An evaluation of HMM-based Techniques for the Recognition of Screen Rendered Text

Sheikh Faisal Rashid[1], Faisal Shafait[2], and Thomas M. Breuel[1]
[1]*Technical University of Kaiserslautern, Kaiserslautern, Germany*
[2]*German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany*
s_rashid09@informatik.uni-kl.de, faisal.shafait@dfki.de, tmb@informatik.uni-kl.de

***Abstract*—Segmentation and recognition of screen rendered text is a challenging task due to its low resolution (72 or 96 ppi) and use of anti-aliased rendering. This paper evaluates Hidden Markov Model (HMM) techniques for OCR of low resolution text–both on screen rendered isolated characters and screen rendered text-lines–and compares it with the performance of other commercial and open source OCR systems. Results show that HMM-based methods reach the performance of other methods on screen rendered text and yield above 98% character level accuracies on both screen rendered text-lines and characters.**

***Keywords*-Low resolution text recognition; Screen rendered text; OCR; Hidden Markov Model (HMM)**

I. Introduction

Most OCR systems are developed for text recognition from scanned document images. However, recognition of low resolution text is quite interesting due to wide range of applications and occurrence of low resolution text in screen-shots, images and videos. For example, recognition of screen rendered text can facilitate dictionary or language translation tools [1] to provide meanings or translation of text from screen-shots of documents or web images. Other possible applications may include:

- Augmenting screen reading tools for blind or visually impaired people for reading text from screen images where ASCII text is not available on clipboard.
- Recognizing low resolution text in videos [2].
- Automating GUI testing tools for correcting the spelling mistakes on GUI screen-shots.
- Useful for correcting web page rendering errors due to bad foreground and background color combination.
- Enabling web indexing tools to capture semantically important information from web images.
- Protection against phishing [3] attacks by verifying URLs of potentially important websites against similar looking characters that have different Unicode.

OCR of screen rendered text is challenging due to low resolution (72 or 96 ppi) and small font size. For example the x-height of most of lower case letters can only be four pixels and width can only be two pixels. For better visual perception, the rendering process also smooths the low resolution text by means of anti-aliasing. This smoothing process causes another problem of segmentation. As most of the OCR methodologies are based on recognition of isolated characters, the smoothed and small sized screen rendered characters are difficult to segment due to touching or noise. Figure 1 shows some example images taken from screen rendered text-lines.

Figure 1. Exmples: screen-rendered text images

In this paper, we evaluate segmentation free HMM-based techniques for the recognition of screen-rendered characters and text-lines and compare their performance with current state-of-the art open-source and commercial OCR engines. We use edit distance [4] to compute character level recognition accuracy. Results show a promising performance of the proposed HMM based method in recognition of screen rendered low resolution text.

A. Related Work

Although majority of existing research in optical character recognition (OCR) deals with standard (300 dpi) printed text, a few techniques handle the special case of low resolution text recognition. Wachenfeld et al. [5] presented a hybrid classification approach for recognition of screen rendered characters and words. They normalized and over segmented the gray scaled word pixels into small subcomponents. Their approach is inspired by Breuel's method [6] for handwritten text segmentation using dynamic programming but with different cost criteria. During over-segmentation, the resulting segment is instantly classified by using a gray scale

character classifier. Further splitting is stopped if a character is classified with higher plausibility. They evaluate their method on a public database of screen-rendered character and words [7]. By using leave-one-out validation technique they achieved 98.91% recognition accuracy on 15,808 non-italic, non-bold characters. Einsele et al. [8], [9] proposed HMM based techniques to recognize isolated characters and words taken from screenshots of web pages. The potential application is to provide an OCR facility for web indexing tools to extract and index semantic information present in web images. They evaluate performance of their system on a dataset of 3,000 word images using two font families in variable sizes. They achieved overall 96% recognition accuracy at 10 pts font size, but the accuracy drops to 90% at 6 pts font size. Jacobs et al. [10] presented a convolutional neural network approach for recognition of low resolution text captured by 1024x768 web-cam and 10 pts font size. The overall performance of their system is in between 80% to 95% word level recognition accuracy. Yanadume [11] proposed a method that uses multi-frame images taken by DV or phone camera. Their method is based on subspaces and they achieved 92% recognition rate with phone camera and 99.9% with DV camera that provides good quality images. Krämer [12] employed HMM based techniques to recognize screen rendered text using simple pixel based features.

II. System Description

A. Hidden Markov Models

Hidden Markov Models (HMMs) have been successfully used in continuous speech, handwritten and cursive script text recognition tasks [13], [14], [15]. The advantage of HMMs in cursive or handwritten text recognition is to recognize connected characters without segmenting them into smaller units. As segmentation of low resolution text is hard due to anti-aliasing, therefore we evaluate HMM based techniques on this specific task. HMMs are statistical models in which system being modeled is considered as a Markov process that have unobserved or hidden states [16]. In a Hidden Markov Model, state is not directly visible but it is associated with a probability distribution over all possible output values. Each state is associated to an input pattern and is modeled by a probability distribution function (pdf). All the experiments for building HMMs are done using Hidden Markov Toolkit (HTK) [17]. HTK is a portable toolkit that is preliminary developed for building speech recognition systems and is also used for optical character recognition tasks [18].

B. Problem Description

HMMs can model the variability of underlying data as function of one independent variable. In speech processing systems, time is the natural independent variable but in case of images which have 2-dimensions horizontal axis has been taken as an independent variable by most of researchers. A text-line can be considered as sequence of characters and each character can be represented by a sequence of features or observations O, defined as $O = o_1, o_2, ...o_N$ where o_t is the observation or feature vector at pixel position t along horizontal axis. The character recognition problem is regarded as computing the $argmax_i\{P(c_i/O)\}$. where c_i is the specific character to be recognized. By using Bayes theorem, we have $P(c_i/O) = \frac{P(O/c_i)P(c_i)}{P(O)}$ and thus for a given set of prior probabilities $P(c_i)$, the most probable character is given by the likelihood $P(O/c_i)$.

C. Data Sets

We use two different datasets of screen rendered text for evaluation of proposed HMM based methods and current state-of-the art OCR engines. A dataset of screen rendered characters has been develped by Wachenfeld et al. [7]. This dataset contains 28,080 upper and lower case Roman characters both in different font styles and sizes. We use a subset that have 15,808 no-italic and non-bold characters for evaluation of all the participating OCR techniques. This subset is also used by Wachenfeld et al. [5] for the evaluation of their own method. Due to visual ambiguity in some lowercase and uppercase characters, these characters are merged into lowercase characters. These merged characters are: c/C, o/O, p/P, s/S, v/V, w/W, x/X and z/Z. This character level dataset is limited (isolated characters only) and is not sufficient for the true evaluation of OCR techniques. A screen rendered text-line could be a better choice because of having all the variations and challenges of the screen rendered text. The choice of text-lines is to avoid complex layout analysis, as the focus is only to measure text recognition accuracy for screen rendered data. Due to unavailability of proper text-lines dataset, we have generated our own dataset of screen rendered text-lines. We use mixture of public serif and sans-serif fonts for rendering text-lines from Project Gutenberg [19] e-book. Text-lines are rendered with different line heights ranging from 10 to 20 pixels high. This dataset comprises of 2,873 text-lines having upper and lower case Roman alphabets, numerals, punctuations and brackets.

D. Preprocessing and Features Extraction

The preprocessing mainly concerns with text image height normalization. The screen rendered character images are normalized to a height of 10 pixels. This is the average height of all characters in the dataset. In case of screen rendered text-lines, we first remove extra white space around the text-line and then normalize it to height of 20 pixels. The text-line image height normalization is empirically determined. We get better recognition accuracies with 20 pixels height normalized text-lines because at this height, the x-height of lower case characters present in the text-line is sufficient for recognition. After text image height normalization, we

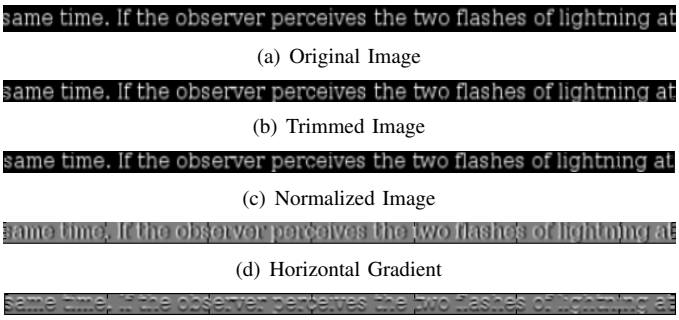
extract two different kind of features from gray scale text images using traditional sliding window approach.

1) Gray scale raw pixel features: Since we are working with low resolution text images and in this special case direct use of gray scale values is possible. The gray scale raw pixel features are obtained by vertically slicing the normalized character image or normalized text-line image into one pixel wide slices. A feature vector is constructed by simply picking gray scale values in each vertical slice. The text image is traversed in horizontal direction and a feature vector is extracted at each pixel position from left to right. The dimension of feature vector is fixed and is based on normalized height of each text image. These kind of features have also been used by Krämer [12].

2) Gradient based gray level intensity features: Gradient based gray level intensity features are modified form of features that have been used for Arabic script recognition [18]. A height normalized text-line image is first divided into a sequence of overlapping windows in horizontal direction. These windows are then subdivided into overlapping cells in vertical direction. Features are extracted by sliding a window over text image from left to right and concatenating the gray level intensities and change in gray level intensities along horizontal and vertical direction of the image. The gray level intensity values are computed from top to bottom by counting the numbers of gray values that are greater than zero in each cell of a particular window. Similarly, the change in gray level intensities are measured from top to bottom by counting the numbers of positive and negative gradients in each cell of a particular window. These gradients are computed with the help of Sobel operator along horizontal and vertical axes of the image. Two different gradient based intensity features are evaluated for screen rendered text-lines, namely G-3231 and G-3232. In both features, the width of window and height of each cell is fixed to three pixels i.e, the dimensions of each cell are 3 x 3. The difference is that, in G-3231, we have two pixels overlap in consecutive windows (along horizontal axis) and one pixel overlap in consecutive cells (along vertical axis) and in G-3232, we have two pixels overlap in consecutive windows and two pixels overlap in consecutive cells. These features are computed for text-lines only because some screen characters are not large enough to have a 3 x 3 window inside. Figure 2 shows an example of text-line image along with its normalized, and gradient images from text-lines dataset.

E. HMMs Topology and Training

The presented method models each character with a multi-state, left to right, continuous density HMM. A text-line is modeled by concatenating the models for each character in the text-line. Each state has an associated output probability distribution over the features and is modeled by Gaussian mixture densities. A Gaussian mixture is parameterized by



(a) Original Image

(b) Trimmed Image

(c) Normalized Image

(d) Horizontal Gradient

(e) Vertical Gradient

Figure 2. Features extraction steps from screen rendered text-lines

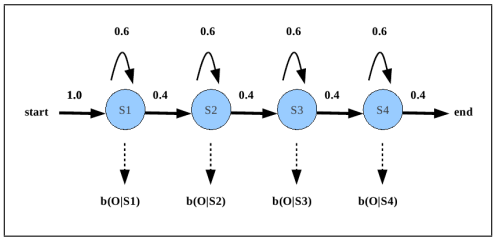


Figure 3. Four states left to right HMM topology for character models

means and variances of component Gaussians. The number of mixture model densities, number of states and allowable transitions among states are system parameters. The required number of densities in a mixture model depends on variability in features along vertical axis. The number of states that are adequate for each character model depends on horizontal variability of each character. The two parameters–numbers of state per model and number of Gaussian mixture densities–are needed to be tuned empirically. In our experiments we used 3 to 6 states HMMs with 256 Gaussian mixture densities and each class of character is modeled with same number states. Figure 3 shows a four state, left to right HMM with self loops and transition to adjacent states with no skip. The "start" and "end" are non-emitting states and are used to provide transitions from one character model to other character model. Text lines are modeled

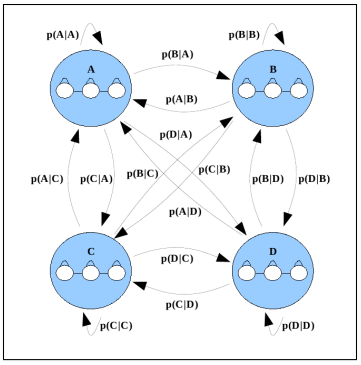
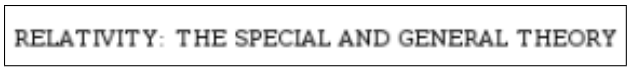


Figure 4. Ergodic HMM Architecture



(a) Original Image

(b) OCR of Original Image

R I ITY: TH I ND N R TH ORY

(c) OCR of Image Text Rendered in Notepad

RELATIVITY: THE SPECIAL AND GENERAL THEORY

(d) OCR of Image Text Rendered in Command Prompt

Figure 5. OCR outputs on an example image by Screen OCR 7.9

by concatenating the character models in ergodic structure as shown in Figure 4. Training or estimating the HMM parameters is performed using Baum-Welch re-estimation algorithm [20], which iteratively aligns the feature vectors with the character models in maximum likelihood sense.

F. Recognition

After height normalization and feature extraction, as described above, recognition process searches for character models or a sequence of character models that has the highest probability of having generated the given sequence of feature vectors. This search process requires trained character models, a possible word lexicon or dictionary, and a statistical language model. The recognition is performed using a variant of Viterbi algorithm called "Token Passing Model" [20] to perform best path search in combinations of different character level models. The choice of lexicon and language model is optional. In presented system, we employ lexicon free, open-vocabulary approach by building HMM model with ergodic topology at character level. This ergodic structure, as shown in Figure 4, has an implicit character level bi-gram. Higher level language modeling at character or word level can be possible and its use generally results in a lower error rate. For simplicity, we are not providing any explicit language model at this stage but this can be a potential future step in system enhancement.

III. Experimental Results and Evaluations

We evaluate state-of-the-art open-source and commercial OCR engines for low resolution text recognition and compare their performance with the proposed HMM based method. The participating OCR engines are ABBYY FineReader 10 professional [21] and Tesseract OCR engine [22]. We also evaluate the specialized screen capture and recognition tools like Screen OCR 7.9 (trial version) [23] and ABBYY FineReader built in Screenshot Reader utility on some example text-lines. The evaluation of these utilities are performed on limited test examples due to lack of batch processing support for large number of images. We obtain almost same results for both ABBY Screenshot Reader utility and ABBY FineReader. The OCR results of a sample text-line from Screen OCR 7.9 are shown in Figure 5. Screen OCR 7.9 did not perform well on actual text-line images but it gives good recognition for text rendered with black background.

The performance evaluation is carried out by computing character recognition accuracy percentage (CRA%) with the help of following formula

$$CRA\% = \frac{N - ED}{N} * 100 \quad (1)$$

where N = Total number of characters and
ED = Edit Distance = Nos. of deletions + Nos. of insertions + Nos. of substitutions (with equal cost).

Table I and Table II present experimental results for both screen rendered characters and text-lines datasets respectively. HMM based techniques are evaluated with three different kind of features and we obtain above 98% character level recognition accuracies for these two datasets. In our experiments we model HMMs with different number of states, varying from 3 to 6 states per character model. We obtain better results with 3 state HMM for isolated character dataset and with 4 state HMM for text-line dataset. Experimental results show that HMM based OCR techniques reach the performance of commercial OCR engine–ABBYFine Reader–with around 1% less recognition accuracy for screen rendered text-lines. In the case of screen rendered characters, HMM based OCR techniques perform better than other participating OCR engines but we have around 0.5% less recognition accuracy in comparison to method proposed by Wachenfeld et al.[5]. We use 20% from each dataset for the evaluation of all participating methods and 80% is used for HMM training. Tesseract OCR does not perform well on these special low resolution screen rendered text datasets, therefore we rescale the images to 2 times and 3 times of their original sizes. After rescaling we re-evaluate the performance of participating OCR engines. This rescaling improves the recognition accuracies of Tesseract. ABBYY FineReader behavior is consistent with these rescaled images due to built in normalization mechanism that enhances the resolution of these low resolution screen images to 300 dpi.

IV. Conclusion

In this work we evaluate the HMM based techniques and existing state-of-the-art OCR engines for recognition of screen rendered characters and text-lines. Due to very low resolution and small size, OCR of screen rendered text requires specialized approaches. Segmentation of screen rendered text lines is also challenging due to touching of characters with each other. This touching appears as a side effect due to anti-aliasing in rendering process. Recognition

Table I
RECOGNITION ACCURACIES FOR SCREEN RENDERED CHARACTERS DATABASE

Algorithms	Standard		2 times Rescaled		3 times Rescaled	
	Merged Classes	Non-merged Classes	Merged Classes	Non-merged Classes	Merged Classes	Non-merged Classes
HMM	98.35	na	na	na	na	na
Wachenfeld	98.91	na	na	na	na	na
ABBYY	52.63	42.1	39.17	29.52	46.67	35.68
Tesseract	53.78	45.84	68.89	58.22	69.46	58.16

Table II
RECOGNITION ACCURACIES FOR SCREEN RENDERED TEXT-LINES

Algorithms	Standard	2 times Rescaled	3 times Rescaled
HMM-Pixels	98.54	na	na
HMM-G3231	98.27	na	na
HMM-G3232	98.30	na	na
ABBYY	99.73	99.67	99.57
Tesseract	75.32	97.24	98.41

results show that HMM based, open vocabulary, segmentation free techniques performs quite well in recognition of low resolution text. ABBY FineReader gives almost perfect recognition and Tesseract significantly improves from 76% to above 98% at 3 times size rescaling using Lanczos interpolation.

REFERENCES

[1] http://www.babylon.com/.

[2] X. Wang, L. Huang, and C. Liu, "A video text location method based on background classification," *Int. Jour. on Document Analysis and Recognition*, vol. 13, no. 3, pp. 187–207, Sep. 2010.

[3] http://en.wikipedia.org/wiki/Phishing.

[4] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[5] S. Wachenfeld, H.-U. Klein, and X. Jiang, "Recognition of screen-rendered text," in *Proceedings of the 18th International Conference on Pattern Recognition*, vol. 02, 2006, pp. 1086–1089.

[6] T. M. Breuel, "Segmentation of handprinted letter strings using a dynamic programming algorithm," in *International Conference on Document Analysis and Recognition*, vol. 02, 2001.

[7] S. Wachenfeld, H.-U. Klein, and X. Jiang, "Annotated databases for the recognition of screen-rendered text," in *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, vol. 02, 2007, pp. 272–276.

[8] F. Einsele, R. Ingold, and J. Hennebert, "A HMM-based approach to recognize ultra low resolution anti-aliased words," in *Proceedings of the 2nd international conference on Pattern recognition and machine intelligence*, 2007, pp. 511–518.

[9] ——, "A language-independent, open-vocabulary system based on HMMs for recognition of ultra low resolution words," in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008, pp. 429–433.

[10] C. Jacobs, P. Y. Simard, P. Viola, and J. Rinker, "Text recognition of low-resolution document images," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, 2005, pp. 695–699.

[11] S. Yanadume, Y. Mekada, I. Ide, and H. Murase, "Recognition of very low-resolution characters from motion images captured by a portable digital camera," in *PCM (1)*, 2004, pp. 247–254.

[12] M. Krämer, "Optical character recognition using Hidden Markov Models," Master's thesis, Technical University of Kaiserslautern, 2007.

[13] F. Jelinek, *Statistical methods for speech recognition*. Cambridge, MA, USA: MIT Press, 1997.

[14] I. Bazzi, R. Schwartz, and J. Makhoul, "An omnifont open-vocabulary OCR system for English and Arabic," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, pp. 495–504, June 1999.

[15] U.-V. Marti and H. Bunke, *Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2002, pp. 65–90.

[16] L. R. Rabiner, "Readings in speech recognition," A. Waibel and K.-F. Lee, Eds., 1990, ch. A tutorial on Hidden Markov Models and selected applications in speech recognition, pp. 267–296.

[17] http://htk.eng.cam.ac.uk/.

[18] M. S. Khorsheed, "Offline recognition of omnifont Arabic text using the HMM ToolKit (HTK)," *Pattern Recogn. Lett.*, vol. 28, pp. 1563–1571, September 2007.

[19] http://www.gutenberg.org/.

[20] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[21] http://finereader.abbyy.com/.

[22] http://code.google.com/p/tesseract-ocr/.

[23] http://www.screenocr.com/.