EUROGRAPHICS Workshop on Graphics and Cultural Heritage (2024)
M. Corsini, D. Ferdani, A. Kuijper, and H. Kutlu (Editors)

Short Paper

Wedge Detection for Predictive Graphical Annotation of Cuneiform Tablets in 3D

J.P. Bullenkamp[1] and H. Mara[2]

[1]Martin-Luther-University of Halle-Wittenberg, Institute of Computer Science, FCGLab
[2]Free University of Berlin, Institute of Archaeoinformatics and -software, FCGLab

Abstract

As Digital Ancient Near Studies (DANES) and Ancient Language Processing (ALP) require larger volumes of annotated documents, we focus on automatically computed graphical annotations for cuneiform tablets, which are among the oldest documents spanning at least three millennia of human history. Cuneiform consists of wedges imprinted on clay tablets, which are best captured by high-resolution 3D acquisition. Existing Open Access 3D models are therefore our data base and have been post-processed by the GigaMesh software framework, which ensures clean meshes and provides MSII filter responses. This refers to the Gaussian curvature, which is a function value for each vertex of the mesh used for watersheds along the 2D manifold. Watershed labels are determined by non-minimum suppression using a merge parameter. The label contours, i.e., the polygonal lines enclosing the wedges, are smoothed using the Savitzky-Golay filter with the goal of projecting a low-poly line in 2D that can be quickly corrected by the image-based annotation tool known as Cuneur, which allows experts to annotate wedges, signs and semantics. Once the wedges have been determined, wedge types can be automatically proposed using the orientation of the wedges, which is the first step towards a paleographic representation of cuneiform signs, known as PaleoCode.

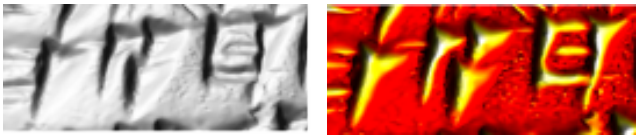
CCS Concepts

• ***Computing methodologies*** → *Computer graphics;* • ***Applied computing*** → *Archaeology;*

1. Introduction

Cuneiform is one of the earliest known writing systems and offers invaluable insights into the ancient civilizations of Mesopotamia over more than three millenia. Originally written on clay tablets, cuneiform is a 3D script, that was created using a reed stylus to imprint wedge-shaped marks and create cuneiform signs. Due to the easy availability of the required resources, clay and a reed stylus, as well as their better chances of preservation compared to papyrus for example, the number of cuneiform tablets found today is in the hundreds of thousands. Being used for more than three millenia and written in several different languages, changing over time with the civilizations, cuneiform tablets contain insight in the cultural, historical and political context in the Ancient Middle East.

Therefore deciphering and analyzing cuneiform tablets is crucial for understanding these early societies. Traditionally, the interpretation of cuneiform has been a meticulous and labor-intensive process, requiring experts to manually transliterate, transcribe and translate the wedge-shaped marks inscribed on the clay tablets. In recent years 3D scanned cuneiform tablets have become more popular as they are the best method for digitizing this 3D script and also allow further processing [Mar19], [MH23]. Figure 1 shows enhanced visibility of wedges by calculating curvature values, making the tablet easier to read. In Digital Ancient Near Eastern Studies (DANES) † now wedges and signs are annotated on 3D renderings of cuneiform tablets using the *Cuneur Annotator* [HZMB22].



(a) *Solid color* **(b)** *MSII curvature values*

Figure 1: *Cuneiform tablet HS 0275A cutout*

One cuneiform sign is often build on average from about five to ten wedges. In today's practice wedge annotation requires to manually set polygon points along the wedge outline, while a sign can just be drawn as a rectangle. Asking experts that do the annotations, especially the wedge annotations are therefore inconvenient and tedious compared to sign annotations, which also are most important from an assyriological point of view. From a computational point of view though, wedge annotations are the first step in automating the manual workflow, similar to optical character recognition being the first step in the conversion of an image to a (translated) text. Further, due to the temporal development of the cuneiform scriptures and signs, we are more interested in wedges as a reliable source

† https://opendanes.org/

DOI: 10.2312/gch.20241260

and getting the sign interpretation from groupings of wedges in a future step.

There are virtual no wedge annotations because the manual annotation is very time intense as five to ten times per sign a polygon has to be selected. Machine learning approaches for wedge detection and classification require large annotated groundtruth datasets though and in [SHBM23] the lack of wedge annotation was one reason why sign detection performed better than wedge detection.

The presented work has the goal to simplify the annotation process of wedges by detecting them on the 3D model and proposing a polygon outline as well as a wedge classification according to the wedge types used in PaleoCodage [Hom21] machine-readable encoding for cuneiform signs. We will use a watersheding based approach on a mesh with MSII curvature values, computed in *GigaMesh*[‡] to segment the wedges. The outline of each wedge is then extracted as a simplified polygon, which is used to predict the wedge type. This will allow to compute wedge predictions before starting the annotation process and therefore not require to set polygon and wedge type for all wedges of the tablet.

2. Related Work

In this section we review works related to sign or wedge detection on cuneiform tablets as well as annotation efforts and datasets.

Our goal is to aid the wedge annotation process in the annotation tool *Cuneur Annotator* [HZMB22]. This tool allows to annotate signs and wedges on renderings of cuneiform tablets and link the annotations to the 3D model. Besides the visual annotation as a polygon, it allows to annotate sign transliterations, wedge type and further information on line, column etc.

One relevant question that arises for us, is how to classify wedge types. Gottstein [Got12] developed a classification system which was then expanded in the PaleoCodage [Hom21] encoding used in the *Cuneur Annotator*. It classifies the wedge types shown in Table 1, where the wedge types $a-f$ are defined by their orientation, while the special wedge types contain Winkelhaken (w), full seal wedges (x) and half seal wedges (y).

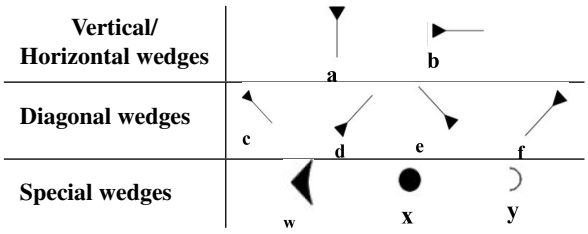


Table 1: *Wedge types in PaleoCodage*

Some open-source cuneiform datasets are the *Heidelberg Cuneiform Benchmark Dataset* (HeiCuBeDa) containing the *Frau Professor Hilprecht Collection of Babylonian antiquities* [Mar19] and the *Haft Tappeh* datasets [HZMB22]. Annotations of [Mar19] made in the *Cuneur Annotator* are also published in the *Mainz Cuneiform Benchmark Dataset* [MH23].

‡ https://gigamesh.eu/

Most work is done on cuneiform sign detection rather than wedges, as for example [DKMO20], [WSS*23] or [CSKJ24]. [HFP*22] did work on detecting horizontal wedges and [SHBM23] also used a learning-based approach to detect cuneiform signs and wedges on 2D renderings and photographs.

[Fis19] used a watersheding based segmentation of wedges to create skeleton models of the wedges similar to drawings, but not to prepare any annotation which allow for corrections.

3. Method

We now show how to detect and classify wedges. Taking cuneiform tablets from [Mar19], the meshes have been cleaned, oriented and MSII curvature values were computed for each vertex. The meshes require to have no non-manifold defects and be oriented, which is necessary to make wedge classification possible. We then use a watersheding on the curvature values to segment the wedges, determine the simplified outlines and predict classifications according to the wedge types in PaleoCodage [Hom21]. In a last step towards making it an annotation prediction, we project the outlines in 2D, resembling the renderings used in the *Cuneur Annotator*.

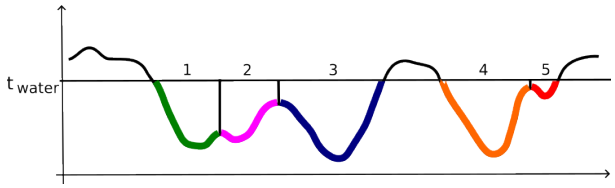
3.1. Watershed wedge segmentation

Watershed algorithms can be used for segmentation of images, but also on 3D meshes [MW99]. In this case the aim is to segment the wedges on the 3D mesh by watersheding the curvature values on the vertices. We do this following a bottom up approach without relying on predefined starting points, as we can determine these from the curvature values.

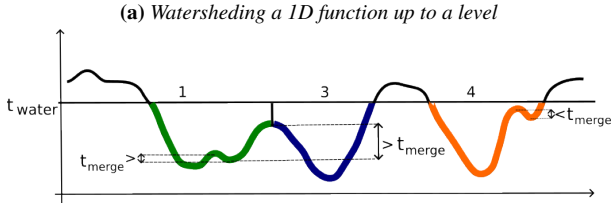
The bottom up approach means we traverse the vertices of the mesh ordered by their curvature values, starting at the lowest curvature valued vertex. Always if we have a local minimum, so no neighboring vertex has a lower curvature value, we start a new watershed label. Every other vertex should have a neighboring vertex with a lower curvature value that already has been labeled, so the same label can be assigned. If there are several neighboring vertices with lower curvature values but different labels, the vertex can be assigned to the label of the lowest neighboring curvature value and can be marked as a boundary vertex between the two labels. An upper limit t_{water} can be set to only label vertices with a curvature value below it. This allows to segment only the concave parts of the mesh that contain the wedges. Now the mesh is separated into background (unlabeled) and labelled components for each local minimum below the curvature value limit. Figure 2(a) shows this concept on a 1D function.

This method currently has no non-minimum suppression, as every local minimum will result in a new label, even if there is a deeper minimum right beside it. To address this problem, we define another parameter t_{merge} to merge adjacent labels. We already track vertices that form a boundary between two adjacent labels, so we can define the crossing vertex v_c of label a and b as the vertex along their boundary with the lowest curvature value. The minimum of the curvature difference to the local minima v_a, v_b in the two labels then defines a metric

$$d(a,b) = \min(v_c - v_a, v_c - v_b)$$



(a) *Watersheding a 1D function up to a level*



(b) *Watershed labels by non-minimum suppression using a merging parameter*

Figure 2: *Watersheding*

we can use to merge adjacent labels. So if $d(a,b) < t_{merge}$, labels a and b are merged, keeping the label of the lower minimum. Figure 2 (b) shows the result after such a merging. Note that this is just a 1D function, so there is only one possibility for a crossing point. On a mesh surface, the boundary between two labels can contain several vertices, requiring the definition of the lowest crossing vertex v_c.

3.2. Wedge outline

In order to give a possible annotation of a wedge, the outline of the 3D wedge needs to be given as a polygon with only a few points in 2D. Following the boundary of the wedge, we can define a function along its vertices. Taking the euclidean distance of each vertex to the local minimum of the wedge yields a function that has minima or maxima at the shape defining vertices of the wedge. Figure 3 shows an example of the distances plotted along the ordered boundary vertices.

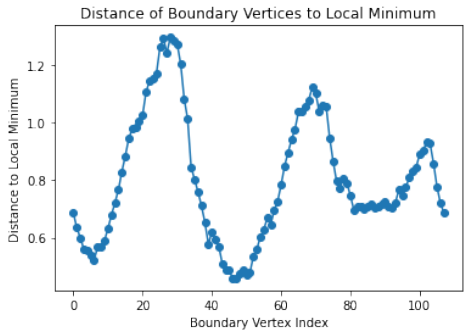


Figure 3: *Function of boundary vertices euclidean distance to local minimum*

Smoothing this function with a heuristically chosen Savitzky–Golay filter to eliminate noisy local extrema, we can get extrema of this function, representing corner points of the wedge. Figure 4 shows the smoothed function with marked extrema.

Mapping the extrema into 2D yields us a good approximation of the wedge outline in just a few points. Having in mind that the annotation process is done on renderings of the 3D mesh, that are

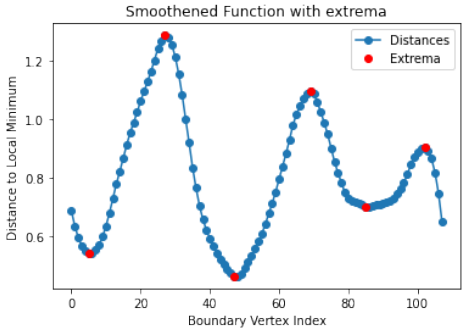


Figure 4: *Extrema on smoothed function*

usually taken from all six bounding box directions as used in the so-called fat cross presentation of tablets, we also project the wedge outline into that coordinate system. The mapping can be done by getting the average normal of all vertices of the wedge, comparing which side of the bounding box that corresponds to and fixing this coordinate, giving us 2D points. The resulting 2D polygon for our example wedge is shown in Figure 5.

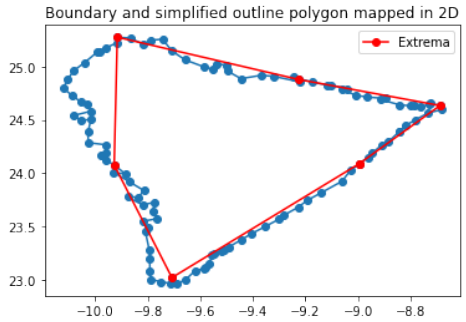


Figure 5: *Boundary and extrema of an example wedge mapped in 2D*

3.3. Wedge classification

We will classify wedges according to the wedge types in the PaleoCodage [Hom21] classification shown in Table 1, where especially wedge types $a-f$ can be classified by the angle of their longest edge, if the writing direction is known. The special wedge types x and y occur only rarely, while w would be difficult to distinguish from c and d, so we neglect these in our classification.

Looking at the vector pointing from the local minimum to the farthest outline point we can classify the wedge by comparing that to the ideal vectors of the different wedge types. This step requires the mesh to be oriented, since this is the only way to compare the orientation of the wedge vector to the expected writing direction and we then assign the closest fitting wedge type.

4. Results

We now show the results of our method on a cuneiform tablet from the Hilprecht Collection [Mar19]: tablet *HS 0275A*. The watersheding result with $t_{water} = -0.1$ (curvature values in $[-1,1] \triangleq$ [concave,convex]) and $t_{merge} = 0.2$ can be seen in Figure 6. These parameters yield good results for the segmentation of wedges on various cuneiform tablets, so we usually do not need to change them.

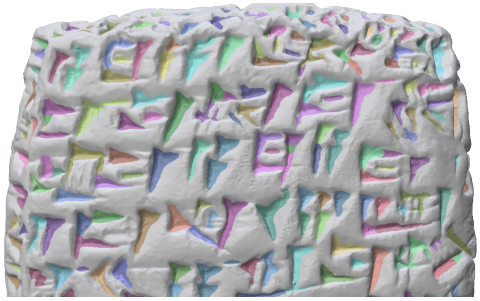


Figure 6: *Watershed on cuneiform tablet HS 0275A*

Obtaining the simplified outline of all wedges on the tablet and mapping them onto the six sides of the mesh bounding box is shown in Figure 7. This will allow the bounding boxes to be exported as a proposed annotation in a future step quite easily, as annotations are currently performed on renderings of exactly these six views.

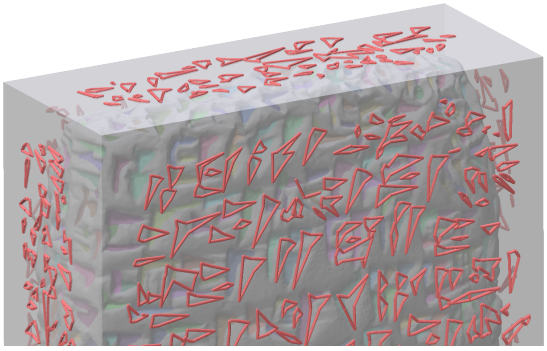


Figure 7: *Wedges outlines projected onto the bounding box of cuneiform tablet HS 0275A*

Finally we predict the wedge type for each wedge and map them as colors on the mesh as well, shown in Figure 8.

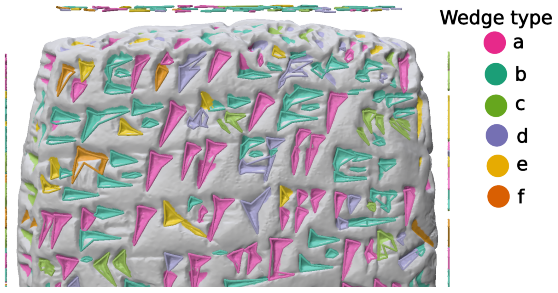


Figure 8: *Classified wedges with outlines projected to the bounding box on cuneiform tablet HS 0275A*

5. Future Work

As improving the wedge annotation process is the goal of this work, we are incorporating the wedge detection to export predictions that can be loaded in the *Cuneur Annotator*. The predictions then can be edited or removed in the *Cuneur Annotator* by experts if needed, speeding up the process and giving experts the option to adjust annotations. For the wedge classification this is important, as the assumption of a horizontal writing direction does not always hold true due to the diversity of cuneiform tablets. Once an interface to the *Cuneur* is build, we can also compare quantitatively and qualitatively to expert annotations and include feedback into our method.

A large number of wedge annotations will also allow to develop approaches that statistically analyze the number, type and density of wedges on cuneiform tablets and learning-based algorithms that segment and classify wedges. Learning-based methods require large numbers of annotated groundtruth data, for which our method offers annotations in 2D as well as 3D.

Acknowledgements

We thank Florian Linsel and Timo Homburg for their support and fruitful discussions.

References

[CSKJ24] COBANOGLU Y., SÁENZ L., KHAIT I., JIMÉNEZ E.: Sign detection for cuneiform tablets. *it - Information Technology* (2024). doi:doi:10.1515/itit-2024-0028. 2

[DKMO20] DENCKER T., KLINKISCH P., MAUL S. M., OMMER B.: Deep learning of cuneiform sign detection with weak supervision using transliteration alignment. *PLOS ONE 15*, 12 (12 2020), 1–21. doi:10.1371/journal.pone.0243039. 2

[Fis19] FISSELER D. B.: Contributions to computer-aided analysis of cuneiform tablet fragments. 2

[Got12] GOTTSTEIN N.: Ein stringentes identifikations-und suchsystem für keilschriftzeichen, 2012. 2

[HFP*22] HAMPLOVÁ A., FRANC D., PAVLÍČEK J., ROMACH A., GORDIN S.: Cuneiform reading using computer vision algorithms. In *Proceedings of the 2022 5th International Conference on Signal Processing and Machine Learning* (New York, NY, USA, 2022), SPML '22, Association for Computing Machinery, p. 242–245. doi:10.1145/3556384.3556421. 2

[Hom21] HOMBURG T.: Paleocodage—enhancing machine-readable cuneiform descriptions using a machine-readable paleographic encoding. *Digital Scholarship in the Humanities 36*, Supplement_2 (2021), ii127–ii154. 2, 3

[HZMB22] HOMBURG T., ZWICK R., MARA H., BRUHN K.-C.: Annotated 3D-Models of Cuneiform Tablets. *Journal of Open Archaeology Data (JOAD) 10*, 92 (2022), 4. doi:10.5334/joad.92. 1, 2

[Mar19] MARA H.: HeiCuBeDa Hilprecht - Heidelberg Cuneiform Benchmark Dataset for the Hilprecht Collection, 2019. doi:10.11588/data/IE8CCN. 1, 2, 3

[MH23] MARA H., HOMBURG T.: MaiCuBeDa Hilprecht - Mainz Cuneiform Benchmark Dataset for the Hilprecht Collection, 2023. doi:10.11588/data/QSNIQ2. 1, 2

[MW99] MANGAN A., WHITAKER R.: Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 308–321. doi:10.1109/2945.817348. 2

[SHBM23] STÖTZNER E., HOMBURG T., BULLENKAMP J. P., MARA H.: R-CNN based PolygonalWedge Detection Learned from Annotated 3D Renderings and Mapped Photographs of Open Data Cuneiform Tablets. In *Eurographics Workshop on Graphics and Cultural Heritage* (2023), Bucciero A., Fanini B., Graf H., Pescarin S., Rizvic S., (Eds.), The Eurographics Association. doi:10.2312/gch.20231157. 2

[WSS*23] WILLIAMS E. C., SU G., SCHLOEN S. R., PROSSER M. C., PAULUS S., KRISHNAN S.: Deepscribe: Localization and classification of elamite cuneiform signs via deep learning, 2023. URL: https://arxiv.org/abs/2306.01268, arXiv:2306.01268. 2