

# Generation and Performance Evaluation of Synthetic Handwritten CAPTCHAs

Achint Oommen Thomas  
[aothomas@cedar.buffalo.edu](mailto:aothomas@cedar.buffalo.edu)

Venu Govindaraju  
[govind@cedar.buffalo.edu](mailto:govind@cedar.buffalo.edu)

Center of Excellence for Document Analysis and Research (CEDAR)  
Center for Unified Biometrics and Sensors (CUBS)  
University at Buffalo, The State University of New York  
Amherst, NY

## Abstract

*In this paper we explore the potential of handwriting for use in CAPTCHAs. A synthetic handwriting generation method is presented, where the generated textlines need to be as close as possible to human handwriting without being writer-specific. The primary application of such a synthetic generator is in the design of handwritten CAPTCHAs (Completely Automatic Public Turing Test to Tell Computers and Humans Apart). Such CAPTCHAs can exploit the differential in reading proficiency between humans and computers, while dealing with handwritten text images. This makes them viable for use in human verification by online services. Test results show that when the generated textlines are further obfuscated with a set of deformations, machine recognition rates decrease considerably, compared to prior work, while human recognition rates remain the same.*

**Keywords:** CAPTCHA, handwriting generation

## 1. Introduction

CAPTCHAs have recently come into the spotlight in cyber security applications. Specifically, these tests are being used for human verification services online. Spam control for blogs and automated account sign-up by bots are some of the applications that require to test if the entity accessing a service is a human or an automated machine. Most CAPTCHAs being used today are text-based. An image consisting of a series of printed text characters are rendered, distorted and obfuscated to varying degrees. This image is then presented to a user. If the user correctly guesses the characters present in the CAPTCHA he/she is granted access to some service. Circumventing the challenge posed by a CAPTCHA is an area that hackers are actively looking into. Several printed text based CAPTCHAs have already been broken as reported in [9].

Automated recognition of unconstrained handwriting continues to be a challenging research task. This fact can

be exploited to develop human verification systems - that utilize handwritten image challenges - for cyber security applications. To be suitable for online applications, we need to be able to automatically generate infinitely-many distinct artificially handwritten samples. A program must be able to generate a challenge as well as score the answer to it. Various models of human-like writing generation are available in the literature [4, 5, 6 and 7]. Most of the existing approaches are on-line based since it is more convenient to change the trajectory and shape of the letters based on the on-line information such as pen-down, pen-up, and velocity profiles. However, on-line information is not always available and as an alternative, researchers are applying various character and image level perturbations directly on real characters images or templates. We base our generation technique on pre-existing character images.

## 2. Generation Method

In this section, we describe a method for the generation of cursive English handwritten textline samples that uses pre-existing character images. This is part of the authors' prior work as published in [10]. The generation algorithm consists of several steps:

- i) character auto-scaling,
- ii) automatic baseline determination,
- iii) ligature endpoint detection,
- iv) ligature parameterization,
- v) ligature joining,
- vi) skeleton perturbation,
- vii) skeleton thickening.

A dataset of over 20,000 images<sup>1</sup> which contains multiple handwritten samples of each English character has been used. The characters have been segmented out

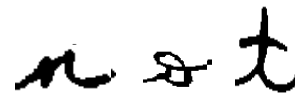


Figure 1: Sample Characters

manually from actual pieces of US mail. For a large fraction of the cases, the beginning and ending ligatures

---

<sup>1</sup> Dataset collection and character segmentation done by CEDAR, University at Buffalo, NY.

are also present with the character. Figure 1 shows some examples of the character images.

We first construct a preliminary image, which is a concatenation of individual character templates. Character templates are one-pixel wide representations (skeletons) of the original character image. We use Blum's Medial Axis Transform [1] to generate the character templates.

## 2.1. Character Auto-Scaling

To form the preliminary image, we concatenate individual character templates to form a textline. First we perform auto-scaling of the characters so that all characters maintain their correct relative sizes with respect to each other. Auto-scaling is based on the absolute size of the first character in the string. We maintain a lookup table of character heights known as the scaling factor for a character  $sf_c$ . The scaling factor gives the number of segments of a three segment vertical space occupied by a particular character. The height for character  $i$  is calculated as  $(sf_{fc} / sf_i) * h_{fc}$  where  $sf_{fc}$  is the scaling character for the first character,  $sf_i$  is the scaling factor for character  $i$  and  $h_{fc}$  is the height in pixels of the first character.

## 2.2. Automatic Baseline Determination

To string together individual characters to form a textline, we need to make sure that the characters are aligned vertically at their true baselines. We have exploited the fact, that in our dataset, the ligatures give us clues regarding the location of the true baseline. The procedure to determine the true baseline is as follows. Sum up the horizontal projections of  $n$  consecutive rows and store this value  $sHP_R$  for row  $R$  where

$$sHP_R = \sum_{r=R-n}^R HP_r$$

and  $HP_r$  is the horizontal projection value for row  $r$ . Normalize  $sHP_{Rows...n+1}$  so that  $sHP_{Rows...n+1} \in (0,1]$  and declare the true baseline as row

$$TB = \text{first}(sHP_{Rows...n+1} > \text{threshold}_B)$$

where  $TB$  is the total number of pixel rows in the character image,  $\text{threshold}_B$  is a cut-off value that is determined empirically from the dataset and the function  $\text{first}()$  returns the first value, in some ordered set  $X$ , that matches some given criteria.  $\text{first}()$  looks from the bottom-most horizontal projections to the top-most horizontal projections. Figure 2 shows a sample character image for 'g' and the corresponding horizontal projections;  $n$  was taken as 1 in this case and  $\text{threshold}_B$  was determined to be 0.75.

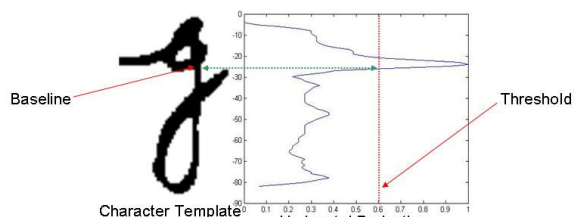


Figure 2: Automatic Baseline Determination for character image 'g'

## 2.3. Ligature Endpoint Detection

Ligature handling is the most important part of the procedure since they make a textline look more human-like. Following a procedure similar to the automatic baseline detection method, we compute a statistic known as the pseudo-inking profile from the image.

The pseudo-inking profile captures what could have been the pen activity as the writer generated the character image. We compute the sums of vertical projections of  $n$  consecutive columns and store this value

$$sVP_C \text{ for row } C \text{ where } sVP_C = \sum_{c=C-n}^C VP_c$$

and  $VP_c$  is the vertical projection value for row  $c$ . We now consider the first derivative of the pseudo-inking profile. We define the first derivative  $fdsVP_C$  as,

$$fdsVP_C(i) = sVP_C(i+1) - sVP_C(i) \text{ where } i \in [1 \dots C-1]$$

To detect the ligature endpoints we consider the left and right half images of the character separately to perform normalization of the first derivative plot.

Figure 3 shows how the first derivative plot is divided and normalized separately. We can use the first derivative plots to detect the ligature endpoints by defining a threshold value  $\text{threshold}_L$ . We now look in the normalized half plots of the first derivative for the first peak above some threshold. We compute

$$\text{ligature}_B = \text{first}_{1 \dots C} (fdsVP_{1 \dots C/2} > \text{threshold}_L)$$

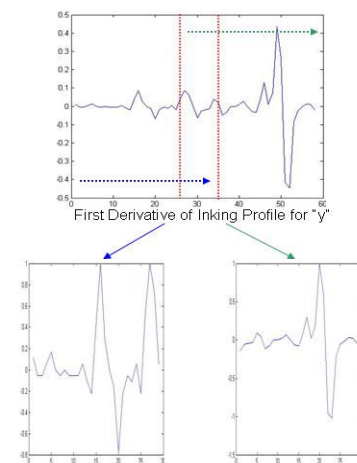


Figure 3: Split First Derivative plot allowing overlap and then normalize

and

$$ligature_E = first_{C \dots 1}(fdsVP_{C/2 \dots 1} > threshold_L)$$

These give us the columns where the beginning and ending ligatures join the character body. We empirically decide on a single global  $threshold_L$  based on the dataset.

Figure 4 shows the correctly detected ligature points for character images ‘i’, ‘o’, and ‘d’.



Figure 4: Correctly detected ligature points for ‘i’, ‘o’ and ‘d’

Our tests have proven that the ligature points have been detected correctly even for ‘d’ which has hardly any beginning ligature.

## 2.4. Ligature Parameterization

We determine the points at which the ligatures join the main character using regression to fit an  $n^{\text{th}}$  order polynomial to the ligature. For some characters, such as ‘f’ or ‘t’, more than one ligatures are possible. We use heuristic rules to decide on which ligature to use, either we can pick the largest/smallest ligature component or pick one component at random. We have used a random pick in our method.

To fit an  $n^{\text{th}}$  order polynomial  $p(x) = a_n x^n + a_{n-1} x^{n-1} \dots a_1 x + a_0$  to the extracted ligature, we use regression to perform polynomial approximation. To determine the order of the polynomial to represent the ligature, we first start with a polynomial of order 1, determine the polynomial and then compute a reconstruction error. The reconstruction error is defined as the number of mismatched points between the original ligature and the reconstructed ligature. We move to progressively higher order polynomials until the reconstruction error stabilizes and then choose the polynomial with the lowest reconstruction error.

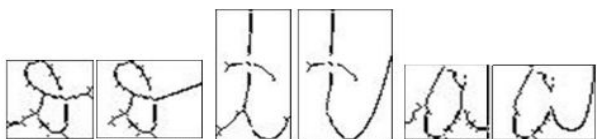


Figure 5: Parameterized Ending Ligatures: ‘o’, ‘t’, ‘a’  
Original image on left, processed image on right

Figure 5 shows the character images for ‘o’, ‘t’ and ‘a’ and the corresponding parameterized ending ligatures. The beginning ligatures can be parameterized in a similar manner.

## 2.5. Ligature Joining

We use a set of heuristic rules to join the ending ligature of character  $i$  to the starting ligature of character  $i+1$  similar to the method of curve smoothness optimization described in [8]. This method is suitable for our application as it only uses existing information from a basic character set and defines the inter-character joins at runtime using interpolation.

## 2.6. Skeleton Perturbation

We use the perturbation model proposed in [2] and [3] for the distortion of cursive handwritten textlines. The model incorporates a set of parameters over a range of possible values, from which a random value is picked before an existing textline is distorted. Each geometrical transformation is controlled by a continuous nonlinear function, (underlying function), which determines the strength of the transformation at each horizontal or vertical coordinate position of the textline. The underlying functions control geometrical transformations that affect a whole line of text. The transformations include shearing, horizontal and vertical scaling, and baseline bending. Refer [2] and [3] for details.

## 2.7. Skeleton Thickening

At this point, the textlines are still single pixel wide images. To thicken the textlines, we use the inverse

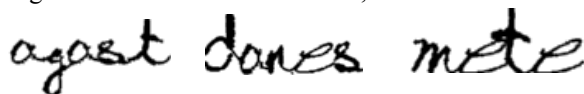


Figure 6: Synthetic handwritten word images

Medial Axis Transform. To add to the image complexity, we also apply an underlying function to vary the stroke width across the textline. Several examples of final images are shown in Figure 6.

## 3. CAPTCHA Generation

The primary application of our synthetic handwriting generator is automatic random generation of infinitely-many distinct handwritten (image) challenges for cyber security. Our ability to generate infinitely-many handwritten word images implies that we are not limited by a finite size database of CAPTCHA challenges. This makes the method suitable for online applications. Once a handwritten word image or textline has been generated, we add further distortions to obfuscate the image to a greater extent. Figure 7 shows some examples of the kind of distortions that we have used. Note that these are just an arbitrarily chosen set of distortions. It is certainly possible to think of other distortions.

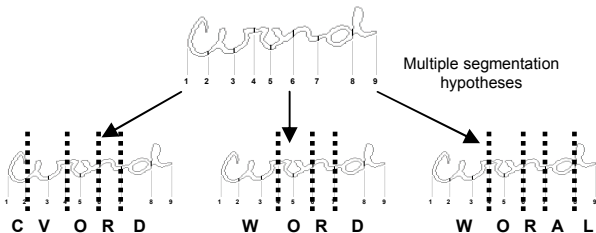


**Figure 7: Different types of distortions applied to generated handwritten words**

#### 4. Performance Evaluation

In this section, we look at the performance of the handwritten CAPTCHA as perceived by humans and handwriting recognizers (OCRs). Two recognizers were used, Word Model Recognizer (WMR) [12] and Accuscript [13]. WMR is a segmentation-based recognizer. It considers each word to be a model and finds the best match between an entry in a lexicon and the image. Accuscript is a grapheme-based recognizer. It extracts features from sub-characters (loops, turns, junctions, arcs, etc), without explicit segmentation. Both recognizers take advantage of using static lexicons in the recognition process, as well as using pre-processing techniques to enhance image quality and remove noise, thus making the performance evaluation for machines a fair test. The recognizers were run on a set of 2800 generated images distributed evenly among the various types of distortions.

One could argue that using lexicon based recognizers is not representative of the real world scenario for present day CAPTCHA implementations, because we reduce the problem to include only a limited number of challenges. However, for the tests with human subjects, we chose to only generate words that occur in the English language. The reason for this can best be described as seen in Figure 8. For handwritten word images, character formation ambiguity arising from different writing styles can lead to multiple segmentation hypotheses for a given word image. As seen in Figure 8, three segmentation hypotheses are possible for the handwritten word shown at the top of the figure.



**Figure 8: Multiple segmentation hypotheses for a handwritten word image**

Humans (and recognizers relying on lexicons), decide on the correct segmentation hypothesis based on either context information, or knowledge of the lexicon. Human recognition accuracy would decrease if the handwritten CAPTCHA challenges were allowed to be lexicon independent. This defeats the purpose of using handwriting for CAPTCHAs. So, for the tests, we limit ourselves to a lexicon that comprises all the words in the English language. This makes it feasible for humans to use context information (that the word is a legitimate word in the English language), to guess the handwritten word.

Table 1 shows the recognition accuracy of two types of OCRs on CAPTCHA challenges with different types of distortions.

**Table 1: Recognition accuracy of OCRs for different CAPTCHA distortions**

HW Recognizer (OCR)	WMR	Accuscript
<i>All Distortions</i>	<b>1.42 %</b>	<b>2.58 %</b>
No Distortion	23.69 %	37.78 %
Perturbed Image	0.18 %	2.63 %
Edges	0.18 %	0 %
Fragmentation	1.43 %	3.45 %
Displacement	0 %	3.61 %
Mosaic	0 %	3.78 %
Jaws / Arcs	5.91 %	5.83 %
Occlusion by circles	0.36 %	5.75 %
Occlusion by waves	0 %	2.30 %
Exploded Image	0 %	0 %
Vertical Overlap	1.35 %	1.32 %
Horizontal Overlap	4.91 %	1.16 %
Sideways Overlap	2.69 %	1.16 %

Even with no distortions applied on the generated handwritten word images, the recognizers are not able to cross the 40% mark. When distortions are applied, the recognition rate drops much below 10%. The average recognition rate is comparable for the two recognizers with Accuscript doing marginally better.

**Table 2: Comparison with prior work - Machine accuracy**

HW Recognizer	WMR	Accuscript
All Distortions (IWFHR 2006)	12.7%	6.4%
All Distortions (Current)	1.42%	<b>2.58%</b>

The current technique improves on prior work published in 2006 [11]. As shown in Table 2, for all distortions, the highest machine accuracy was about 13% as opposed to the current, much lower, 2.6%. Note that the lower the score, the better, when considering machine accuracy.

Table 3 shows the corresponding human performance on a subset of the images used for the OCR tests. For human testing, random CAPTCHA samples were presented to users through a website. 2800 responses were collected from around 100 users. With no distortions, the average recognition rate for humans is about 84%. The recognition rate drops for other distortions, by varying extents. These results along with results in Table 1 are helpful in determining which types of distortions would be more suitable for use in the CAPTCHA application. For all distortions, the average human recognition rate is about 76%, which is the same as in the prior work of 2006 [11].

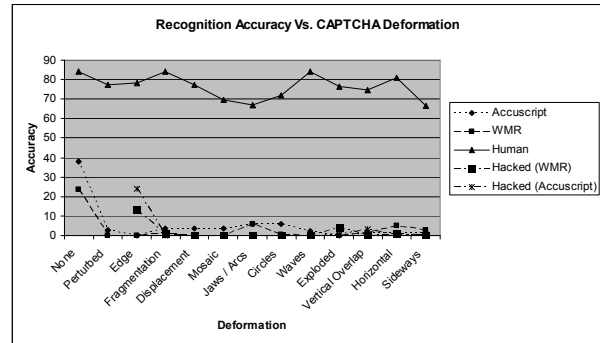
**Table 3: Recognition accuracy of humans for different CAPTCHA distortions**

Human Performance	Accuracy
<i>All Distortions</i>	<b>76.29 %</b>
No Distortions	83.97 %
Perturbed Image	77.24 %
Edged Image	78.08 %
Fragmentation	84.17 %
Displacement	77.55 %
Mosaic	69.53 %
Jaws / Arcs	69.96 %
Occlusion by circles	71.71 %
Occlusion by waves	84.25 %
Exploded Image	76.62 %
Vertical Overlap	74.45 %
Horizontal Overlap	80.77 %
Sideways Overlap	66.67 %

Figure 9 summarizes the results and shows a clear gap between human and machine recognition abilities for the handwritten CAPTCHA. This shows that handwriting has potential for use in CAPTCHA applications.

For suitable use in a cyber security scenario, we must ensure that the human recognition rate stays high when compared to machine recognition rates (which should ideally be zero). Even a seemingly low recognition rate for machines (say, less than 0.01%), would not necessarily mean that a given cyber security application can be considered bot-proof. We must bear in mind that

a recognition rate of  $x\%$  means that, statistically,  $x$  out of every 100 attempts will be successful. Since it is possible to have distributed attack networks, repeatedly trying to



**Figure 9: Gap in recognition abilities between humans and machines for the handwritten CAPTCHA**

gain access to a secured application, the sheer volume of requests would render the low recognition rate itself, irrelevant. On the other hand, while it would be ideal to have 100% recognition accuracy for human users, it would be safe to assume that human users would tolerate some lack of CAPTCHA ease. For instance, a human user might not be overly concerned with having to re-try a CAPTCHA once in every 6 – 8 attempts. This fact can be exploited while designing CAPTCHAs. Some human recognition performance can be deliberately sacrificed, if it degrades machine performance by a considerable amount.

## 5. Conclusion and Future Work

We have explored the recognition performance of humans as compared to machines, on handwritten CAPTCHAs. We have shown how to generate synthetic handwriting samples and then apply various distortions to make them near-unreadable by automatic computer programs, so that they can be used as CAPTCHA challenges over the Internet. Test results show a large gap between human and machine recognition abilities. There is also a significant decrease in machine recognition rates for our synthetic samples as compared to prior work. However, human recognition rates remain the same. This directly translates as the method being a better CAPTCHA generation technique.

We plan to improve the method by automatically learning the threshold values  $threshold_B$  and  $threshold_L$  from a given dataset of character images. We plan to research on deformation techniques that exploit the knowledge of the common source of errors in automated handwriting recognition systems and also take advantage of the cognitive aspects of human reading. We plan to invite programmers to try to reverse the distortions applied to the generated word images. These pre-

processed images will then be fed to the recognizers for machine recognition. It would be interesting to see if machine performance improves considerably. We plan to conduct more detailed tests that will involve using non-sensical words that are generated by stringing together random characters and also words formed by concatenating two or more legitimate syllables. If the human recognition performance is comparable to the current results, it would mean that we can further reduce the machine recognition performance since the lexicon size will increase considerably.

Another application of this generator is to improve the accuracy of handwriting recognizers by generating large synthetic training data sets. Since our technique does not generate writer-specific handwritten textline samples we could use it for training generic handwriting recognizers.

## References

- [1] H. Blum, "A Transformation for Extracting New Descriptors of Shape", Models for the perception of Speech and Visual Form, W. Wathen-Dunn, Ed. Cambridge, MA: MIT Press, 1967, pp. 362-380.
- [2] Varga, T. & Bunke, H., "Generation of Synthetic Training Data for an HMM-based Handwriting Recognition System", In Proceedings of the 7th ICDAR 2003 (pp. 618-622), Edinburgh, Scotland.
- [3] Varga, T. & Bunke, H., "Effects of Training Set Expansion in Handwriting Recognition using Synthetic Data", 2003.
- [4] Zhouchen Lin and Liang Wan, "Style Preserving English Handwriting Synthesis", Microsoft Research Asia, 2005.
- [5] Jue Wang, Chenyu Wu, Ying-Qing Xu and Heung-Yeung Shum, "Combining Shape and Physical Models for Online Cursive Handwriting Synthesis", Intl. Journal on Document Analysis and Recognition, 2004.
- [6] Wacef Guerfali & Rkjean Plamondon, "The Delta Log Normal Theory for the Generation and Modeling of Cursive Characters"; IEEE September 1995
- [7] Hala Bezzine, Adel M. Alimi, and Nabil Derbel, "Handwriting Trajectory Movements Controlled by a Beta-Elliptic Model"; Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003).
- [8] Michael Kokula, "Automatic Generation of Script Font Ligatures Based on Curve Smoothness Optimization"; Electronic Publishing 7(4): 217-229 (1994)
- [9] K. Chellapilla, K. Larson, P.Y. Simard and M. Czerwinski, "Building Segmentation Based Human-Friendly HIPs"; Proceedings of the Second International Workshop, HIP 2005.
- [10] Achint Oommen Thomas, Amalia Rusu, Smruthi Mukund and Venu Govindaraju, "Non Writer Specific Synthetic Handwriting Generation for the CAPTCHA Application"; 2007 IEEE Western New York Image Processing Workshop.
- [11] A. Rusu and V. Govindaraju, "The Influence of Image Complexity on Handwriting Recognition"; Proceedings of IWFHR 2006.
- [12] J. T. Favata, "Character model word recognition", Proceedings of the Fifth International Workshop on Frontiers in Handwriting Recognition, 1996, pp 437-440.
- [13] H. Xue, and V. Govindaraju, "A stochastic model combining discrete symbols and continuous attributes and its applications to handwriting recognition", Proceeding of the International Workshop on Document Analysis and Systems, 2002, pp 70-81.