Capturing Handwritten Ink Strokes with a Fast Video Camera

Chelhwon Kim
FX Palo Alto Laboratory
Palo Alto, CA USA
kim@fxpal.com

Patrick Chiu
FX Palo Alto Laboratory
Palo Alto, CA USA
chiu@fxpal.com

Hideto Oda
FX Palo Alto Laboratory
Palo Alto, CA USA
oda@fxpal.com

***Abstract*—We present a system for capturing ink strokes written with ordinary pen and paper using a fast camera with a frame rate comparable to a stylus digitizer. From the video frames, ink strokes are extracted and used as input to an online handwriting recognition engine. A key component in our system is a pen up/down detection model for detecting the contact of the pen-tip with the paper in the video frames. The proposed model consists of feature representation with convolutional neural networks and classification with a recurrent neural network. We also use a high speed tracker with kernelized correlation filters to track the pen-tip. For training and evaluation, we collected labeled video data of users writing English and Japanese phrases from public datasets, and we report on character accuracy scores for different frame rates in the two languages.**

I. Introduction

Our goal is to develop effective methods for using a fast camera to capture handwriting with ordinary pen and paper at sufficiently high quality for online handwriting recognition. "Online" means having the temporal ink stroke data, as opposed to "offline" where we only have a static image of the handwriting. Online recognition can perform better than offline. This is especially important for Japanese and other Asian languages in which the stroke order of a character matters. With the recognized text data, there are many possible applications including indexing & search systems, language translation and remote collaboration.

While there exist commercial products that can record ink strokes, they require special pens and in some cases paper with printed patterns. Examples are: Livescribe Pen [1], Denshi-Pen [2], and Bamboo Spark [3]. These can be useful for vertical applications such as filling out forms, but for general usage it would be advantageous to be able to use ordinary pen and paper.

Previous research on using a video camera to capture ink strokes written with pen and paper include [4]–[8]. These systems have frame rates of up to 60 Hz. In comparison, a stylus digitizer can run at 133 Hz (e.g. Wacom Intuous Pen Tablet [3]). Using a high frame rate camera (Point Grey Grasshopper at 163 Hz [9]) that exceeds the above devices, we investigate ink stroke capture for online handwriting recognition for English and Japanese.

II. Related Work

Anoto Livescribe Pen [1] and Fuji Xerox Denshi-Pen [2] use special pen and paper with printed markings to track and capture ink strokes. Wacom Bamboo Spark [3] uses a special pen with ordinary paper placed on top of a tablet which senses the pen location.

The system by Munich & Perona ([4], [5]) uses a video camera with 60 Hz (30 Hz interlaced) and resolution of 640 x 480. The pen up/down detection uses a Hidden Markov Model (HMM) with the ink absence confidence measure based on brightness of the detected pen tip's surrounding pixels. The pen tip initialization is semi-automatic, in which the user places pen tip inside a display box to acquire the pen tip template. The tracking is an ad hoc method using Kalman filter prediction, template matching on each frame and fine localization of the ballpoint by edge detection. The system was tested for signature verification application, but not for handwriting recognition.

Fink et al. [6] developed a system that is similar to [4] and supports handwriting recognition by integrating a Hidden Markov Model (HMM) recognizer. It was trained and tested on handwritten names of German cities. It used a camera with 50 Hz and resolution of 768 x 288 pixels.

The work by Seok et al. [7] is also similar to [4] with its Kalman filter pen tip tracker. It has a step to perform global pen up/down correction by segmenting the trajectory into pieces by high curvature points and classifying them on features based on length, continuity, and ratio of nearby written ink pixels. An evaluation was done on pen up/down classification performance.

Bunke et al. [8] is another system that uses a camera with 19 Hz and resolution of 284 x 288 pixels. It uses ink traces to reconstruct the stroke by image differencing consecutive frames. To deal with occlusion and shadows, it examines aggregated subsequences of frames and the last frame with the complete ink traces. However, there are still unresolved problems with low contrast regions. Experiments were performed with an existing handwriting recognition algorithm using a small set of collected data for training and testing.

The system by Chikano et al. [10] uses a camera (30 Hz) attached to a pen and relies on the "paper fingerprint" (image features caused by the uneven paper surface) and the printed text background for tracking. It does not handle pen up/down detection and works only for single stroke words and annotations. It showed that the Lucas-Kanade tracker [Lucas et al., 1981] performed better than a SURF feature tracker in

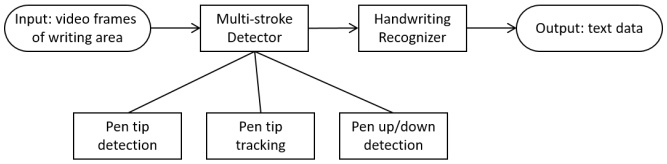


Fig. 1: System overview.

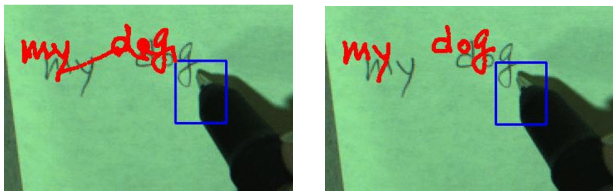


Fig. 2: Left: Trajectory of top-left corner of the pen tip region (blue rectangle) obtained by the KCF tracker. Right: Pen down strokes only.

the recovery of the ink trajectories.

In comparison to these systems, our method employs more recent algorithms. For pen up/down detection, we use a deep learning neural network which will be explained in detail below. For our tracking method, we use the more recent KCF tracker that has been shown to perform well against state-of-the-art tracking algorithms [11]. Furthermore, we investigate using our system with a high frame rate camera, and conducted an evaluation with a handwriting recognition engine on English and Japanese at different frame rates.

III. Ink Stroke Capture

We use a high-speed camera (Point Grey Grasshopper, 163 Hz) mounted above a desk to capture handwriting with pen and paper. An overview of the processing pipeline is shown in Fig. 1. The video frames are processed in a Multi-stroke Detector module to obtain ink stroke data, which provides the input to an online handwriting recognizer engine (MyScript [12]) that outputs text data. The Multi-stroke Detector module consists of several submodules that we explain in detail in the following sections.

A. Pen Tip Detection

The pen tip location is required to initialize our pen tip tracker. One method to accomplish the pen tip detection is to use template matching, which is a common technique that has been used for detecting pen tips [5]. It is possible to use modern object detection methods based on convolutional neural networks [13]–[16], which have been shown to perform fast and accurate object detection over 20 categories. In our processing pipeline, we have not yet implemented pen tip detection, and we currently mark the pen tip manually.

B. Pen Tip Tracking

Our pen tip tracker employs a high-speed tracker with kernelized correlation filters: the KCF tracker [11]. For initialization, a region containing the pen tip is required, which we mark manually as noted above. Then the KCF tracker is applied to learn a classifier to discriminate appearance of the pen tip and of the surrounding background. The classifier is efficiently evaluated at many locations in the proximity of the pen tip to detect it in subsequent frames, and the classifier is updated using the new detection result [11]. The KCF tracker shows stable performance in tracking normally paced pen tip motion in a video. Fig. 2 (Left) shows a trajectory of the top-left corner of the pen tip region over the ink traces.

C. Pen Up/Down Detection

To extract the ink strokes, the pen up parts of the pen tip trajectory must be removed. This requires pen up/down detection to determine, at every point in a video, whether the pen is in contact with the paper, and thus writing, or the pen is lifted. See Fig. 2 (Right).

The pen up/down detection is a challenging problem. The camera from above cannot accurately see the height of the pen tip. However, when the pen is in contact with the paper, there are ink traces. The difficulty is that sometimes when writing, the pen occludes the traces.

To address this problem, we use a deep learning neural network, which recently has had great success for pattern recognition problems involving images and video [17]. For pen up/down detection, intuitively we suppose that humans can perceive the pen up/down motion accurately at every point in a video based on what has been written on the paper (i.e. ink traces) and how the pen-tip has been moved in a short period of time. We model this human perception with a recurrent neural network (RNN). RNNs can process inputs sequentially and capture information from the previous inputs in their internal memory and predict the next event based on it. In our case, we process sequential image frames of a handwriting video through a recurrent neural network which outputs probabilities of them being a pen down state based on information in the RNN's internal memory. When each video frame is processed by the RNN, features are extracted from ink traces in a region around the pen tip location.

The last image of a sequence can also be used to obtain additional information about the ink trace, as has been observed in [8]. We use the last image of a sequence to extract features from the complete ink traces in the sequential frames by taking the difference between ink traces at the current image frame and at the last frame (see the two small patches in Fig. 3). The feature extraction is performed by convolutional neural networks that are pre-trained effectively to extract optimal features for the pen up/down detection task. This learning based feature extraction relieves the burden of feature design.

Next, we describe the detailed process flow and the architecture of the proposed neural network. Our neural network model consists of two parts: feature representation and classification (see Fig. 3). The feature representation part comprises convolutional neural networks (Conv); at each time step t, our model extracts patches around the pen-tip at $\mathbf{l}_t$ from the current image frame I_t and the last image frame I_T, where $\mathbf{l}_t$ is coordinates

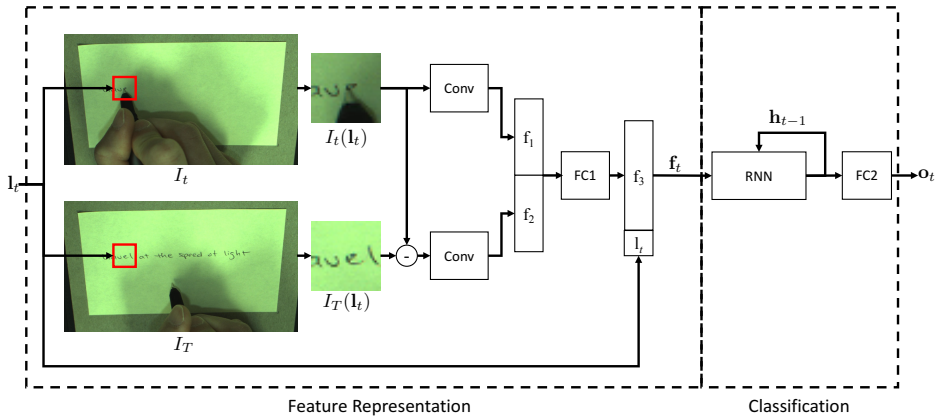


Fig. 3: The architecture of the proposed neural network for pen up/down detection. The network comprises convolutional neural networks (Conv), fully connected networks (FC), and a recurrent neural network (RNN). See text and Table I for more details.

TABLE I: Detailed configuration of the proposed network with number of feature maps or hidden state vector size (n), kernel size (k), stride (s), padding size (p), and dropout rate (d).

Conv
Convolution(n32 k5x5 s1 p0) → Batch Normalization → ReLU → MaxPool(k3x3 s3 p1) → Convolution(n64 k5x5 s1 p0) → Batch Normalization → ReLU → MaxPool(k2x2 s2 p0)
FC1
Dropout(d0.5) → Linear(n126) → Tanh
FC2
Linear(n1) → Sigmoid
RNN
GRU(n128 d0.5)

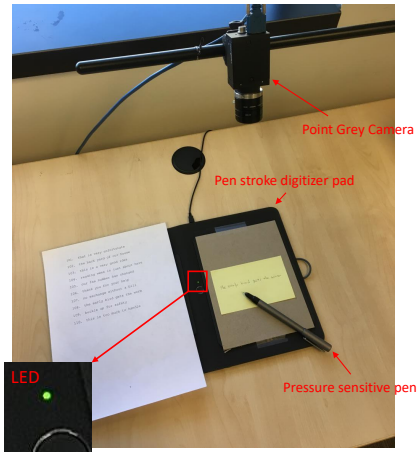


Fig. 4: System for collecting labeled data.

of the pen-tip obtained by the pen tip tracker. We denote those patches by $I_t(\mathbf{l}_t)$ and $I_T(\mathbf{l}_t)$ respectively. $I_t(\mathbf{l}_t)$ and difference between $I_t(\mathbf{l}_t)$ and $I_T(\mathbf{l}_t)$ are sent to two independent convolutional networks and transformed into feature vectors $\mathbf{f}_1$, $\mathbf{f}_2$. These two feature vectors are concatenated into one vector that is sent to a fully connected network (FC1). The output of FC1 $\mathbf{f}_3$ and the pen-tip location $\mathbf{l}_t$ are concatenated into one feature vector $\mathbf{f}_t$.

The classification part is a recurrent neural network; the feature vector $\mathbf{f}_t$ is send to the RNN with its previous hidden state vector $\mathbf{h}_{t-1}$, and the updated hidden state vector $\mathbf{h}_t$ of the RNN is sent to a fully connected network (FC2) that outputs a probability $\mathbf{o}_t$ of the pen down state for the current image frame. Our convolutional network block (Conv) is inspired by [18], [19]. We use two convolutional layers with 5 x 5 kernel size, each of which is followed by batch normalization [20], ReLU, and MaxPooling layer. For RNN, we use one of the popular variants of RNN: Gated Recurrent Unit (GRU) [21] with dropout rate 0.5. Detailed configuration of each component of our model is in Table I. We use Torch [22] to implement the proposed neural network.

IV. Collecting Labeled Data

We collected handwriting data for English and Japanese. A task consisted of a user writing a phrase. The English phrases are taken from a public dataset of phrases for evaluating text input [23]. The Japanese phrases are from a public corpus [24], with phrases taken from the titles of the "culture" topics that are 3 to 7 characters long.

For each language, 10 users performed 10 tasks of writing a phrase. A total of 20 users participated. All the users were right-handed.

Each handwriting task was recorded with a high frame rate camera mounted above a desk. The camera used was a Point Grey Grasshopper 3 (163 Hz, 1920 x 1200 pixels, global shutter) [9]; in our setup the actual frame rate was 162 Hz. Users were asked to write each phrase on a single line on a single blank Post-it note (3x5 inches). Examples of handwritten phrases are shown in Fig. 6.

To obtain the ground truth labels for the pen up/down states, we used the Wacom Bamboo Spark [3] device which has a special pen that writes ink on ordinary paper placed on a digitizer pad. In our setup, we put a single Post-It note on

the digitizer to collect handwriting data for each phrase. See Fig. 4. This device has an LED on the side (see the zoomed image in Fig. 4) which flashes when a pressure sensor inside the pen is activated by pressing its pen-tip on the surface. We utilize this LED as an indicator of pen up/down states and developed a simple image processing algorithm that checks the brightness of the LED in the video images and automatically assigns pen up/down labels to all the video frames. Note that pen stroke data recorded by the digitizer pad is not used for training our neural network.

V. Training

We train our network on a GPU (Nvidia GTX 1070) using our labeled handwriting video dataset. All handwriting video frames are decomposed into sub-segments with a stride of 25. We set the size of each sub-segment to 150 frames which is around 1 second for 162Hz frame rate. This ensures that each sub-segment of video contains at least $1 \sim 2$ strokes for English, $2 \sim 3$ strokes for Japanese. For each mini-batch, we use 10 consecutive sub-segments for training our neural network.

We extract two 100 x 100 size of image patches at the pen-tip location $\mathbf{l}_t$ from both the current image frame of each sub-segment and the last frame of the corresponding video from where the sub-segment is sampled (i.e. $I_t(\mathbf{l}_t)$ and $I_T(\mathbf{l}_t)$ respectively. See Sec. III-C). These patches are down-sampled to 32 x 32 by bi-cubic interpolation before they are sent to the network. All patches are converted to gray images and their pixel values are normalized to zero mean and unit standard variance. The coordinates of the pen-tip location $\mathbf{l}_t$ are normalized to $0 \sim 1$ range by dividing them with the width and the height of the video frame which are 1920, 1200 respectively.

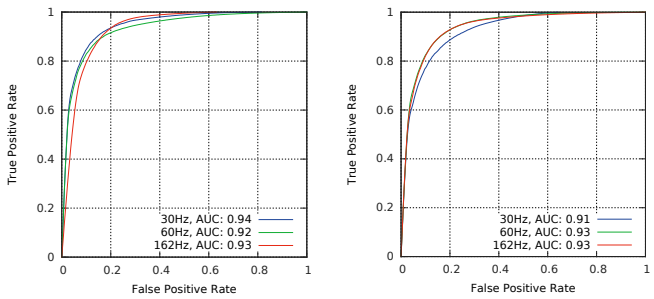
For optimization, we use Adam [25] with 0.9 of first moment coefficient and 1e-4 of learning rate for the first 30 epochs and 1e-5 of learning rate for the rest epochs. We use a loss function that measures the Binary Cross Entropy between the target and the output. We observed that the optimization converges within 50 epochs and we stop training at 50 epochs. The whole training session takes roughly 10 hours on a single GPU.

VI. Evaluation

A. Quantitative/qualitative evaluation result

We performed quantitative assessment of our multi-stroke detector using 5-fold cross validation. The dataset is randomly partitioned into 5 equal sized sub-samples (i.e. 20 videos per sub-sample) and each sub-sample is used for testing for each round. For training, each video is decomposed into sub-segments with a stride of 25, and the 80 training videos provide around 8,000 sub-segments to train the proposed neural net. For testing, each video in the test set is processed by the trained neural net, and all the video frames are assigned to pen up/down states by thresholding the network's outputs.

We chose a receiver operation characteristic (ROC) curve for our evaluation. ROC can be obtained by computing the true positive and false positive rate for all thresholds. Fig. 5 shows ROC curves of our pen up/down detection for English (left) and Japanese (right) phrases. When we compute the true positive rate and the false positive rate all the test results from the 5-fold cross validation rounds are considered. We also compute area under the ROC curve (AUC) and our detector achieves 0.93 on 162Hz handwriting videos for both English and Japanese phrases.



English phrases Japanese phrases

Fig. 5: ROC curves and area under the ROC curve (AUC) of the proposed pen up/down detection on 30, 60, 162Hz handwriting videos for English phrases (left) and Japanese phrases (right).

In Fig. 6, we show qualitative examples of the results. For each panel, we show ink strokes of the handwritten phrase, the pen-tip tracker trajectory, the pen-down strokes detected by our method, and the handwriting recognition result. Overall our pen-down strokes are similar to the ink strokes, but in some cases our method fails to detect pen-up states between letters such as "protect" in the fourth English phrase and "reading week" in the fifth English phrase. Although the pen-up/down detection for Japanese is more difficult due to its complex character structure, the proposed neural network detects most of the ink strokes well. We also observed that some of Japanese characters are over-segmented by the handwriting recognition engine and recognized as multiple characters. For example, 明 is recognized as 回 and 目 in the fourth Japanese phrase. In some cases, our detector fails to reconstruct all strokes in Japanese characters (e.g. 歴 and 景 in the fifth Japanese phrase).

Quantitative assessment of handwriting recognition based on our detected pen-down strokes is also performed. We threshold the network's output at 0.5 to get only the pen-down strokes. The MyScript [12] handwriting recognition engine is used to convert these strokes to text. Then we compute the *character accuracy score*, which is 1.0 minus the edit-distance (normalized for length). Our proposed method at 162 Hz achieved scores of 0.880 for English and 0.821 for Japanese. See Fig. 7.

B. Performance at different frame rates

We investigated the effect of the video frame rate on the proposed pen-up/down detection and the handwriting recognition performance. To this end, we made synthesized 30 Hz and 60 Hz videos by downsampling 162 Hz videos with a stride of 5 and 3 frames respectively. Note that we do not apply the

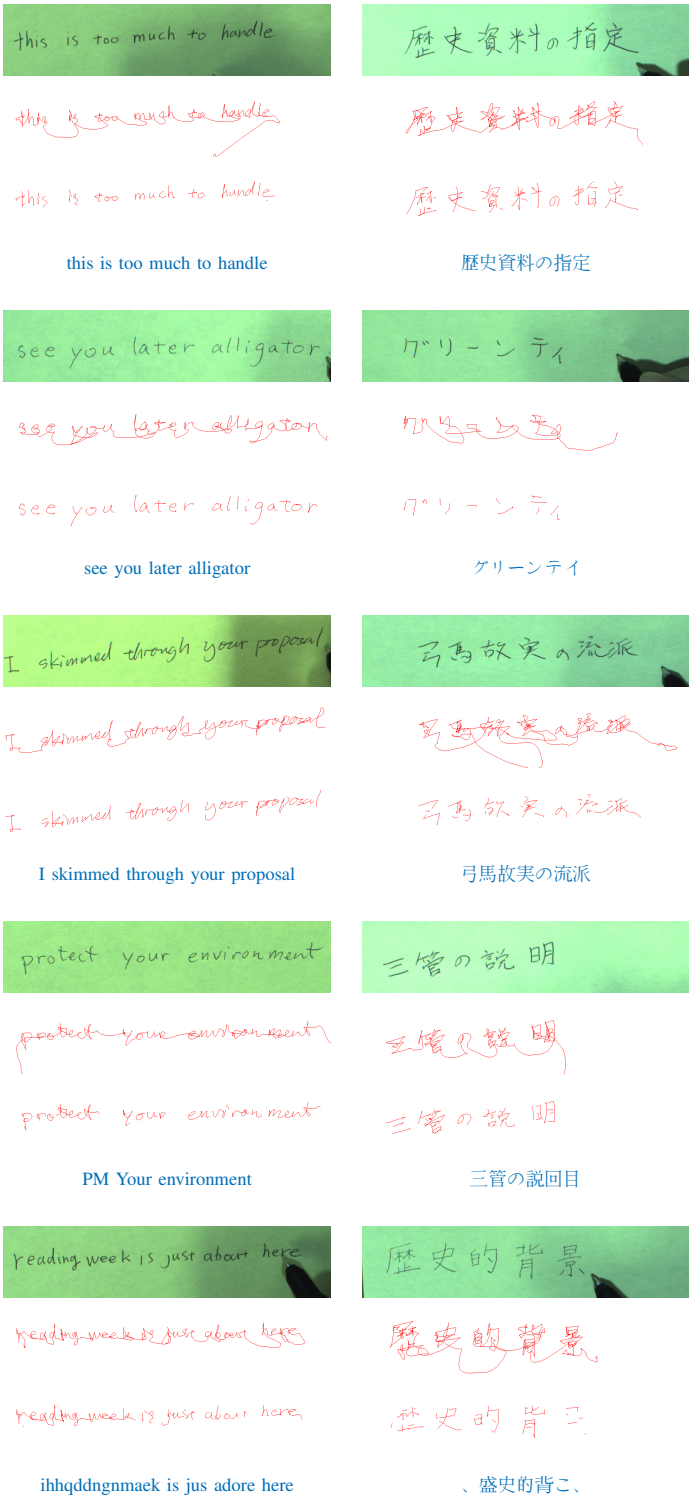


Fig. 6: Left column: English phrases, Right column: Japanese phrases. For each panel and from top to bottom; ink strokes of handwritten phrase, the pen-tip tracker trajectory, the pen-down strokes detected by our method, and handwriting recognition result using the pen-down strokes.

pen-tip tracker directly to the downsampled videos. Instead, we used downsampled pen-tip tracking results (i.e. the pen-tip locations) of 162 Hz videos with the same strides. This leads

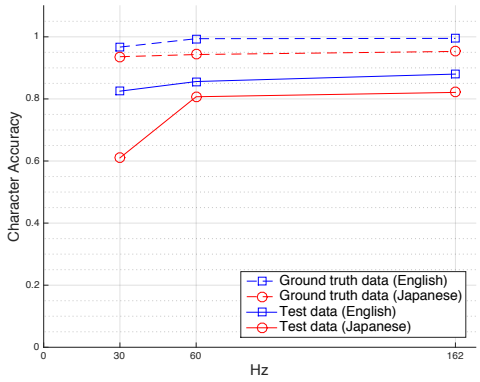


Fig. 7: The character accuracy scores of the handwriting recognition results considered with 30 Hz, 60 Hz, and 162 Hz video frame rate. Test data (solid lines): English {0.825, 0.856, 0.880} and Japanese {0.610, 0.807, 0.821}. Ground truth data (dashed lines): English {0.967, 0.994, 0.995} and Japanese {0.936, 0.943, 0.953}.

to consistent tracking results even for the low frame rates so as to be fair across all the frame rates and does not consider the frame rate effects on the pen-tip tracking, but only on the pen-up/down detection.

We trained separate neural networks with the downsampled 30 Hz and 60 Hz videos which are decomposed into sub-segments with a stride of 5 and 9 frames, where the size of each sub-segment is 28 and 56 frames respectively. This ensures that each sub-segment lasts the same amount of time across the different frame rates. The other parameters for training and evaluation are same as in Sec. VI.

Examples of reconstructed strokes from videos with different frame rates are depicted in Fig. 8. For each panel, we show the ink stroke image and the detected pen-down strokes for 30 Hz, 60 Hz, and 162 Hz frame rate. Adjacent strokes are distinguished by different colors. There are small dots on the strokes [1], which represent the locations of the pen-tip classified as the pen-down state. Overall, the reconstructed ink strokes for 60 Hz and 162 Hz showed smoother and more complete shape of strokes than 30 Hz.

For quantitative performance analysis, we plot the character accuracy scores for the strokes resulting from the proposed pen up/down detection and from the ground truth pen up/down data against the frame rate. See Fig. 7. As one might expect, Japanese showed lower scores than English due to its more complex character structure. Moreover, compared to English the Japanese strokes are shorter and written faster (see Table II) and thus will have relatively lower scores at low frame rates insufficient to sample them.

The performance on the ground truth pen up/down data shows high accuracy and stable results over all frame rates and slightly decreases as the frame rate decreases. See the dashed lines in Fig. 7. The high accuracy (0.953 and 0.995) indicates that the KCF tracker performed well. From 162 to 60 Hz, for both languages the drop off is small. From 60 to

[1]The strokes and small dots can be seen more clearly by zooming in on a digital version of this document.

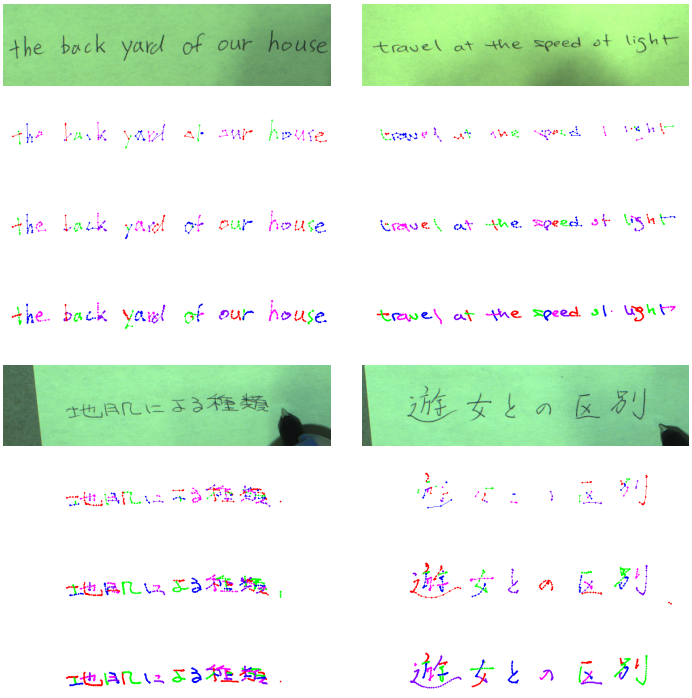


Fig. 8: Reconstructed ink strokes for different frame rates. For each panel and from top to bottom, the ink stroke image, detected pen-down strokes for 30, 60, and 162 Hz frame rate. See text for details.

30 Hz, for English there is a slight drop off (2.7%), and for Japanese the difference is smaller (0.7%).

The performance on the strokes detected by our method shows decreases as the frame rate decreases. From 162 to 60 Hz, for English there is a slight drop off (2.4%), and for Japanese the difference is smaller (1.0%). From 60 to 30 Hz, the drop off is more substantial and it drops off much more for Japanese (see Fig. 7). The drop off *between* languages is much greater with our method than with the ground truth data. A possible factor is that our method is based on video which has occlusions (unlike on a digitizer), and this effectively reduces the overall frame rate since at the occluded time intervals useful data cannot be sampled. Another factor is that English is written from left to right and this leads to relatively less occlusion than with Japanese where more back-and-forth pen movement is required to form a character.

TABLE II: Stroke Properties. These are computed from the ground truth pen up/down data and our tracker data.

	stroke length (px)	time per stroke (sec)
English	50.9	0.236
Japanese	46.5	0.195

VII. Conclusion

In this paper, we presented a system for capturing ink strokes written with ordinary pen and paper using a high frame rate video camera. We collected a labeled video dataset for handwriting in English and Japanese, and experiments demonstrate that the proposed system achieves a high degree of accuracy for the pen tip tracking and the pen up/down detection. Our results show that handwriting recognition character accuracy drops off slightly from 162 to 60 Hz and more drastically from 60 to 30 Hz. Comparison of the performance from our pen up/down detection method and ground truth pen up/down data between the two languages indicates some issues about occlusion in video based systems and in the way the languages are written.

References

[1] "Anoto Livescribe Pen." [Online]. Available: https://www.livescribe.com

[2] "Fuji Xerox Denshi-Pen." [Online]. Available: https://www.fujixerox.co.jp/product/stationery/denshi-pen

[3] "Wacom Technology Corporation." [Online]. Available: http://www.wacom.com

[4] M. E. Munich and P. Perona, "Visual input for pen-based computers," in *Proc. ICPR 1996*, pp. 33–37.

[5] ——, "Visual input for pen-based computers," *TPAMI*, vol. 24, no. 3, pp. 313–328, 2002.

[6] G. A. Fink, M. Wienecke, and G. Sagerer, "Video-based on-line handwriting recognition," in *Proc. ICDAR 2001*, pp. 226–230.

[7] J.-H. Seok, S. Levasseur, K.-E. Kim, and J. Kim, "Tracing handwriting on paper document under video camera," in *ICFHR 2008*.

[8] H. Bunke, T. Von Siebenthal, T. Yamasaki, and M. Schenkel, "Online handwriting data acquisition using a video camera," in *Proc. ICDAR 1999*, pp. 573–576.

[9] "Point Grey cameras." [Online]. Available: http://www.ptgrey.com

[10] M. Chikano, K. Kise, M. Iwamura, S. Uchida, and S. Omachi, "Recovery and localization of handwritings by a camera-pen based on tracking and document image retrieval," *Pattern Recognition Letters*, vol. 35, pp. 214–224, 2014.

[11] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *TPAMI*, vol. 37, no. 3, pp. 583–596, 2015.

[12] "MyScript handwriting recognition engine (v7.2.1)." [Online]. Available: http://www.myscript.com

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[14] Y. Li, K. He, J. Sun *et al.*, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR 2016*, pp. 779–788.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[18] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. ECCV 2016*, pp. 694–711.

[19] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[22] "Torch." [Online]. Available: http://torch.ch

[23] I. MacKenzie and R. Soukoreff, "Phrase sets for evaluating text entry techniques," in *CHI 2003 Extended Abstracts*, pp. 754–755.

[24] "NICT Corpus: Japanese-English bilingual corpus of Wikipedia's Kyoto articles, (version 2.01, 2011)." [Online]. Available: https://alaginrc.nict.go.jp/WikiCorpus/index_E.html

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.