Handwritten Character Recognition in Ancient Manuscripts

Peter Kajenski

***Abstract*—The eigenface method is a technique that has been widely used for facial recognition algorithms. The method relies on the use of a one-dimensional Singular Value Decomposition (SVD), but recently it has been argued that a two dimensional SVD would be more effective. In this study, both methods were applied to a handwritten character recognition problem. The two methods yielded similar accuracy rates, but in some applications the two-dimensional SVD may offer some computational advantages.**

***Index Terms*—Singular Value Decomposition, handwritten character recognition, Support Vector Machines**

I. Introduction

Facial recognition is currently an active area of research, and one technique that has frequently been used for this application is the eigenface method [1]. Eigenfaces are created from a set of two-dimensional training images that have been vectorized, and then a Singular Value Decomposition (SVD) is applied to the data set; the eigenvectors associated with the most significant principal components are retained, and thus comprise the "eigenfaces." A Two-Dimensional Singular Value Decomposition (2D-SVD) algorithm has recently been described in the literature, which is a generalization of the well-known SVD [2]. In [2] the properties of the 2D-SVD were defined, and it was demonstrated that it can be used to improve the accuracy of a facial recognition algorithm, as well as offer better compression of image data sets than would be the case with the conventional SVD. Given that written texts are two-dimensional in nature, it is only natural to ask whether a 2D-SVD would offer any benefits for the handwritten character recognition problem.

The problem of handwriting recognition have been studied for many years [3]–[9], and several such algorithms have been incorporated in commercialized products [10]. Handwriting recognition can be broadly classified as either an "online" problem, whereby the writer's pen strokes can be used to assist in the classification process, or an "offline" problem, where only the written characters themselves are used. Of particular interest in this study are ancient manuscripts which, of course, were written by hand. The first step to analyzing such documents is to implement an algorithm that can record the individual characters, and put them into a digital form that a machine can manipulate.

During the period from about 400 BCE until about 600 CE, many Western documents were written in a style known as "scriptura continua" [11], which roughly translates to "continuous writing." This style of writing is typically characterized by the use of uncial letters (all upper case), with no spacing between the words, and with little or no punctuation indicating the beginning or end of a given sentences.

P. Kajenski is with the Raytheon Company, 150 Lowell Street, Andover, MA, 01810 USA e-mail: kajenski@raytheon.org

Manuscript received December 14, 2012

For this particular effort the Codex Sinaiticus manuscript was chosen, as it has a number of features that make it convenient for study. It is believed that Codex Sinaiticus was written somewhere between the years 325 and 360 CE, and it is perhaps a little unusual in that while it was written in the continua scriptura writing style, the characters are organized in distinctive columns and rows [11]. Codex Sinaiticus is currently held in four separate libraries, which are in entirely different countries. In 2009 the components of the document were reunited in a virtual manner, and it is now available on the Internet [12], [13]. The website provides photographs of the actual manuscript itself, as well as a transcription of the Greek text, thus providing an experts opinion on the identity of each individual character.

Character recognition algorithms can provide scholars with a new set of very powerful techniques for studying ancient documents. For example, it may offer the ability to ascertain something about the scribes who drafted the manuscript. It is believed that at least three scribes participated in writing Codex Sinaiticus, although some contend that a fourth was also present, and there were also numerous corrections to the manuscript made by later editors [11]. Most character recognition algorithms are designed to be able to recognize a given letter, independent of who the author may have been. In this kind of application it would actually be more helpful to have an author-dependent character recognition algorithm, one that could distinguish between the letter "ω" written by two different scribes. Such a tool could help estimate how many writers helped produce the document, who wrote what sections, and perhaps even suggest how many different editors may subsequently altered it. Further, handwriting analysis tools might even permit scholars to determine what other extant manuscripts a given scribe might have worked on.

In Section II a brief overview of the SVD algoirhtms are given. In Section III the application of the two SVD algorithms to obtaining the principal components from the character data will be described, and experimental results using a Support Vector Machine (SVM) will be presented. Finally, in Section IV conclusions will be given.

II. Singular Value Decompositions

A. The 1D-SVD

We begin by describing the application of a one-dimensional SVD for character recognition. This generally entails converting a given image into a single vector. Starting with N images of size $h \times w$, one creates a vector of size, $D = hw$, and each is placed into a set $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_N\}$. A zero-mean vector is computed as $\Phi_i = \Gamma_i - \Psi$, where:

$$\Psi = \frac{1}{N}\sum_{i=1}^{N}\Gamma_i \tag{1}$$

The one dimensional covariance matrix, $\boldsymbol{C}$, is defined as:

$$\boldsymbol{C} = \frac{1}{N}\sum_{i=1}^{N}\Phi_i\Phi_i^T = \boldsymbol{B}\boldsymbol{B^T} \tag{2}$$

where $\boldsymbol{B} = \{\Phi_1, \dots, \Phi_N\}$. Then $\boldsymbol{C}$ can be decomposed using a SVD as:

$$\boldsymbol{C} = \boldsymbol{U} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{V^*} \tag{3}$$

where $\boldsymbol{U}$ and $\boldsymbol{V^*}$ are unitary matrices, and $\boldsymbol{\Lambda}$ is a diagonal eigenvalue matrix:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdot & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix} \tag{4}$$

B. The 2D-SVD

The description of the 2D-SVD presented here largely follows that in [2]. We start by defining averaged row-row and column-column covariance matrices:

$$\boldsymbol{F} = \sum_{i=1}^{n} \left(\boldsymbol{A}_i - \bar{\boldsymbol{A}}\right)\left(\boldsymbol{A}_i - \bar{\boldsymbol{A}}\right)^T$$
$$\boldsymbol{G} = \sum_{i=1}^{n} \left(\boldsymbol{A}_i - \bar{\boldsymbol{A}}\right)^T\left(\boldsymbol{A}_i - \bar{\boldsymbol{A}}\right) \tag{5}$$

where $\bar{\boldsymbol{A}} = \frac{1}{n}\sum_i \boldsymbol{A}_i$. We perform a 1D-SVD on both $\boldsymbol{F}$ and $\boldsymbol{G}$, to create a matrix $\boldsymbol{U}_r$ which contains the r eigenvectors of $\boldsymbol{F}$, and a matrix $\boldsymbol{V}_s$ corresponds to the c eigenvectors of $\boldsymbol{G}$. Accordingly, we have:

$$\boldsymbol{F} = \sum_{\ell=1}^{r} \chi_\ell \boldsymbol{u}_\ell \boldsymbol{u}_\ell^T, \quad \boldsymbol{U}_r \equiv (\boldsymbol{u}_1, \dots, \boldsymbol{u}_r);$$
$$\boldsymbol{G} = \sum_{\ell=1}^{c} \zeta_\ell \boldsymbol{v}_\ell \boldsymbol{v}_\ell^T, \quad \boldsymbol{V}_c \equiv (\boldsymbol{v}_1, \dots, \boldsymbol{v}_c); \tag{6}$$

where χ_ℓ and ζ_ℓ are the ℓ^{th} eigenvalues associated with $\boldsymbol{F}$ and $\boldsymbol{G}$, respectively. We define an eigenmatrix for a given image, $\boldsymbol{A}_i$ as:

$$\boldsymbol{M_i} = \boldsymbol{U_r^T A_i V_c}, \quad i = 1, \dots, n, \tag{7}$$

where $\boldsymbol{M}_i$ forms a matrix of eigenvalues. Note that unlike in the 1D-SVD case where $\boldsymbol{\Lambda}$ is a diagonal matrix, $\boldsymbol{M}_i$ is not constrained to be diagonal. The $\boldsymbol{A}_i$ matrix can then be recovered by:

$$\boldsymbol{A}_i = \boldsymbol{U_r M_i V_c^T} \tag{8}$$

III. Experiments

Pages from the book 1 Timothy Chapter 1 in the Codex Sinaiticus manuscript were downloaded from the Internet, and the color images were submitted to a threshold to convert them into binary data sets. A bounding box routine was used to identify potential characters. Characters that overlapped each other were excluded from the set, but any fragments from other characters that may have "leaked" into the bounding box were allowed, and noisy and faded images were also included. The frame size for each character was 67×76 pixels, and each character was centered in the frame. A data set of approximately 1,800 character images were collected and used in these experiments.

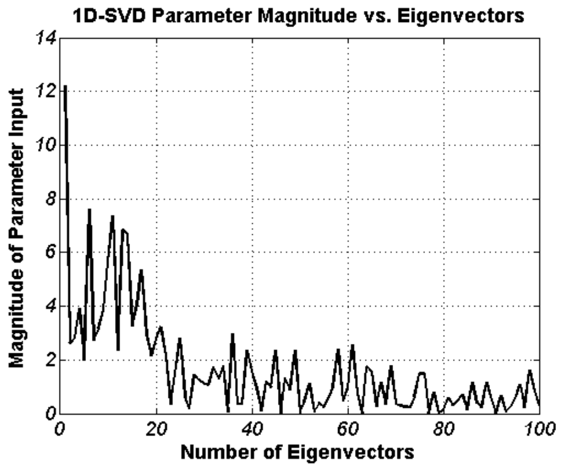


Fig. 1. The computed vector $\widehat{\Phi}$ for the letter Λ.

The classification was performed with the LIBSVM Support Vector Machine routine [14], which was configured in a One-Versus-All (OVA) manner [15]. Each character image was made up of 5,092 pixels, and the 1D-SVD and 2D-SVD algorithms were used to reduce the images to their principal components. For the 1D-SVD, the approach used to pre-process the images was similar to the method used for facial recognition. A subset of eigenvectors, $\widehat{\boldsymbol{U}}_g = \{\boldsymbol{u_1}, \dots, \boldsymbol{u_g}\}$, associated with the largest g eigenvalues were used to convert the vectorized character image into the test-data space as:

$$\widehat{\Phi}_n = \widehat{\boldsymbol{U}}_g^T \Phi_n \tag{9}$$

Similarly, transforming the character images into the test-data spaces with the 2D-SVD involved using a subset of the eigenvectors computed in (5), $\tilde{\boldsymbol{U}}_p = \{\boldsymbol{u_1}, \dots, \boldsymbol{u_p}\}$, $\tilde{\boldsymbol{V}}_q = \{\boldsymbol{v_1}, \dots, \boldsymbol{v_q}\}$, with $\quad p \le k, \quad q \le s$. For the n^{th} character image, $\boldsymbol{A}_n$, this is given as:

$$\tilde{\boldsymbol{M}} = \tilde{\boldsymbol{U}}_{\boldsymbol{p}}^{\boldsymbol{T}} \boldsymbol{A_n} \tilde{\boldsymbol{V}}_{\boldsymbol{q}} \tag{10}$$

Figures 1 and 2 show representative examples of the magnitudes of $\widehat{\Phi}$ and $\tilde{\boldsymbol{M}}$ for the letter Λ.

The elements in the transformed vector $\widehat{\Phi}_n$, and the matrix terms from $\tilde{\boldsymbol{M}}$ were used as inputs into the SVM. The problem at hand now is how to make a fair comparison of the performance of the 1D-SVD and 2D-SVD. It was reasoned that if these algorithms were to be used in a practical system, the end user wouldn't be at all concerned with how long to takes to train the system. Rather, what would matter would be the time needed to classify all of characters in the test data. Accordingly, it decided to compare the classification accuracy of each, for a given average execution time. This approach immediately gave rise to an observation regarding the computational speed of the two pre-processing algorithms, which was found experimentally to be proportional to the number of multiplication steps. As noted above, the images were 67×76 pixels, and so if we used the eigenvectors associated with the g highest eigenvalues, then the number of multiplication steps needed for the application of (9) to the

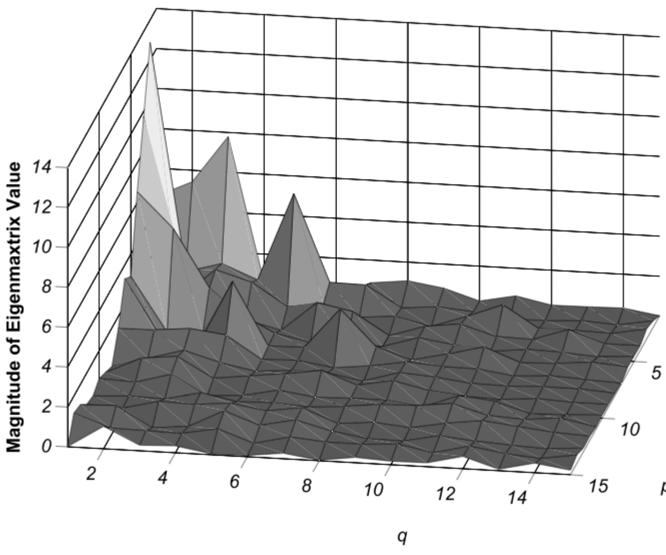


Fig. 2. The magnitude of the matrix $\tilde{\boldsymbol{M}}$ computed for the letter Λ.

TABLE I
CLASSIFICATION USING A LINEAR KERNEL SVM

1D-SVD			2D-SVD		
Size	Time (s)	Accuracy (%)	Size	Time (s)	Accuracy (%)
16	0.6040	95.46	4×4	0.5942	95.35
20	0.6971	96.01	5×5	0.6933	96.12
25	0.8273	96.23	6×6	0.8276	97.56
33	1.031	96.35	7×7	1.001	97.8

TABLE II
CLASSIFICATION USING A SECOND ORDER POLYNOMIAL KERNEL SVM

1D-SVD			2D-SVD		
Size	Time (s)	Accuracy (%)	Size	Time (s)	Accuracy (%)
12	0.6068	95.76	4×4	0.6101	97.79
21	0.8201	97.79	5×5	0.8207	97.23
26	.9844	97.78	6×6	0.9849	98.45
30	1.097	98.34	7×7	1.1045	98.3

vectorized image would require:

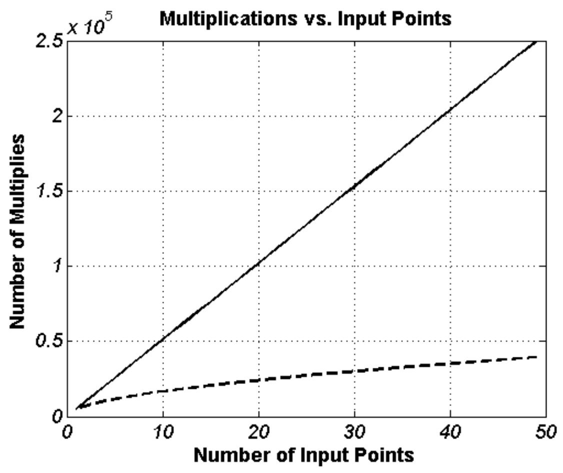
$$N_1 = g \times 5092 \tag{11}$$

In contrast, if we use p eigenvectors for the construction of $\tilde{\boldsymbol{U}}_{\boldsymbol{p}}$, and q for $\tilde{\boldsymbol{V}}_q$, the number of multiplications, N_2, needed to construct $\boldsymbol{M_i}$ would be:

$$N_2 = p \times 67 \times 76 + p \times 76 \times q \tag{12}$$

Figure 3 shows plots of the number of multiplication steps for N_1 (solid line) and N_2 (dashed line), as function of the number of input parameters. Note that these trends depend on the dimensions of the image, and the difference between N_1 and N_2 would be even more pronounced for larger images.

Approximately 800 of the characters were randomly selected to be used in the training set, and the remaining 1000 were used for testing. First, a forward search was conducted to determine the classification accuracy for different combinations and numbers of input parameters. This step was repeated for a linear kernel, a radial basis kernel, as well as higher order polynomial kernels. It quickly became apparent that the best classification accuracy was obtained for the linear



pFig. 3. The number of multiplies for the 1D-SVD (solid line) and the number of multiplies fo the 2D-SVD (dashed line), as a function of the number of input points.

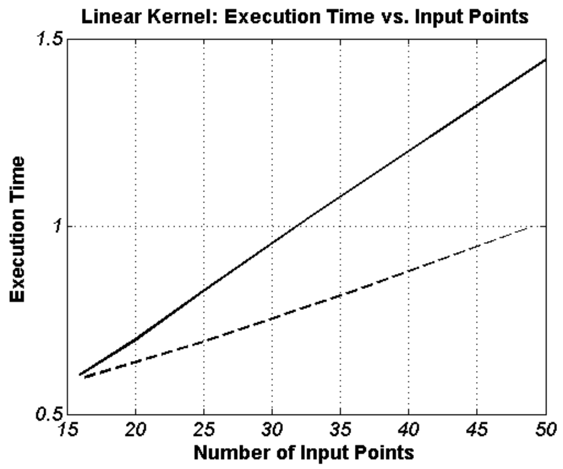


Fig. 4. The execution times for the 1D-SVD (solid line) and the number of multiplies fo the 2D-SVD (dashed line), as a function of the number of input points.

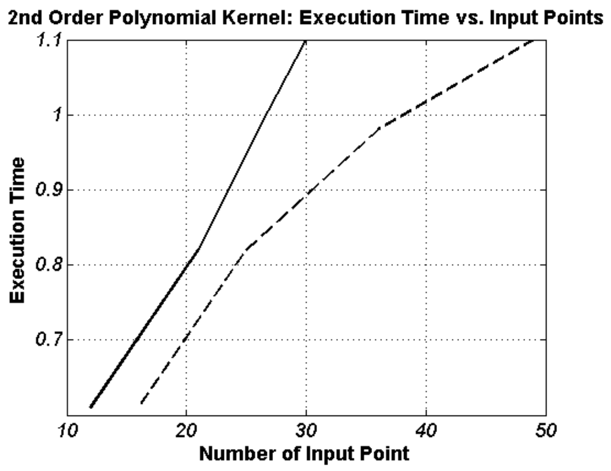


Fig. 5. The execution times for the 1D-SVD (solid line) and the number of multiplies for the 2D-SVD (dashed line), as a function of the number of input points.

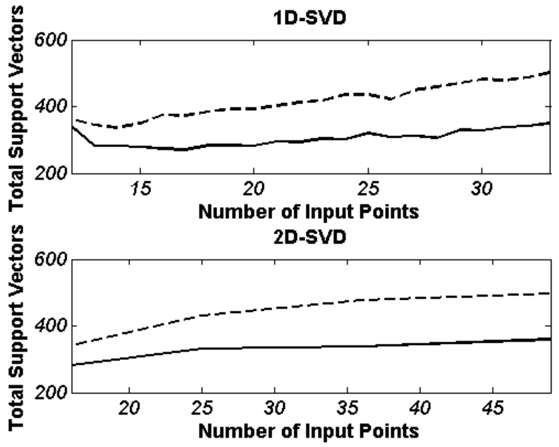


Fig. 6. The total number of support vectors for the one-versus-all classification scheme (20 SVM classifiers). The upper plot shows the cases for the 1D-SVD, with a linear kernel (solid line) and the second order polynomial kernel (dashed line). The lower plot shows the same result for the 2D-SVD.

and second order polynomial kernels. Once the accuracy rates were established, a timing study was conducted in which the classification task was repeated 500 times, and the execution times for all of the runs were averaged. The results from the 1D-SVD and 2D-SVD were then paired according to execution time. The results for the case where a linear kernel was used are shown in Table I, and those for the second order polynomial kernel are shown in Table II. The size parameter indicated for the 1D-SVD represents the first "g" eigenvectors used in (9), and the size indicated for the 2D-SVD represents the first "$p \times q$" eigenvectors from (10). Overall, both methods yield comparable accuracies for the given execution times.

The execution times for the linear kernel SVM were plotted in Figure 4, and it was noted that the trends generally followed the pattern in Figure 3. The execution times for the second order polynomial kernel SVM were plotted in Figure 5, which quite strikingly, did not at all follow the expected trends, suggesting that complexity of the classification algorithm dominates for the higher order kernel. It was noted in [16] that the computational complexity of the SVM is of the order $\mathcal{O}\left(N^2 m\right)$, where N is the number of input points and m is the number of support vectors. This suggests that while some speed advantage may be afforded by the fewer multiplication steps needed for the 2D-SVD, the advantage could be entirely lost if it comes at the expense of an increase in the number of required support vectors. Figure 6 show plots of the total sum of the number of support vectors, for all 20 of the character classifiers in the OVA code, as a function of the number of input points. The upper plot shows the results for the 1D-SVD, with a linear kernel (solid line) and with a second order polynomial kernel (dashed line). The lower plot shows the same result for the 2D-SVD. The number of support vectors evidently increases considerably when going from a linear to a second order polynomial kernel, and generally seems to increase with the number of input points.

To gain a better insight as to the differences in performance of the classifiers, confusion matrices [17] were created for the

TABLE III
CONFUSION MATRIX FOR 1D-SVD, POLYNOMIAL KERNEL

		PREDICTED									
		A	Δ	Γ	H	K	Λ	M	N	O	Θ
	A	113	0	0	0	0	4	0	0	0	0
	Δ	6	13	0	0	0	0	0	0	0	0
A	Γ	1	0	15	0	0	0	0	0	0	0
C	H	0	0	0	27	0	0	0	0	0	0
T	K	0	0	0	0	26	0	0	1	0	0
U	Λ	9	0	0	0	0	15	0	0	0	0
A	M	0	0	0	4	0	0	27	1	0	0
L	N	0	0	0	2	0	0	0	92	0	0
	O	0	0	0	1	0	0	0	0	62	0
	θ	0	0	0	0	0	0	0	0	3	23

TABLE IV
CONFUSION MATRIX FOR 2D-SVD, POLYNOMIAL KERNEL

		PREDICTED									
		A	Δ	Γ	H	K	Λ	M	N	O	Θ
	A	117	0	0	0	0	0	0	0	0	0
	Δ	0	18	0	0	0	0	0	0	0	0
A	Γ	0	0	15	0	0	0	0	0	0	0
C	H	0	0	0	27	0	0	0	0	0	0
T	K	0	0	0	0	27	0	0	0	0	0
U	Λ	2	0	0	0	0	22	0	0	0	0
A	M	0	0	0	2	0	0	30	0	0	0
L	N	0	0	0	2	0	0	0	92	0	0
	O	0	0	0	1	0	0	0	0	62	0
	θ	0	0	0	0	0	0	0	0	3	23

second order kernel case where $g = 12$ for the 1D-SVD, and for the case where $p = q = 5$ for the 2D-SVD, as indicated in Table IV. Note that these tables are abbreviated due to space limitations, and only show half of the characters used in the study; characters that were entirely error-free were not included in these matrices, but were included in the accuracy computations. From these tables it is clear that there was some difficulty with sorting the letters "Λ ", "Δ", and "A." To investigate the nature of these errors, an experiment was run in which the character images were reconstructed using the eigenvectors and eigenmatrices, as shown in Figure 7. The image on the top left is the original image, the top center image is the reconstruction of the image using the first 12 eigenvectors of the 1D-SVD, and the top right image is the reconstruction using (8) with $p = q = 5$. The bottom row shows the case where 30 eigenvectors were used for the 1D-SVD, and a 7×7 eigenmatrix was used for the 2D-SVD. In this latter case it appears that the 1D-SVD does a somewhat better job of reconstructing the original image, though both methods yield similar classification accuracies in the SVM.

IV. CONCLUSION

The 1D-SVD has been used for solving recognition problems for many years, and has proven to be quite effective in many applications. The 2D-SVD has been found to also be effective for handwritten character recognition, and in fact may potentially offer some computational advantages over the 1D-SVD. However, the advantages for the 2D-SVD can be lost if, as a consequence of using it, it imposes a higher computational burden on whatever classification algorithm is used. In the experiments described here the performance of both the 1D-SVD and 2D-SVD were comparable.

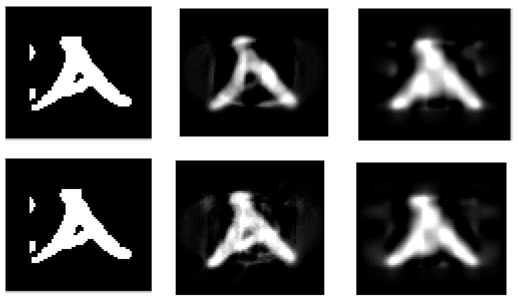


Fig. 7. The figure on the top left is the original image,the top center image is the reconstruction of the image using the first 12 eigenvectors with the 1D-SVD, and the top right image is the reconstruction using (6) with $p = q = 5$.The figure on the bottom left is again the original image,the bottom center image is the reconstruction of the image using the first 30 eigenvectors with the 1D-SVD, and the bottom right image is the reconstruction using (6) with $p = q = 7$.

REFERENCES

[1] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, jun 1991, pp. 586 –591.

[2] C. Ding and J. Ye, "Two-dimensional singular value decomposition (2dsvd) for 2d maps and images," in *Proc. SIAM Int'l Conf. Data Mining*, vol. SDM'05, 2005, p. 3243.

[3] R. Alhajj and A. Elnagar, "Multiagents to separating handwritten connected digits," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 5, pp. 593 – 602, sept. 2005.

[4] T. Artieres, S. Marukatat, and P. Gallinari, "Online handwritten shape recognition using segmental hidden markov models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 2, pp. 205 –217, feb. 2007.

[5] D. Ghosh, T. Dube, and A. Shivaprasad, "Script recognition, a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 12, pp. 2142 –2161, dec. 2010.

[6] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 855 –868, may 2009.

[7] R. Jayadevan, S. Kolhe, P. Patil, and U. Pal, "Offline recognition of devanagari script: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 6, pp. 782 –796, nov. 2011.

[8] J. Park, "An adaptive approach to offline handwritten word recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 920 –931, jul 2002.

[9] J. Zeng and Z.-Q. Liu, "Markov random field-based statistical character structure modeling for handwritten chinese character recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 5, pp. 767 –780, may 2008.

[10] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 63 –84, jan 2000.

[11] B. Metzger and B. D. Ehrman, *The Text of the New Testament: An Introduction to the Critical Editions and to the Theory and Practice of Modern Textual Criticism*. Oxford University Press, 2005.

[12] E. Henschke, "Digitizing the hand-written bible: The codex sinaiticus, its history and modern presentation," *Libri*, vol. 57, p. 4551, 2007.

[13] Z. M. Dogan and A. Scharsky, *Virtual Unification of the Earliest Christian Bible: Digitisation, Transcription, Translation and Physical Description of the Codex Sinaiticus*. European Conference on Digital Libraries, 2008.

[14] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[15] C. L. C. Hsu, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, pp. 415 – 425, 2002.

[16] L. Kaufman, *Advances in Kernel MethodsSupport Vector Learning*, C. B. B. Schlkopf and A. Smola, Eds. MIT Press, 1999.

[17] S. Stehman, "Selecting and interpreting measures of thematic classification accuracy," *Remote Sensing of Environment*, vol. 62, p. 7789, 1997.