CHARACTER RECOGNITION OF HANDWRITTEN HEBREW USING STRUCTURED ARTIFICIAL WARPING

Avi Bleiweiss
Intel Corporation
Santa Clara, California, USA
avi.bleiweiss@intel.com

ABSTRACT
We present a novel, high performance character recognition system for handwritten Hebrew scripts. Specifically, we explore a technique that bootstraps from an extremely limited hand-made seed of examples, to orderly sample a much amplified synthetic set, at training runtime. This flexible formulation, effectively trades off learning performance vs. memory footprint and scales down to fit the resource limited, small form factor compute devices. We exploit a compressed format of histogram of gradients (HOG) features, trained on a one-against-all SVM classifier, and our evaluation shows a markedly high accuracy rate when compared to other OCR systems.

KEY WORDS
Optical character recognition, histogram of oriented gradients, bootstrapping, artificial warping, support vector machines.

1 Introduction
Optical Character Recognition (OCR) is probably one of the more challenging pattern recognition domains to date. Extensive research for the past three decades proved great strides and led to embracing OCR in disparate application areas. Notably, and not limited to, are reading text from photographs, communication aid to hearing and speech impaired, navigating foreign traffic signage, and postal label reading. But it is in recent years, specifically for handwritten OCR to emerge as one of the principal, alphanumeric input alternatives to the rapidly changing landscape of small compute devices. This mindset shift is apparently mostly attributed to the perceived limited flexibility of the device, embedded virtual keyboard. Alike, both handwritten OCR and speech recognition, with the latter being the most hailed input modality, are nowadays approved substitutes to the tiny key pad [9]. While offering a promising ability to achieve high recognition rates, handwritten OCR must nonetheless perform interactively on pen based devices with constraint compute resources.

In a classical handwritten OCR system, input letters are commonly represented as fixed size images, typically monotone on a relatively benign background. On the other hand, features extracted for each character, tend to yield a high dimensional vector. Since available training data sets are relatively small for non-Latin handwritten languages, avoidance of over-fitting becomes a real challenge. To mitigate this problem, our design bootstraps a small seed of manually coded training examples, and artificially generates many synthetic, character class variants that are studied warps of the initial set. Drawing from the Boltzmann relationship between entropy and probability in statistical thermodynamics, we designate a character in our initial seed a macrostate, from which individual microstates populate a character class, by means of image distortion techniques. Conventionally, entropy of microstates implies disorder or randomness and proved successfully in some state-of-the-art research [2]. Conversely, in our framework, the process of augmenting data sets is both ordered, in selecting a relevant array of rotation and scaling parameters that fits human-computer interaction, and quantitatively controlled to conform to computational resources available on the device.

A key component of a classification system is the feature set selection method that best fits the underlying learning algorithm. Lev and Furst [10] introduced recognition of cursive one stroke, handwritten Hebrew letters, exploiting geometric features. In their system, characters are represented by a chained collection of basic shapes that include lines, arcs and loops. Their methodology embodies an iterative approach to train their model and demonstrated about 85% accuracy. In spite this marked performance, the increase of internal knowledge base, in the event of unrecognized letters, owes partially to position dependent feature that deems a geometric based approach of lesser scalability. In contrast, a pixel based founded method, while intuitive and straight forward, has a $O(n^2)$ vector complexity and often suffers for being over informative, and hence tends to yield under par, runtime performance. Remedy by down sampling the character image improves speed, but excessive blur reduces recognition accuracy [1], [6]. A more robust and effective feature set, is one that attempts to characterize a shape by the distribution of local intensity gradients. Wavelets and edge descriptors fit this feature class well, but notably, the multi grid HOG descriptor [3] stands out as the highest feature scoring in numerous machine learning applications [1], [6], [12]. In our framework, we have implemented an enhanced HOG version, to capture character information.
The main contribution of our research is the development of high performance software for recognizing handwritten Hebrew scripts. Structured and adaptable autogeneration of microstates at runtime, from an exceptionally small base set of hand-crafted, character class macrostates, adds flexibility to gracefully enhance recognition rate, and considerably alleviates the over-fitting concern. To gain robust statistical measure, we have evaluated our system using cross validation on a three-way microstates split, running our experiments in a multi class, one-against-all SVM classifier formation, and reporting the behavior of different kernel types. Next, we provide a brief overview of the Hebrew alphabet in Section 2. Followed by describing our learning workflow end-to-end in Section 3. Then, we discuss quantitative results and analysis of our experiments in Section 4, and conclude with future prospect remarks in Section 5.

2 Hebrew Alef Bet

Hebrew is probably one of the oldest world known languages, dating back to as early as the first millennium B.C. From a descendant of the Aramaic script with 22 original consonants and no vowels, it has evolved into its more modern Square script, adding five more word termination letter variants, thus making the set to total 27 symbols (Figure 1). Later, a Hebrew Cursive script type for free hand writing has been introduced. Cursive letters (Figure 1) are more circular shaped and often vary markedly from their square analogs. The Hebrew alphabet, also known as Alef Bet, has one case for each the square and the cursive styles. It is written horizontally, from right to left and characters are disconnected. The writing system for Hebrew is abjad, a consonant only script, letting the reader supply the appropriate vowels. Optionally, diacritic marks, a combination of dots and lines placed above or below the letter, represent the syllabic onset and indicate variations in the pronunciation of the consonants. Though most texts appear with no diacritic marks, also known as Nikkud, they are commonly found in poetry and children books, to aid early adapters of the language.

In this research, our main goal concerns the recognition of a random blend of the Hebrew square and cursive scripts, often presents itself in free form handwriting. The presence of highly identical shaped symbols, some differ only so slightly in scale, and similarities between word terminating and non-terminating letters, pose an added challenge compared to Latin script recognition. Yet, for numerals, the Hebrew script abides by Latin symbols, and hence state-of-the-art, digits recognition subsystems, are fully reusable and orthogonal to our design. Note that the identification of Nikkud diacritic marks is outside the scope of this work.

3 Learning Workflow

The basic setup for our recognition system proceeds in several stages. With no publicly available, standard data sets of handwritten Hebrew, we start by hand coding images for a basic set of letter macrostates, one for each of the symbols in the square and cursive scripts (Figure 1). Then, for each character macrostate, we artificially produce many microstates that inherit from their parent macrostate, by applying controlled warp transformation. Each microstate in a character class follows a HOG feature set extraction, provided to train a classifier for recognition. We now describe each of the learning steps in more detail.

3.1 Macro and Micro States

Scanning paper sheets of handwritten characters is the more intuitive alternative to obtain images for the unique 54 Hebrew letters. But this tends to incur excess processing cost of character block splitting, and the overhead for removing scanner irregularities [6]. Rather, to faithfully simulate a single pen interface modality, for text input to a device, we use the pencil tool in Microsoft Paint, to draw free form black lines over a white background. This simple scheme merits full user control in emitting a single grayscale image, per letter, and more importantly, aliasing is implicitly retained in rendering a letter, as clearly seen in Figure 1. Our captured image resolution is 162 by 162 pixels, with the letter centered in the middle, providing sufficient margins to accommodate characters extending both upwards and downwards, explicitly for word terminating
characters. Once captured in our system, macrostate images are subsampled down to an internal canonical 128 by 128, 2D array of 8 bit intensities.

Figure 2. Artificially warped, handwritten Hebrew Alef character: square script (a) and cursive script (b). Rotation and scale parameters are limited to an applicable set.

To spawn microstate instances, we use image warping. Given a 2D macrostate image \(f(x,y)\), and a coordinate system transform \((x',y') = h(x,y)\), we compute a transformed image \(g(x',y') = f(T(x',y'))\). Each pixel of the source image is mapped onto a corresponding position in the destination image \((x',y') = T(x,y)\). Where \(T\) is a 3 by 3 affine transformation matrix, though limited in our design to rotations around the center of the letter, followed by uniform scaling. Furthermore, intensity to neighboring pixels is distributed by applying either nearest neighbor or splat filters.

For character recognition, reasonable distortion parameters to macrostate images are practically guarded by human strokes. To produce tangible resemblance of handwriting variability across individuals, we set the span for rotation angle \(\alpha \in \{-30, +30\}\), in increments of one degree. And for each angle \(\alpha\), the range of the scaling factor \(s \in \{1.0, 0.1\}\), in decrements of 0.1. This gives us 600 microstates per macrostate class, in a data set that combines a total of 32,400 images, each of 128 by 128 pixels. Figure 2 shows artificially warped variants of the handwritten Alef macrostate, in both the square and cursive scripts, for a fixed scale.

### 3.2 HOG Features

The construction of histogram of oriented gradients (HOG) features, proved particularly effective in distributing local image intensity, without any prior knowledge of an absolute, physical pixel location. In using HOG features, Dalal and Triggs [3] demonstrated an order of magnitude reduction in false positive rates, relative to a Haar wavelet based system, for the purpose of human detection in scene images. Similarly, Felzenszwalb et al. [5] explored dimensionality reduction of the HOG feature vector using principal component analysis (PCA), with little to no observed performance loss.

Here, we briefly formalize the HOG algorithm as used in our framework. Given a gray scale, single component image \(I(x,y)\) that depicts each of the Hebrew letter microstates, let \(r(x,y)\) and \(\theta(x,y)\) be the magnitude and orientation of the intensity gradient at pixel \((x,y)\), respectively. Gradients are computed using finite difference filters \([-1,0,+1]\) with no smoothing, and the gradient orientation is uniformly discretized into \(p\) values, in the \(0^\circ -180^\circ\) range. Concretely, we define a feature map \(F(x,y)\) that represents a sparse histogram of gradient magnitudes, per pixel. Let \(b \in \{0,\ldots,p-1\}\) span over orientation channels with \(B(x,y)\) the constant insensitive value, then the feature vector per channel at each pixel is defined as

\[
F(x,y)_b = \begin{cases} r(x,y) & \text{if } b = B(x,y) \\ 0 & \text{otherwise} \end{cases} \quad (1)
\]

Next, we subdivide the image extent into a dense grid of non-overlapping cells, each a square pixel region of side length \(k\). Pixel based features of a cell, are summed and averaged to form a cell based feature map \(C\), leading to a marked footprint reduction of the feature vector that models a letter microstate. Rather than simply mapping each pixel to a unique cell \((x/k, y/k)\), a pixel contributes to the feature vector of its four neighboring cells, a block, using bilinear interpolation, weighted by the distances from pixel \((x,y)\) to the boundaries of its surrounding block (Figure 3). This provides some invariance to small local intensity deformations, and to further improve invariance to gradient gain, Dalal and Triggs [3] used four different normalization factors \(N_{\delta,\gamma}(i,j)\) to a vector \(C(i,j)\) with \(\delta, \gamma \in \{-1, +1\}\)

\[
N_{\delta,\gamma}(i,j) = (\|C(i,j)\|^2 + \|C(i + \delta, j)\|^2 + \|C(i, j + \gamma)\|^2 + \|C(i + \delta, j + \gamma)\|^2)^{\frac{1}{2}} \quad (2)
\]

Each factor measures the gradient energy in a square block of four cells containing pixel \((x,y)\). The final HOG feature map is obtained by chaining the results of normalizing the cell based feature map \(C\), with respect to each factor, followed by truncation [3].

The input for extracting a HOG feature vector is a 128 by 128 gray-scale, pixel array. In our implementation, we use a default HOG cell of 8 by 8 pixel region, and a block constitutes of 2 by 2 cells, or 16 by 16 pixel area (Figure 3). Each microstate image is therefore composed of 16 by 16 cells. This leads to a normalized feature vector of \(p = 9\) linear orientation channels, each viewed as a matrix of 16 by 16 float elements. Or, an aggregate 2,304
vector dimensionality, considerably reduced compared to a naïve pixel based vector of 16,384 elements. Still, the constructed HOG feature vector is fairly sparse and often contains many zero elements, leading to a suboptimal storage format. To mitigate this shortcoming, we compress the feature vector into a list of index-non-zero-value pairs, further paring down feature storage space by an average factor of about 4. Sustaining this appreciable system value implies a classifier choice that accepts a matching feature vector interface.

![Figure 3. Microstate subimage depicting cell partitions and overlapping 2 by 2 cell blocks (in gray). Also showing bilinear interpolation weights for pixel feature contribution to four neighboring cells.](image)

### 3.3 SVM Classifier

OCR systems are commonly identified as a bag-of-features representation that can be easily trained using a discriminative classifier, such as support vector machines, SVM. Respectively, we pass the sparse HOG feature vectors, extracted from each of the microstate images, to an SVM classifier for both training and for testing the learned model with new examples. For our classifier, we selected SVM-Light [8] for its robust, large-scale SVM training, and have implemented a C++ wrapper on top, to seamlessly communicate with our recognition software components. SVM-Light core competency, of effectively rendering compact format of sparse vector instances, appeals highly to our design. Furthermore, we extended our performance study to experiment with a broader SVM set of linear, polynomial, radial basis function (RBF or Gaussian) and sigmoid kernel methods (Table 1).

Our classification scenario constitutes an $M$-class problem that we need to decompose into a series of two-class binary problems, using one-against-all method [11]. With 54 character macrostates, we train a total of 54 SVM models, each separating one macrostate from the rest. The $i$-th SVM trains microstates in the $i$-th macrostate labeled as ground-truth true, and microstates of the remaining macrostates are labeled false. At the classification step, a development or a test microstate $x$ belongs to macrostate $i$ that produces the largest value of the hyper-plane distance in feature space, formulated as:

$$
\text{argmax}_{i=1,...,M} (\omega_i^T \text{HOG}(x) + b_i) .
$$

Where weight $\omega \in \mathbb{R}^m$, with $m$ matching our HOG vector dimensionality, and $b$ being a scalar. Note that our software controls the one-against-all outer loop, invoking SVM-Light for training repeatedly 54 times, once for each macrostate vs. rest configuration.

### 4 Evaluation

To evaluate our system in practice, we have implemented a Direct2D imaging application that reads raw, and loads scaled macrostate images, each of 128 by 128 pixels, into our recognition library. We use the holdout method with cross validation to rank the performance of our system. Formally, our library sets up one of random, 10-fold and leave-one-out resampling modes, and each character macrostate becomes a three-way data split of microstates, with train, development and test sets, owning 60/20/20 percent share, respectively. The development set role is primarily for tuning the classifier parameters, and specifically to select the optimal performing SVM kernel type. Recall that cross validation involves training, development and testing the SVM classifier repeatedly, each time with a different section as the held-out test set.

Unless stated otherwise, the results we recorded concern linear SVM. In spite the inherent unbalanced labeling in one-against-all training, our 54 class system consistently shows low average classification error of 1.54%. Figure 4 plots Precision-Recall, interpolated and uninterpolated curves of our system and we report a commensurate area under curve (AUC) performance of 0.39. Whereas the effect of varying the number of microstates per macrostate on average accuracy, is further captured in Figure 5. By fixing the warp scale factor and allowing rotation angles the full specified extent, accuracy climbs as a function of

![Figure 4. Interpolated (solid) and uninterpolated (dashed) Precision vs. Recall curves, extracted from training 54, one-against-all binary classifier models.](image)
Table 1. Basic kernel functions of a Support Vector Machine (SVM) classifier.

<table>
<thead>
<tr>
<th></th>
<th>Linear</th>
<th>Polynomial</th>
<th>RBF</th>
<th>Sigmoid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((x_i^T x_j))</td>
<td>((x_i^T x_j + \gamma)^d)</td>
<td>(\exp\left(\frac{</td>
<td>x_i - x_j</td>
</tr>
</tbody>
</table>

Figure 5. Accuracy of classifying development microstates, plotted as a function of increasing training microstates; shown for a fixed warp scale factor and full extent of rotation angles.

Figure 6. Classification scores comparing separation behavior of linear vs. RBF kernel types.

Increasing the number of synthetic warps, settling at about 99.1% around 50 microstates per macrostate, for a total of 2700 system microstates.

For computational running time, we observed a rather large disparity in kernel selection. With polynomial and sigmoid methods slower by about 5X, and an adverse impact of 50X speed drop for RBF, compared to linear kernel (Table 2). Additionally, Figure 6 contrasts separation behavior of linear vs. Gaussian kernels. For a held-out set of 11 test microstates, linear SVM yields a close to a solid score line of +1 and a clean inter-class transition, whereas RBF shows inexplicable inconsistency, leading us to mainly use the linear kernel. Finally, Figure 7 shows a typical separation spread in one, one-against-all iteration, with a noted distance span from -1 to -7, for false labels.

We compared recognition performance of our synthetic dataset approach against manual, hand coded, state-of-the-art OCR systems. We chose a set of non-Latin scripts of isolated characters, including Bangla [13], Urdu [7], Chinese [4], and Kanji [14] languages. Key system parameters and average recognition rate for each, are depicted in Table 3. For data acquisition, we observed the use of pen tablets and scanned documents, while we used a mouse based tool. Macrostates for Bangla and Urdu are comparable to Hebrew, a few tens, whereas Chines and Kanji deploy thousands of classes. For manual systems, microstates are true human writers and amount to a few hundreds for Bangla and Urdu, inline with our system, yet for Chinese and Kanji, number of writers appears low statistically, only a few tens. Features wise, image based perform slightly better than geometric, and our higher resolution microstate coupled with HOG, edges moment invariants (MI).

5 Conclusion

In this paper, we have demonstrated an effective handwritten Hebrew, character recognition system that is trained with an artificially produced data set. With position agnostic HOG features, our experiments prove marked classification rates for linear SVM, comparable to manual-built systems. While conventional intuition has focused on developing hand coded data model and extensively relying on prior knowledge, our structured and automated approach, provides a more scalable solution and has the potential to achieve compelling performance on small form factor computation devices, with limited storage resources. Finally, we envision our software to be incorporated in emerging application domains, such as scene OCR systems that de-
Table 2. Kernel relative runtime performance, normalized to linear SVM.

<table>
<thead>
<tr>
<th></th>
<th>Linear</th>
<th>Polynomial</th>
<th>RBF</th>
<th>Sigmoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>1</td>
<td>0.21</td>
<td>0.02</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Table 3. Artificial against manual training set construction - system parameters and average recognition performance.

<table>
<thead>
<tr>
<th>Language Script</th>
<th>Data Acquisition</th>
<th>Macrostates</th>
<th>Microstates</th>
<th>Dataset Size</th>
<th>Feature Extraction</th>
<th>Classification</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bangla</td>
<td>Pen Tablet</td>
<td>50</td>
<td>490</td>
<td>24,500</td>
<td>Substroke, Geometric</td>
<td>HMM</td>
<td>87.7</td>
</tr>
<tr>
<td>Urdu</td>
<td>Scanned Document</td>
<td>46</td>
<td>800</td>
<td>36,800</td>
<td>Image (60x60), MI</td>
<td>SVM</td>
<td>93.6</td>
</tr>
<tr>
<td>Chinese</td>
<td>Scanned Document</td>
<td>6763</td>
<td>60</td>
<td>405,780</td>
<td>Shape, Spatial Modeling</td>
<td>Fuzzy Inference</td>
<td>88.4</td>
</tr>
<tr>
<td>Kanji</td>
<td>Pen Tablet</td>
<td>1378</td>
<td>68</td>
<td>93,704</td>
<td>Substroke, Geometric</td>
<td>HMM</td>
<td>92.0</td>
</tr>
<tr>
<td>Hebrew</td>
<td>Synthetic</td>
<td>54</td>
<td>600</td>
<td>32,400</td>
<td>Image (128x128), HOG</td>
<td>SVM</td>
<td>95.6</td>
</tr>
</tbody>
</table>

tect text regions in a photo, performs character segmentation, and pass a set of character images for classification, to be carried out by our system.

Acknowledgement

We would like to thank the anonymous reviewers for their constructive and helpful feedback.

References


