A Survey on Thai Input Methods on Smartphones

1. Introduction

Recently, smartphones have, undoubtedly, become popular among mobile phone users. This is because of their functionality and computing power. Unfortunately, most of the smartphones available on the market have not been equipped with physical keyboards. Text input on smartphones is typically performed via a virtual keyboard, which is a software component displaying a group of key buttons arranged in the similar way to the commonly-used keyboard. A character can be input by touching on the button corresponding to the particular character. However, the small size of the touchscreen makes the size of each button small. It is generally smaller than the size of most users’ finger tip. This then causes difficulty in choosing characters, and typically results in errors or typos. Moreover, the virtuality of the keyboard does not provide users the feeling of touch. This is also another cause of errors or typos.

Various techniques have been proposed to improve the performance of the virtual keyboards in term of usability, accuracy, and typing speed. Word completion techniques and next word prediction techniques are among those. Word completion techniques use typically a list of known words and the approximate string matching algorithm to generate a list of word candidates from the string input. Users can then choose their preferred words from the list. This helps reduce the number of input keystrokes and correct some mistyped characters. The next word prediction techniques generally utilize the word usage statistics and some language models to generate a list of the possible next words. This similarly improves the performance in terms of typing speed and usability.

The Thai writing system forms words by using 44 consonants (aka consonant letters in some literatures), 21 vowels (aka vowel letters), and a number of punctuation marks. Thai text can be input via a physical keyboard in a similar manner to inputting English text. However, two letters are assigned to a button in the keyboard due to the larger number of letters. Thus, using virtual keyboards without additional converter software, as in Japanese or Chinese, is the common technique to input Thai text on smartphones. In fact, virtual keyboards in the Thai language are used in the same manner as in physical keyboards. Each button, which can be thought of as group of pixels, is assigned to specific Thai text. A significant difference in terms of the number and size of buttons can easily be seen here.

Recently, a number of techniques have been proposed to improve the performance of Thai virtual keyboards and their input methods. Some of the techniques are designed to fit the characteristics of the Thai writing system. Table 1 summarizes Thai virtual keyboards currently available as well as input methods. We compare them according to their features and techniques used to improve the text inputting performance. Some of them are proprietary software where the techniques are not revealed. We thus mark their features...
Table 1  Summary of the techniques reviewed in this paper.

<table>
<thead>
<tr>
<th>Name</th>
<th>Layout</th>
<th>Character Candidate Generation Technique</th>
<th>Word Candidate Generation Technique</th>
<th>Candidate Ranking Technique</th>
<th>Model Update</th>
<th>Next Word Prediction Technique</th>
<th>Other Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS Keyboard</td>
<td>Ketmanee-based / Four-row</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>word correction</td>
</tr>
<tr>
<td>Janpinijrut et al.[7]</td>
<td>Ketmanee-based</td>
<td>distance-based</td>
<td>--</td>
<td>word frequency</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Janpinijrut et al.[8]</td>
<td>Vowel-separated</td>
<td>distance-based</td>
<td>prefix matching</td>
<td>word frequency</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Ballungpattama et al.[9]</td>
<td>Consonant-only</td>
<td>--</td>
<td>approx. matching</td>
<td>word frequency</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Phanchaipetch and Nattee[10]</td>
<td>Ketmanee-based</td>
<td>bivariate normal dist.</td>
<td>prefix matching</td>
<td>word frequency</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Phanchaipetch and Nattee[11]</td>
<td>Ketmanee-based</td>
<td>bivariate normal dist.</td>
<td>prefix matching</td>
<td>word freq. + user list</td>
<td>included</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Ballungpattama et al.[12]</td>
<td>Consonant-only</td>
<td>--</td>
<td>approx. matching</td>
<td>word freq. + user list</td>
<td>included</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

2. Thai Keyboard Layouts on Smartphones

We first review a number of keyboard layouts used for Thai input methods on smartphones. Typically, we can categorize the layouts into two categories, i.e. the Ketmanee-based layout and the other layout.

2.1 Ketmanee-Based Layouts

The Ketmanee keyboard layout is the standard Thai layout for computers announced by the Thai Industrial Standard Institute (TISI) [14]. All the physical keyboards sold in Thailand are based on this layout. Most Thai input methods on smartphones are designed based on this layout since it significantly reduces the period of time required by the users to get familiar with the layout. Some input methods use variants of the Ketmanee layout to improve their usabilities by removing some rarely-used buttons. Figure 2 shows the Thai input methods using the variants of Ketmanee as their keyboard layouts. The input methods shown in subfigures (a) – (f) are designed for Android OS. The one in subfigure (g) is the standard keyboard provided by Apple iOS.

Here are the issues that we can observe from Ketmanee-based layouts:

1. All the layouts arrange the keys into five rows based on the original Ketmanee layout. This is different from the Qwerty layouts for English characters where the alphabet characters can be fit in four rows, and the top row is dedicated for the numeric and punctuation keys. According to this reason, Thai keyboard layouts require more space of the screen to display the keys. Some layout designers handle this problem by decreasing the height of the keys to make them fit in the same height as the Qwerty layout. This has resulted in smaller key buttons. Some designers decide to add an additional row to the Qwerty layout and place the numeric and punctuation keys in this row.

2. Most of the layouts place the 'backspace' key at the fourth row of the layout. This is different from the original Ketmanee layout where the 'backspace' key is placed in the first row. The main reason behind this change is the width of the keys. The original Ketmanee layout was designed to place the largest number of keys in the first row. This results in the smaller width of each key. To relax this problem, the designers of many layouts move the 'backspace' key to the fourth row where the number of keys is fewer.

2.2 Other Layouts

Since the Ketmanee layout was originally designed for mechanical typewriters, many limitations of virtual keyboard
were not considered at the time of the creation of Ketmanee. A number of Thai keyboard layouts have been proposed for smartphones to improve the performance in various aspects. However, users may not be familiar with the newly-designed keyboard layouts. It may take some time for users to remember the positions of the characters and utilize the new layout effectively.

Four-row Thai Keyboard on iOS was designed to make Thai characters arranged in four rows, similar to the Qwerty layout for English, as shown in Fig. 2 (h). It is done by removing a number of vowels and punctuation marks from the layout. They are grouped together and represented by one key on the layout. A user can choose those vowels and punctuation marks by pressing on the key. The keyboard will show a dialog box that allows the user to choose a character. Moreover, a number of infrequently-used letters are moved from the main layout to the shifted layout to reduce the number of buttons.

Conslide is a Thai input method proposed by Sukswai and Piromsopa [13]. It is designed to allocate less space on the touchscreen by providing two types of buttons, i.e., presskey and slidekey. A presskey acts the same as an ordinary button as in the other virtual keyboards. A slidekey displays up to 6 characters on the button. As hinted by its name, the user can choose to input a character on the slidekey by sliding. The direction is detected by Conslide and a character is selected according to the direction. Therefore, the layout for Conslide requires only two rows of buttons as shown in Figs. 3 and 4.

Conslide has introduced a number of new ideas to improve Thai input methods. However, its performance in terms of typing speed may not be the best, compared to the others. Conslide provides a layout that is completely different from the existing Thai layout. Users may need a period of time to get used to the unique layout.

Vowel-separated Thai keyboard layout is a Thai keyboard layout and input method proposed by Janpinijrut et al. [8]. The layout was designed based on the characteristics of the Thai writing system. Figure 5 shows a Thai phrase where the consonants are shown in gray, the vowels and punctuation marks are in black. It can be seen that the vowels and punctuation marks can appear on the left of, the right of, above, or below a consonant. The layout then arranges those vowels and punctuation marks based on their relative positions to the consonant. The consonants are arranged in the central part of the layout according to the Ketmanee layout. Figure 6 shows the layout of this virtual
Separating the consonants from the vowels and punctuation marks improves the usability and convenience of the keyboard, as all the Thai native speakers know where to find the vowels and punctuation marks. Originally, this keyboard layout was designed for people with disabilities. From their experiments, the movement of fingers required is decreased, compared to Ketmanee layout keyboards. However, users may need a short period of time to get familiar with it.

Consonant-only Thai keyboard layout was proposed by Ballungpattama et al. [9], [12]. This layout consists of only the Thai consonants. This drastically reduces the number of keys on the layout. It results in bigger button size and more convenience for users, to select the preferred characters. However, the layout cannot be used to input new words since any vowels are not included in the layout. A technique to generate a list of candidate words has been proposed together with the layout. We will review the technique in the next section. Figure 7 shows the consonant-only Thai keyboard layout.

3. Techniques for Improving Thai Text Input Methods

Several techniques have been proposed to improve the performance of the Thai keyboard in various aspects. Combining these techniques with the Thai virtual keyboard, we can consider each Thai virtual keyboard as an input method.

Before reviewing available techniques, we need to understand the overall framework for an input method. Figure 8 shows the framework of Thai input methods on smartphones. This framework conforms to the multi-channel architecture for sokgraph recognition proposed by Kristensson and Zhai [15]. With this framework, we can review available techniques and identify where in the framework a particular technique is located. Hopefully this will help with the future development of the virtual keyboards and input method.

---

![Fig. 7 Consonant-only Thai keyboard (Source: [12]).](image)

![Fig. 8 The framework of Thai input method on smartphones.](image)

---

where,
- Coordinate Buffer $\{(x_i, y_i)\}_{i=1}^{N_t}$ is a sequence of coordinates previously touched by the user.
- Character Candidates $c_t = \{(c_{ij}, k_{ij})\}_{j=1}^{N_t}$ are the pairs of a character candidate $c_{ij}$ and its weights $k_{ij}$.
- Word Candidates $w_t = \{(w_{ij}, k_{ij})\}_{j=1}^{M_t}$ is the set of words $w_{ij}$ and their corresponding weights $k_{ij}$.
- Character Buffer and Word Buffer stores the previously generated character and word candidates.
Note here that this framework also works with English and other letter-based languages as well.

The Framework of Thai Input Methods

Here are the descriptions of the processes in the framework in the order that they are used.

**Character Candidate Generation** transforms the coordinate of a touch point into a character based on the keyboard layout. This can be easily done by checking the boundaries of buttons on the layout. Then, a character can be generated by comparing the touch point and the boundaries. A number of techniques have been proposed to generate a set of character candidates instead of one character candidate from a touch point. They improve the flexibility and accuracy of the input method.

**Word Candidate Generation** generates a list of words from the predefined word list and the sequence of character candidates. These word candidates will allow users to save a number of keystrokes. Instead of touching all the characters in the word, the users can input a number of characters and choose one word from the word candidate list. The technique is typically called “word completion”.

**Word Candidate Display** ranks the generated word candidates using a number of criteria, and displays them to the user. A user interface is needed to make it convenient for the user to select his/her preferred word. The criteria used in the candidate ranking typically combine the user’s behaviors and the word candidate’s features.

**Model Update** adjusts the weighting factors and parameters in the user model to reflect the characteristics and the behaviors of the users. This process improves the performance of the input method in terms of prediction accuracy since the model is geared to the particular user. This update can be done in all the processes in the input method. This process may include the “next word prediction” that generates a list of word candidates from the previously input words.

4. Character Candidate Generation Techniques

The character candidate generation is a process that transforms an \( (x, y) \) coordinate of the touch point on the keyboard layout to a character. It can be written in form of a function as:

\[
\text{CharCandidateGen} : \mathcal{P} \to \mathcal{C}
\]

where \( \mathcal{P} \) is a set of touch points, and \( \mathcal{C} \) is a set of characters.

However, the size of buttons typically prevents a user from effectively selecting the preferred character. It is comparatively small because of the screen size and the number of characters. To improve the performance, a number of techniques have been proposed to efficiently transform a given touch point into a character candidate. We will review them in this section.

**Button Size Adjustment** is the technique proposed in Keyboard ManMan [6]. This technique adjusts the size of buttons based on the statistics related to the characters. The buttons with a high chance to be the next characters are adjusted to be larger than the other keys. This makes it easier for a user to select those characters. The size of the buttons can be adjusted based on a statistical language model, such as the character \( n \)-gram model. However, this technique may cause a negative effect when the user intends to type a character different from the language model, for example, inputting a proper noun.

**Distance-based Candidate Generation** is the technique used in Janpinijrut et al. [7]. Instead of generating a character from a given touch point, this technique generates a set of character candidates for the given touch point. Thus, the above function for character candidate generation should be rewritten as:

\[
\text{CharCandidateGen} : \mathcal{P} \to 2^\mathcal{C}
\]

where \( \mathcal{C} \) is a set of character candidates, and \( 2^\mathcal{C} \) denotes the power set of \( \mathcal{C} \). A set of character candidates is based on the distance between the touch point and the buttons around the touch point. A button with the distance to the touch point less than a predefined radius is selected as a character candidate. Then, the generated set of character candidates can be used to construct a list of word candidates.

A number of experiments were conducted to get an appropriate distance value as well as the center of each button. The results showed that users tended to touch lower from the center of the buttons so that they can see the button labels before touching. Nevertheless, the technique models that the touch points are distributed as a circle of the predefined radius around the center of the button. This may not match the actual behaviors of users.

**Bivariate Normal Distribution-based Candidate Generation** is used in Phanchaipetch and Nattee [10]. The technique applies the bivariate normal distribution to model the \( (x, y) \) coordinates for each individual button. Thus, each button on the keyboard layout can be represented as five parameters of the bivariate distribution i.e. \( \mu_x, \mu_y, \sigma_x, \sigma_y, \) and \( \rho \). From a given touch coordinate, the probability that the coordinate belongs to a particular button can be calculated. A number of character candidates can, then, be generated by ranking the probability values:

\[
P(c | \mathbf{p}) = \frac{1}{\sqrt{2\pi\Sigma}} \exp\left( -\frac{1}{2} (\mathbf{p} - \mu)^T \Sigma^{-1} (\mathbf{p} - \mu) \right)
\]

where \( c \) is a character candidate, \( \mathbf{p} = [x \ y]^T \) is a touch point vector, \( \mu = [\mu_x \ \mu_y]^T \) is a mean vector of the coordinates on \( x \) and \( y \) axes, and \( \Sigma \) is covariance matrix of the coordinates i.e.:

\[
\Sigma = \begin{bmatrix}
\sigma_x^2 & \rho \sigma_x \sigma_y \\
\rho \sigma_x \sigma_y & \sigma_y^2
\end{bmatrix}
\]

Here, \( \rho \) represents a correlation between \( x \) and \( y \). We also have \( \sigma_x > 0 \) and \( \sigma_y > 0 \).

A number of experiments have been conducted based
on the usage data collected from real-world users. The results showed that the candidate generation based on bivariate normal distribution performed more accurately than the technique based on distance. This is because the bivariate normal distribution is more flexible and capable of capturing the behaviors of the users.

Candidate Generation using sliding input is a technique used in a number of Thai input methods including Swype [2] and TSwipe-Pro [3]. They allow a user to input Thai text by sliding a finger on the Ketmanee layout to form a word. Both methods are commercial and proprietary software. Thus, the techniques used by them are not disclosed. We can only observe their advantages and disadvantages by observing their functions and behaviors. From our observations, the sliding technique improves the text input speed since the user can move his/her finger through a sequence of intended characters. Then, a list of word candidates can be built from the sliding trail using the pre-collected word list.

Discussion

The Button Size Adjustment, used in Keyboard ManMan [6], technique provides a positive effect in terms of convenience when it predicts a high probability for the character the user intends to input. It may become a negative effect when the user wants to input an uncommon or unknown word. The presented Distance-based Candidate Generation and Bivariate Normal Distribution-based Candidate Generation, presented in [7] and [10], techniques generate multiple character candidates from a touch point. They increase the chance that the intended character is chosen as one of the candidates. However, too many candidates may confuse the user. The models for generating character candidates require more detailed study to provide a higher accuracy.

The candidate generation using sliding input, used in Swype [2] and TSwipe-Pro [3], allows faster word input since it does not require up and down movements. Moreover, one input method can be developed to support both sliding and touch inputs. This sliding input technique is a promising method for future development as suggested in [16].

5. Word Candidate Generation

Word candidate generation is a process that generates a list of word candidates from a number of character candidates. It can be defined in form of a function as:

\[ \text{WordCandidateGen} : S \rightarrow 2^W \]  

where \( S \) is a set of character sequences, and \( W \) is a set of known words.

A list of word candidates can be generated by matching the character sequence in the character buffer against words in the known word list, or applying a language model to construct a list of possible words. The retrieved words are ranked based on a ranking technique before showing to the user. Here are the techniques proposed for constructing a word list in Thai input methods.

Generating word candidates based on character n-gram model is a technique used in [10]. A word candidate is formed in this technique by connecting the character candidates in the Character Buffer. Figure 9 shows how the word candidates are constructed. Here, \( x_i, y_i \) denotes the \( i \)th touch coordinate, \( c_i \) is the \( i \)th character candidate from the \( i \)th coordinate. A word candidate is generated from selecting one character candidate from each touch coordinate in the buffer. These word candidates are evaluated and filtered by the character n-gram model. For example, a character trigram model can be used to evaluate a word candidate, i.e.:

\[ P(c_1 c_2 \ldots c_n) = \prod_{i=1}^{n} P(c_i|c_{i-2} c_{i-1}) \]  

where \( c_1 c_2 \ldots c_n \) represents a word candidate forming from a sequence of characters, \( c_i \) is the \( i \)th character in the word candidate, and \( P(c_i|c_{i-2} c_{i-1}) \) is the trigram probability of the \( i \)th character.

Generating word candidates based on known word list is an approach that generates a list of candidates from a sequence of character candidates and a list of known words. This approach is performed by applying an approximate string matching algorithm [17] to select a list of words similar to the input sequence of characters.

This approach has been applied in various Thai input methods since it helps reduce the number of keystrokes to input a word. Janpinijrut et al. [8] applied this approach with the vowel-separated keyboard layout by using a sequence of character candidates as a prefix to match the words in the known word list. The experimental results have suggested that the approach is able to reduce the number of keystrokes required to input a word.

Generating word candidates from only consonants is a technique proposed by Ballungpattama et al [9], [12] for a consonant-only Thai input method. This technique aims to generate a list of words from a sequence of Thai consonants without any vowels. It is done by applying the Boolean retrieval technique. The known word list is kept in the system using a pair of consonants as the index. Each pair is composed of a consonant from the first consonant in a word, and
a consonant in that word. Figure 10 shows an example of a small word list.

Given a sequence of consonants, a list of word candidates can be retrieved by (1) forming a set of indexes from the input sequence, (2) retrieving lists of words from the set of indexes, and (3) intersecting the retrieved lists.

The technique is able to generate a list of word candidates from a sequence of consonants. The input sequence needs to start with the first consonant in the intended word, and is followed by a number of consonants in the word in any order. The technique allows users to input words in the form of acronyms/abbreviations since the users may decide to omit some consonants. However, unknown words cannot be handled by the technique since the keyboard does not provide all the available Thai characters.

Discussion

Generating word candidates using a word list, used in [8], [9], is able to construct a list of meaningful words for users. It reduces the number of touches required to complete a word. However, this technique cannot deal well with unknown words such as proper names. Generating word candidates from consonant only [9], [12] reduces the number of buttons on the layout. The technique has a severe disadvantage that the user cannot input any unknown word. Generating word candidates using a statistical language model (e.g. n-gram model) [10] provides an advantage on generating unknown word candidates. Nevertheless, the candidates may not be meaningful words. Combining both techniques will utilize the advantages from both of them. Moreover, a language model of meaningful unknown words should be studied in the future in order to generate more appropriate word candidates. This can be done by analyzing the structure of Thai proper nouns.

6. Word Candidate Ranking

After the word candidate generation process, a list of word candidates is constructed. This list will then be shown to a user to select the preferred word. However, due to the number of candidates and the limited space on the touch screen, the candidates must be ranked so that the most probable candidates are displayed to the user. Here are the criteria used to rank the word candidates:

<table>
<thead>
<tr>
<th>Index</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>bh</td>
<td>birth</td>
</tr>
<tr>
<td>br</td>
<td>birth</td>
</tr>
<tr>
<td>bt</td>
<td>bat, birth, but</td>
</tr>
<tr>
<td>ct</td>
<td>cat</td>
</tr>
<tr>
<td>nd</td>
<td>and</td>
</tr>
<tr>
<td>nt</td>
<td>ant</td>
</tr>
</tbody>
</table>

Table 1 Example of the data structure used in consonant-only keyboard.

Fig. 10 Example of the data structure used in consonant-only keyboard.

Word frequency from standard corpus is a basic feature of word candidates that can be obtained from a corpus. A word that is frequently used in the corpus is assumed to be used frequently in the smartphones. Thus, the word candidates with high frequency are highly ranked. This criteria is used in various input methods [8]–[10]. However, the criterion has a disadvantage on dealing with unknown words since they cannot be found in the standard corpus.

Weighted word frequency from user word list [12] improves the disadvantage of the word frequency from the corpus by collecting all the words inputted by the user. Then, the word frequency from the standard corpus and the frequency from the user word list are combined and used as a ranking criterion. Thus, the weighted word frequency $f_w$ can be computed as:

$$f_w(w) = \alpha f_u(w) + (1 - \alpha) f_c(w)$$

where $w$ represents a word candidate, $\alpha$ is a weighting factor with $0.0 \leq \alpha \leq 1.0$, $f_c$ and $f_u$ are the word frequency in the standard corpus and the user-defined word list, respectively.

Edit distance represents the difference between the input string and a word candidate. By using this criterion, the word candidate that is more similar to the input string is highly ranked. This criterion is helpful for input methods that do not require the user to input the exact string e.g. [12].

Discussion

Ranking word candidates using word frequency from standard corpus, used in [8]–[10], has a disadvantage that it is unable to deal with unknown words. Utilizing weighted word frequency from user word list, used in [12], provides an advantage on dealing with both known and unknown words since the user is allowed to define his/her own words. Edit distance [12] can also be applied with the other two techniques in order to emphasize inputting the exact string.

To improve the performance of the candidate ranking, the statistical data should reflect the actual word usage on smartphones. However, all the Thai standard corpora are now collected from written documents. The statistics obtained those corpora may not be a good representation of the conversational language typically used in smartphones. A corpus from informal language usage is necessary to improve the candidate ranking.

7. Model Update and Personalization

To improve the performance of the input methods, a number of input methods use feedback from the user to update the candidate generation model, as well as to shape the parameters in the model to fit the characteristics and behaviors of the user. In this section, we will review a technique proposed to provide the personalization.

Phanchaipech and Nattee [11] have proposed the adaptive bivariate normal distribution technique to personalize the character candidate generation. In this technique, the character candidates are initially generated from the bivariate normal distributions using the pre-collected data as shown in Eq. 3. Each distribution can be represented by a
mean vector $\mu$, and a covariance matrix $\Sigma$. A word candidate is generated from matching the combination of character candidates with the prefix of words in the word list. When the user inputs a number of characters and selects a word candidate, the prefix of the selected word is considered the sequence of characters the user intended to input. We can then match each character in the prefix with a touch coordinate. This feedback can be used to recalculate the parameters of the bivariate normal distribution. This results in parameters that more fit the behavior of the user. Thus, a more accurate set of character candidates can be generated.

Discussion

The adaptive bivariate normal distribution technique [11] adjusts the bivariate parameters when the user selects one of the candidates. Utilizing feedback from the user will allow an input method to adjust the models to suit the user’s characteristics. However, the process needs to be done without disturbing the usability of the input method. Moreover, it should be able to efficiently handle errors that may cause from the user, for example, updating the models when the user chose an unintended word from the candidates. This is still a challenge in the user interface design.

8. Conclusion

We have reviewed a number of Thai input methods and virtual keyboards and located them in the framework for a Thai Input Method on Smartphones. However, the number of techniques proposed is still low, there is more room for improvement for researchers and developers. We hope that our work will serve as a tool for future development of Thai input methods and virtual keyboards.

Acknowledgements

This research has been supported by Thammasat University Research Fund. The author would also like to thank Dr. Nirattaya Khamsemanan for her support and many helpful suggestions. Lastly the author would like to thank the reviewers for all the useful suggestions.

References

http://www.google.com/mobile/android/

Cholwich Nattee  
graduated BEng in Computer Engineering from Chulalongkorn University, Thailand. He received both MEng and DEng in Computer Science from Tokyo Institute of Technology, Japan. Currently, he is an Assistant Professor at Sirindhorn International Institute of Technology (SIIT), Thammasat University.