



# **Khmer OCR for Limon R1 Size 22 Report**

**09 July, 2009**

**Prepared by: Mr. ING LENG IENG**

**Cambodia Country Component**

**PAN Localization project**

**PAN Localization Cambodia (PLC) of IDRC**

## Table of Contents

1.	Introduction .....	3
2.	OCR Processes .....	3
2.1.	Pre-processing (Noise removal).....	3
2.1.1.	Connected Component .....	3
2.1.2.	Defining Threshold.....	4
2.2.	Segmentation .....	4
2.2.1.	Line Separation.....	4
2.2.2.	Character Separation.....	5
2.3.	Recognition .....	5
2.4.	Mapping.....	7
2.4.1.	Character Validation .....	7
2.4.2.	Character Sorting.....	8
2.4.3.	Character Elimination.....	9
2.4.4.	Character Swapping .....	10
3.	Training System.....	13
4.	Experimental Results.....	13
5.	Conclusion .....	14
6.	Future Works .....	14
	Appendix.....	15

## 1. Introduction

Khmer OCR for Limon R1, size 22 is the second research on Khmer OCR conducted by PAN Localization Cambodia of IDRC. The project aims at providing better algorithm for segmenting clean Khmer document printed in Limon R1 font. It demonstrates the significant changes in algorithm, code optimization and memory/resource management from the previous research on Limon S1 font. Moreover, it also illustrates researches we conducted such as Connected Component Labeling which is used throughout the project. The technique is very important because it is used to filter noise and extract character from the text.

## 2. OCR Processes

OCR life cycle involves four main processes.

### 2.1. *Pre-processing (Noise removal)*

Prior to the segmentation, the scanned document needs to be cleaned and segment-ready. This involves few processes such as skew correction, noise removal, etc. In our implementation, skew correction is still behind the scope. However, noise removal has been implemented. Although there are many types of noise, only one type of noise is taken into account, i.e. salt-and-pepper noise.

Salt-and-pepper noise is a kind of noise which is usually caused by small unnecessary dots produced by either the scanner or the source document itself. This type of noise is easy to eliminate through just two processes. First of all, we get all connected components. Then, by defining a constant threshold, those that are smaller than the pre-defined threshold are eliminated by altering their pixel value into 0s (white color).

#### 2.1.1. *Connected Component*

A Connected Component is a group of pixels accumulating together to form a shape. Getting the connected component is not an ease if memory management is weak. In other words, working with connected component extraction, memory should be first taken into account. There are two methods we have undertaken the experiments such as flood fill and two-pass methods. These methods can be found in another report, titled "Noise Removal".

From the experiment, the flood fill method is bulky in terms of both memory consumption and time consumption. Since we use recursive method and stack, there is one drawback. It could easily cause a stack overflow when working with large images. On the other hand, the two-pass method works great with large images because instead of recursively flood filling the connected

component, it walks pixel by pixel from left to right, top to bottom in order to label any pixel that belongs to the same group and should be extracted all as one. This will consume lesser memory. Moreover, this method has another advantage, i.e. we can record the coordinates of the pixel along the way. This advantage helps improve the speed of extracting the connected component by telling where in the image should we go to instead of restarting from the beginning of the image.

### 2.1.2. Defining Threshold

In order to determine whether the connected component should be eliminated, a threshold is defined. In this implementation, we define a static threshold value of 7 pixels for comparing the shape's width and height. If its width and height are less than 7 pixels, it is eliminated.

## 2.2. Segmentation

Since the implementation is done on the plain text, the segmentation is broken into two steps. The first step is Line Separation which involves the detection of line and the broken down of page into lines. The second step is Character Separation which involves the process of separating connected components from the line.

### 2.2.1. Line Separation

Line separation is a process which breaks the scanned document into lines. In this process, a projection profile method is used. Starting from the beginning of the image, we first of all detect each line by finding the start and the end of it. Then, within that range we extract that line and store it in a box object. We continue doing so until we reach the end of the document.

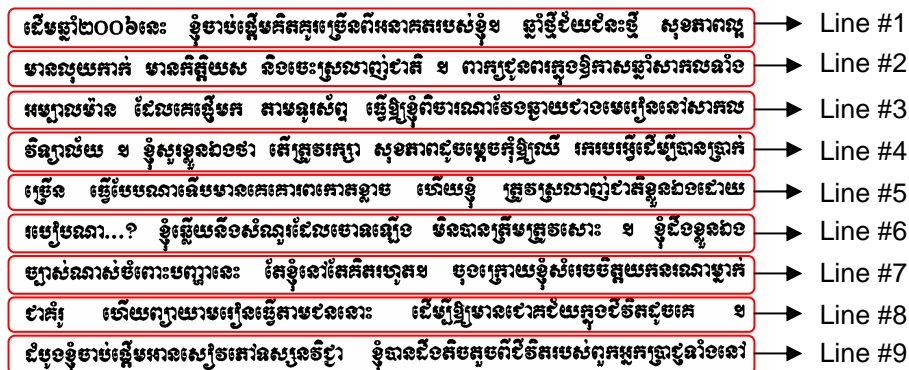


Figure 1 Line Separation

Furthermore, it should be noted that one line may be separated into several others. This caused by the gap between lower parts of the sentence. However, this rarely happens since it is

an ease to find distinct gap in a word but it almost never occurs in the sentence due to the dependency of other words in the same sentence. Therefore, we have got another condition before we accept the end of the line. This is done by measuring the gap between the start and the end of the line that have just been found. If the gap is less than 7 pixels, we keep the start but continue finding the new end of line until the gap between both the start and the end of line is greater or equal to 7 pixels.

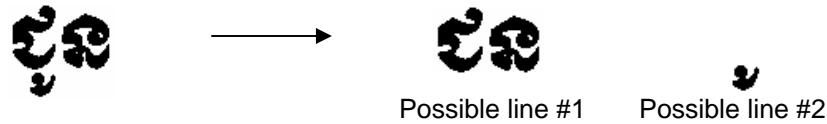


Figure 2 A Khmer word. Without a condition, this image will be extracted as two lines

### 2.2.2. Character Separation

Character separation is another process which separates all individual characters from the line. A character is any shape that is separable from others in the line. Connected component labeling and extraction technique are used. Two-pass method is used to label each connected component. Further details on how this method works can be found in "Noise Removal" report.

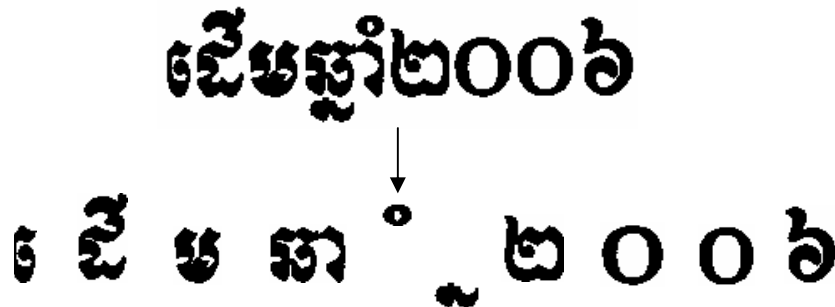


Figure 3 Character separation

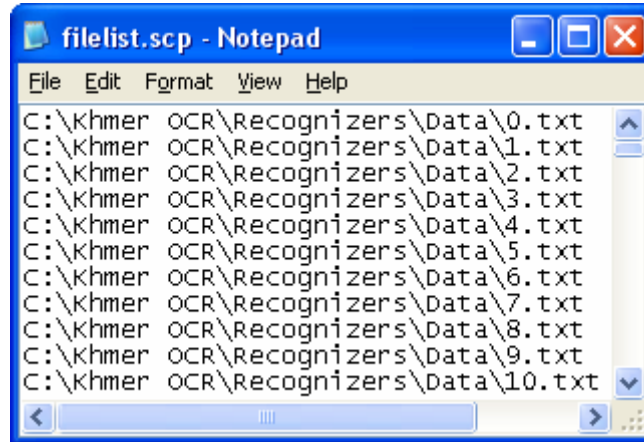
### 2.3. Recognition

Now it is ready for our shapes to get recognized. Prior to recognition, each must have gone through two processes such as framing and DCT (Discrete Cosine Transform) calculation. So, each shape is calculated into its equivalent DCT file which is the input for the recognition. Since our training system is based on Hidden Markov Model Toolkit (HTK) 3.2.1 developed at the University of Cambridge, the recognition is done under a tool shipped with the HTK called HVite. According to the HTK book, HTK provides HVite.exe for basic recognition as well as performing forced alignments, lattice rescoring, and recognize direct audio input. Likewise, we use HVite to

## PAN localization project

perform our character recognition. HTK provides HVite.exe as a shell program which means we can use the tool as a command line.

In this implementation, after all DCT files have been created in `\Recognizers\Data`, a file list called **filelist.scp** is also created. This file is an important parameter for HVite because it provides all file names for recognition. The following illustrates how it lists all the file names.



**Figure 4 filelist.scp, file list parameter for HVite.exe**

Then, in the command line, we call HVite.exe as follow:

```
HVite.exe -T 1 -S filelist.scp -H model/Single.mmf -i
Recognizers/Result.mlf -w def/net.slf def/dict.txt
model/hmmlist.txt"
```

Where:

-T 1: Set trace flag to 1

-S filelist.scp: Set script file to **trainlist.scp**. This file contains a list of all DCT files.

-H model/Single.mmf: Load Master Macro File (**model/Single.mmf**) which contains all HMM definitions. It should be noted that according to HTK book, large Master Macro File (MMF) will often be distributed as several files.

-i Recognizers/Result.mlf: Output transcription to the Master Label File (**Recognizers/Result.mlf**). This file is important because it contains the results for our recognition.

-w def/net.slf: Recognize from network file, **def/net.slf**.

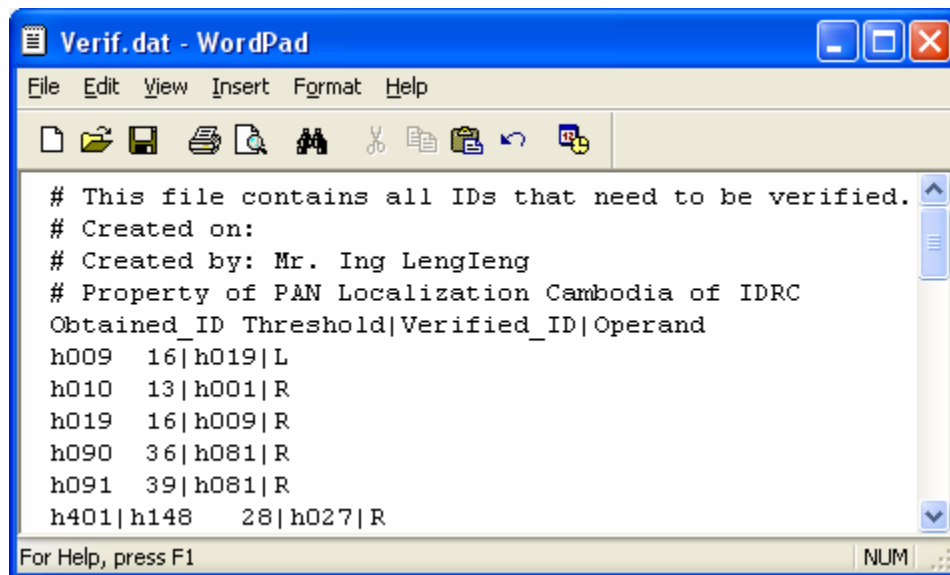
def/dict.txt: Dictionary file

## 2.4. Mapping

Mapping is the last process in Khmer OCR life cycle. This process, as the name suggests, involves significantly getting back the ASCII code of each obtained ID. Since Khmer is the language known in Asia as one of the complex writing systems, obtaining all the ASCII codes does not mean the end of the process yet. Further processes need to be done to gain higher accuracy rate. These processes include Character Validation, Character Sorting, Character Elimination, and Character Swapping.

### 2.4.1. Character Validation

Character Validation is a process to verify that the returned ID matches the real shape in the image. Frequently, confusion arises between two shapes is when the first one is identical to the portion of the second one which means the first is smaller in width to the latter. Consequently, once the given shape is the first, the recognition system returns the ID of the second shape. Therefore, the solution to dealing with this issue is to have all confused IDs written in a file (Verif.dat in our project) together with the threshold width. Then, the code will compare the width of the given shape with the threshold width in the file. As the result, whether ID of the given shape is preserved depends on two things. First, whether it is greater or smaller than the threshold width. Second, the information given in file indicates which one, the ID or the replacement ID, to be kept. The following is the structure of the file.



```
# This file contains all IDs that need to be verified.
# Created on:
# Created by: Mr. Ing LengIeng
# Property of PAN Localization Cambodia of IDRC
Obtained_ID Threshold|Verified_ID|Operand
h009 16|h019|L
h010 13|h001|R
h019 16|h009|R
h090 36|h081|R
h091 39|h081|R
h401|h148 28|h027|R
```

Figure 5 Verif.dat

In the above figure, the first several lines are comment lines, each of which starts with a # (pound sign). Then, there is a column header line which includes two columns such as

Obtained\_ID and Threshold|Verified\_ID|Operand. Any ID that belongs to the Obtained\_ID column is the ID which is obtained from the recognition system. In other words, this ID is the ID that needs to be verified. Threshold is the mean value obtained from (**highest width of the Verified\_ID - highest width of the Obtained\_ID**) / 2 + **highest width of the Obtained\_ID**. For example, the highest width value of the Verified\_ID is **20** and the highest width value of the Obtained\_ID is **10**. The mean value is **(20-10)/2 + 10 = 15**. Verified\_ID is the ID with which the Obtained\_ID is verified. Operand indicates the comparison operation the code should take. The **L** value tells the code to do the following comparison:

If the Obtained\_ID's width **is greater than** the Threshold value, assigns the Verified\_ID to the Obtained\_ID.

On the other hand, the **R** value tells the code to perform the opposite comparison operation.

If the Obtained\_ID's width **is less than** the Threshold value, assigns the Verified\_ID to the Obtained\_ID.

In addition, the Obtained\_ID in the last line in figure 5 indicates that there are two IDs (h401 and h148) confuse with the ID on the second column (h027). The | (pipe) sign tells either one of the two IDs should be checked in this condition.

## 2.4.2. Character Sorting

Character sorting is a process used to arrange all the characters of each line by Limon writing order base on their coordinate. The idea is very straightforward. First from left to right we arrange all the shapes by their coordinate. We implement bubble sort algorithm for this. Each shape's started position is used for the sorting. The following illustrates how this process works.

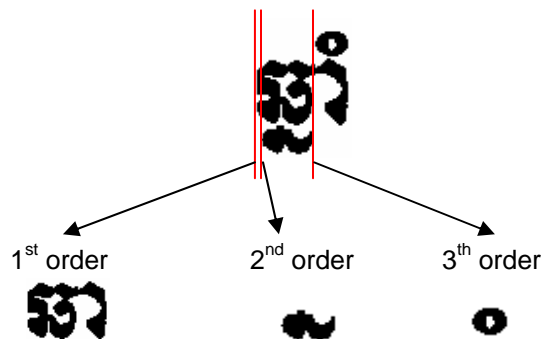


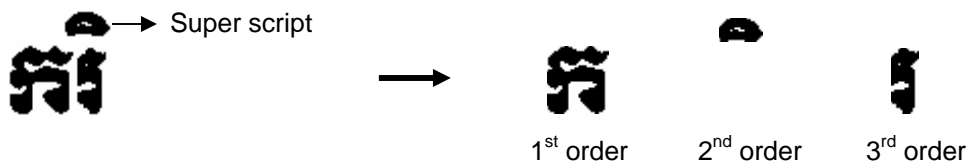
Figure 6 Character Sorting

The idea to sort as this started from the analysis of the Limon font nature. Generally, walking from left to right, we will encounter Complex Character (CC) first followed by the base glyph, the

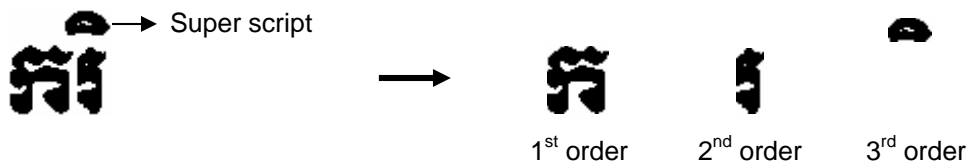


sub script and finally the super script. However, such case does not work for all cases. In some cases, the sub script is encountered first followed by the super script and finally the base glyph. In such case not only is the order wrong, but also does the complexity arise on which base glyph both the sub script and the super script belong. This means that when this case occurs, either sub script or super script covers on both the base glyph on its left hand side and the base glyph on its right hand side (see figure 7). Such issue requires further investigated code to give a better decision on where should the sub script or the super script reside.

In order to determine whether the detected sub script or super script belongs to the left hand side (body1) or the right hand side (body2) base glyph, we compare number of pixels the shape covers on body1 with those it covers on body2. The larger amount the shape covers determines where it belongs. Hence, if the shape covers more on body1, it belongs to the body1.



**Figure 7 Super script covers on both base glyphs. Without applying the investigated code, the order is incorrect**



**Figure 8 Super script covers on both base glyphs. After applying the investigated code, the order is correct**

### ***2.4.3. Character Elimination***

Character elimination is also an important process in this project. In some cases, two shapes represent only one letter, and thus they produce only one ASCII code. In other cases, two shapes represent more than one letter, and that produces more than one ASCII code. Therefore, this process will take care of those cases.

The code will look at the information written in the `Elim.dat` file under `/Recognizers/Code Files` directory. This file has structure as the below figure.

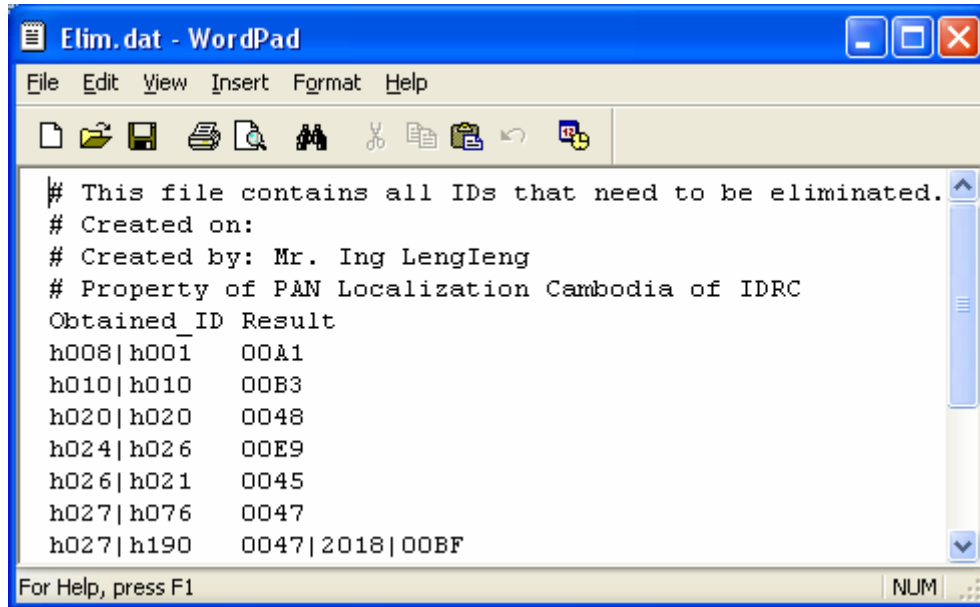


Figure 9 Elim.dat

The first few lines are comment lines. The code will ignore these because they start with the # (pound) sign. After the comments, there is a column header line which consists of two column headers. The first column header is `Obtained_ID` followed by the second one, `Result`. Then, the rest is the information needed by the application. In the first column, there are two IDs separated by the | (pipe) sign. Instead of looking at one of these, the code matches two IDs obtained from the recognition system with both of them. If only one side matches, the condition will not have an effect. For instance, if the two IDs obtained are `h008` and `h011`, by comparing the IDs in the `Elim.dat` file, in the first column (see figure 9) `h008` matches but not the other; thus, this condition is not true.

Furthermore, as above mentioned, two IDs may represent more than one shape which produces more than one ASCII code. The last line in figure 9 is the example proves this.

#### ***2.4.4. Character Swapping***

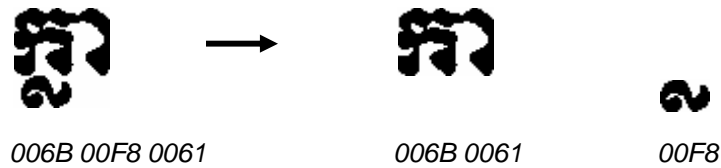
Character swapping is another process used once the obtained ASCII code is needed to be inserted in between the previous two ASCII codes. This applies to the following four cases:

- The obtained ASCII code is Khmer sign MUUSIKATOAN (003A), and the shape before it has two ASCII codes, particularly the second one is either Khmer vowel sign AA (0061) or Khmer vowel sign AU (0041).



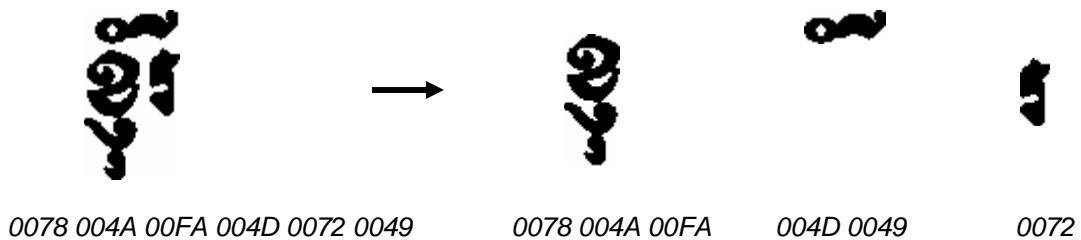
**Figure 10 Character Swapping. Case 1**

- The obtained ASCII code is any sub script, and the shape before it has two ASCII codes, particularly the second one is either Khmer vowel sign AA (0061) or Khmer vowel sign AU (0041).



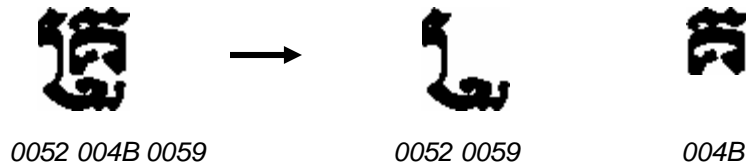
**Figure 11 Character Swapping. Case 2\**

- The obtained ASCII code is a parallel super script, i.e. one shape consists of two ASCII codes. This case one part of the two belongs to the left hand side body, whereas the other belongs to right hand side body of it. Therefore, after the operation the counter is incremented by 1 since the next body is already encountered.



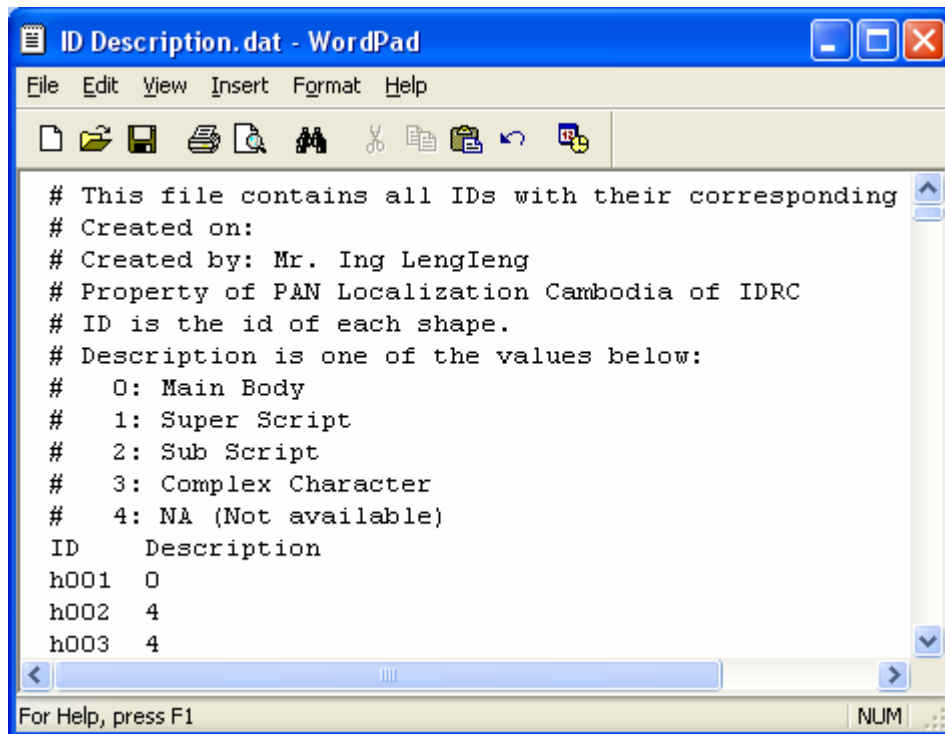
**Figure 12 Character Swapping. Case 3**

- The obtained shape has two ASCII codes, where one is a Complex Character (CC) and the other is a subscript. In this case, the code will check for Main Body (consonant) on the next index. If the next index is the Main Body, the code will swap that Main Body with the second ASCII code.



**Figure 13 Character Swapping. Case 4**

Moreover, it should be noted that the code can understand whether the shape belongs to which of the categories such as main body, super script, sub script, or Complex Character (CC) by getting the information from one file named `ID Description.dat` under `/Recognizers/Code Files` directory. The following is the structure of the file:



**Figure 14 ID Description.dat**

Being created with the same structure as other files, `ID Description.dat` starts with a few lines as comment lines followed by one line as a column header, and the rest as the information which consists of ID on the first column with their corresponding description on the second column. The description in this file is essential for the code since it tells which shape belongs to the one of the below categories:

- 0 means Main Body
- 1 means Super Script
- 2 means Sub Script
- 3 means Complex Character

- 4 means Not Available

### 3. Training System

In whatever circumstances, training is much important for every aspect of learning procedure. As human get trained to be a professional, animal get trained to be smarter, machine is also trained to understand and work on the specific task. In reality, the more you get trained, the more professional you are. Likewise, if the machine is trained to the maximum capacity, the result will be satisfied.

In this project, the training is based on HTK 3.2.1 developed at the University of Cambridge. HTK which stands for Hidden Markov Model Toolkit is a tool primarily designed to build and manipulate the hidden markov model for speech recognition. However, it has been widely used in many natural language processing fields. In that, character recognition can also use HTK for training.

In our project, each shape has at most 999 samples for training although some have less than 20 samples due to its rareness. Prior to the training, each shape is framed with a frame size of 84x4. Then, all the frame parts of each shape are calculated into a DCT (Discrete Cosine Transform) file which is the input for the training. Each shape is identified by id in the Master Macro File (MMF) which is the important database because it contains all the HMM definitions. In HTK there are important numbers of file such as Dictionary, Grammar, Network, HMMList, and Master Macro File (MMF), each of which is appended during the training procedure.

### 4. Experimental Results

Text #	Number of shapes	Correct	Incorrect	Accuracy (%)
1	1883	1877	6	99.68
2	1439	1423	16	99.88
3	1666	1653	13	99.21
4	1666	1645	21	98.73
5	1517	1505	12	99.20
6	1576	1562	14	99.11
7	1823	1804	19	98.95
8	1859	1816	43	97.68
9	1839	1818	21	98.85

10	1752	1727	25	98.57
<b>Total</b>	<b>17020</b>	<b>16830</b>	<b>165</b>	<b>98.88</b>

From the experiment, it can be said that the result reaches the satisfaction level. There are four problems arise from the recognition system. First of all, most of the shapes in the super script category confuse with each other, i.e. this code 0049 almost always confuse with 0057. Although our system has the ability to distinguish between two shapes once the ID is obtained, this problem is a challenging task for the system since both of the shapes are of the same size, i.e. their width are equal. Secondly, a few sub scripts also have confusion issue; for instance, this one 00B5 confuses with this one 00F0. Not a far cry different from the first issue, this case is not an ease for the system to distinguish which one is the genuine one. Thirdly, a partial error caused by the shapes which haven't trained to the system yet. This problem is not a severe one since it can be solved by getting those shapes to be trained. Lastly, there are few problems caused by the mapping. Such problems can be solved by improving the mapping code.

## 5. Conclusion

In conclusion, this project is an important research for any further Khmer OCR development since we have implemented the generic algorithms which can be found very useful. Such an algorithm as Connected Component Labeling and Extraction is widely practiced in most of the image processing fields such as image segmentation, noise removal, so on and so forth. Moreover, this project is highly designed for easy modification and enhancement because we have the configuration files which drive the application's operation. It also provides an image control which provides some basic image functionalities such as Fit-to-Window, View original size, Zoom In, and Zoom out.

However, the application has several drawbacks. First of all, HMM database (`Single.mmf`) is very large in size which requires longer loading time once HVite application is called. This is due to the fact that we trained all 491 shapes altogether in solely one database file. Furthermore, for each operation, it consumes a large amount of memory plus the small amount of memory usage increment for the next operation. Hence, the memory optimization should be taken into account.

## 6. Future Works

Further enhancement on the project should be conducted. First of all, the application's output should be in Unicode format which can be viewed universally. Secondly, the application should support multiple input formats since currently it supports only BMP file format. Last but not least,

the application should provide more features such as the ability for the user to optionally select the region in the text for recognition, separate the image from the text, detect and correct document skew, etc.

## Appendix

ID	Shape	ID	Shape	ID	Shape
h001	•	h002	•	h003	o
h004	,	h005	,	h006	)
h007	)	h008	-	h009	•
h010	•	h011	+	h012	+
h013	/	h014	/	h015	?
h016	२	h017	३	h018	४
h019	५	h020	६	h021	७
h022	८	h023	९	h024	०
h025	१	h026	२	h027	३
h028	४	h029	५	h030	६
h031	७	h032	८	h033	९
h034	०	h035	१	h036	२
h037	३	h038	४	h039	५
h040	६	h041	७	h042	८
h043	९	h044	०	h045	१
h046	२	h047	३	h048	४
h049	५	h050	६	h051	७
h052	८	h053	९	h054	०
h055	१	h056	२	h057	३
h058	४	h059	५	h060	६
h061	७	h062	८	h063	९
h064	०	h065	१	h066	२

h067	ക	h068	ച	h069	ഘ
h070	ഘ	h071	ഘ	h072	ഘ
h073	ഘ	h074	ഘ	h075	ഘ
h076	ഘ	h077	ഘ	h078	ഘ
h079	ഘ	h080	ഘ	h081	ഘ
h082	ഘ	h083	ഘ	h084	ഘ
h085	ഘ	h086	ഘ	h087	ഘ
h088	ഘ	h089	ഘ	h090	ഘ
h091	ഘ	h092	ഘ	h093	1
h094	2	h095	3	h096	4
h097	5	h098	5	h099	6
h100	7	h101	8	h102	9
h103	0	h104	9	h105	0
h106	൫	h107	൫	h108	൫
h109	൫	h110	൫	h111	൫
h112	൫	h113	൫	h114	൫
h115	൫	h116	൫	h117	൫
h118	൫	h119	൫	h120	൫
h121	൫	h122	൫	h123	൫
h124	൫	h125	൫	h126	൫
h127	൫	h128	൫	h129	൫
h130	൫	h131	൫	h132	൫
h133	൫	h134	൫	h135	൫



h136	ନ	h137	ଫ	h138	ଝ
h139	ଦ	h140	ଖ	h141	ଟ
h142	ଢ	h143	ଝ	h144	ଠ
h145	ଢ	h146	ଝ	h147	ଠ
h148	ଝ	h149	ଝ	h150	ଠ
h151	ଠ	h152	ଠ	h153	ଠ
h154	ଠ	h155	ଠ	h156	ଠ
h157	ଠ	h158	ଠ	h159	ଠ
h160	ଠ	h161	ଠ	h162	ଠ
h163	ଠ	h164	ଠ	h165	ଠ
h166	ଠ	h167	ଠ	h168	ଠ
h169	ଠ	h170	ଠ	h171	ଠ
h172	ଠ	h173	ଠ	h174	ଠ
h175	ଠ	h176	ଠ	h177	ଠ
h178	ଠ	h179	ଠ	h180	ଠ
h181	ଠ	h182	ଠ	h183	ଠ
h184	ଠ	h185	ଠ	h186	ଠ
h187	ଠ	h188	ଠ	h189	ଠ

h190	ഹ	h191	ജ	h192	ള
h193	ള	h194	ള	h195	ള
h196	ച	h197	ച	h198	ച
h199	ച	h200	ച	h201	ച
h202	ച	h203	ച	h204	ച
h205	ച്ച	h206	ച്ച	h207	ച്ച
h208	ച്ച	h209	ച്ച	h210	ച്ച
h211	ജ	h212	ജ	h213	ജ
h214	ജ	h215	ജ	h216	ജ
h217	ജ	h218	ജ	h219	ജ
h220	ജ	h221	ജ	h222	ജ
h223	ള	h224	ള	h225	ള
h226	ള	h227	ള	h228	ള
h229	ള	h230	ള	h231	ള
h232	ള	h233	ള	h234	ള
h235	ള	h236	ള	h237	ള
h238	ള	h239	ള	h240	ള

h241	အ	h242	အ	h243	အ
h244	အ	h245	အ	h246	အ
h247	အ	h248	အ	h249	အ
h250	အ	h251	အ	h252	အ
h253	အ	h254	အ	h255	အ
h256	အ	h257	အ	h258	အ
h259	အ	h260	အ	h261	အ
h262	အ	h263	အ	h264	အ
h265	အ	h266	အ	h267	အ
h268	အ	h269	အ	h270	အ
h271	အ	h272	အ	h273	အ
h274	အ	h275	အ	h276	အ
h277	အ	h278	အ	h279	အ
h280	အ	h281	အ	h282	အ
h283	အ	h284	အ	h285	အ
h286	အ	h287	အ	h288	အ
h289	အ	h290	အ	h291	အ

h292	ಶಿ	h293	ಶಿ	h294	ಶಿ
h295	ಶಿ	h296	ಶಿ	h297	ಶಿ
h298	ಶಿ	h299	ಶಿ	h300	ಶಿ
h301	ಶಿ	h302	ಶಿ	h303	ಶಿ
h304	ಶಿ	h305	ಶಿ	h306	ಶಿ
h307	ಶಿ	h308	ಶಿ	h309	ಶಿ
h310	ಶಿ	h311	ಶಿ	h312	ಶಿ
h313	ಶಿ	h314	ಶಿ	h315	ಶಿ
h316	ಶಿ	h317	ಶಿ	h318	ಶಿ
h319	ಶಿ	h320	ಶಿ	h321	ಶಿ
h322	ಶಿ	h323	ಶಿ	h324	ಶಿ
h325	ಶಿ	h326	ಶಿ	h327	ಶಿ
h328	ಶಿ	h329	ಶಿ	h330	ಶಿ
h331	ಶಿ	h332	ಶಿ	h333	ಶಿ
h334	ಶಿ	h335	ಶಿ	h336	ಶಿ
h337	ಶಿ	h338	ಶಿ	h339	ಶಿ

h340	ညို့	h341	ညို့	h342	န့
h343	န့	h344	န့	h345	န့
h346	န့	h347	န့	h348	န့
h349	န့	h350	န့	h351	န့
h352	န့	h353	န့	h354	န့
h355	န့	h356	န့	h357	န့
h358	န့	h359	န့	h360	န့
h361	န့	h362	န့	h363	န့
h364	န့	h365	န့	h366	န့
h367	န့	h368	န့	h369	န့
h370	န့	h371	န့	h372	န့
h373	န့	h374	န့	h375	န့
h376	န့	h377	န့	h378	န့
h379	န့	h380	န့	h381	န့
h382	န့	h383	န့	h384	န့
h385	န့	h386	န့	h387	န့
h388	န့	h389	န့	h390	န့
h391	န့	h392	န့	h393	န့
h394	န့	h395	န့	h396	န့

h397	ဖာ	h398	ဖာ်	h399	ဖာ
h400	ဖာ်	h401	ဂ	h402	ဂ်
h403	ဂဖာ	h404	ဂဖာ်	h405	ဂဖာ
h406	ဂဖာ်	h407	ဟာ	h408	ဟာ်
h409	ဟာ	h410	ဟာ်	h411	အျာ
h412	အျာ်	h413	အာ	h414	ဂ
h415	ါ	h416	ါ်	h417	ါ
h418	ါ်	h419	ဲ	h420	ဲ
h421	ဲ	h422	ဲ	h423	ဲ
h424	ဲ	h425	ဲ	h426	ဲ
h427	ဲ	h428	ဲ	h429	ဲ
h430	ဲ	h431	ဲ	h432	ဲ
h433	ဲ	h434	ဲ	h435	ဲ
h436	ဲ	h437	ဲ	h438	ဲ
h439	ဲ	h440	ဲ	h441	ဲ
h442	ဲ	h443	ဲ	h444	ဲ
h445	ဲ	h446	ဲ	h447	ဲ

h448	ကော	h449	ကု	h450	ကုာ
h451	ကူ	h452	ကုာ	h453	ကွ
h454	ဈ	h455	ဉ	h456	ဉ
h457	ဉ	h458	ဉ	h459	ဉ
h460	က	h461	က	h462	က
h463	ကော	h464	က	h465	က
h466	က	h467	က	h468	က
h469	ဆ	h470	ဆ	h471	ဆ
h472	က	h473	က	h474	က
h475	က	h476	က	h477	က
h478	န	h479	န	h480	န
h481	ဆ	h482	ဆ	h483	ဆ
h484	က	h485	က	h486	က
h487	က	h488	က	h489	က
h490	ဆ	h491	ဆ		