Support Vector Machine (SVM) Based Classifier For Khmer Printed Character-set Recognition

Pongsametrey Sok* and Nguonly Taing†
Royal University of Phnom Penh, Cambodia
pongsametrey.mite248@rupp.edu.kh*, taing.nguonly@rupp.edu.kh†

***Abstract*— this paper describes on the use of Support Vector Machine (SVM) based classification method on Khmer Printed Character-set Recognition (PCR) in bitmap document. Khmer language has been identified as one of the most complex language with the total of 74 alphabets and the wording compound can has up to 5 vertical levels. This paper proposes one new method, SVM for Khmer character classification system by using 3 different SVM kernels (Gaussian, Polynomial and Linear Kernel) on data training and recognition to find out the best kernel for Khmer language. The method allows us to use small training dataset by training different pieces of character training instead of training big amount of clusters. The classification uses binary data of 0 as white space and 1 as black pixel area of the character; each training piece of character has been stretched into a matrix of the binary data in all kinds of image size. Feature extraction is extracted from the matrix to use in SVM classification. After recognition, there are some rules to combine each cluster or character by using character levels or common mistake correction. The experiment of about pure 750 Khmer words or around 3000 characters show that SVM method with Gaussian Kernel produces a good result with better performance among all kernels. The system uses one font "Khmer OS Content" of the training data with font size 32pt to recognize 3 different font sizes. The accuracy of 28pt font size is 98.17%, 32pt is 98.62% and 36pt is 98.54% respectively.**

Keywords: SVM, Khmer OCR, Optical Character Recognition, Khmer Unicode

I. INTRODUCTION

Khmer language is one of the most complex written language which combines of current using 33 consonants, 23 dependent vowels, 12 independent vowels and other signs [1]. The written system is left-to-right with vowel signs that can be placed after, before, above or below the consonant they follow [2]. That is why the combined of Khmer characters is identified up to 5 vertical levels which are two levels as superscript, one main character body and two levels as subscript [3].

Optical Character Recognition (OCR) is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text [4]. Printed character recognition (PCR) is a part of Optical Character Recognition (OCR) which is about the image of handwritten, typewritten or printed text.

OCR System has been developed maturely for different languages such as English, Japanese or Chinese etc. but Khmer OCR is still infancy because there are still few researches available, the difficulty of the language itself and also because some researches attempt to solve basic problem of Khmer OCR only [5].

A. Previous Researches

As shown in "Table 1", until today, there is some previous Khmer OCR researches are being published as reports, papers or thesis. There are some methods already uses in segmentation or classification part as well.

TABLE I
PREVIOUS KHMER OCR RESEARCHES

Author	Feature / Classifier / Pre-Processing / Template	Test / Accuracy
C.Chey et al [8]	Lagendre / Euclidean / None / 1 font	10 characters: បពជកភណាឃសវទ Accuracy: 92%
C. Chey et al [9]	Temporal domain plus Wavelet descriptor / Euclidean / Binarization and Skeletonization / 20 fonts	Font size 22, 18, 12 Accuracy: 92.85% for 22pt, 91.66% for 18pt and 89.27% for 12pt
Ing Leng Ieng [6]	Discrete Cosine Transform / Hidden Markov Model / Assume skew-free and noise-free/ Limon S1 with 22pt size	10 pages from newspaper Accuracy: 92.48%
Ing Leng Ieng [7]**Error! Reference source not found.**	Discrete Cosine Transform / Hidden Markov Model / Noise removal (salt-pepper noise) / Limon R1 with 22pt size	10 pages from newspaper Accuracy: 98.88%
Kruy Vanna [5]	Scale Invariant Fourier Transform (SIFT) / Euclidean (SIFT Score) / None / Khmer OS System	1104 words from newspaper Accuracy: 77%
	SIFT and Fourier Descriptor / Euclidean (Fuse Distance) / None / 5 fonts: Khmer OS System, Khmer OS Bokor, Khmer OS Freehand, Khmer OS Muol and Khmer OS MuolPali	1104 words from newspaper Accuracy: 97.4%
	SIFT, Fourier Descriptor, Component Hole and Component Location / Euclidean (Fuse Distance) / Noise and Particle Removal / 14 fonts	1104 word from newspaper. Font size 22. Accuracy: 97% average
Iech Setha and Taing Nguonly [3]	Edge Detection, Component Location and Connected Component / Template Matching / None / Khmer OS Content with 36pt size	Khmer OS Content with 36pt size Accuracy: 98%

B. Support Vector Machine

Support Vector Machine (SVM) introduced by Vladimir Vapnik in 1963. It is belong to the broad family of kernel-based learning algorithms and can be used for regression as well as classification algorithms. SVM is similar to logistic regression in the sense that it tries to separate data between groups. SVM is considered to be the best stock classifier [10].

At first, SVM comes with Linear SVM by considering when two groups of data are be able to separate into two parts by a straight line, means the data is linearly separable as shown in "Fig. 1".

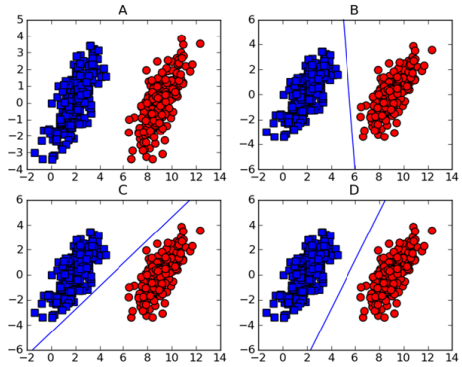
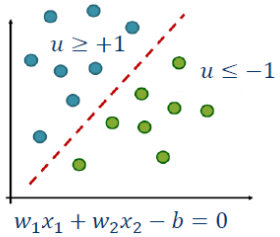


Fig. 1 Linearly separable data is shown in frame A. Frame B, C, and D show possible valid lines separating the two classes of data.

Basically, the separable case defined as in (1) where input space $x \in X$, output space $y_i \in \{-1, +1\}$ in the case of classification, w is the weight containing the weight coefficients for a specific hyperplane and b is the bias term or threshold.

$$u = \vec{w}.\vec{x} - b = 0 \quad (1)$$



Support Vectors are the data points that lie closest to the decision surface on the margin. The Maximize Margin forms as in "Fig. 2" and can be expressed by (2).

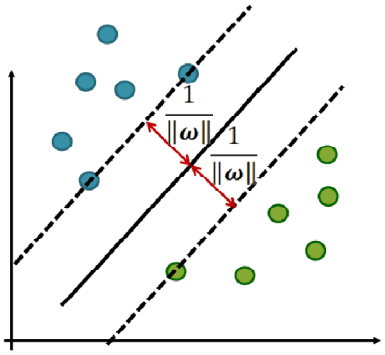


Fig 2. Maximize Margin

Equation (2), x_i is the ith training example, and y_i is the correct output of SVM for the ith training example. Value of y_i is $+1$ for positive and -1 for negative example.

$$\min_{w,b} \frac{1}{2} \|\vec{w}\|^2 \; subject\ to \; y_i(\vec{w}\vec{x}_i - b) \geq 1, \forall i \quad (2)$$

Support vector machines are a type of classifier. They're called machines because they generate a binary decision; they're decision machines. Support vector machines try to maximize margin by solving a quadratic optimization problem [10]. Lagrange Multiplier is used in the optimization problem with following Quadratic Programming (3, 4, 5).

$$\min_{\propto} \Psi(\vec{a}) = \min_{\propto} \frac{1}{2} \sum_{i=0}^{N} \sum_{j=0}^{N} y_i y_j (\vec{x}_i.\vec{y}_i)\alpha_i\alpha_i - \sum_{i=0}^{n} \alpha_i \quad (3)$$

Where N is the number of training examples
subject to $\alpha_i \geq 1, \forall i, \sum_{i=0}^{N} y_i \alpha_i = 0$

Once Lagrange multiplier α is derived weight $\vec{w}$ and threshold b can be calculated by (4):

$$\vec{w} = \sum_{i=0}^{n} y_i \alpha_i \vec{x}_i \quad (4)$$
$$b = \vec{w}.\vec{x}_k - y_k \text{ for some } a_k > 0$$

Once weight $\vec{w}$ and threshold b is defined, classification of easy by detecting only the sign of linear equation (5):

$$u = sign(\vec{w}.\vec{x} - b) \quad (5)$$

where $\vec{x}$is the input vector to classify

However, in real world the data are not always linear "Fig. 3" and it is not possible to classify by linear classifier, in mean time, the Non-linear SVM classifier is proposed [10].

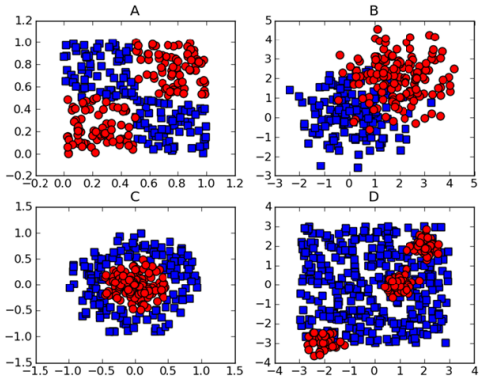


Fig. 3 Examples of datasets those aren't linearly separable

The non-linear SVM comes with Kernel trick. The Kernel trick is a very interesting and powerful tool, to bridges from linearity to non-linearity to any algorithm [10] and this can be represented by (6).

$$u = \sum_{j=1}^{N} y_j \alpha_j K(\overrightarrow{x_j}, \vec{x}) - b \quad (6)$$

Where:

- K is a kernel function
- $\vec{x}$ is the input vector (training data)
- $\overrightarrow{x_j}$ is the training vector at j

In Kernel trick, there are 3 standard kernels for SVM. Linear Kernel (7) is a simplest kernel function. Polynomial Kernel (8) is a well suited for problems where all data is normalized. Gaussian Kernel (9) is one of the most versatile kernels. It is a radial basis function kernel, and is the preferred kernel when we don't know much about the data we are trying to model.

$$k\,(x,y) = x^T y + c \tag{7}$$

$$k\,(x,y) = (ax^T y + c)^d \tag{8}$$

$$k\,(x,y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2}) \tag{9}$$

Originally, SVM is solved by dynamic programming which is time consuming and CPU intensive until John Platt introduces [10] a new way to solve SVM with minimal time spent. It is called Sequential Minimal Optimization (SMO), the optimization algorithm to solve SVM classifier. Platt's algorithm is composed of 3 main procedures:

- run: which iterates over all points until convergence to a tolerance threshold
- examineExample: which finds two points to jointly optimize
- takeStep: which solves the 2-dimensional optimization problem analytically

The full pseudo code represents as in "Fig. 4".

```
procedure run
    numChanged = 0;
    examineAll = 1;
    while (numChanged > 0 | examineAll)
    {
      numChanged = 0;
      if (examineAll)
        loop I over all training examples
          numChanged += examineExample(I)
      else
        loop I over examples where alpha is not 0 & not C
          numChanged += examineExample(I)
      if (examineAll == 1)
        examineAll = 0
      else if (numChanged == 0)
        examineAll = 1
    }
endprocedure
```

Fig. 4 Research and Implementation Modules

C. *Proposed Method*

SVM based method classification is a new method for Khmer OCR system. The focus of is research is to experiment the SVM in different kernels to find out which one is a good choice for Khmer language and also produce a better performance and accuracy. Meanwhile, SVM method is not applied in segmentation part, so that edge detection presented in the research as in [3] has been used. And there are two parts of the implementations; they are about data preparation (Raw Data Training) and an OCR system, "Fig. 5".

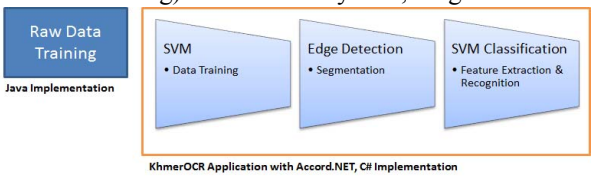


Fig. 5 Research and Implementation Modules

II. Data Preparation and Training

Preparation the raw data is the main part of the system.

D. *Define Level of Khmer Characters*

There are three kinds of Khmer character levels in one cluster but among them, there are each two levels for the superscript and the subscript level [3]. Here are some examples:

- Middle level characters

  ក ខ គ ឃ ង ច ឆ ជ ឈ ញ ! ឿ ()

- Subscript level characters (another level is lower)

  ◌្ក ◌្ខ ◌្គ ◌្ឃ ◌្ង ◌្ច ◌្ឆ ◌្ជ ◌្ឈ ◌្ញ

- Superscript level characters (another level is higher)

  ◌ិ ◌ី ◌ឹ ◌ឺ ◌ុ ◌ូ ◌ួ ◌ើ ◌ឿ ◌ៀ

E. *Classified Types of Khmer Characters*

The appearance of the character could be different by font face (the design). As the research uses only "Khmer OS Content" font, there is a process to cut each piece of each character by detecting the black and white pixels. There are the four types of characters [3]. "Fig. 6" and "Fig. 7" are about the single consonant that has cut in pieces if white spaces found.

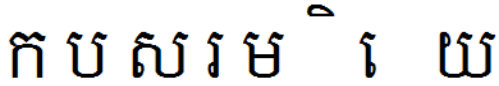


Fig. 6 Connected-point Single Character

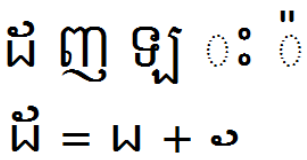


Fig. 7 Disconnected-point Single Character

"Fig. 8" and "Fig. 9" is the combined of consonant and vowel called connected-component. Khmer writing style does not have visible space to separate word. That is why we detect different appearance when they stay close to each other.
Some methods such as template matching as shown in [3] has defined more types on group of vowels but with SVM, we do not need to do so, we use correction rules instead.

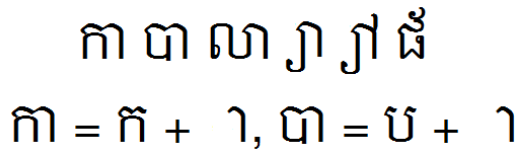


Fig. 8 Connected-point Combined Character

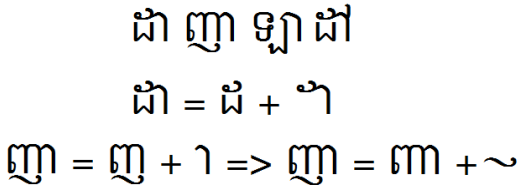


Fig. 9 Disconnected-point Combined Character

F. Raw Data and SVM Data Training

The preparation data have produced the bitmap images for each piece of character and transform it into binary of 0 for white pixel and 1 for black pixel. There is an image processing to identify black or gray pixel. “Fig. 10” is the matrix of 32x32 dimensions of a character.

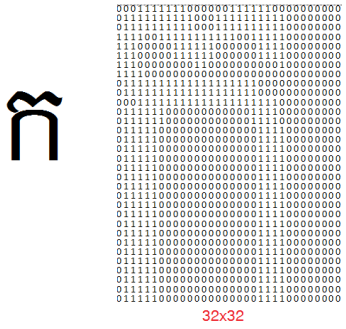


Fig. 10 Sample binary data to represent a character for computing

SVM classifier is a binary classification and kernel-based learning algorithms. However, finding the optimal solution is quite challenging until [10] introduced SMO to simplify the optimal solution. As SVM is two classes classifier, there is a trick to enhance it for working with multi-classes. Accord.NET is a software framework that has implemented for SVM [12]. It uses feature extraction of 1024 blocks array which is the extraction from the matrix of 32x32 dimensions.

We implement the training module with character-based database by separated them into different vertical level. “Fig. 11” is the interface of the data training module. We prepare 4 datasets which are “All Characters Set” which has 250 support vectors, “Main Level Characters Set” has 177 support vectors, “Top Level Characters Set” has 21 and “Bottom Level Characters Set” has 52.

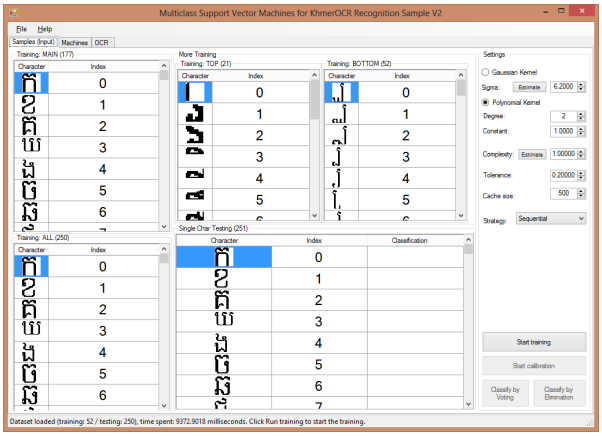


Fig. 11 SVM Training Module

III. OCR SYSTEM

The system is constructed only from segmentation, feature extraction, classification and character assembling as shown in “Fig. 12”. There is no preprocessing part since we are using printed characters on bitmap; there is no noise removal or line stretch as normal in OCR system.

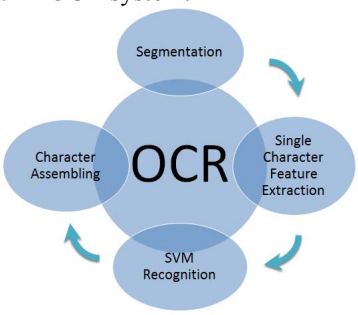


Fig. 12 OCR Features

A. Segmentation

The segmentation module uses edge detection by reading the image document pixel by pixel within a line separation under the main body block from top-to-bottom and from left-to-right to extract black block. When the system finds the black pixel, it starts to detect surrounding connected black blocks until no more black pixel founds [3, 13]. As in “Fig. 13”, the segment starts first detect line and then by each line it segments each character into different pieces.

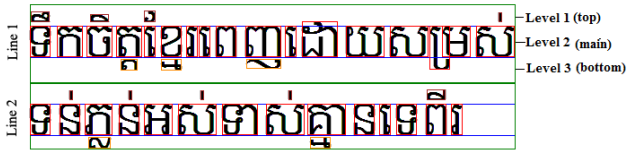


Fig. 13 Line & Character Segmentation

B. Feature Extraction

During the character segmentation, each character has been extracted and the system calculates the object feature. The character object is transformed into binary data and store as

matrix of 32x32 dimensions and then it extracts 1024 ($32 * 32$) features from the single object and stores each binary character of 0 or 1 into each block of the vector array.

C. *Classification and Recognition*

There are many methods for designing pattern classifier such as Naïve Bayes classifier, neural network, etc. [11]. Here we use multi-class SVM with One-Again-One (OAO) model with kernels of non-linear SVM.

The OAO is used to discard the redundant option during classify by (1), n classes can be decomposed in a small subset.

$$k = \frac{n(n-1)}{2} \quad (10)$$

In the system, the classification is separated into 4 subsets of SVM kernel objects. We classify "All Characters Set" of 250 classes support vectors (n = 250), with OAO model we could find the number of decision pairs (machines) of any support vector as in (12).

$$k = \frac{250(250-1)}{2} = 31125\ machines \quad (12)$$

The same way applies for other 3 subsets on "Main Level Characters Set" as in (13), "Top Level Characters Set" as in (14) and "Bottom Level Characters Set" as in (15).

$$k = \frac{177(177-1)}{2} = 15576\ machines \quad (13)$$

$$k = \frac{21(21-1)}{2} = 210\ machines \quad (14)$$

$$k = \frac{52(52-1)}{2} = 1300\ machines \quad (15)$$

There are two computing schemes for OAO. Voting scheme is the max-voting method which is also known as 1vs1 decision and elimination scheme is also known as Directed Acyclic Graph (DAG) decision. We use "Voting Scheme" for all the experiments.

After recognition, the system produces output in ASCII format of font "Limon S1". Since this is a legacy font, it allows the character to be written as it is appeared, not as spelling in Unicode format "Fig. 14". And then the system uses Legacy-to-Unicode converter tool to produce the output in Khmer Unicode.

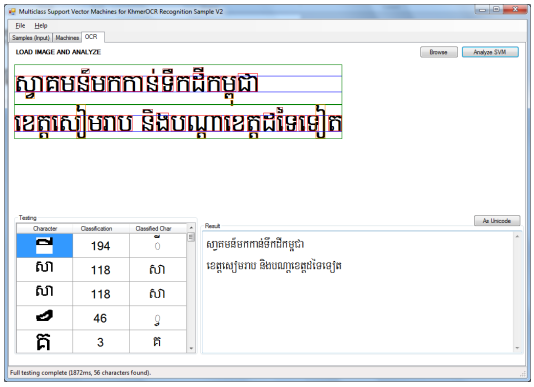


Fig. 14 Sample Testing with Gaussian Kernel
(Welcome to Cambodia, Siem Reap and other provinces)

D. *Character Assembly and Corrections*

Character Assembling is very important after character is recognized from the classifier. We combine the result of each piece of character to produce a final result of each character and then put it into the line as in order.

The character assembling is to construct the following type of character classifier as shown in section II.B. In "Fig. 15", the last process of the system, the output of the SVM classifier is in ASCII code but the disconnected-point character as example is not an actual ASCII code, it uses the notation then recombine.

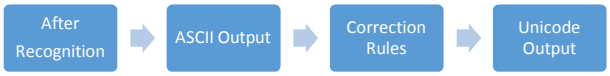


Fig. 15 The process at Character Assembly

Once each code is obtained from the recognition process, it is passed to check some mistakes or errors by using some rule conditions of Khmer spelling of each word as in "Table 2". Some cases, there is an issue linking to segmentation that could not get the right character bitmap and its pixels.

TABLE 2: SOME IDENTIFIED ISSUE AND ITS SOLUTION

Disconnected-point Single/Combined-character	
ម + ◌ = ដ១ដ២ = ដ ញ + ◌ = ញ១ញ២ = ញ ទ + ◌ = ទឡ២ = ឡ	ព + ◌ = ល្1ល្2 = ល្ ប + ◌ = ប1ប្ = ប្ ប + ◌ = ប1ប្ = ប្
SVM wrongly classification but can solve by character spelling condition	
ែ + ◌ (ដ) = ដែ២ = ែ រ + ែ = រែ១ = ែ រ + ◌ (ដ) = រដែ២ = ែ	◌ (ដ១) + ◌ = ដ១ = ដ ◌ (ដ១) + (ម "missed")= ដ១ = ដ ព + ◌ + ◌ = ពញ២ = ញ ញ(ញ១) = ព + ◌ = ព

IV. EXPERIMENT

An experiment is carried out to verify the propose method. An application is built with C# Application on Visual Studio 2012 by using Accord.NET framework 2.9 [12], "Table 3" is the detail environment used in the experiment.

This application demonstrates how Khmer OCR with SVM works with printed character images. This section also demonstrates on different parameters of different SVM Kernels (Gaussian, Linear and Polynomial).

TABLE 3: EXPERIENCE ENVIRONMENT

No.	Item	Description
1	Operation System	**PC1**: Windows 8-32bits, Intel Pentium ® DualCore 2.70 GHz, RAM 2GB **PC2**: Windows 8-32bit, Intel Core2Duo 2.53GHz, RAM 3GB **PC3**: Windows 8-64bits, Intel Core i3 1.80GHz, RAM 4GB
2	Dev. tool	Microsoft Visual Studio 2012
3	Programming language	C#
4	Runtime Environment	.Net Framework 4.0
5	Framework	Accord.NET 2.9 (latest version 2.11)
6	Algorithms	• Edge Detection (Segmentation) • SVM (Multi-class, Non-Linear, SMO)
7	Functions	Raw & SVM Training
8	Experiment Input	Bitmap files (*.bmp)
9	Experiment Output	• Text files (training set, log) • Machine-code character • Machine-code text document
10	Training files	• khmer_all.txt • khmer_main.txt • khmer_top.txt • khmer_bottom.txt
11	Participation	• Experiment involvement: 4 people

A. *Experiment Condition*

"Table 4" shows the experiment data uses "Khmer OS Content" font with different font sizes.

TABLE 4: SAMPLE DATA

No.	Description	Num. of words	Num. of Characters
1	Font size: 36pt Resolution 1186x738	81	385
2	Font size: 36pt Resolution 1806x630	85	526
3	Font size: 36pt Resolution 1760x798	119	616
4	Font size: 28pt Resolution 1030x652	79	427
5	Font size: 32pt Resolution 1030x917	79	427
6	Font size: 36pt Resolution 1257x917	79	427
7	Font size: 36pt Resolution 1672x1552	209	993
8	Others: font size: 24pt, 32pt, 36pt, 48pt, 60pt		

The focus font sizes are 28pt, 32pt and 36pt. And there is also some tests on other font sizes range from 24pt to 60pt as well to learn about the flexibility of the method.

We use the data as shown in "Table 4" and with different SVM Kernel's parameter; we need to do data training by each kernel since SVM store the calculation by its kernel. And then we can analyze the input images dependently.

B. *Parameters Setting*

"Fig. 16" on the top right shows the setting of some parameters of each kernel and for adapting the calculation algorithms. To switch each kernel, two parameters are important, Sigma (σ) is for Gaussian Kernel and Degree is for Linear Kernel when it is equal to 1 and for Polynomial Kernel when it is greater than 1. The possible setting provides by the Framework Accord.NET via class MulticlassSupportVectorLearning with a delegate function, denoted Configure, which can be used to select and configure any learning algorithm [12].

Adaptation parameters in this training module are:

- Kernel uses:
  - Sigma (σ)
  - Degree
  - Constant
- SMO algorithms parameters:
  - Complexity
  - Tolerance
  - Cache Size
  - Strategy (Sequential, Worst Pair)

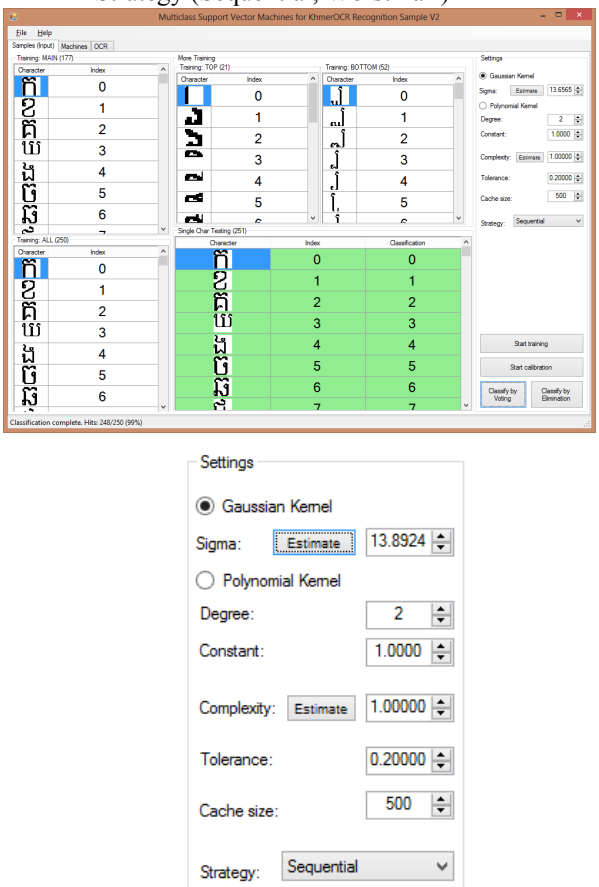


Fig. 16 Setting Training Interface, Gaussian Kernel

In the experiment, most of the time, we use default value by using the estimate button, especially to provide the value of Sigma for Gaussian Kernel which always depends on number of training data and the complexity value for SMO algorithms. The calculation of the estimation value is provided by Accord.NET.

The application demonstration focus on 3 kinds of results: Time consuming during classification, CPU usage by manually tracking as average (AVG) between the recognition and the accuracy which is calculated using the output characters, words and counting errors as in (16).

$$Accuracy(\%) = \frac{(total\ char - error\ char)}{total\ char} * 100 \qquad (16)$$

"Table 5" is the detail of parameters setting, training and test result of font size 36pt that uses in the experiment.

TABLE 5: PARAMETERS SETTING, DATA TRAINING AND TEST RESULT OF FONT SIZE 36PT

Parameters	Gaussian	Linear	Polynomial
Kernel Setting			
Sigma (σ)	13.6565		
Degree		1	2
Constant		1	1
Sequential Minimal Optimization (SMO)			
Complexity	1	0.00351	0.00001
Tolerance	0.2	0.2	0.2
Cache Size	500	500	500
Strategy	Sequential	Sequential	Sequential
Training Result Info			
Time consuming (ms)	3690.4678	1837	1030
Classified and Recognition Result (font size: 36pt)			
Avg. CPU Usage	43%	62%	53%
Avg. Time cons. (ms)	3285.857	30314.42857	3102.286
Accuracy	98.54%	98.44%	96.78%

The input data is the characters printed on image document without any noise since the system does not include the preprocessing part. The data will be fully loaded into the system and the analyze starts from segmentation until the output. "Fig. 17" shows the sample data tested of 3 lines and uses Gaussian Kernel.

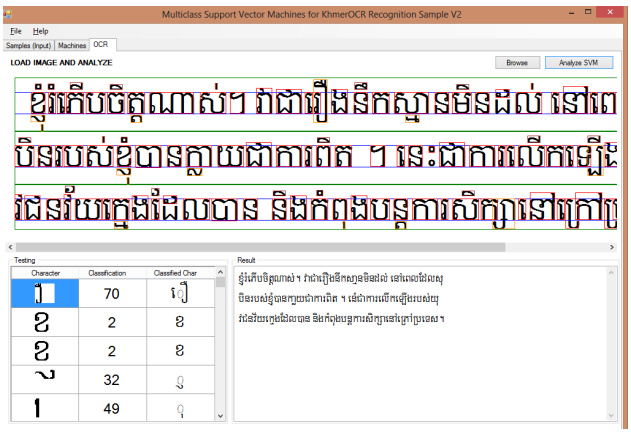


Fig. 17 Testing interface, 3 lines testing

V. RESULT AND DISCUSSION

A. *Time Consuming, Resource and Accuracy*

The result differs depending on font sizes. The following result uses data with font size 36pt. In the experiment with the application, we track per each kernel, time consuming, resource usage and accuracy of the output.

We track resource consuming by using Windows task manager and only the OCR system is running, manually record CPU graph before and after classification as in "Fig. 18".

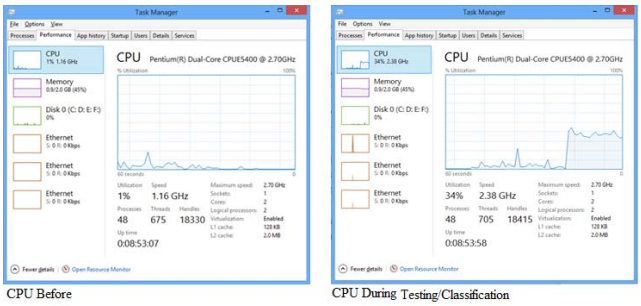


Fig. 18 SVM Kernels Resource Usage Comparison

"Fig. 19" shows average percentage of CPU usage comparison of each kernel.

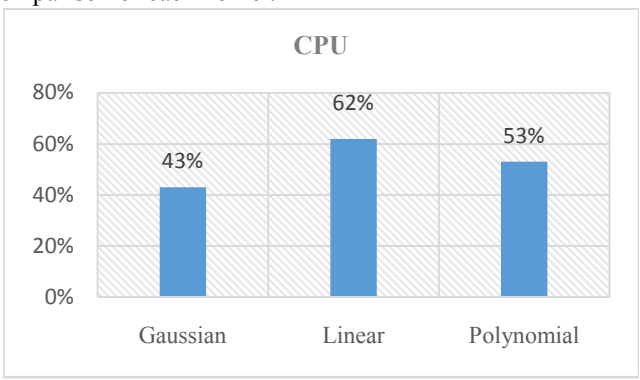


Fig. 19 SVM Kernels Resource Usage Comparison

The CPU usage shows that Gaussian kernel has better result with 43% comparing to Linear kernel has 62% and

Polynomial kernel has 53% as average. Time consuming comparison shows in "Fig. 20" both training time and recognition time. Gaussian Kernel uses similar time as Polynomial Kernel but Linear Kernel uses too much time in recognition.

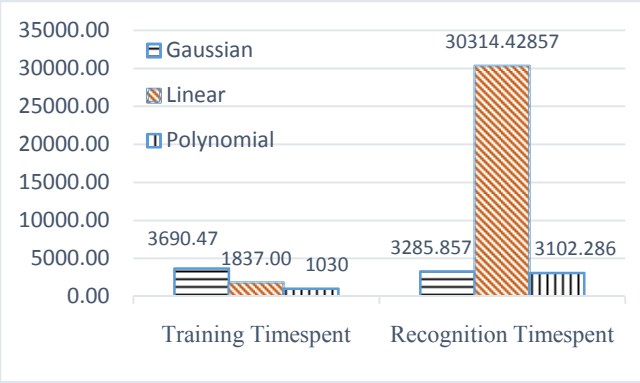


Fig. 20 SVM Kernels Training and Recognition Time Consuming Comparison

The accuracy, as in "Fig. 21", Gaussian Kernel has 98.54% of accuracy, Linear Kernel has 98.44% and Polynomial has 96.78%.

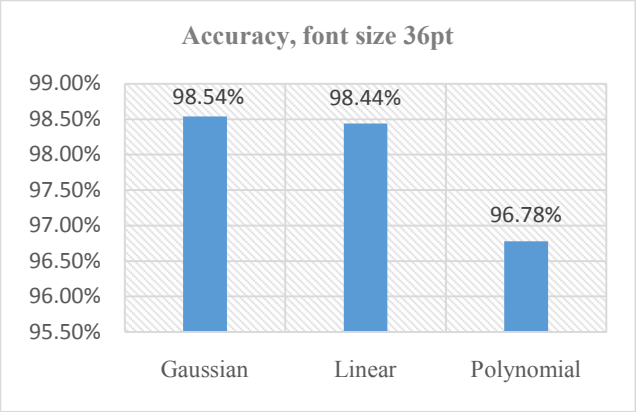


Fig. 21 SVM Kernels Accuracy Comparison

As result, receives from the test of files with font size 36pt shows that Gaussian Kernel is better than other kernels among CPU usage, 43% and accuracy, 98.54%. If we choose time consuming as the main point, Polynomial plays a bit better but its accuracy is too low.

B. Accuracy of Different Font Sizes

Most of previous researches as in section "I.A", the training dataset depends on font size and to recognize the same size. This research, we want to find output the result in different font size while we are using only one font size in our dataset which is size 32pt. We first target to experiment three sizes: 28pt, 32pt and 36pt. And then we continue on more other sizes such as: 24pt, 48pt and 60pt and as result as in "Fig. 22", by using Gaussian Kernel and could see that the bigger size and clear, the method received more better accuracy but the lower pixels such as 28pt and 24pt receives not a good result.

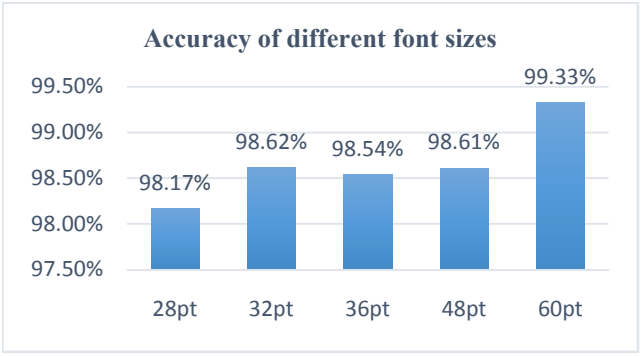


Fig. 22 Gaussian Kernels on Different Font Size

"Fig. 22" does not show the result of font size 24pt since the result is very low, the issue here because the pixels receive from the segment and stretch into the matrix 32x32 dataset could produce wrong pixel replacement that could make the classification at the recognition output the wrong result. "Table 6" shows some errors from the recognition of font size 24pt and 48pt.

TABLE 6: SOME CHARACTER ERRORS OF DIFFERENT FONT SIZES EXAMPLE

24pt, Pixel Image / ID / Recognized			48pt, Pixel Image / ID / Recognized		
ច	7	ឆ	ទ្ទាំ	122	ខ្ចៅ
ក	7	ឆ	ជ	8	ជ
ទ្ទាំ	122	ខ្ចៅ			
ញ	97	ឆា			

C. Discussion

This paper illustrates the new method to classify Khmer printed characters with SVM via 3 standard kernel tricks. Gaussian Kernel SVM works just fine for Khmer character-set recognition.

The Gaussian Kernel is the best SVM Kernel for classifying the Khmer characters and recognition on top of accuracy, resource usage and reasonable time consuming. According to the section "I.B" explains that Gaussian kernel is a preferred kernel when we don't know much about the data we are trying to model. It could be the answer since from this proposed method, we use character-based dataset for classification, and some of the characters have been cut into piece and combined the result as described in section "II.E, III.D".

Anyway, there are some characters that SVM could not classify well and also because of application itself. "Table 7" lists characters that are a most impacted on result. The issue would also because of only one font size choose for training, 32pt that would not make SVM well understood each character's feature in one font and size. The result would be much better if we could train more font sizes of each dataset and find more character condition to produce more samples for the experiment as well.

TABLE 7: UN-RECOGNIZED CHARACTERS

Found but wrong	Character Expected	Issued on Tested Font Size	Explain
◌ុ	◌ះ	28pt, 36pt	Error because of the segmentation
ជ	ជ	32pt, 36pt	SVM classification error
ខ	ទ	28pt, 32pt	SVM classification error
ណ	ញ	32pt	SVM classification error
ឈ	ញ	28pt	SVM classification error
◌ី or ◌ិ	◌ៈ	28pt, 32pt	SVM classification error & Segmentation issue

VI. CONCLUSIONS

This paper has presented the method of SVM based classifier on Khmer printed character-set which has illustrated different standard kernels. The method takes printed Khmer character as input image and converts it into machine-codes document format. The experiment shows that Gaussian Kernel is the best kernel for Khmer language.

The study uses kernel tricks of multi-class SVM, so it is very important to choose training dataset and SVM kernel as it is a key for applying SVM classification for better result. Each character need to place on a right condition for classifying as well as assembling; as example, defined its level and well extract its pixels during segmentation.

The computed result shows the accuracy of 98.54% font size of 36pt, 98.62% for font size 32pt and 98.17% for font size of 28pt by using Gaussian SVM Kernel which works better than Linear and Polynomial kernel for Khmer character classification.

Anyway, there are still some challenges to do with different font size, SVM classifier, segmentation and post processing. Segmentation is one of the important parts to help cluster clear character pixel for feature extraction. In this research, segmentation produces some error clusters that lead wrong classifier. At the recognition process as briefly shown in "Table 7", we need to solve different issues for different font size in order to adapt the SVM kernel to work for most of font sizes and font faces of Khmer language in the future. Post processing is the process after receiving the result such character assembly, there are many other Khmer spelling rule to apply in this correction part as well.

REFERENCES

[1] Khmer alphabet. Online, Sept. 2014. Available http://en.wikipedia.org/wiki/Khmer_(script)

[2] Khmer Language. Online, Sept. 2014. Available: http://en.wikipedia.org/wiki/Khmer_language.

[3] Iech Setha and Taing Nguonly, A Combined Edge Detection and Template Matching for Khmer Printed Character-set Recognition. The paper presented at the First Asian Conference on Information Systems: Siem Reap, Cambodia, December 2012.

[4] Optical Character Recognition. Online, Sept. 2014. Available: http://en.wikipedia.org/wiki/Optical_character_recognition

[5] Kruy Vanna, Khmer Optical Character Recognition. Master Thesis, Kameyama aboratory, Graduate School of Global Information and Telecommunication Studies, Wesda University, Tokyo, Japan, January 2011.

[6] Ing Leng Ieng, Khmer Segmentation Study Report. PAN Localization Project, September 2008, unpublished.

[7] Ing Leng Ieng. Khmer OCR for Limon R1 Size 22 Report. Pan Localization Project, July 2009, unpublished.

[8] Chey Chanoeurn, Kosin Chamnongthai and Pinit Kumhom, Khmer Printed Characters Recognition Using Lagendre Moment Descriptor. Department Electronic and Telecommunication, King Mongkut's University of Technology Thonburi St. Pracha Utid, Bangmut, Bangkok, 10140, Thailand.

[9] Chey Chanoeurn, Kosin Chamnongthai and Pinit Kumhomm, Khmer Printed Character Recognition by Using Wavelet Descriptor. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems Vol. 14, No. 3 337-350: World Scientific Publishing Company. 2006.

[10] John Platt, Published Book, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Microsoft Research, 1998.

[11] ARVIND C.S., NITHYA E. AND NABANITA BHATTACHARJEE, Kannada Language OCR System using SVM classifier. Journal of Information Systems and Communication, Volume 3, Issue 1, 2012.

[12] Accord.NET Framework. Online, November 2013. Available: http://accord-framework.net/

[13] Edge Detection. Online, September 2014. Available: http://en.wikipedia.org/wiki/Edge_detection