 DOI: 10.3217/978-3-85125-528-7-25

# Ridge Point Extraction with Non-Maximum Suppression on Irregular Grids

Richard Schönpflug and Hubert Mara
r.schoenpflug@stud.uni-heidelberg.de
hubert.mara@iwr.uni-heidelberg.de

Ruprecht-Karls-Universität Heidelberg
IWR – Interdisciplinary Center for Scientific Computing
FCGL – Forensic Computational Geometry Laboratory
Klaus-Tschira-Platz, 69120 Heidelberg, Germany

***Abstract***
*Assyriology is the study of cultures related to cuneiform writing, which was used for more than three millennia before Christ in the ancient Middle East. Drawing hundreds of thousands of documents with cuneiform script manually is a tedious task and leads to a demand for automated tools assisting the daily work of assyriologists. The cuneiform script is a handwriting using wedges (Latin: cunei) imprinted into clay tablets. Therefore the digitization of cuneiform tablets is increasingly using 3D-scanners that provide irregular triangular grids in* $\mathbb{R}^3$*. These grids i.e. meshes are discrete manifolds, which are first filtered by using* Multi-Scale Integral Invariants *(MSIIs) for visualization. Secondly the MSII filter results are used to extract points along the or ridges within the 3D-model leading to a digital drawing of e.g. a cuneiform tablet. Therefore we choose the idea of the* non-maximum suppression *as used by the Canny edge detector for raster images. In contrast to the Canny edge detector we had to (i) to adapt to an arbitrary number of neighboring vertices, which have to be reduced locally in case of flat areas; (ii) to implement an estimator for the gradient direction, which cannot be provided by the MSII filter; and (iii) to provide a border treatment as real world meshes have missing parts. All the work was embedded within our modular* GigaMesh *software framework. Results are shown for synthetic and real data, demonstrating a computational complexity of* $O(n)$*, which requires only one parameter. Finally a summary and an outlook are given.*

## 1. Introduction

Cuneiform script was used for more than three millennia before Christ and is one of the oldest known writing systems. It is a handwriting in 3D, where imprints were made into clay tablets, using a reed styli [13]. This results in groups of wedge shaped imprints forming the characters. The name *cuneiform*, originates from the word *cuneus* for wedge. Drawing a replication of the cuneiform tablets is an integral part of their decipherment. This drawing step is traditionally done by manually tracing photographs of the tablets and can take hours or even days. This is an almost impossible task taking into account the hundreds of thousands of unpublished tablets. These tablets are important for many other disciplines as they provide insights into a wide variety of topics ranging from the economics of ancient societies to the first great works of literature, e.g. the epic of *Gilgamesh* [10].

This work is motivated by the task to extract cuneiform characters and other imprinted features out of 3D-models of tablets. The models are acquired using optical scanners based on the principle of structured light [12]. Having a robust filter using *Multi-Scale Integral Invariants* (MSIIs) [7] we extended, the filtering using the principle of the non-maximum suppression as known from the *Canny edge detector* [2]. Therefore we had to extend the algorithm for an arbitrary number of neighboring vertices as there is no fixed number of neighboring pixels/vertices. As MSII filtering does not provide a gradient direction we had to add an estimator using the normals of the triangles (faces) connecting the vertices. To improve robustness we apply a local mesh simplification for flat areas. These processing steps are described within the next sections and are embedded within our modular *GigaMesh* software framework [8, 9], which provides the MSII – and other filter results – as precomputed function values $f(\cdot)$ for irregular meshes. This work is used for further processing to gain high-level knowledge of cuneiform tablets as known from the domain of *Handwriting Text Recognition* (HTR) [1].

## 2. Ridge Tracing on Irregular Grids

The acquired 3D-models consist of meshes described by lists of vertices $\mathbf{p_i} = (x_i, y_i, z_i)^T$ and faces (triangles) $\mathbf{t}_i := \{\mathbf{p}_{A_i}, \mathbf{p}_{B_i}, \mathbf{p}_{C_i}\}$ having an orientation. The mesh is a discrete two-dimensional manifold $\mathcal{M}_2$ in $\mathbb{R}_3$ having orientated edges $\{\mathbf{e}_{a_i}, \mathbf{e}_{b_i}, \mathbf{e}_{c_i}\}$, which are implicitly given by the oriented faces [7]. The orientated faces allow to determine the space enclosed by the mesh. The index $i$ is used to address all the elements of the mesh processed consecutively, while $j$ addresses all elements next to the element with index $i$. Note that computational expensive calculations – especially the MSII filter – are parallelized within *GigaMesh*. The vertices of the 1-ring neighborhood are denoted as $\mathbf{p}_j$ around the central vertex $\mathbf{p_i}$. The 1-ring contains all faces sharing $\mathbf{p_i}$. Additionally each face $\mathbf{t}$ has a normal vector denoted as denoted by $\mathbf{n}$, which are normalized $\hat{\mathbf{n}} = \mathbf{n}/|\mathbf{n}|$ before e.g. computing the dot product $\langle \hat{\mathbf{n}}_i, \hat{\mathbf{n}}_j \rangle$. Furthermore we compute a normal vector $\mathbf{n_i}$ for each vertex $\mathbf{p_i}$ using the normals $\mathbf{n}_j$ of the adjacent faces $\mathbf{t}_j$. Experiments have shown that this approximation is sufficient for our algorithm and more complex methods like normal vector voting [11] are not necessary.

### 2.1. Retrieval and simplification of ordered 1-rings

For the following steps of the non-maximum suppression the vertices next to each other are required to be in the sequence given by the orientation of the edges. Our algorithm then uses the implicitly given adjacencies of the mesh to fetch all faces ot the 1-ring of $\mathbf{p_i}$ following the orientation of the edges, adding the vertices $\mathbf{p_j}$ to a sorted list without duplicates excluding $\mathbf{p_i}$. *GigaMesh* ensures that non-manifold vertices and edges are removed [7, p. 121] before computing the sorted list. If $\mathbf{p_i}$ is a vertex on the border $\partial\mathcal{M}_2$ of the mesh, a second iteration using the opposite orientation of faces' edges is necessary – otherwise an arbitrary number of vertices of the 1-ring will be missing.

As subsets of consecutive vertices $\mathbf{p_j}$ can be on a plane the 1-ring has to be simplified to provide a robust tracing of ridge points. For this reason each subset of consecutive vertices are reduced to one representative vertex denoted as $\mathbf{p}'$ in the following example, which is shown in Figure 1. It shows consecutive vertices $\{\mathbf{p}_5, .., \mathbf{p}_9\}$, which are located together with $\mathbf{p}_i$ in one plane, i.e. the faces defined by those vertices have the same direction of their normals. Theoretically we can detect flat parts within the 1-ring by pairwise computing the dot product of adjacent triangles' normals. Such sets of triangles could be replaced by one bigger triangle. As triangle normals can only provide gradient directions within its 1-ring, we have chosen to use the vertex normals, which can store arbitrary normals computed from a range of methods, e.g. a weighted average or computed using

normal vector voting. Therefore the dot products of consecutive pairs of $\{\hat{\mathbf{n}}_5, .., \hat{\mathbf{n}}_9\}$ is computed, where values of $\approx 1$ indicate flat parts. The color map represents the distance to the $xy$-plane to show the three-dimensional nature of the 1-ring. Figure 1b shows that the neighboring vertices $\mathbf{p}_4$ and $\mathbf{p}_1$ are added to the simplified mesh creating a slight artificial valley to generally avoid flat areas having no gradient direction. The threshold determining if these dot products are $\approx 1$ is called $\epsilon$ and it is the only parameter to be set by the user.

The first vertex is stored in the list $\mathcal{L}_{group}$ with label ID 0. For each $\langle \hat{\mathbf{n}}_i, \hat{\mathbf{n}}_j \rangle$ within the range $\epsilon$ to the previous entry then $\mathbf{p_j}$ will be added to $\mathcal{L}_{group}$ with the same label ID. If not, the label ID will be incremented before inserting the item. The algorithm continues until all vertices $\mathbf{p_j}$ in the 1-ring are processed. When all items are processed, the dot product of the first and last entry of the adjacent vertices list needs to be compared because they are contiguous. If the condition to group the two vertices is met, the label ID of all elements with the current label is changed to 0. Now all adjacent vertices are traversed and a new vertex is created for every label, which is assigned the average function value, position vector and normal vector of the corresponding vertices. The grouping process is equivalent to a run-length encoding. In Figure 1b, this results in the new vertex $\mathbf{p}'$ which is the average of vertices $\mathbf{p}_5$ to $\mathbf{p}_9$. The reduced 1-ring has to contain at least 3 vertices to be a manifold otherwise $\mathbf{p_i}$ is not further considered to be a maximum. In case $\mathbf{p_i}$ is a border vertex the minimum amount of required vertices in the 1-ring is 2.

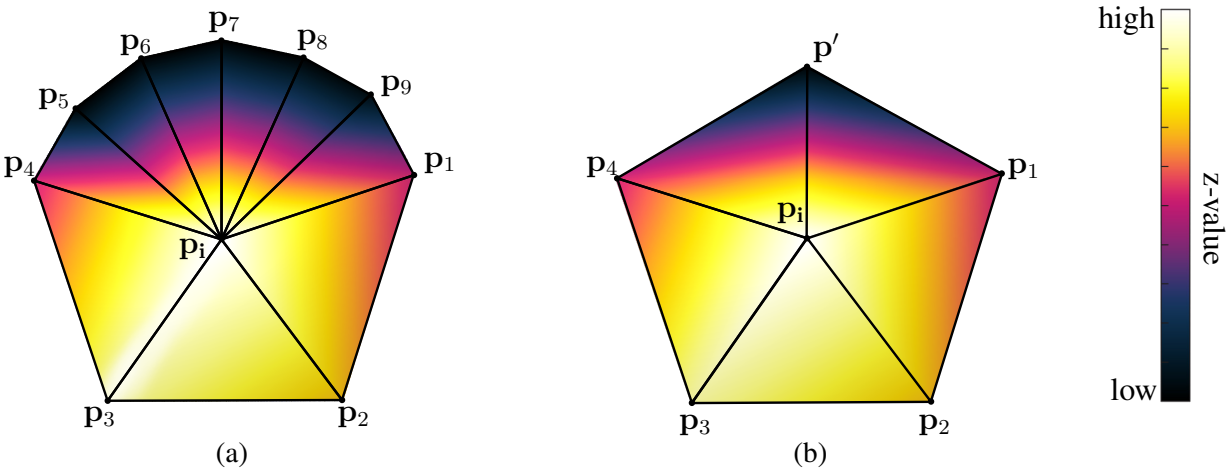



**Figure 1**: Example of the mesh simplification process. (a) The contiguous vertices $\mathbf{p}_5$ to $\mathbf{p}_9$ lie on a plane. (b) The related faces between vertices have been grouped, resulting in the new vertex $\mathbf{p}'$.

### 2.2. Principal direction of the gradient value $f(\cdot)$

Analogously to the Canny algorithm, we have to compute the principal direction $\mathbf{t}$ of the gradient. As we typically use the MSII-filter for $f(\mathbf{p_i})$, we have to use the normals to detect $\mathbf{t}$ and its orthogonal secondary direction $\mathfrak{b}$. To achieve this, the dot product $\langle \hat{\mathbf{n}}_\mathbf{i}, \hat{\mathbf{n}}_\mathbf{j} \rangle$ is computed. The vertex $\mathbf{p_j}$ with the largest dot product is the principal direction $\mathbf{t}$ and is saved for later computations. This is illustrated in Figure 2a with $\mathbf{t} = \mathbf{p}' - \mathbf{p_i}$. In Figure 2b $\pm\mathfrak{b} = \pm\mathbf{t} \times \hat{\mathbf{n}}_\mathbf{i}$ is shown. The normal, the principal, and the secondary direction span a *Frenet-Serret frame* (TNB frame) with the planes $\boldsymbol{\tau}_{nt}$ and $\boldsymbol{\tau}_{nb}$.

According to Canny we need the gradient values $\mathbf{p}$ and $\mathbf{q}$ on the secondary directions $\pm\mathfrak{b}$. These are found on the intersections $\mathbf{p}_{\overline{jk}} := \boldsymbol{\tau}_{\mathfrak{bt}} \cap \mathbf{e_{jk}}$ and $\mathbf{p}_{\overline{lm}} := \boldsymbol{\tau}_{\mathfrak{bt}} \cap \mathbf{e_{lm}}$. To compute $f(\mathbf{p}_{\overline{jk}})$ we interpolate linear between the two vertices $\mathbf{p}_j$ and $\mathbf{p}_k$ with the respective function values $f(\mathbf{p}_j)$ and $f(\mathbf{p}_k)$.

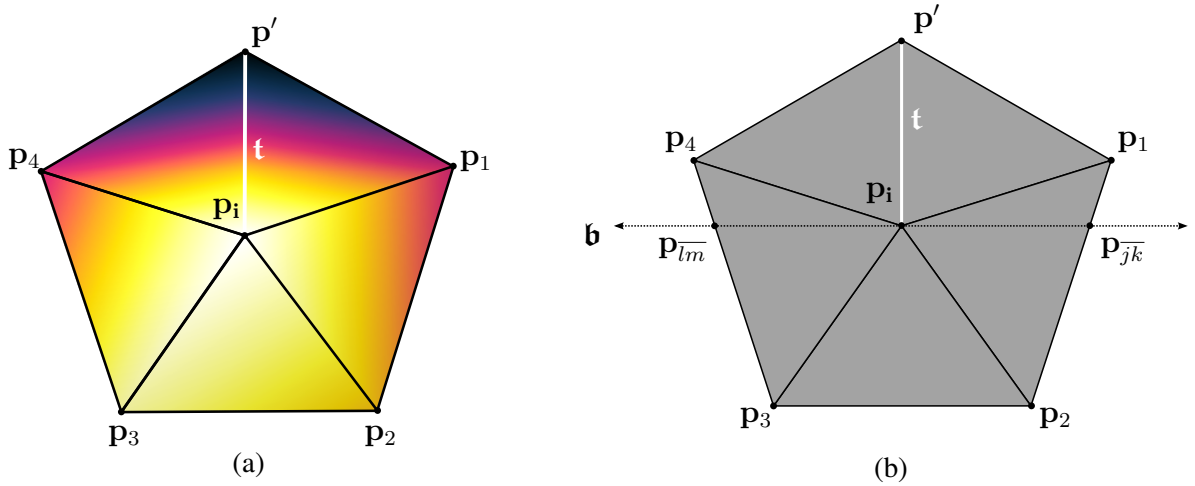



**Figure 2**: (a) The vector $\mathfrak{t}$ describes the principal direction outbound from $\mathbf{p_i}$. (b) The vertices on the orthogonal secondary direction $\mathfrak{b}$ are $\mathbf{p}_{\overline{jk}} = \mathbf{p}_{\overline{12}}$ and $\mathbf{p}_{\overline{lm}} = \mathbf{p}_{\overline{34}}$.

### 2.3. Non-maximum suppression with border treatment

Finally we distinguish vertices being maxima from those being non-maxima:

- If either $f(\mathbf{p}_{\overline{jk}})$ or $f(\mathbf{p}_{\overline{lm}})$ is larger than $f(\mathbf{p_i})$, then $\mathbf{p_i}$ is suppressed by discarding this vertex.
- Otherwise $\mathbf{p_i}$ is a maximum and added to the list $\mathcal{L}_{max}$.

In case $\mathbf{p_i}$ is on the border $\partial\mathcal{M}_2$, we treat the vertex by checking the existance of the edges $\mathbf{e}_{jk}$ and $\mathbf{e}_{lm}$ intersecting the plane $\boldsymbol{\tau}_{nb}$. For existing edges we proceed as described above. Otherwise we have to choose a function value of $\mathbf{p_j}$ close to $\boldsymbol{\tau}_{nb}$: If there is an edge $\mathbf{e}_{ij}$ with $\langle \hat{\mathbf{e}}_{ij}, \mathfrak{b} \rangle > 0$ we choose the function value $f(\mathbf{p}_j)$ of the edge having the dot product closest to 1. Having no positive value for the dot product leads to suppression of the vertex. This procedure is repeated using $-\mathfrak{b}$ for the second secondary direction.

## 3. Results

The execution times for various real world and synthetic test cases behave linear, depending on the number of vertices of the mesh. This heuristically determined computational complexity of $O(n)$ with $n$ being the number of vertices is shown in Table 1. The resulting ridge points on a detail of a three-dimensionally acquired cuneiform tablet is shown in Figure 3. These selected points can be exported using the current view and its underlying *OpenGL* projection matrix within *GigaMesh* either as perspective or as orthogonal projection. The latter is true to scale assuming a calibrated 3D-model. While the surfaces are rendered as raster images, the ridge points are exported as overlays using the *Scalable Vector Graphics* (SVG) [4] file format, which describes their exact location using the *eXtensible Markup Language* (XML), commonly used within the *Digital Humanities*.

Results on a high resolution data set are shown in Figure 4a. Due to the high density of vertices, the algorithm responds to small disturbances i.e. noise of the surface, leading to false positives. These can be eliminated by smoothing the surface prior to the application of our algorithm. In Figure 4b a combination of Taubin and TwoStep smoothing was applied using *MeshLab* [3]. This increase in robustness behaves – as expected – like the Canny edge detector, which has a smoothing step as a prerequisite.

| Data set | Type | Vertices | Runtime |
|---|---|---|---|
| Chars4Testing | measured | 10,492 | 0.044s |
| cuneus_ideal | synthetic | 15,521 | 0.071s |
| Half4Testing | measured | 282,428 | 0.789s |
| HOS_G10_Preview | measured | 371,711 | 1.809s |
| VAT 10908 | measured | 3,034,899 | 12.269s |
| HOS_G10_Full | measured | 6,596,964 | 33.319s |

**Table 1**: Performance of the algorithm on multiple data sets. Dataset name, type, number of vertices and the respective runtime are given.

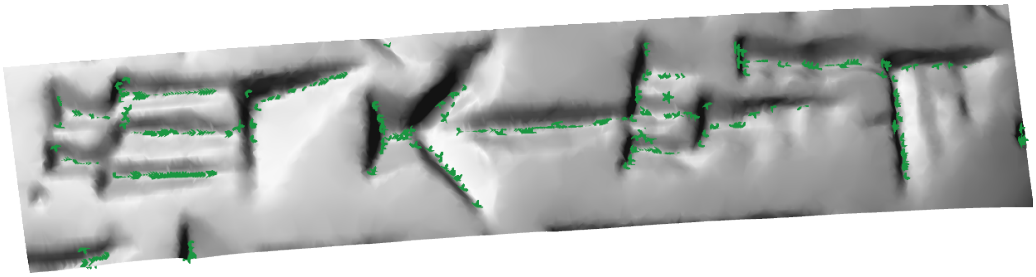
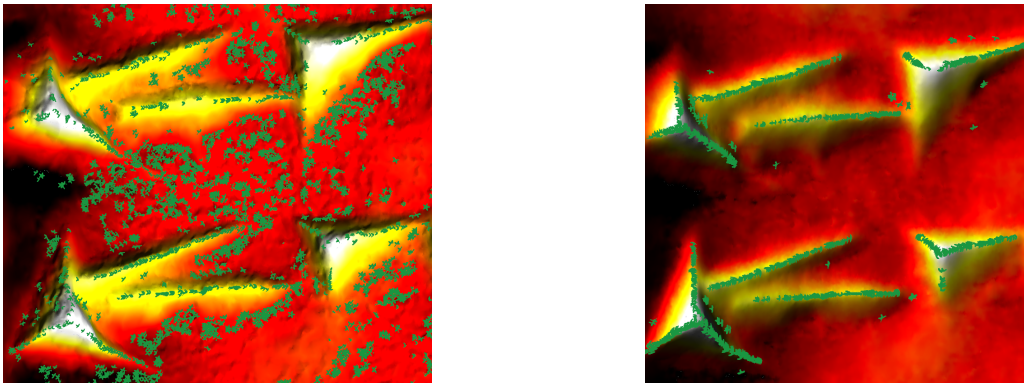


**Figure 3**: Detected ridge points in the real world data set Chars4Testing. It can be seen that the points follow the ridges of the mesh nicely.



(a) (b)

**Figure 4**: High resolution dataset HOS_G10_Full (a) before and (b) after smoothing.

## 4. Outlook and Summary

Future enhancements of our algorithm are the implementation of a marching front to connect the ridge points to lines, making them exportable as SVG. Following the Canny approach, hysteresis tracking is a future extension providing an even more robust selection of feature points. Furthermore smoothing of the function values instead of smoothing the mesh will improve the performance by reducing the computational overhead of processing $\mathcal{M}_2$. The final vision is to have a completely autonomous system, which begins transcribing the ancient tablets immediately after their acquisition, exporting the digital drawings with automated annotations directly into a searchable database [6].

The algorithm implemented in this work succeeds in extracting ridge points from irregular grids, using non-maximum suppression. Although the execution on an irregular surface mesh architecture

contains various challenges, all of them could be resolved. The most important challenges were the mesh simplification step and the determination of the maximum gradient direction. We could show the adaptation of the Canny edge detector, used on regular grids to irregular triangular meshes in $\mathbb{R}^3$. The necessary user input is kept to a minimum, namely only one parameter, which controls the strength of the local and temporary mesh simplification. In general our algorithm delivers robust approximations with high performance used for further processing with methods from machine learning [5].

## References


[1] B. Bogacz, M. Gertz, and H. Mara. Character Retrieval of Vectorized Cuneiform Script. In *Proc. of Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 326–330. IEEE, 2015.

[2] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[3] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In *Italian Eurographics Conf.*, pages 129–136, 2008.

[4] J. David Eisenberg. *SVG Essentials*. O'Reilly, 1 edition, 2002.

[5] D. Fisseler, F. Weichert, G. Müller, and M. Cammarosano. Towards an Interactive and Automated Script Feature Analysis of 3D Scanned Cuneiform Tablets. In *Proc. of the 4th Scientific Computing and Cultural Heritage*, Heidelberg, Germany, 2013.

[6] B. Groneberg, F. Weiershäuser, T. Linnemann, and D. Ullrich. Digitale Keilschriftbibliothek Lexikalischer Listen aus Assur. In *Max-Planck-Gesellschaft – Jahrbuch*. Max-Planck-Gesellschaft, 2005.

[7] H. Mara. *Multi-Scale Integral Invariants for Robust Character Extraction from Irregular Polygon Mesh Data*. PhD thesis, Ruprecht-Karls-Universität, Interdisciplinary Center for Scientific Computing (IWR), Heidelberg, Germany, 2012.

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

[9] H. Mara, S. Krömker, S. Jakob, and B. Breuckmann. GigaMesh and Gilgamesh - 3D Multiscale Integral Invariant Cuneiform Character Extraction. In A. Artusi et. al., editor, *Proc. VAST Int. Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pages 131–138, Paris, France, 2010. Eurographics Association.

[10] S.M. Maul. *Das Gilgamesch-Epos*. Beck, 2005.

[11] D. L. Page, Y. Sun, A. F. Koschan, J. Paik, and M. A. Abidi. Normal Vector Voting: Crease Detection and Curvature Estimation on Large, Noisy Meshes. *Graphical Models*, 64(3–4):199–229, 2002. Special issue: Processing on large polygonal meshes.

[12] R. Sablatnig and C. Menard. Stereo and Structured Light as Acquisition Methods in the Field of Archaeology. In *Mustererkennung 1992*, pages 398–404. Springer, 1992.

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