21^{st} *Computer Vision Winter Workshop*
Luka Čehovin, Rok Mandeljc, Vitomir Štruc (eds.)
Rimske Toplice, Slovenia, February 3–5, 2016

Cuneiform Detection in Vectorized Raster Images

Judith Massa[1], Bartosz Bogacz[1], Susanne Krömker[2] and Hubert Mara[1]
Interdisciplinary Center for Scientific Computing (IWR)
[1]Forensic Computational Geometry Laboratory (FCGL)
[2]Visualization and Numerical Geometry (NGG)
Heidelberg University, Germany
{judith.massa|bartosz.bogacz|susanne.kroemker|hubert.mara}@iwr.uni-heidelberg.de

Abstract. *Documents written in cuneiform script are one of the largest sources about ancient history. The script is written by imprinting wedges (Latin: cunei) into clay tablets and was used for almost four millennia. This three-dimensional script is typically transcribed by hand with ink on paper. These transcriptions are available in large quantities as raster graphics by online sources like the Cuneiform Database Library Initative (CDLI). Within this article we present an approach to extract Scalable Vector Graphics (SVG) in 2D from raster images as we previously did from 3D models. This enlarges our basis of data sets for tasks like word-spotting. In the first step of vectorizing the raster images we extract smooth outlines and a minimal graph representation of sets of wedges, i.e., main components of cuneiform characters. Then we discretize these outlines followed by a Delaunay triangulation to extract skeletons of sets of connected wedges. To separate the sets into single wedges we experimented with different conflict resolution strategies and candidate pruning. A thorough evaluation of our methods and its parameters on real word data shows that the wedges are extracted with a true positive rate of 0.98. At the same time the false positive rate is 0.2, which requires future extension by using statistics about geometric configurations of wedge sets.*

1. Introduction

Documents were written in cuneiform script for more than three millenia in the ancient Middle East [26]. Cuneiform characters were typically written on clay tablets by imprinting a rectangular stylus and leaving a wedge (*cuneus* in Latin) shaped trace, i.e., triangular markings. As clay was always cheaply and easily available, everybody capable of writing could produce robust documents. Therefore, the content of cuneiform tablets ranges from simple shopping lists to treaties between empires. The number of known tablets is assumed to be in the hundreds of thousands, which is constantly increasing as new tablets are excavated by archaeologists on a regular basis. By roughly estimating the number of words on those tablets, we can assume that the total amount of text in cuneiform script is comparable to those in Latin or Ancient Greek.

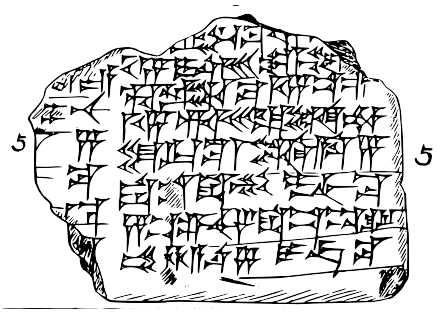
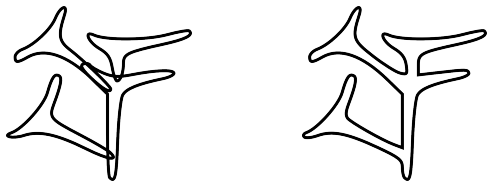


Figure 1: A tracing of the tablet VAT6546 [23]

Since 1999, a number of projects have been launched to facilitate the work of Assyriologists. The *Digital Hammurabi Project* is concerned with the digitization of cuneiform tablets [27]. Achievements of the project include the creation of high-resolution 3D models [17] as well as 3D and 2D visualization techniques for the models. Similarly, projects in Leuven deal with the efficient production of 3D models of tablets [28] and techniques to visualize the models [13]. The *Cuneiform Digital Library Initiative* [15] incorporates a number of projects aimed at cataloging cuneiform documents and making them available online as transliteration, tracing and 2D image. In [11], the software framework *CuneiformAnalyzer* is introduced. It assists the researchers in script



(a) (b)

Figure 2: (a) Paths described by four distinct wedge shapes and (b) the path described by the compound shape formed by them.

analysis by detecting and segmenting wedge impressions of 3D models [10]. Furthermore, the program simplifies collation of fragments and reconstruction of tablets with methods from 3D computer graphics. The *GigaMesh* project contributes with visualization methods and extraction of cuneiform characters from tablets [21, 20].

Our method extracts wedge-shaped impressions from raster images. These images are hand-drawn transcriptions of cuneiform tablets of varying quality and two different styles of marking wedges. We vectorize images of the transcriptions and match patterns and shapes to detect these constellations of wedges in the vectorized transcriptions.

2. Related Work

In [7] the problem of content-based image retrieval of Scalable Vector Graphics (SVG) documents is tackled. Their approach uses a description language to simplify comparisons between shapes. It represents an object by a basic shape, like a unit circle, and a transformation entailing its scale and translation from the origin. The resulting framework handles composites of simple SVG shapes, but no SVG path elements, which are able to represent arbitrary shapes. The similarity measure chosen is a weighted sum of shape, color, transformation, spatial and position similarity.

In [18] the problem of hierarchically clustering shapes described as vector graphics is addressed. Based on [29], Kuntz uses Fourier descriptors [6] to describe and compare single basic SVG shapes. The descriptors serve as feature vectors which then are clustered using state-of-the-art clustering algorithms.

We cannot apply Kuntz' method since it does not deal with shapes that are part of a compound described as one object. Yet, our input consists of SVG *path* elements that usually describe such compound shapes (Figure 2).

3. Implementation

Our implementation proceeds in five distinct steps. The first three steps transform raster image transcriptions into a set of skeletonized wedge constellations. The final two steps extract and prune wedge candidates from these constellations.

3.1. Vectorization

For the vectorization step, Selinger's *potrace* algorithm [1] is used. A directed graph G_1 is constructed by traveling along the edges between black and white pixels. Thereby, each vertex v in the graph corresponds to a pixel corner, which is adjacent to four pixels in the bitmap image of which at least one has to be black and one has to be white. An edge (v_i, v_{i+1}) between two vertices is created if the corresponding corners are neighbors in the bitmap image and the edge separates a black and a white pixel. Then, a path $p = \{v_0, \ldots, v_n\}$ is a sequence of vertices, where there is an edge between each pair of consecutive vertices v_i and v_{i+1} for $i = 0, \ldots, n-1$. A path is called *closed* if $v_0 = v_n$. Whenever a closed path is found, the color of the pixels enclosed by it is inverted. The algorithm is applied recursively to the new image until there are no black pixels left.

For each of the resulting paths a polygon is calculated. Therefore another directed graph G_2 is constructed, where each edge represents a straight path and the set of vertices of Graph G_2 is a subset of the vertices of Graph G_1 reduced to the endpoints of the straight paths. A path $p = \{v_0, \ldots, v_n\}$ is called straight if for all index triples (i, j, k) with $0 \leq i < j < k \leq n$ there exists a point w on the straight line through v_i and v_k such that $d(v_j, w) \leq 1$. The function d denotes the Euclidean norm. Furthermore, not all four possible vertex-to-vertex directions $v_{(i+1)} - v_i$ may occur in the path (Figure 3).

Each edge then is assigned a penalty $P_{i,j}$ for using the corresponding straight path for the resulting polygon. The penalty is the product of the Euclidean length and the standard deviation of vertex distances.

$$D_{i,j} = \sum_{k=i}^{j} dist(v_k, \overline{v_i v_j})^2 \tag{1}$$

$$P_{i,j} = |v_i - v_j| \cdot \sqrt{\frac{1}{j \ominus i + 1} \cdot D_{i,j}} \tag{2}$$

[1] http://potrace.sourceforge.net. Project page of *potrace*. Last visited on 4/11/15.

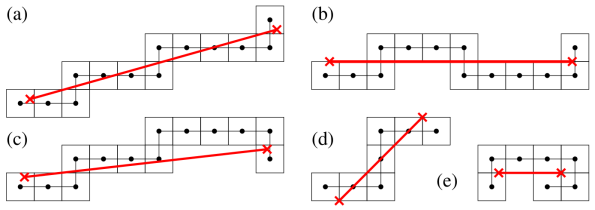


Figure 3: Shows how *potrace* checks if paths are straight. The dots represent the vertices of the paths and the squares the 1/2-neighborhoods of the vertices. Paths in (a), (b) and (d) are straight and (c) and (e) are not.

with $j \ominus i = j - i$ if $i \leq j$ and $j \ominus i = j - i + n$ if $j \leq i$ and $dist(a, \overrightarrow{cd})$ the Euclidean distance of a point to a straight segment. Finding an optimal polygon then is the equivalent to finding an optimal cycle in graph G_2, with the quality measured by the tuple (k, P), where k is the number of straight paths that make up the cycle and P is the sum of respective penalties. With that, a polygon with a smaller penalty but more segments is considered worse than a polygon with less segments but higher penalty.

After choosing a polygon, Bézier curves are calculated and by doing this, a smoothing of the corners is achieved where it seems reasonable. Optionally, consecutive curves are joined if the segments agree in convexity and the total direction change does not exceed 89 degrees.

3.2. Discretization

We tested minimizing the maximum distance between the polygon line segments and the contour segments, but found that even if a polygon approximates an arbitrary shape well, it is still not assured that the resulting Voronoi skeleton will be a good approximation to the skeleton of the wedge constellation. The key for a good discrete skeleton turned out to be the limitation of the distance of two sample points along the shape outline. However, calculating the distance along a Bézier curve $B(t)$ is a complex task [12] since the length s of the complete curve is

$$s = \int_0^1 \sqrt{B'_x(t)^2 + B'_y(t)^2} dt, \tag{3}$$

which has no closed-form solution. Yet, we know that the curve length s of a Bézier of degree 3 has the sum of the distances of consecutive control points C_i as upper bound:

$$s \leq \sum_{i=0}^{2} \|C_{i+1} - C_i\|. \tag{4}$$

Since smaller distances along the path can only improve the quality of the resulting skeleton, this upper bound is used for discretizing the silhouette.

3.3. Skeletonization

In order to deal with occluding wedge marks in the detection step, shape skeletons are used as intermediate representations. Different definitions for shape skeletons have been stated since [2, 22, 25, 19]. Based on [25], we define a shape skeleton as the infinite set of points within the shape boundaries that have more than one closest point on the shape outline. The skeleton can be computed efficiently with time complexity $O(n \log n)$ by using Voronoi diagrams [16]. The Voronoi diagram for a set of sites S divides a space into $|S|$ partitions called the Voronoi regions. In $\mathbb{R}^2$ a Voronoi region is the interior of a convex polygon, whose boundaries, the Voronoi edges, are equidistant to two of the input sites. As input sites, the polygon vertices obtained in the previous discretization step are used.

The Voronoi diagram (Figure 4b) is computed by solving the dual problem first: the Delaunay triangulation (Figure 4a). Each Voronoi vertex represents the circumcenter of a Delaunay facet and a Voronoi ridge connects two such points of neighboring facets. An implementation of the *quickhull* algorithm [1] is used to calculate the 2-dimensional Delaunay triangulation from a 3-dimensional convex hull.

The Voronoi ridges with end points outside the original shape boundaries and ridges crossing the contour are removed. The skeleton, represented as an undirected graph (Figure 4c), consists of more and in general shorter segments. These, in turn, are made up of longer segments the more vertices form the approximated polygon. Since the short segments are rarely meaningful, considering their directions, a new skeleton (Figure 4d) is constructed. The computation is done by a graph traversal algorithm. It follows a series of consecutive edges until an end node is incident to more than two edges in the original skeleton.

3.4. Extraction

The basic shape of a wedge impression can be described by a Y- or T-junction. We call this junction

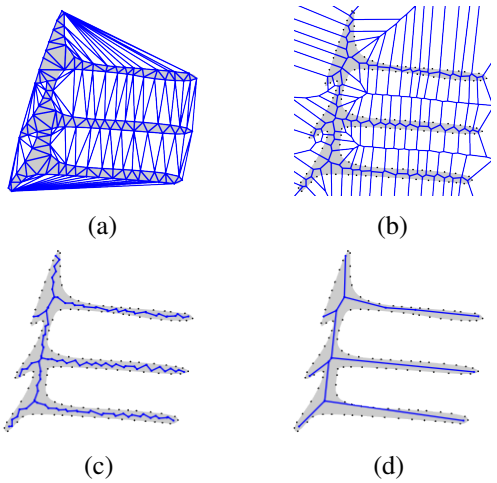


Figure 4: Visualization of important steps of the skeleton computation and simplification process: (a) the Delaunay triangulation, (b) the Voronoi diagram, (c) the inner elements of the Voronoi diagram and (d) the resulting skeleton.

the wedge-head and the ridges extending from the junction the wedge-arms.

After having computed a shape skeleton, the detection of the wedge-heads of the impressions can be approached. There are two different ways a wedge impression can be drawn: as contour lines or shapes filled with ink. For a single wedge impression, the filled shape results in a single closed curve after bitmap tracing and the shape contour is represented as two closed curves, where only the area between both curves is filled with color. Usually, the representation as unfilled shape contour is intended, but for small wedges, the thickness of the pencil used for the original ink tracing sometimes leads to solid shapes. The two representations result in two different skeletons as shown in Figure 5. These two cases are considered separately and certainty values w_{loc} are calculated for locations in the skeleton graph that seem likely to contain a wedge-head. In both cases, w_{loc} ranges from zero to one with values close to one indicating a high probability of a wedge-head at the considered location. The result of this step is a set of wedge-heads for which the certainty value exceeds the threshold $t_{loc}^{contour}$ for contour wedges or t_{loc}^{solid} for solid wedges.

Wedge-Head Detection of Shape Contours Having a wedge impression represented as a contour, the respective skeleton graph shows a cycle resembling a triangle at the position of the wedge-head (Figure 5a). This fact is used to locate the wedge-head of a wedge contour. A cycle in an undirected graph is an ordered set of vertices

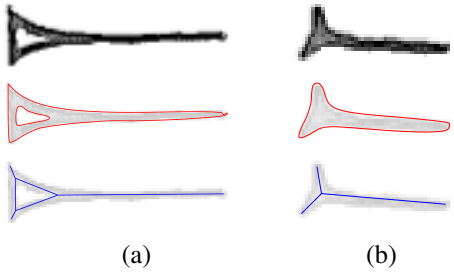


Figure 5: Two different ways of representing a wedge impression in ink tracings: a) as unfilled contour and b) as filled shape. The result of the bitmap trace is drawn in red, the simplified Voronoi skeleton in blue.

$$\mathcal{C} = (v_0, v_1, \ldots, v_n) \tag{5}$$

where circular consecutive vertices are adjacent and no vertex appears twice.

To avoid outliers to be taken into consideration as wedge-heads, we only look for short cycles. Two concepts of length are possible: the number of edges forming the cycle

$$l_{edge}(\mathcal{C}) = |\mathcal{C}| \leq t_{edge} \tag{6}$$

and the accumulated distances l_{dist} along the cycle path, using P_v as the coordinate of a node v, that can be formally defined by

$$l_{dist}(\mathcal{C}) = \sum_{i=0}^{|\mathcal{C}|-1} |P_{v_i} - P_{v_j}| \leq t_{dist} \tag{7}$$

with $v_i, v_j \in \mathcal{C}$ and $j = (i+1) \bmod |\mathcal{C}|$ thresholds t_{edge} and t_{dist}.

Using l_{edge} as the cycle length, this results in a time complexity of $O(|E| \cdot t_{edge})$, using l_{dist}, the complexity will be $O(|E| \cdot \left\lfloor \frac{t_{dist}}{\min\{\|P_u - P_v\| : (u,v) \in \mathcal{E}\}} \right\rfloor$ in the worst case. These concepts are used next to each other in the algorithm.

The cycle extraction proceeds as follows: A depth-first search tree is built and the back-edges are extracted. For each back-edge (u, v) a depth-limited search for node v is conducted with u as root node; paths from v to u represent cycles when joined with $\{(u, v)\}$, except the direct path $\{(v, u)\}$. At last, the set of cycles is reduced to contain only unique cycles.

A set of unique cycles contains no two cycles that are equivalent, i.e., if they are induced by the same

set of graph edges. This is tested by:

$$\mathcal{C}_1 \equiv \mathcal{C}_2 \quad \Longleftrightarrow \quad \mathcal{E}_{\mathcal{C}_1} \setminus \mathcal{E}_{\mathcal{C}_2} = \emptyset, \tag{8}$$

where $\mathcal{E}_{\mathcal{C}}$ denotes the edge set of a cycle $\mathcal{C}$.

The depth-limited search stops following a search branch when either l_{edge} or l_{dist} exceed their respective thresholds, t_{edge} or t_{dist}, or when a target node is discovered. It returns a list of paths from the root to the target node.

Triangle Similarity Once all unique cycles of a skeleton graph have been extracted, their resemblance to a triangle can be analyzed. This can be achieved by comparing the triangle with the smallest error that can be created with the cycle's vertices with the original cycle (Figure 6). With

$$A_{err}^{\mathcal{C},(i_0,i_1,i_2)} = \sum_{j=0}^{2} A_{(c_{i_j}, c_{i_j+1 \bmod |\mathcal{C}|}, \ldots, c_{i_{j+1}})} \tag{9}$$

being the sum of error areas between triangle and polygon, we have

$$w_{loc}^{contour}(\mathcal{C}) = 1 - \min_{i_0,i_1,i_2;} \left\{ \frac{A_{err}^{\mathcal{C},(i_0,i_1,i_2)}}{A_{\mathcal{C}} + A_{err}^{\mathcal{C},(i_0,i_1,i_2)}} \right\} \tag{10}$$

for $0 \leq i_0 < i_1 < i_2 \leq |\mathcal{C}| - 1$, $\mathcal{C} = (c_0, \ldots, c_n)$ and $n \geq 2$.

The advantage of this similarity measure is that the three vertices of the triangle are also vertices of the skeleton graph, thus providing us with feasible starting points for the wedge-arm tracing. Since the cycles form simple polygons, the enclosed areas can be calculated with the shoelace or surveyor's area formula [4].

Wedge-Head Detection of Solid Shapes Solid imprints have their centers at junctions v of a skeleton $\mathcal{S}$ with a particularly long distance from the shape contour. This distance is approximated by the distance of the coordinate P_v of the vertex v to all sites s with $s \in S(v)$, where S is the site set of the underlying Voronoi diagram and

$$S(v) = \{s \in S | P_v \text{ is vertex of } V(s)\} \tag{11}$$

is the set of sites with P_v being a vertex of their Voronoi region $V(s)$. The equation

$$d(P_v, s_1) = d(P_v, s_2) \qquad \forall s_1, s_2 \in S(v) \tag{12}$$

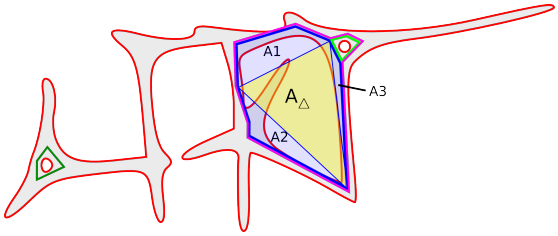


Figure 6: The triangle with area $A_\triangle$ shows the best triangle for the blue polygon. Since the sum of the error areas A_{err} = A1+A2+A3 is almost equal to the triangle area $A_\triangle$, the polygon with area $A_\triangle$+$A_{err}$is not one of the polygons chosen for wedge-head positions.

always holds by definition of the Voronoi diagram. Therefore, any random $s \in S(v)$ can be chosen to get a measure for the distance to the contour and use it as hint for plausible locations of heads of solid wedges (Figure 7). Percentiles of the site-to-vertex distances

$$\mathcal{D} = \{d(P_v, s) | v \in \mathcal{S} \wedge s \in S(v)\} \tag{13}$$

are used instead of the minimum and maximum to account for outliers. In order to arrive at a range from zero to one with values close to one for long distances $d(p_v, s)$ and close to zero for short distances, the first percentile is used as minimum d_{lower}, the 99th percentile of these distances as maximum d_{upper} and the position within this range is used as certainty value.

$$w_{loc}^{solid}(v) = \begin{cases} 0 & \text{if } d(P_v, s) \leq d_{lower} \\ 1 & \text{if } d(P_v, s) \geq d_{upper} \\ d^*(P_v, s) & \text{else} \end{cases} \tag{14}$$

$$d^*(P_v, s) = \frac{d(P_v, s) - d_{lower}}{d_{upper} - d_{lower}} \tag{15}$$

For a junction, where n edges meet, $\binom{n}{3}$ wedge-heads are retrieved.

After locating the position of a wedge-head, the extents of the impression are calculated. For a wedge-head, multiple wedges are proposed. Vertices $\{v_i\}_{i=1,2,3}$ must fulfill two conditions:

1. The line segment between the coordinates of a wedge vertex v_i and tracing start point S may not intersect with the shape boundary.

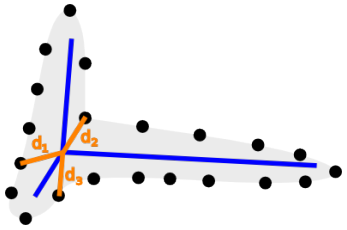


Figure 7: Possible locations of solid wedges are found by looking at skeleton junctions: if their distance to the closest discretization points is above a threshold, it is likely that the center of a wedge impression is located here. Due to the definition of the Voronoi skeleton, the equation $d_1 = d_2 = d_3$ holds at every skeleton junction.

2. A vertex of a wedge must be located within the infinite area between the lines through the coordinates of S and the wedge-head vertices v_j^H and v_k^H as shown in Figure 8. The condition can be checked by testing if

$$\measuredangle(\overrightarrow{P_S P_{v_i}}, \vec{v}) \leq \frac{\alpha}{2} \tag{16}$$

with

$$\alpha = \measuredangle(\overrightarrow{P_{v_j^H} P_S}, \overrightarrow{P_{v_k^H} P_S}) \tag{17}$$

and $\vec{v}$ being the angle bisector of $\overrightarrow{P_{v_j^H} P_S}$ and $\overrightarrow{P_{v_k^H} P_S}$.

For contour wedges, the tracing start node for a wedge vertex v_i is the respective wedge-head vertex v_i^H and for a solid wedge, the start node is the wedge center. The reason why for the contour vertex, the line checked for the conditions above starts at the head vertex instead of the wedge center is that the center in this case is not a part of the shape skeleton and is typically located inside a hole in the shape.

From the respective start node the algorithm follows all paths within the area of valid nodes shown in Figure 8. A path may have sections of a certain number of nodes that are inadmissible as wedge vertices. The algorithm returns multiple arms for one direction resulting in multiple wedge suggestions for a wedge-head. If $n_{arm\,1}$, $n_{arm\,2}$ and $n_{arm\,3}$ are the number of arms returned for the respective wedge-head vertex, the number of wedges is $n_{arm\,1} \cdot n_{arm\,2} \cdot n_{arm\,3}$.

3.5. Wedge Set Reduction

The certainty measures for wedge-head locations w_{loc} can only be used as first hints to possible locations. After wedge-arm tracing, we still get wedge

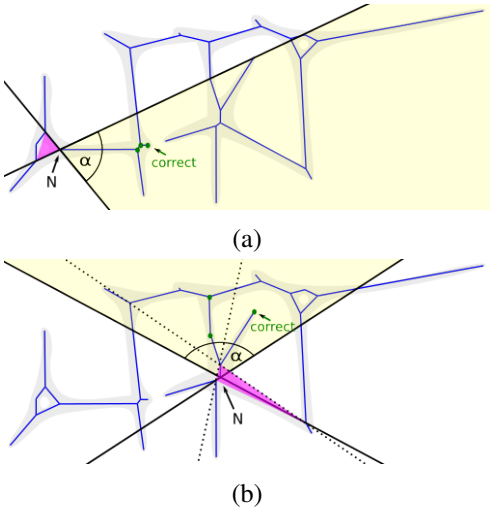


Figure 8: The pink area shows the place where wedge vertices may be located given the wedge-head of (a) a contour wedge and (b) a solid wedge. The marked junctions are valid as wedge vertex since the straight line to N does not cross the shape boundary. The dotted line in (b) shows that if we had chosen N for solid wedges as for contour wedges, we would not be able to find the correct wedge vertex.

candidates that can be easily identified as improbable by computing the angles between the arms. We want the arms to be evenly spread, so we punish angles that deviate from 120 degrees. Having α_1, α_2 and α_3 as internal wedge angles, we use

$$w_{angle}(\alpha_1, \alpha_2, \alpha_3) = p(\alpha_1) \cdot p(\alpha_2) \cdot p(\alpha_3) \tag{18}$$

with

$$p(\alpha) = \frac{120}{120 + |120 - \alpha|} \tag{19}$$

as measure for the angle quality of wedges. This measure is used in a preliminary reduction step to eliminate all wedges whose angle quality exceed a given threshold.

The simplest strategy using a *threshold* takes the remaining wedges and removes those that share heads with other wedges having higher angle quality. All of the other strategies proceed by iteratively testing wedges against the set of chosen wedges and adding them to the result set if no conflicts arise. Wedges are not added if the number of wedge-head edges, that are not used by any other wedge as head or arm edge, goes below a threshold. Furthermore, contour wedges may not share a head edge with another wedge.

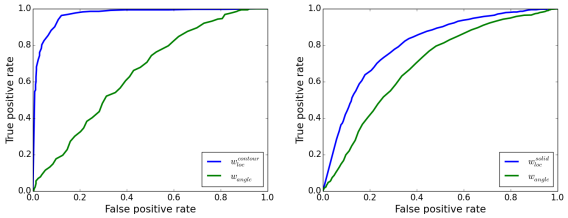
Balanced Strategies For documents where the number of solid wedges is greater or equal to the number of contour wedges, we implemented balanced strategies. There are six different strategies of this kind: *Balanced-loc*, *balanced-angle* and *balanced-size* sort the wedges by w_{loc}, w_{angle} and size respectively. *Balanced-sides-loc*, *balanced-sides-angle* and *balanced-sides-size* sort the wedges first by the number of arms that contain at least one edge that is not already used by a chosen wedge. The second kind of balanced strategies recalculates the number of free arms after each iteration. As measure for the size of a wedge, the average length of the lines from the center to the three edges is taken.

Contour-Fill Strategies Most documents contain more contour wedges than solid wedges. For these documents, the contour-fill strategies have been implemented. The strategies are *contour-fill-loc*, *contour-fill-angle* and *contour-fill-size*, *contour-fill-sides-loc*, *contour-fill-sides-angle* and *contour-fill-sides-size*. They proceed as their respective balanced counterpart but consider the set of contour wedges first before adding solid wedges to the set of chosen wedges. The candidate set of solid wedges only calculated after the set of chosen contour wedges is computed and vertices that are incident to a wedge-head edge are excluded.

4. Results

The algorithm has been tested on 94 tracings from [23] and [14]. The groundtruth is determined by manually deciding for each cycle and skeleton junction if they are valid positions of wedge-heads. Since a tracing rarely contains less than 500 wedge-marks, two typical tracings have been chosen for the evaluation. They differ in the representation of fractures, in size and in the percentage of solid wedges. As result we have 1252 annotated cycles and 3792 annotated junctions serving as groundtruth.

For the evaluation of the discrimination capabilities of w_{loc} and w_{angle}, the Receiver Operating Characteristic (ROC) [9] will be used. The ROC shows the quality of a detector by assigning it a point in the ROC space, with the false positive rate (FPR) as x-coordinate and the true positive rate (TPR) as y-coordinate. Therefore, the point assigned to an optimal detector has the coordinates (0,1). As measure for the overall performance of a discriminator function, the F-score is given [24].



(a) (b)

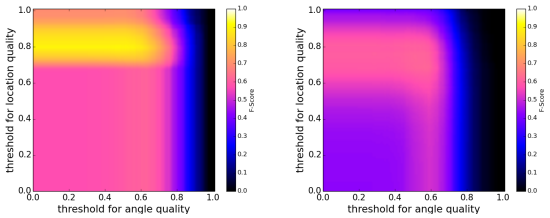
Figure 9: ROC curves of the measures used for the detector of (a) contour wedges and (b) solid wedges

Contour Wedges Candidates for contour wedges are found by searching for cycles in the skeleton graph. The similarity of the cycle to a triangle is then used as quality measure for the detector for early rejection of improbable locations for wedge heads. Figure 9a shows that early rejection is reasonable, since the chosen function for the location proves to be a good estimator. However, the green curve shows that the chosen measure for the angles between the arms of a reconstructed wedge is less optimal as discriminator.

Figure 10a shows the F-score for the contour wedge detector using thresholding only. It shows high scores of about 0.8 to 0.9 for thresholds for the location quality of about 0.7 to 0.9. The threshold for the angle quality should not be chosen too high, but 0.7 at most. The maximum score of 0.90 is achieved for $t_{loc}^{contour} = 0.79$ and the threshold $t_{angle} = 0.45$.

Solid Wedges Candidates for solid wedges are found by searching for skeleton junctions with great distance to the shape contour. Figure 9b shows that weight is a worse discriminator for solid wedges than for contour wedges. The discrimination quality for the angle quality measure for solid wedges looks very similar to the respective curve for contour wedges.

Figure 10b shows the F-scores for the solid wedge detector. For solid wedges, this detector achieves a score of about 0.62 at maximum. In contrast to the F-scores for the contour wedge detector this score is quite low. The reason for this can be found in the fact, that there are a lot more locations to check since every skeleton junction is considered. Especially when junctions are located next to each other, false hits occur frequently. The best score is achieved for $t_{loc}^{solid} = 0.68$ and $t_{angle} = 0.56$.



(a) (b)

Figure 10: F-scores for detector of (a) contour wedges and (b) solid wedges.

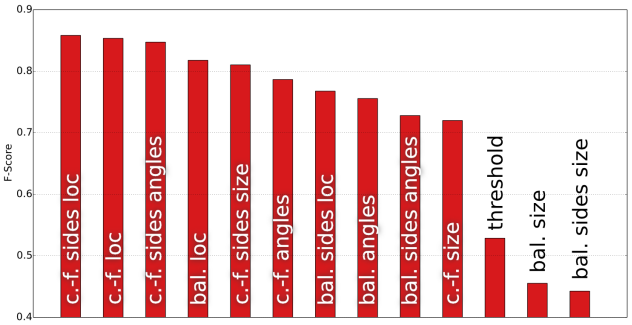


Figure 11: F-scores of reduction strategies for test case VAT6546. The strategies are sorted in descending order by their performance.

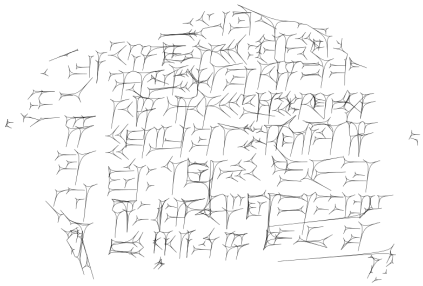
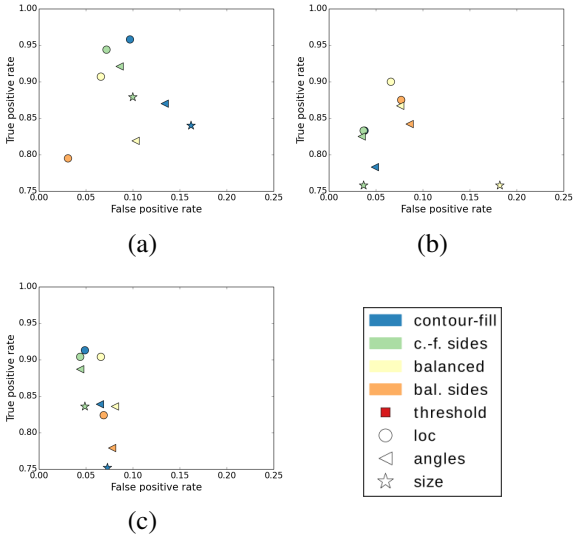


Figure 12: The extracted wedge marks of test case VAT6546.

4.1. Wedge Set Reduction

The reduction strategies serve to overcome the shortcomings of pure thresholding. We demonstrate their differences with a tracing of the tablet VAT6546 [23]. It represents fractures with lines and shows 215 contour and 120 solid wedges. Figure 11 shows the F-scores for this case. The best F-score is achieved by the *contour-fill-sides-loc* method with 86% (Figures 12 and 11).

Figure 13 compares the strategies concerning TPR and FPR. It shows a clear ordering between similar methods that differ merely in the measure for the sorting of the wedges.



(a) (b) (c)

Figure 13: Receiver Operating Characteristic (ROC) space showing performance of wedge set reduction strategies for test case VAT6546 (a) for contour wedges and (b) for solid wedges and (c) for all wedges.

5. Summary and Outlook

In this work we presented an algorithm that uses bitmap tracing and skeletonization as intermediate steps to detect wedge impressions in raster graphics of cuneiform documents. We have shown the weaknesses of the measures used to construct an initial wedge set and have shown how we can use conflict set reduction strategies to improve the results significantly.

This work is part of ongoing research on optical character recognition for cuneiform characters [3] and used as one of many sources of wedge constellations. The presented method will allow us to perform word spotting on raster image databases as the CDLI. We will also examine if statistic approaches as in [5] or [8] can be used to enhance the detection results.

References

[1] C. Barber, D. Dobkin, and H. Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Transaction on Mathematical Software (TOMS)*, 22(4):469–483, 1996. 3

[2] H. Blum. A Transformation of Extracting New Descriptors of Shape. In *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967. 3

[3] B. Bogacz, J. Massa, and H. Mara. Homogenization of 2D & 3D Document Formats for Cuneiform

Script Analysis. In *Proc. of the 3rd International Workshop on Historical Document Imaging and Processing (HIP15)*, 2015. 8

[4] B. Braden. The Surveyor's Area Formula. *The College Mathematics Journal*, 17(4):326–337, 1986. 5

[5] M. Cammarosano, G. Müller, D. Fisseler, and F. Weichert. Schriftmetrologie des Keils: Dreidimensionale Analyse von Keileindrücken und Handschriften. *Die Welt des Orients*, 44(1):2–36, 2014. 8

[6] R. Cosgriff. Identification of Shape. ASTIA AD 254 792 820-11, Ohio State University Research Foundation, 1960. 2

[7] E. Di Sciascio, F. Donini, and M. Mongiello. A Knowledge Based System for Content-based Retrieval of Scalable Vector Graphics Documents. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 1040–1044, 2004. 2

[8] D. Edzard. Keilschrift. In *Ĭa... - Kizzuwata*, volume 5 of *Reallexikon der Assyriologie und vorderasiatischen Archäologie*, pages 545–567. de Gruyter, 1980. 8

[9] T. Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. 7

[10] D. Fisseler, F. Weichert, G. Müller, and M. Cammarosano. Towards an interactive and automated script feature analysis of 3D scanned cuneiform tablets. In *The 4th Conference on Scientific Computing and Cultural Heritage (SCCH)*, pages 1–10, 2013. 2

[11] D. Fisseler, F. Weichert, G. Müller, and M. Cammarosano. Extending Philological Research with Methods of 3D Computer Graphics Applied to Analysis of Cultural Heritage. In *12th Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, pages 165–172, 2014. 1

[12] J. Gravesen. Adaptive Subdivision and the Length and Energy of Bézier Curves. *Computational Geometry*, 8(1):13–31, 1997. 3

[13] H. Hameeuw and G. Willems. New Visualization Techniques for Cuneiform Texts and Sealings. *Akkadica*, 132(2):163–178, 2011. 1

[14] S. Jakob. *Die mittelassyrischen Texte aus Tell Chuēra in Nordost-Syrien*, volume 3 of *Ausgrabungen in Tell Chuēra in Nordost-Syrien*. Harrassowitz, 2009. 7

[15] J. Kantel, P. Damerow, S. Köhler, and C. Tsouparopoulou. 3D-Scans von Keilschrifttafeln – ein Werkstattbericht. In *26. DV-Treffen der Max-Planck-Institute*, pages 41–62. Gesellschaft für wissenschaftliche Datenverarbeitung, 2010. 1

[16] D. Kirkpatrick. Efficient Computation of Continuous Skeletons. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science*, pages 18–27, 1979. 3

[17] S. Kumar, D. Snyder, D. Duncan, J. Cohen, and J. Cooper. Digital Preservation of Ancient Cuneiform Tablets Using 3D-Scanning. In *Proceedings of Fourth International Conference on 3-D Digital Imaging and Modeling*, pages 326–333, 2003. 1

[18] M. Kuntz. Clustering SVG Shape. In *8th International Conference on Scalable Vector Graphics*, 2010. 2

[19] D. Lee. Medial Axis Transformation of a Planar Shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 4(4):363–369, 1982. 3

[20] H. Mara and S. Krömker. Vectorization of 3D-Characters by Integral Invariant Filtering of High-Resolution Triangular Meshes. In *12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 62–66, 2013. 2

[21] H. Mara, S. Krömker, S. Jakob, and B. Breuckmann. GigaMesh and Gilgamesh – 3D Multiscale Integral InvariantCuneiform Character Extraction. In *Proceedings of the 11th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, 2010. 2

[22] U. Montanari. Continuous Skeletons from Digitized Images. *Journal of the ACM (JACM)*, 16(4):534–549, 1969. 3

[23] O. Neugebauer, editor. *Register, Glossar, Nachträge, Tafeln*, volume 2 of *Mathematische Keilschrift-Texte*. Springer, 1935. 1, 7, 8

[24] D. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011. 7

[25] F. Preparata. The Medial Axis of a Simple Polygon. In *Mathematical Foundations of Computer Science 1977*, pages 443–450. Springer, 1977. 3

[26] W. von Soden. *The ancient Orient: an introduction to the study of the ancient Near East*. Wm. B. Eerdmans Publishing Co., 1994. 1

[27] L. Watkins and D. Snyder. The Digital Hammurabi Project. In *Proceedings of Museums and the Web (MW)*, 2003. 1

[28] G. Willems, F. Verbiest, W. Moreau, H. Hameeuw, K. Van Lerberghe, and L. Van Gool. Easy and Cost-Effective Cuneiform Digitizing. In *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, pages 73–80, 2005. 1

[29] C. Zahn and R. Roskies. Fourier Descriptors for Plane Closed Curves. *IEEE Transactions on Computers (TC)*, 21(3):269–281, 1972. 2