WALAILAK JOURNAL
Applied Sciences
http://wjst.wu.ac.th https://doi.org/10.48048/wjst.2021.11233

Automatic Thai Finger Spelling Transcription

Pisit NAKJAI[1] and Tatpong KATANYUKUL[2,*]

[1]*Department of Computer Science, Uttaradit Rajabhat University, Uttaradit 53000, Thailand*
[2]*Department of Computer Engineering, Khon Kaen University, Khon Kaen 40002, Thailand*

(Corresponding authors' e-mails: tatpong@kku.ac.th)

Received: 19 July 2020, Revised: 1 January 2021, Accepted: 7 January 2021

Abstract

This article explores a transcription of a video recording Thai Finger Spelling (TFS)—a specific signing mode used in Thai sign language—to a corresponding Thai word. TFS copes with 42 Thai alphabets and 20 vowels using multiple and complex schemes. This leads to many technical challenges uncommon in spelling schemes of other sign languages. Our proposed system, Automatic Thai Finger Spelling Transcription (ATFS), processes a signing video in 3 stages: ALS marking video frames to easily remove any non-signing frame as well as conveniently group frames associating to the same alphabet, SR classifying a signing image frame to a sign label (or its equivalence), and SSR transcribing a series of signs into alphabets. ALS utilizes the TFS practice of signing different alphabets at different locations. SR and SSC employ well-adopted spatial and sequential models. Our ATFS has been found to achieve Alphabet Error Rate (AER) 0.256 (c.f. 0.63 of the baseline method). In addition to ATFS, our findings have disclosed a benefit of coupling image classification and sequence modeling stages by using a feature or penultimate vector for label representation rather than a definitive label or one-hot coding. Our results also assert the necessity of a smoothening mechanism in ALS and reveal a benefit of our proposed WFS, which could lead to over 15.88 % improvement. For TFS transcription, our work emphasizes the utilization of signing location in the identification of different alphabets. This is contrary to a common belief of exploiting signing time duration, which are shown to be ineffective by our data.

Keywords: Sign Language Transcription, Sign Sequence Classification, Signing Video Transcription, Thai Finger Spelling Transcription

Introduction

Sign language is the main communication channel of a deaf community. However, sign languages are not universal nor mutually intelligible with each other. There are many different sign languages around the world and also numerous different signing gestures to represent their alphabets. A fingerspelling is used in sign language to represent alphabets for spelling proper names and any word whose meaning has not been defined as a semantic sign. Although signs may be defined differently, how signs are performed can be categorized into a few signing schemes. American, French, and Russian Sign Languages [1-4] employ a one-hand scheme; Australian and British Sign Languages [5,6] employed a two-hand scheme. To cope with 42 alphabets and 20 vowels, TFS employs a one-hand scheme with an extension using movement and multi-posture signings for the alphabets and a two-hand scheme for the vowels and intonation marks [7]. Noted that, several finger spellings of large-alphabet-repertoire languages, e.g., Chinese and Japanese, resort to a phonetic system for a manageable set of spelling signs so that their simple signing schemes are adequate.

Resorting to various signing schemes, TFS poses a challenge on automatically disambiguating a word from a signing video. Previous works [7,8] frame TFS recognition as an image classification.

Despite their spectacular results, their approaches cannot address the pragmatic issue of transcribing a signing video.

Our work addresses ATFS, which has not been addressed in the literature. To achieve such a task, our proposed system utilizes image classification, sequence modeling, frame grouping, and a few smoothening mechanisms. An application of smoothening operation is found to be crucial. Our results also show a benefit of coupling image classification and sequence modeling stages by using a feature or penultimate vector for label representation rather than a definitive label or one-hot coding. It should be noted that pipelining image classification and sequence modeling is quite a common configuration for video processing and decoupling the stages is often the 1st intuition. This finding may help bring awareness when a pipeline is designed. Regarding the evaluation of the transcription, we adopt a concept of Word Error Rate (WER) [9] to propose AER for the overall performance index.

In addition to technical contributions, our work found out that (based on general practice among TSF signers) signing locations can viably be utilized to group frames by alphabet. Noted that, we have found that a notion of effective use of signing time duration for alphabet separation is quite commonly perceived by our peers. However, our evidence has strongly disproved this notion and supports our choice of signing locations. Therefore, we emphasize the effectiveness of signing locations over signing time duration as a major cue for alphabet separation.

Literature review

A Finger Spelling Recognition (FSR) system takes an image of a signing hand and automatically gives a corresponding sign label. FSR is therefore naturally framed as a classification problem. Nakjai and Katanyukul [7] have addressed FSR for TFS using their customized CNN and reported an average accuracy of 91.26 % on a single-image setting. A subsequent study [10] employing a Yolo-based Darknet-19 [11] to simultaneously locate a hand and read a sign has reported mAP 82.06 % also on a single-image setting, but with a complex background. Another study [8] has addressed FSR on a plain background using VGG-16 [12] and reported mAP 97.59 % on a single-image setting, as well as discovered Latent Cognizance-an innovative mechanism to allow an out-of-context awareness.

Despite these impressive results, these previous works have framed the problem under a single-image setting, while a video setting would be more likely in a practical situation. Although addressing TFS video transcription is in need of a practical application as Nakjai and Katanyukul [7] have discussed, an investigation into this issue has so far been lacking. Therefore, this article proposes ATFS to address transcription of a video clip signing TFS. Unlike other sign languages, TFS employs multiple and complex schemes, as shown in **Table 1**. This uniqueness renders challenges and the necessity of addressing video settings, which may not be as critical in other sign languages.

As shown in **Table 1**, British, American, Russian, Arabic, French, Chinese, and Japanese sign languages rely mainly on still postures (either in a one- or two-handed scheme) with only a few exceptions for movement signing.

On the contrary, Thai sign language relies significantly on a multi-posture scheme. It employs 26 multi-posture signings and 1 movement. (For simplicity, we followNakjai's and Katanyukul's [7] practice, i.e., perceiving movement signing as multiple still signing postures. This simplification allows more straightforward conceptualization and efficient implementation. Therefore, we may refer that TFS has 27 multi-posture signings later on). The fact that TFS strongly relies on a multi-posture scheme makes a transcription task ambiguous, e.g., a sequence of TFS signs: 4, 7, 8, and 14 can be transcribed as [4,7,8,14], which all are treated as single postures, resulting "ดหวน" in Thai script, or transcribed as [4,7,8,14], whose first two signs are treated as a multi-posture signing, resulting "หวน" in Thai script. Our work proposes to address this through a sequence model.

Although sequence modeling in TFS has not been examined previously, the idea is not uncommon in sign language transcription studies. Liwicki and Everingham [6] employ Hidden Markov Model (HMM) to transcribe British Sign Language (BSL). They use Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM) to turn a signing video to a sequence of class labels, which includes all BSL signs and a non-sign label. Liwicki and Everingham [6] treat non-sign as an extra class and have

trained their model with non-sign images along with BSL sign images. They report a word recognition accuracy of 98.9 %.

Also using HMM as a sequence model, Sidig *et al*. [13] study its application on Arabic sign language. Their approach starts from processing a video of signing hand and turning it into a sequence of representative optical flows. Sidig *et al*. [13] define a sum of all magnitudes of optical flows in an image to be a representative optical flow for that image. It is intended to catch the transition, which is supposed to cause large magnitudes of optical flows. Then, the sequence is smoothened through a one-dimensional convolution with a smoothing filter, before going through thresholding. Any smoothened optical flow whose magnitude is smaller than a designated threshold is assumed to be a sign. Then, the assumed signs go through a modified fourier transform to produce a sequence of features, which eventually go into the HMM. Sidig *et al*. [13] have reported an average accuracy of 99.11 %.

Looking beyond transcription, Moryossef *et al*. [14] have their attention on distinguishing signing and non-signing sessions from a video performing German sign language. They employ part affinity field [15] to provide human pose estimation of each image frame. Then, Long Short-Term Memory (LSTM) as their sequence model takes in a sequence of the optical flows of pose estimations and identifies signing and non-signing frames.

Our work employs LSTM to handle sequential data dependency but using signing location as a cue to identify signs of the same alphabets as well as to exclude the non-signs. As mentioned earlier, TFS relies on a multi-posture signing. Multiple signs may associate with the same alphabet. Therefore, an ability to effectively identify frames of different signs belonging to a distinct alphabet is advantageous in TFS, but this characteristic may be irrelevant to British or Arabic sign languages.

The rationale behind a choice of signing location is provided later in the Experiments and Results.

Table 1 Example of fingerspelling schemes in various sign languages.

Sign Language	Number of Alphabets /Characters	Number of Signs	Signing Scheme	Remark
British [6]	26	26 signs	Two-handed posture	
American [1,2]	26	24 signs and 2 movements	Single-posture	
Russian [3]	32	24 signs and 8 movements		
Arabic [16,17]	30	28 signs and 2 movements		
French [4]	26	23 signs and 3 movements		
Chinese [18]	3000+	30 signs		Phonetic System (30 sounds of Pinyin characters)
Japanese [19,20]	46 (Hiragana)	41 signs and 5 movements		Phonetic System (46 Hiragana characters)
Thai [7,21]	42 consonants, 4 intonations and 20 vowel visuals	25 signs (consonants), 6 signs (non-consonants) and 17 palm locations	Single-posture, Multi-posture, and Hand-mapping	26 alphabets are represented by multi-posture signing. One alphabet is represented by movement.

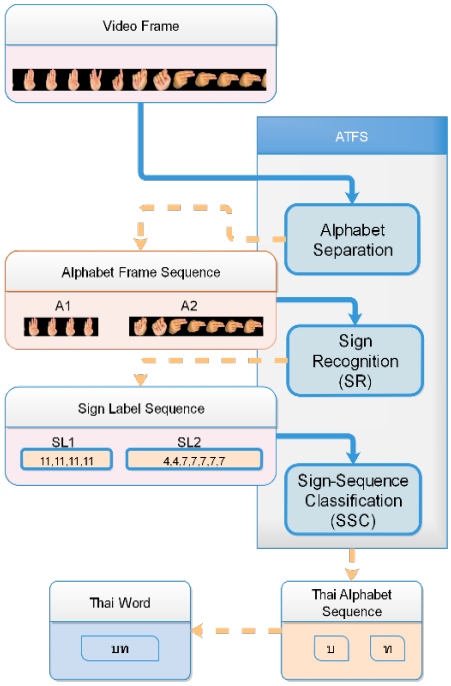

Figure 1 ATFS. The ATFS has 3 stages, i.e., alphabet separation stage, sign recognition stage, and sign-sequence classification (SSC) stage. Video data (a sequence of image frames) as the system input goes to the alphabet separation stage. The alphabet separation stage groups consecutive signing frames by alphabet. Each group is called an Alphabet Frame Sequence. The 2nd stage takes the frame sequence and provides a corresponding sequence of sign labels (or equivalence). The final stage then transcribes the given sequence of sign labels to a Thai word, which is a sequence of Thai alphabets.

The proposed system

This section describes the process of our ATFS system, targeted 25 signs (covering all 42 Thai alphabets, not including 20 vowels and 4 intonation marks). The ATFS system takes a video clip as input and transcribes to a Thai word. Our ATFS system can transcribe only 1 word per video clip. It has 3 main stages: the 1st stage is alphabet separation, the 2nd stage is sign recognition, and the last stage is SSC. **Figure 1** illustrates the ATFS system and its 3 main stages, along with examples of information going in and out of each stage.

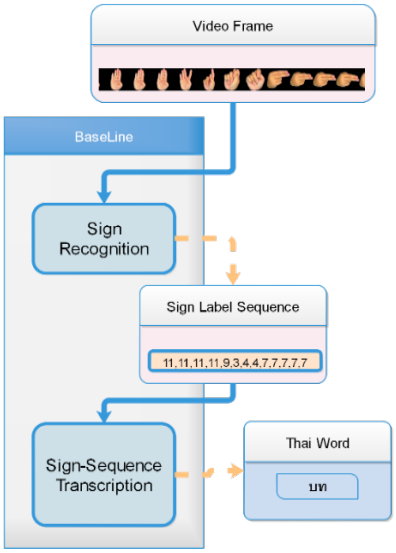

Figure 2 Baseline approach is a simple sign transcribing system. The baseline approach takes video data and eventually provides its predicted Thai word. It is run by a much simpler mechanism than ATFS. The baseline approach transcribes an alphabet from sign labels based on a greedy algorithm.

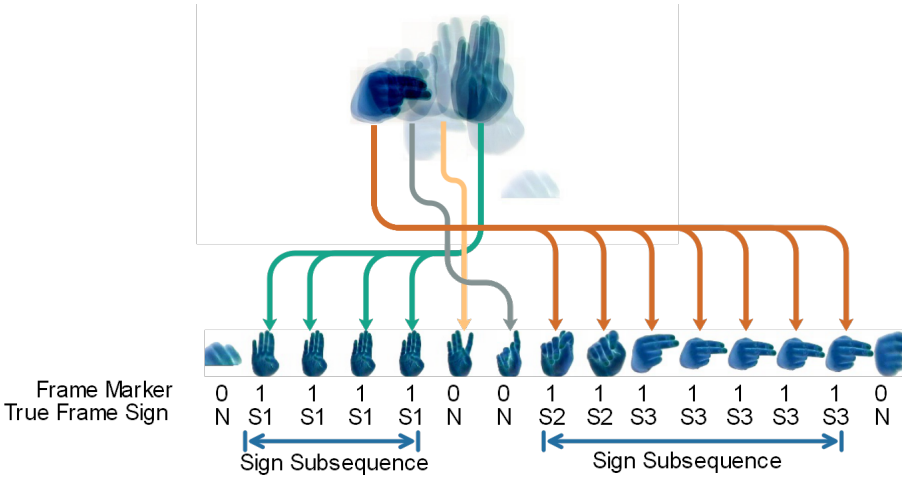

Figure 3 Alphabet-separation (ALS) stage. A sequence of image frames, shown as a superimposed image on the top, illustrates the signing of a word with 2 alphabets. A signer generally positions each signing at a slightly different location. Signing postures are broken down and displayed in order from left to right; arrows associate each signing to its position. Below the signing postures, frame markers (output from ALS stage), and true frame signs are shown. Frame markers 0 and 1 indicate non-sign and sign, respectively. The whole signing spells a two-alphabet word "บท", composed of 3 signs: S1 → "บ"; (S2,S3) → "ท". Symbol N represents a non-sign.

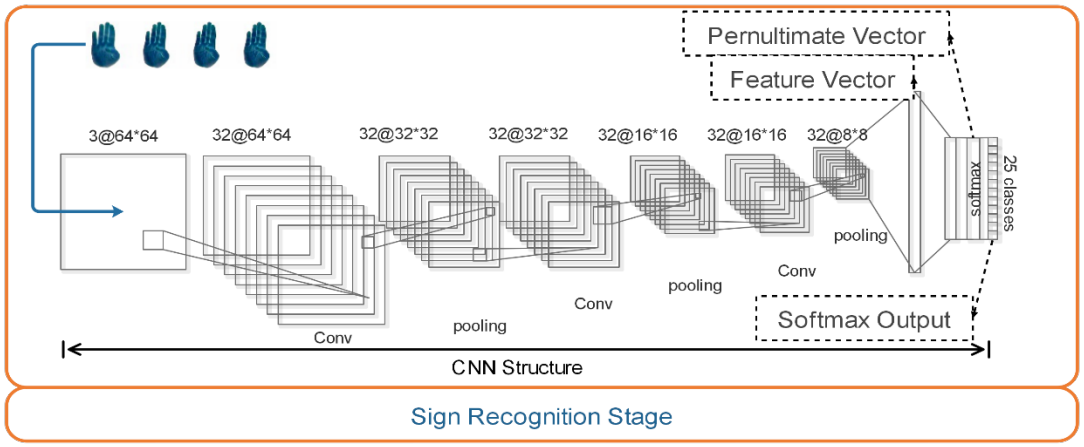

Figure 4 CNN structure of SR stage (followed [7]). The CNN structure has 3 convolution layers each is followed by a ReLU activation and a max-pooling layer. Each convolution layer has 32 filters each whose size is 3×3. After the 3rd max-pooling layer, there are 2 fully-connected layers with 1,024 nodes of a ReLU activation and 25 nodes of a softmax function, respectively, to deliver the final predicted 25 classes.

The alphabet-separation stage

The 1st stage, called the alphabet-separation (ALS) stage, takes a video clip and provides frame markers to indicate which frames belong to the same alphabet and which frames are non-signing and should be discarded. Each video frame will be marked as either being a sign frame (labeled 1) or being a non-sign frame (labeled 0). Consecutive sign frames are called sign subsequence. **Figure 3** illustrates an alphabet separation stage.

Our ALS stage marks signing frames based primarily on signing positions. A signer generally positions their signing of a different alphabet at a slightly different location. A centroid of a hand area in an image frame is used to represent a signing position. We investigate 5 ALS approaches, i.e., D1, D2, D2M, D2S, and HM. The D1, D2, D2M, and D2S employ thresholding on a Euclidean distance between centroids of 2 consecutive frames. D1 simply uses a (single) threshold to decide a marker, that can be calculated as follows:

$$m^t = 1 \text{ if } D^t < \tau_d, \text{ and } m^t = 0 \text{ otherwise,} \quad (1)$$

where m^t is a marker of the t^{th} frame; τ_d is a pre-defined threshold; Euclidean distance, $D^t = \sqrt{(C_x^t - C_x^{t-1})^2 + (C_y^t - C_y^{t-1})^2}$, for $t = 2,3,4,\ldots,T$ and $D^1 = 0$; C_x^t and C_y^t are x and y coordinates of a centroid of the t^{th} frame. D2, D2M, and D2S employ double thresholding as in Eq. 2.

$$m^t = \begin{cases} 1, & \text{if } d^t < \tau_l, \\ 0, & \text{if } d^t > \tau_u, \\ m^{t-1}, & otherwise, \end{cases} \quad (2)$$

where τ_l and τ_u are lower and upper thresholds; d^t is a distance at the t^{th} frame. D2 uses a Euclidean distance $d^t = D^t$. We speculate a benefit of a smoothening mechanism. Therefore, D2M and D2S are equipped with smoothening operations. D2M uses moving average distance, $d^t = V^t$, whose value is calculated by $V^t = \frac{\sum_{i=(t-n)}^{t} D^i}{n}$, for $t > n$, where D^i is a Euclidean distance of the i^{th} frame; n is a user-

specific value (moving average window); and $V^t = D^t$ for $t = 1, \ldots, n$. D2S uses a skipping distance, i.e., $d^t = D^t$ for $t = 1,3,5, \ldots$ and $d^t = D^{t-1}$ for $t = 2,4,6, \ldots$.

HM implements a signing location concept through a heatmap mechanism. It also takes centroid frequency into account. It is to utilize frequent proximity of centroids. More frequent proximity of centroids indicates a more likely that it is a signing frame. Specifically, a heatmap $H \in \mathbf{R}^{w \times h}$ is constructed through the Gaussian kernel density method, that can be calculated as follows:

$$H(x,y) = \sum_{t=1}^{T} \exp\left(-\left(\frac{(x-C_x^t)^2}{2\sigma^2} + \frac{(y-C_y^t)^2}{2\sigma^2}\right)\right), \quad (3)$$

where $H(x,y)$ is a heatmap pixel at (x,y) coordinate; C_x^t and C_y^t are x and y coordinates of a centroid at the t^{th} frame; σ^2 is a user-specific span parameter. Then, an alphabet area $A(x,y)$ can be identified through thresholding, $A(x,y) = 1$ if $H(x,y) \geq \tau_h$ and $A(x,y) = 0$ otherwise. Threshold τ_h is user-specific. To mark a sign frame, any frame whose centroid lies inside the alphabet area where $A(x,y) = 1$ will be marked as a signing frame, that is $m^t = 1$ if $A(C_x^t, C_y^t) = 1$ and $m^t = 0$ otherwise.

According to our pilot study whose results show the benefits of smoothening, our study employs the Window Frame Smoothening technique (WFS) to improve the output from the ALS stage. Given a sequence of frame markers from ALS as $[b^1, b^2, b^3, \ldots, b^T]$, WFS with a hyperparameter s corrects the sequence by $b^j = b^i$ if $b^i = b^{i+s}$ for $j = i+1, \ldots, i+s-1$ in a step-by-step manner from $i = 1$ to $i = T - s$. For example, given s = 5, when the frame markers are 0101110, WFS will output 0111110; when frame markers are 1001010, WFS will output 1000000.

The sign recognition stage

The output from ALS is used to break down an entire word signing sequence into an alphabet sequence or alphabet sequences, as shown in **Figure 1** Each image frame in every alphabet sequence is then fed to the 2nd process-the sign recognition (SR) stage. The SR stage takes an image frame and identifies a TFS sign corresponding to that frame. Our SR implementation takes after Nakjai and Katanyukul [7], using an image classification by Convolution Neural Network (CNN).

The SR stage works on a still image and processes 1 image at a time. However, feeding it with an alphabet sequence results in running SR multiple times and the outcome is a sign-label sequence.

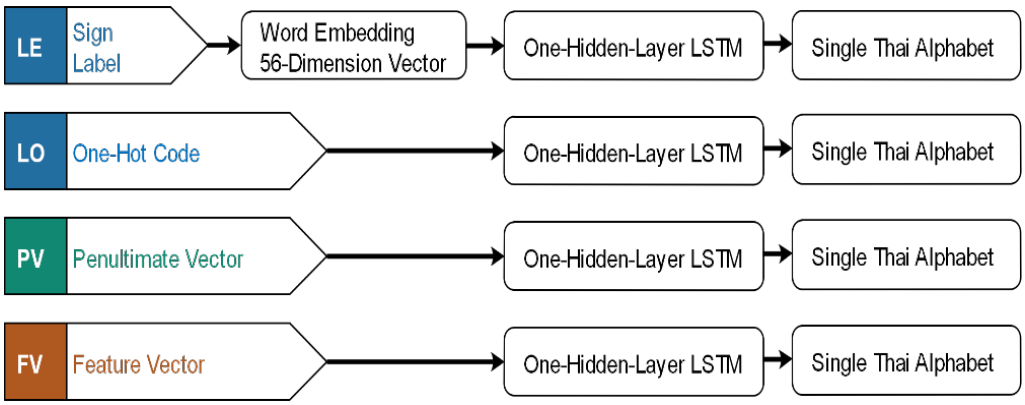

Figure 5 Four sign representation alternatives are explored for how they affect the SSC performance.

The sign-sequence classification (SSC) stage

The SSC stage turns sign-label sequences into alphabets. Similar to the SR process in the sense that SSC machinery also works 1 at a time. The SSC process takes a sign-label sequence and predicts a single Thai alphabet corresponding to the sequence. A series of sign-label sequences results in running SSC multiple times and the collective outcome is a sequence of alphabets, which is the ATFS outcome-a transcribed word.

Intrigued by results from our pilot experiments, we explore how the representation of sign labels affects the transcription performance. Four alternatives of a sign-representation sequence are explored. Each alternative represents how SR and SSC are connected. The SSC stage takes sign labels from the SR stage. The SSC stage may represent sign labels simply by one-hot coding (denoted LO) or it may even elaborate the label representation through word embedding [22] (denoted LE). Optionally, the SSC stage can reach further and take a penultimate vector [8]-a value vector before softmax calculation- (denoted PV) or a feature vector (denoted FV) deeper inside the SR stage. **Figure 5** illustrates these 4 alternatives. Given the input sequence $[x_1, \dots, x_T]$, the SSC predicts an alphabet label y using LSTM with 1 hidden layer. The LSTM has been widely used in sequence classification [23], machine translation [24], speech recognition [25], and video description [26]. Concisely, LSTM predicts label y from $p(y|x_1, \dots, x_T) = \text{argmax}(\text{softmax}(h_T))$, where h_T is obtained through $h_t = o_t * tanh(c_t), o_t = \sigma(U_o h_{t-1} + W_o x_t + b_o), c_t = (f_t * c_{t-1}) + (i_t * c'_t), c'_t = tanh(U_c h_{t-1} + W_c x_t + b_c), f_t = \sigma(U_f h_{t-1} + W_f x_t + b_f), i_t = \sigma(U_i h_{t-1} + W_i x_t + b_i)$, for $t = 1, \dots, T$, when all U's and W's are weight parameters; b's are bias parameters; operator $*$ represents an elementwise multiplication; $\sigma(\cdot)$ represents a logistic sigmoid function; and $h_0 = 0$. Values of U's , W's, and b's are obtained through a training process.

Experiments and results

Our experiments are to evaluate our proposed system both entirely and stagewise. To prepare and evaluate our system, 2 dedicated TFS datasets are acquired.

TFS video dataset

The TFS video dataset is collected from a professional TFS signer. The dataset contains 212 video clips (162 clips for training and 50 clips for testing). The 212 samples are chosen from the top 200 bi-grams that most frequently appear in Thai names. Each video shows a normal pace and clear signing. Each video clip (lasts 5.37s on average at 29 fps) records a bi-alphabet Thai word (a word with exactly 2 alphabets) corresponding to 2~6 TFS signs. The 1st alphabet lasts 0.3 - 0.8 s on average. The 2nd alphabet lasts 0.68 - 1.1 s on average. Annotation has been marked on video frames for TFS signs as well as non-signs. This dataset is intended for ALS, SSC, and the evaluation of the entire system.

TFS image dataset

The image dataset is collected from 11 signers. It contains all 25 hand sign classes. Each sign class is posed 5 times by each signer that makes it to $11 \times 25 \times 5 = 1{,}375$ original images, before being augmented to 30,000 images. The 15,000 images are used as a training set and the remainings are used as a test set. The image dataset is intended for the development and stagewise evaluation of the SR stage.

Experiment setting

The experiments explore various combinations of approaches. Stagewise, the SR stage is implemented by CNN with configurations as specified by the previous study [7]. The CNN, whose structure is shown in **Figure 4**, is trained with a TFS image dataset. Five approaches (HM, D1, D2, D2M, and D2S) are investigated for ALS stage. HM uses σ of 10. D2M uses a 3-frame moving average. The HM and D1 use a single threshold, i.e., $\tau_{hm} = 4.63$ and $\tau_d = 6.5$, respectively. The double thresholds of $\tau_l = 4.2$ and $\tau_u = 6.5$ are applied to D2 and D2M. The double thresholds of $\tau_l = 6.5$ and $\tau_u = 11.22$ are applied to D2S. The effects of WFS on alphabet-separation results are also examined on 2 options, i.e., no WFS and WFS with a hyperparameter s is 5. The SSC stage is implemented by a one-hidden-layer LSTM. Five alternatives of LSTM have been assessed, i.e., LSTM with 2 hidden nodes, LSTM with 28

hidden nodes, LSTM with 56 hidden nodes, LSTM with 128 hidden nodes, and LSTM with 256 hidden nodes.

In addition, our experiment investigates 4 alternatives of sign-label representation (i.e., an input of the SSC stage) as discussed earlier (**Figure 5**). LO using one-hot coding is straightforward. Our experiment uses LE with the embedding vector of size 56. PV using a penultimate vector of the SR stage is also straightforward. FV uses the last convolution layer of the SR stage as a sign-label representation. Evaluation of SSC and the entire system was repeated 5 times.

We have also employed a rule-based approach as a baseline (**Figure 2**) to have others compare against. The rule-based approach employs CNN (the same structure and weights as one using by other approaches) to provide a sequence of sign labels from a video clip. Then, the sequence of sign labels is translated to a Thai word using prespecified rules. The rules are that (1) a subsequence of consecutive signs of the same label whose length is longer than a prespecified number N_b ($N_b = 5$ in our experiment) will be transcribed as an output sign of that label; otherwise, the subsequence is ignored; (2) to translate from the output signs to a corresponding alphabet, a sequence of output signs is greedy-wisely matched to alphabets using the signs-to-alphabet mapping table. It is noted that some alphabets are corresponding to a single sign and some alphabets are corresponding to a series of multiple signs.

Another word, this baseline scheme proceeds from the beginning of the sign sequence and tries to match the longest subsequence possible; then proceeds along the sequence to the position after the matched subsequence until the sequence is exhausted. When no match is possible even for the shortest subsequence (length 1) the subsequence is discarded and the translation proceeds to the position after.

Evaluation

AER is used as a metric for the entire system assessment. Inspired by WER [9], AER measures the minimum number of operations, i.e., substitutions, deletions, and insertions, to make a word under evaluation match the reference. The AER is calculated as follows: $AER = \frac{S+D+I}{N}$, where S is a number of substituted alphabets; D is a number of deleted alphabets; I is a number of inserted alphabets and N is a total number of alphabets in the reference word. Stagewise, the SR stage is tested on the image dataset for its accuracy. The ALS and SSC stages are tested on the video dataset for their F-score and accuracy, respectively.

Stagewise evaluations of ALS and SR are conducted independently, but to evaluate SSC stage wisely, the SR stage is employed to provide sign-label representation for each image frame. That is, the evaluation process starts from the hand selection of an alphabet sequence (a subsequence of images all associating to the same alphabet) and feed the alphabet sequence into the SR. Then, a sign-label sequence (output from the SR) is fed to the SSC. Finally, the output of SSC is evaluated against the ground truth for its accuracy.

Experimental results

Table 2 shows test results of the ALS stage in the F-score. The last column shows the percentage of improvement using WFS over not using WFS. All options have been shown to perform well with F-score over 0.8. D2M delivered the best performing results either with or without WFS.

WFS has been shown to contribute from 1.6 to 3.1 % improvement on any distance-based approachbut shown to contribute very little on the HM approach. This may be explained by that HM has its own intrinsic smoothening effect (from a gaussian basis) and therefore an extra smoothening effect may be just redundant.

In addition, D2M and D2S employ moving-average and skip-frame schemes, respectively. These schemes also provide some degree of smoothening effect and their superior performance to their counterpart D2 may be attributed to their smoothening effect as well.

Table 2 Alphabet-separation results.

	F-score		Improvement %
	Without WFS	**With WFS**	
D1	0.825	0.851	2.6
D2	0.826	0.857	3.1
D2M	0.861	0.886	2.5
D2S	0.849	0.865	1.6
HM	0.838	0.842	0.4

The SR stage is found to have a test accuracy of 0.818. **Table 3** shows the evaluation results of the SSC stage. The table presents each approach with its best performing hyperparameters-hidden size- along with its accuracy: mean, median, and 1st and 3rd quartiles. Interestingly, despite both deriving their input from sign labels, LO-which straightforwardly uses one-hot coding to represent a sign label- is outperformed by LE-which employs word embedding to represent a sign label. We hypothesize that word embedding may provide an easier (numerically) digestible representation.

A more practical point is that sign label is represented (or, another perspective, how SR and SSC stages connect) hugely affects the transcription performance. LO and LE, which simply take sign labels from SR and pass them to SSC, were more than 10 % outperformed by PV and FV (both reach further into the SR stage and take either penultimate or feature vector for sign-label representation, as SSC input). This indicates a crucial fact that simply taking the conventional output from an earlier stage and pass it as an input to the following stage may significantly sacrifice an overall performance. In addition, a sign label, pernultimate vector, and feature vector are in the order processed information backwardly along the SR stage.

Our results may indirectly provide a peek into how information is lost along the deep neural processing path. Since FV and PV show similar performance, this may imply good information preservation from the last convolution layer (providing a feature vector) to the penultimate layer (providing a penultimate vector). As performance drops significantly as we use a label instead of a penultimate vector, this may imply a substantial loss of information along the way from a penultimate layer to a softmax layer to the final label decision, which may have been done through an argmax function. **Table 4** shows the overall performance of different approaches.

Table 4 presents each approach with its best performing hyperparameters–hidden size–along with its average AER over 5 repeats. The Benefit columns present the improvement of using the WFS technique over not using it.

Our results reveal that a combination of D2M and WFS for alphabet separation and FV for SSC leads to the best performing transcription (AER 0.256). Beyond the best performing set, the big picture showing in **Table 4** is that smoothening mechanisms are crucial to the overall performance. D1 and D2 without WFS do not have any smoothening mechanism and they deliver the worst (highest) AERs. Performances of D1 and D2 without WFS are multiple times worse than the other treatments. Comparing D2 to its smoothened counterparts (D2S, D2M, D2 with WFS, D2S with WFS, and D2M with WFS) reveals that smoothening can improve the transcription performance from 2.6 times (LO: D2/D2M) to 16 times (FV: D2/D2M+WFS).

Table 5 provides an example of how prediction outputs look like. It appears that WFS helps reduce alphabet duplication significantly. **Figure 6** shows an example illustrating how smoothening mechanism helps improve the transcription performance by keeping the consecutive signing intact.

Table 3 Sign sequence classification results.

Sign Label Representation	Hidden Size	Transcription Accuracy			
		Mean	Median	Q1	Q3
LO	256	0.744	0.75	0.74	0.75
LE	128	0.790	0.79	0.78	0.80
FV	128	0.894	0.91	0.89	0.92
PV	256	0.924	0.92	0.91	0.97

Table 4 The overall performance (in AER) of the entire pipeline.

	Without WFS				With WFS				Benefit of WFS (over not using WFS)			
	LE	LO	FV	PV	LE	LO	FV	PV	LE	LO	FV	PV
D1	4.49	4.5	4.186	4.374	0.342	0.434	0.278	0.348	13.129	10.369	15.058	12.569
D2	4.432	4.446	4.13	4.318	0.34	0.414	0.26	0.342	13.035	10.739	15.885	12.626
D2M	1.658	1.708	1.332	1.542	0.344	0.408	0.256	0.298	4.820	4.186	5.203	5.174
D2S	1.328	1.364	0.996	1.216	0.514	0.552	0.336	0.452	2.584	2.471	2.964	2.690
HM	1.052	1.118	0.778	0.962	0.528	0.594	0.344	0.442	1.992	1.882	2.262	2.176
Baseline	0.63											

Table 5 Examples showing effects of frame smoothening. The examples are arbitrarily selected from prediction results when using D2M with a feature vector. The reference column shows the correct answers.

FV with D2M		Without WFS		With WFS		Baseline	
Sample	Reference	Prediction	AER	Prediction	AER	Prediction	AER
Data_63	ธน	ตหชน	1.5	ชน	0.5	ตด	1
Data_84	ชล	ชหชกลล	2	ชล	0	ล	0.5
Data_196	ชพ	คชหชพพ	2	ชพ	0	หพ	0.5
Data_182	ธง	ตหกนง	2	ชง	0.5	-	1
Data_26	ธร	ตหธร	1	ธร	0	ร	0.5
Data_111	ชอ	ภงหชอ	1.5	ชอ	0	อ	0.5
Data_117	ชว	ชงหกว	1.5	ชว	0	ว	0.5

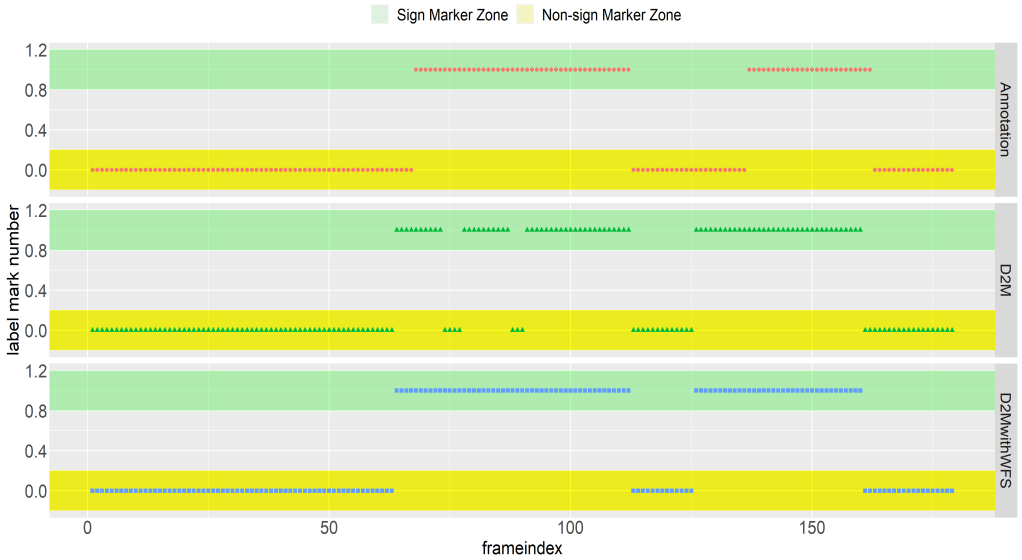

Figure 6 Sign and non-sign frames: human annotation (reference, the uppermost plot), marking using D2M (the middle plot), and marking using D2M with WFS (the lowermost plot). A marking of 1 indicates a 'sign'. Marking of '0' indicates a 'non-sign'. WFS has corrected D2M output and resulted in marking more resemble human annotation.

Affirming the stagewise results, using word-embedding seems to provide an extra beneficial effect on the transcription as shown by LE outperforming LO in both stage-level accuracy and comparable pipeline AER (most noticeable at ~21 % improvement on D1, with WFS). Noted that we speculate a marginal benefit of using a word embedding with features or penultimate vectors. However, the decisive conclusion may require a dedicated study.

As discussed earlier, taking information deeper into the SR stage seems to allow the SSC stage and the entire pipeline to perform better (as AERs from using FV are lower than ones from using PV and AERs of PV are lower than ones of LO). **Figure 7** illustrates this point. Regardless of the treatment, a higher degree of coupling seems to be more beneficial and the trend is monotonic (almost linear).

Apparently from **Table 4**, smoothening (WFS) is beneficial in every case, with a larger effect on D1 and D2. Noted that, both D1 and D2 do not have moving average or skip-frame mechanisms. With WFS, all approaches outperform the baseline as shown in **Figure 8**.

To compare different smoothening techniques, **Table 4** can be rearranged, as shown in **Table 6**. We can see that all smoothening techniques help improve the overall transcription performance, as AERs of moving average, skipping a frame, and WFS are smaller than no smoothening (D2) in multitude. Among these techniques, WFS is most beneficial. However, we view them rather as options that can be combined if suitable than as strickly competing choices. Speaking of combing the smoothening techniques, it is interesting that skipping frame seems to work better than moving averagewhen applied alone. However, when combined with WFS, it is moving average that delivers a better performance regardless of what sign-label representation is used. We speculate that the explanation may lie in the smoothening details: moving average and WFS may well complement each other, while skipping frame may have some effect overlapping with WFS.

Table 6 Direct comparison of different smoothening techniques (Re-arranged **Table 4** for a better perspective on different smoothening techniques).

Smoothening		Sign-Label Representation			
		LE	**LO**	**FV**	**PV**
No smoothening	(D2)	4.432	4.446	4.13	4.318
Moving Average	(D2M)	1.658	1.708	1.332	1.542
Skipping Frame	(D2S)	1.328	1.364	0.996	1.216
WFS	(D2 + WFS)	0.34	0.414	0.26	0.342

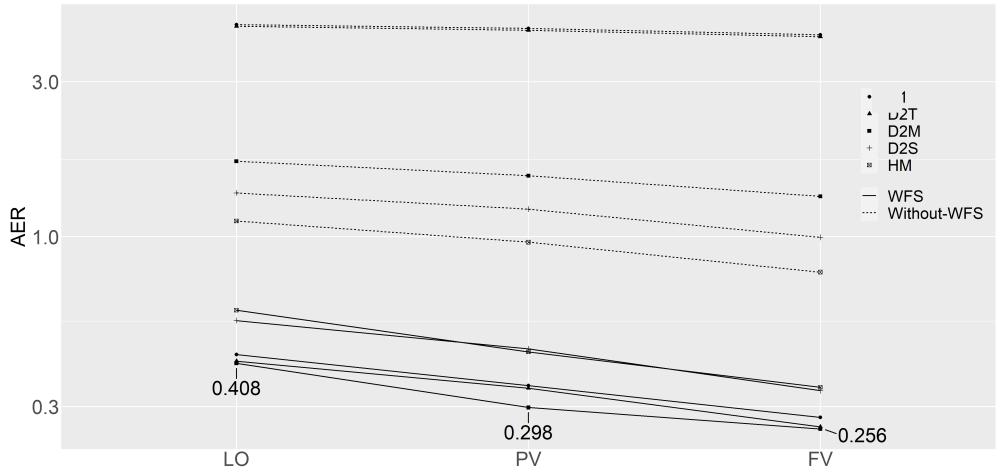

Figure 7 Performance (in AER) of various treatments over a degree of stage coupling, from low coupling (LO) to medium coupling (PV) to high coupling (FV).

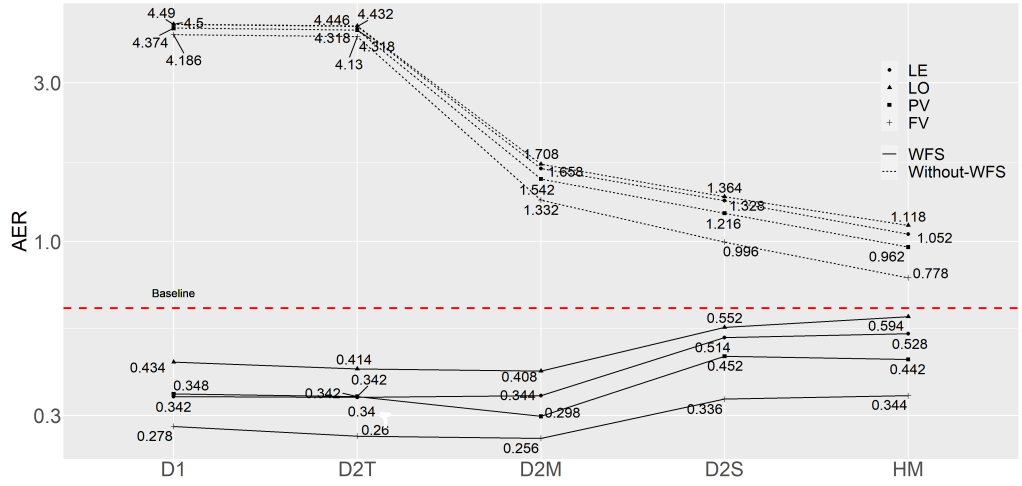

Figure 8 AERs of different treatments. A lower AER indicates a better performance.

Regarding how sign and non-sign frames are identified, there are many approaches. Some use image features for identification. Liwicki and Everingham [6] use a supervised model to identify a non-sign frame. Ingeniously, Nakjai and Katanyukul [8] use a probabilistic inference—Latent Cognizance—taking advantage of what the model has already learned. Some use image transition to identify non-sign frames. Sidig *et al*. [13] use optical flows. Some dedicate an entire system just to identify non-sign frames. Generally, the transition approach is based on the assumption that frames during the transition are non-signing frames; otherwise, they are signing. Therefore, the transition approach is likely to be computationally cheaper than an image-feature-based approach, but this comes with a risk for unintended hand postures during non-signing sessions or fast signing during a signing session. To partially address it, Moryossef *et al*. [14] propose a system utilizing part affinity field [15], optical flow, and LSTM dedicated to identifying signing and non-signing sessions.

Close in spirit to image transition, our approach is based on signing location. It is a convenient choice because the signing location is already used for alphabet separation. The rationale behind using signing location is from our study on TFS signers' behavior to find an effective cue to identify distinct alphabet signing.

To identify distinct alphabet signing, we conjectured 2 possible indicators, i.e., (1) that distance between centroids of hands in consecutive frames was small when the 2 consecutive frames associate to the same alphabet and the distance was large otherwise; (2) that posing time duration of the alphabet was long compared to a transition time. These 2 conjectures were examined as follows. It should be noted that while using hand location, i.e., a TFS signer performs each alphabet signing at a distinct hand position, is a common practice, the direct implementation of this signing location is quite computationally expensive; therefore, we develop an assumption 1 so that if it is validated, the implementation can be done more efficiently.

Assumption 1 is that the distance between centroids of hands in consecutive frames is small when the 2 consecutive frames contain the same alphabet and the distance is large otherwise. That is, 2 consecutive frames are corresponding to the same alphabet when their centroids of hands appearing in both frames locate close to each other (when superimposed). **Figure 3** illustrates the rationale.

Assumption 1 Verification. To evaluate the assumption 212 signing video clips were hand-marked for signs and non-signs on all image frames. Given that all clips correspond to 2-gram words, sign and non-sign labels along with frame orders in the sequence can be used to group frames into 3 categories: signing frame of the 1st alphabet (called "First Alphabet"), signing frame of the 2nd alphabet (called "Second Alphabet"), and non-signing frame (called "Transition"). Then, a centroid of a hand in each frame can be located and the distance between centroids in 2 consecutive frames can be computed using Euclidean distance. **Figure 9** shows the distances in boxplots.

Figure 9 has the y-axis representing the distance in a number of pixels. Due to the possibility of variation in a signer's hand sizes and shapes and visual sizes due to a camera effect, we derive a measure, which is relative to a signer's hand size. Given a signer's hand contained in a bounding box of 107×205 pixels (hand sign #11 in Nakjai and Katanyukul [7]; hand sign #11 revealing the entire palm is selected for clarity purpose), the diagonal is $d = \sqrt{107^2 + 205^2} = 231.24$ pixels. Our proposed measure is a percentage of the underlying distance (in pixels) to the diagonal length of the signer's hand (also in pixels). For example, a median distance of the 1st alphabet is 5.32 pixels, which is equivalent to $\frac{5.32}{231.24} \times 100\ \% = 2.3\ \%$ (of the diagonal length of the palm posed as sign #11).

Figure 9 shows the distances of the 3 groups. The distances in the alphabet groups (median 2.3 and 1.6 % in the First Alphabets and the Second Alphabets, respectively) are much smaller than the distance in the Transition group (median 14.35 %). This finding supports Assumption 1.

Assumption 2 is that the posing time duration of the alphabet is long comparing to a transition time.

Assumption 2 Invalidation. To evaluate Assumption 2, the same dataset as in Assumption 1 evaluation is used. Time duration can be inferred from a number of consecutive frame labels of a particular category. For example, frames No.16 to No.30 all are marked as the 1st alphabet. The time duration of the 1st alphabet is 15 frames (@ 29 fps, 15/29 = 0.517 s).

Figure 10 Figure 10 shows the time duration of each group. While transition time is quite short (17/29 = 0.586 s on median) compared to the time duration of the Second Alphabets (24/29 = 0.828 s on median), it is still difficult to distinguish the transition time from the time duration of the First Alphabets. This finding is against Assumption 2, especially when considering the First Alphabets and Transition. Therefore, as Assumption 2 is invalidated, we discard time duration as a factor for alphabet separation. We emphasize this point since we have found that Assumption 2 is a common misconception perceived by many of our peers.

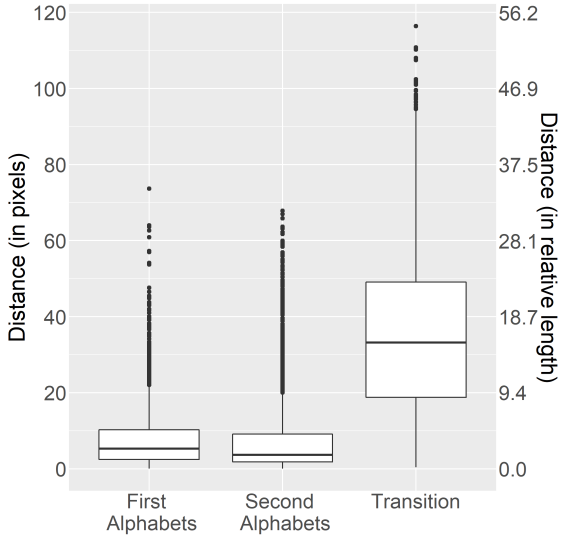

Figure 9 Boxplots of the distance between centroids in consecutive frames by category. Median distances: 5.32 pixels (2.3 %), 3.7 pixels (1.6 %), 33.18 pixels (14.35 %) for First Alphabets, Second Alphabets, and Transition, respectively.

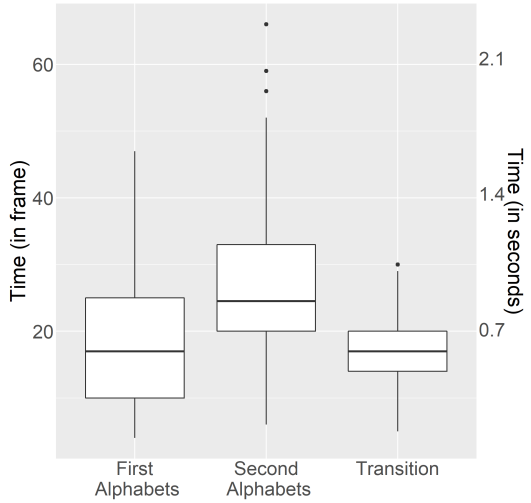

Figure 10 Time duration of signing per category. Median times: 17 (0.59 s), 24.5 (0.83 s), 17 (0.59 s) for First Alphabets, Second Alphabets, and Transition, respectively.

Discussion

Among alternatives explored for the alphabet separation process (ALS), our study has found D2M, which uses the distance between centroids of hand areas in 2 successive frames along with double thresholding and moving average, to be the most effective. This approach is based on location signing assumption, which our investigation has shown to be a viable cue and following the practice among TFS signers. The underlying spirit of a signing-location-based approach may be different from a concept of optical flow [6,13,14], but some implementation of optical flow estimation may arguably be close to D2M or other alternatives explored here. (An investigation of implementations of optical flow estimation is beyond the scope of this article.)

In addition to identifying frames signing for the same alphabet, our 1st stage ALS also indicates which frames are non-signing and should be discarded. Identifying non-signs is a crucial and interesting issue. Some [6] train a model with non-signing frames to label out non-signs in a supervised manner. Some [8] use a creative approach exploiting already learned probabilities and infer non-signs through approximation and Bayesian inference. Some [13] use transition, such as optical flow, as an indicator to distinguish signs from non-signs. As discussed earlier, a transition-based approach tends to be computationally cheaper than an image-based approach. However, a transition-based approach is susceptible to situations, such as unintended hand postures or rapid signing. Being aware of the risk of transcribing unintended hand postures during a non-signing session, some [14] partially address the issue using a full video-sequence analytic system, employing state-of-the-arts in deep computer vision and sequence modeling to distinguish between signing and non-signing sessions. Our ATFS is based on signing location, which is a convenient choice for that it is our cue for alphabet separation. The underlying rationale of signing location and 1 of transition are quite different, but the implementations may appear similar. While the transition is quite a common conception, signing location is derived from our dedicated study revealing a customary TFS practice, i.e., positioning each alphabet signing at each location.

This study has proved the viability of signing location to effectively indicate distinct alphabet signing. However, not only do TFS signers position each alphabet signing at each location, but they also perform it in order along with a linear progression. For example, to sign "ทวน" (in Thai script), a sequence of signs (4,7), 8, and 14 is performed where signs 4 and 7 representing the same alphabet are performed at the same location; then sign 8 is performed on another location horizontally next to the location of signs 4 and 7; lastly sign 14 is performed on the location horizontally next to the location of sign 8. This TFS practice of linear progression in order has not been sufficiently investigated nor exploited. The investigation of this matter may lead to a reliable yet computationally efficient TFS transcription system.

Recalling that our 2nd stage—Sign-recognition (SR) stage, taking an image frame and outputs a TFS sign (or a vector associating to a sign)—was implemented with CNN following [7] and found to be 0.818 accurate. Conferring to previous reporting accuracy of 0.91, Nakjai and Katanyukul [7] have tested their system on 125 images, while our SR was tested on an augmented data of 15,000 images. The difference in sizes of the test data may explain the difference in our reporting accuracy and 1 from the original.

Regarding the performance, ATFS with an AER of 0.256 may seem modest, comparing to still-image systems [7,8,10]. Firstly, those previous works are targeted still image settings, while ATFS is addressing a video setting. Secondly, AER is a different index intended for transcription of a word-level evaluation, while accuracy or mAP used in still-image systems [7,8,10] is meant for a sign-level evaluation, i.e., input is an image and output is a sign (not a word; it is not even an alphabet). To transcribe a word, one needs to read a sequence of signs from a signing video, before transcribing a sequence of signs to a sequence of alphabets, i.e., a word. Therefore, accurately identifying a sign from a given image is just a starting point of a transcription system. Also, while accuracy and mAP are "optimistic" indices, i.e., the higher number the better (perfect score is 1), AER is a "perfectionist" index, i.e., the lower number the better (perfect score is 0). Lastly, we see this discrepancy as a strong indicator for a challenge in transcribing TFS, as it stresses how difficult it is to progress from recognizing signs to reading a word. This points out the need to progress beyond sign recognition to achieve a practical TFS

transcription system that people can rely on. In addition, impressive performance reported in studies of other sign languages [6,13] has shown us the prospect TFS transcription might become and the prospect is likely to be with sequence modeling. Regarding studies of other sign languages, the investigation of the association between linguistic sign language characteristics and recognition issues is of crucial significance. It could allow progress in different sign languages to transfer across and mutually complement each other. That would lead to substantial progress in the field and is seemingly a good potential research direction.

As our findings reveal a benefit of the WFS mechanism, our experiment has investigated a few smoothening techniques other than WFS, including moving average (treatment D2M) and skipping distance (treatment D2S). Treatment HM explores the application of heatmap in alphabet separation. Although HM may not perform a smoothening operation in the sense of smoothening the subject over time, it gives the smoothening effect over the location. Despite WFS superiority in our experimental results, we believe that in general, a proper application of smoothening will help the system performance. Therefore, other smoothening techniques, e.g., convolution with a smoothing filter [13] can be beneficial as well. Further investigation into smoothening techniques, which are accounted for both smoothening over time and location, may be intriguing and might even be achieved simply through convolution over combined dimensionality of time and location.

On evaluation, instead of using word accuracy, which is rather rough especially for development in an early stage, we adopt AER. AER is an alphabet version of WER. Having a finer performance index is more reflective and allowing a better chance for inspection and improvement, which we believe is more suitable for the current state of TFS transcription research.

In addition to our major objective of building a reliable TFS transcription system, our findings may additionally provide some insight into where information might have lost along the neural processing path. The information seems to be well intact as it passes from the last convolution layer to the penultimate layer. However, the information seems to be substantially lost somewhere along the path from the penultimate layer to the softmax layer and then to the final class decision.

Interestingly, although both one-hot coding and word embedding are representing a sign, the word-embedding approach (LE) apparently outperforms the one-hot-coding approach (LO) stagewise and overall. We speculate that word-embedding may have provided a computationally easier representation than that of the straightforward one-hot coding. This observation has brought curiosity whether word-embedding could help feature or penultimate vector on a similar matter. This may worth a dedicated study. Noted that, our ATFS relies on an alphabet-separation stage to mark out any non-signing frame. Our previous studies [8,27] have discovered a promising approach–Latent Cognizance (LC)–to identify an out-of-context question, which in this case is a non-sign image. Employing LC either alone or with transition or with sequence modeling, to clean up non-signing images from a sequence of image frames has not yet been explored.

Lastly, our experiment has evaluated ATFS on video clips of bi-gram words, but ATFS structure or any of its components does not limit it from the capability to transcribing a word of any length. Our investigation here has pioneered a TFS recognition into a transcription realm. There are a lot of works to be done, including improvement on its accuracy, coverage of vowels, intonation marks and how to integrate everything, and thorough evaluation on rich datasets, before a practical full-featured automatic TFS reading system could be realized.

Conclusions

In this article, we proposed an ATFS, using a three-stage transcription process as well as a WFS mechanism to improve the overall performance of ATFS. The ATFS addresses an ambiguity issue of transcribing a TFS signing video data and has achieved an AER of 0.256. Our findings disclose an advantage of coupling image-classification and sequence-modeling stages, affirm the importance of smoothening mechanism, and also reveal a benefit of WFS mechanism in both stagewise and overall performance, with possible improvement up to 15.88 %. Data and code are made public at https://github.com/beebrain/ATFS_Thai_Finger_Spelling. In addition, signing locations are an effective cue for marking alphabet signing, conferred to signing time duration.

References

[1] C Oz and MC Leu. American sign language word recognition with a sensory glove using artificial neural networks. *Eng. Appl. Artif. Intell.* 2011; **24**, 1204-13.
[2] L Ding and AM Martinez. Modelling and recognition of the linguistic components in American Sign Language. *Image Vis. Comput.* 2009; **27**, 1826-44.
[3] M Hrúz, P Campr, E Dikici, AA Kındıroğlu, Z Krňoul, A Ronzhin, H Sak, D Schorno, H Yalçın, L Akarun, O Aran, A Karpov, M Saraçlar and M Železný. Automatic fingersign-to-speech translation system. *J. Multimodal User Interfaces* 2011; **4**, 61-79.
[4] AB Jmaa, W Mahdi, YB Jemaa and AB Hamadou. A new approach for hand gestures recognition based on depth map captured by RGB-D camera. *Comput. Sist.* 2016; **20**, 709-21.
[5] P Goh and EJ Holden. Dynamic fingerspelling recognition using geometric and motion features. *In*: Proceedings of the International Conference on Image Processing, Atlanta, GA, USA. 2006, p. 2741-44.
[6] S Liwicki and M Everingham. Automatic recognition of fingerspelled words in british sign language. *In*: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Miami, FL, USA. 2009, p. 50-7.
[7] P Nakjai and T Katanyukul. Hand sign recognition for thai finger spelling: An application of convolution neural network. *J. Signal Process. Syst.* 2019; **91**, 131-46.
[8] P Nakjai and T Katanyukul. Automatic hand sign recognition: Identify unusuality through latent cognizance. *In*: Proceedings of the Artificial Neural Networks in Pattern Recognition, Siena, Italy. 2018, p. 255-67.
[9] A Laurent, S Meignier and P Deléglise. Improving recognition of proper nouns in ASR through generating and filtering phonetic transcriptions. *Comput. Speech Lang.* 2014; **28**, 979-96.
[10] P Nakjai, P Maneerat and T Katanyukul. Thai finger spelling localization and classification under complex background using a YOLO-based deep learning. *In*: Proceedings of the ACM International Conference Proceeding Series, Australia. 2019, p. 230-3.
[11] J Redmon and A Farhadi. YOLO9000: Better, faster, stronger. *In*: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA. 2017, p. 6517-25.
[12] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. *In*: Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA. 2015, p. 1-14.
[13] A Sidig, H Luqman and S Mahmoud. Arabic sign language recognition using optical flow-based features and HMM. *In*: Proceedings of the International Conference of Reliable Information and Communication Technology, Johor Bahru, Malaysia. 2018, p. 297-305.
[14] A Moryossef, I Tsochantaridis, R Aharoni, S Ebling and S Narayanan. Real-time sign language detection using human pose estimation. *In*: Proceedings of the European Conference on Computer Vision, Glasgow, UK. 2020, p. 237-48.
[15] Z Cao, T Simon, SE Wei and Y Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. *In*: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA. 2017, p. 1302-10.

[16] N El-Bendary, HM Zawbaa, MS Daoud, AE Hassanien and K Nakamatsu. ArSLAT: Arabic sign language alphabets translator. *In*: Proceedings of the International Conference on Computer Information Systems and Industrial Management Applications, Krakow, Poland. 2010, p. 590-5.
[17] M Mohandes, S Aliyu and M Deriche. Prototype Arabic sign language recognition using multi-sensor data fusion of two leap motion controllers. *In*: Proceedings of the IEEE 12th International Multi-Conference on Systems, Signals & Devices, Mahdia, Tunisia. 2015, p. 1-6.
[18] W Jiangqin and G Wen. The recognition of finger-spelling for Chinese sign language. *In*: Proceedings of the International Gesture Workshop, London, UK. 2001, p. 96-100.
[19] N Mukai, N Harada and Y Chang. Japanese fingerspelling recognition based on classification tree and machine learning. *In*: Proceedings of the Nicograph International, Kyoto, Japan. 2017, p. 19-24.
[20] HTC Machacon and S Shiga. Recognition of Japanese finger spelling gestures using neural networks. *J. Med. Eng. Technol.* 2010; **34**, 254-60.
[21] M Suwanarat and C Reilly. *The Thai sign language dictionary*. Thailand: National Association of the Deaf in Thailand, 1986.
[22] D Guo, W Zhou, H Li and M Wang. Hierarchical LSTM for sign language translation. *In*: Proceedings of the 32nd AAAI conference on artificial intelligence, USA. 2018, p. 6845-52.
[23] C Zhou, C Sun, Z Liu and FCM Lau. A c-lstm neural network for text classification. *arXiv* 2015. http://arxiv.org/abs/1511.08630.
[24] I Sutskever, O Vinyals and QV Le. Sequence to sequence learning with neural networks. *In*: Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, Canada. 2014, p. 3104-12.
[25] X Tian, J Zhang, Z Ma, Y He, J Wei, P Wu, W Situ, S Li and Y Zhang. Deep LSTM for large vocabulary continuous speech recognition. *arXiv* 2017. https://arxiv.org/abs/1703.07090.
[26] N Laokulrat, S Phang, N Nishida, R Shu, Y Ehara, N Okazaki, Y Miyao and H Nakayama. Generating video description using sequence-to-sequence model with temporal attention. *In*: Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan. 2016, p. 44-52.
[27] P Nakjai, J Ponsawat and T Katanyukul. Latent cognizance: What machine really learns. *In*: Proceedings of the 2nd International Conference on Artificial Intelligence and Pattern Recognition, Beijing, China. 2019, p. 164-9.