

4005-898-01  
Independent Study Report  
Character Segmentation and Classification

Kurt Alfred Kluever (kurt@klover.com)  
Department of Computer Science  
Golisano College of Computing and Information Sciences  
Rochester Institute of Technology

February 28, 2008

**Abstract**

Optical Character Recognition (OCR) is the process of translating images of hand-written, typewritten, or printed text into a format understood by machines for the purpose of editing, indexing/searching, and compression. The process can be broken down into 3 tightly coupled tasks: pre-processing, segmentation, and classification.

Human Interactive Proofs (HIPs) are methods used to differentiate between humans and machines on the internet, and are typically implemented as distorted text which the user must correctly transcribe. HIPs should be easy for a machine to automatically generate, easy for a human to solve, and difficult, or impossible, for a machine to solve even if the generation algorithm is publicly available. Ideally, HIPs are a user-friendly method of exploiting the gap between human and machine intelligence.

The ASP Security Image Generator v2.0 <sup>1</sup> generates an 8-digit, color image where the digits have been vertically shifted, warped and peppered with noise. We present a 3 step process for breaking this HIP. In the pre-processing step, binarization and thresholding techniques remove the majority of the noise from the input image. Segmentation is performed by using candidate split positions and vertical projections to determine segmentation locations. Digit classification is performed by using correlation values of inputs and templates as a distance metric. Using the *averaged template* set, an overall recognition rate of 72% has been achieved on a set of 100 testing images.

---

<sup>1</sup><http://www.tipstricks.org/example.asp>

# 1 Optical Character Recognition

Optical Character Recognition is the process of translating images of handwritten, type-written, or printed text into a format understood by machines for the purpose of editing, indexing/searching, and compression. There is a great need to accurately OCR printed materials:

Much of the worlds information is held captive in hard-copy documents. OCR systems liberate this information by converting the text on paper into electronic form. [17]

OCR can be broken down into the following steps: pre-processing of inputs, segmentation of characters, and classification of characters.

Pre-processing step can occasionally be omitted if the input images are guaranteed to be clean of any noise. However, OCR is typically performed on scanned documents that contain noise from old paper, poor printers, or bad scanners. Generally, the pre-processing stage consists of noise removal, skew correction, and thresholding. Noise removal can be achieved through filtering the image to remove extraneous or stray marks. Skew correction can be performed by estimating the angle of lines of text [26]. Thresholding is often used as a method of binarizing an image. It works well when the salient information has either very low or very high intensity values. These techniques attempt to provide clean (or as clean as possible) input to the segmentation step.

## 1.1 Segmentation

Character segmentation is the decomposition of an image into subimages which only contain a single character. Segmentation is dependent on local decisions with regards to shape similarity, as well as global decisions with regards to surrounding context. It is a critical step in most OCR systems, and typically the cause of a high proportion of OCR errors. In 1996, Richard Casey and Eric Lecolinet surveyed the available methods and defined three categories for offline character segmentation methods, based on how segmentation and classification interact in the overall process [5]:

- Dissection Approach: a single partitioning of the image into subimages based on “character-like” properties followed by classification of the subimages
- Recognition-based Approach: segmentation where the image is iteratively searched for components that most closely match the classes in the alphabet
- Holistic Approach: segment and recognize words as single units (character segmentation is avoided!)

The task of segmenting characters varies in difficulty based on the input type. Fixed pitch machine printed text can typically be segmented fairly easily using simple projection analysis. However, cursive handwritten text must be segmented using more advanced methods such

as Hidden Markov Models (HMMs), Artificial Neural Nets (ANNs), and contextual methods (i.e., what character makes most sense in the surrounding context). In order to achieve high accuracy on complex problem domains, segmentation and recognition cannot be treated independently. However, a simple problem domain can use more basic techniques and still achieve high accuracy.

### 1.1.1 Dissection Approach

Dissection is the decomposition of the input image into a sequence of subimages. The criterion for good segmentation using the dissection approach is the agreement of character properties in the segmented subimage and the expected symbol. The character properties include height, width, separation from neighboring components, disposition along the baseline, etc. Interaction with the classifier is limited to reprocessing of ambiguous recognition results. For example, if the classifier can't make any decision at all, the segment may need to be split again into 2 new segments.

In the late 1950s and early 1960s, in order to get good segmentation results, strict constraints were placed on the input documents. The writer or printer was constrained to place characters into printed boxes to ease segmentation. Fonts of printed materials were chosen with strong leading edges for easier detection. Due to the constrained inputs and the research focus on classification, segmentation was not well developed before the 1970s.

The earliest and simplest form of dissection relies on vertical whitespace between successive characters. Pitch is the number of characters per unit of horizontal distance. To make segmentation even easier, a fixed pitch was used. In applications that used machines to print the text, the text was printed with a fixed pitch using limited font sets. In applications with handwritten forms, separate boxes were provided for each character. Hoffman and McCullough [12] designed a system that could aid in segmentation when a fixed pitch could not be enforced. The system consisted of three steps: 1) detection of the start of a character based on an a priori pitch measurement, 2) a decision to begin testing for the end of the character, and 3) detection of the end of a character. The authors reported a 97% accuracy, but the results were heavily dependent on the quality of the input image. The input data consisted of machine-printed lines of 10-, 11- and 12-pitch serif-type multifont characters.

Projection analysis is a segmentation method that uses the vertical projection (or vertical histogram) of black pixels to determine dissection candidates. A vertical projection is simply a running count of black pixels in each column. If the count falls below a predefined threshold, the column is a candidate for splitting the image. Peaks of the derivative of this data indicate a potential split. AND'ing adjacent columns of the derivative of the vertical projection leads to a better defined peak at the character boundaries. The AND'ing can also be implemented as a prefilter on the input image to result in a similar, desirable effect. Projection analysis, a one-dimensional analysis, works well on good quality machine printed documents. However, vertical projection performs poorly on text which has not been properly skew-corrected, italicized machine printed text, or hand written text (naturally slanted).

In graph theory, a connected component is a maximal connected subgraph where two vertices belong to the same connected component if and only if there is a path between

them. Connected component analysis can be applied to character segmentation by viewing the characters as block or line adjacency graphs. Intuitively, most characters are connected components because all pixels “touch” each other (with the exceptions of i’s and j’s). Connected component analysis is a two-dimensional analysis that works well on proportional fonts and handwritten characters. Connected component analysis can also be used to segment blocks of characters into individual words. One experiment showed that connected component analysis is 4 times faster and yielded half as many errors as projection analysis [25]. The method works by labeling connected areas of black pixels as components. The components are further processed by drawing bounding boxes around them or based on a detailed analysis of the image. Predefined rules are specified to determine the maximum or minimum size of the bounding boxes. Others have proposed using a recognizer that does not perform character classification, but instead determines if the connected component is a single character or not. Dynamic programming methods have been proposed which use a cost function over the image to locate a minimal cost path through the image to split the connected components [24], while avoiding excessive curvature in the cut line [11].

### 1.1.2 Recognition-based Approach

Recognition based segmentation in effect bypasses the requirement to discretely segment the word. No complex dissection algorithm is designed or implemented. This method directly interacts with the classifier. Without regard to content, a mobile variable width window blindly divides the image into many overlapping pieces and chooses the correct segmentation based on the recognition confidence. Therefore, the criterion for good segmentation is the recognition confidence given by the recognizer of the subimage.

Recognition based segmentation systems generally work as follows. First, the windowing is performed on the image to generate segmentation hypotheses. After this, the best hypothesis (as determined by the classifier) is chosen during a verification step.

Early methods used windowing techniques that classified the character based on a prototype character (templating). The system would exhaustively search all possible cut points in the image until all characters had been matched against a prototype library within a given threshold.

Another approach was proposed that used a shortest path to find the most accurate segmentation of the word. It calculated all the possible segmentation boundaries and constructed a graph, where all paths through the graph resulted in potential matches. The left to right path with the lowest weight corresponded to the best recognition rates (and therefore the best segmentation rates as well).

Furthermore, N-gram statistics can be introduced to recognition based approaches to prune the search space [13]. Assuming we have already recognized the letters ‘t’ and ‘h’, there is a higher probability that the next letter is an ‘e’ instead of a ‘c’. Methods have also incorporated a word dictionary into the recognition process. After narrowing the possible word guesses, a dictionary can be used to eliminate incorrectly spelled words in favor of correctly spelled words.

### 1.1.3 Holistic Approach

Holistic approaches attempt to recognize the entire words as a whole. The criterion for good segmentation are same as the criterion for good dissection, but using words as the alphabet instead of individual symbols or characters. Unlike the previously mentioned methods, a holistic approach is limited in application to a predefined lexicon. For some applications, such as check recognition or postal code reading, this constraint might be OK, and in some cases, preferable.

Holistic approaches typically follow the same scheme: First perform feature extraction to create a representation of the word (ascenders, descenders, directional strokes, cusps, diacritical marks, etc.). Next, perform global recognition by matching the representation of the word with a representation of a predefined word in the lexicon. The representation can be a feature vector, or even a template of the word (known as a “holograph”). Generating all of the representations of the words in the lexicon can be expensive, so some systems dynamically generate the holistic descriptions when needed.

## 1.2 Classification

Character classification is typically performed based on feature vectors. Feature extraction is the process of transforming the input data into a reduced representation. It is commonly employed when there is too much input data to efficiently process or if the input data is redundant (lots of data but not much information). This simplification of the input data provides an accurate description of a larger set of data. The set of application-dependent features are typically chosen by domain experts. However if no such experts exist, other dimensionality reductions, such as principle component analysis (PCA), can still be performed. PCA is used to reduce the dimensionality of multi-dimensional data sets. It works by removing characteristics about the data which have low impact on the variance of the overall dataset. Similarly, it attempts to retain characteristics which contribute most to its variance.

Once the feature vectors are computed, classification can be performed. Classification can be done by nearest neighbor techniques based on templates [21], neural networks [15] [1] [19], etc. The classification mechanism is largely non-important in the resulting decision. The most important step is the feature extraction, of which many methods exist. Naive methods feed entire image matrices, while others require experts to develop visual cues to distinguish characters from one another. Depending on the problem domain, classifiers can be aided by the inclusion of a dictionary or N-gram statistics.

## 2 Human Interactive Proofs

Human Interactive Proofs (HIPs) are a method used to differentiate between humans and machines on the internet, and are typically implemented as distorted text which the user must correctly transcribe. HIPs should be easy for a machine to automatically generate, easy for a human to solve, and difficult, or impossible, for a machine to solve. Current

implementations include recognizing a string of distorted characters [9], choosing the correct annotation for an image [8] [10], or transcribing an audio clip with noise [14]. HIPs should be an effective and user friendly method of exploiting the gap between human intelligence and machine learning.

The task of differentiating between a user and machine over the internet has significant importance in the fields of internet security, artificial intelligence, and machine learning. Currently, HIPs prevent robots from signing up for free online services (such as email accounts), abusing online polls, providing biased feedback, and spamming innocent users.

In 1950, Alan Turing provided an operational definition of intelligence using the imitation game [22]. The proposed Turing test is administered by a human judge and taken by a machine and a human contestant. If the machine can trick the human judge into believing that it is the human contestant, the machine is said to have passed the Turing test. In contrast, HIPs are administered by a machine but taken by a human. The burden is on the human to convince the machine that he/she is in fact human. In this way, HIPs are analogous to *reverse* Turing tests.

In 1996 unpublished draft, Moni Naor theorized nine possible sources for HIPs [18]. They included text based, image based, and speech based challenges. These three media formats (text, image, audio) have served as popular choices for HIP implementations. However, prior research has shown that requiring a user to recognizing characters in a string of distorted text is not the only reliable method. Image-based HIPs have proved to be both secure and easy to use. They include tasks such as identifying a common object in a set of images, determining if a set of images all contain the same object, and identifying the “odd one out” in a set of images [8]. Microsoft has also developed a fun and effective image-based HIP where the user must correctly label a set of images as cats or dogs [10].

There has been extensive research done in the area of HIPs at the Palo Alto Research Center [9] [7], Bell Laboratories [14], Microsoft Research [10] [6], Carnegie Mellon University [23], Lehigh University [9] [7] [3], University of California at Berkeley [8], and the University of Buffalo, CEDAR [20] . There have been two international workshops on HIPs [4] [2]. Independent user studies have been conducted to determine the “user friendliness” of the different HIPs. Most text based HIPs [23] [7] are no longer secure against attacks such as recognition via neural networks [6] and shape matching [16].

### 3 Experiment: Break the ASP Image-Based HIP

Due to the significant amount of “pepper” noise present in the ASP input images, pre-processing is required to ensure successful digit recognition. Here is an example of an image generated by the HIP script:



The image is a color bitmap image of size  $86 \times 21$  pixels (width  $\times$  height) with a white background and red foreground. To make noise removal more difficult, the pepper noise is

the same color (red) as the 8 digits. The digits have also been slightly warped and then shifted in the vertical direction.

In order to smooth the image, a two-dimensional averaging filter of size  $3 \times 3$  is generated:

$$h = \begin{bmatrix} 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \end{bmatrix}$$

The two-dimensional correlation of the input image and  $h$  is computed. Correlation is performed by computing the convolution of the input image and the filter after a 180 degree rotation. The output dimensions of the correlation are matched to the dimensions of the input image by retaining only the central part of the image. Here is an example of the image after applying the averaging filter:

$$i_{AVG} = \text{95869404}$$

Median filtering is a nonlinear operation that is often used to reduce “salt and pepper” noise, such as that present in the current image. Each output pixel contains the median value in the  $1 \times 1$  neighborhood around the corresponding input pixel in  $i_2$ . If an image has been peppered with 1 pixel dots, the median filter will remove most of these. Intuitively, if a  $3 \times 3$  block of pixels has only 1 peppered dot, the median of the  $3 \times 3$  block will be the background (which will remove the pepping). The result of the median filtering the image  $i_{AVG}$ :

$$i_{MED} = \text{95869404}$$

Since the characters are the only dark parts of  $i_{MED}$ , a simple threshold will remove all of the background noise. Through empirical testing, it was found that a threshold value of 0.6 preserved the digits while removing the background noise. Here is an example after thresholding the image  $i_{MED}$ :

$$i_{THR} = \text{95869404}$$

Correlation often introduces edge artifacts. As seen above, the averaging filter introduces some errors around the border of the image which must be corrected. The digits are never touching the border, so the edges can safely restored using a simple region-filling algorithm. The results of this step are shown below:

$$i_{RF} = \text{95869404}$$

The final step in pre-processing is to place a bounding box around the digits. This has the effect of eliminating any horizontal shift of the digits. An example is shown below:

$$i_{BB} = \text{95869404}$$

## 4 Segmentation

Segmentation was performed via vertical projections on 4 candidate split positions. The fact that the HIP always displays an 8-digit string makes the segmentation task easier. Candidate split positions are located at 8, 9, 10, and 11 pixels to the right of the previous split. These values were found through empirical testing and observation of the input strings. The vertical projections of all four positions are computed. The column with the least number of “on” pixels is selected as the right segmenting location. The results of segmentation for  $i_{BB}$  are as follows:

$$S_1 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline \text{9} & \text{5} & \text{8} & \text{6} & \text{9} & \text{4} & \text{0} & \text{4} \\ \hline \end{array}$$

As explained earlier, the digits are also randomly shifted vertically within the original canvas. In order to achieve better correlation with the templates, the digits are centered by placing a bounding box around them:

$$S_2 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline \text{9} & \text{5} & \text{8} & \text{6} & \text{9} & \text{4} & \text{0} & \text{4} \\ \hline \end{array}$$

In order to perform a correlation with the template images, the segmented images must be of the same size as the templates. Therefore, the segments are padded out to the same size as the templates ( $20 \times 15$  pixels).

$$S_3 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline \text{9} & \text{5} & \text{8} & \text{6} & \text{9} & \text{4} & \text{0} & \text{4} \\ \hline \end{array}$$

The segmentation method works correctly 100% of the time on the 100 sample images used for testing. Any errors of the system are due to classification mistakes.

## 5 Classification

The classification task is simplified by the small number of output classes,  $O = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ . A random guess algorithm can serve as the lower bound for classification accuracy. A random guess would yield an estimated per-digit accuracy of 10%. Given that there are always 8 digits, the same algorithm would have an estimated per-string accuracy of  $0.10^8$ . The three template-matching methods presented below perform substantially better.

The digit recognition is performed via a nearest neighbor method. The distance metric used is the correlation value of the segmented digit and a set of training template digits. Correlation indicates the strength of a linear relationship between two random variables. The template with the highest correlation coefficient is returned as the recognized digit.

The correlation coefficient between two images  $A$  and  $B$  is calculated using the following equation, where  $\bar{A}$  represents the mean intensity value of  $A$ :

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \bar{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \bar{B})^2\right)}}$$

The denominator is the square root of the product of the variances of the template and the segmented digit. Note that if either input image contains no variance in the pixels (i.e., all 0's or all 1's), the summation in the denominator will evaluate to 0 and cause the correlation to fail (i.e., NaN). However, neither the templates nor the segmented digits are variance-free and this error will never occur.

The data used to train and evaluate the system was automatically generated. The source code for the ASP Security Image Generator v2.0 is freely available for download <sup>2</sup>. A simple modification to the source code allowed for specifying what digit string to build the HIP from. Using a simple Java program, 100 random images were generated and filed as training images. An independent set of 100 random images were filed as testing images. Without having access to the source code, the training and testing images would have to be manually labeled.

## 5.1 Templates

In this section, the template creation method for each of the template sets is described in detail and a description of the results is provided. In the appendix, the following is provided: an enlarged view of the 3 sets of templates (Appendix A), the correlation matrix of the templates  $\times$  templates (Appendices A1, B1, C1), the confusion matrix (Appendices A2, B2, C2), and the correlation matrix of the templates  $\times$  an example testing image (Appendices A3, B3, C3).

*Noisy templates* are templates which were extracted directly from a single training image. Since a given training image contains exactly 8 digits, two training images containing all 10 output classes were used for creation of the templates. The training images were pre-processed and segmented using the same techniques as mentioned above.

These templates are called *noisy templates* because they are warped and include artifacts from the distortions in the original image. This technique is rather naïve because it assumes that the distortions of the testing images will be similar to the distortion of the training images which were used to generate the templates.

Not surprisingly, the noisy template method yields rather poor performance when compared with the other template generation schemes. However, it is worth noting that even with the naïve template choice, this method still achieves a per-digit accuracy of 0.7188 (71.88%), as compared to the per-digit accuracy of 0.10 (10%) of a random guess method. Given that each HIP has 8 digits, the estimated per-string accuracy is  $0.7188^8 \approx 0.0712$ . When testing with 100 images, the overall per-string accuracy was 0.04 (4%).

---

<sup>2</sup><http://www.tipstricks.org/examplev20.zip>

*Clean templates* refer to a set of digit templates that were created from training images that contained no noise. If the generation algorithm was not publicly available, this task would have been more difficult. A similar font would have had to be chosen to create these templates. Two noise-less training images were generated which contain all 10 digits:

**01234567**  
**98765432**

The images were then segmented using the same process detailed above.

As expected, the clean templates performed better than the noisy templates, yielding a per-digit accuracy of 0.8687 (86.87%). Given that each HIP has 8 digits, the estimated per-string accuracy is  $0.8687^8 \approx 0.3245$ . When testing with 100 images, the overall per-string accuracy was 0.30 (30.0%).

The *averaged templates* were computed from a collection of 100 training images. The images were pre-processed into binary images and then segmented into digits using the methods described above. Using 100 training images with 8 digits per image, a total of  $100 \times 8 = 800$  digits were used for computing the templates. In the template image, pixels that are “on” indicate that more than half of the training images have “on” pixels at the same location. The procedure, `COMPUTE-AVERAGED-TEMPLATE( $I, n, m$ )`, is located in appendix D.4.

The averaged templates performed extremely well, with a per-digit accuracy of 0.9563 (95.63%). Given that each HIP has 8 digits, the estimated per-string accuracy is  $0.9563^8 \approx 0.6992$ . When testing with 100 images, the overall per-string accuracy was 0.72 (72.0%).

## 6 Discussion

The experiment described above indicates that this HIP is vulnerable against a relatively simple attack that can be broken down into 3 steps: pre-processing, segmentation, and classification.

It is hard to develop a metric for evaluating the pre-processing step. However, upon manual inspection, most of the noise was removed from the images. Slight artifacts are occasionally introduced into the images. This is because when a dot of pepper noise is touching a digit, it will not remove it. The pre-processing step may produce better results if a smoothing operation was performed on the digit strings to remove any jaggedness of the thresholded digits.

The segmentation step is also hard to evaluate. However, all of the test images were successfully segmented into single digits. Not once were two digits joined together or was part of a neighboring digit included in the wrong segment.

Therefore, the only step which can be easily evaluated with a known metric (correct vs. incorrect classification) is the classification step. The most concise reference are the confusion matrices located in Appendices B2, C2, and D2.

The noisy templates produced the worst results. The correlation of the noisy templates  $\times$  the noisy templates helps explain the poor results. High correlations between the following tuples exist: (0, 5), (0, 6), (0, 8), (1, 3), (2, 7), (3, 5), (3, 6), (4, 8), (5, 8), (6, 8), (8, 9). In the confusion matrix, the following tuples exhibited large misclassifications: (0, 8), (0, 9), (2, 7), (6, 8), (1, 4). The overlap between these two sets show that templates which were not distinct enough from other templates yielded misclassifications. In the example correlation with the noisy templates, the artifacts from preprocessing of the templates creates misclassifications. Out of the 8 digits, 4 were misclassified. However, the difference between the correlation value of the correct digit and the chosen digit were typically very small (i.e. 0.588 vs. 0.599, 0.717 vs. 0.751, 0.686 vs. 0.687). The correct choice was typically the next best correlation, if not the first.

The clean templates yielded much fewer classification errors. In the example correlation with the clean templates, only 1 out of the 8 digits was misclassified. The '5' exhibits some strange artifacts from the noise removal in the preprocessing stage which makes it look very similar to a '6' or a '9'. The third best correlation was actually the correct one in this case, but it was still close to the top choices (0.410 vs. 0.431).

The most effective set of templates were the averaged templates. In the example correlation with the clean templates, all of the 8 digits were successfully classified. The difference in correlation values between the second best digit and the chosen digit was typically very large, indicating a successful set of templates. The errors in the confusion matrix properly align with high correlation values in the averaged templates  $\times$  averaged templates matrix.

The averaged templates yield a per-string accuracy of  $\approx 72.0\%$ . According to the current research community, an attack that yields  $> 70\%$  accuracy which indicates that this HIP is no longer an effective means of exploiting the gap between human intelligence and machine learning. The proposed attack method is therefore a success and the ASP Security Image Generator v2.0 has been broken. Using a larger set of training data, the averaged templates could be improved even further.

A more complex version<sup>3</sup> has been released that contains both characters and digits. This substantially increases the size of the template library. Instead of using peppering noise like v2.0, the newer version uses random thin lines through the image. It is suggested that filtering would be able to remove most of these lines and a similar attack would be successful as well.

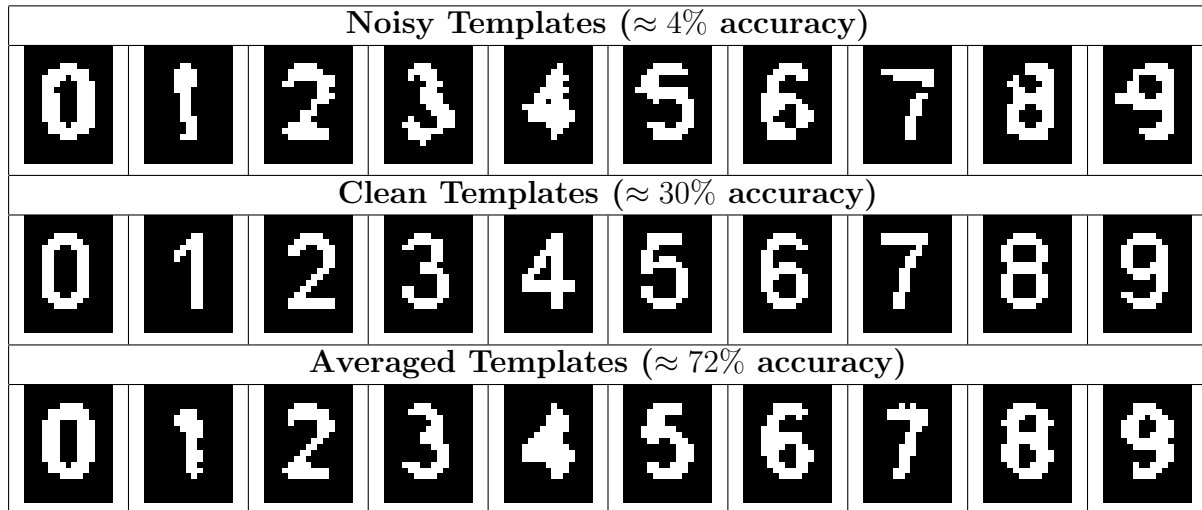
---

<sup>3</sup><http://www.tipstricks.org/aspsig/examplev3.asp>

# Appendices

## A Comparison of Templates

The three sets of templates used in the experiment. They were pre-processed (filtering and thresholding) and segmented using the same techniques as discussed in section 3 and 4 above.



## B Noisy Templates

### B.1 Correlation of Noisy Templates

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
<b>0</b>	1	0.174	0.525	0.423	0.395	0.646	0.695	0.376	0.662	0.593
<b>1</b>		1	0.380	0.418	0.386	0.281	0.318	0.394	0.225	0.237
<b>2</b>			1	0.429	0.291	0.473	0.349	0.577	0.448	0.385
<b>3</b>				1	0.481	0.510	0.531	0.320	0.402	0.315
<b>4</b>					1	0.311	0.419	0.341	0.476	0.350
<b>5</b>						1	0.566	0.211	0.636	0.559
<b>6</b>							1	0.276	0.631	0.556
<b>7</b>								1	0.309	0.399
<b>8</b>									1	0.740
<b>9</b>										1

### B.2 Recognition Confusion Matrix using Noisy Templates

		Prediction Outputs										
		0	1	2	3	4	5	6	7	8	9	Sum
Actual Outputs	0	80								9	2	91
	1		60		1	16			7			84
	2			87								87
	3	1			79		5					85
	4					74						74
	5				2		70	10		2	1	85
	6	1						67		5	2	75
	7		16	34	4	1			19			74
	8	34				1	1	20		14		70
	9	33		2	2	1	7	5			25	75
Sum		149	76	123	88	93	83	102	26	30	30	

### B.3 Example Correlation with Noisy Templates

	<b>9</b>	<b>5</b>	<b>8</b>	<b>6</b>	<b>9</b>	<b>4</b>	<b>0</b>	<b>4</b>
<b>0</b>	<b>0.615</b>	0.454	<b>0.751</b>	0.735	<b>0.687</b>	0.402	<b>0.836</b>	0.435
<b>1</b>	0.379	0.384	0.246	0.324	0.313	0.450	0.204	0.229
<b>2</b>	0.536	0.304	0.584	0.387	0.529	0.209	0.577	0.304
<b>3</b>	0.553	0.488	0.562	0.505	0.504	0.475	0.449	0.403
<b>4</b>	0.462	0.396	0.494	0.517	0.413	<b>0.803</b>	0.318	<b>0.821</b>
<b>5</b>	0.525	0.588	0.684	0.656	0.659	0.248	0.584	0.342
<b>6</b>	0.522	0.546	0.639	<b>0.826</b>	0.654	0.491	0.639	0.384
<b>7</b>	0.490	0.316	0.376	0.309	0.399	0.306	0.364	0.316
<b>8</b>	0.417	0.597	0.717	0.658	0.643	0.366	0.532	0.498
<b>9</b>	0.495	<b>0.599</b>	0.575	0.604	0.686	0.325	0.495	0.378

## C Clean Templates

### C.1 Correlation of Clean Templates

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
<b>0</b>	1	0.313	0.447	0.666	0.365	0.742	0.799	0.243	0.833	0.799
<b>1</b>		1	0.309	0.422	0.698	0.288	0.246	0.199	0.313	0.356
<b>2</b>			1	0.331	0.197	0.372	0.376	0.376	0.358	0.422
<b>3</b>				1	0.392	0.692	0.714	0.431	0.782	0.690
<b>4</b>					1	0.336	0.362	0.203	0.431	0.295
<b>5</b>						1	0.768	0.352	0.742	0.680
<b>6</b>							1	0.335	0.862	0.740
<b>7</b>								1	0.319	0.361
<b>8</b>									1	0.820
<b>9</b>										1

### C.2 Recognition Confusion Matrix using Clean Templates

		Prediction Outputs										
		0	1	2	3	4	5	6	7	8	9	Sum
Actual Outputs	0	75		14		2						91
	1		64	1		1			18			84
	2			68	10				3		6	87
	3	1			84							85
	4					74						74
	5				2		73	10				85
	6					3		72				75
	7		4	6	1				61		2	74
	8			6		1		4		55	4	70
	9		1	5							69	75
Sum		76	69	100	97	81	73	86	82	55	81	

### C.3 Example Correlation with Clean Templates

	<b>9</b>	<b>5</b>	<b>8</b>	<b>6</b>	<b>9</b>	<b>4</b>	<b>0</b>	<b>4</b>
<b>0</b>	0.652	0.348	0.674	0.665	0.607	0.336	<b>0.892</b>	0.369
<b>1</b>	0.437	0.335	0.374	0.406	0.421	0.538	0.359	0.496
<b>2</b>	0.404	0.374	0.485	0.399	0.525	0.077	0.533	0.241
<b>3</b>	0.594	0.383	0.652	0.565	0.495	0.394	0.571	0.337
<b>4</b>	0.346	0.338	0.447	0.509	0.358	<b>0.802</b>	0.368	<b>0.753</b>
<b>5</b>	0.582	0.410	0.571	0.658	0.533	0.334	0.645	0.324
<b>6</b>	0.597	<b>0.431</b>	0.682	<b>0.753</b>	0.612	0.418	0.699	0.304
<b>7</b>	0.383	0.289	0.322	0.305	0.295	0.255	0.188	0.239
<b>8</b>	0.652	0.410	<b>0.731</b>	0.685	0.627	0.438	0.732	0.410
<b>9</b>	<b>0.801</b>	<b>0.431</b>	0.662	0.651	<b>0.735</b>	0.334	0.699	0.368

## D Averaged Templates

### D.1 Correlation of Averaged Templates

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
<b>0</b>	1	0.214	0.559	0.461	0.473	0.670	0.756	0.308	0.823	0.748
<b>1</b>		1	0.343	0.500	0.533	0.358	0.233	0.529	0.302	0.354
<b>2</b>			1	0.490	0.318	0.432	0.421	0.633	0.543	0.540
<b>3</b>				1	0.459	0.625	0.524	0.451	0.589	0.584
<b>4</b>					1	0.376	0.504	0.405	0.593	0.458
<b>5</b>						1	0.751	0.289	0.698	0.703
<b>6</b>							1	0.312	0.824	0.681
<b>7</b>								1	0.372	0.425
<b>8</b>									1	0.778
<b>9</b>										1

### D.2 Recognition Confusion Matrix using Averaged Templates

		Prediction Outputs										
		0	1	2	3	4	5	6	7	8	9	Sum
Actual Outputs	0	90				1						91
	1		78			2		1	3			84
	2	1		86								87
	3				83						2	85
	4					74						74
	5						79	6				85
	6							73		2		75
	7			4	1	1			68			74
	8	2						2		63	3	70
	9	3					1				71	75
Sum		96	78	90	84	78	80	82	71	65	76	

### D.3 Example Correlation with Averaged Templates

	<b>9</b>	<b>5</b>	<b>8</b>	<b>6</b>	<b>9</b>	<b>4</b>	<b>0</b>	<b>4</b>
<b>0</b>	0.601	0.499	0.736	0.757	0.691	0.428	<b>0.803</b>	0.481
<b>1</b>	0.453	0.307	0.294	0.274	0.313	0.577	0.229	0.410
<b>2</b>	0.540	0.345	0.591	0.389	0.533	0.206	0.582	0.281
<b>3</b>	0.628	0.420	0.619	0.555	0.530	0.431	0.518	0.397
<b>4</b>	0.458	0.376	0.542	0.510	0.471	<b>0.828</b>	0.398	<b>0.847</b>
<b>5</b>	0.564	<b>0.609</b>	0.607	0.716	0.699	0.390	0.604	0.383
<b>6</b>	0.546	0.571	0.697	<b>0.904</b>	0.736	0.516	0.662	0.451
<b>7</b>	0.520	0.313	0.430	0.326	0.389	0.376	0.308	0.362
<b>8</b>	0.629	0.524	<b>0.821</b>	0.806	0.795	0.528	0.740	0.543
<b>9</b>	<b>0.789</b>	0.584	0.725	0.723	<b>0.864</b>	0.431	0.654	0.405

### D.4 Algorithm for Computing Averaged Templates

Inputs:  $I$ , collection of segmented training images for a given digit  
 $n$ , number of rows in the images (default: 25)  
 $m$ , number of columns in the images (default: 15)

Output:  $V$ , the averaged template for the input digit

COMPUTE-AVERAGED-TEMPLATE( $I, n, m$ )

```

1   $c \leftarrow 0$ 
2  for each Image  $img \in I$ 
3      do  $c \leftarrow c + 1$ 
4      for  $row \leftarrow 1$  to  $n$ 
5          do for  $col \leftarrow 1$  to  $m$ 
6              do  $V[row][col] \leftarrow V[row][col] + img[row][col]$ 
7  for  $row \leftarrow 1$  to  $n$ 
8      do for  $col \leftarrow 1$  to  $m$ 
9          do if  $V[row][col] > (c/2)$ 
10             then  $V[row][col] \leftarrow 1$ 
11             else  $V[row][col] \leftarrow 0$ 
12 return  $V$ 

```

## References

- [1] H.I. Avi-Itzhak, T.A. Diep, and H. Garland. High accuracy optical character recognition using neural networks with centroid dithering. *Transactions on Pattern Analysis and Machine Intelligence*, 17(2):218–224, Feb 1995.
- [2] Henry S. Baird and Daniel P. Lopresti, editors. *Human Interactive Proofs, Second International Workshop*, volume 3517 of *Lecture Notes in Computer Science*, Bethlehem, PA, USA, May 19-20 2005. Springer.
- [3] Henry S. Baird, Michael A. Moll, and Sui-Yu Wang. Scattertype: A legible but hard-to-segment captcha. In *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 935–939, Washington, DC, USA, 2005. IEEE Computer Society.
- [4] Manuel Blum and Henry S. Baird, editors. *Human Interactive Proofs, First International Workshop*, Palo Alto, CA, January 2002. Xerox Palo Alto Research Center.
- [5] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):690–706, July 1996.
- [6] Kumar Chellapilla and Patrice Y. Simard. Using machine learning to break visual human interaction proofs (HIPs). In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 265–272. MIT Press, Cambridge, MA, 2005.
- [7] Monica Chew and Henry S. Baird. Baffletext: A human interactive proof. In *Proceedings of SPIE-IS&T Electronic Imaging, Document Recognition and Retrieval X*, pages 305–316, 2003.
- [8] Monica Chew and J. Doug Tygar. Image recognition captchas. Technical Report UCB/CSD-04-1333, EECS Department, University of California, Berkeley, Aug 2004.
- [9] Allison L. Coates, Henry S. Baird, and Richard J. Fateman. Pessimial print: A reverse turing test. In *Proceedings, IAPR 6th Int'l Conf. on Document Analysis and Recognition*, pages 1154–1158, Seattle, WA, Sept 2001. IEEE Computer Society.
- [10] John Douceur, Jeremy Elson, Jon Howell, and Jared Saul. Asirra: a captcha that exploits interest-aligned manual image categorization. In *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 366–374, New York, NY, USA, 2007. ACM.
- [11] P. Gader, Magdi Mohamed, and JH Chiang. Segmentation-based handwritten word recognition. *Proceedings of USPS Fifth Advanced Technology Conference*, pages 215–225, November 1992.

- [12] Richard L. Hoffman and J. Warren McCullough. Segmentation methods for recognition of machine-printed characters. In *Mechanics of Materials*, volume 15, pages 153–165. IBM Corporation, March 1971.
- [13] Jr. Kenneth Crawford Hayes. Reading handwritten words using hierarchical relaxation. In *Computer Graphics and Image Processing*, volume 14, pages 344–364, 1980.
- [14] Daniel P. Lopresti, C. Shih, and G. Kochanski. Human interactive proofs for spoken language interfaces. In *Proceedings of the 1st Workshop on Human Interactive Proofs*, pages 30–34, Palo Alto, CA, January 2002.
- [15] N. Mani and B. Srinivasan. Application of artificial neural network model for optical character recognition. *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'. 1997 IEEE International Conference on*, 3:2517–2520, October 1997.
- [16] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: breaking a visual captcha. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 1:134–141, June 2003.
- [17] George L. Nagy, Stephen V. Rice, and Thomas A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1999.
- [18] Moni Naor. Verification of a human in the loop or identification via the turing test. Sept 1996.
- [19] E.M. de A. Neves, A. Gonzaga, and A.F.F. Slaets. A multi-font character recognition based on its fundamental features by artificial neural networks. *Cybernetic Vision, 1996. Proceedings., Second Workshop on*, pages 196–201, December 1996.
- [20] Amalia Rusu and Venu Govindaraju. Handwritten CAPTCHA: using the difference in the abilities of humans and machines in reading handwritten words. *Frontiers in Handwriting Recognition, 2004. Ninth International Workshop on*, pages 226–231, October 2004.
- [21] A. Thayananthan, B. Stenger, P.H.S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 1:127–133, June 2003.
- [22] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, October 1950.
- [23] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using Hard AI Problems for Security. In *Advances in cryptology-EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques*, Warsaw, Poland, May 2003. Springer.

- [24] Jin Wang and Jack Jean. Segmentation of merged characters by neural networks and shortest-path. In *SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*, pages 762–769, New York, NY, USA, 1993. ACM.
- [25] R. Allen Wilkinson and Michael D. Garris. Comparison of massively parallel hand-print segmenters. Pb-93-113561/xab;nistir-4923, National Inst. of Standards and Technology (CSL). Advanced Systems Div., Gaithersburg, MD, September 1992.
- [26] Bin Yu and Anil K. Jain. A generic system for form dropout. *Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1127–1134, November 1996.