Abstract

Human Interactive Proofs (HIPs) are a method used to differentiate between humans and machines on the internet. Providers of online services such as PayPal.com use HIPs to prevent automated signups and abuse of their services. In this experiment, a three step algorithm has been developed to break the PayPal.com HIP. The image is preprocessed to remove noise using thresholding and a simple cleaning technique, and then segmented using vertical projections and candidate split positions. Four classification methods have been implemented: pixel counting, vertical projections, horizontal projections and template correlations. The system was trained on a sample of twenty PayPal.com HIPs to create thirty-six training templates (one for each character: 0-9 and A-Z). A sample of 100 PayPal.com HIPs were used for testing. The following HIP success rates have been achieved using the different classifiers: 8% pixel counting, vertical projections 97%, horizontal projections 100%, template correlations 100%. Three of the classifiers outperform the 88% HIP success rate of [6].

1 Introduction

Human Interactive Proofs (HIPs) are a method used to automatically differentiate between humans and machines on the internet. 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. They are typically implemented as an image of distorted text which the user must correctly transcribe. However, nearly all of the text recognition based HIPs are insecure against attacks based on neural networks [3], shape matching [11], and simple pattern recognition [17]. In this experiment, the PayPal.com HIP has been successfully broken using the conventional OCR process: pre-processing, segment, classify. Four classifiers have been implemented and the re-
results are compared using the following metrics: HIP accuracy, character accuracy, confidence, and running time. The classifiers are based on the following methods: pixel counting, vertical projections, horizontal projections, and template correlations. Section 2 provides a background on OCR and HIPs. The experiment’s methodology is explained in Section 3. Section 4 details the training stage. Section 5 explains and compares the results of the system. Section 6 summarizes the approaches and the results of the work presented.

2 Background

2.1 OCR

*Optical Character Recognition* (OCR) is the process of translating images of handwritten, typewritten, or printed text into a format understood by machines for the purpose of editing, indexing, searching, or compression [14]. The OCR process can be broken down into three tasks: pre-processing, segmentation, and classification.

2.1.1 Pre-processing

Pre-processing is necessary if the input image is noisy due to old paper, poor printers, bad scanners, etc. Generally, the pre-processing task 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 the text. Thresholding is the process of setting all intensity values greater than some threshold value to “on” and is often used as a method of binarizing an image. Thresholding is often used to remove noise 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 segmenter.

2.1.2 Segmentation

Segmentation is the process of breaking the input image into segments which contain a single entity. Character-based 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, and sometimes 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 OCR process [2]:

- **Dissection Approach**: a single partitioning of the image into subimages based on “character-like” properties, followed by classification of the subimages
- **Classification-based Approach**: segmentation where the image is iteratively searched for components that most closely match the classes in the alphabet
- **Holistic Approach**: recognize words as single units (no character segmentation)

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 (we have done so in this paper). 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.

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 re-split into two new segments.

The earliest and simplest form of dissection relies on vertical whitespace between successive characters. To make segmentation even easier, a fixed pitch if often used (pitch is the number of characters per unit of horizontal distance). In applications that use machines to print the text, text is often printed with a fixed pitch using limited font sets. Hoffman and Mccullough [7] 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.

Projection analysis is another simple one-dimensional segmentation method that uses the vertical projection (or vertical histograms) of “on” pixels to determine dissection candidates. A vertical projection is simply a column-wise count of “on” pixels. If the count falls below a predefined threshold, the column is a candidate for splitting the image. The segmentation boundaries can be further emphasized by observing the derivative of the vertical projection data (well-defined peaks will occur at the character boundaries). Projection analysis works well on high quality machine printed documents. However, vertical projection performs poorly on text which is italicized machine printed text, handwritten text (naturally slanted), or has not been properly skew-corrected.

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 character’s pixels as block or line adjacency graphs. Intuitively, a character is a single connected component 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 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. Unfortunately, if characters are broken into multiple pieces (due to pre-processing artifacts, noise, etc.), connected component analysis yields poor segmentation results.
Classification-based segmentation bypasses the requirement to discretely segment the word. No complex dissection algorithm is required. Instead, the segmenter interacts directly 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 classifier’s confidence of the sampled window. Therefore, the criterion for good segmentation is the classification confidence given by the classifier of the subimage.

If the words to be recognized are dictionary words, N-gram statistics can be introduced to classification-based approaches to prune the search space [8]. For example, 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’. After narrowing the possible guesses, a dictionary can be used to eliminate incorrectly spelled words in favor of correctly spelled words.

Holistic approaches attempt to recognize entire words as single units. 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 requires a pre-defined lexicon. For many applications, such as check recognition or postal code reading, this constraint is satisfiable.

2.1.3 Character classification

Character classification is strongly dependent on feature vectors which are extracted from the characters. Feature extraction is the process of transforming the input data into a reduced representation. It is commonly employed when their 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 image provides an accurate description of a larger set of data. A common feature vector is image projections which represent the character as a vector of projection counts (discussed above). Naive methods feed entire image matrices to the classifier, while others require experts to develop visual cues to distinguish characters from one another [13]. 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 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 finding the nearest neighbor, neural networks [10, 1, 13], or other techniques. However, the classification method is largely non-important in the recognition process; by far the most important decision is the selection of features.

2.2 HIPs

Human Interactive Proofs (HIPs) are a class of automated challenges used to differentiate between legitimate human users and automated, malicious robots on the internet. HIPs have many practical security applications, including preventing the abuse of online services such as free email providers. The term HIP is preferred over the more common (and unfortunately trademarked) term, Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHAs). HIP challenges should be easy for a machine to automatically generate, easy for a human to solve, and difficult, or im-
possible, for another machine to solve. The key to developing a successful HIP challenge is to choose a difficult artificial intelligence problem where a gap exists between human and machine capabilities.

Researchers have suggested HIPs based on hard artificial intelligence problems such as natural language processing [16], character recognition [4], image understanding [5], and speech recognition [9]. Most commercial implementations require the user to transcribe a string of distorted characters with background noise. This type of HIP can be considered broken through techniques such as shape matching [11], distortion estimation [12], and even simple pattern recognition [17]. Unfortunately, most commercial implementations are even easier to break than the research/academic implementations (as this experiment clearly demonstrates).

3 Method

To break the PayPal.com HIP, the problem can be reduced to an OCR task. As mentioned before, the OCR task can be broken down into three steps: pre-processing, segmentation, and classification. For clarity sake, the code is also separated into these three distinct steps.

3.1 Pre-processing

The pre-processing step is arguably the most important step when the image contains adversarial noise (such as HIPs). The noise placed on top of the HIP challenge images is designed to confuse off-the-shelf OCR systems. Before successful segmentation or classification can occur, the noise must be removed. The first step in our process is to convert the image to greyscale. The image is then thresholded to remove the noise (horizontal and vertical lines). The background thresholding technique is incredibly simple but removes nearly all of the noise in the image. Occasionally, additional noise still remains after this step. Therefore, additional cleaning is performed to remove pixels where the entire row has very few “on” pixels. A bounding box is then placed around the string of characters and cropped out.

Figure 1: The pre-processing step.

3.2 Segmentation

Next, the pre-processed image is fed into the segmenter. A connected components approach was first attempt, but unfortunately the pre-processing step occasionally breaks characters into multiple segments (see the first character in Figure 2a). However, the segmentation process is simplified because the PayPal HIP is always rendered with exactly five characters. Vertical projections [7] and candidate split positions are used to de-
termine segmentation boundaries. Splitting on every projection with zero “on” pixels occasionally causes characters to be split into multiple segments (as was the problem with the CC-based approach). However, empirical exploration shows that every character is at least ten pixels wide. A Hoffman and McCullough style approach [7] is used and a column-wise scan is performed from the left side of the image to the right side. When the start of a character is detected, ten pixels are skipped and the scan is continued. When the end of a character is detected, the segment is cropped out of the image. The segment is padded out using 0’s to a fixed size (20 × 20) as a requirement of the classification process (correlation requires that the dimensions of the two matrices must agree). This process is repeated until the end of the image has been reached.

Figure 2: The segmentation step.

3.3 Classification

The segmenter feeds the five individual characters to the classifier. Note that the input images are binary images, consisting of 1’s for the foreground (the character) and 0’s for the background. The classification procedures (defined below) are invoked with the unknown sample image $I$ and the set of template images $T$. Several of the classifiers computed correlation coefficients. The correlation coefficient (CORR2) between two input vectors or matrices $i$ and $j$, can be computed as follows:

$$
\frac{\sum_m \sum_n (i_{mn} - \bar{i})(j_{mn} - \bar{j})}{\sqrt{\left(\sum_m \sum_n (i_{mn} - \bar{i})^2\right)\left(\sum_m \sum_n (j_{mn} - \bar{j})^2\right)}}
$$

where $\bar{i}$ is the mean of the input matrix $i$ and $\bar{j}$ is the mean of the input matrix $j$.

3.3.1 Pixel Counting

The pixel counting classifier compares the Euclidean distance between pixel counts. The pixels for a binary image $I$ can be counted using the following algorithm:

```plaintext
PixelCount(I)
1  k ← 0
2  for r ← 1 to I_{numRows}
3    do for c ← 1 to I_{numCols}
4      do k ← k + I[r][c]
5  return k
```

The pixel count of the input image is then compared against the pixel counts of each of the template images. The index of the template that has the least difference in pixel count is returned as the match:

```plaintext
ClassifyPC(I, T)
1  D ← ∅
2  for each Template $t_i \in T$
3    do $d_i ← abs(PixelCount(t_i) - PixelCount(I))$
4  return $k$ such that $d_k = min(D)$
```
Note that this method does not take any spatial layout into account. Therefore, an image with a pixel in each of it’s four corners will match perfectly with a template image with a block of four pixels in the center of the image, even though they are visually dissimilar.

### 3.3.2 Vertical Projections

The vertical projection classifier compares correlation coefficients of vertical projections. The vertical projection of an image $I$ can be calculated using the following algorithm:

$\text{VerticalProjection}(I)$

1. $V \leftarrow \emptyset$
2. for $r \leftarrow 1$ to $I_{\text{numRows}}$
3. do for $c \leftarrow 1$ to $I_{\text{numCols}}$
4. do $v_c \leftarrow v_c + I[r][c]$
5. return $V$

The vertical projection of the input image is then compared against the vertical projections of each of the template images. The index of the template whose vertical projection has the the highest correlation coefficient with the input image’s vertical projection is returned as the match:

$\text{ClassifyVP}(I, T)$

1. $R \leftarrow \emptyset$
2. for each Template $t_i \in T$
3. do $r_i \leftarrow \text{CORR2}($
4. $\text{VerticalProjection}(t_i),$
5. $\text{VerticalProjection}(I))$
6. return $k$ such that $r_k = \max(R)$

### 3.3.3 Horizontal Projections

The horizontal projection classifier compares correlation coefficients of horizontal projections. The horizontal projection of an image $I$ can be calculated using the following algorithm:

$\text{HorizontalProjection}(I)$

1. $V \leftarrow \emptyset$
2. for $r \leftarrow 1$ to $I_{\text{numRows}}$
3. do for $c \leftarrow 1$ to $I_{\text{numCols}}$
4. do $v_r \leftarrow v_r + I[r][c]$
5. return $V$

The horizontal projection of the input image is then compared against the horizontal projections of each of the template images. The index of the template whose horizontal projection has the the highest correlation coefficient with the input image’s horizontal projection is returned as the match:
3.3.4 Template Correlations

The template correlation classifier calculates the 2D correlation coefficients for the input image $I$ and the templates $T$. The index of the template with the highest 2D correlation coefficient with the input image is returned as the match:

$$\text{CLASSIFYHP}(I, T)$$
1. $R \leftarrow \emptyset$
2. for each Template $t_i \in T$
3. \hspace{1em} do $r_i \leftarrow \text{CORR2}($
4. \hspace{2em} HorizontalProjection($t_i$),
5. \hspace{2em} HorizontalProjection($I$))
6. return $k$ such that $r_k = \max(R)$

$$\text{CLASSIFYTC}(I, T)$$
1. $R \leftarrow \emptyset$
2. for each Template $t_i \in T$
3. \hspace{1em} do $r_i \leftarrow \text{CORR2}($
4. \hspace{2em} $t_i$, $I$)
4. return $k$ such that $r_k = \max(R)$

Figure 4: HP confidences for “C6X62”.

Figure 5: TC confidences for “C6X62”.

4 Training

The templates were created from a set of twenty training PayPal HIPs. The set of images were randomly chosen and contain all characters in the character set (note that PayPal does not use $I$, $O$, $Q$, $0$, or $1$ in the character set to increase usability for humans). The images were processed using the same pre-processing and segmentation algorithm as specified above. In many cases, the training data has multiple samples for a given
character $c$: $s^c_1, s^c_2, \ldots, s^c_n$. If multiple samples for a single character $c$ exist, the final template $t^c$ is computed by averaging all samples for a given character:

$$t^c = \left( \sum_{i=1}^{n} s^c_i \right) / n$$

Informally, this creates more robust templates, as we are using multiple training samples to generate the templates. It can be thought of as many training samples voting on whether or not the ground truth should contain a given pixel.

5 Results

5.1 Testing Results

A sample of 100 random PayPal HIPs were used for testing. The samples were manually downloaded and labeled by visual inspection. The same pre-processing and segmentation algorithms were used for all classifiers. The classifiers are evaluated with several metrics: HIP accuracy refers to the percentage of the 100 testing samples which were correctly recognized. Character accuracy refers to the percentage of the 500 characters of the 100 testing samples which were correctly recognized. The HIP accuracy should be roughly equal to character accuracy raised to the fifth power (serial repetition). The classifiers which utilize correlation can also return a confidence value. The character confidence can be represented by the correlation coefficient $r_i$ (1.0 means a perfect match). A overall string confidence $C$ can be calculated by multiplying each of the character correlation coefficients $r_i$ together:

$$C = \prod_{i=1}^{5} r_i$$

Note that this confidence metric cannot be used with the pixel counting classifier because the that classifier does not utilize correlation during classification. Running time, measured in seconds, was clocked on a 2 GHz Intel Core 2 Duo with 2 GB of memory, running Mac OS X 10.4.11 and MATLAB R2007a. Full outputs from all four classifiers are located in Appendix B.

The following is a comparison of the four classifiers: pixel counting (PC), vertical projections (VP), horizontal projections (HP), and template correlations (TC).

<table>
<thead>
<tr>
<th></th>
<th>PC</th>
<th>VP</th>
<th>HP</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{Char}$</td>
<td>63.2%</td>
<td>99.4%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>$A_{HIP}$</td>
<td>8%</td>
<td>97%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>$C_{avg}$</td>
<td>n/a</td>
<td>98.9%</td>
<td>98.8%</td>
<td>95.9%</td>
</tr>
<tr>
<td>$C_{min}$</td>
<td>n/a</td>
<td>89.1%</td>
<td>93.9%</td>
<td>67.9%</td>
</tr>
<tr>
<td>$T_{avg}$</td>
<td>0.02s</td>
<td>0.06s</td>
<td>0.06s</td>
<td>0.06s</td>
</tr>
</tbody>
</table>

where $A_{Char}$ is the character accuracy, $A_{HIP}$ is the HIP accuracy, $C_{avg}$ is the average overall string confidence, $C_{min}$ is the lowest overall string confidence observed during testing, and $T_{avg}$ is the average recognition running time in seconds.

The only benefit that the pixel counting method has over the others is run time. The vertical projection classifier made three misclassifications: ‘27LP5’ recognized as ‘27LP2’, ‘5RESL’ recognized as ‘2RESL’, and ‘5SWMJ’ recognized as ‘2SMWJ’. We can see that all three mistakes were made due to a misclassification of a ‘5’ as a ‘2’. This shows that vertical projections are not a sufficient method for differentiating between 5’s and 2’s.

Figure 6 plots the confidence of the three classifiers (vertical projections shown in blue, horizontal projections shown in green, and template correlations shown in red) for all 100 testing samples. We can see that even
though they use different classification techniques, the confidence values seem to be consistent with one another. That is to say, that all three classifiers achieve low confidences on the same images. For example, sample #72 achieved very low confidence for both the horizontal projection and template correlation classifiers. Similarly, all three classifiers performed very well on sample #28. This indicates that samples with low confidence values may exhibit some pre-processing artifacts that make classification a difficult task, no matter what the technique.

5.2 Example

This section visually demonstrates the entire recognition process using an example PayPal HIP image. Figure 1 illustrates the preprocessing step. Figure 2 displays the the segmentation process. Figure 3 shows the confidence values for the vertical projection classifier. The overall confidence for the example HIP is 0.992 (0.995 * 0.999 * 1.0 * 0.998 * 1.0). Notice that there are several other characters with very high confidences. Figure 4 shows the confidence values for the horizontal projection classifier. The overall confidence for the example HIP is 0.994 (0.995 * 1.0 * 1.0 * 0.999 * 1.0). Notice that the range of the confidence values is fairly small and several peaks exist. Figure 5 shows the confidence values for the template correlation classifier. The overall confidence for the example HIP is 0.992 (1.0 * 0.999 * 0.997 * 0.996 * 1.0). Notice that there is only a single peak in the confidence values for every character. This indicates that the classifier is very good at discriminating between character classes.

6 Conclusion

We have presented a robust way to automatically recognizing the character strings inside of a PayPal.com HIP using a three step pre-process, segment, classify algorithm. Four classifiers (pixel counting, vertical projections, horizontal projections, and template correlations) were implemented, evaluated, and compared. Two of the classifiers have achieved perfect HIP accuracy on the test set of 100 images. Upon visual inspection of the correlation coefficients for several test images, we see that the template correlation classifier discriminates better than the other classifiers and is strongly recommended.
References


Appendices

A MATLAB Code

This section contains the MATLAB code used to break the PayPal HIP. The code is thoroughly commented and should be self-explanatory.

A.1 recognizeAll.m

function recognizeAll()
% Performs recognition of the entire testing set of PayPal CAPTCHA images
% by preprocessing, segmentation, and classification.
%
% Created by Kurt Alfred Kluever (kurt@kloover.com)

testingDir = 'testing/';
testingSamples = dir(strcat(testingDir, '*.jpg'));
umTestingSamples = size(testingSamples, 1);
charCorrect = 0;
charWrong = 0;
hipCorrect = 0;
confidences = zeros(1, numTestingSamples);
tic
% For each of the testing images...
for i=1:numTestingSamples
    fn = strcat(testingDir, testingSamples(i).name);
    % Perform recognition and record the result and confidence
    [chars c] = recognize(fn);
    confidences(i) = c;
    fn = strrep(fn, testingDir, '');
    fn = strrep(fn, '.jpg', '');
    % Print out the results
    fprintf('Actual: %s Decoded: %s Confidence: %f', fn, chars, c);
    if (strcmp(fn, chars) == 0)
        fprintf(' Correct
');
        hipCorrect = hipCorrect + 1;
    else
        fprintf(' Incorrect
');
        charWrong = charWrong + 1;
    end
    for j=1:5
        if (strcmp(fn(j), chars(j)) == 0)
            charCorrect = charCorrect + 1;
        else
            charWrong = charWrong + 1;
        end
    end
end
toc
charAcc = charCorrect / (charCorrect + charWrong);
hipAcc = hipCorrect / numTestingSamples;
avgConfidence = sum(confidences) / numTestingSamples;
minConfidence = min(confidences);
fprintf('Character Accuracy: %f\n', charAcc);
fprintf('HIP Accuracy: %f\n', hipAcc);
fprintf('Average confidence: %f\n', avgConfidence);
fprintf('Minimum confidence: %f\n', minConfidence);
end

12
**A.2 recognize.m**

```matlab
function [decoded confidence] = recognize(imageFileName)
% Performs recognition of PayPal CAPTCHA images by preprocessing,
% segmentation, and classification. To switch which classifier is being
% used, simply uncomment out the one you wish to use!
% Created by Kurt Alfred Kluever (kurt@kloover.com)

% If the templates haven't been created yet, make them now!
if (exist('templates.mat', 'file') == 0)
    fprintf('Training templates do not exist. Creating them now...');
    makeTemplates();
    fprintf('DONE
);
end
% 1. Load and preprocess the image
preprocessed = preprocess(imread(imageFileName));
% 2. Segmentation the image into characters
segmented = segment(preprocessed);
% 3. Classifiy the characters
[decoded confidence] = classify(segmented, 'PixelCounts');
[decoded confidence] = classify(segmented, 'VerticalProjections');
[decoded confidence] = classify(segmented, 'HorizontalProjections');
[decoded confidence] = classify(segmented, 'TemplateMatching');
end
```

**A.3 preprocess.m**

```matlab
function bounded = preprocess(i)
% Performs preprocessing on the input image. The image is first converted
% to greyscale and then thresholded. Random noise is removed via
% thresholding. A bounding box is then placed around the entire image.
% Returns the preprocessed image.
% Created by Kurt Alfred Kluever (kurt@kloover.com)

% Convert the color image to grey scale
greyScale = rgb2gray(i);
% Threshold out the background noise
thresholded = greyScale < 30;
% Remove any random noise that will hurt the char offset
% row = size(thresholded, 1);
row = 1;
while (row < rows)
    rowSum = sum(thresholded(row,:));
    if (rowSum < 5 && rowSum > 0)
        thresholded(row,:) = 0;
    end
    row = row + 1;
end
% Place a bounding box around the image
bb = regionprops(double(thresholded), 'BoundingBox');
% Crop out the contents of the bounding box
bounded = imcrop(thresholded, bb.BoundingBox);
end
```

**A.4 segment.m**

```matlab
function [retVal] = segment(bounded)
% Performs character segmentation of the preprocessed input image.
% Returns the segmented set of characters.
% Created by Kurt Alfred Kluever (kurt@kloover.com)

% Create the return value (5 images, 20x20 in size)
retVal = zeros(20, 20, 5);
% Get the size of the input image
[rows cols] = size(bounded);
col = col + 10;
% Scan forward while columns contain data
while (col < cols) && (sum(sum(bounded(:,col:col+2))) > 0)
    col = col + 1;
end
% Crop the character out of the image
a = incrop(bounded, [startCol 1 (col - startCol) rows]);
% Pad out to 20 rows and 20 cols with 0's
a = padarray(a, [20 - rowsCharColsCol 20 - charCols], 'post');
% Set the character into the return value
retVal(:,charIndex) = a;
% Increment the characters index
charIndex = charIndex + 1;
% Increment the column counter
col = col + 1;
startCol = col;
% Advance the column counter while there is white space
while (col < cols) && (sum(sum(bounded(:,col))) == 0)
    col = col + 1;
    startCol = startCol + 1;
end
end
```

**A.5 classify.m**

```matlab
function [decoded confidence] = classify(chars, method)
% Performs character classification of the segmented input image using
% pixel counts, vertical projections, horizontal projections, or template
% matching.
% Returns the classified string of characters and the confidence.
% Created by Kurt Alfred Kluever (kurt@kloover.com)

% Load the templates
load templates;
% Turn off the warnings about dividing by zero
warning off MATLAB:divideByZero
% Setup the decoded result
decoded = char(zeros(1,5));
% Confidence starts at 1.0 (perfect)
confidence = 1.0;
% For each of the 5 characters in the image...
for i=1:5
% Correlation results for all the template images
allCorrs = zeros(1,36);
% For each of the templates...
for j=1:36
    if (strcmp(method, 'PixelCounts') == 1)
        tempSum = sum(sum(templates(:,:,j))); % Note that we subtract it from 50 so that we can still
        inSum = sum(sum(chars(:,:,i))); % use max() to find the correct index below!
        allCorrs(j) = 50 - abs(tempSum - inSum);
    elseif (strcmp(method, 'VerticalProjections') == 1)
        tempVP = sum(templates(:,:,j));
        inVP = sum(chars(:,:,i));
        allCorrs(j) = corr2(tempVP, inVP);
    elseif (strcmp(method, 'HorizontalProjections') == 1)
        tempHP = sum(templates(:,:,j)');
        inHP = sum(chars(:,:,i)');
        allCorrs(j) = corr2(tempHP, inHP);
    else % Do template matching by default
        temp = templates(:,:,j);
        in = chars(:,:,i);
        allCorrs(j) = corr2(temp, in);
    end
end
% Used for plotting data
subplot(5,1,i); bar(abs(allCorrs));
set(gca,'XTick',1:36)
end
```
A.6 makeTemplates.m

function makeTemplates()
    % Creates the classification templates using a set of training images. The
    % training images are labeled as ground truth by their filenames.
    %
    % Created by Kurt Alfred Kluever (Kurt@klover.com)
    %
    % Set aside room for the templates and the counts
    % templates = zeros(20,20,36);
    % counts = zeros(1,36);
    %
    % Load the images from the directory
    trainingDir = 'training/';
    trainingSamples = dir(strcat(trainingDir, '*.jpg'));
    % For each of the training images...
    for i=1:numTrainingSamples
        % For each of the characters...
        for i=1:36
            % Convert from ASCII character to 32x32 int (an index)
            template = uint8(filename(i));
            if (template >= 65 && template <= 90) % upper case character
                % 'A','B','C','D','E','F','G','H','I','J','K','L','M,'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'}
                index = index + 54;
            elseif (template >= 48 && template <= 57) % number
                % '0','1','2','3','4','5','6','7','8','9',
                index = index + 47;
            else
                % We should never get here
                end
        end
        % Store the decoded character
        decoded(i) = char(index);
    end
end

B MATLAB Output

This section contains the output of the recognizeAll MATLAB script using the different classifiers. The output contains the recognition attempts for each of the 100 testing samples, as well as the metrics on which the classifiers are evaluated.

B.1 Pixel Counting

Actual: 2TLPs Decoded: 27V6E Confidence: 302190625.000000 Incorrect
Actual: 2CDQ2 Decoded: 2AD20 Confidence: 304666666.66667 Incorrect
Actual: 2CKTT Decoded: 28HT Confidence: 303120000.000000 Incorrect
Actual: 29R6A Decoded: 582A2 Confidence: 300152222.22222 Incorrect
Actual: 2YQ54 Decoded: 2Y042 Confidence: 295082083.33333 Correct
Actual: 379M5 Decoded: 379M5 Confidence: 291946875.00000 Correct
Actual: 38503 Decoded: 38503 Confidence: 286800000.00000 Incorrect
Actual: 3EY9U Decoded: 33YHT Confidence: 309375000.00000 Incorrect
Actual: 3H3STY Decoded: 3H3ST Confidence: 307312500.00000 Incorrect
Actual: 4597T Decoded: 4597T Confidence: 29899816.203704 Incorrect
Actual: 48654 Decoded: 48654 Confidence: 30016742.59293 Incorrect
Actual: 43633 Decoded: 43633 Confidence: 296961111.11111 Incorrect
Actual: 4862W Decoded: 4862W Confidence: 30775937.50000 Incorrect
Actual: 48AX4 Decoded: 48AX4 Confidence: 293114869.44444 Incorrect
Actual: 5L2X9 Decoded: 5L2X9 Confidence: 302190525.00000 Incorrect
Actual: 5332L Decoded: 5332L Confidence: 293321875.00000 Incorrect
Actual: 5334L Decoded: 5334L Confidence: 298020000.00000 Incorrect
Actual: 5NN63 Decoded: 5NN63 Confidence: 306583333.33333 Correct
Actual: 6V2BP Decoded: 6V2BP Confidence: 30173101.38185 Incorrect
Actual: 7Y2B3 Decoded: 7Y2B3 Confidence: 301166250.00000 Correct
Actual: 82658 Decoded: 82658 Confidence: 29682083.33333 Incorrect
Actual: 82GL4 Decoded: 82GL4 Confidence: 30439375.00000 Incorrect
Actual: 9362G Decoded: 9362G Confidence: 290183720.75000 Incorrect
Actual: 93LL2 Decoded: 93LL2 Confidence: 30569125.00000 Incorrect
Actual: 94L7L Decoded: 94L7L Confidence: 298154587.50000 Incorrect
Actual: A04CT Decoded: A04CT Confidence: 301104166.66667 Incorrect
Actual: A0J4M Decoded: A0J4M Confidence: 30114968.75000 Incorrect
Actual: A3Z7V Decoded: A3Z7V Confidence: 308343750.00000 Incorrect
Actual: B44XZ Decoded: B44XZ Confidence: 304208333.33333 Incorrect
Actual: B6L3C Decoded: B6L3C Confidence: 303120000.00000 Incorrect
Actual: B6EST Decoded: B6EST Confidence: 297123750.00000 Incorrect
Actual: B9J86 Decoded: B9J86 Confidence: 299410000.00000 Incorrect
Actual: C3MLE Decoded: C3MLE Confidence: 29810666.66667 Correct
Actual: C3THE Decoded: C3THE Confidence: 298105250.00000 Incorrect
Actual: C6S2Z Decoded: C6S2Z Confidence: 301104166.66667 Incorrect
Actual: C6526 Decoded: C6526 Confidence: 29772247.00000 Incorrect
Actual: C29HA Decoded: C29HA Confidence: 306250000.00000 Incorrect
Actual: C6D4V Decoded: C6D4V Confidence: 293621416.62500 Incorrect
Actual: C6SRS Decoded: C6SRS Confidence: 300165742.592593 Correct
Actual: C6YD7 Decoded: C6YD7 Confidence: 300165742.592593 Correct
Actual: F3N36 Decoded: F3N36 Confidence: 300731666.66667 Incorrect
Actual: EF33R Decoded: EF33R Confidence: 300731666.66667 Incorrect
Actual: F3T3Y Decoded: F3T3Y Confidence: 29413073.61111 Incorrect
Actual: FG390 Decoded: FG390 Confidence: 304166666.66667 Incorrect
Actual: FG52P Decoded: FG52P Confidence: 302148583.33333 Incorrect
Actual: FGT7C Decoded: FGT7C Confidence: 298792619.50000 Incorrect
Actual: FTYTV Decoded: FTYTV Confidence: 292683333.33333 Correct
Actual: GBR33 Decoded: GBR33 Confidence: 301041666.66667 Incorrect
Actual: GF535 Decoded: GF535 Confidence: 298426785.41667 Incorrect
Actual: G652P Decoded: G652P Confidence: 302393750.00000 Incorrect
Actual: G6Y3E Decoded: G6Y3E Confidence: 298124166.66667 Incorrect
Actual: G8LY2 Decoded: G8LY2 Confidence: 297123750.00000 Incorrect
Actual: GKV3K Decoded: GKV3K Confidence: 302190625.00000 Incorrect
Actual: G5533 Decoded: G5533 Confidence: 303120000.00000 Incorrect
Actual: HX72C Decoded: H872C Confidence: 297902812.50000 Incorrect
Actual: HTJGD Decoded: HTASD Confidence: 297062500.00000 Incorrect
Actual: HTZC2 Decoded: HTZC2 Confidence: 297062500.00000 Incorrect
Actual: J2L3S Decoded: J2L3S Confidence: 306525000.00000 Incorrect
Actual: J24CB Decoded: J24CB Confidence: 298800000.00000 Incorrect
Actual: J95M2 Decoded: J95M2 Confidence: 302166666.66667 Incorrect
Actual: J5E5E Decoded: J5E5E Confidence: 301241666.66667 Incorrect
Actual: KBLT2 Decoded: KBLT2 Confidence: 297123750.00000 Incorrect
Actual: K8V0X Decoded: K8V0X Confidence: 305225166.66667 Incorrect
B.2 Vertical Projections

Actual: 7KLF5 Decoded: 7KLF5 Confidence: 0.976989 Correct
Actual: 2CL2D Decoded: 2CL2D Confidence: 0.99403 Correct
Actual: 2G7TT Decoded: 2G7TT Confidence: 0.997374 Correct
Actual: 2XN4A Decoded: 2XN4A Confidence: 0.990826 Correct
Actual: 292B4 Decoded: 292B4 Confidence: 0.996232 Correct
Actual: 3378M Decoded: 3378M Confidence: 0.997336 Correct
Actual: 38273 Decoded: 38273 Confidence: 0.971717 Correct
Actual: 3EYHZ Decoded: 3EYHZ Confidence: 0.999960 Correct
Actual: 3HHY Decoded: 3HHY Confidence: 0.999076 Correct
Actual: 3R7RT Decoded: 3R7RT Confidence: 0.998384 Correct
Actual: 46354 Decoded: 46354 Confidence: 0.992146 Correct
Actual: 4583C Decoded: 4583C Confidence: 0.997075 Correct
Actual: 4941C Decoded: 4941C Confidence: 0.998483 Correct
Actual: 59328 Decoded: 59328 Confidence: 0.999348 Correct
Actual: 5BSL2 Decoded: 5BSL2 Confidence: 0.970642 Correct
Actual: 5EEAL Decoded: 5EEAL Confidence: 0.997650 Correct
Actual: 5PN6J Decoded: 5PN6J Confidence: 0.995604 Correct
Actual: 62ZEP Decoded: 62ZEP Confidence: 0.998750 Correct
Actual: 7G7F3 Decoded: 7G7F3 Confidence: 0.997531 Correct
Actual: 8558S Decoded: 8558S Confidence: 0.997187 Correct
Actual: 87L84 Decoded: 87L84 Confidence: 0.997677 Correct
Actual: 8862G Decoded: 8862G Confidence: 0.995977 Correct
Actual: 91L3X Decoded: 91L3X Confidence: 0.998297 Correct
Actual: 92L7 Decoded: 92L7 Confidence: 0.996173 Correct
Actual: 9D72X Decoded: 9D72X Confidence: 0.999481 Correct
Actual: 9E48Z Decoded: 9E48Z Confidence: 0.993810 Correct
Actual: 9X25Z Decoded: 9X25Z Confidence: 0.998547 Correct
Actual: 9E3E7 Decoded: 9E3E7 Confidence: 0.998217 Correct
Actual: 9H588 Decoded: 9H588 Confidence: 0.997814 Correct
Actual: 9L21L Decoded: 9L21L Confidence: 0.995888 Correct
Actual: 9S622 Decoded: 9S622 Confidence: 0.997867 Correct
Actual: 9S622 Decoded: 9S622 Confidence: 0.997867 Correct
Actual: 9C622 Decoded: 9C622 Confidence: 0.999225 Correct
Actual: 9C622 Decoded: 9C622 Confidence: 0.999225 Correct
Actual: 9C347 Decoded: 9C347 Confidence: 0.998417 Correct
Actual: 9C347 Decoded: 9C347 Confidence: 0.998417 Correct
Actual: 9B988 Decoded: 9B988 Confidence: 0.998688 Correct
Actual: 9E97C Decoded: 9E97C Confidence: 0.998692 Correct
Actual: 9E97C Decoded: 9E97C Confidence: 0.998692 Correct
Actual: 85793 Decoded: 85793 Confidence: 0.989143 Correct
Actual: 8D69H Decoded: 8D69H Confidence: 0.998789 Correct
Actual: 8EY1T Decoded: 8EY1T Confidence: 0.986692 Correct
Actual: 87GQT Decoded: 87GQT Confidence: 0.995325 Correct
Actual: 83EYM Decoded: 83EYM Confidence: 0.998932 Correct

B.3 Horizontal Projections

Actual: 27LFS Decoded: 27LFS Confidence: 0.994007 Correct
Actual: 2CD2D Decoded: 2CD2D Confidence: 0.984106 Correct
Actual: 2G7TT Decoded: 2G7TT Confidence: 0.986640 Correct
Actual: 295AS Decoded: 295AS Confidence: 0.987428 Correct
Actual: 2525Q Decoded: 2525Q Confidence: 0.976627 Correct
Actual: 295AS Decoded: 295AS Confidence: 0.987428 Correct
Actual: 3738M Decoded: 3738M Confidence: 0.993564 Correct
Actual: 38573 Decoded: 38573 Confidence: 0.988514 Correct
Actual: 3EYHZ Decoded: 3EYHZ Confidence: 0.997517 Correct
Actual: 3KHY Decoded: 3KHY Confidence: 0.987876 Correct
Actual: 4577T Decoded: 4577T Confidence: 0.998639 Correct
Actual: 456XY Decoded: 456XY Confidence: 0.984647 Correct
Actual: 4824N Decoded: 4824N Confidence: 0.996247 Correct
Actual: 5L3Z9 Decoded: 5L3Z9 Confidence: 0.995281 Correct
Actual: 5K35L Decoded: 5K35L Confidence: 0.987301 Correct
Actual: 5OEAL Decoded: 5OEAL Confidence: 0.989664 Correct
Actual: 5SCNJ Decoded: 5SCNJ Confidence: 0.994678 Correct
Actual: 6V2EP Decoded: 6V2EP Confidence: 0.998594 Correct
Actual: 7K7P3 Decoded: 7K7P3 Confidence: 0.995650 Correct
Actual: 8285R Decoded: 8285R Confidence: 0.989046 Correct
<table>
<thead>
<tr>
<th>Actual</th>
<th>Decoded</th>
<th>Confidence</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z69AB</td>
<td>Z69AB</td>
<td>0.991632</td>
<td>Correct</td>
</tr>
<tr>
<td>YT4TZ</td>
<td>YT4TZ</td>
<td>0.992761</td>
<td>Correct</td>
</tr>
<tr>
<td>YRN26</td>
<td>YRN26</td>
<td>0.994175</td>
<td>Correct</td>
</tr>
<tr>
<td>YBAUK</td>
<td>YBAUK</td>
<td>0.994175</td>
<td>Correct</td>
</tr>
<tr>
<td>YAWG2</td>
<td>YAWG2</td>
<td>0.99535</td>
<td>Correct</td>
</tr>
<tr>
<td>XWET3</td>
<td>XWET3</td>
<td>0.996135</td>
<td>Correct</td>
</tr>
<tr>
<td>W9B2K</td>
<td>W9B2K</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>UWUCH</td>
<td>UWUCH</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>UE64X</td>
<td>UE64X</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>U3V4C</td>
<td>U3V4C</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>TBLU2</td>
<td>TBLU2</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>S9FES</td>
<td>S9FES</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>R32WR</td>
<td>R32WR</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>P2N4P</td>
<td>P2N4P</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>NU8JM</td>
<td>NU8JM</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>MG8ES</td>
<td>MG8ES</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>M8H4U</td>
<td>M8H4U</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>LL6FJ</td>
<td>LL6FJ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>L4HGA</td>
<td>L4HGA</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>JZ53E</td>
<td>JZ53E</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>JM9SZ</td>
<td>JM9SZ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>J2LE3</td>
<td>J2LE3</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>HN94A</td>
<td>HN94A</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GRY8X</td>
<td>GRY8X</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GMLVU</td>
<td>GMLVU</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GF3GD</td>
<td>GF3GD</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>FYUYM</td>
<td>FYUYM</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>FUSGP</td>
<td>FUSGP</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>F93UV</td>
<td>F93UV</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EUYPS</td>
<td>EUYPS</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EU4XP</td>
<td>EU4XP</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EL2Z7Y</td>
<td>EL2Z7Y</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>CR6NC</td>
<td>CR6NC</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>B4ABZ</td>
<td>B4ABZ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>AZX7V</td>
<td>AZX7V</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>XG734</td>
<td>XG734</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>W9B2K</td>
<td>W9B2K</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>UE64X</td>
<td>UE64X</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>U3V4C</td>
<td>U3V4C</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>TBLU2</td>
<td>TBLU2</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>S9FES</td>
<td>S9FES</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>R32WR</td>
<td>R32WR</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>P2N4P</td>
<td>P2N4P</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>NU8JM</td>
<td>NU8JM</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>MG8ES</td>
<td>MG8ES</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>M8H4U</td>
<td>M8H4U</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>LL6FJ</td>
<td>LL6FJ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>L4HGA</td>
<td>L4HGA</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>JZ53E</td>
<td>JZ53E</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>JM9SZ</td>
<td>JM9SZ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>J2LE3</td>
<td>J2LE3</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>HN94A</td>
<td>HN94A</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GRY8X</td>
<td>GRY8X</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GMLVU</td>
<td>GMLVU</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GF3GD</td>
<td>GF3GD</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>FYUYM</td>
<td>FYUYM</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>FUSGP</td>
<td>FUSGP</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>F93UV</td>
<td>F93UV</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EUYPS</td>
<td>EUYPS</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EU4XP</td>
<td>EU4XP</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EL2Z7Y</td>
<td>EL2Z7Y</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>CR6NC</td>
<td>CR6NC</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>B4ABZ</td>
<td>B4ABZ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>AZX7V</td>
<td>AZX7V</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>XG734</td>
<td>XG734</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>W9B2K</td>
<td>W9B2K</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>UE64X</td>
<td>UE64X</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>U3V4C</td>
<td>U3V4C</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>TBLU2</td>
<td>TBLU2</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>S9FES</td>
<td>S9FES</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>R32WR</td>
<td>R32WR</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>P2N4P</td>
<td>P2N4P</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>NU8JM</td>
<td>NU8JM</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>MG8ES</td>
<td>MG8ES</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>M8H4U</td>
<td>M8H4U</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>LL6FJ</td>
<td>LL6FJ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>L4HGA</td>
<td>L4HGA</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>JZ53E</td>
<td>JZ53E</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>JM9SZ</td>
<td>JM9SZ</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>J2LE3</td>
<td>J2LE3</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>HN94A</td>
<td>HN94A</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GRY8X</td>
<td>GRY8X</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GMLVU</td>
<td>GMLVU</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>GF3GD</td>
<td>GF3GD</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>FYUYM</td>
<td>FYUYM</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>FUSGP</td>
<td>FUSGP</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>F93UV</td>
<td>F93UV</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EUYPS</td>
<td>EUYPS</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
<tr>
<td>EU4XP</td>
<td>EU4XP</td>
<td>0.996492</td>
<td>Correct</td>
</tr>
</tbody>
</table>

**B.4 Template Correlations**
Actual: UWUCH Decoded: UWUCH Confidence: 0.964454 Correct
Actual: W4GKY Decoded: W4GKY Confidence: 0.955301 Correct
Actual: W9B2K Decoded: W9B2K Confidence: 0.956179 Correct
Actual: XA3FW Decoded: XA3FW Confidence: 0.959466 Correct
Actual: XCEHZ Decoded: XCEHZ Confidence: 0.977761 Correct
Actual: XSFLR Decoded: XSFLR Confidence: 0.984414 Correct
Actual: XKRT3 Decoded: XKRT3 Confidence: 0.943594 Correct
Actual: YAMG2 Decoded: YAMG2 Confidence: 0.959516 Correct
Actual: YBAUK Decoded: YBAUK Confidence: 0.981986 Correct
Actual: YNH26 Decoded: YNH26 Confidence: 0.977672 Correct
Actual: YT4TZ Decoded: YT4TZ Confidence: 0.977199 Correct
Actual: Z69AB Decoded: Z69AB Confidence: 0.978042 Correct
Actual: ZXTBV Decoded: ZXTBV Confidence: 0.990738 Correct
Actual: ZY2BH Decoded: ZY2BH Confidence: 0.987960 Correct
Elapsed time is 7.277644 seconds.
Character Accuracy: 1.000000
HIP Accuracy: 1.000000
Average confidence: 0.959318
Minimum confidence: 0.679171