking & Neural Network Recognition

Yang Wu & Lu Yu

The Milton W. Holcombe Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
E-mail {wuyang, lyu}@clemson.edu

Abstract—Microsoft Touchless SDK introduces a new way of interacting with the computers by means of object tracking through webcams. Unlike other traditional input devices like mouse or keyboard, the input data from Touchless SDK (markers' position data) are usually unstable and inaccurate in nature, which limits the application of Touchless device as a replacement of the traditional input devices. In this paper, we explore a new way of utilizing the convenience of Touchless device by combining it with handwriting recognition Neural Network, so that the limitation of Touchless could be properly compensated. We will show in the later result that, without specialized device like lightpen or touchscreen, Touchless device provides pretty good performance of speed and accuracy in handwriting recognition.

Index Terms—Touchless SDK, Neural Network, Handwriting Recognition, Object Tracking, Pattern Matching, Image Processing

I. INTRODUCTION

THE Touchless SDK released by Microsoft provides the world an innovative way to interact with the computers. Touchless is an SDK that allows users to create and experience multi-touch applications. Touchless started as Mike Wassermans college project at Columbia University. The main idea: to offer users a new and cheap way of experiencing multi-touch capabilities, without the need of expensive hardware or software. All the user needs is a camera, which will track colored markers defined by the user.[1] By tracking multiple moving objects(markers) with a webcam, the Touchless SDK obtains a number of position data in realtime, which simulates a multi-touch positioning device.[2]

Though it shows great potential, there are certain limitations that prevent it from being a more powerful input device. The implementation of Touchless SDK is based on image processing: webcam continuously captures images processed by a color matching algorithm, thus the movement of color-marked object is analyzed and tracked. Being an assumption that the color of a certain object seen by webcam is recognizable and remains the same when the object moves, it is not always true in reality. Firstly, the color of the object must be contrasting to make it easily distinguished from the environment. Secondly, the color response is heavily influenced by the ambient light conditions. A slight change of environmental light makes significant change of the color captured by camera, thus introduce large error into the tracking algorithm. Furthermore, being a CPU time consuming algorithm, the performance relies greatly on the computing power of the system. As a result,

Touchless can not provide smooth movement and accurate positioning as other pointing devices. So proper application for Touchless may be those that are more tolerant to such inaccuracies.

Handwriting recognition well fits into this category. Handwriting recognition is the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation.[3] On the one hand, it is convenient and useful to write using a pen, track its movement through a camera, and transform the writing into digital document. On the other hand, Touchless works well for handwriting tracking because inaccuracy does not count that much in this—human handwriting, as well as many other human activities, is fundamentally not accurate.(Fig. 1) Not only do different people have different types of handwritings, even if for the same person, the script could be different in different time or when he/she is in different mood. Compared to the inaccuracy introduced by the above factors in handwriting, the inaccuracy of Touchless is not considerably significant.

Handwriting data is converted to digital form either by scanning the writing on paper or by writing with a special pen on an electronic surface such as a digitizer combined with a liquid crystal display. The two approaches are distinguished as off-line and on-line handwriting, respectively.[3] In our case, we choose on-line recognition in this case because Touchless tracks position data of object in real time and on-line recognition uses two-dimensional coordinates of successive points of the writing as a function of time. The data can then be processed by pre-trained Artificial Neural Network and recognized.

II. METHOD

In this section, we will discuss the methods we used for the Hardware Configuration, the System Architecture, the Software Specification and the Experiment Settings.



Fig. 1. Handwriting Samples

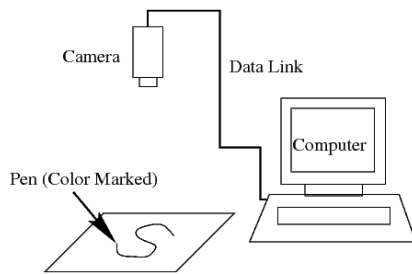


Fig. 2. The Hardware Configuration

A. Hardware Configuration

As is shown in Fig. 2, We are using the following hardware in the system:

- Laptop
- Camera
- Pure color paper
- Contrasting colored pen

When the application runs, the camera continuously captures images of the pen moving on the paper and send these images to the computer.

B. System Architecture

As is shown in Fig. 3, images captured by camera are passed to the image processing module, namely Touchless SDK in our case. Touchless processes these images, matches them with marked color pattern and translates the result into movement data. The movement data are then received by the pre-recognition processing module, where these data are preprocessed—smoothing, deduplicating, and normalization. The data are then translated into path information that indicates the strokes of the writing. Followed by that, the handwriting recognition module processes the path information and recognizes the character being written.

C. Software Specification

As mentioned previously, our application contains three submodules:

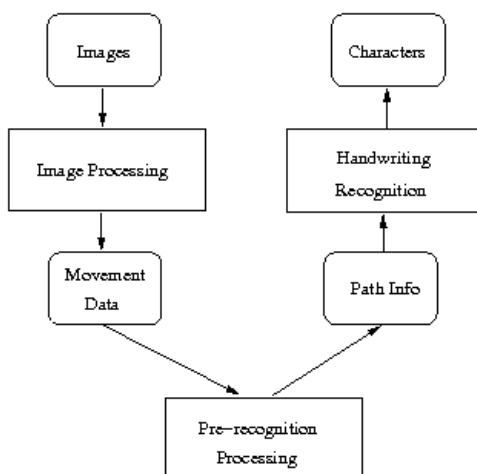


Fig. 3. The System Architecture

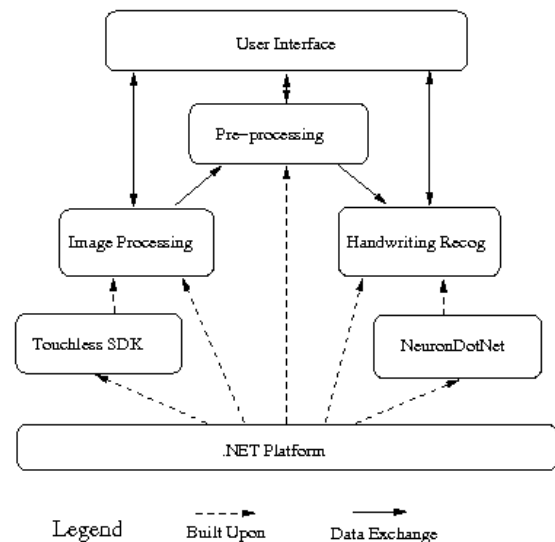


Fig. 4. The Software Specification

- Image Processing Module
- Pre-recognition Processing Module
- Handwriting Recognition Module

Their relation can be seen as in Fig. 4. The whole system is built upon Microsoft .NET Platform using Visual C# Express 2008.

1) *Touchless SDK*: The Microsoft Touchless SDK offers the ability to track multiple objects through cameras. In this project, we specifically track one object: the pen that we use to write, precisely, the pen nib. Note that we include a pure-colored paper in our hardware configuration. We do this to provide the maximal contrast for pen nib from the background. For Touchless to effectively track the pen nib, it is required that the color of the pen nib should be unique in the environment—to be different from the body of the pen, and contrast from the color of the paper. Though we use a paper as the background, we do not literally write on it, by moving the pen nib upon the paper, the movements are tracked by Touchless SDK.

In the experiment, we use a pen with green pen nib and a white paper as background. Once we mark the green pen nib as the target object, the position data of its movement is tracked automatically. As shown in Fig. 5, the square in black highlights the pen nib being tracked.

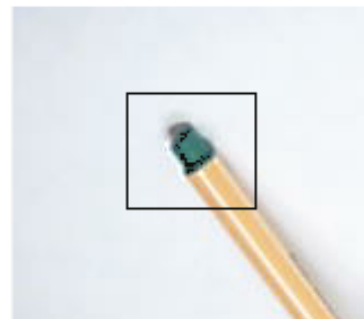


Fig. 5. Object Tracking In Action

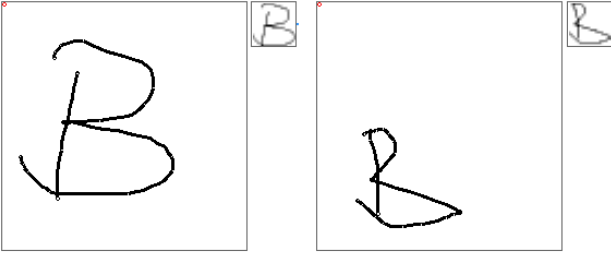


Fig. 6. Pre-recognition Processing

2) *NeuronDotNet*: Our handwriting recognition neural network is built upon NeuronDotNet, an open source engine which can be used to build different types of neural networks and use them in various applications.[4] It also provides API to extend existing features and incorporate new algorithms. It is written in C# and is compatible with the .NET platform.

3) *Pre-recognition Processing*: Before the path information obtained by Touchless SDK could be processed by neural network, they need to be preprocessed. Most digital tablets have low-pass hardware filters. Thus, when such devices are used to capture handwriting strokes, the shapes of the strokes present the jagged forms.[5] This is also the case for Touchless captured strokes. On the other hand, most handwriting strokes contain some hooks and duplicated sampled points caused by hesitated writing. Furthermore, some points of a stroke may be missing, and some wild points might exist in a stroke, etc. Generally, such noise information influences the exploration of the profile of the handwriting in such a way as to influence further processes, such as feature extraction and classification.[5] Three operations are involved here:

- Smoothing: Remove hooks and sharp points by down-sampling the stroke data.
- Deduplicate: Remove duplicated points from the strokes.
- Spatial Normalization: Scale the strokes into the same size.

The result of the above operations can be seen in Fig. 6.

4) *Handwriting Recognition Neural Network*: In this project, we use Back Propagation Neural Networks with one input layer, two hidden layers and one output layer for handwriting recognition.(Fig. 7) For better accuracy, we utilize one neural network for distinguish each pair of the character. Only the 26 upper-case Latin characters are recognized in the system. That is to say, we have $C_{26}^2 = 26 \times 25 \div 2 = 325$ networks for recognition. The result is obtained after 25 compares.

To train these networks, we use handwriting sample from NeuronDotNet that contains 185 different scripts for each upper-case characters. We train each network with these samples for 200 cycles so that the average mean square error of the networks can be reduced to as low as 0.0014.(As in Table I)

D. Experiment Settings

We conduct two groups of experiments in this research, the first of which is to examine the effect of training cycle to the

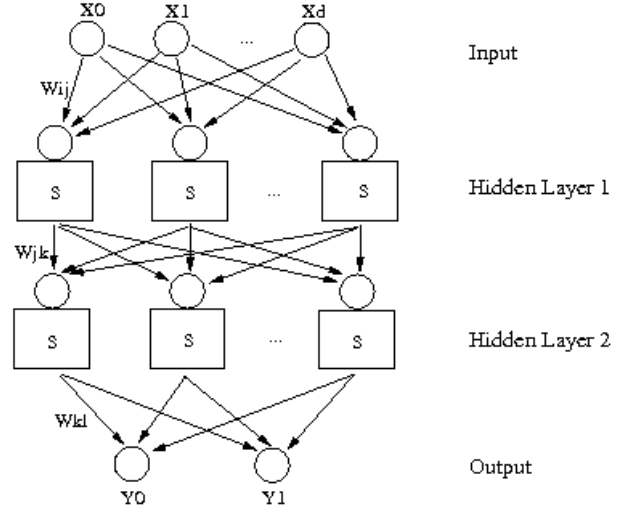


Fig. 7. A Sample Neural Network

network mean square error and training time. The second one is to test the performance of Touchless input in handwriting recognition with pre-trained neural network.

1) *Neural Network Training*: Given that the training samples are fixed, it is usually the case that larger amount of training cycles yield better mean square error, which possibly indicates better test result. However, more training cycles spend more training time as well. This is because for general training algorithm, majority of the training time is spent on propagating data along the connection between neurons, which determines the traversing time for each cycle.[6] We run the training algorithm upon the same training samples for different cycles and examine the relation between the number of cycles and the output mean square error.

2) *Handwriting Recognition*: With our pre-trained neural network, we compare the performance of Touchless and mouse as input device in handwriting recognition. The input sample sentence of the experiment is “THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG”, which is a pangram (a phrase that contains all of the letters of the alphabet). It has been used to test typewriters and computer keyboards, and in other applications involving all of the letters in the English alphabet, because it is also a short coherent sentence.[7] Each set of the experiment includes one person to draw each character of the sentence sequentially into the computer, first by using mouse, then by using pen under the camera(Touchless). Every drawing of character is followed by recognition the character. If the drawing fails to be recognized, the person need to re-draw the same character until the drawing can be recognized as the correct character. The whole process is timed, including the recognition time and the time for re-drawing. The error rate and other statistics are included as well. We have results from 4 different sets of experiments in this section.

III. EXPERIMENT RESULT

A. Neural Network Training

The raw result is stated in Table I. As is shown by Fig. 8, the training time increases as the number of training cycles goes

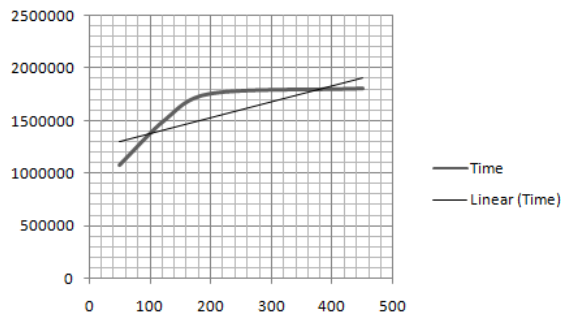


Fig. 8. Training Time

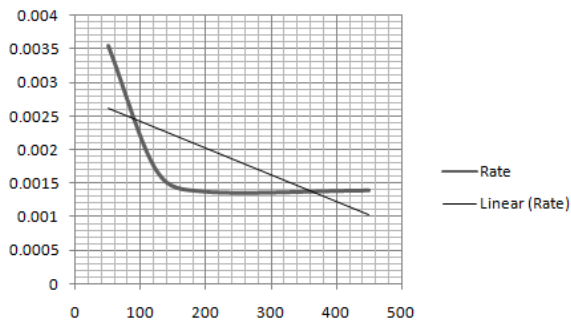


Fig. 9. Training Result

up. We can see that the time curve climbs really fast before 200 cycles, then it comes to an inflection point and remains steady. As is shown by Fig. 9, the error rate decreases as the number of training cycles increases and the curve becomes steady after 200 cycles. To gain best performance, we choose 200 training cycles in our experiment.

B. Handwriting Recognition

As is shown in Table II and Table III, the error rate and unique errors of Touchless and mouse are pretty close, and Touchless does a slight better job. This means that by utilizing Touchless in handwriting recognition with neural network, its inaccuracy is well compensated, thus the performance is not affected by this factor at all.

However, the time consumed by using mouse is considerably less than by using Touchless. The first reason is that by doing handwriting with input device, you will need to simulate stroke-down and stroke-up. This could easily accomplished by pressing and releasing mouse button. However, with Touchless, there is no convenient way of doing so in this experiment (we use keyboard press to simulate this), hence certain time is wasted in this. Secondly, mouse positioning is done mostly by its hardware and is fast. Touchless on the other hand, depends greatly on the performance of its matching algorithm running on the system, and the capture rate of the camera is quite limited in this experiment, thus obtaining data from Touchless could be slower than mouse.

IV. CONCLUSION

In this research, we are able to show the powerful ability of Touchless SDK and how it allows multi-touch like experience without requiring expensive hardware and software. By analyzing and understanding the advantages as well as the limitations of Touchless SDK, we explore a field in which Touchless has the potential to gain massive success. The use of neural network works pretty good in avoiding the errors introduced by the inaccuracy of Touchless, and careful tuning of the network gives us good performance in handwriting recognition.

Due to the fundamental limitation of Touchless, we observe in the experiment that Touchless can not compete with other input device in speed when doing handwriting recognition. Two solutions can actually help on this defect:

- 1) By utilizing specialized cameras that are sensitive to colors and can react quickly to color variations, the performance of Touchless is believed to boost due to increased capture rate.
- 2) By implementing more effective color matching algorithm on hardware like FPGA&CPLD, the response time of Touchless will be significantly reduced.

Last but not least, we would like to thank Dr. Stan Birchfield for the wonderful lecture, and every classmate from ECE847 for all the help—we had great fun this year.

REFERENCES

- [1] Microsoft, Internet. [Online]. Available: <http://www.officelabs.com/projects/touchless/Pages/default.aspx>
- [2] T. Ishikawa, Y. Horry, and T. Hoshino, "Touchless input device and gesture commands," *Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers. International Conference on*, pp. 205–206, Jan. 2005.
- [3] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 63–84, Jan 2000.
- [4] "Neurondotnet," Internet. [Online]. Available: <http://neurondotnet.freehostia.com/index.html>
- [5] B. Huang, Y. Zhang, and M.-T. Kechadi, "Preprocessing techniques for online handwriting recognition," *Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on*, pp. 793–800, Oct. 2007.
- [6] L. Prechelt and F. F. Informatik, "Proben1 - a set of neural network benchmark problems and benchmarking rules," Tech. Rep., 1994.
- [7] "The quick brown fox jumps over the lazy dog," Internet. [Online]. Available: http://en.wikipedia.org/wiki/The_quick_brown_fox_jumps_over_the_lazy_dog

Count	Time(ticks)	Min	Max	Average
50	1078498	0.00065	0.05823	0.00355
125	1510448	0.00049	0.02793	0.00167
200	1759973	0.00031	0.01671	0.00137
450	1808239	0.00024	0.02153	0.00139

TABLE I
NEURAL NETWORK TRAINING

Set#	Valid Inputs	Total Inputs	Unique Errors	Error Rate	Time(min)
1	35	44	9	20.5%	11
2	35	43	5	18.6%	7
3	35	49	9	28.6%	6
4	35	50	9	30.0%	6
Average	35	46.5	8	24.4%	7.5

TABLE II
TOUCHLESS HANDWRITING RECOGNITION

Set#	Valid Inputs	Total Inputs	Unique Errors	Error Rate	Time(min)
1	35	46	6	23.9%	8
2	35	48	10	27.0%	4
3	35	44	7	20.5%	3
4	35	50	9	30.0%	3
Average	35	47	8	25.35%	4.5

TABLE III
MOUSE HANDWRITING RECOGNITION