Vol. 12, No. 1, pp. 36 – 47

Feature Extraction and Modification for Illustrating 3D Stone Tools from Unorganized Point Clouds

Enkhbayar Altantsetseg[1)] **Yuta Muraki**[2)] **Katsutsugu Matsuyama**[2)]
Fumito Chiba[3)] **Kouichi Konno**[2)]

1) Graduate School of Engineering, Iwate University, Japan
2) Faculty of Engineering, Iwate University, Japan
3) Laboratory for Archaeology and Geoinformatics, LANG Co., Ltd, Japan

[1),2)]{bayar@lk., murakiyuta@lk., kmatsu@, konno@}cis.iwate-u.ac.jp
[3)]f-chiba@lang-co.jp

Abstract
This paper presents a method for extracting and modifying features for illustrating stone tools. Features are detected from unorganized point clouds obtained by a 3D laser scanner. The curvature of each point in the point clouds is estimated by local surface fitting algorithm and used for detecting potential feature points. Feature lines are extracted by directionally growing algorithm. Our directionally feature line growing method is simple to detect feature lines from unorganized point data. The main idea of our method is to extract feature lines using principal curvatures and principal directions of the potential feature points along the axis directions and to merge all extracted lines. In the illustration of stone tools, to maintain form and manufacturing information, it requires to modify the detected features by specific knowledge on illustrating stone tools. Using the power of data parallel computation on the GPU, our method can be accelerated multiple times. Finally, our experimental results are compared to the actual illustrations drawn by archaeological illustrators.

Keywords: feature extraction, feature modification, point clouds, stone tools, GPGPU.

1 Introduction

In recent years, 3D digitizing technology is widely used for conserving and studying cultural heritages. Laser scanning technology allows us to visualize and illustrate highly accurate archeological artifacts [15, 19, 25]. In this paper, we will introduce a new method to extract and modify features of stone tools for illustrating them from unorganized point clouds obtained by a 3D laser scanner. The study of stone tools is one of the important subjects in cultural heritage because stone tools are the common evidences of the prehistoric human culture. On the other hand, illustration of stone tools is important in archaeological reports, publications and study of stone tools.

Stone tools used by early humans were not of a formal form and have sharp edges as well as rough surfaces such as dents and cracks on the surfaces. The illustration of stone tools requires to convey the information on manufacture, modification and usage of tools and takes a long time because of the complexity of tools [18].

Illustrators use view dependent and view independent lines for illustrating archeological artifacts. View dependent lines are lines dependent on viewing directions such as silhouette lines, suggestive outlines and apparent ridges. View independent lines are lines that are not changed with respect to viewing direction such as ridge and valley lines and hatching lines. In our work, we considered only ridge and valley lines that are valuable and important candidate for the illustration.

For extracting ridge and valley lines, the principal curvatures and principal directions of each point are computed by local surface fitting algorithm [4, 9]. Potential feature points are detected by the curvature of points.

In point cloud slicing technique, feature lines are extracted from detected potential feature points. The main idea of our method is to grow feature lines along the axis directions by using the principal curvatures and principal directions of feature points.

There are several techniques to extract feature lines from detected feature points. [7, 11, 23] compute minimum spanning tree to extract feature lines. This technique is capable of extracting feature lines by connecting existing feature points. Therefore, high quality of point cloud is required to get robust and accurate result. [5, 22] use projection based method for extracting features from detected potential feature points. [5] projects potential feature points to the intersections of locally fitted surfaces. After smoothing projected points, polylines are grown through the projected points. [22] smoothes feature points by projecting them onto their principal axis of neighborhoods.

In contrast to previous methods our method does not requires any smoothing operation and extracts feature lines directly from potential feature points. Our method is suitable for parallel computation.

The extracted feature lines are insufficient to convey form and manufacturing information of stone tools because of their surface complexity. Thus, in order to illustrate stone tools, we need extra modifications on the feature lines.

In practice, since it requires to illustrate thousands of stone tools, it is very time consuming. Illustrating stone tools directly from point clouds rather than surface reconstruction can reduce time consumption drastically.

Our main contributions are as follows:

- We propose a simple algorithm for extracting feature lines from detected potential feature points. The main idea of our algorithm is to extract feature lines along the axis directions by using the principal curvatures and principal directions of points;
- We modify the detected feature lines according to the specific knowledge on illustrating stone tools to represent form and manufacturing information [2];
- We accelerate time consuming steps of our algorithm using data parallel computation concept on the GPGPU.

The rest of our paper is organized as follows. Section 2 introduces the related works. In section 3, we describe our method including potential feature point detection, feature line extraction and modification algorithms. Section 4 introduces acceleration of our algorithm on the GPU. Section 5 shows experimental results including comparison results of the archaeological illustrator. Finally, we conclude this paper in section 6.

2 Related work

Our problem is closely related to feature detection and extraction problems, which have im-

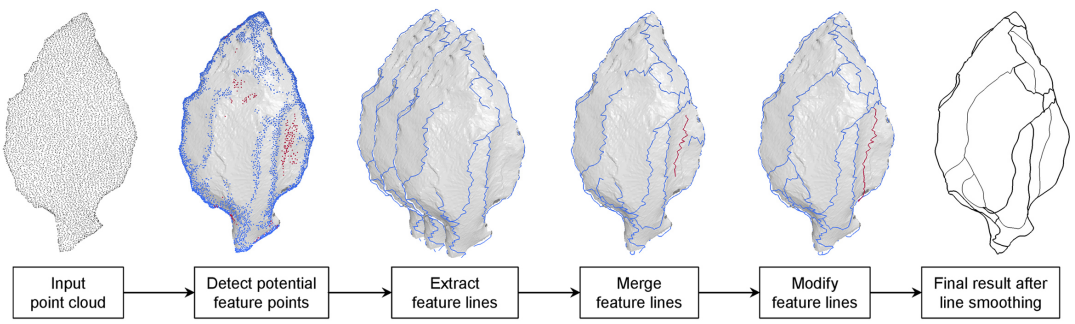

Figure 1: Feature extraction and modification pipeline.

portant application in visualization, geometric modeling and image analysis. There are two kinds of methods: one is point based methods and the other is triangle mesh based methods.

2.1 Point-based feature extraction methods

Our algorithm belongs to point-based methods. Feature extraction from point data is not a simple task because there is no information on connectivity and normal vectors. Most algorithms consider only line-type features. For example, [11] uses Riemann graph to build connectivity information in point data and uses principal component analysis to determine feature points. [23] extends this approach with multi-scaling analysis of the neighborhoods. Both methods use the minimum spanning tree to extract feature lines.

[7] uses segmentation to identify the regions of sharp features and to process them as a graph to detect the feature lines. [28] detects sharp features from point data by computing Gauss map clustering on local neighbors.

[5, 6] detect feature points by locally fitting robust moving least squares polynomial. They project detected potential feature points to the intersections of multiple surfaces then grow poly-lines through the projected points. [22] is similar to above method, but the feature points are smoothed by projecting them onto their principal axis of neighborhoods.

In [13], features are detected by local surface curvatures. Local description of a surface is generated by fitting the surface to neighbor points and estimating the curvature at multiple scales. [3] fits the local representation of the manifold using a truncated Taylor expansion. By using the Taylor's expansion, all local geometric quantities such as curvatures are encoded. In their work, it shows that the estimated curvatures converge to the true ones in the case of smooth surface.

[29] introduces the method for salient edge extraction which does not depend on curvature derivatives. This method based on a topological analysis of the principal curvatures. [17] slices 3D point clouds into 2D cross sections and finds feature points for each of the cross sections. The properties of the convex hull and Voronoi diagram of the point cloud are used for extraction of feature points and reconstruction of feature lines.

2.2 Mesh-based feature extraction methods

Many different techniques have been developed to detect feature lines on 3D meshes. [14] introduced a multi-resolution framework and normal based classification operator for feature extraction on triangle meshes. [12] uses anisotropic filtering on higher order surface derivatives to extract smooth feature lines on surface mesh whereas [27] computes the mean and Gaussian curvature by using focal surface.

[30] detects crest lines on triangle meshes. This method based on estimating the curvature tensor and curvature derivatives via local polynomial fitting. [21] detects ridge and valley lines by combining multi-level implicit surface fitting and finite difference approximations whereas [16] used local MLS fitting technique.

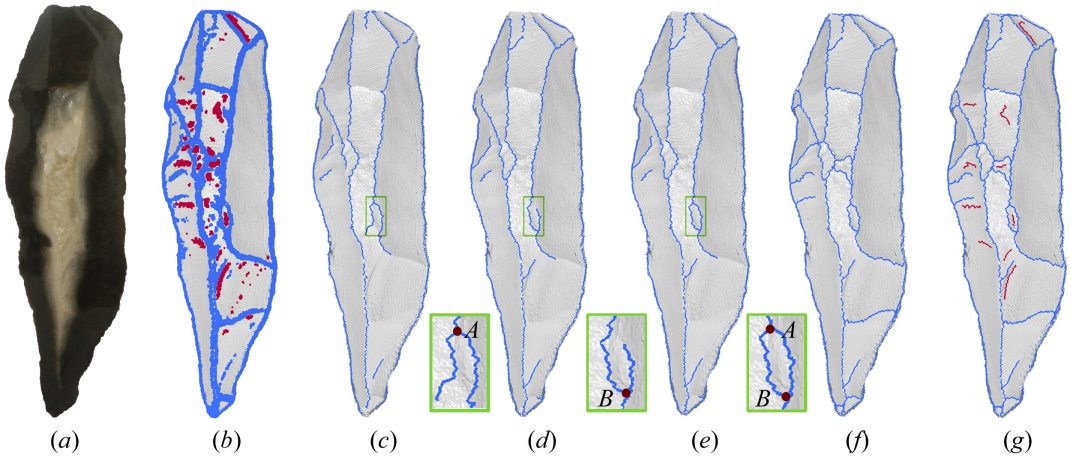

Figure 2: A process of feature line extraction. (a) Original image of the stone knife. (b) Detected potential feature points. Blue dots represent ridge points; red ones represent valley points. (c) Result of ridge line extraction along the $+Y$ axis direction. (d) Result of ridge line extraction along the $-Y$ axis direction. (e) Result of merging two line sets. (f) Extracted and merged ridge lines along $\pm X$ and $\pm Y$ axis directions, (g) Ridge (blue) and valley (red) lines

3 Feature extraction and modification

Suppose that potential feature points are the points whose maximum absolute value of the principal curvatures is greater than the user-defined threshold value σ. Additionally, the potential feature points can be divided into ridge or valley points depending on the sign of curvatures. A line reconstructed from ridge (valley) points is called a ridge line (valley line).

Figure 1 shows the overview of our feature extraction and modification algorithm. In the first stage of the procedure, potential feature points are detected. In the second stage, feature lines are extracted along the axis aligned directions. In the next stage, extracted feature lines are merged. After that, feature lines are modified to illustrate stone tools. Finally, smooth curves are generated from the feature lines.

3.1 Potential feature point detection

Input of our algorithm is unorganized $3D$ point clouds $P = \{p_i \subset R^3\}$ obtained from a laser scanner.

Smoothing First, point clouds are smoothed by using Gaussian smoothing method [26] to robustly detect features of the surface of a stone tool because the surface is rough and the curvature values of points are sensitive to noise. For smoothing point clouds, M nearest neighbors for each point are detected by using KD-tree (degree = 3) and the normal vector of each point is estimated by using Principal Component Analysis (PCA) [10]. Then, project all the neighbors onto tangent plane of each point and select neighbors by angle criterion [20]. In the Gaussian smoothing algorithm, the position of each point is replaced with a convex combination of itself and its selected neighbors.

Surface fitting After the number of iterations of the smoothing, for each point p_i, its normal vector is estimated by [10] and then its selected neighbors are transformed to local coordinate system (u, v, w). In this local coordinate system, point p_i becomes $(0, 0, 0)$, its normal vector lies along the positive w-axis and point p_i and its selected neighbors are fitted by the following cubic polynomial in canonical form [4].

$$w = F(u,v) = \frac{1}{2}(k_1u^2 + k_2v^2) + \\ + \frac{1}{6}(b_0u^3 + b_1u^2v + b_2uv^2 + b_3v^3), \quad (1)$$

where k_1 and k_2 are principal curvatures. If p_i is not umbilical point ($k_1 \neq k_2$), the two vectors $t_1 = (b_0, b_1), t_2 = (b_2, b_3)$ become the principal directions of k_1 and k_2 along their curvature lines, respectively. The selected neighbors of point p_i are obtained by a method described in the section Smoothing.

The coefficients k_1, k_2 and $b_i (i = 0, \ldots, 3)$ of Equation (1) are calculated by [9]. The normal vector of the surface given by Equation (1) is calculated as follows:

$$N(u, v) = (F_u(u, v), F_v(u, v), -1). \quad (2)$$

By using Equations (1) and (2), we can obtain $3 \times n$ linear equations with six unknowns, where n is the number of selected neighbors. These linear equations can be easily solved by standard least square fitting method. Finally, principal curvature values k_1 and k_2, and their principal directions are obtained.

In the next step, set of potential ridge points $P_r \subset P$ and set of potential valley points $P_v \subset P$ are created using the following formulations:

$$\begin{aligned} P_r &= \{p_i | p_i \in P \text{ and } \sigma < max(k_1, k_2)\} \\ P_v &= \{p_i | p_i \in P \text{ and } \\ &\qquad -\sigma > min(k_1, k_2)\}, \end{aligned} \quad (3)$$

where σ is the user-defined threshold value. Figure 2(b) shows detected potential feature points. Blue dots indicate ridge points and red ones indicate valley points.

3.2 Feature line extraction

Shape of stone tools is complex and its surface contains many sharp and smooth features. In order to robustly and accurately extract features, the ridge and valley lines are extracted along all axis directions $(\pm X, \pm Y, \pm Z)$ respectively using the principal curvatures and principal directions of potential feature points. Then, all feature lines derived by all directions are merged.

To extract feature lines along the axis directions, the point cloud is respectively sliced along the X, Y and Z axis directions and sets of slices D_x, D_y, and D_z are created. For example, set

$$D_y = (D_{1,y}, D_{2,y}, \ldots, D_{n,y}) \quad (4)$$

is created by slicing point cloud along the Y axis direction. $D_{i,y}$ is the i^{th} slice of the set D_y. The number of slices of the set depends on user defined slice thickness d.

Consequently, feature lines are grown along the axis directions from slice to slice. To grow feature line, potential feature points of each slice are clustered by distance based clustering and the radius r_{p_i} is computed for each point p_i by Equation (5):

$$r_{p_i} = s_i + d, \quad (5)$$

where s_i is the distance between point p_i and its farthest selected neighbor, d is the slice thickness of the point cloud.

Endpoint $\bar{p}$ of feature line is detected by finding maximum (minimum) curvature point of the cluster. To grow feature line from the endpoint $\bar{p}$ to the next slice, sphere with radius $r_{\bar{p}}$ centered at the endpoint is used according to Equation (5).

The Algorithm 1 shows our technique on ridge line extraction as an examples of extraction along the $+Y$ axis direction.

Algorithm 1. Algorithm for extracting ridge lines along the $+Y$ axis direction.

```
for each slice D_{i,y} ∈ D_y
  find potential ridge points using Eq. 3
  create clusters from potential ridge points
  for each cluster C_j
    if any endpoint p̄ not found in C_j then
      p̄ ← a point with the highest curvature
    end if
    //find a candidate point from next slice
    //to grow ridge line
    while p ∈ D_{i+1,y} and ||p − p̄|| ≤ r_p̄
      register point curvatures
    end while
    p′ ← a point with the highest curvature
    if p′ is found
      β ← compute angle between principal
            direction of p′ and the axis Y
      if β < threshold angle then
        add [p̄p′] to ridge line set
        p̄ ← p′
      end if
    end if
  end for
end for
```

Figure 3 shows different cases for line growing process. In the case of Figure 3(a), the sphere centered at point $\bar{p}$ contains three potential feature points from the next slice. Point p' has the highest curvature value and the angle between its principal direction vector t and growing axis direction is less than user defined threshold angle $(= \pi/3)$, therefore point p' is selected as an

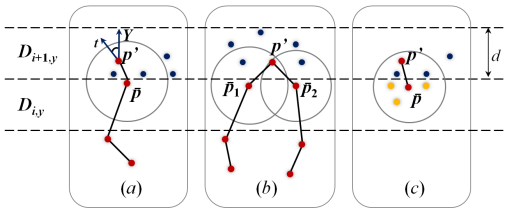

Figure 3: Different cases of feature line growing.

endpoint of the line, and points $\bar{p}$ and p' should be connected. In the case of Figure 3(b), feature point p' is the point with the highest curvature for the spheres centered at points $\bar{p_1}$ and $\bar{p_2}$. Thus both lines will intersect at point p', which will be the endpoint of the lines as well. In Figure 3(c), the cluster of the potential feature points does not contain any endpoint thus point with the highest curvature will be a new endpoint $\bar{p}$.

Figure 2(c) shows the result of the ridge extraction algorithm along the $+Y$ axis direction. There are some missing intersection points of ridge lines depending on the growing directions because our line growing technique can find only intersection point of feature lines which are growing along given the axis direction as well as cannot split feature lines.

In order to find the missing intersection points and to improve the quality of the lines, the same steps are required to implement along the $-Y$ axis direction as shown in Fig. 2(d). Figure 2(e) shows the result of ridge line can be improved by merging two line sets along $+Y$ and $-Y$ axis directions. For example, Figure 2(c) and (d) show our line extraction algorithm, with which intersection points A and B are detected along directions $+Y$ and $-Y$, respectively. After merging two line sets, both intersection points A and B can be illustrated on the ridge lines. Corresponding to the Y axis, the ridge line set LR_y is created, by merging ridges lines extracted along $+Y$ and $-Y$ axis directions.

By repeating this process for all directions, all ridge lines can be detected as shown in Figure 2(f). For the valley lines, overall process is the same as ridge line detection as shown in red lines in Figure 2(g).

After line extraction along all axis directions, ridge line sets LR_x, LR_y, LR_z as well as valley line sets LV_x, LV_y, LV_z are obtained. Therefore the final result is defined by merging these line sets. At the final, unnecessary shorter lines and branches are removed. The length limit of short lines and branches will be given by the user.

3.3 Feature line modification

Stone tools illustration follow a set of conventions that convey the information on manufacture and modification of tools. An outline of the tool is drawn first and next, the outlines of the flake scars are drawn. Flake scars are principal features of stone tools and moderately concave areas where material has been removed. Each flake scar has its own outline. Both the outline of the tool and the outline of the flake scars are drawn with thick line. Once the outlines of the flake scars are drawn in, ripples from the force of the blow are added in lighter lines [18].

Thus detected feature lines are insufficient to convey information of form and manufacturing of tools. Our detected lines should be modified by using the following simple rules:

1. In order to distinguish flake scars from each other, all edges should be connected to other curves.
2. Sharp edge lines and outlines of flake scars are drawn with thick lines.
3. The flake scare carved last is illustrated by smooth curves at the intersection of ridge lines.

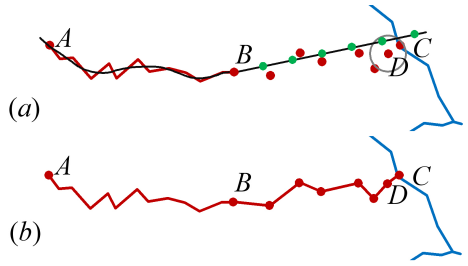

Figure 4: Algorithm of feature line continuity.

Connect lines to other curves There are a lot of open lines and line segments in the extracted lines. In order to make closed area, all open line segments should be detected. After that, each open line segment is fitted by cubic Bezier curve and extended along the tangent vector of the Bezier curve. To extend the open line segments, the tangent line of the Bezier curve at the open end point is created. Then the point on the tangent line will be selected in

a given distance from open end points. For the selected point on the tangent line, its nearest point is detected on the surface of the object. This point will be considered as a new point for extending line segment and it is connected to the open end point of a line segment. This process will be repeated along tangent line until reach to any other feature line. If a sphere centered at the new point contains some points of the feature line, the extended line segment reaches the feature line. Radius of the sphere is given by formula (5).

Figure 4(a) shows the process of line closing. Open line segment AB is shown with red, Bezier curve and its tangent line are represented with black. Green dots represent selected points on the tangent line. The red ones are the new points found on the surface. Point C will be an end point of the line segment since the sphere centered at new point D contains point C. Figure 4(b) shows the result of the closed line segment AC.

Sharp edge To distinguish sharp edges and outlines of flake scars from smooth ridge lines and valley lines, there are illustrated with thick lines. Average curvature value of the each ridge line segment is evaluated by following equation:

$$K(l) = \frac{\sum_{i=1}^{N} C(a_i)}{N},$$

where l is the ridge line segment with N points, $K(l)$ is the average curvature of the line segment l, a_i is the vertex of line segment l, $C(a_i)$ is the corresponding curvature of vertex a_i.

By using a certain threshold value, a sharp edge can be easily determined and is illustrated with a thick line.

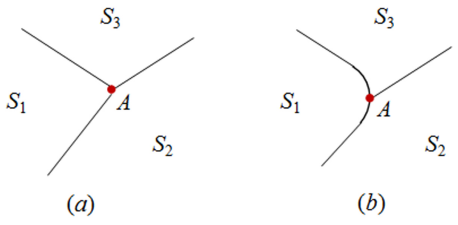

Figure 5: Continuity of lines at intersection point A.

Line continuity at the intersection The continuity of lines at an intersection point preserves important information on stone tool manufacturing such as reduction procedure. When stone tools are illustrated, the flake scare removed last is represented with smooth curves at intersection points. Figure 5(a) shows intersection point A with three ridge lines as an example. The three ridge lines separate areas S_1, S_2 and S_3. If area S_1 is removed last, S_1 will be illustrated with smooth curves (see Figure 5(b)). It is a very important feature in stone tool illustration, but it is very hard to detect the area carved last since the broad experience of illustrators is required. Unfortunately, this problem has not been solved in our method. Our algorithm cannot detect the order of flake scare removing.

Figure 6 shows the results of our algorithm before and after feature line modification.

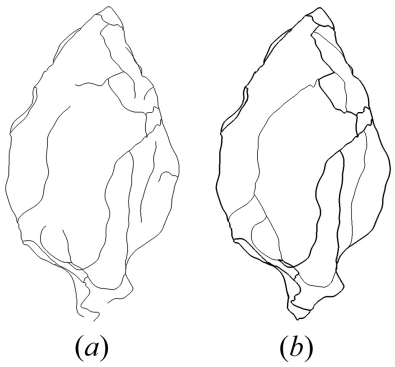

Figure 6: Result of feature line modification process. (a) before modification, (b) after modification.

3.4 Line smoothing

After extracting and modifying the feature lines, the resultant lines look zigzag. These lines are smoothed by a cubic B-spline curve fitting. To fit the lines to a B-spline curve, line segments are extracted from intersections of the lines until no intersection is found in the lines.

Each line segment is fitted to a B-spline curve separately. By solving the following minimization problem, control points for each B-spline curve are derived [24].

$$\min(\sum_{i=1}^{n} |p_i - S(u_k)|^2),$$

where p_i is the vertex points of the line segment, $S(u)$ is the cubic B-spline curve and u_k is pre-

computed parameters. We solve this problem by standard least square fitting method.

4 Acceleration of our algorithm on GPGPU

We use parallel data computation on the GPU to shorten the time consumption. We implemented our approach using OpenCL on the NVIDIA GeForce GTX 570 graphics card. Figure 7 shows the overall procedure of our feature detection algorithm. The blue boxes represent implementation in the GPU and the green ones represent implementation in the CPU.

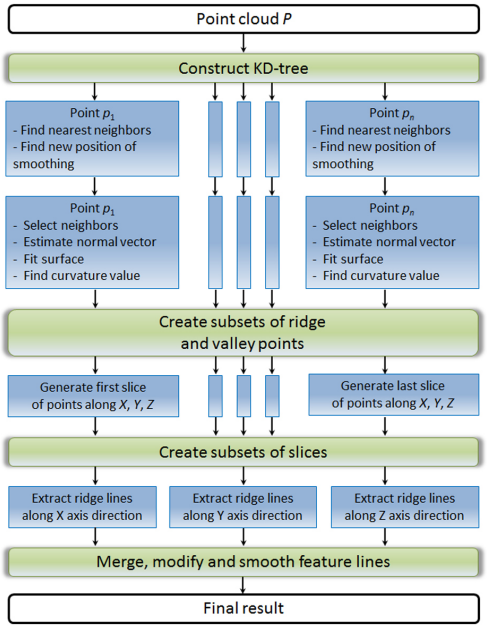

Figure 7: Overall procedure of our algorithm.

First KD-tree (degree=3) is constructed from a point cloud in the CPU. By applying KD-tree construction, we can investigate a certain process in the GPU by timing the OpenCL kernel to find M nearest neighboring points for each point and then find a new position for each point for smoothing. In the second kernel, the curvature of each point is calculated by fitting a local surface. After calculating the curvature for each point, subsets of ridge points P_r and valley points P_v are created in the CPU. The slicing process of the point cloud can be implemented independently for each slice in the GPU. In the last kernel, feature lines are extracted using our line growing method described in section 3.2. Since feature line extraction for each axis direction is independent each other, extraction process can be implemented in parallel.

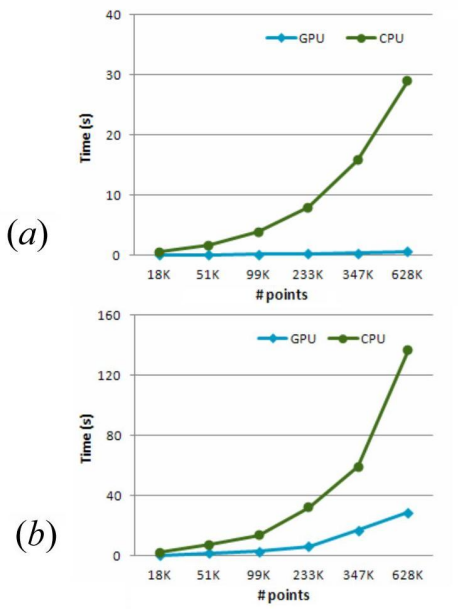

Figure 8: (a) Computation time of surface fitting algorithm. (b) Computation time of overall procedure.

By using a shared memory [1] in our implementation, a remarkable performance gain is obtained. As shown in Figure 8(a), computation time of our surface fitting algorithm is reduced down to $\frac{1}{30}$ times depending on the number of points. Figure 8(b) shows the result of comparison of the computation time for the overall procedure under implementation in GPU and CPU. Our algorithm was implemented for a lot of different objects with up to 650,000 points. More details of the numerical results are shown in the next section.

5 Experimental results and application

This section presents the experimental results. We applied our algorithm to point data of stone tools obtained by the four-directional 3D laser scanners, whose scanning precision is 0.2 mm [8]. The experiment was performed in an Intel Core i5-650 3.2 GHz machine, with 3 GB of RAM and an NVIDIA GeForce GTX 570 graphics card.

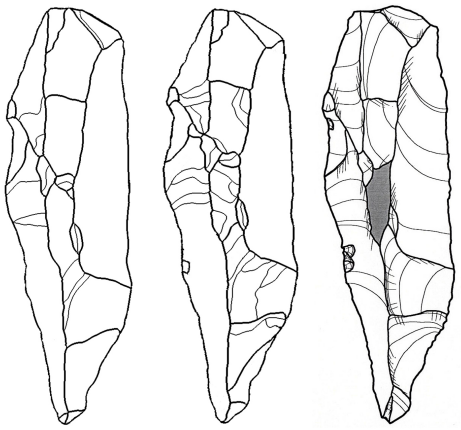

Figure 9: (a) and (b) are results obtained from threshold values of 0.30 and 0.15, respectively and (c) is the result drawn by a professional illustrator.

For the experiments, the following parameters are selected: the number of the nearest points M is 30, the number of smoothing iterations is 5, the angle of criterion α is 35 degrees, the slice thickness d of point cloud is chosen two times of average distance between the point and its nearest neighbor. Point clouds are sliced to create feature lines along the X and Y axis directions only. All the stone tools illustrated in the experiments were thin objects, therefore slicing the point clouds along the Z axis direction is not necessary.

Figure 9 presents the result of illustration of a stone knife (see Figure 2(a)). Figure 9(a) and Figure 9(b) show the results of our algorithm with different threshold values in formula (3) and Figure 9(c) is the result drawn by a professional illustrator. The experimental results show that our algorithm can detect sharp edges well. There are some differences, however, in the smooth ridge and valley lines, because the lines drawn by an illustrator look more artistic than the result of our algorithm. Some hatching lines are also shown in the illustrator's result. Figures 10 to 12 show the results of different stone tools. Table 1 lists the numerical results of implementation of our algorithm for the experimental point data. The timings of the implementation results show that total execution time of our algorithm has been reduced significantly by implementing the computationally most expensive steps in the GPU.

As mentioned earlier in the introduction section, a process of stone tool illustration is very time-consuming. Thus, our feature detection algorithm is very useful for stone tool illustrators. The result of our algorithm cannot be replaced with the result drawn by professional illustrators. However, our algorithm helps to simplify illustrator's tasks and to make results more efficient by detecting hardly visible feature lines. Parallel data implementation allows us to accelerate our algorithm and gives a lot of opportunities to the users. Our method provides precise illustration in almost real time just by changing parameter values interactively.

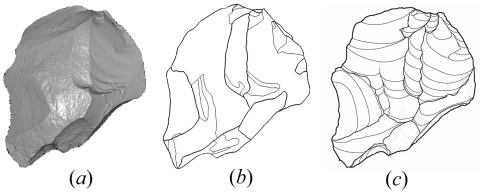

Figure 10: (a) is the 3D model of a tool, (b) is the result of our algorithm and (c) is result of professional illustrator.

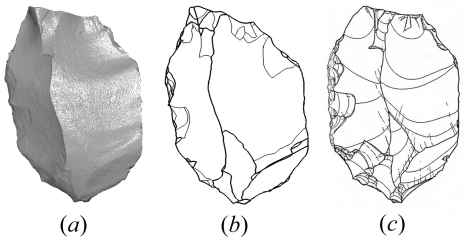

Figure 11: (a) is the 3D model of a tool, (b) is the result of our algorithm and (c) is the result of a professional illustrator.

6 Conclusion

In this paper, we have presented a novel algorithm for extracting features of stone tools. Potential feature points are detected by local surface fitting algorithm. Feature lines are extracted by the directional line growing algorithm. In order to maintain form and manufacturing information of stone tools, the detected features are modified with certain rules used in stone tool illustration. By implementing computation in the GPU, we could accelerate the performance much more. The experimental results show that our algorithm is useful for stone tool illustration.

Table 1: Timings (in seconds) of the implementation steps: (a) smoothing, (b) surface fitting, (c) line extraction, and (d) line modification

Figure	# of points	# of feature points		(a)	(b)	(c)	(d)	Overall procedure
1	18270	4351	CPU	0.866	0.467	0.134	0.033	1.622
			GPU	0.151	0.049	0.096	–	0.497
9	101588	19816	CPU	6.22	3.586	1.22	0.193	12.61
			GPU	0.358	0.242	0.851	–	3.15
12	238290	39564	CPU	12.875	7.7	3.29	0.524	25.89
			GPU	0.673	0.292	2.391	–	6.21

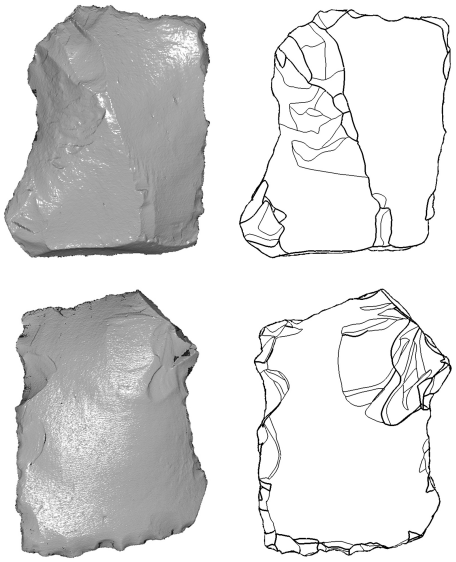

Figure 12: Illustrations of the front and back sides of a stone scraper.

References

[1] M. Aaftab, R. G. Benedict, G. M. Timothy, F. James, G. Dan, OpenCL Programming guide, AddisonWesley Professional, 2011.

[2] L. R. Addington, Lithic illustration: Drawing flaked stone artifacts for publication, The University of Chicago Press, Chicago, Illinois, 1986.

[3] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, Computer Aided Design, Vol. 22, No. 2, pp. 121–146, 2005.

[4] F. Cazals, M. Pouget, Differential topology and geometry of smooth embedded surfaces: selected topics, International Journal of Computational Geometry and Applications, Vol. 15, No. 5, pp.511–536, 2005.

[5] J. II Daniels, L. K. Ha, T. Ochotta, C.T. Silva, Robust Smooth feature extraction from point clouds, In Proceedings of the IEEE International Conference on Shape Modeling and Applications, pp.123–136, 2007.

[6] J. II Daniels, L. K. Ha, T. Ochotta, C. T. Silva, Spline-based feature curves from point-sampled geometry, Visual Comput, Vol. 24, No. 6, pp.449–462, 2008.

[7] K. Demarsin, D. Vanderstraeten, T. Volodine, D. Roose, Detection of closed sharp edges in point clouds using normal estimation and graph theory, Computer Aided Design, Vol. 39, No. 4, pp. 276–283, 2007.

[8] A. Enkhbayar, Y. Muraki, F. Chiba, K. Konno, 3d surface reconstruction of stone tools by using four-directional measurement machine, The International Journal of Virtual Reality, Vol. 10, No. 1, pp. 37–43, 2011.

[9] J. Goldfeather, V. Interrante, A novel cubic-order algorithm for approximating principal direction vectors, ACM Transactions on Graphics, Vol. 23, No. 1, pp. 45–63, 2004.

[10] M. Gopi, S. Krishnan, C. Silva, Surface reconstruction using lower dimensional localized Delaunay triangulation, In Proceedings of the EUROGRAPHICS, Vol. 19, No. 3, pp. 467–478, 2000.

[11] S. Gumhold, X. Wang, R. Macleod, Feature extraction from point clouds, In Proceedings of the 10th International Meshing Roundtable, pp. 293–305, 2001.

[12] K. Hildebrandt, K. Polthier, M. Wardetzky, Smooth feature lines on surface meshes, In Proceedings of the Symposium on Geometry Processing, pp. 85–90, 2005.

[13] H. T. Ho, D. Gibbins, Multi-scale feature extraction for 3D models using local surface curvature, In Proceedings of the 2008 Digital Image Computing: Techniques and Applications, pp.16–23, 2008.

[14] A. Hubeli, M. Gross, Multiresolution feature extraction for unstructured meshes, In Proccedings of the IEEE Visualization, pp. 287–294, 2001.

[15] K. Ikeuchi, T. Oishi, J. Takamatsu, R. Sagawa, A. Nakazawa, R. Kurazume, K. Nishino, M. Kamakura, Y. Okamoto, The great Buddha project: digitally archiving, restoring, and analyzing cultural heritage objects, International Journal of Computer Vision, Vol. 75, No. 1, pp. 189–208, 2007.

[16] S.-K. Kim, C.-H. Kim, Finding ridges and valleys in a discrete surface using a modified MLS projection, Computer-Aided Design, Vol. 37, No. 14, pp.1533–1542, 2005.

[17] I. Kyriazis, I. Fudos, L. Paios, Detecting features from sliced point clouds, In Proceedings of the Second International Conference on Computer Graphics Theory and Applications, 2007.

[18] A. Lesley, A. Roy, Archaeological Illustration (Cambridge Manuals in Archaeology), Cambridge Unversity Press, 1989.

[19] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. E. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk, The Digital Michelangelo project: 3D scanning of large statues, In Proceedings of the SIGGRAPH, pp. 131–144, 2000.

[20] L. Linsen, Point cloud representation. University of Karlsruhe, Germany Technical Report, Faculty of Informatics, 2001.

[21] Y. Ohtake, A. Belyaev, H. -P. Seidel, Ridge-valley lines on meshes via implicit surface fitting, In Proceedings of the ACM SIGGRAPH, pp. 609–612, 2004.

[22] X. F. Pang, M. Y. Pang, Z. Song, Extracting feature curves on point sets, International Journal of Information Engineering and Electronic Business, Vol. 3, No. 3, pp.1–7, 2011.

[23] M. Pauly, R. Keiser, M. Gross, Multi-scale feature extraction on point-sampled surfaces, Computer Graphics Forum, Vol. 22, pp. 281–289, 2003.

[24] L. Piegl, W. Tiller, The NURBS book, Springer Verlag, 1997.

[25] L. Renju, L. Tao, Z. Hongbin, 3D Digitization and its applications in cultural heritage, In Proceedings of the international conference on digital heritage, pp. 381–388, 2010.

[26] G. Taubin, Curve and surface smoothing without shrinkage, In Proceedings of the fifth International Conference on Computer Vision, pp.852–857, 1995.

[27] K. Watanabe, A. G. Belyaev, Detection of salient curvature features on polygonal surfaces, Computer Graphics Forum, Vol. 20, No. 3, pp.385–392, 2001.

[28] C. Weber, S. Hamann, H. Hagen, Sharp Feature Detection in Point Clouds, In Proceedings of the IEEE International Conference on Shape Modeling and Applications, 2010.

[29] T. Weinkauf, D. G*ü*nther, Separatrix persistence: Extraction of salient edges on surfaces using topological methods, Computer Graphics Forum (Symp. on Geometry Processing), Vol. 28, pp.1519–1528, 2009.

[30] S. Yoshizawa, A. Belyaev, H. -P, Seidel, Fast and robust detection of crest lines on meshes, In Proceedings of the ACM symposium on Solid and physical modeling, pp. 227–232, 2005.

Enkhbayar Altantsetseg received the BS and MS in mathematics from National University of Mongolia in 1995 and 1997, respectively. He is currently working toward the PhD degree in computer science at Iwate University. His research interests include computer graphics, geometric modeling, illustrative visualization, medical visualization and simulation.

Yuta Muraki is currently a Post-doctoral Fellow in Faculty of Engineering at Iwate University. His research interests include geometric modeling, CG, CAD and 3D scanning. He received the B.E. and M.E. degrees in Computer and Information Sciences, and the D.E. degree in Electronic Information Science from IWATE University in 2005, 2007, and 2010, respectively. He is a member of JSPE.

Katsutsugu Matsuyama is currently an assistant professor at Iwate University. His research interests include computer graphics, information visualization and interactive systems. He received BE, ME, DE degrees in computer science from Iwate University in 1999, 2001 and 2005, respectively. He was a research associate at Future University-Hakodate from 2005 to 2011.

Fumito Chiba is a director of LANG Co., LTD. He received a bachelor of engineering from Iwate University in 1996. He earned his Dr. Eng. in Electrical Engineering and Computer Science from Iwate University in 2000. He worked on an assistant professor of Faculty of Engineering at Iwate University from 2000 to 2005. His research interests include 3D measurement systems and image processing. He is a member of Japan Society for Archaeological Information.

Kouichi Konno is a professor of Faculty of Engineering at Iwate University. He received a BS in Information Science in 1985 from the University of Tsukuba. He earned his Dr. Eng. in precision machinery engineering from the University of Tokyo in 1996. He joined the solid modeling project at RICOH from 1985 to 1999 and the XVL project at Lattice Technology in 2000. He worked on an associate professor of Faculty of Engineering at Iwate University from 2001 to 2009. His research interests include virtual reality, geometric modeling, 3D measurement systems, and computer graphics. He is a member of IEEE CS.