20^{th} *Computer Vision Winter Workshop*
Paul Wohlhart, Vincent Lepetit (eds.)
Seggau, Austria, February 9-11, 2015

Cuneiform Character Similarity Using Graph Representations

Bartosz Bogacz[1], Michael Gertz[2] and Hubert Mara[1]

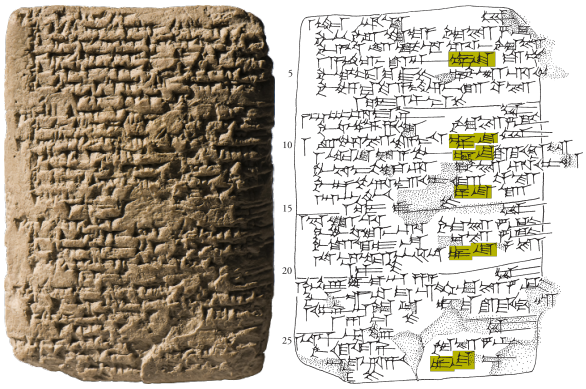
[1]Interdisciplinary Center for Scientific Computing (IWR),
Forensic Computational Geometry Laboratory (FCGL)
[2]Institute of Computer Science
Heidelberg University, Germany
{bartosz.bogacz|hubert.mara}@iwr.uni-heidelberg.de
gertz@informatik.uni-heidelberg.de

Abstract.

Motivated by the increased demand for computerized analysis of documents within the Digital Humanities we are developing algorithms for cuneiform tablets, which contain the oldest handwritten script used for more than three millennia. These tablets are typically found in the Middle East and contain a total amount of written words comparable to all documents in Latin or ancient Greek. In previous work we have shown how to extract vector drawings from 3D-models similar to those manually drawn over digital photographs. Both types of drawings share the Scalable Vector Graphic (SVG) format representing the cuneiform characters as splines. These splines are transformed into a graph representation and extend these by triangulation. Based on graph kernel methods we show a similarity metric for cuneiform characters, which have higher degrees of freedom than handwriting with ink on paper. An evaluation of the precision and recall of our proposed approach is shown and compared to well-known methods for processing handwriting. Finally a summary and an outlook are given.

1. Introduction

Cuneiform tablets are one of oldest textual artifacts comparable in extent to texts written in Latin or ancient Greek. Since those tablets were used in all of the ancient Near East for over three thousand years [22], many interesting research questions can be answered regarding the development of religion, politics, science, trade, and climate change [9]. These tablets were formed from clay and written on by impressing a rectangular stylus [2]. The result is a wedge shaped impression in the clay tablet. The word cuneiform derives from the Latin word "cuneus" *wedge* and "forma" *shaped*.



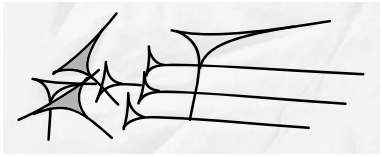
(a) (b)

Figure 1. Cuneiform tablet No. TCH92, G127 [8]: (a) Photograph and (b) its drawing. Six instances of the same two character tuple have been highlighted in yellow. A method for cuneiform character recognition would ideally classify those wedge configurations as highly similar.

There is an increasing demand in the Digital Humanities domain for handwriting recognition focusing on historic documents [20]. Even the recognition of ancient characters sharing shapes with their modern counterparts e.g. ancient Chinese Sutra [14] is a challenging task. For digitally processing cuneiform script there exist only a few recent related approaches like proposed in [6] using geometric features of cuneiform tablets acquired with a 3D-scanner [16].

However, with the aim of building a search tool for cuneiform tablets we have to consider the complexity of cuneiform characters in their de facto standardized

(a)



(b)

Figure 2. The cuneiform character for the syllable "zum" and its drawing.

2D-representation. Figure 1 shows a photograph of a cuneiform tablet and its drawing. Properties like the lack of fixed word length together with a wide variety of infixes, suffixes and prefixes prevents the application of existing machine learning methods on pictographs.

The extraction of the wedge shaped impressions of a cuneiform character is currently being approached by manually tracing a photograph of a cuneiform tablet using tools like Inkscape or by automatically extracting the boundary of a wedge configuration on basis of a 3D-model of the cuneiform tablet [15]. In either case, the result of an extraction is a document in the Scalable Vector Graphic (SVG) format. Figure 2 shows the cuneiform character for syllable "zum" and its drawing. The extraction of the wedges, looking like Ys, is challenging because these wedges are described with splines to retain all the damage and complexity of being written by hand. A clean extraction of the wedges is not sufficient to easily compare cuneiform characters. The configuration (position, orientation, grouping and overlap) and the shape of the wedges varies among different instances of the same character to a degree that requires a sophisticated character model to properly classify such characters.

2. Character recognition in raster images

Virtually all related research uses raster data as input. Word spotting is performed either on segmented lines [4, 24, 10] or on whole documents [18, 17]. The usage of Hidden Markov Models (HMMs) in these approaches circumvents the problem of learning fixed-length features for words or characters by decomposing the document or its lines into smaller features. The observations of the HMM are thin slices of a word, less than a character in width but with the same height as a line. A word is the represented as a succession of hidden states, each emitting a set of word slices. The advantage of this representation is that each slice is a fixed-length feature.

Wshah and colleagues [24] use direction gradients and a set of four intensities as features for a sliding window approach over already segmented lines. The query word is modeled to match a complete line by beginning and ending with filler characters modeling non-keywords to reduce the false positive rate.

To reduce the amount of required training on words lexicon-free handwritten word spotting approach using character HMMs [4] is applied. Then, the training of the HMM classifier only requires a small number of character classes. Just like in the work of Wshah and colleagues [24] filler models are employed, now consisting of a space character and all other character classes, to improve the retrieval precision.

Instead of directly using features extracted from the bitmap data, Rothaker and colleagues [17] use a Bag-of-Features representation with densely sampled Scale Invariant Feature Transform (SIFT) descriptors. These descriptors are then clustered into a dictionary with a limited set of words and quantized onto a regular grid overlapping the top of the document. No preprocessing of the document is necessary because the SIFT descriptors work directly on gray-scale data. A HMM classifier determines the most probable positions for the query word for all possible positions on the aforementioned grid. A segmentation of the document is therefore not necessary.

The work presented by Fischer and colleagues in [5] uses graphs as features to describe characters and measure similarity. Their approach requires an document already segmented into words. Images are first transformed into a color-binary representation and then thinned to one pixel medial axis curves. Graph vertices are created at endpoints, intersections and corner points of the medial axis curves. A HMM classifier is trained on thin slices of these word graphs.

The nature of writing cuneiform script poses a problem for HMM based classifiers. Cuneiform character traces have significantly more foreground-background transitions in the vertical axis than a word written in Latin. Classification with a HMM based approach would necessitate a larger feature

space of thin slices and therefore more training data for robust classification. Training data in the form of traced clay tablets is not readily available.

Furthermore, these approaches assume that word slices are always rigidly in the same order. Wedge shaped impressions, on the other hand, can locally interchange position, both in the vertical direction as well as in the horizontal direction, and yet still describe the same word. Graph based methods are more robust against such changes in topology.

A method for segmentation free word spotting is presented by Almazan and colleagues in [1] that uses exemplary SVMs to train one positive sample versus many negative samples. The document and the query are represented by grid of Histogram of oriented Gradient (HoG) descriptor cells. Training the SVM is done by using slightly translated windows of the query as positive examples. Negative examples are randomly selected windows of the document.

Although this approach does neither require any labeled samples nor a segmented document, the resulting SVMs only work very well on typeset or script written without much variation. Cuneiform text is highly variable in the expression of the wedges due to various factors such as the age of the clay tablet or the nature of the tool being used to impress wedges. An approach is necessary that offers more flexibility with respect to the deformation of the query word.

Howe presents a one-shot word spotting approach in [7] that does not require any training data. Words are binarized and represented as a tree of points connected by spring-like potentials. The document itself is then transformed using the structure of the tree of the query word. Locations where the transformation leads to a local energy maximum are those where the query word can be found.

Leydier and colleagues [12] use basic visual features found in written text to spot words in a document. The first order oriented image gradient is compared in specific image patches of the query word and document, so called zones of interest, to asses the similarity of words. The zones of interest themselves allow for an initial rough matching. The query word zones of interest are aligned to those locations of the document that share the same shape.

Both approaches do not assume a specific writing direction nor do they require segmented documents, but they do not allow for enough variation in character shape. The approach presented by Leydier and colleagues in [12] allows changes in positioning of the structuring elements using elastic cohesive matching, but does not allow for sufficient variability in the structuring elements themselves. Cuneiform wedges can be slightly rotated or elongated for aesthetic purposes and deform the zones of interest enough to preclude a successful match. Conversely, Howe [7] allows local variability but does not account for swapped characters.

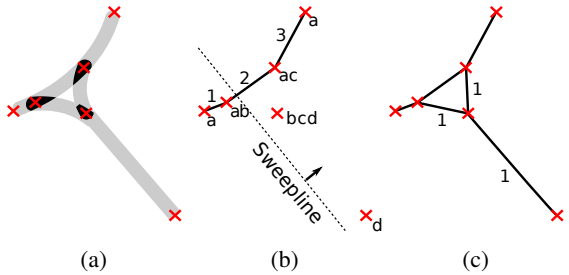


Figure 3. A cuneiform character in spline representation. (a) Closed spline paths forming strokes (gray) are pairwise intersected (black). (b) The vertices (marked X) of a stroke are first ordered from bottom right to top left. (c) Edges of the final graph are created by connecting the vertices. The sequence is indicated by the numbers.

3. From splines to graphs

Before any graph matching methods can be applied, the cuneiform characters first need to be transformed into well-formed graphs. Further, we segment cuneiform characters manually since cuneiform script has no visual word boundaries that would allow for automatic segmentation. The recognition of a word in the Assyrian language requires the knowledge of its grammatical case and context in which it is used. The clusters of wedges that have been manually segmented do not necessarily represent distinct characters in the original text. Nevertheless, they will be referred to as character in the following. A character consists of a set of strokes. Each stroke is a geometric shape bounded by a closed path of splines. These strokes are drawn in an vector graphic editor by assyriologitsts tracing a cuneiform tablet.

All strokes are pairwise intersected to create a set of key points. Most strokes are drawn in a way that there is only one closed unambiguous area of intersection. A vertex is placed at the center of such an area. Since more than two strokes can intersect in the same vicinity, the set of key points is pruned so that no two points are closer than some threshold ϵ. Figure 3 and Figure 4 illustrate this process. We set

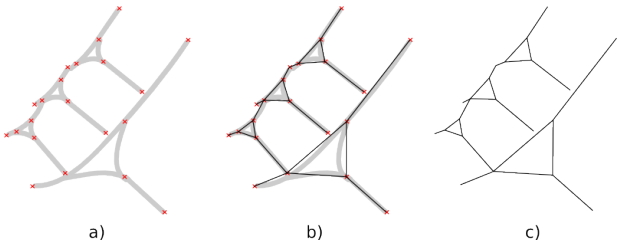


Figure 4. This figure shows the steps from Figure 3 applied to a complex set of strokes representing a character. In the majority of cases a small intersection of the arms is not semantically relevant.

$\epsilon = 1$. The threshold ϵ is expressed in tenths of a millimeter. The typical line height of cuneiform script is 5 millimeters. The choice of ϵ did not matter in our experiments as long as it was two orders of magnitude smaller than a character. We add the two endpoints of each stroke to the set of key points. Endpoints are calculated by finding the two most distant points on the spline enclosing a stroke.

The extracted points are not yet connected by edges. There is no inherent order of key points on a stroke between its endpoints. All points belonging to a stroke, that is its endpoints and points created with intersections with other strokes, are order geometrically and connected in sequence. In rare cases this may create incorrectly connected points if the stroke is slightly curved and the geometric ordering does not correctly represent the curvature of the stroke. Only few instances have been observed where this is the case.

4. Graph Similarity

After transforming the cuneiform characters from a collection of strokes into a graph representation, the characters can now be compared in similarity using common graph matching methods. Since small differences in position and orientation in wedge impressions do not change the meaning of a character, we assume that the graph topology is sufficiently descriptive to measure the similarity of cuneiform characters. We present three graph matching methods and extend each to work on the Delaunay triangulation of the cuneiform character graph. The Delaunay triangulation is used to catch big structural differences, significant translation or rotation of wedges, or wedge impressions in distinct graph components that do not modify the topology of the cuneiform character graph.

4.1. Weisfeiler-Lehman graph kernel

The graph kernel presented by the authors in [19] is an extension of the graph isomorphism test introduced by Weisfeiler and Lehman [23]. The kernel works by by counting how many subtrees the two graphs being compared share.

Each vertex of a cuneiform character graph is assigned a unique label. (Using the same label for each vertex results in significantly worse results.) On every iteration of the algorithm each vertex label is expanded with the labels of adjacent vertices. To increase computational performance all vertex labels can be converted into a shorter representation using hashing. Adjacent vertex labels have been, in turn, extended in an earlier iteration by their adjacent vertex labels. The label of each vertex is therefore an enumeration of a subtree rooted at this specific vertex.

The label, and therefore the subtree, contains multiple repetitions of itself since the root vertex is adjacent to each of its adjacent vertices. This behavior is called *tottering* [19] and degrades the quality of the labels and the quality of the similarity metric.

The similarity of two graphs G_A and G_B and their label sets N_A^k and N_B^k is the count of matching labels at iteration k. We denote the labels of N_A^k and N_B^k with e and f.The graphs are considered to be highly similar if most of the subtrees extracted from either graph are present in both graphs. $\delta(e, f)$ is the Kronecker delta, that is, $\delta(e, f) = 1$ if $e = f$, and 0 otherwise. We perform four relabeling iterations $n = 4$. More relabeling iterations ($n = 10$) did not result in better classification performance.

$$K = \sum_{k}^{n} \sum_{e \in N_A^k} \sum_{f \in N_B^k} \delta(e, f) \qquad (1)$$

4.2. Spectral decomposition

The spectral decomposition [3] of a graph has a variety of applications in the field of graph matching [13]. A graph is decomposed by computing the eigenvectors and eigenvalues of its adjacency matrix that has been at first converted into a normalized Laplacian matrix.

The resulting multi-set of eigenvalues and eigenvectors have many interesting properties [3] and are often used for clustering where the multi-set of eigenvectors can be used to find a nearly minimal cut. Additionally, the spectral decomposition of a graph

is used as an approximation for the random walk kernel on a graph [21]. We make use of the property of the spectral decomposition that two identical graphs have the same multi-set of eigenvalues. Such graphs are called to be *isospectral*. Small changes to those graphs result in only small changes to the multi-set of eigenvalues.

We compute the normalized Laplacian matrix L with components l_{uv} from the adjacency matrix A of a graph G with vertices u, v and vertex degrees d_u, d_v. L has the eigenvectors ϕ_i and the multi-set of eigenvalues λ_i.

$$\begin{aligned} L =& [l_{uv}] \\ l_{uv} =& \begin{cases} 1, & \text{if } u = v \text{ and } d_u \neq 0 \\ -\frac{1}{\sqrt{d_u d_v}}, & \text{if } u \neq v \text{ and } a_{uv} = 1 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

$$\begin{aligned} \phi_i^T L \phi_i =& \lambda_i \\ & \lambda_1 \leq \cdots \leq \lambda_i \leq \cdots \leq \lambda_n \end{aligned} \tag{3}$$

The multi-set of eigenvalues is usually used as an embedding into a feature space where graphs are compared using Euclidean distance. We have found that using the cosine similarity to measure the angle between the eigenvalues of both decomposed graphs yields better classification performance.

The similarity of two graphs G_A and G_B can therefore be computed by measuring the angle between the features vectors λ^A and λ^B (the multi-sets of eigenvalues of the respective graphs A and B).

$$K = \frac{\langle \lambda^A, \lambda^B \rangle}{|\lambda^A||\lambda^B|} \tag{4}$$

4.3. Random walk graph kernel

The random walk kernel is based on the idea that two similar graphs share many identical walks [21] and their similarity can be measured by counting the number of identical walks.

A naive and computationally very expensive approach would be to generate walks for two graphs randomly, and to compare all pairs of walks. A faster approach makes use of the properties of the product graph of the two graphs being compared. The product graph is constructed by computing the Kronecker product of the two graph adjacency matrices. The product graph has an edge only if the corresponding nodes in both of the original graphs are adjacent.

Exponentiating an adjacency matrix of graph is used to count the number of walks in a matrix. Exponentiating the adjacency matrix of a product graph therefore leads to the number of shared walks in both original graphs. The computation of such a random walk kernel is a deterministic process that converges towards the stochastic solution with each iteration of the exponentiation. The count of iterations is also the maximal walk length to be found and is denoted by n. We set $n = 10$. We found that higher values ($n = 20$) did not improve classification performance. A_A and A_B are the adjacency matrix of the graphs G_A and G_B being compared. The operator $\otimes$ is the Kronecker product of two matrices. K is the resulting kernel computing the similarity of two graphs.

$$\begin{aligned} R &= A_A \otimes A_B \\ K &= \sum_k^n R^k \end{aligned} \tag{5}$$

4.4. Delaunay triangulation

Transforming cuneiform characters in spline representation to a representation as graphs can result in a graph with multiple disconnected components. Topology based graph kernels as described in the previous sections do not pick up differences in graphs if one of these components is geometrically translated or rotated.

We extend all three presented methods by triangulating the extracted points using the Delaunay triangulation and additionally measuring the similarity between the Delaunay triangulated characters graphs.

The Delaunay triangulation should also consider geometrical translations and rotations where the spatial relationship (a wedge is below/above/right of/left of another wedge) significantly changes between the wedges. Small changes in position or shape do not matter for the classification of a character.

The graph kernel methods are extended by computing a new adjacency matrix (therefore new edges) from the key point set without considering the strokes the key points originate from. Edges are created instead by the Delaunay algorithm. Let D_A be a adjacency matrix of a Delaunay triangulated character graph G_A and A_A be the original adjacency of character graph G_A and K a graph kernel from one of the presented methods. K' is then a graph kernel that computes the similarity between two character graphs G_A and G_B.

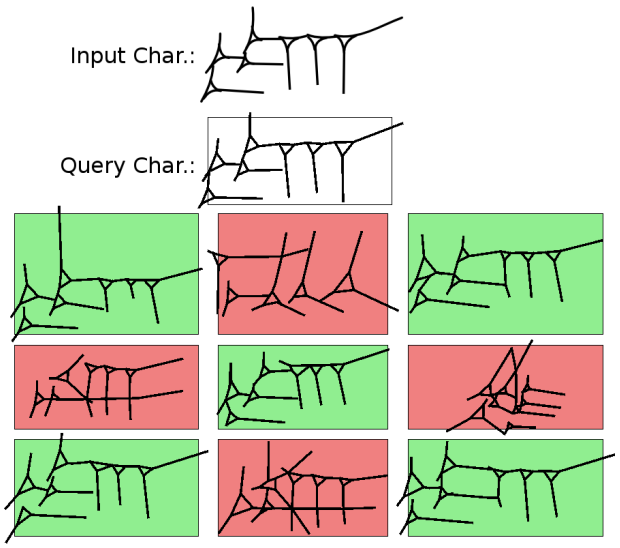


Figure 5. The top 9 results for the task to retrieve the Query Character (the prototype) are displayed. The Input Character is represented as a set of strokes. This set is then transformed into the graph representation of the Query Character. The kernel used for similarity is the spectrum kernel extended with Delaunay triangulation. The results are ordered from best (top left) to worst (bottom right). Characters with a green background have been correctly classified, characters with a red background have been incorrectly classified.

$$K' = \max\{K(A_A, B_A), K(D_A, D_B)\} \qquad (6)$$

We also tried the $\min$ operator but the classification performance was worse for all kernels except for the random walk kernel where the improvement in performance was negligible.

5. Experimental Evaluation

The data set used are a subset of several hundred 3D-scanned cuneiform tablets and tablets provided and manually transcribed into a vectorized file format by assyriologists. Only a subset of the words has been segmented manually since the tablets were partly damaged. There are 23 distinct word classes and 73 word instances used in the data set.

The task to test the classification performance of the presented methods was performed by hiding a prototype instance of the segmented words and comparing the remaining word instances against the prototype instance. The retrieved candidates were ranked by similarity from most similar to least similar.

The classification performance of the presented methods was then compared using a precision recall graph.

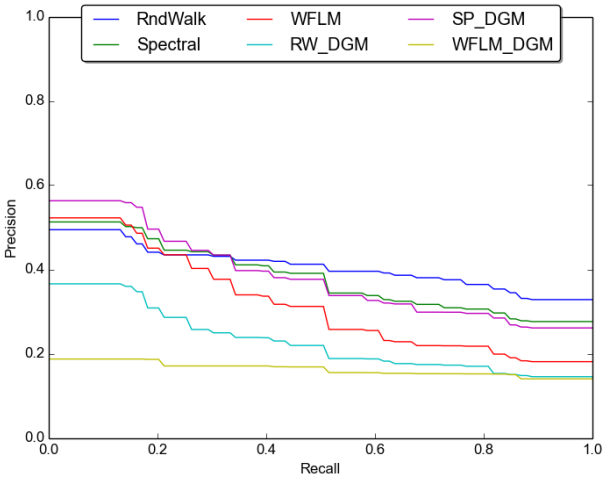


Figure 6. The precision recall graph for all the presented methods. A high precision implies that most of the retrieved character have the correct label, the false positive rate is low. If we increase the recall (the count of true positives and false positives, therefore ask for more results) the false positive rate climbs and the precision falls.

5.1. Precision and Recall

Figure 6 shows the classification performance of the various methods. The three basic methods are: the Weisfeiler-Lehman Graph Kernel (*WFLM*), the spectral decomposition (*Spectral*) and the random walk graph kernel (*RndWalk*). Then, the methods extended with the delaunay triangulation are *WFLM_DGM*, *SP_DGM* (for the spectral decomposition) and *RW_DGM* (for the random walk kernel), respectively.

The Delaunay transformation reduces precision greatly for the Weisfeiler-Lehman graph kernel. This kernel counts the number identical subtrees in both graphs. Since many vertices in a Delaunay triangulated graph have the same degree, two geometrically dissimilar triangulated graphs will share a high number of subtrees rendering them indistinguishable for the Weisfeiler-Lehman graph kernel.

The decrease in performance for the random walk kernel can be attributed to the same problem. Dissimilar triangulated graphs share a lot of random walks since most vertices are reachable by a high number of different walks.

The spectral decomposition, on the other hand, has better precision when extended with delaunay transformed graphs. The spectral decomposition can be seen as a series of minimal cuts [3] of a graph where the edge density is lowest. Translation and ro-

tation of wedges are therefore detectable by changes in connectivity of the graph partition leading to a better classification performance than just using the graph topology.

The random walk method and the Weisfeiler-Lehman graph kernel achieve better classification performance when the untransformed cuneiform graphs are used. Much more variating node degree and unique walks in the untransformed graphs enable those methods to differentiate cuneiform characters graphs better.

6. Conclusions and Outlook

Common handwriting recognition methods are not applicable to cuneiform characters. The Assyrian language has no means of separating words, thus making word segmentation very difficult. Cuneiform characters are very variable with respect to the positioning and rotation of their wedges and also exhibit a lot of complexity in the vertical direction without having a fixed shape that can be used by fixed-length feature vector classification methods.

We transform cuneiform characters into graphs and find that such a representation does not lose any structural elements of cuneiform and is very suitable for further analysis of the characters.

We applied graph kernels to classify cuneiform characters with the result that the random walk kernel performs best. The spectral decomposition, on the other hand, performs best when extended with the Delaunay triangulation and achieves the highest classification precision of all the presented methods.

We are currently working on using the wedge shaped impressions as a basic structural feature of cuneiform characters. A template shaped like an ideal wedge is used to match and extract wedges in a cuneiform character. The characters, decomposed into wedge shaped templates, are compared based on the similarity of their wedge shapes and the quality of the matching of the wedge configuration (position, orientation, overlap). To support our claim that conventional OCR methods are not suitable for cuneiform script we are currently investigating the classification performance of standard HMM and DTW methods on rasterized cuneiform script.

Additionally, we are investigating a method that represents the cuneiform characters as point clouds. Query word and candidate alignment and subsequent matching is performed with Iterated Closest Points [11].

Acknowledgements

We thank Prof. Stefan M. Maul and the members of the *Assur-Forschungsstelle* in Heidelberg, Germany for their support and fruitful discussions. This work is partially funded by the *Massnahme 5.4 – Zukunftskonzept* (institutional strategy) of the 2nd *German University Excellency Initiative*. Within this initiative the work is part of the *Field of Focus 3: Cultural Dynamics in Globalised Worlds*. Additional support is provided by the *Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences* (HGS MathComp).

References

[1] J. Almazan, A. Gordo, A. Fornes, and E. Valveny. Segmentation-free word spotting with exemplar svms. *Pattern Recognition*, 47(12):3967–3978, 2014. 3

[2] R. Borger. *Mesopotamisches Zeichenlexikon*. Alter Orient und Altes Testament / Alter Orient und Altes Testament. Ugarit Verlag Münster, 2004. 1

[3] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997. 4, 6

[4] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character hmms. *Pattern Recogn. Lett.*, 33(7):934–942, May 2012. 2

[5] A. Fischer, K. Riesen, and H. Bunke. Graph similarity features for hmm-based handwriting recognition in historical documents. In *Proceedings of the 2010 12th International Conference on Frontiers in Handwriting Recognition*, ICFHR '10, pages 253–258. IEEE Computer Society, 2010. 2

[6] D. Fisseler, F. Weichert, G. G. Mller, and M. Cammarosano. Extending Philological Research with Methods of 3D Computer Graphics Applied to Analysis of Cultural Heritage. In *Proc. of 12th Eurographics Workshop on Graphics and Cultural Heritage*, pages 165–172, Darmstadt, Germany, 2014. 1

[7] N. R. Howe. Part-structured inkball models for one-shot handwritten word spotting. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ICDAR '13, pages 582–586. IEEE Computer Society, 2013. 3

[8] S. Jakob. *Die mittelassyrischen Texte aus Tell Chuera in Nordost-Syrien*. Harrassowitz, O., July 2009. Tafel 26. 1

[9] D. Kaniewski, E. V. Campo, and J. G. et al. Environmental roots of the late bronze age crisis. *PLoS ONE*, 8(8), 2013. 1

[10] Y. Kessentini, C. Chatelain, and T. Paquet. Word spotting and regular expression detection in handwritten documents. In *ICDAR*, pages 516–520. IEEE, 2013. 2

[11] W.-S. Kim and R.-H. Park. Fast icp algorithm using a one-dimensional search. In *MVA*, pages 435–438, 2000. 7

[12] Y. Leydier, F. Lebourgeois, and H. Emptoz. Text search for medieval manuscript images. *Pattern Recognition*, 40(12):3552 – 3567, 2007. 3

[13] B. Luo, R. C. Wilson, and E. R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36(10):2213 – 2230, 2003. 4

[14] H. Mara, J. Hering, and S. Krömker. GPU based Optical Character Transcription for Ancient Inscription Recognition. In *Proc. of 15th International Conference on Virtual Systems and Multimedia (VSMM) – "Vision or Reality? Computer Technology and Science in Art, Cultural Heritage, Entertainment and Education"*, pages 154–159, Vienna, Austria, September 2009. 1

[15] H. Mara and S. Krömker. Vectorization of 3D-Characters by Integral Invariant Filtering of High-Resolution Triangular Meshes. In *Proc. of the 12th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 62–66, Washington D.C., USA, 2013. IEEE. 2

[16] H. Mara, S. Krömker, S. Jakob, and B. Breuckmann. GigaMesh and Gilgamesh - 3D Multiscale Integral Invariant Cuneiform Character Extraction. In A. Artusi, M. Joly, G. Lucet, D. Pitzalis, and A. Ribes, editors, *Proc. VAST Int. Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pages 131–138, Palais du Louvre, Paris, France, 2010. Eurographics Association. 1

[17] L. Rothacker, M. Rusiol, and G. A. Fink. Bag-of-features hmms for segmentation-free word spotting in handwritten documents. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ICDAR '13, pages 1305–1309. IEEE Computer Society, 2013. 2

[18] M. Rusiol, D. Aldavert, R. Toledo, and J. Llads. Efficient segmentation-free keyword spotting in historical document collections. *Pattern Recognition*, 48(2):545 – 555, 2015. 2

[19] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, Nov. 2011. 4

[20] M. Thaller, editor. *Optical Character Recognition in the Historical Discipline. Proc. of an Int. Workgroup by the Netherlands Historical Data Archive, Nijmegen Institute for Cognition and Information*, volume A18 of *Halbgraue Reihe zur Historischen Fachinformatik*, Göttingen, Germany, 1993. Max-Planck-Institut fr Geschichte. 1

[21] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, Aug. 2010. 5

[22] W. von Soden. *The Ancient Orient: An Introduction to the Study of the Ancient Near East*. Lightning Source Incorporated, 1994. 1

[23] B. Weisfeiler and A. A. Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9), 1968. 4

[24] S. Wshah, G. Kumar, and V. Govindaraju. Statistical script independent word spotting in offline handwritten documents. *Pattern Recognition*, 47(3):1039–1050, 2014. 2