TFS recognition using MediaPipe Hands

No Author Given

No Institute Given

Abstract. Thai Finger Spelling (TFS) is an essential part of Thai sign language, adopted nationwide in deaf education and community in Thailand. TFS is for a spelling part of the language and is used for spelling out proper names. To handle a large set of Thai alphabets, TFS employs various signing schemes.
Some of TFS schemes, i.e., single-hand schemes, have been actively studied in previous studies, while others, i.e., point-on-hand schemes, are largely unexplored. Point-on-hand (PoH) signings rely on precise localization of key landmarks on a palm and fingers. This nature of PoH schemes demands a more fine-grained approach than a classifier-based method, widely used for single-hand schemes.
This article examines application of MediaPipe Hands (MPH) to an automatic recognition of signings in TFS, particularly PoH signings. Our work studies effectiveness and limitations of MPH under the context of TFS sign recognition. For PoH sign inference on half-body images with natural background, our best-performing MPH has reached 60.4% accuracy, conferring to 42.95% accuracy shown using a trained VGG classifier. Our findings reveal MPH limitations particularly on handling hand-hand interaction (contributing to 32.8% error). Since PoH signings involve a high degree of hand-hand interaction (average 4.37%; max 89.88%), MPH in its current state is not recommended for addressing PoH sign recognition. In addition, %palm-overlapping is proposed to quantify a degree of hand-hand interaction.

Keywords: Sign language recognition · Thai sign language · Thai finger spelling · MediaPipe Hands · Keypoint localization · hand-hand interaction issues.

1 Introduction

This article addresses an automatic recognition of a static point-on-hand scheme of Thai Finger Spelling (TFS)—a spelling part of modern Thai Sign Language (TSL). Covering one of the largest sets of alphabets in sign languages[6], TFS signings employ four signing schemes based on their dynamicity and a number of hands involved. Single-hand (SH) signings employ only one hand, either left or right, to spell out consonants. To spell out non-consonants, e.g., vowels, intonations, and special marks, point-on-hand (PoH) signings employ both hands: one pointing to a key location on another. SH or PoH signings can be static, i.e., using one still posture, or dynamic, i.e., using multiple still postures or using

movement. These signing schemes pose many unique technical challenges rarely present in other languages. E.g., a PoH signing requires more precise localization of hand keypoints than what required by SH signing, widely used in many sign languages[6], including American, Arabic, Chinese, and Japanese signings.

Our work investigates MediaPipe Hands (MPH[1]) for SH and PoH signings. Our main contributions are (1) to address the PoH scheme, which has not been previously studied, (2) to provide an insight into factors and mechanisms related to addressing the PoH scheme, and (3) to propose % palm-overlapping as a metric to quantify hand-hand interaction. Hand-hand interaction relates to difficulty of sign recognition. The metric facilitates better comprehension of different results on different sign languages.

2 Related Works

Thai Finger Spelling (TFS) recognition is still in an early stage of development. While TFS employs various signing schemes, previous works[2–6] address only single-hand schemes covering only consonant part of the spelling. Automatic recognition of signs in other schemes (i.e., static- and dynamic-point-on-hand schemes covering non-consonants) has been left unexplored.

Early works[2–4] frame the task as classifying a signing posture, rather than an alphabet. Still addressing single-hand schemes, later works[5, 6] have progressed to take signing video in and give out alphabets or even words[6] (consonant-only words are not un-common in Thai) with the modest success[6]. All were on single-hand signings, but under various settings and achieve various degrees of success. Some have run into technical issues[2], e.g., often-confusing signs and non-signing postures. Often-confusing signs are, e.g., signs 'M' and 'N'[2], whose difference is in the precise location of a thumb in their fist postures. To resolve such confusion, we need a more precise approach than classification. Another emerging issue—non-signing postures—may appear unintended and could confuse the system. This issue of non-signing postures relates directly to limitation of multi-class classification. It may be addressed using different problem framing or re-examining a theoretical aspect of multi-class classification itself[7, 8].

One of the alternatives to classification is keypoint localization[9]. A notable off-the-shelf one is MediaPipe Hands (MPH[1]), which has recently been publicly available. MPH—targeted for AR/VR applications—has taken a top-down design: performing hand detection then estimating hand keypoints on each detection. Its hand keypoint estimator follows a topology similar to Multiview Bootstrapping (MB[10]). While MB itself employs Convolutional Pose Machines (CPM[11]) for keypoint estimation, the essence of MB is indeed in using specialized multiview camera set and triangulation to provide annotated training data and its iterative method to improve training of the estimator.

Our article addresses TFS recognition, particularly point-on-hand (PoH) signings, using MPH. PoH signings pose a more serious requirement on preciseness than most single-hand signings. MPH provides estimated locations of hand keypoints. These finer-grained inference could address such requirements.

3 Background

Convolutional pose machines. Addressing pose estimation or body keypoint detection, Convolutional Pose Machine (CPM[11]) frames body keypoint detection as regression of score maps, when each map corresponds to a designated body part. A location with a high score on a map indicates a high likelihood that the corresponding body part locates there.

Specifically, given an RGB image $\mathbf{m} \in \mathbb{R}^{w \times h \times 3}$, the output from CPM is $\mathbf{S} \equiv \{\mathbf{s}_p\}_{p=1,\ldots,P+1}$ where P is a number of keypoints and $\mathbf{s}_p \in \mathbb{R}^{w' \times h'}$ is a final score map corresponding to the p^{th} keypoint. Note CPM estimates score maps for all P keypoints and an extra one for background. A location of the keypoint can be deduced from the score map: $\arg\max_{i,j} \mathbf{s}_p[i,j]$ with a scale adjustment.

To achieve accurate score maps, CPM uses multiple stages of convolution neural networks (CNNs) in a successive refinement manner. That is, the first stage estimates a set of score maps $\mathbf{S}_1 = q_1(f(\mathbf{m}))$, where $f(\cdot)$ is a CNN designed as a feature extractor and $q_1(\cdot)$ is a stage-1 CNN estimating score maps out of the features.

A subsequent stage-t ($t > 1$) produces a set of refined score maps

$$\mathbf{S}_t = q_t(f'(\mathbf{m}), \psi_t(\mathbf{S}_{t-1})), \tag{1}$$

where $q_t(\cdot,\cdot)$ is a stage-t CNN taking both image features and contextual features hinted by the previous stage prediction. A different feature extractor $f'(\cdot)$ can be used for later stages $t > 1$. CPM conceptualizes ψ_t to be mapping from previous estimation to contextual features. The last stage output represents the model output $\mathbf{S}_T \equiv \mathbf{S}$. Fig 1 illustrates this topology.

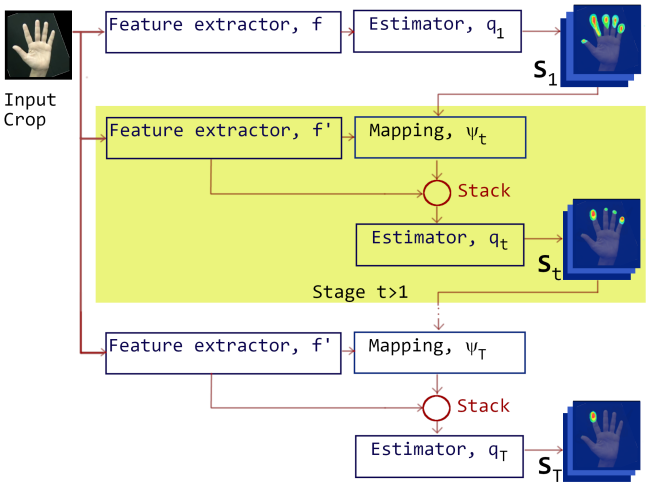

Fig. 1. Configured as a multi-stage CNN, each stage estimates a set of score maps each for each keypoint. A subsequent stage takes hints from its predecessor.

In its own implementation CPM takes an effective receptive field as the contextual mapping. It designs CNNs such that an effective receptive field of stage-2 is likely to cover an entire body, providing a necessary contextual information for capturing long-distance spatial dependencies in body pose. To achieve a large effective receptive field, CPM uses combination of pooling (stride > 1), increasing kernel sizes of convolution filters, and increasing convolution layers. Each factor comes with downsides, i.e., pooling costs spatial resolution, increasing kernel sizes costs more parameters, and increasing convolution layers risks the problem of vanishing gradients[16]. CPM uses pooling and increasing kernel sizes to an acceptable degree of their expense. A stage-wise design is intended to facilitate an intermediate feedback to counter vanishing gradients during training.

To train the multi-stage keypoint model, ground-truth score maps $\mathbf{S}^* = \{\mathbf{s}_p^*\}_{p=1,\ldots,P+1}$ are constructed from Gaussian basis functions whose centers are at ground-truth locations. Training was to minimize multi-stage loss $\mathcal{L} = \sum_{t=1}^{T} \ell_t$, where stage loss $\ell_t = \sum_{p=1}^{P+1} \sum_{i,j} \|\mathbf{s}_p^{(t)}[i,j] - \mathbf{s}_p^*[i,j]\|^2$ when $\mathbf{s}_p^{(t)}[i,j]$ is a score at location (i,j) of a stage-t map for the p^{th} keypoint.

MediaPipe Hands. MediaPipe Hands (MPH[1]) is a two-phase hand-keypoint estimator: detecting hands and estimating hand keypoints on each detected. The keypoints from MPH can be further processed to determine a TFS sign.

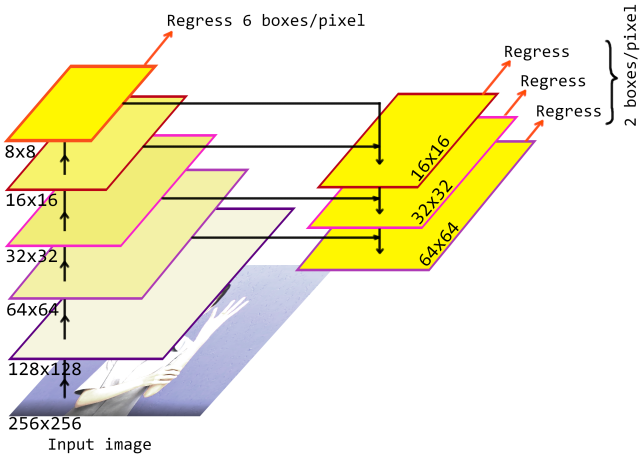

Fig. 2. Hand detection model employed in MPH. The upstream pathway is derived from BlazeFace[12], while the downstream pathway and lateral connections are taken after Feature Pyramid Network[13]. The four regressors share weights.

Intended to be an on-device real-time hand tracker, MPH uses various techniques for both detection quality and speed. To detect a hand, it follows BlazeFace[12] with a few tweaks. I.e., (1) the detector aims for a rigid-shape palm,

rather than an entire hand with articulated fingers. This reduces problem complexity and allows use of a square bounding box—reducing a number of localization variables. (2) Feature extraction is configured as a Feature Pyramid Network[13] to allow detecting various hand sizes with a target of $\sim 20\times$ scale span. (3) It is trained using focal loss[14] to mitigate an issue of extremely imbalanced data that negative examples of the background overwhelms positive examples of the foreground when using cross-entropy loss.

Fig 2 shows MPH palm detector model. The CNN configured similar to BlazeFace[12] extracts high-level features from an input image, shown as an upstream pathway on the left where spatial resolution shrinks as feature information progresses up to a higher level. To solve efficient multi-scale detection, an architecture following Feature Pyramid Network (FPN[13]) is employed. FPN provides high-level feature maps with different spatial resolutions, shown as a downstream pathway on the right. The key mechanism of FPN is using a combination of an up-sampled low-resolution high-level feature map and a high-resolution low-level feature map to produce a high-resolution high-level feature map, ready for prediction. As BlazeFace is in turn adapted from SSD[15], object detection is framed as regression of localization variables with respect to default bounding boxes or anchors. MPH is configured to regress 6 bounding boxes/pixel from 8×8 map and 2 boxes/pixel from the other resolutions. This makes a total prediction of 11,136 bounding boxes.

Specifically, given an RGB image $\mathbf{x}$, a model regresses a collection of detections or bounding boxes, $\{\mathbf{b}_i\}_{i=1,\ldots,B} = palm(\mathbf{x})$, where B is a total number of bounding boxes and $palm(\cdot)$ is a palm detection model, whose topology is shown in Fig 2. Each detection $\mathbf{b}_i$ provides $p^{(i)}, \hat{x}_c^{(i)}, \hat{y}_c^{(i)}, \hat{x}_1^{(i)}, \hat{y}_1^{(i)}$ presenting a confidence score, x- and y-coordinates of a center and a reference point. These coordinates are relative to the anchor position. The global coordinates can be computed, i.e., $x_c^{(i)} = \hat{x}_c^{(i)} \cdot s_x^{(i)} + x^{(i)}$; $y_c^{(i)} = \hat{y}_c^{(i)} \cdot s_y^{(i)} + y^{(i)}$; $x_1^{(i)} = \hat{x}_1^{(i)} \cdot s_x^{(i)} + x^{(i)}$; $y_1^{(i)} = \hat{y}_1^{(i)} \cdot s_y^{(i)} + y^{(i)}$, where $s_x^{(i)}$, $s_y^{(i)}$, $x^{(i)}$ and $y^{(i)}$ are scales and offsets of the i^{th} anchor. Recall that MPH employs only square bounding boxes. Locations of the boxes are provided by the centers. The reference points are to provide box sizes and rotations. Box size $s = 2\sqrt{(x_c - x_1)^2 + (y_c - y_1)^2}$. Rotation $r = \tan^{-1}(y_c - y_1)/(x_c - x_1)$. The superscript was omitted for tidiness.

Redundant detections, identified by criteria on intersect-over-union, are handled through weighted non-maximum suppression, where redundant detections are weighted averaged using normalized confidence scores for weights. This version of non-maximum suppression was proposed in BlazeFace[12] (called "tie-resolution") to mitigate a jitter issue in video applications.

Given these detections (as oriented bounding boxes), hand regions can be cropped out and passed on to the hand keypoint estimator.

Regarding training a hand detector, focal loss[14] is used to fully utilize training data without suffering from the extreme foreground-background class imbalance. The focal loss lessens the effect of easy examples (e.g., easy but overwhelmingly abundant background patches), which results as emphasizing the effect of hard examples during training. The focal loss is defined as $\text{loss}(p_t) =$

$-(1-p_t)^\gamma \log(p_t)$ where $p_t = p$ if ground truth is 1 (object is present) and $p_t = 1 - p$ otherwise, when p represents a predicted confidence score from the model. I.e., a good prediction would have p_t close to 1 and a bad prediction would have p_t close to 0. The focusing parameter $\gamma \geq 0$ is user-specific. The closer is γ to unity, the more focus on hard examples does the training.

To do keypoint estimation, with a Region of Interest (RoI) of a detected palm from the previous phase, MPH crops an image by an enlargement of the RoI so that the crop is likely to have the entire hand. With the cropped image as its input, MPH keypoint estimator predicts 21 hand keypoints along with handedness and hand presence—the probability that a well aligned hand is really present in the crop—using a CNN with three dedicated prediction heads for 3D hand locations, handedness, and hand presence. These prediction heads are also convolutional layers, but trained with its own datasets.

Specifically, the input crop is resized to 256×256 and processed through convolutional layers acting collectively as a feature extractor, before the prediction heads. The prediction heads compute hand keypoints, handedness, and hand presence. The hand keypoints can be seen as a hand tensor $\mathbf{A} = [\mathbf{a}_0, \ldots, \mathbf{a}_{20}]$, where $\mathbf{a}_i \in \mathbb{R}^3$ is a 3D coordinate of the i^{th} keypoint. Handedness h and hand presence c are both scalar and regulated to be within $[0, 1]$.

Applications to sign language recognition. MPH has been subjects in some finger-spelling-recognition studies[17, 18]. Shin et al.[17] address American Finger Spelling (AFS) using hand keypoints obtained from MPH to compute distance and angle features, before classify them into AFS signs using Support Vector Machine (SVM[19]) and light Gradient Boosting Machine (GBM[20]). Their system was shown to be accurate across various datasets. Halder and Tayade[18] use MPH along with various classifiers to addresss American, Indian, Italian and Turkish Finger Spelling with up to 99% accuracy.

Sign languages addressed above have only static signing scheme without hand-hand interaction. TFS has four schemes covering static and dynamic postures with various degrees of hand-hand interaction (hands in close proximity and even occluding each other). Therefore, addressing TFS signing involves challenges not present in these works.

Although MPH could provide a fast development for sign recognition, many issues remain unexplored. E.g., CPM—a key milestone in pose estimation—was reported degradation when target objects are in close proximity. MB, whose topology is same as MPH, has been shown to underperform when there is hand-hand interaction, while it can handle single-hand and hand-object cases pretty well.

In addition, MPH top-down design may suffer from (1) loss of global context when performing keypoint estimation and (2) violation of a single-hand assumption on which its keypoint estimator is developed. Therefore, while MPH has been shown viable in many studies, its design has not been tested against the demanding of TFS recognition. How suitable MPH is for TFS recognition is largely unknown. Our objective is to address sign recognition of a static-PoH scheme as well as to explore this new technology under TFS context.

4 Methodology

An approach based on MediaPipe Hands (MPH) is examined on the three TFS signing schemes[1]: static- and dynamic-single-hand schemes and a static-PoH scheme. A VGG classifier (following V-TFS[3]) is examined on a static point-on-hand scheme to provide a perspective on this new setting.

Our MPH application to TFS sign recognition. Two dedicated pipelines (Fig 3) are designed. The first pipeline, denoted MPH-ANN, employs Artificial Neural Network (ANN) intended for both single-hand schemes. The second pipeline, denoted MPH-RB, employs a rule-based method crafted for a static-point-on-hand (PoH) scheme. A simple decision rule is employed to pre-screen the signing so that a scheme-specific pipeline can be properly applied. Given a number of appearing hand H from MPH, if H is one, the single-hand case is assumed: MPH-ANN pipeline is activated. Otherwise (H is two), the point-on-hand case is assumed: MPH-RB pipeline is activated.

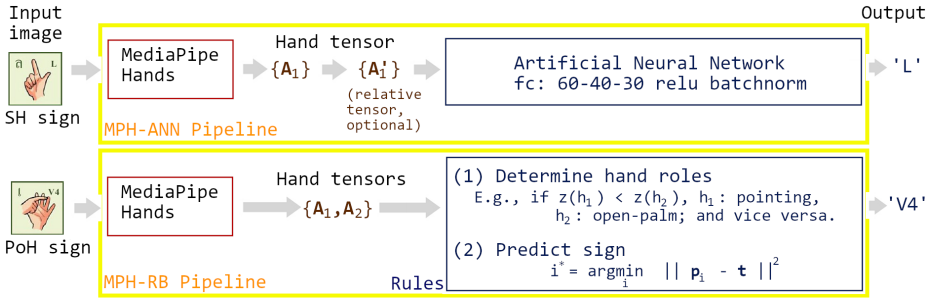

Fig. 3. MPH-ANN pipeline for static- and dynamic-single-hand schemes (top) and MPH-RB pipeline for a static-point-on-hand scheme (bottom). Both take an image as input and give a TFS signing posture as output. MPH provides hand tensors for detected hands. A number of hand tensors indicates which pipeline should be called.

MPH-ANN pipeline. Addressing single-hand schemes, a hand tensor from MPH is passed on to an ANN model to predict a TFS sign. The MPH-ANN pipeline is designed for a still image. As the previous work[3], a dynamic-single-hand scheme is simplified, i.e., key frames are used instead of an entire series of frames.

We explore two alternatives: (1) an absolute version that passes on absolute coordinates of the hand tensor directly to the ANN and (2) a relative version that computes relative coordinates before passing them to the ANN.

The absolute version takes a 63-element vector from a 21×3 hand tensor $\mathbf{A} = [\mathbf{a}_0, \ldots, \mathbf{a}_{20}]$ representing 21 absolute three-dimensional coordinates of the

[1] Details of TFS signings are described in [2]. They refer to PoH scheme as "two-hand scheme".

keypoints. The relative version takes a 60-element vector from a 20×3 relative hand tensor $\mathbf{A}' = [\mathbf{a}'_1, \ldots, \mathbf{a}'_{20}]$ whose coordinates are relative to a reference. I.e., $\mathbf{a}'_i = \mathbf{a}_i - \mathbf{r}$ for any i^{th} keypoint that is not the reference, where $\mathbf{r}$ is the reference coordinate—the wrist keypoint is chosen for our experiment.

This effectively makes the ANN of the relative version have a little less number of parameters. We hypothesize that this relative version could provide a superior result due to less noisy information feeding to the ANN. Both versions use three fully connected layers with 60, 40 and 30 units as their hidden layers. The output layer uses softmax as its activation function, the hidden layers use rectified linear unit (relu) with batchnorm.

MPH-RB pipeline. Addressing a static-PoH scheme, deterministic rules are specified to deduce a TFS sign from the hand information obtained from MPH. Given hand tensors $\{\mathbf{A}_1, \mathbf{A}_2\}$ from MPH, (1) MPH-RB determines hand roles: which hand assumes an open-palm posture and which assumes a pointing posture. Various role criteria have been explored. E.g., finger-vector criteria assign finger-vectors of every finger using keypoints. Then, relative directions among the vectors—obtained by vector operations—are used to determine roles. Depth criterion uses depths (z-coordinates provided by MPH): a hand with a smaller average z (appearing on the front) presumes a pointing role and the other presumes an open-palm role. Among the criteria, the depth criterion is found to be the best-performing and our presentation is based on this criterion. Note that despite being our best-performing criterion, depth criterion is susceptible to non-signing postures.

Once hand roles are realized, (2) MPH-RB determines the sign from the nearest keypoint on the open-palm hand to the keypoint of the pointing tip. I.e., sign index $i^* = \arg\min_i \|\mathbf{p}_i - \mathbf{t}\|^2$ where $\mathbf{p}_i$ represents an x-y location of the i^{th} keypoint on the open-palm hand and $\mathbf{t}$ is an x-y location of the pointing tip.

Datasets. The main dataset, called PoH dataset, has 851 and 149 images for training and testing, respectively. Each image is of 720×720 pixels. Used for all our experiments on PoH signing, PoH dataset covers 11 signings of TFS point-on-hand scheme. Another dataset, called single-hand-signing (SHS) dataset, has 2518 and 1801 images for training and testing, respectively. Each image is of 1280×720 pixels. Used for our experiment on single-hand signing, SHS dataset covers 30 signings from TFS static- and simplified dynamic-signle-hand schemes.

Results. Fig 4 summarizes the overall performance of MPH-ANN on single-hand signing recognition (left) and one of MPH-RB on PoH signing recognition (right). Best performing settings achieve accuracy of 61.15% (relative-version MPH-ANN on single-hand signing) and 60.40% (depth-criterion MPH-RB on PoH signing).

As our emphasis on PoH signing, the error investigation has been conducted. Failing to detect hands seems largely contributed to the resulting performance. Fig 5 (left) shows error analysis by a number of hands detected by MPH.

As hand-hand interaction being an issue of concern (as discussed in §3), to measure hand-hand interation, we define %palm-overlapping as a percentage

of an intersection area—between two circular areas each approximates a palm area of each hand—over an area of an open-palm hand. It is designed to be easy to obtain as well as reflecting the degree of hand-hand interaction. Fig 5 (right) shows %palm-overlapping of different groups: a group of images that MPH cannot detect any hand ($H = 0$), a group of images that MPH can detect only one hand ($H = 1$), and a group of images that MPH can detect both hands ($H = 2$). Mann-Whitney U test confirms the differences at p-values of 6.1775e-27 ($H = 2$ vs $H = 1$) and of p=0.000013 ($H = 2$ vs $H = 0$). That is, a high percentage of palm overlapping—reflecting a high degree of hand-hand interaction—causes MPH to fail to detect one or both hands (contributing to 1.3% + 31.5% = 32.8% error). With less than two hands detected, PoH sign cannot be inferred.

In addition to palm overlapping, there seems to be other factors causing recognition error as shown in 6.7% incorrect recognition (Fig 5, left) even when both hands are detected. Fig 6 (left) examines if error in case $H = 2$ still associates with %palm-overlapping. The difference shown in Fig 6 (left) seems less apparent. When both hands are detected, at 0.05 p-value Mann-Whitney U test cannot confirm the difference in palm overlapping between correct and incorrect recognition (p-value 0.882540). That is, once MPH can detect two hands, error is not likely to be associated with palm overlapping.

Fig 6 (right) reveals another factor—a localization error on the pointing tip. We measure this error by %tip-location error, $\epsilon = \|\mathbf{t} - \mathbf{t}'\|^2/D \times 100\%$ where $\mathbf{t}$ and $\mathbf{t}'$ are predicted and true tip locations, respectively; D is an approximate palm length from top to bottom. Mann-Whitney U test confirms the difference in tip location errors with p-value of 0.0017. That is, %tip-location error found in incorrect predictions are significantly higher than one found in correct prediction. Failing to precisely locate a keypoint of a pointing tip is a cause of wrong inference when both hands have been detected (contributing to ∼6.7% error).

Fig 7 shows some examples with error predictions. After examining, the errors can be categorized into: tip location errors, palm-point errors, errors due to our minimum-distance assumption. (1) The tip location error seems to be more pronunced when there is hand-hand interaction such that a pointing finger crosses either other fingers or other part of the body with similar skin color. (2) The palm-point error, which MPH mislocates the keypoints on a open-palm hand, is seen in a blurry open-palm hand with highly hand-hand interaction. (Fig 7, 3rd from left, lower row.) (3) In addition, some errors arise even when MPH has delivered sufficiently accurate locations of all keypoints. (Fig 7, 2nd from left, lower row.) These disclose lacking in our MPH-RB pipeline. MPH-RB decides a sign on minimum-distance basis: predicting the TFS sign whose corresponding keypoint is closest to the pointing tip. This minimum-distance assumption is insufficient. When a signer wants to point to a target, sometimes the target might not be the closest one to the pointing tip, but it is in the projected direction from the pointing. Therefore, a more practical sign-inference criteria should take this fact into account.

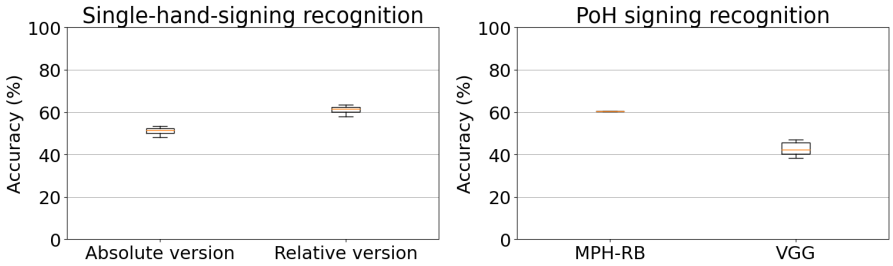

Fig. 4. Left: performance of MPH-ANN on single-hand signing. Right: performances of MPH-RB and VGG classifier on PoH signing.

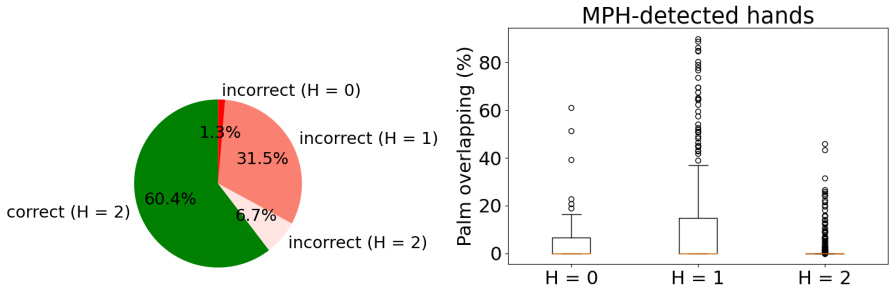

Fig. 5. Left: error analysis. Right: numbers of hands detected versus %palm-overlapping.

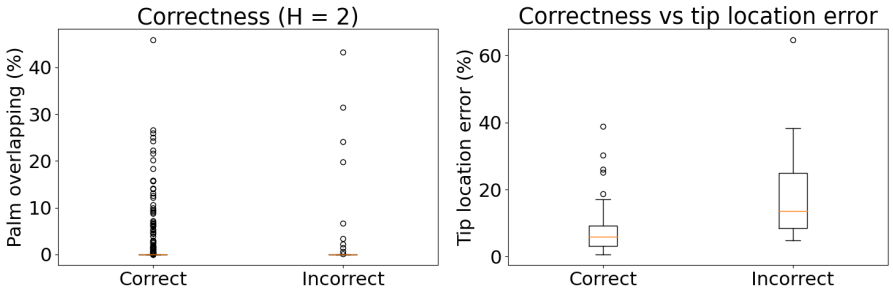

Fig. 6. Correctness in the two-hand case versus %palm-overlapping (left) and %tip-location error (right).

Regarding the difficulty of the dataset, the % palm overlapping of our dataset is 4.38 on average (min: 0, max: 89.88, std 13.4), c.f. 0 % palm overlapping for all single-hand signings, used in many other languages, e.g., American, Arabic, Chinese.

In addition to our main results, our side experiment reveals limitation of MPH on determining handedness: recognizing a hand as a left or right one.

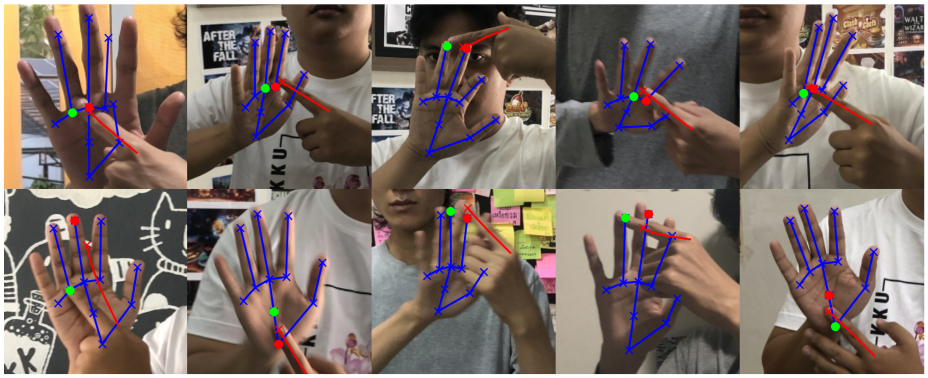

Fig. 7. Examples of sign-recognition error when two hands are detected.

We have found that MPH handedness recognition performance deteorates when both hands are presented and particularly when one hand is presented in its front view and another in its back view.

Note that experimental settings, e.g., controlled background (plain background), may largely affect the results. MPH is an off-the-shelf software. It has been trained on massive images, presumably with natural background, then it is fixed. We cannot fine tune it. Comparing performance of MPH to a customized model under an un-natural setting, e.g., a plain background, may come up with totally different conclusion[2].

Data, code, and supplementary materials can be found at
`http://github.com/s0ngkran/MPH_INVESTIGATION_ON_TFS`

5 Conclusion and Discussion

Our study has investigated applications of MPH to TFS signing recognition and revealed mediocre performance in both single-hand and point-on-hand signing recognition. A feature-engineering technique could provide mitigation, e.g., using relative locations, rather than absolute coordinates, shows significant performance improvement (Fig 4, left). Further investigation reveals limitations of MPH: specifically MPH could not perform well under hand-hand interaction (contributing to 32.8% error). We have proposed palm overlapping to measure a degree of hand-hand interaction. Finger crossing—finger interaction with either other fingers or other parts with similar skin color—is another type of hand-hand interaction, which could not be sufficiently measured using palm overlapping. A practical issue of pointing—sometimes meant for a target in the pointing direction, not for the closest one to the pointing tip—is realized and recommended to take into account when addressing point-on-hand signing recognition.

[2] Another of our side experiment under a plain background has shown just that.

Our dataset is found to have palm overlapping of 4.37% on average (89.88% max). Given a nature of TFS signing, MPH in its current state is not recommended. A bottom-up design of Part Affinity Fields (PAF[9]), which emphasizes relations among keypoints, seems to be more capable of handling hand-hand interaction and a better prospect for TFS sign recognition. In addition, to properly address the dynamicity of TFS, a sequence-capable model, like transformer[21], should be explored.

References

1. Zhang et al. Mediapipe hands: on-device real-time hand tracking. ArXiv. 2020;abs/2006.10214.
2. Nakjai, Katanyukul. Hand sign recognition for thai finger spelling: an application of convolution neural network. Journal of Signal Processing Systems 91(2), 2019.
3. Nakjai, Katanyukul. Automatic hand sign recognition: identify unusuality through latent Cognizance. ANNPR 2018.
4. Nakjai et al. Thai finger spelling localization and classification under complex background using a YOLO-based deep learning. ICCMS 2019.
5. Pariwat, Seresangtakul. Multi-stroke Thai finger-spelling sign language recognition system with Deep Learning. Symmetry 13(2), 2021.
6. Nakjai, Katanyukul. Automatic thai finger spelling transcription. Walailak Journal of Science and Technology 18(13), 2021.
7. Nakjai et al. Latent cognizance: What machine really learns. AIPR 2019.
8. Katanyukul, Nakjai. Recognition awareness: adding awareness to pattern recognition using latent cognizance. Heliyon 8(4), 2022.
9. Cao et al. Openpose: realtime multi-person 2d pose estimation using part affinity fields. IEEE TPAMI 43(1), 2021.
10. Simon et al. Hand keypoint detection in single images using multiview bootstrapping. CVPR 2017.
11. Wei et al. Convolutional pose machines. CVPR 2016.
12. Bazarevsky et al. BlazeFace: sub-millisecond neural face detection on mobile gpus. CVPR Workshop 2019.
13. Lin et al. Feature pyramid networks for object detection. CVPR 2017.
14. Lin et al. Focal loss for dense object detection. IEEE TPAMI 42(2), 2020.
15. Liu et al. SSD: single show multibox detector. ECCV 2016.
16. Glorot, Bengio. Understanding the difficulty of training deep feedforward neural networks. AISTATS 2010.
17. Shin et al. American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation. Sensors 21(17), 2021.
18. Halder, Tayade. Real-time vernacular sign language recognition using mediapipe and machine learning. International Journal of Research and Reviews 2(5), 2021.
19. Boser et al. A training algorithm for optimal margin classifiers. CLT workshop 1992.
20. Ke et al. Lightgbm: a highly efficient gradient boosting decision tree. NIPS 2017.
21. Vaswani et al. Attention is all you need. NIPS 2017.