Ph.D Thesis

# A Study on Fine-tuned Flake Surface Matching Algorithm for Interactive Reassembly System of Stone Tools

Altansukh Amgalan

Department of Design and Media Technology
Graduate School of Engineering
Iwate University

September 2024

# Abstract

The Paleolithic era and the Jomon period in Japan produced a wide range of stone tools, including cutting implements and weapons. Given their durability compared to organic materials, such as bones, antlers, and wood, stone artifacts provide significant evidence of the locations and periods of early human activities because of their geographical distributions and adaptive capacities in various environments. A rock is knocked off with a stone hammer to make a stone tool, producing flakes of different sizes referred to as flake stones. A core is the left stone as raw material for stone tools when flake stones are knocked off. The reassembly process on stone flakes is a one of important tool for understanding ancient human activities. It is a reverse process of making stone tool process and requires to identify firstly a core stone and reassemble stone flakes on the core stone.

These reassembled stone tools can also have educational value as exhibition materials at history museums. However, reassembling stone tools is challenging archaeology; it consumes many human resources, time, and special archaeological knowledge. In order to efficiently reassemble stone tools, this thesis studied computer graphics techniques to assist this archaeological research. This work is mainly focused on developing a fine-tuned flake surface matching algorithm for interactive reassembly system of stone tools.

In recent years, several reassembly approaches have been developed for the reconstruction of fragmented archaeological artifacts such as pottery and fresco wall paintings; however, the successful application of these methods to reassemble stone tools has been limited for the following reasons: (1) Irregular shapes of stone fragments: stone tools exhibit highly irregular shapes, making it challenging for descriptor-based methods to extract crucial features. (2) The lack of distinct regional features on the flake surfaces further reassembly process. Flake surfaces are fractured surfaces created when a stone flake is

knocked off from a core stone and tend to be smooth surfaces. (3) Global matching algorithms have proven inadequate, given the requisite for partial matching between pairs of flake surfaces.

The first study calculates point normal vectors of each input point cloud of stone flakes. Then, stone flakes are segmented to extract the flake surface. Also, to define the shape of the flake surfaces, a boundary correction and contour points identification process are performed. Next, according to several reassembly principles in archaeology, the stone flakes are matched starting on the stone core by searching for the best matching flake surface. In order to find the best matching flake surfaces, a new fine-tuned flake surface matching algorithm based on contour points is developed. Additionally, the surface of matched stone tools are detected and reconstructed. The matching process is repeated until all data are matched. To confirm the fine-tuned matching algorithm's efficiency and usefulness, an interactive system for stone tool reassembly is developed. The system's interface allows the control of the reassembly process, enabling users to show visual representation and evaluate the progress and accuracy of the reassembly work. Also, it overcomes the familiar challenges in traditional reassembly methods, such as the irreversible errors in incorrect matching. The experiment of this system demonstrates its effectiveness and efficiency, highlighting its practical utility in archaeological research.

We have implemented the proposed methods and tested the system with three groups of stones. All groups are reassembled through our interactive system with the fine-tuned matching algorithm. A limitation of the matching algorithm is highlighted for future research. Experiment results of the system indicate that the interactive system with the fine-tuned matching algorithm can achieve superior matching results compared to existing methods.

# Contents

**List of Figures** **vi**

**List of Tables** **viii**

**1 Introduction** **1**
- 1.1 Background . . . . . 1
- 1.2 Motivation . . . . . 2
- 1.3 Purpose . . . . . 3
- 1.4 Integration of studies . . . . . 4
- 1.5 Thesis outline . . . . . 6

**2 Related Works** **8**
- 2.1 Normal vector estimation . . . . . 8
- 2.2 Flake surface matching . . . . . 9
- 2.3 Stone tools reassembly system . . . . . 10

**3 Normal vector estimation** **12**
- 3.1 Overview . . . . . 12
- 3.2 Voxelization . . . . . 14
- 3.3 Identifying voxel state . . . . . 15
- 3.4 Finding voxel size for closed surface . . . . . 16
- 3.5 Determining normal vector orientation . . . . . 17

**4 Flake surface extraction and contour point identification** **21**
- 4.1 Overview . . . . . 21
- 4.2 Region growing segmentation . . . . . 22

4.3 Boundary correction . . . . . 23
4.4 Contour point identification . . . . . 25

**5 Fine-tuned flake surface matching 28**
5.1 Overview . . . . . 28
5.2 New fine-tuned flake surface matching algorithm . . . . . 30
5.3 Flake surface reconstruction . . . . . 35

**6 Stone Tool Reassembly system 37**
6.1 Overview . . . . . 37
6.2 Pipeline . . . . . 37
6.3 User Interface . . . . . 40
6.3.1 Candidate viewer . . . . . 40
6.3.2 Matching viewer . . . . . 41

**7 Experimental results 43**
7.1 Normal vector estimation . . . . . 43
7.1.1 Data . . . . . 43
7.1.2 Results . . . . . 44
7.1.3 Discussion . . . . . 44
7.2 Flake surface extraction and contour point identification . . . 46
7.2.1 Data . . . . . 46
7.2.2 Results . . . . . 47
7.3 Fine-tuned flake surface matching algorithm . . . . . 51
7.3.1 Data . . . . . 51
7.3.2 Evaluation . . . . . 51
7.3.3 Results . . . . . 55
7.3.4 Discussion . . . . . 55
7.4 Reassembly system for stone tools . . . . . 59
7.4.1 Results . . . . . 59

**8 Conclusion and Future work 64**
8.1 Conclusions . . . . . 64
8.2 Future work . . . . . 65

**Bibliography 67**

**List of Publications** **i**

# List of Figures


1.1 A photo of an excavation site . . . . . 2
1.2 Producing stone tool . . . . . 3
1.3 Flake surface matching . . . . . 3
1.4 The reassembly process . . . . . 4
1.5 Example of excavated stones . . . . . 5
1.6 Connection of studies . . . . . 6

3.1 The third eigenvector calculated by PCA . . . . . 13
3.2 Relation between voxel space and bounding box . . . . . 14
3.3 3D flood filling . . . . . 15
3.4 The process of dividing voxels . . . . . 17
3.5 Neighborhood of the voxel . . . . . 18
3.6 Combinations: a) "OI", b) "SI", c) "OS", d) "SS" . . . . . 19
3.7 Combination "OO" . . . . . 20

4.1 a) Result of the region growing with an angle threshold $\alpha$ set to 1.5°, a curvature threshold $c$ to 0.1 and the minimum number of point in a valid segment $l$ to 500. b) A case of the unsegmented point with normal vector. . . . . 23
4.2 (a) Determining the fitting plane using PCA. (b),(c) Projecting points onto the fitting plane. (d) The convex hull. (e) The concave hull. (f) Identifying contours of the flake surface. . . . . 27

5.1 Case of making a stone tool (flake $A$ is peeled first followed by flake $B$). . . . . 29
5.2 Tuning the precision parameter. . . . . 31
5.3 Mapping a source into a target surface. . . . . 31
5.4 Flake surface reconstruction. . . . . 36

6.1 The pipeline of the reassembly system. . . . . 38
6.2 Candidate Viewer and Matching Viewer . . . . . 41

7.1 Experiment data for the normal estimation . . . . . 43
7.2 Result of our method: a) 0001, a) 0003, a) 0007, a) 0034 . . . . . 46
7.3 Combination “OO” . . . . . 47
7.4 Experiment data. . . . . 48
7.5 Result of the boundary correction. a) Point cloud. b) Result of region growing segmentation. c) Result of our boundary correction step. d) Result of extracting contour points. . . . . 49
7.6 Result of the flake surface extraction. a) $No.06$. b) $No.07$. . . . . 50
7.7 Parameter sensitivity test. . . . . 53
7.8 Visual representation of comparison result. . . . . 54
7.9 Result of flake surface matching. a) Stone models $No.01$ and $No.06$. b) Correspondence between flake surfaces 01_1 and 06_1. c), d) Result of matching in different views. . . . . 56
7.10 Reconstruction result. a) Reconstructed flake surfaces of $No.6$, $No.10$ and $No.16$. b) Source flake $No.17$. c) Result of matching. . 57
7.11 Limitation of the matching algorithm: Failed matching scenario for stone $No.14$. . . . . 58
7.12 Screenshot of the system . . . . . 59
7.13 Result of reassembly. a), b) Reassembled results by our system. c), d) Reassembled flakes of two groups. . . . . 63

# List of Tables


7.1 Result Table . . . . . 45
7.2 Relation between the voxel size and the accuracy of normal estimation 47
7.3 The absolute difference between the ground truth and the evaluation score. . . . . 55
7.4 Reassembly of Group 1 . . . . . 61
7.5 Reassembly Group 2 . . . . . 62
7.6 Comparison table with previous study . . . . . 62

# Chapter 1

# Introduction

## 1.1 Background

The Paleolithic era and Jomon Period of Japan witnessed the production of a wide range of stone tools, including cutting implements and weapons. A great number of archaeological sites were excavated and analyzed. These sites often contain remnants of debris generated during the production and utilization of stone tools, as shown Figure 1.1. Given their durability compared to organic materials such as bones, antlers, and wood, stone artifacts provide significant evidence of the locations and periods of early human activities because their geographical distribution, and adaptive capacities in various environments[1].

The process of producing stone tools is an prehistorical technique that was fundamental to early human development. It involves several steps as shown Figure 1.2.

The first step in the process is the selection of raw material or raw stone that fractures predictably. The chosen rock is referred to as the "mother rock". To make a stone tool, the mother rock is struck repeatedly with stone hammer. This striking process produces sharp fragments named "flake stones", while the remaining part of the stone can also be shaped into a stone tool. After producing a stone tools, remaining stone of the source is called "core stone" as shown Figure 1.2.

A flake surface is a surface created during the knapping process as shown Figure 1.3. This surface is the result of fracturing that occurs when a flake

Figure 1.1: A photo of an excavation site

stone is detached from a core stone. Therefore, the flake surface found on both detached flake and the remaining core stone. The precise alignment and matching of these flake surfaces are crucial for the accurate reassembly of stone tools, which is integral part of archaeological restoration and analysis.

## 1.2 Motivation

Reassembly of the stone flakes and core offers valuable insights into early human toolmakers, including their production techniques, life patterns, and the transfer of things[2]. The reassembly is a reverse process of making stone tool operation. Therefore, successful reassembly allows archaeologists to analyze the sequence of assembly to investigate how stone tools are created and also it could restore the mother rock which is referred to as the joining material. Moreover, the joining material has significant educational value and can be displayed in historical museums and educational institutions[1].

However, the manual reassembly of stone tools is a daunting task in archaeology, consuming substantial human resource, time and specialized knowledge.

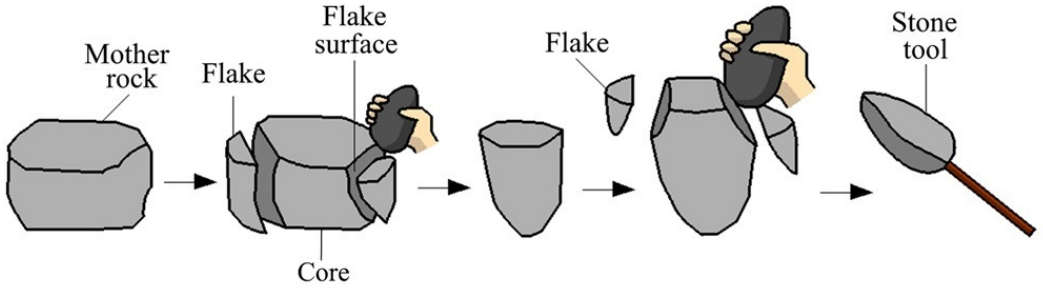



Figure 1.2: Producing stone tool

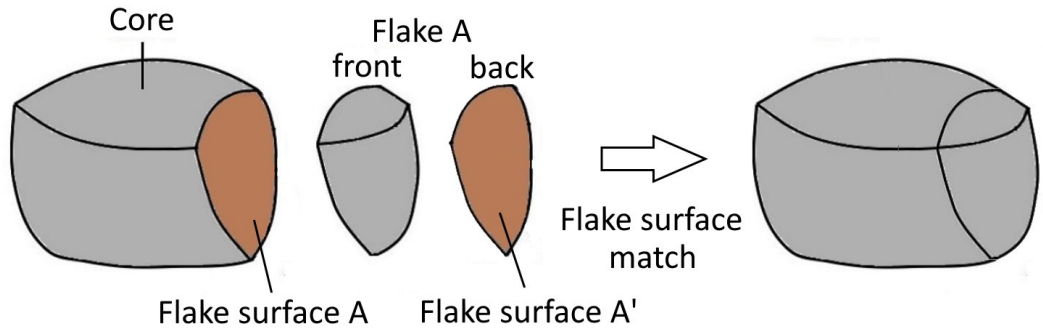



Figure 1.3: Flake surface matching

This thesis motivated by the need to develop efficient methods to assist archaeologists in this intricate task. By using computer algorithms, this thesis aims to streamline the reassembly process for stone tools, making it more efficient and less dependent on extensive manual labor.

## 1.3 Purpose

The purpose of this thesis is to develop a fine-tuned flake surface matching algorithm for an interactive system of stone tools. This system aims to facilitate the reassembly process by providing user-friendly interface, enhancing both speed and accuracy of reassembly. By integrating the fine-tuned flake surface matching algorithm, the system seeks to assist archaeologists in reconstructing fragmented stone tools, thus contributing to a deeper understanding of early human technology.

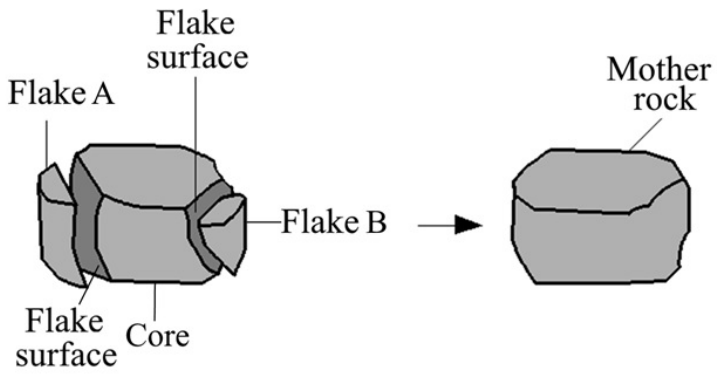



Figure 1.4: The reassembly process

## 1.4 Integration of studies

This thesis integrates multiple research components into an unified workflow for the interactive reassembly of stone tools, as illustrated in the Figure 1.6. Each chapter represents a crucial step in this process, contributing overall system.

- **Chapter 3: Normal Vector Estimation** - This preliminary step involves estimating the normal vectors from the point cloud data of stone fragments. Accurate normal vector estimation is essential for understanding the surface orientations of the stone pieces, forming the basis for subsequent processing.

- **Chapter 4: Flake Surface Extraction and Contour Point Identification** - Building on the normal vector estimation, this necessary step focuses on extracting the flake surfaces from the point cloud. It involves segmenting the point cloud to extract individual flake surfaces and identifying their contour points, which are critical for the matching process.

- **Chapter 5: Fine-tuned Flake Surface Matching** - Using the results from Chapter 4, this chapter details the core component of the system: the development of a fine-tuned matching algorithm. This algorithm is designed to match the extracted flake surfaces based on their contours, ensuring the most accurate reassembly of the stone tools.

Figure 1.5: Example of excavated stones

- **Chapter 6: Stone Tool Reassembly System** - This chapter describes the the interactive reassembly system for stone tools. The system utilizes the fine-tuned flake surface matching algorithm developed in Chapter 5. The input for this system comes from the results of flake surface extraction processes detailed in Chapters 4. The interactive system allows users to control the reassembly process, visualize the matching results, and make necessary adjustments, thereby facilitating efficient and precise reassembly of stone tools.

By integrating these studies, this thesis presents a system that enhances the reassembly of stone tools, making it more efficient and precise. This system streamlines the reassembly process but also contributes to archaeological research and the understanding of early human tool-making techniques.

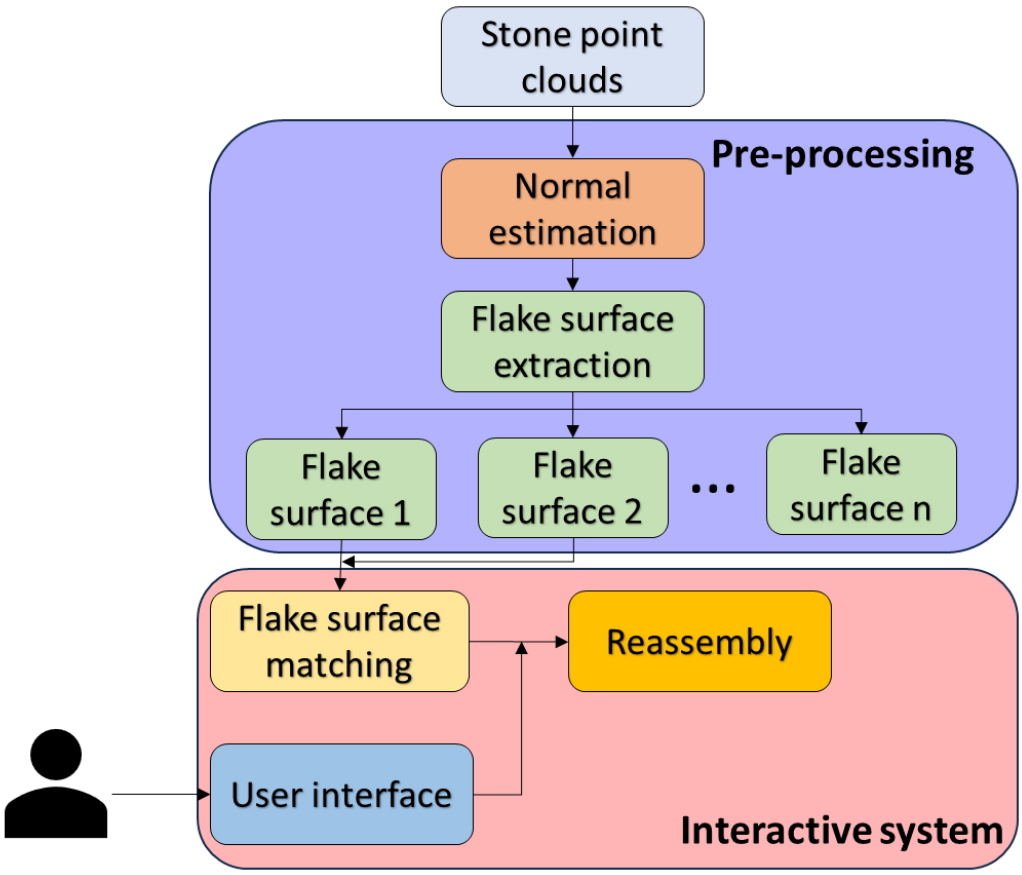



Figure 1.6: Connection of studies

## 1.5 Thesis outline

This thesis is structured to systematically address the challenges and solutions related to the reassembly of stone tools:

- **Chapter 2: Related Works** - Reviews existing literature on normal vector estimation, flake surface extraction, flake surface matching, and stone tools reassembly systems.
- **Chapter 3: Normal Vector Estimation** - Details the methods for normal vector estimation, including voxelization, identifying voxel state, finding appropriate voxel sizes, and determining normal vector orientation.
- **Chapter 4: Flake Surface Extraction and Contour Point Identification** - Describes the process of extracting flake surfaces and identi-

fying contour points through region growing segmentation and boundary correction.

- **Chapter 5: Fine-tuned Flake Surface Matching** - Introduces the fine-tuned matching algorithm and its application in reconstructing flake surfaces.
- **Chapter 6: Stone Tool Reassembly System** - Explains the development of the interactive reassembly system, detailing its pipeline and user interface.
- **Chapter 7: Experimental Results** - Presents the data, results, and discussion of the experiments conducted to validate the proposed methods.
- **Chapter 8: Conclusion and Future Works** - Summarizes the findings and outlines potential future research directions.

This structure ensures a comprehensive exploration of the methodologies and applications developed in the thesis, aiming to significantly enhance the reassembly of stone tools and contribute to the field of archaeological studies and computational archaeology.

# Chapter 2

# Related Works

## 2.1 Normal vector estimation

A study [10] proposed a normal vector estimation algorithm based on KD-Tree along with introducing the PCA method to determine the tangent plane. From the specified viewpoint, the normal vector of the tangent plane is evaluated and the orientation is flipped. However, their method depends on the viewpoint. Thus, using the normal vector for segmentation and other processes is impossible.

The method in presented in [11] precisely estimates the local normal vector in point clouds based on robust optimization by integrating weighted and iterative PCA to smoothly separate neighbors into outliers and inliers. In addition, the method presented in [11] does not require prior local-curvature information. [12] focuses on increasing the precision of determining a point normal vector, which requires a long computational time because of the iterative PCA. In contrast, our method requires PCA only one time.

The neural network proposed in [12] can accurately infer a normal vector from a point cloud. The main idea of the neural network model is to introduce a voxel structure to obtain spatial features from the point clouds. The proposed model takes advantage of two subnets, called the point network and voxel network. The point network extracts the local features corresponding to the local shape of an object, whereas voxel networks transform a point cloud into voxels and extract spatial features. Normal estimation with the point net-

work computes an unoriented normal vector because the point network only encodes the local features. They compensate for this shortcoming by adopting a voxel network and extracting the spatial features. These local and global feature vectors are combined into one vector and transformed into a normal vector using a learnable multilayer perceptron network. It is highly dependent on training data or static information because it employs a machine-learning algorithm. Therefore, training is necessary to initially obtain training data; the more the training data, the better the results.

An another study [13] attempts to define an inside voxel. They calculate the volume of a tree using a point cloud of wood measured using a 3D laser scanner. The bounding box of a point cloud is divided into voxels, and empty voxels containing no points are marked along the X and Y axes. If a voxel is marked on both the X and Y axes, it is a potential candidate for the inside voxel. If there is a potential candidate within the largest stem radius, it is considered an internal voxel. The volume of the tree is calculated by adding bounding voxels containing some points, and internal voxels.

## 2.2 Flake surface matching

Iterative Closest Point (ICP) [15], Super4PCS [16], fast point feature histograms (FPFH) [17] and random sample consensus (RANSAC) [18] algorithms are commonly employed in point cloud registration tasks. Furthermore, these methods have been adapted and applied to surface matching scenarios such as flake surface matching.

The ICP algorithm [15], which is one of the cornerstones of point cloud registration, has demonstrated wide-ranging effectiveness in aligning 3D surfaces. However, it may be difficult to obtain an accurate initial alignment, particularly in cases involving partial flake surfaces in stone tools, where achieving a precise alignment can be particularly challenging.

Super4PCS [16], recognized for its proficiency in matching partial 3D shapes, efficiently identifies local surface correspondences. Considering this capacity, it is a compelling candidate for adapting to the task of matching partial flake surfaces. The inherent ability of the algorithm to handle partial matching aligns well with the result of the present study.

The FPFH [17] calculates the correspondences and correspondence probabilities, providing valuable guidance for subsequent RANSAC [18] matching algorithms. In [18], a robust method for estimating transformation models between data points is presented, particularly in the presence of outliers, and is a crucial step in aligning partial flake surfaces. Their approach, which utilizes the FPFH [17] for the initial correlation, addresses challenges related to partial matching and aligning surfaces with potentially irregular shapes.

Our previous study [19] focused on the same task, considering the shape of the flake surface by utilizing contour points, which is similar to our approach. However, it was characterized by prolonged computation times and limitations in effectively matching partial flake surfaces.

## 2.3 Stone tools reassembly system

Extensive studies have been conducted on the reassembly fractured objects. Huang et al. [20] introduced a feature-based alignment method that effectively addressed these scenarios and achieved remarkable results. However, their method is complex and, comprises various specialized algorithms for tasks such as segmentation, multi-scale feature extraction, correspondence determination, registration, collision detection, and supervised learning. Consequently, implementation and adoption of these methods pose significant challenges. Brown et al. [21] proposed a method specifically designed to reassemble fragments of wall paintings. Willis et al. [22] presented a system tailored to pottery shard reassembly.

In [23], an innovative method was proposed for reassembling axially symmetric ceramic pots from three-dimensional (3D) scanned fragments. Their method addressed specific challenges with ceramic artifacts, particularly axial symmetry. Their approach is noteworthy because it incrementally adds fragments using a beam search technique. This significantly mitigates the false positive matches. Their method not only improves the match accuracy through geometric descriptors but also facilitates the simultaneous reassembly of multiple pots from mixed collections. However, the application of their technique to the reassembly of stone tools is challenging. Reassembly stone tools lack the axial symmetry present in ceramic pots and require consideration of the flake removal sequence from the core stone.

In [24], a new methodology for generating synthetic 3D fragmented data was proposed to facilitate the evaluation of object restoration, particularly in the context of cultural heritage. Their approach was designed to overcome the challenges of limited availability of real test data. Their method produces artificial fracturing from an input object without physical simulation. In addition, their method uses a "cutter object" to create new breaking edges and thereafter fragments the object. It generates a large-scale fragment test dataset from existing cultural heritage models. These datasets have advantage of ground truth (the input object before fracturing), which is often missed. However, it does not directly relate to the reassembly stone tool task or reassembly task. It focuses on the reverse process of reassembling models.

In [25], a mesh-based approach to create restorations integrated with broken objects was introduced. Utilizing the 3D scanned meshes of both broken objects and their intact counterparts, their method generates a restoration piece by smooth. It focuses on objects for which the missing parts are predictable, and its goal is to restore the original form of the objects. However, its application to the reassembly of stone tools poses several challenges. First, it relies on the predictability of the shape of the missing part. This condition is not satisfied in stone tool reassembly in which the shape and size of the flakes are unpredictable. Second, the methodology is mesh-based, whereas stone data were represented in a point cloud. Finally, their proposed method does not address partial matching or the sequential order of assembly, both of which are crucial for stone tool reassembly tasks.

The "Fantastic Breaks" dataset [26] was designed to facilitate machine learning study in automated reassembly by providing a comprehensive collection of 3D scans of real-world broken objects and their counterparts. It focuses on a complete object reconstruction using predefined models. The use of predefined models may not align with the stone tool reassembly tasks. This often requires addressing the unpredictable fragment shapes and sizes. Machine learning approaches for stone tool reassembly may face challenges owing to the requirement for extensive, well-prepared datasets and the computational demands for processing large datasets. Stone tool datasets are often characterized by a lack of extensive pretested data. A highly detailed point cloud representation can limit the applicability of machine learning methods. Specifically, our datasets with 280,000 points per stone can strain the GPU resources [27]. This renders the learning process more challenging.

# Chapter 3

# Normal vector estimation

## 3.1 Overview

Normal vector estimation is a critical step in the process of reassembling fragmented stone tools from point cloud data. This chapter focuses on developing and implementing an algorithm to estimate the normal vectors for each point in a 3D point cloud, which represents the surfaces of stone fragments. The estimated normal vectors are essential for understanding the surface orientations and subsequently facilitating the segmentation for extracting flake surface and matching processes in the reassembly system.

Segmentation is applied to extract the flake surfaces for creating joining material. The curvature [8] and normal vector in the point cloud must be calculated to perform segmentation. The third eigenvector calculated by principle component analysis (PCA) can be used to estimate the surface normal vector [9]. However, to determine the orientation of the normal vector, the surface of an object must be considered. Since the normal vector is directed outside the point cloud, a method for determining the exterior and interior of an object is required. In PCA, the third eigenvector is directed outside the surface or in the opposite direction. Figure 3.1 illustrates this situation. The vector $\mathbf{e}_1$ is the third eigenvector, and $\mathbf{e}_2$ is a opposite vector of $\mathbf{e}_1$. In this case, $\mathbf{e}_2$ is the normal vector.

Since the surface normal vector is necessary to extract flake surfaces for the assembly information of the stone tool, this study proposes fast normal

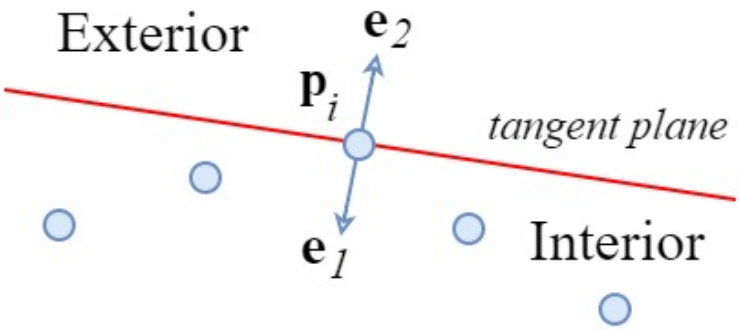



Figure 3.1: The third eigenvector calculated by PCA

vector estimation using space subdivision to overcome the above issue. In the proposed method, an adaptive voxelization technique is introduced. Using voxelization with optimization of voxel size, the surface of the object is suitably determined. We verify that the proposed point normal vector estimation method is effective and useful for a 3D point cloud.

The chapter is structured as follows:

1. **Voxelization:** The initial step involves dividing the 3D space into a grid of voxels. This process simplifies the point cloud data and helps in identifying surface and internal points.

2. **Identifying Voxel State:** After voxelization, each voxel is classified as external, internal, or surface based on its position relative to the point cloud. This classification is crucial for determining the orientation of the normal vectors.

3. **Finding Voxel Size for Closed Surface:** This step ensures that the chosen voxel size accurately represents the surface of the stone fragments without creating gaps or overlaps. The voxel size is iteratively adjusted to achieve a closed surface representation.

4. **Determining Normal Vector Orientation:** Using the voxelized data and the classified voxel states, the algorithm calculates the normal vectors for each point. The orientation of these vectors is determined to point outwards from the surface, ensuring consistency and accuracy in representing the surface normals.

## 3.2 Voxelization

It is necessary to be determine a voxel space and an initial value of voxel size for voxelization. Subsequently, the voxel space is divided into cube of the same voxel size. First, the distance between a given point and its nearest neighboring point is calculated for each point in the point cloud. Among all distances, the maximum distance is selected and set as the initial value of the voxel size $s_0$ in Equation 3.1.

$$s_0 = q\lfloor\frac{argmax(distance(\mathbf{p}_i, N(\mathbf{p}_i)))}{q}\rfloor \tag{3.1}$$

Where, $\mathbf{p}_i$ is the $i$-th point of the point cloud, $N(\mathbf{p}_i)$ is the nearest neighbor of the point $\mathbf{p}_i$, $q$ is the tolerance of voxel size and is manually set.

Our method has three types of voxel states. These are external, internal, surface voxels. External voxels are outside the point cloud, whereas the internal voxels are located inside the point cloud. A surface voxel is a voxel that contains points on the point cloud. A voxel cube has one of the three states.

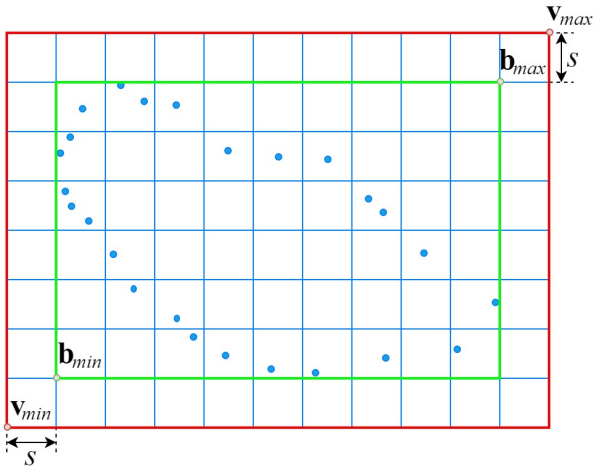



Figure 3.2: Relation between voxel space and bounding box

The voxel space is like a bounding box for the point cloud. However, for the proposed method, the voxel space is larger than the bounding box by two voxel sizes in each dimension, as shown in Figure 3.2. In Figure 3.2, the green box is the bounding box and the red box represents the voxel space. This is because a surface voxel must have a neighboring external voxel in the

proposed method. Equations 3.2 and 3.3 show relationship between the voxel space and bounding box.

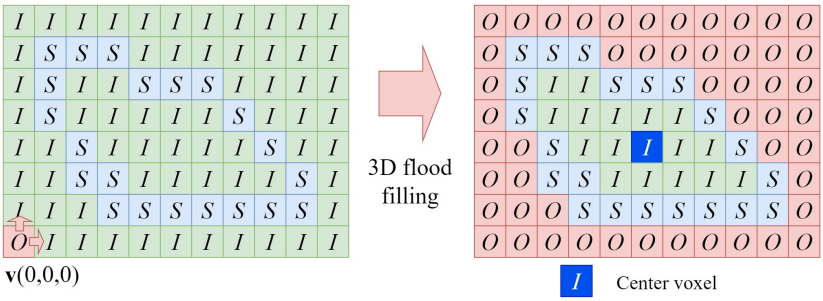
$$\mathbf{v}_{max} = \mathbf{b}_{max} + s \tag{3.2}$$

$$\mathbf{v}_{min} = \mathbf{b}_{min} - s \tag{3.3}$$

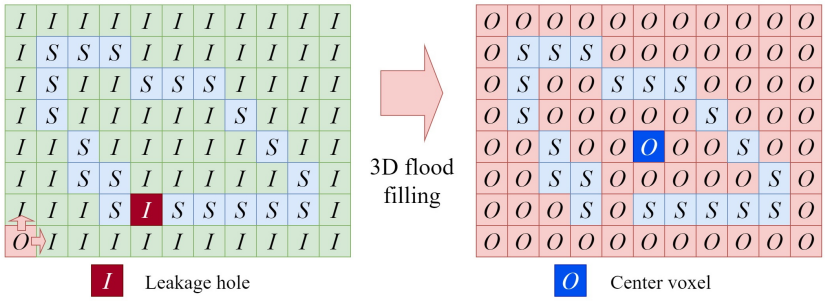
Where, $\mathbf{b}_{max}$ is a maximum point of the bounding box, $\mathbf{b}_{min}$ is a minimum point of the bounding box, $\mathbf{v}_{max}$ is a maximum point of voxel space, $\mathbf{v}_{min}$ is a minimum point of the voxel space, and $s$ is voxel size.

Voxel cubes in voxel space need to be indexed, making it easier and faster to find neighboring voxels and determine which voxel contains the given point in the point cloud. To do this, a 3-dimensional index vector $\mathbf{v}(i, j, k)$ is defined, where $i$ is the index for the $x$ axis, $j$ is the index for the $y$ axis, and $k$ is the index for $z$ the axis.

## 3.3 Identifying voxel state




(a) Closed surface voxel




(b) Open surface voxel with leakage hole

Figure 3.3: 3D flood filling

At first, all voxels are marked as "I" indicating that they are internal voxels. Then, if the voxel contains points, it is a the surface voxel and marked as "S" as shown in the left side of Figure 3.3. External voxels marked as "O" are determined using the 3D flood filling algorithm. A voxel cube, whose index vector is (0,0,0), is obviously external voxel, as shown in Figure 3.3(a), it is marked as "O" and selected as the seed voxel cube for the 3D flood filling algorithm. If the neighboring voxel state is "I", the filling operation is continued, and the state is replaced by "O".

## 3.4 Finding voxel size for closed surface

The voxel size has a crucial effect on the results of this study. This is because a smaller voxel size results in a smaller voxel cube, which increases the accuracy of the proposed method. Unfortunately, the 3D flood filling algorithm cannot work correctly if the voxel size is too small. In other words, the surface voxels cannot determine the closed 3D surface when too small voxel size is applied. Thus, identifying the external and internal voxels becomes impossible because the external voxel flows through the leakage hole, as shown in Figure 3.3(b). Therefore, a sufficient condition to use the 3D flood filling algorithm is that surface voxels must be closed.

$$\mathbf{c} = (\mathbf{b}_{min} + \mathbf{b}_{max})/2 \tag{3.4}$$

Consequently, the center of the bounding box defined using Equation 3.4, where $\mathbf{c}$ is the center of the bounding box. A voxel containing $\mathbf{c}$, is defined as the center voxel, blue voxel as shown of Figure 3.3. It is assumed that the center voxel must be an internal voxel if the surface voxel is closed. After identifying the voxel state, we test whether the center voxel is an internal or external. If the center voxel is an internal voxel, the closed area is represented by surface voxels, and the internal and external voxels are correctly identified, as shown in Figure 3.3(a). Otherwise, if the center voxel is an the external voxel, the surface is not closed, and the external and the internal voxels cannot be identified correctly, as shown in Figure 3.3(b). If the center voxel is an external voxel, the voxel size must be enlarged using Equation 3.5. Subsequently, voxelization and voxel state identification processes are repeated. This process is repeated until the center voxel is an internal voxel.

After performing the above process, a suitable voxel size for the point cloud is determined, indicating that the internal and external voxels are correctly identified.

$$s = s + q \tag{3.5}$$

Where, $s$ is voxel size and, $q$ is tolerance of voxel size.

To increase accuracy, all voxels in the voxel space are divided, as shown in Figure 3.4. Each voxel cube is divided into eight equal child voxel cubes. To maintain processing speed, the voxel state of these child voxels is inherited the same as the state of the parent voxel that contains them. The next step, determining normal vector orientation is performed on the child voxels.

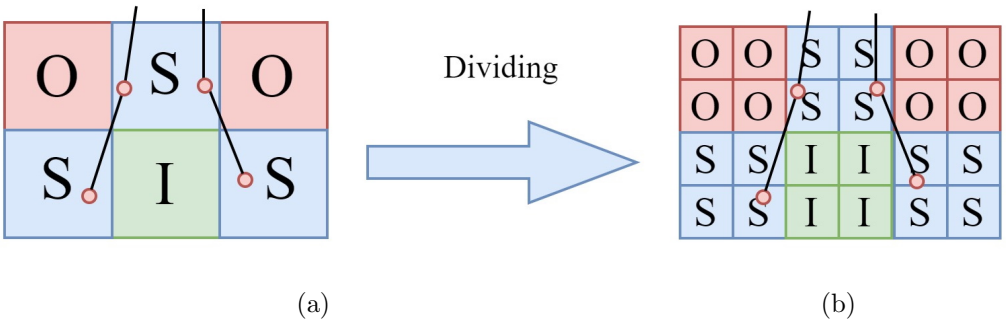



Figure 3.4: The process of dividing voxels

## 3.5 Determining normal vector orientation

After the voxels are divided, the state of all child voxels in the voxel space is one of the three states. Neighboring voxels are named "First-Ring" neighbors, colored in blue voxels, as shown in Figure 3.5. An infinite line $l(t)$ is defined by Equation 3.6. The line intersects two voxels in the neighboring voxels of the surface voxel containing point $\mathbf{p}_i$. These two intersecting voxels become a combination with, any of the five possible combinations, namely "OI" (external-internal), "OS" (external-surface), "OO" (external-external), "SI" (surface-internal), and "SS" (surface-surface).

$$l(t) = \mathbf{p}_i + t\mathbf{e}_i \tag{3.6}$$

Where, $\mathbf{p}_i$ is a target point, $\mathbf{e}_i$ is third eigenvector derived by PCA and, $t$ is a parameter.

The normal vector at $\mathbf{p}_i$ must be directed to an outer region of the point cloud. Therefore, when the combination of intersecting voxels is different, outer voxel is selected as the normal vector orientation, implying that the intersecting voxel "O" is selected as the orientation of the normal vector if the pair is "OI". In the same manner, "O" is selected if the pair is "OS", and "S" is selected if the pair is "SI" as shown in Figure 3.6. This operation can correctly define the orientation of the normal vector.

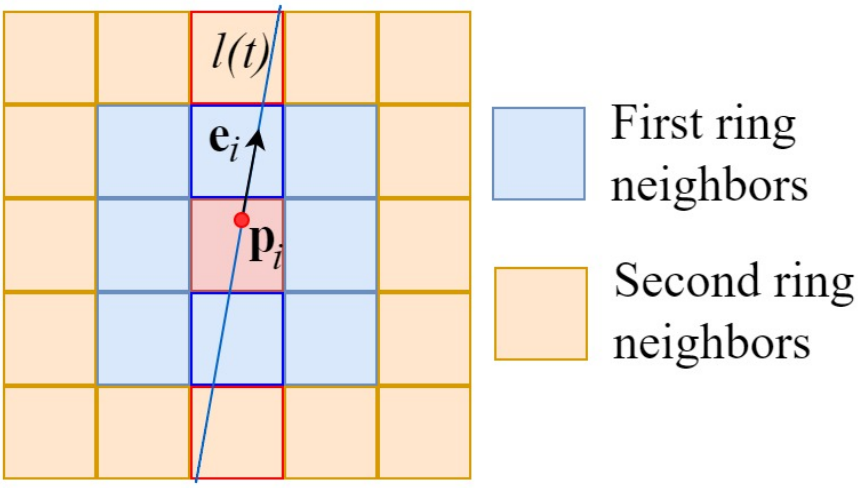



Figure 3.5: Neighborhood of the voxel

If the combination is "SS", neither the external nor the internal voxel is found in the first-ring neighbors. Then, an area of the possible intersecting neighborhood is extended to the second-ring neighbors colored in orange voxels, as shown in Figure 3.5, voxels with blue border intersecting two voxels in "First Ring" and voxels with red border are intersecting voxels in "Second Ring". If it is not found in the second-ring neighbors, the neighborhood area is extended in the same manner until different combinations are found. After finding the different pairs of voxels or the pair "OO" the direction of the normal vector is determined as described above.

When the combination is "OO", both distances, $d_1$ derived by $\mathbf{p}_i$ and the intersection of the line $l(t)$ with voxel, and $d_2$ derived by $\mathbf{p}_i$ and another intersection of the line $l(t)$ and voxels are calculated as shown in Figure 3.7.

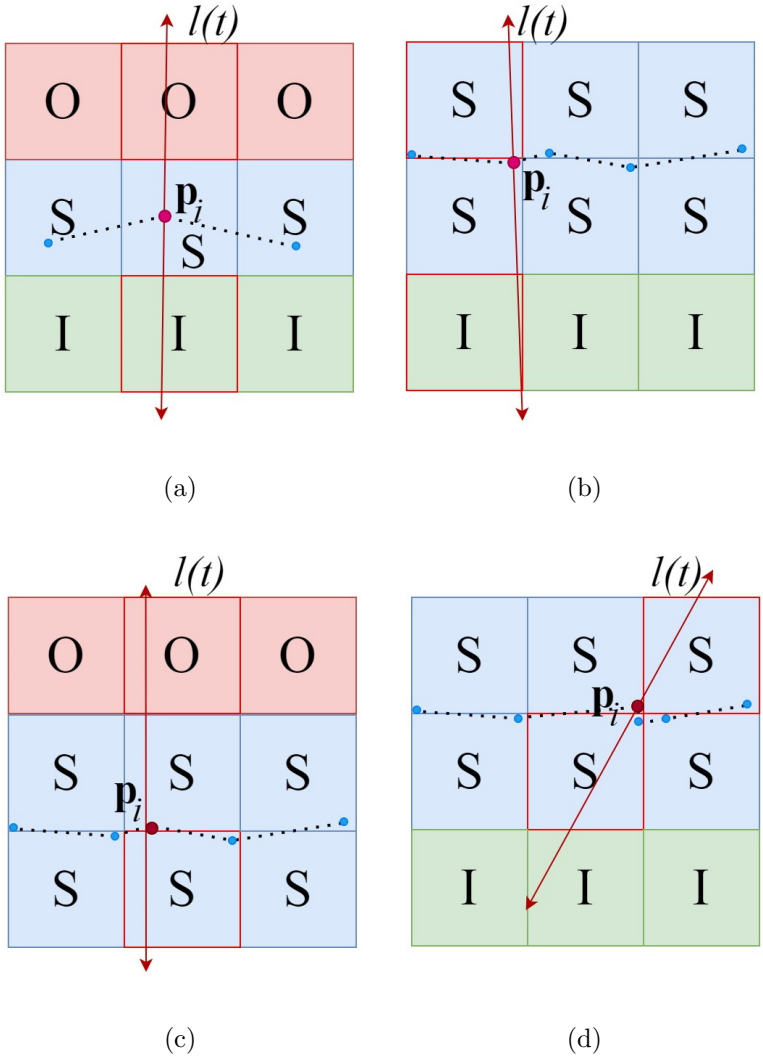



Figure 3.6: Combinations: a) “OI”, b) “SI”, c) “OS”, d) “SS”

The normal vector is directed to the nearest voxel. In other words, $O_1$ is selected because $d_1$ is less than $d_2$.

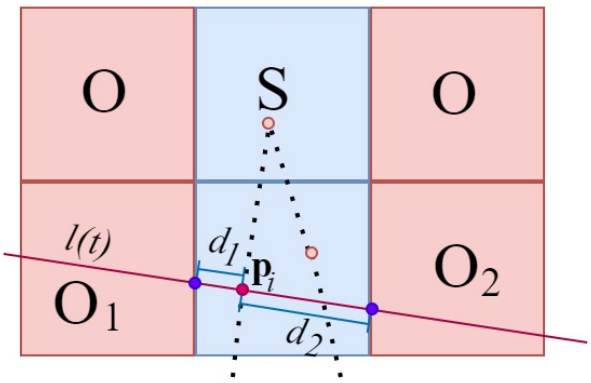



Figure 3.7: Combination “OO”

# Chapter 4

# Flake surface extraction and contour point identification

## 4.1 Overview

This chapter details the methods and algorithms used for the extraction and identification of flake surface contours, which are crucial for the reassembly of fragmented stone tools. The process begins with region growing segmentation, which segments the point cloud data based on geometric features such as point normals and curvatures. This initial segmentation often results in over-segmentation near sharp edges due to the algorithm's sensitivity to abrupt changes in normal vectors and curvatures.

To address this, we employ a boundary correction technique that re-segments the points, ensuring accurate boundary delineation by merging unsegmented points back into valid segments. Once the flake surfaces are accurately segmented, the next step involves identifying the contour points of these surfaces.

A fitting plane is determined using Principal Component Analysis (PCA), which aligns the flake surface to a 2D plane for better analysis. The points are then projected onto this fitting plane, preserving the morphology of the flake surface. The contour extraction is performed using the Jarvis March (Gift Wrapping) algorithm, which efficiently identifies the convex hull of the points.

Further refinement is done by incorporating concave segments to accurately represent the true boundary of the flake surface.

The identified contour points, which are determined in a clockwise direction, provide valuable insights into the characteristics of the flake surface, including its edges and overall shape. This detailed and methodical approach ensures high precision in the extraction and identification of flake surface contours, facilitating the accurate reassembly of stone tools.

The chapter is structured as follows:

- **Region Growing Segmentation.**

  Describes the initial segmentation process and the criteria used for segmenting the point cloud data. Boundary Correction: Details the steps taken to correct over-segmentation and accurately delineate the boundaries of flake surfaces.

- **Contour Point Identification.**

  Explains the process of determining a fitting plane using PCA, projecting points onto the plane, and extracting contour points using the Jarvis March algorithm, including the refinement steps for incorporating concave segments. By following these methodologies, we ensure that the flake surfaces are accurately segmented and their contours precisely identified, providing a robust foundation for the subsequent reassembly of stone tools.

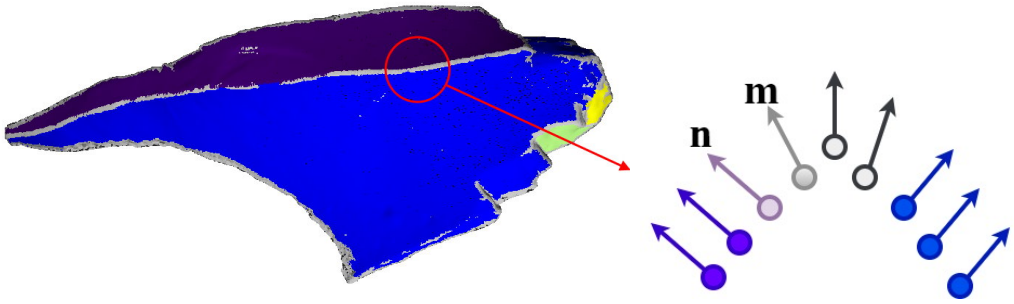
## 4.2 Region growing segmentation

The region growing algorithm segments a point cloud starting from a seed point and adding neighboring points that satisfy the criteria of geometric features such as differences in point normals and curvatures. Near sharp edges, the large amount of shape variation in these local geometric features can result in over-segmentation near them. This is largely owing to the algorithm's sensitivity to abrupt changes in normal vectors and curvatures.

In the region growing segmentation, the angle threshold $\alpha$ limits the addition of point to those with similar normal vectors in a region, ensuring the smoothness of the region. The curvature threshold $c$ excludes points from the

region with abrupt changes in curvatures. The minimum number of points $l$ is the required number of points for a region to be considered valid.

The algorithm categorizes segments as valid or invalid based on the number of points they contain. Regions that are over-segmented, particularly near sharp edges, may be classified as invalid owing to their small size. If the number of points belonging to a segment is less than $l$, points in these invalid segments are considered unsegmented points, as shown in the gray points in Figure 4.1. These unsegmented points are crucial, because they contain essential information regarding the true boundaries of the flake surfaces. Therefore, our method corrects the boundary of the region by merging unsegmented points as a re-segmented operation.




(a) (b)

Figure 4.1: a) Result of the region growing with an angle threshold $\alpha$ set to 1.5°, a curvature threshold $c$ to 0.1 and the minimum number of point in a valid segment $l$ to 500.

b) A case of the unsegmented point with normal vector.

## 4.3 Boundary correction

To correct the boundaries of the flake surfaces, the points were re-segmented using five steps. The first two steps are introduced in Point Cloud Library [28]. Steps from 3 to 5 are additional steps to correct boundary points of the segmentation process. The detail of steps is as follows:

1. Running the region growing segmentation with the angle threshold $\alpha$ and curvature threshold $c$.

2. The segments are categorized into two groups based on the number of points. Regions with a point count exceeding $l$ are considered as valid segments, represented by the purple, blue, green, and yellow points in Figure 4.1(a). Conversely, those with fewer points are identified as over-segmented regions that included unsegmented points, which are depicted as gray points in Figure 4.1(a).

3. Find the $k$ nearest neighbors of each unsegmented point. The $k$ is determined that at least one point included in the valid segment is selected as the nearest neighbor point.

4. For each unsegmented point:

   (a) Determine if the neighboring points belong to a valid segment. If a neighboring point belongs to a valid segment, that segment is considered a bordered segment with the unsegmented point. Thus, the unsegmented point is a candidate of merging to bordered segment. This scenario is visualized with the unsegmented point as a light gray point and the neighboring point as a light purple point in Figure 4.1(b). The neighboring point is considered a border point. When two or more neighboring points are found in the same valid segment, select the nearest neighboring point as the border point.

   (b) Calculate the angle $\phi$ between the normal vectors of border point invalid segment and the unsegmented point, illustrated as $\mathbf{m}$ and $\mathbf{n}$ in Figure 4.1(b), respectively. This calculation is computed by Equation (4.1).

$$\phi = \arccos\left(\frac{\mathbf{n} \cdot \mathbf{m}}{|\mathbf{n}||\mathbf{m}|}\right) \tag{4.1}$$

   (c) Merge the unsegmented point into the bordering valid segment with the minimum degree of $\phi$.

5. Repeat step 4 until all unsegmented point have been merged into valid segments.

## 4.4 Contour point identification

After the boundary correction of a segment (flake surface), the next step involves determining a fitting plane using PCA [30]. The PCA aligns the flake surface to ensure an accurate representation. Once the fitting plane is determined, all points in the segment are projected onto this plane, effectively aligning them with the surface morphology. This step is crucial because one of the the characteristics of flake surfaces is their smoothly changing surface, which means that projecting onto 2D plane should preserve the shape of the flake surface.

Following the projection, a 2D contour points extraction algorithm, specifically the Jarvis March (Gift Wrapping) algorithm [31], is applied. This algorithm is particularly suitable for extracting the contour of the flake surface due to its efficiency and handling point on the convex hull [31]. The steps involved in this process are:

1. **Determining the fitting plane.**

   Apply PCA to the flake surface points to determine the fitting plane that best represents the surface's morphology [30], as illustrated Figure 4.2 (a).

2. **Projecting points onto the fitting plane.**

   All points within the flake surface are projected onto the determined fitting plane, effectively flatting the 3D flake surface into a 2D plane, as illustrated Figure 4.2 (b) and (c).

3. **Extracting convex hull using Jarvis March (Gift wrapping) algorithm.**

   Identify the leftmost point in the 2D plane as the starting point.

   From the leftmost point, use the cross product to determine the next point on the hull, wrapping around the points until returning to the starting point. This process ensures that all points on the convex boundary are included [31], as illustrated Figure 4.2 (d).

4. **Identifying concave segments.**

   Once the convex hull is identified, the algorithm proceed to identify and incorporate concave points by examining segments of the hull and

adding points that minimize the angle between neighboring points while maintaining the integrity of the flake surface.

For each segment defined by two point on the convex hull, calculate the distance and angle to potential interior points. If the angle is minimal and distances are within a threshold or a maximum segment length of concave hull, incorporate the point into the hull, , as illustrated Figure 4.2 (e).

5. **Finalizing contour points.**

   The contour point identified through the above process represents the true boundary of the flake surface. These points are identified in the clockwise direction. The identified contour points provide valuable insight into characteristics of the flake surface, including its edges and overall shape, , as illustrated Figure 4.2 (f).

By following these steps, the extraction and identification of flake surface contours are performed with precision that enables the accurate reassembly of stone tools.

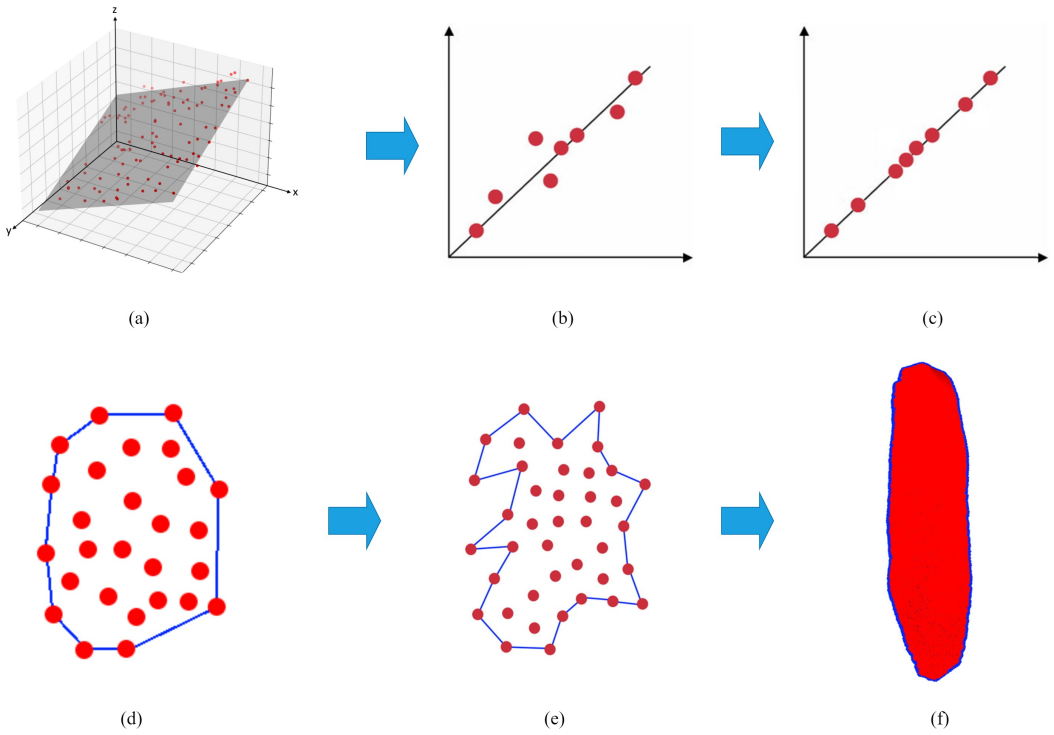



Figure 4.2: (a) Determining the fitting plane using PCA.

(b),(c) Projecting points onto the fitting plane.

(d) The convex hull.

(e) The concave hull.

(f) Identifying contours of the flake surface.

# Chapter 5

# Fine-tuned flake surface matching

## 5.1 Overview

The fine-tuned flake surface matching algorithm is a crucial component for the interactive reassembly system for stone tools. This chapter focuses on the development and application of this algorithm, which is designed to accurately match the extracted flake surfaces based on their contour points. The algorithm's primary objective is to ensure precise alignment and reassembly of the fragmented stone tools.

Lithic materials exhibits distinct characteristics representing the joining surface shape and separation from the reassembly of other fractured objects [19], as shown in Figure 5.1. The first distinguishing property is the sequence of flake generation from a single core [33]. Unlike arbitrary flake matching, stone tool reassembly requires a sequential order. For instance, if flake $A$ precedes flake $B$ in the peeling process, then, considering the sequential extraction of flakes from the core stone, the matching procedure entails a reverse order.

The second property pertains to the flat and smooth of most flake surfaces in the matching. This is because of the capacity of the mother stone to divide sharply [1]. Consequently, conventional matching algorithms that rely on surface features may not be suitable.

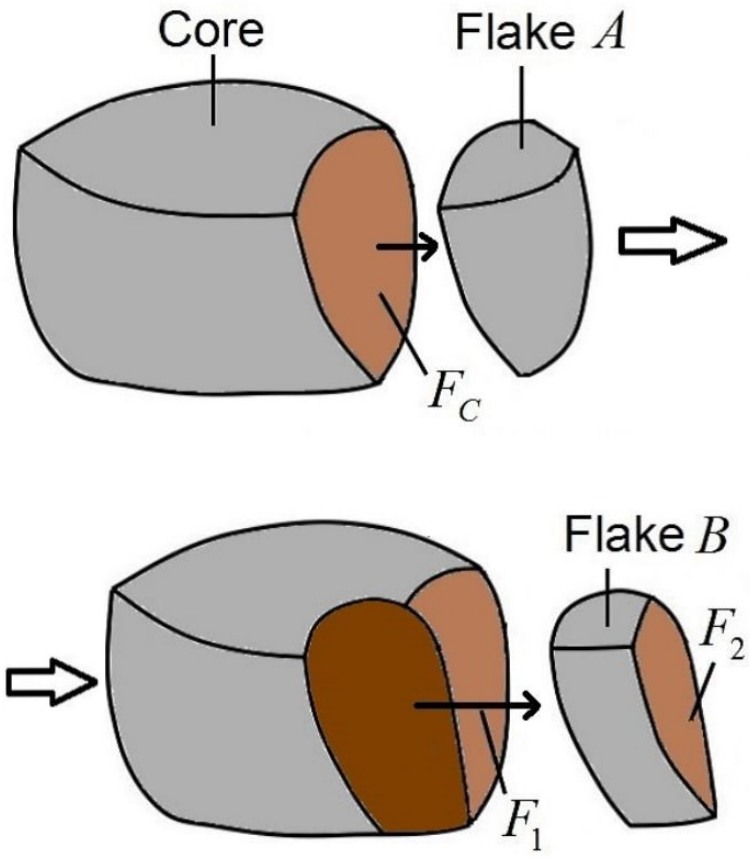



Figure 5.1: Case of making a stone tool (flake $A$ is peeled first followed by flake $B$).

The third property arises when a single flake surface can fracture into several fragments. As shown in Figure 5.1, the matching of flake $A$ requires prior matching of flake $B$ with the core stone to create flake surface $F_c$, which is a composite derived from flake surfaces $F_1$ and $F_2$. Consequently, the matched flake surfaces required a reconstruction process to identify the subsequent surface.

In our approach, stone cores are designated manually, initiating the matching process on the flake surfaces of the core stones. As our dataset comprised multiple core stones, they are reassembled in succession. Each flake surface of the core stone is matched with every flake surface of the flakes to identify the best match. To enhance the efficiency and reduce the number of matching tasks, the candidate order for each flake surface is calculated. This order is sorted based on the corresponding candidate score.

The chapter is structured as follows:

- **New fine-tuned flake surface matching algorithm:** This section provides a detailed explanation of the matching algorithm, including the methodologies and mathematical principles behind it. The focus is on the use of contour points, alignment, and scoring to evaluate and identify the best matches.
- **Flake surface reconstruction:** This section discusses the reconstruction process following the matching. It covers the verification of matches, the accurate alignment of surfaces, and the maintenance of the overall integrity of the reconstructed stone tools.

The algorithm leverages the result from the normal vector estimation and flake surface extraction process detailed in previous chapters, addressing the challenges posed by the irregular shapes and smooth surface of stone fragments.

## 5.2 New fine-tuned flake surface matching algorithm

A fine-tuned matching algorithm is developed for two purposes during reassembly. It is designed to calculate both the candidate and matching scores in pairwise matching. The matching algorithm calculates a matrix $\mathbf{M}_{st}$, by which a source surface in the candidate stone is mapped onto a target surface in the core stone. The algorithm calculates both the candidate and matching scores by tuning the matching parameter $n$, which determines the level of detail during the matching process.

Figure 5.2 shows the difference between the candidate and matching scores for different values of $n$. For instance, when the matching parameter $n$ is set to 6, the algorithm works on all combinations between every 6 points in the target surface and source contours. When it is set to 12 as an example, the algorithm works on all combinations between every 12 points in the target and source contours. In this case, the number of combinations is reduced, and the working process faster than $n$ is set to 6. In contrast, the accuracy tends

to decrease depending on the contour shape. Therefore, $n$ is determined by considering the balance of speed and accuracy.

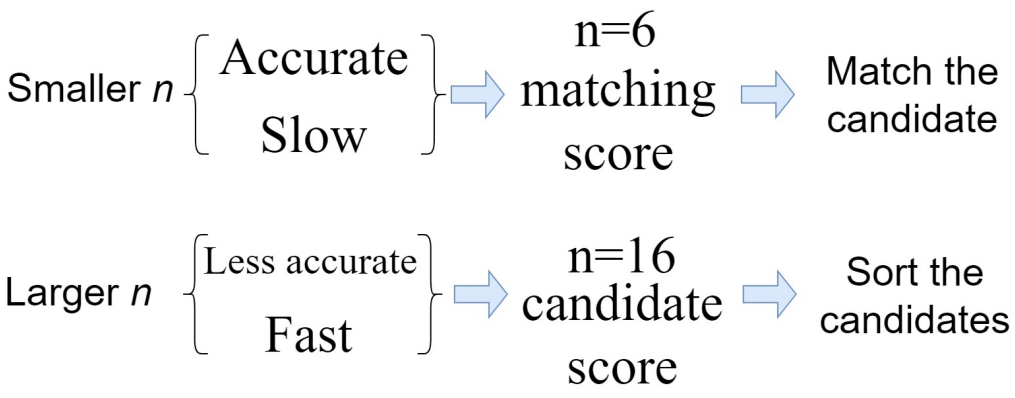



Figure 5.2: Tuning the precision parameter.

Our reassembly method requires less computational time to calculate the candidate score. However, the matching accuracy must be high when refitting a candidate for the core stone and calculating the matching score.

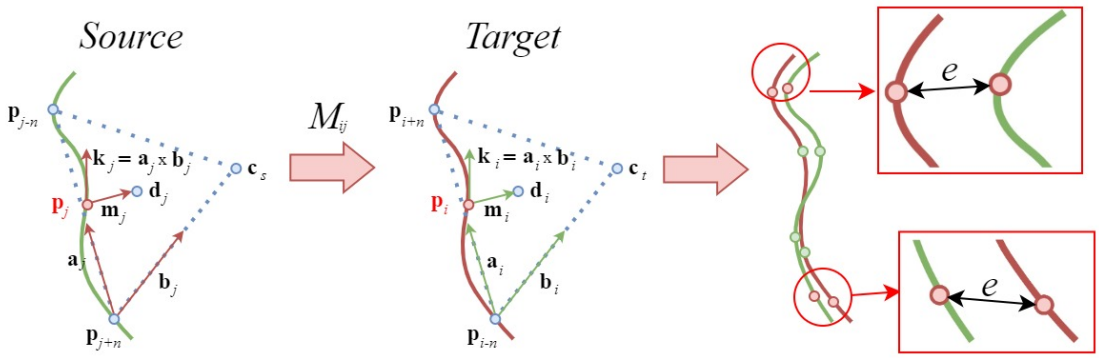



Figure 5.3: Mapping a source into a target surface.

A five-point set $\{\mathbf{p}_i, \mathbf{p}_{i+n}, \mathbf{p}_{i-n}, \mathbf{d}_i, \mathbf{c}_t\}$ is constructed, where $\mathbf{p}_i$ is $i$-th contour point of the target surface $t$, $n$ is the matching parameter, $\mathbf{p}_{i+n}$ and $\mathbf{p}_{i-n}$ are separated from $\mathbf{p}_i$ by $n$, $\mathbf{c}_t$ denotes the center point of the target surface, and $\mathbf{d}_i$ denotes the centroid point of the triangle whose vertices are $\mathbf{p}_{i+n}$, $\mathbf{p}_{i+n}$, and $\mathbf{c}_t$. Similarly, another five-point set $\{\mathbf{p}_j, \mathbf{p}_{j+n}, \mathbf{p}_{j-n}, \mathbf{d}_j, \mathbf{c}_s\}$ is constructed in the same manner on the source surface $s$, as shown in Figure 5.3.

The normal vector $\mathbf{k}_i$ of the triangle, control vector $\mathbf{m}_i$, and centroid $\mathbf{d}_i$ of the triangle are calculated using Equations (5.1), (5.2), and (5.3), respectively.

All the points and vectors are constructed in the same manner on the source surface. The 3D rotation matrix $\mathbf{R}_{ij}$ is determined by satisfying Equation (5.5) owing to $\mathbf{k}_i \perp \mathbf{m}_i$ and $\mathbf{k}_j \perp \mathbf{m}_j$. The 3D translation matrix $\mathbf{T}_{ij}$ is calculated using Equation (5.4). The overall transformation matrix $\mathbf{M}_{ij}$ is thereafter computed using Equation (5.6), integrating both rotation and translation to facilitate transformation in a 3D space.

$$\mathbf{k}_i = (\mathbf{c}_t - \mathbf{p}_{i-n}) \times (\mathbf{p}_{i+n} - \mathbf{p}_{i-n}) \tag{5.1}$$

$$\mathbf{d}_i = (\mathbf{p}_{i+n} + \mathbf{p}_{i-n} + \mathbf{c}_t)/3 \tag{5.2}$$

$$\mathbf{m}_i = \mathbf{d}_i - \mathbf{p}_i \tag{5.3}$$

$$\mathbf{d}_i = \mathbf{T}_{ij} \cdot \mathbf{d}_j \tag{5.4}$$

$$\begin{cases} \mathbf{k}_i = \mathbf{R}_{ij} \cdot \mathbf{k}_j \\ \mathbf{m}_i = \mathbf{R}_{ij} \cdot \mathbf{m}_j \end{cases} \tag{5.5}$$

$$\mathbf{M}_{ij} = \mathbf{T}_{ij} \cdot \mathbf{R}_{ij} \tag{5.6}$$

Equation (5.7) computes error metric $e$ at point $\mathbf{p}_i$, which is the contour point of the target surface and determines the minimum distance between $\mathbf{p}_i$ and any contour point $\mathbf{p}'_j$ on the transformed source surface by $\mathbf{M}_{ij}$. The number of matched target contour points $r_{ij}$ is calculated using Equation (5.8), where $E_s$ denotes the maximum edge length in the concave hull [32]. The matching transformation matrix $\mathbf{M}_{st}$, which aligns the source surface $s$ to the target surface $t$, is selected based on $max\_r_{st}$, which is the highest value of $r_{ij}$ as shown in Equation (5.9). Algorithm 1 and 2 outline our flake surface matching algorithm; $n_t$ and $n_s$ denote the respective numbers of contour points; $\mathbf{c}_t$ and $\mathbf{c}_s$ denote the center points; $n$ denotes the matching parameter in the input; $\mathbf{M}_{st}$ denotes the matching transformation; $max\_r_{st}$ denotes the highest number of matched contour points in the output.

$$e(\mathbf{p}_i) = \min(|\mathbf{p}_i, \mathbf{p}'_j|); \quad j, i+ = n \tag{5.7}$$

$$r_{ij} = \sum_{i+=n} \begin{cases} 1, & \text{if } e(\mathbf{p}_i) < n \cdot E_s, \\ 0, & \text{otherwise} \end{cases} \tag{5.8}$$

$$max\_r_{st} = \max(r_{ij}) \tag{5.9}$$

**Algorithm 1:** Flake Surface Matching Algorithm Part 1

**Input :** contour points of the target, $n_t$, $\mathbf{c}_t$, contour points of the source, $n_s$, $\mathbf{c}_s$, $n$

**Output:** $\mathbf{M}_{st}$, $max\_r_{st}$

$\mathbf{M}' \leftarrow \emptyset$;

**for** $i \leftarrow 0$ **to** $n_t$ **do**

  $\mathbf{p}_i \leftarrow i$-th contour point of the target;

  **for** $j \leftarrow 0$ **to** $n_s$ **do**

    $\mathbf{p}_j \leftarrow j$-th point of the source;

    $\mathbf{M}_{ij} \leftarrow$ getTransformMatrix$(\mathbf{p}_i, \mathbf{c}_t, \mathbf{p}_j, \mathbf{c}_s, n)$;

    `// by Equation 5.6`

    $\mathbf{M}' \leftarrow \mathbf{M}' \bigcup \mathbf{M}_{ij}$;

    $j += n$;

  **end**

  $i += n$;

**end**

The uniform point density of the dataset enables a straightforward representation of the matched area using the number of corresponding points. In this context, $\mathbf{q}_i$ denotes any point on the target surface $t$, whereas $\mathbf{q}_j$ is a point on the source surface $s$ that is nearest point to the tangent plane of $\mathbf{q}_i$.

The two orthogonal distances can be calculated by surface $s$ and $t$. One is the distance from the tangent plane of $\mathbf{q}_i$ to $\mathbf{q}_j$, and the other is the distance from the tangent plane of $\mathbf{q}_j$ to $\mathbf{q}_i$. In our method, if both distances are less than $d_c$ and the distance between $\mathbf{q}_i$ and $\mathbf{q}_j$ is also less than $d_c$, $\mathbf{q}_i$ is identified as a correspondence point of $\mathbf{q}_j$. $d_c$ is determined 1.5 by experiment.

Subsequently, matched point pairs of surface $s$ and $t$ are derived and the matched surface percentages are determined using Equation (5.10). In this

**Algorithm 2:** Flake Surface Matching Algorithm Part 2

$r_{st} \leftarrow 0$;
**foreach** $\boldsymbol{M}_{ij}$ ***in*** $\boldsymbol{M}'$ **do**
  $S' \leftarrow \{\mathbf{p}'_j \mid \mathbf{p}'_j = \mathbf{M}_{ij} \cdot \mathbf{p}_j,\ \mathbf{p}_j$ is any contour point of the source $\}$;
  $r_{ij} \leftarrow 0$;
  **for** $i \leftarrow 0$ **to** $n_t$ **do**
    $\mathbf{p}_i \leftarrow i$-th contour point of the target;
    $e = \min \|\mathbf{p}_i - \mathbf{p}'_j\|, \quad \mathbf{p}'_j \in S'$;
    **if** $e < E_s \cdot n$ **then**
      $r_{ij} += 1$;
    **end**
    $i += n$;
  **end**
  **if** $r_{ij} > max\_r_{st}$ **then**
    $max\_r_{st} \leftarrow r_{ij}$;
    $\mathbf{M}_{st} \leftarrow \mathbf{M}_{ij}$;
  **end**
**end**

equation, $P_u$ denotes the matched area percentage of the surface $u$, $N_u$ denotes the number of corresponding points of the surface $u$, $T_u$ denotes the total number of points on surface $u$. Specifically, $P_t$ and $P_s$ are the instances of $P_u$ for the target surface $t$ and $s$, respectively. $P_t$ and $P_s$ are indicated the percentage of each matched surface. Moreover, the matching score, denoted as $S_{st}$, or candidate score, is computed using Equation (5.11), where $max\_r_{st}$ denotes the number of matched contour points, and $P_t$ and $P_s$ represent the matched surface percentage of the target and source surfaces, respectively.

A higher $S_{st}$ value indicates a greater level of matching between the target $t$ and source $s$ surfaces, implying that there are more matched contour points and a larger matched area between them. Conversely, a lower $S_{st}$ value implies a lower matching level, indicating fewer matched contour points and a smaller matched area between the surfaces.

$$P_u = \frac{N_u}{T_u} \tag{5.10}$$

$$S_{st} = max\_r_{st}\sqrt{P_t P_s} \tag{5.11}$$

## 5.3 Flake surface reconstruction

The process of reconstructing the original flake surface involves the detection and integration of divided flake surfaces. Figure 5.4 shows a visual representation of the reconstruction, with the flake surfaces $F_a$ in blue and $F_b$ in green. The reconstruction proceeds through the following steps:

1. Each pair of flake surfaces $F_a$ and $F_b$ from matched flakes undergoes a search for the closest contour point $\mathbf{p}_{bj}$ of $F_b$ for each contour point $\mathbf{p}_{ai}$. If the distance between $\mathbf{p}_{ai}$ and $\mathbf{p}_{bj}$ is less than the threshold named $d_r$, the pair is designated as a corresponding contour point pair, red points in Figure 5.4.

2. Calculate the angle between $\mathbf{n}_a$ and $\mathbf{n}_b$, average normal vector of $F_a$ and $F_b$, respectively. In addition, the number of corresponding pair is counted.

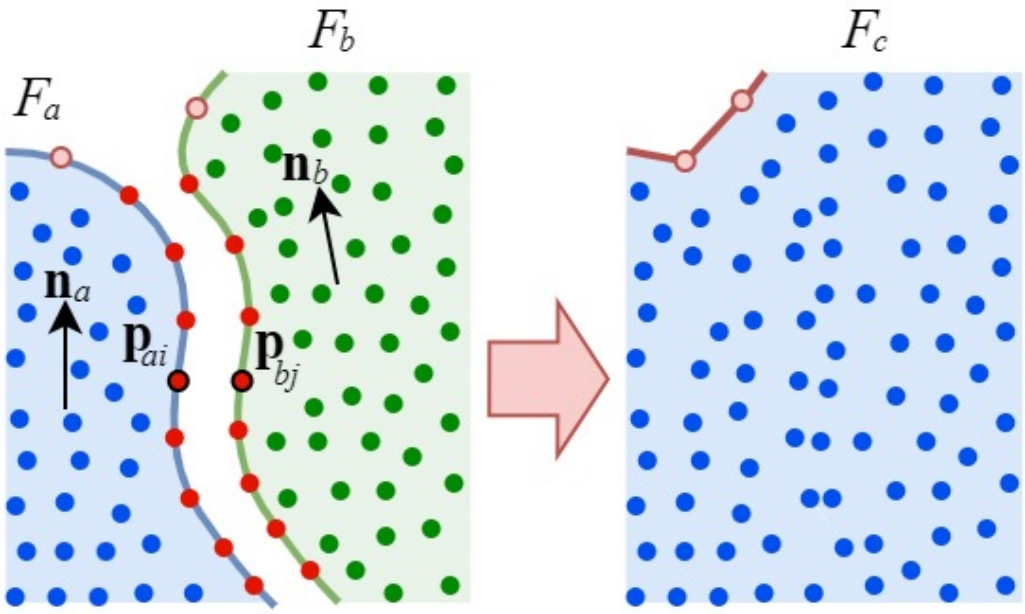



Figure 5.4: Flake surface reconstruction.

3. The corresponding pair derived by Step 2 is higher than the threshold $w$ and the angle between $n_a$ and $n_b$ is less than the threshold $w_\theta$, then the flake surface pair $F_a$ and $F_b$ is merged into a single flake surface $F_c$.

# Chapter 6

# Stone Tool Reassembly system

## 6.1 Overview

The stone tool reassembly system integrates the fine-tuned flake surface matching algorithm into an interactive system designed for archaeologists. This system take as input a set of fragmented stone point clouds, which have been processed through normal estimation and flake surface extraction. The system allows archaeologist to judge and control the reassembly process through a user-friendly interface.

The chapter is structured as follows:

- **Pipeline:** Describes the sequential stages of the reassembly process, from input data preparation to all flakes are assembled.
- **User interface:** Details the features and functionalities of the interactive interface, designed to facilitate user interaction and control during the reassembly process.

## 6.2 Pipeline

Our proposed reassembly system's pipeline builds upon the pipeline outlined in [19]. In the system a core stone is manually selected. The pipeline is executed using the following procedure and illustrated in Figure 6.1:

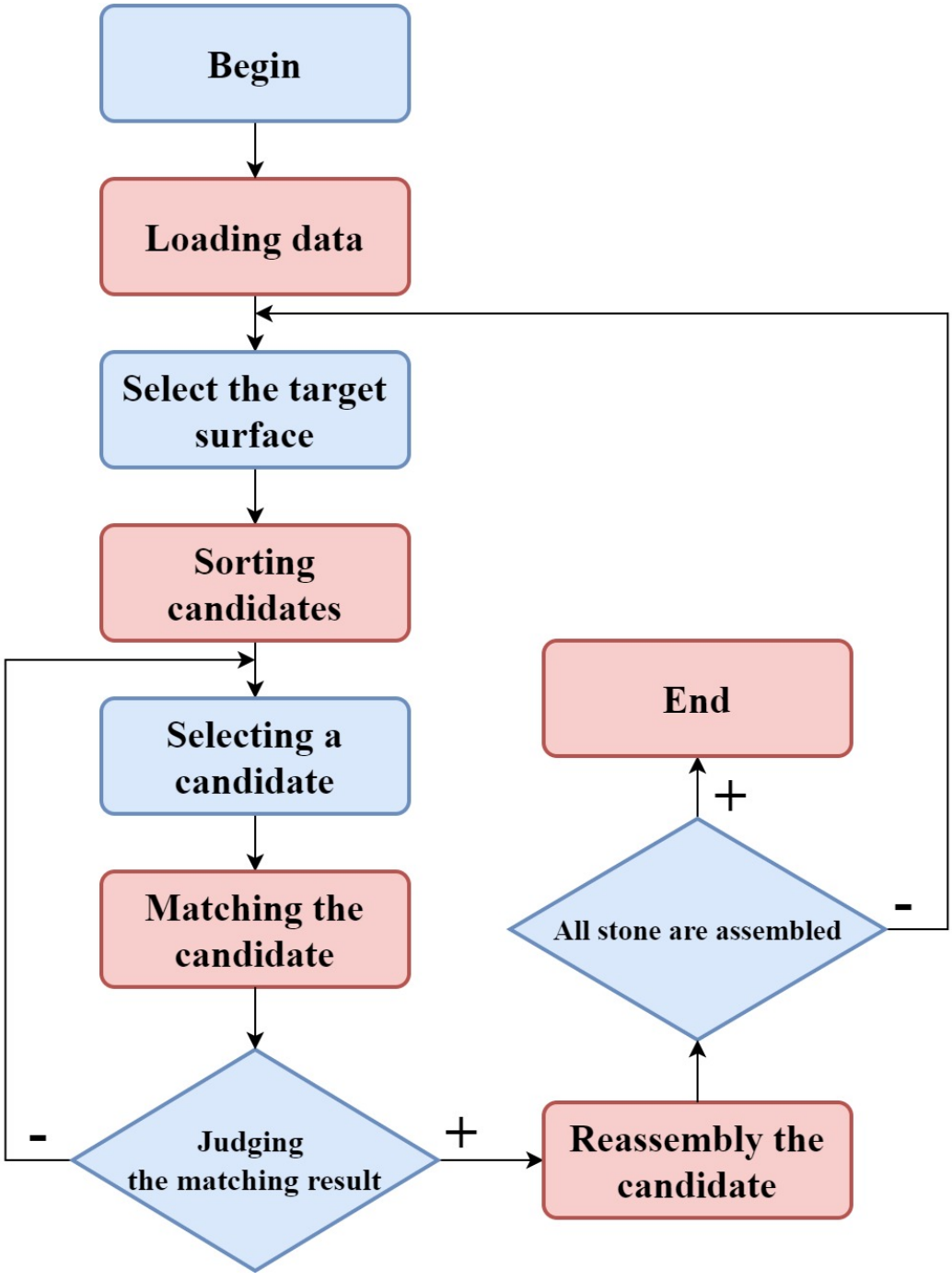



Figure 6.1: The pipeline of the reassembly system.

1. **Loading data.**

   The first step involves loading the point cloud data of the flake stones and the core stone. It has been pre-processed through normal vector estimation and flake surface extraction. The core stone is manually selected.

2. **Selecting a target surface from a core stone.**

   This selected flake surface from the core stone is named the target surface in the matching process.

3. **Sorting flake stones.**

   For each flake surface of candidate flake stones, matching algorithm calculates a candidate score between the flake surface and target surface. The system sorts candidate stone in descending order based on their scores. The flake surface of candidate stone is named the source surface in matching process.

4. **Selecting the candidate stone.**

   The system matches selected candidate on the core stone and calculate matching score between the target and source surfaces.

5. **Judging matching result.**

   The user judges the matching result. If the result is satisfactory, the reassembly process is performed. Otherwise, the user can select other candidates for matching.

6. **Reassembly.**

   The candidate stone is integrated into the core stone through the reconstruction process and reassembly the stone flake.

7. **Iterative matching.**

   The reassembly process is iterated until all candidate stone flake are successfully matched and integrated into the core stone.

## 6.3 User Interface

Our system's user interface comprises two main viewers: a candidate viewer and matching viewer. The matching viewer allows users to examine matching results, whereas the candidate viewer allows for the select of a candidate stone and visualization of individual stone flakes. The fine-tuned matching algorithm is utilized the interactive reassembly system, as shown in Figure 6.2.

The viewers were designed to emulate a professional restoration environment. The candidate viewer serves as a display for individual stone models or flakes that are considered potential candidates for reassembly. It provides a visual representation of these candidate stones, enabling users to examine their features and characteristics and assess their suitability for reassembly purposes.

Similarly, the matching viewer shows the results of the reassembly process on the core stone. It displays the selected candidate stone model and illustrates how they match the core stone using matching algorithms. The matching viewer provides a visual representation of the matching outcome and matching score, allowing users to evaluate the progress and accuracy of the reassembly work. Additionally, users can manipulate the core stone effortlessly using mouse controls, enabling rotation and zooming functionalities for both viewers.

The core stone was initially selected by default and displayed in the matching viewer. A flake surface can be selected by left-clicking the mouse, triggering the calculation of the ordered candidate stones for the selected flake surface. This selected surface was visually distinguished by turning red, whereas the contour points were displayed in blue to aid in observing the surface shape, as shown in Figure 6.2(a).

### 6.3.1 Candidate viewer

Using this calculation, the candidate viewer presents the candidate stone tools sorted based on the candidate score. In the candidate viewer, stone tools are arranged in the recommended sequence and the recommended surface of each stone tool can be examined. The recommended surfaces are highlighted in red,as shown in Figure 6.2(b). Based on the candidate score, users can select

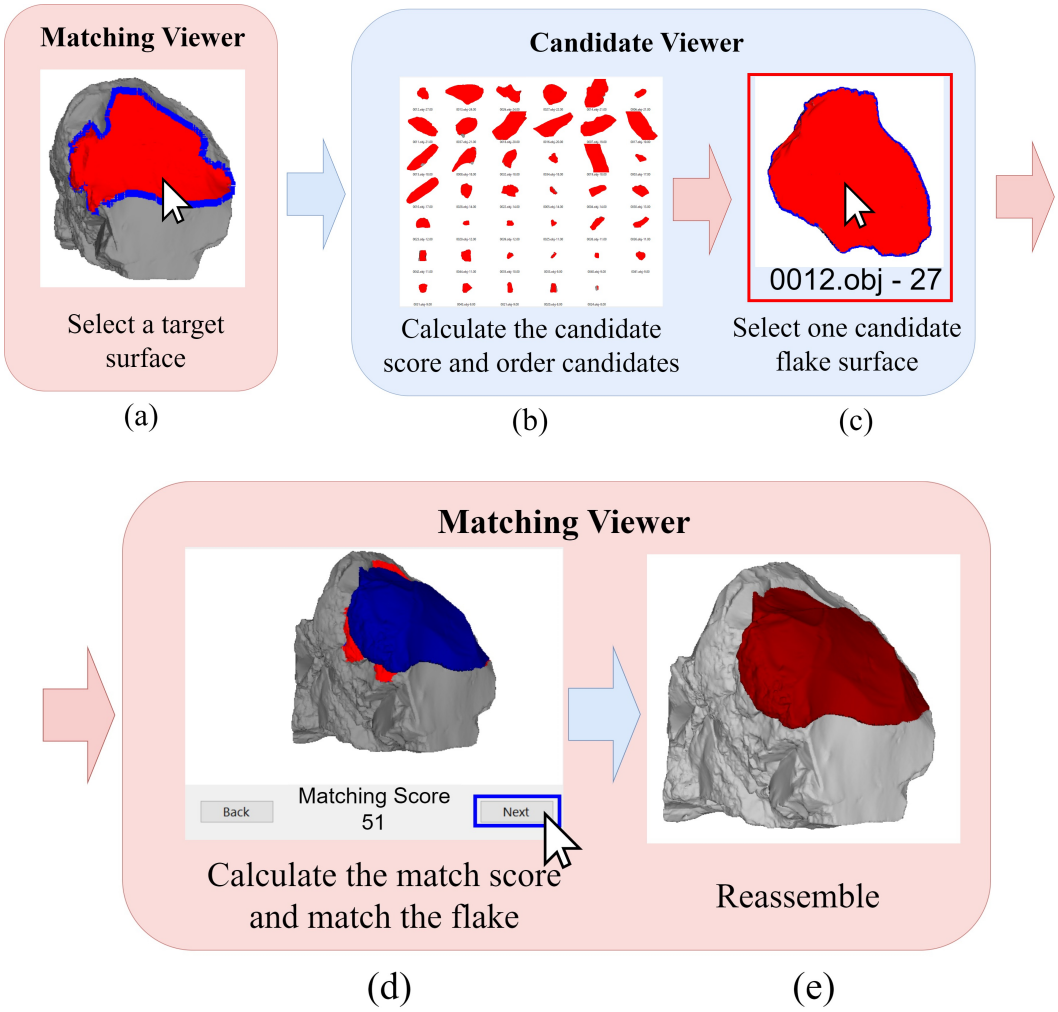



Figure 6.2: Candidate Viewer and Matching Viewer

a candidate stone that matches the core stone, as shown in Figure 6.2(c). This selection initiates the calculation of the matching score and visualization of the matching result. Additionally, information, such as the identification and candidate score of each stone, is presented at the bottom. In Figure 6.2(c), "0012.obj" is the identification of the stone tool, and "27" is the candidate score.

### 6.3.2 Matching viewer

The matching results displayed in the matching viewer and the users can be examined by considering the matching score and visualization. If users are

satisfied with the result, they can reassemble the selected candidate to the core stone by clicking the "Next" button, as shown in Figure 6.2(d). The reassembled core stone can then be saved for subsequent reassembly tasks, as shown in 6.2(e). Finally, to mitigate irreversible errors, users can undo their actions by clicking the "Back" button.

# Chapter 7

# Experimental results

## 7.1 Normal vector estimation

### 7.1.1 Data

The experiments examined 43 point clouds of stone tools. The total number of point is and 69441818 and the averaging 1614926 points per stone point cloud.

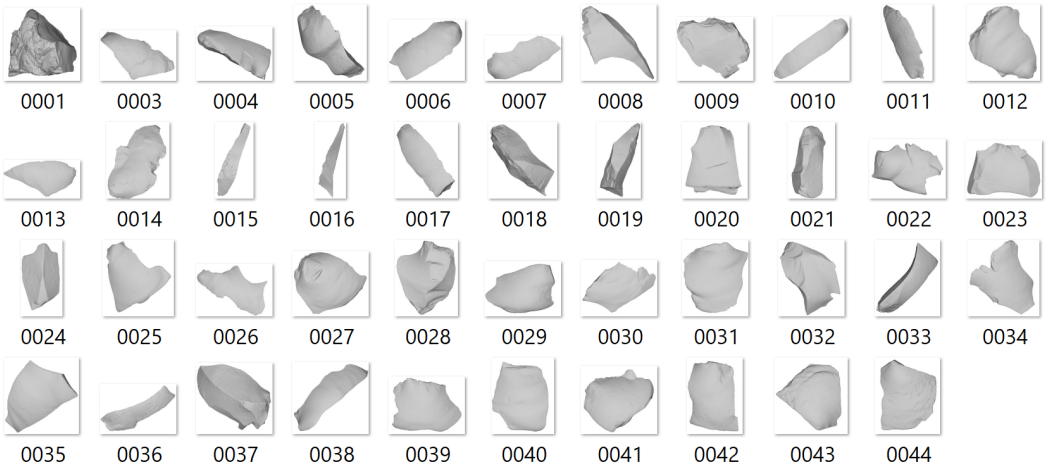



Figure 7.1: Experiment data for the normal estimation

### 7.1.2 Results

We implemented the algorithm in C++ and tested it on a computer with an Intel Core i7-10700 CPU and 16 GB RAM. The accuracy is evaluated to compare the normal vector derived by the mesh and our method's normal vector. The vertex normal vector on the mesh is derived from the average of the face normal vectors connected to the vertex.

For an accurate evaluation, the angle difference between the normal vector from the vertex of the mesh and the normal vector from our result must be evaluated using the angle difference. If the angle difference is less than the threshold value 90°, we estimate that the normal vector of proposed method is correct.

As a result, the average accuracy of our algorithm is 99.98%, and the execution time for each point cloud is presented in Table 7.1. Four-point clouds are randomly selected to show the appearance of the point clouds, as shown in Figure 7.2, with 0001, 0003, 0007, and 0034 being 99.99%, 99.97%, 99.99%, and 99.99% respectively. During the experiment, the tolerance value of voxel size $q$ is manually set to 0.05, and the number of iterations to find a suitable voxel size is at most 2.

In Table 7.1, total time is calculated by summing up value of PCA time column and time column. Value of time column in Table 7.1 is time of finding normal vector orientation using voxelization. The average total execution time is 30.13s, and the average number of point is 1614926. PCA execution is 21.9s, 73% of total time, and our method except for PCA consists of 8.25s, 27% of total time. If PCA becomes fast, whole performance is improved. This result can be regarded as being sufficiently fast for practical usage.

### 7.1.3 Discussion

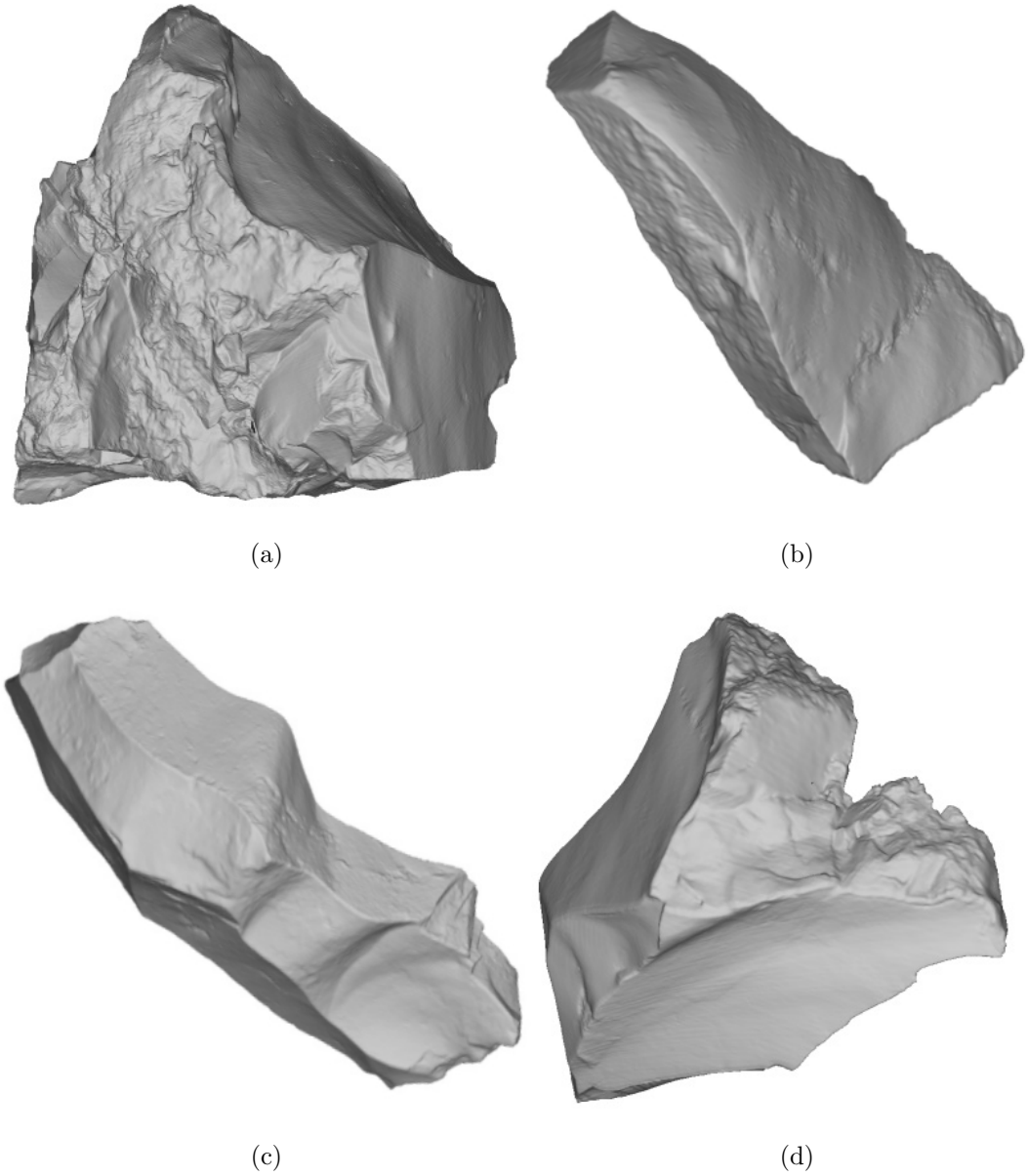
Our method has two limitations. First, its accuracy heavily depends on the voxel size. Table 7.2 shows the relation between accuracy and voxel size of the 0013 point cloud. Accuracy decreases with increasing voxel size, as shown Table 7.2. Accordingly, the accuracy of the 0037 point cloud is 99.88% when the voxel size is 0.5 in Table 7.1. It is the lowest accuracy of Table 7.1. However, 0.5 is the minimum voxel size for closed surface for the point cloud. It depends on the density of the point cloud.

| Stone | Points | Voxel size | It | Time (s) | Acc | Neighbors | PCA time (s) | Total time (s) |
|---|---|---|---|---|---|---|---|---|
| 0001 | 9204822 | 0.8 | 2 | 13.5 | 99.99% | 90 | 122.9 | 136.4 |
| 0003 | 1088935 | 0.2 | 1 | 7.2 | 99.97% | 60 | 11.1 | 18.2 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | |
| 0013 | 2806051 | 0.25 | 1 | 13.5 | 99.92% | 90 | 41.4 | 54.9 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | |
| 0036 | 618684 | 0.15 | 2 | 3.6 | 100% | 60 | 6.1 | 9.7 |
| 0037 | 1604976 | 0.5 | 2 | 1.8 | 99.87% | 90 | 23.7 | 25.5 |
| 0038 | 287133 | 0.15 | 2 | 4.2 | 99.96% | 90 | 3.5 | 7.6 |
| 0039 | 379540 | 0.2 | 2 | 2.1 | 99.98% | 90 | 7.9 | 10.0 |
| 0040 | 356685 | 0.15 | 1 | 1.3 | 99.99% | 30 | 2.5 | 3.8 |
| 0041 | 405475 | 0.15 | 1 | 2.2 | 99.99% | 30 | 2.2 | 4.4 |
| 0042 | 393391 | 0.15 | 1 | 5.6 | 99.99% | 30 | 2.3 | 7.9 |
| 0043 | 360744 | 0.15 | 1 | 1.6 | 99.99% | 30 | 2.3 | 3.9 |
| 0044 | 396679 | 0.25 | 1 | 0.9 | 99.95% | 90 | 7.3 | 8.1 |
| **Avg** | 1614926 | 0.26 | 1.2 | 8.25 | 99.98% | 77 | 21.9 | 30.2 |

Table 7.1: Result Table

Second, for the "OO" combination, orientation of normal vector is determined by the closest voxel of the distance $d_1$ and $d_2$, as shown in Figure 7.3. It fails in some points near to the sharp edge.

This study estimates a normal vector directly from a 3D point cloud. Our method enables the normal vector calculation with an accuracy of more than 99.88% and its average accuracy of 99.98%. This result can be used for further processing, such as segmentation of stone tools.



(a) (b)

(c) (d)

Figure 7.2: Result of our method: a) 0001, a) 0003, a) 0007, a) 0034

## 7.2 Flake surface extraction and contour point identification

### 7.2.1 Data

The experiment involved testing data from 43 stone models, as shown in Figure 7.4. There were 12,300,222 points, with an average of 286,052 points

| Voxel size | Accuracy |
|---|---|
| 0.25 | 99.992% |
| 0.3 | 99.988% |
| 0.35 | 99.977% |
| 0.4 | 99.969% |
| 0.45 | 99.956% |
| 0.5 | 99.933% |

Table 7.2: Relation between the voxel size and the accuracy of normal estimation

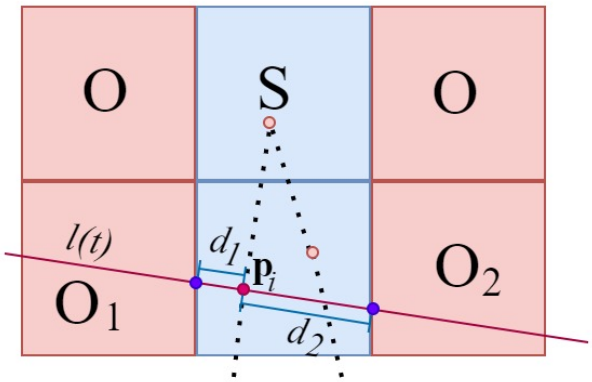



Figure 7.3: Combination "OO"

per stone.

### 7.2.2 Results

The implementation was performed on a PC with an Intel Core i7-10700 CPU and 16 GB memory. We describe the experimental results of our method. The segmentation process for extracting the flake surfaces requires the estimation of point normal vectors and point curvatures. During experiment, the parameters are tuned as follows: the angle threshold $\alpha$ is set to 1.5°, curvature threshold $c$ is set to 0.1, minimum number of points for a valid segment $l$ is set to 500. The number of nearest neighbor $k$ is set to 4 in the boundary correction step. The dataset in Figure 7.4 is segmented, extracting 311 flake

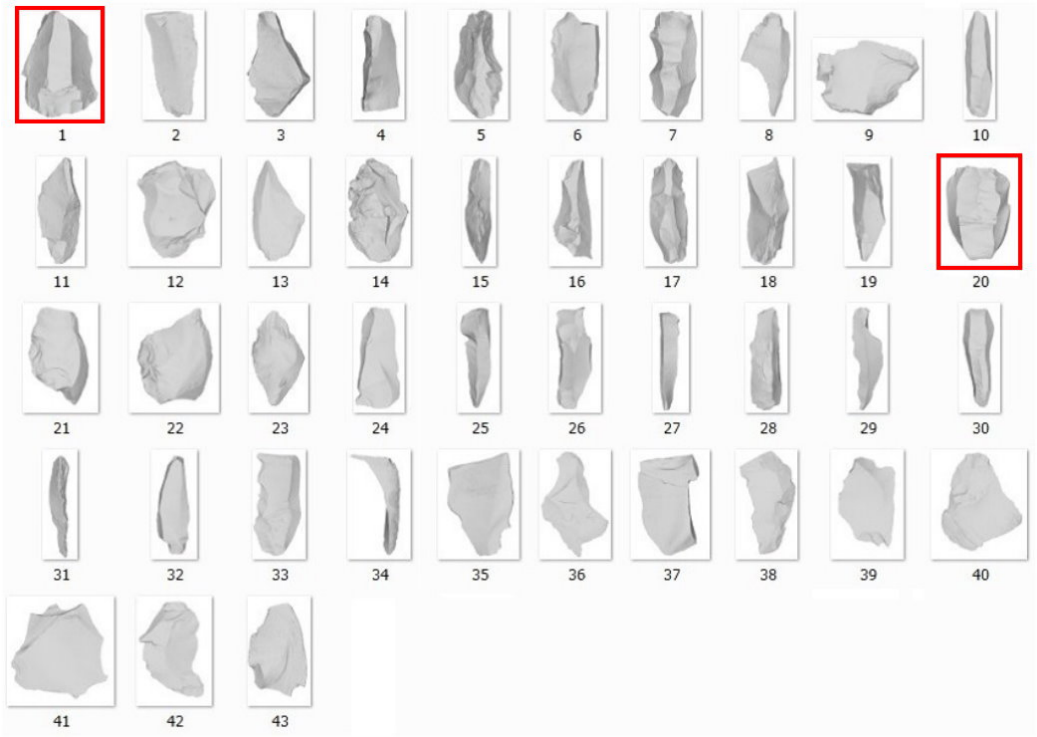
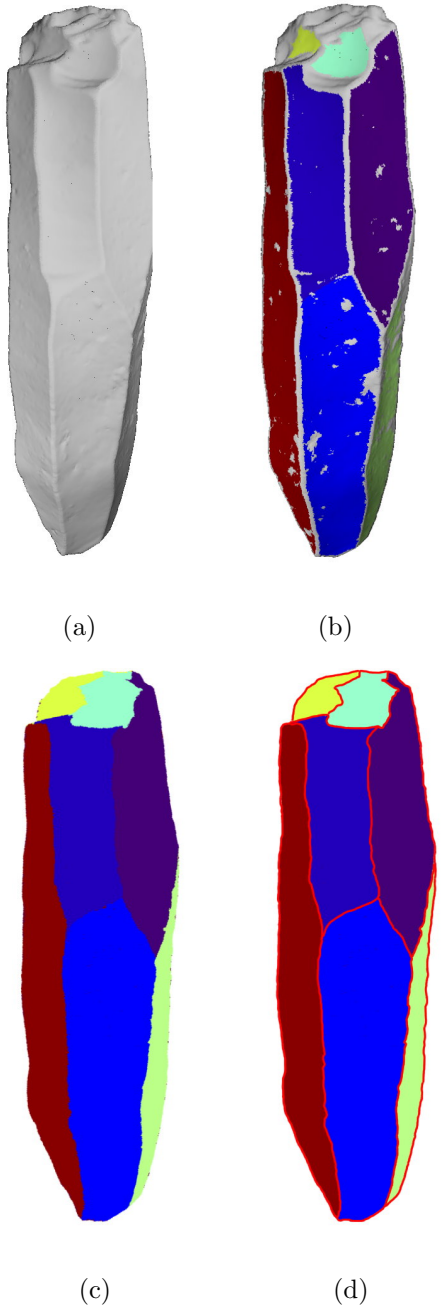



Figure 7.4: Experiment data.

surfaces and averaging 7 surfaces per model for the matching process. Figure 7.5 shows the segmentation results for point cloud $No.10$. Figure 7.5(a) displays the point cloud with point normal vectors, while Figure 7.5(b) shows the outcome of region growing segmentation using [28], where unsegmented points are shown in gray color, resulting in 67962 unsegmented points of 276336. In contrast, Figure 7.5(c) shows our boundary correction result, which extracts 8 flake surfaces, and all points are segmented. In Figure 7.5(b), (c), and (d), the various colors represent different flake surfaces, and the contour points are highlighted in red. Figure 7.6 (a) and (b) show the results of the segmentation for stones $No.06$ and $No.07$, respectively. The left side of each figure shows the result of the region growing, and the right shows the result of the boundary correction step. As shown in Figure 7.5 and 7.6, our method enables to correct boundary.



(a) (b)

(c) (d)

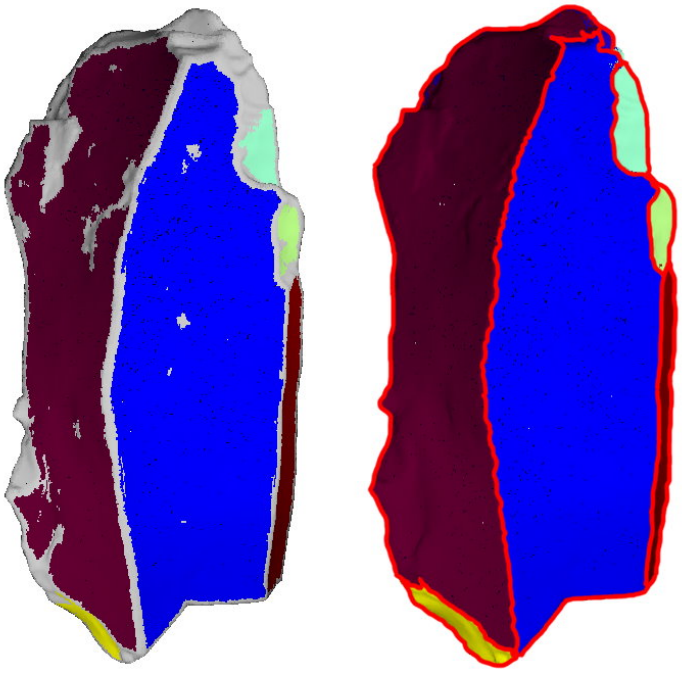
Figure 7.5: Result of the boundary correction.

a) Point cloud.

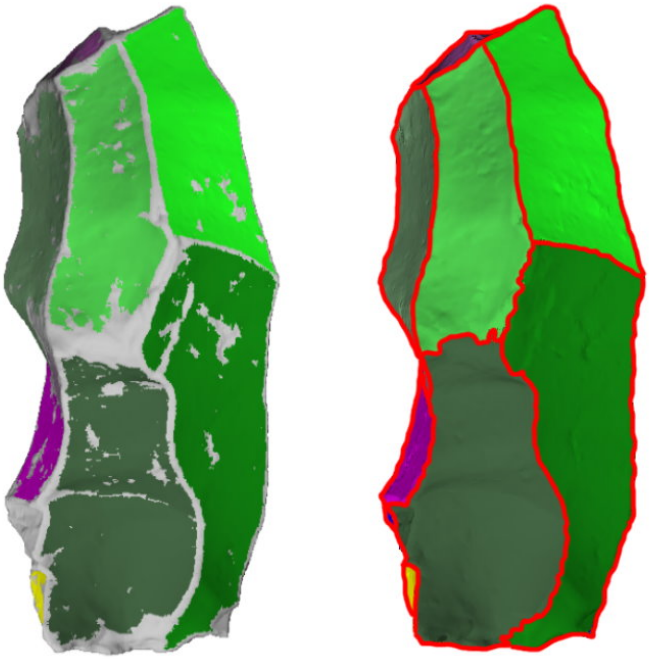
b) Result of region growing segmentation.

c) Result of our boundary correction step.

d) Result of extracting contour points.



(a)



(b)

Figure 7.6: Result of the flake surface extraction.

a) *No.*06.

b) *No.*07.

## 7.3 Fine-tuned flake surface matching algorithm

### 7.3.1 Data

The experiment involved testing data from 43 stone models, as shown in Figure 7.4. The data could be reassembled into three groups. $No.01$ and $No.20$, shown in thick red box are the core stones and Group 1 and 2, respectively. The stone tools belongs to Group 3 are role of dummy data for assembling Group 1 and Group 2. There were 12,300,222 points, with an average of 286,052 points per stone.

### 7.3.2 Evaluation

This section describes the evaluation and experimental results. First, we examined an evaluation score and execution time according to the tunable matching parameter $n$, when $n$ is changed from 1 to 12. Second, we compared our method with related methods. Thereafter, we presented experimental results. Finally, we compared our results with the previous study [19].

To assess the accuracy of the resulting shape, we calculated the evaluation score which is how the matching surface shape coincides. To achieve this, Equation (5.11) which indicates the surface coincident equation between surface $s$ and surface $t$, was used as follows:

$$S_e = S_{st}\big|_{n=1} \tag{7.1}$$

where $S_e$ means score of the surface coincident. To evaluate how to match the surface shape Equation (7.1) was applied to compare the resulting shape of other methods. In Equation (7.1), we set n to 1 to be fair to the shape evaluation when we evaluate the resulting shape. In other words, all points belonging to surfaces $s$ and $t$ are used to evaluate the resulting shape. If $S_e$ is high, it means that the shape is more match.

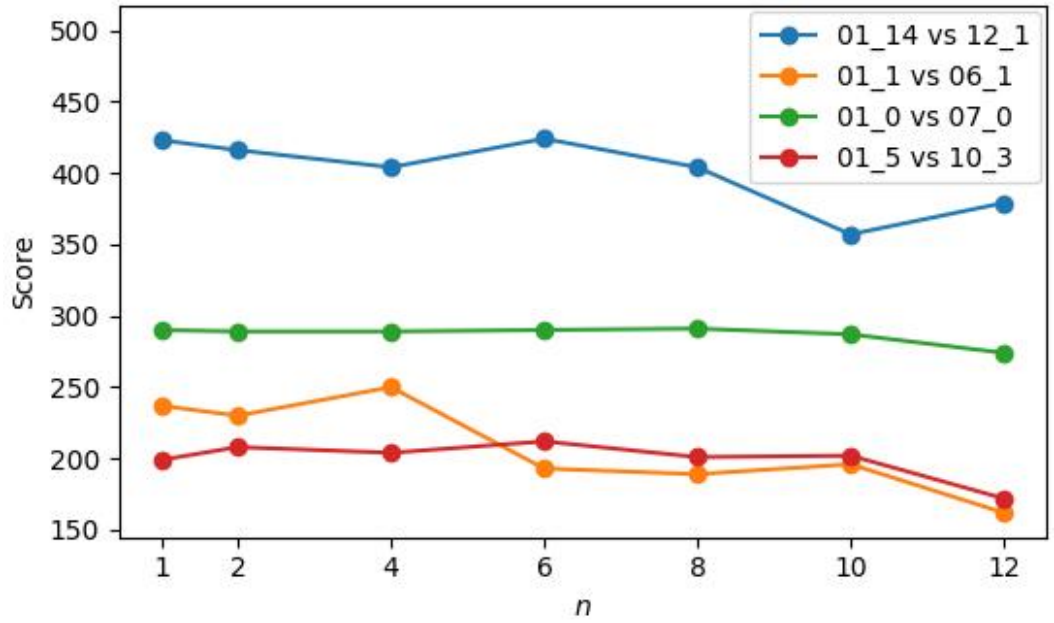
Figure 7.7(a) illustrates that the evaluation score $S_e$ for the four matching pairs when the matching parameter $n$ varied from 1 to 12. Figure 7.7(b) shows the computation time for the same parameter variations. Figure 7.7(a) shows the highest scores for each pair, indicated by red circles. As shown in Figure

7.7(a), three of the four had highest scores when parameter $n$ was set to 6. In contrast, Figure 7.7(b) shows the computation time of the four matching pairs. The average computation time of each matching pair for parameter $n$ =6 was 1.52, and that for $n$ =4 was 6.87 s. This shows that when $n$ =6, the computation time is approximately 4.5 times faster than when $n$ =4. Therefore, to balance computation time with accuracy as reflected by the evaluation score in the performance of the algorithm, we set $n$ =6 for the comparative analysis.

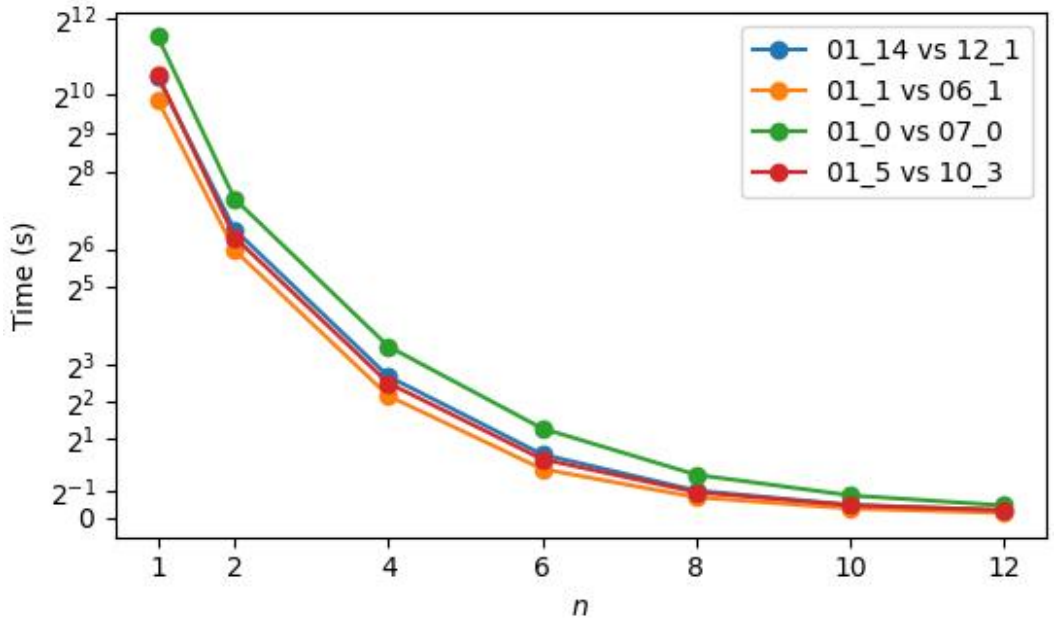
We compare our method and previous works. The comparison involved benchmarking our matching algorithm against Super4PCS [16] and FPFH-SACIA [17] with 100 iterations employed for partial matching tasks. For a fair comparison across all methods, we applied $S_e$, the evaluation score, calculated by Equation (7.1).

The ground truth was manually aligned to imitate the real reassembly results achieved by archaeologists, thereby providing a benchmark for evaluation. Table 7.3 summarizes the results of this comparison. In this table, column "$t$" denotes the ID of the target surface, formed by combining the stone identification and flake surface ID(e.g.,"01_0" indicates core stone ID "01" and flake surface ID "0"). Similarly, column "$s$" refers to the source flake surface ID, following the same identification pattern. "S4PCS" represents the absolute difference between the evaluation score of Super4PCS and the evaluation score of the ground truth. "FPFH" and "Our" indicate the same manner. These differences are calculated using Equation(7.2), where $D$ denotes the absolute difference, $S_{gt}$ denotes the evaluation score of the ground truth, and $S_e$ denotes the evaluation score of the respective algorithm. In this table, the proposed flake surface matching algorithm yields the closest evaluation score to the ground truth across all the target and source pairs. The performance is graphically presented in Figure 7.8. These comparison results highlight that our algorithm performed better than existing methods for flake surface matching, particularly partial matching.

$$D = |S_{gt} - S_e| \tag{7.2}$$

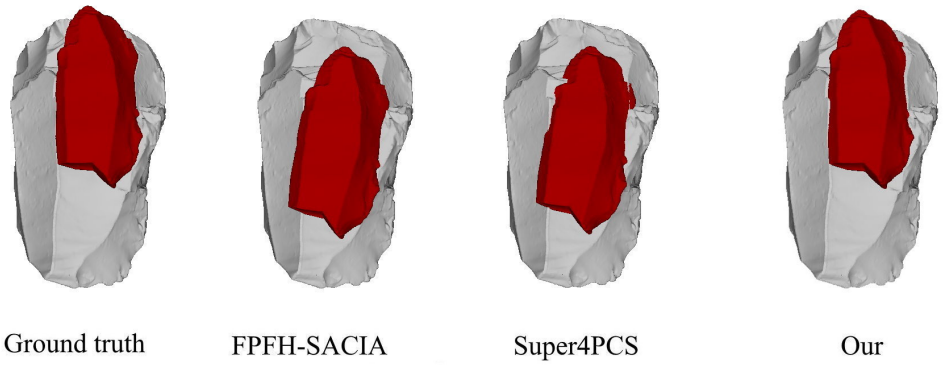



(a) Relationship between $n$ and the evaluation score $S_e$.

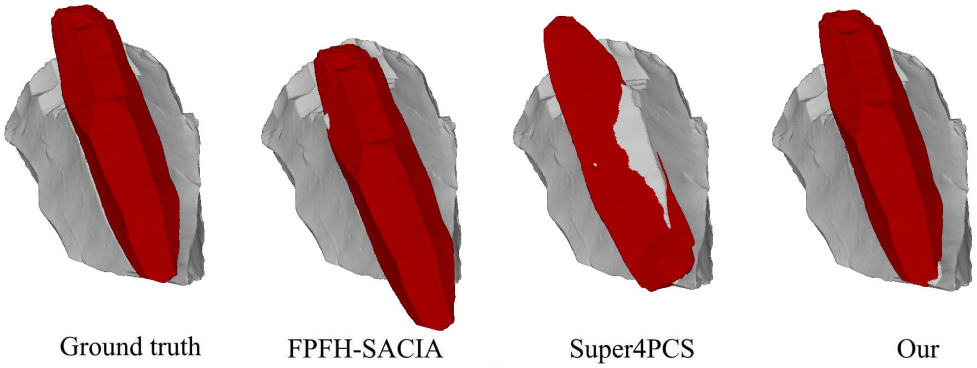



(b) Relationship between $n$ and the computation time.

Figure 7.7: Parameter sensitivity test.




(a) 01_1 vs 06_1.




(b) 01_5 vs 10_3.

Figure 7.8: Visual representation of comparison result.

Table 7.3: The absolute difference between the ground truth and the evaluation score.

| $t$ | $s$ | S4PCS | FPFH | Our |
|---|---|---|---|---|
| 01_14 | 12_1 | 331.4 | 374.6 | 1.7 |
| 01_1 | 6_1 | 142.9 | 115.8 | 19.1 |
| 01_0 | 07_0 | 248.3 | 157.6 | 21.5 |
| 01_5 | 10_3 | 162.8 | 157.2 | 6.5 |

### 7.3.3 Results

Figure 7.9 shows the matching results between the two flake surfaces 01_1 and 06_1. The correlation between the two surfaces is shown in Figure 7.9(b). Green points represent correlation points, red points represent unmatched surface points and yellow points represents matched contour points, and blue points indicate unmatched contour points on the target surface.

Figure 7.10 shows a matching scenario in which flake surfaces require reconstruction for alignment. Three flake surfaces of stone tools $No.10$, $No.06$ and $No.16$ were reconstructed to achieve alignment with the surface of stone $No.17$.

### 7.3.4 Discussion

Figure 7.11 shows a scenario in which proposed method could not correctly match for stone $No.14$. Red circles indicate the correspondence of the correct matching. Matched contour points of the two flake surfaces are indicated in yellow, whereas the unmatched contour points are presented in blue. Our matching method is designed for alignment based on the maximum number of matched contour points $max\_r_{st}$. However, in this case, $max\_r_{st}$ did not achieved the correct matching. Therefore, our method encountered difficulties in this scenario. In the experiments, this case occurred only once.

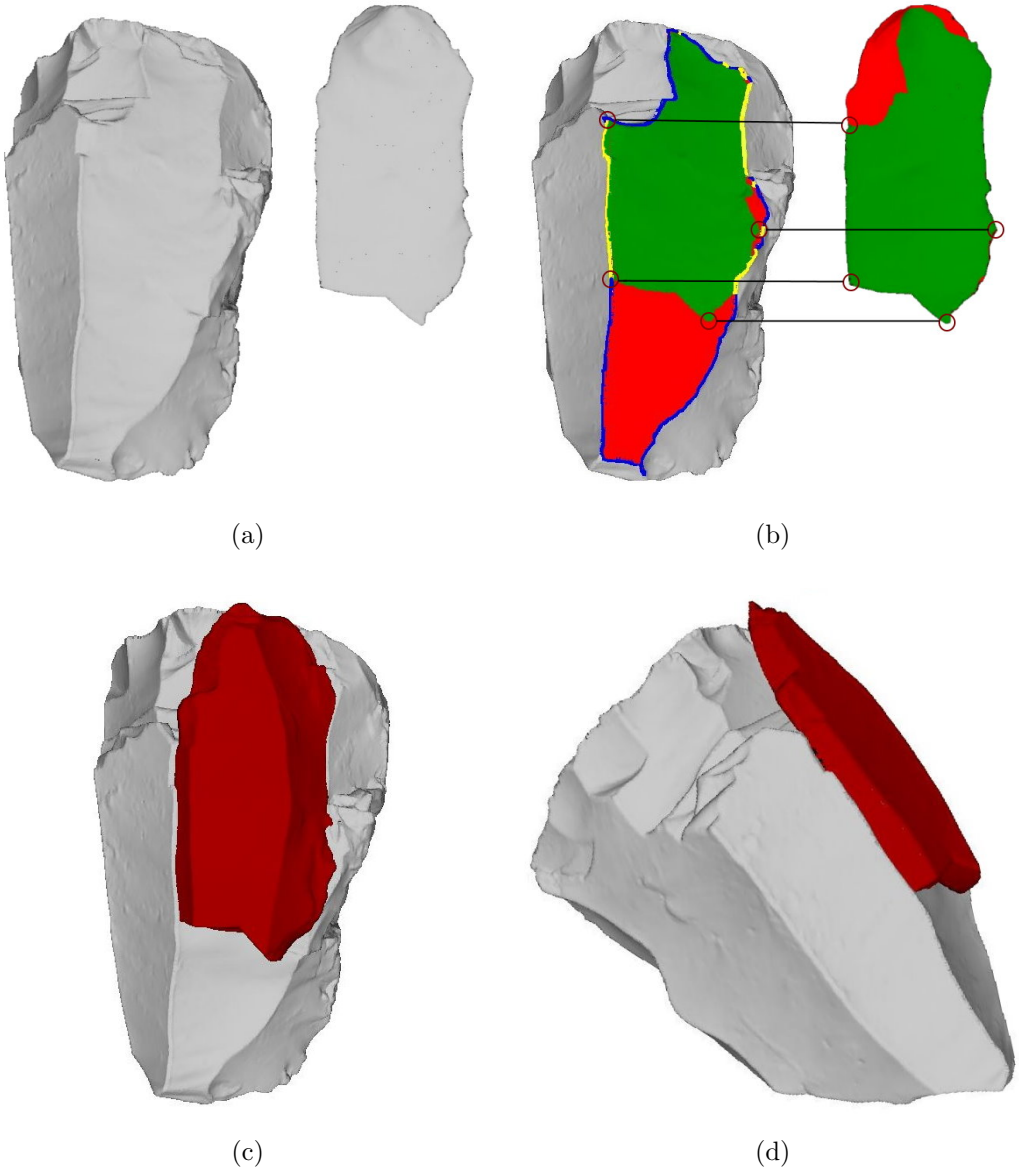
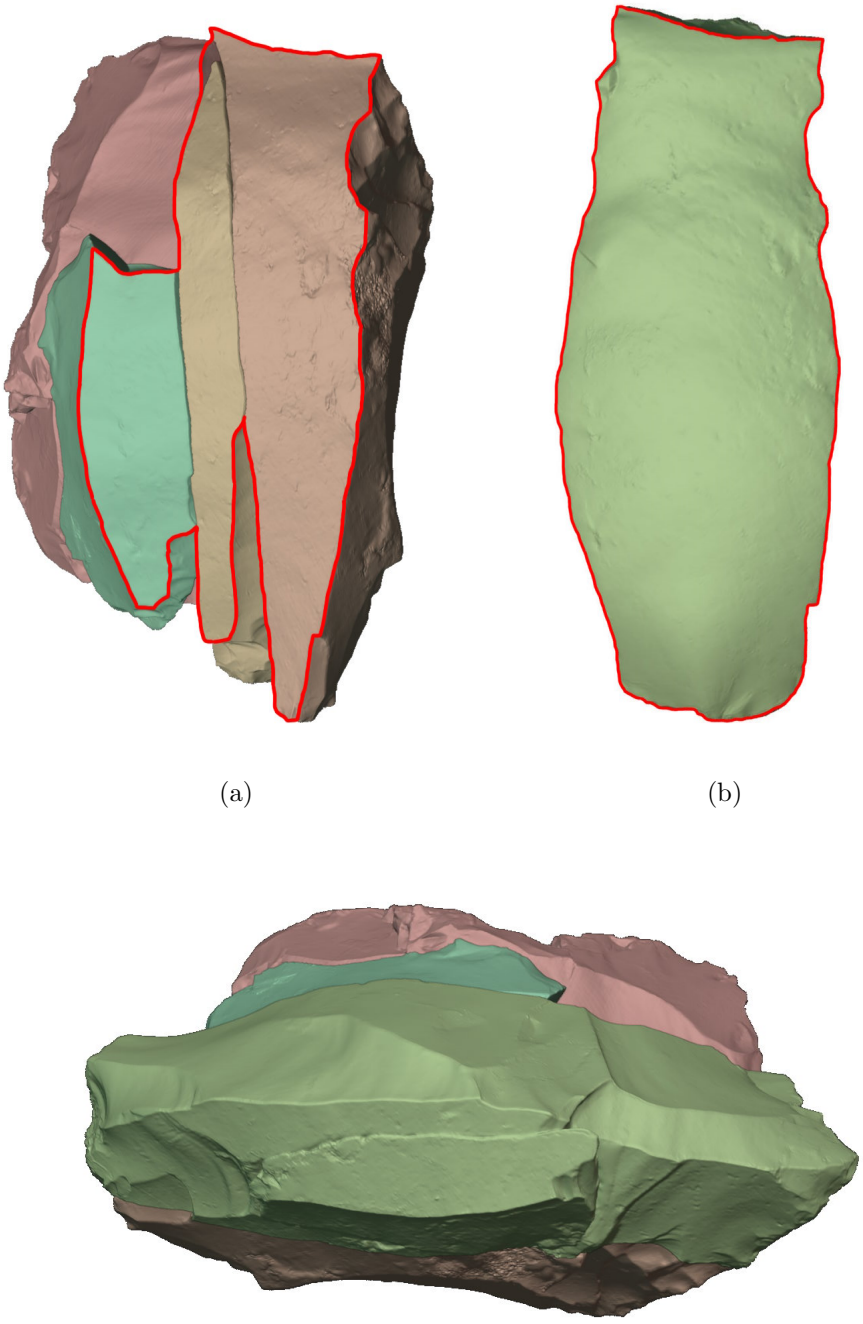


Figure 7.9: Result of flake surface matching.

a) Stone models $No.01$ and $No.06$.

b) Correspondence between flake surfaces 01_1 and 06_1.

c), d) Result of matching in different views.



(a) (b)

(c)

Figure 7.10: Reconstruction result.

a) Reconstructed flake surfaces of *No.*6, *No.*10 and *No.*16. b) Source flake *No.*17. c) Result of matching.

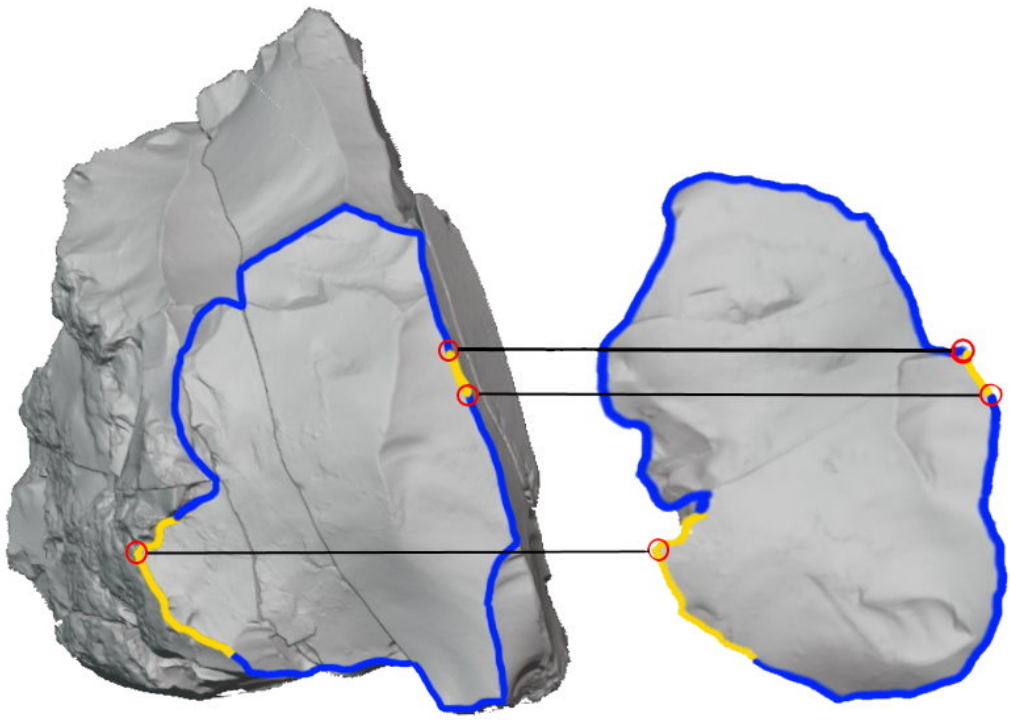


Figure 7.11: Limitation of the matching algorithm: Failed matching scenario for stone *No.*14.

## 7.4 Reassembly system for stone tools

### 7.4.1 Results

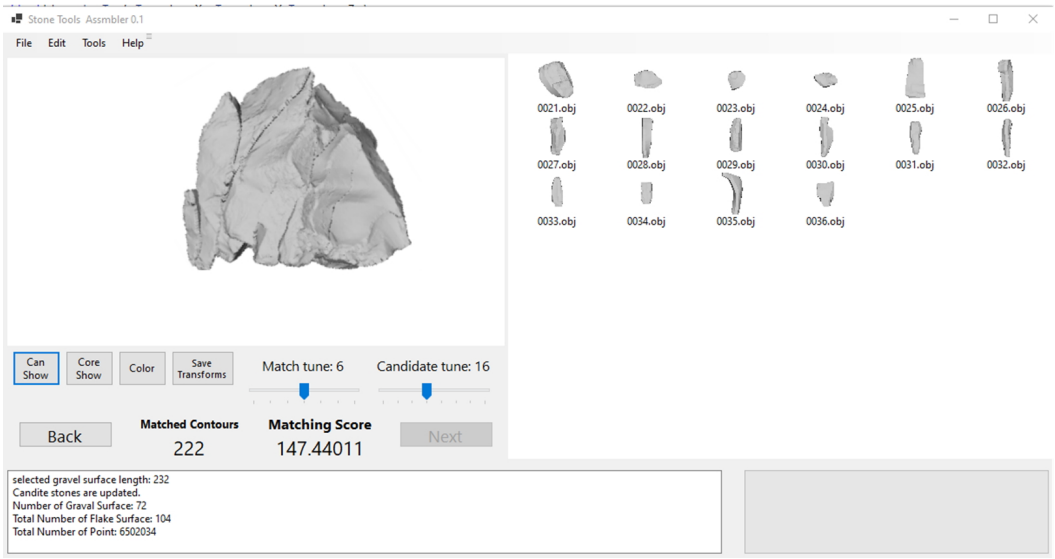



Figure 7.12: Screenshot of the system

The interactive system was implemented using C#. Figure 7.12 shows a screenshot of the system. Tables 7.4 and 7.5 summarize the specific details of matching Groups 1 and 2. In these tables, columns "$t$" and "$s$" maintain the same format as that in Table 7.3, denoting the respective target and source flake surfaces. The "Order" column signifies the order of flake surface for the best matching, determined by the candidate score. "Time" represents the duration required for computing the fitting transformation matrix for each pair. For instance, finding the best match for flake surface 01_1 in Group 1 requires 44.7 s. On average, each matching process uses approximately 7.45 s per pair, considering 6 (candidate order) matching times. All flake surfaces initiated matching from stone core $No.01$ and $No.20$. During the experiment, the maximum edge length in concave hull $E_s$ was set to 0.5 and the matching tune parameter $n$ was set to 6 for matching, 16 for ordering the candidates. Additionally, we defined two distance thresholds, $d_c$ and $d_r$, both set to 1.5. The threshold $w$ is set to 30, and the angular threshold $w_\theta$ is set to 15 degrees.

Group 1 resulted in the reassembly of 16 stone tools, whereas Group 2 was reassembled of 16 stone tools. Flake stone $No.14$ remained unmatched using

our method. Figure 7.13 shows the final matching results for both groups, accompanied by images of the manually matched imitations.

Table 7.6 summarizes the comparison result of our method against our previous study [19] for two groups of stones. In the table, "N.Dis" indicates the normalized distance measurement in millimeters, introduced in previous studies [19], [37]. This measurement method is utilized to identify the most compatible flake surfaces in their studies. It provides a standard to measure the difference between two flake surfaces on a unit area. Essentially, a smaller normalized distance between the surfaces indicates a superior match, signifying a closer similarity or better alignment between two flake surfaces. The column "Time" records the computation time for each method in seconds. The computation time for the previous method is normalized to account for the difference in CPU. Our study was tested on the Intel Core i7-10700, while the previous study utilized an Intel Core i7-4790. The normalization factor of 2.68, derived from the relative floating-point math speed of the two CPUs [38], is applied to the computation times of the previous method to utilize a fair comparison.

In Group 1, our method completed the task in 400 seconds, whereas the previous study required 5912 seconds. In Group 2, our method required 290 seconds, compared with the previous 1641 seconds. In the normalized distance measurements, our method demonstrated an average of 0.012 mm for Group 1, in contrast to 0.033 mm reported in the previous study. Similarly, for Group 2, our method achieved 0.016 mm, compared with 0.027 mm in the previous study. Moreover, we successfully addressed the limitation related to partial matching, allowing us to match 4 more stones in Group 2 and ultimately reassemble all the stones in that group.

| $t$ | $s$ | Order | Score | Time(s) |
|---|---|---|---|---|
| 01_14 | 12_0 | 1/260 | 417.71 | 27.22 |
| 01_1 | 06_0 | 6/250 | 188.40 | 44.70 |
| 01_4 | 10_6 | 2/243 | 306.41 | 20.96 |
| 01_0 | 07_0 | 1/234 | 459.82 | 37.23 |
| 01_49 | 05_1 | 1/225 | 378.01 | 10.37 |
| 01_43 | 16_0 | 2/218 | 219.58 | 29.09 |
| 01_55 | 11_3 | 2/209 | 175.32 | 19.08 |
| 01_39 | 17_0 | 1/205 | 578.64 | 29.75 |
| 01_57 | 02_0 | 4/203 | 140.52 | 15.20 |
| 01_53 | 18_1 | 1/193 | 281.97 | 37.58 |
| 01_59 | 13_1 | 1/187 | 435.26 | 25.39 |
| 01_66 | 08_3 | 1/185 | 554.52 | 11.05 |
| 01_63 | 19_0 | 4/178 | 346.89 | 57.80 |
| 01_64 | 15_0 | 3/172 | 315.48 | 23.83 |
| 01_68 | 04_1 | 6/166 | 147.44 | 11.11 |
| | | | Total time: | 400.36 |

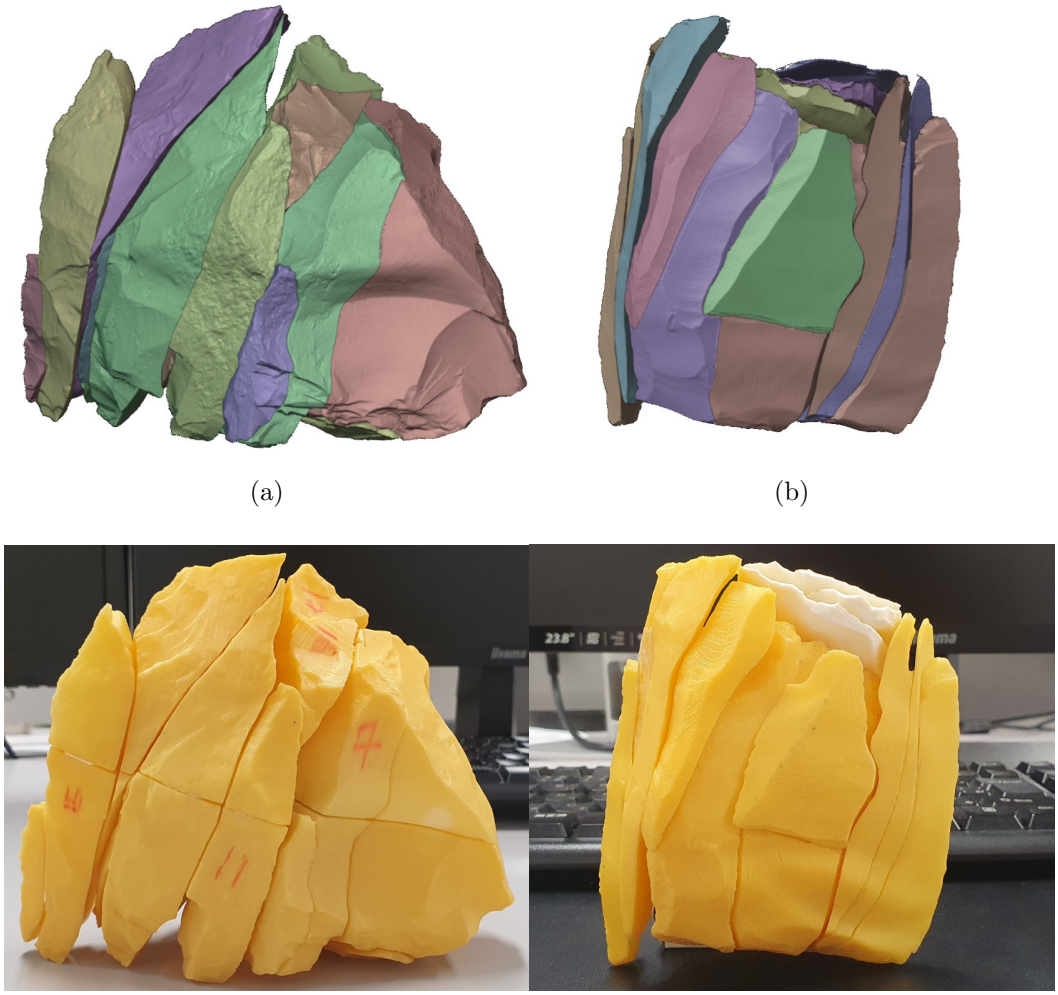
Table 7.4: Reassembly of Group 1

| $t$ | $s$ | Order | Score | Time(s) |
|---|---|---|---|---|
| 20_10 | 34_1 | 1/162 | 442.66 | 7.01 |
| 20_9 | 27_2 | 1/153 | 360.03 | 14.29 |
| 20_10 | 36_0 | 2/146 | 229.89 | 19.12 |
| 20_5 | 33_6 | 1/138 | 331.59 | 6.72 |
| 20_2 | 25_2 | 1/132 | 219.45 | 18.61 |
| 20_7 | 31_1 | 2/127 | 217.92 | 18.76 |
| 20_12 | 26_0 | 4/117 | 196.15 | 51.69 |
| 20_18 | 29_0 | 2/108 | 234.27 | 29.87 |
| 20_33 | 31_0 | 5/102 | 302.02 | 33.77 |
| 20_5 | 30_2 | 3/99 | 278.34 | 17.40 |
| 20_38 | 28_1 | 2/91 | 305.22 | 13.58 |
| 20_43 | 35_2 | 1/83 | 408.63 | 7.76 |
| 20_0 | 22_0 | 3/79 | 166.33 | 18.52 |
| 20_44 | 23_3 | 2/74 | 60.67 | 26.32 |
| 20_54 | 24_1 | 1/64 | 60.12 | 6.57 |
| | | | Total time: | 289.98 |

Table 7.5: Reassembly Group 2

| | | N.Dis (mm) | Time (s) |
|---|---|---|---|
| Group 1 | Our | 0.012 | 400 |
| | Previous | 0.033 | 5912 |
| Group 2 | Our | 0.016 | 290 |
| | Previous | 0.027 | 1641 |

Table 7.6: Comparison table with previous study



(a) (b)

(c) (d)

Figure 7.13: Result of reassembly.

a), b) Reassembled results by our system.

c), d) Reassembled flakes of two groups.

# Chapter 8

# Conclusion and Future work

## 8.1 Conclusions

This thesis presents the interactive system for the reassembly of fragmented stone tools, focusing on the development and application of a fine-tuned flake surface matching algorithm integrated into an interactive reassembly system. The key contributions of this research include:

- **Normal vector estimation:**

  The method for estimating normal vector from 3D point clouds of stone tool fragments using principal component analysis (PCA) and adaptive voxelization. This step ensures precise surface orientation, enabling for subsequent process, such as flake surface extraction and the fine-tuned flake surface matching algorithm.

- **Flake surface extraction and contour point identification:**

  The process involving region growing segmentation, boundary correction, and the identification of contour points of the flake surface. These step are crucial for accurately defining flake surfaces, facilitating for subsequent flake surface matching algorithm and the interactive system.

- **Fine-tuned flake surface matching algorithm:**

  The core contribution of this thesis, this algorithm addresses the unique challenges posed by the irregular shapes and smooth surfaces of flake

surfaces of the stone flake. It ensures high accuracy in aligning and matching flake surfaces, forming the basis for successful reassembly.

- **Interactive reassembly system:** An interactive system that integrates the fine-tuned matching algorithm, providing a user-friendly interface for archaeologists. The system allows controlling and visualizing the reassembly process. Notably, the system includes a tool for controlling the tunning of the matching algorithm, which is critical for managing the execution time of the reassembly process. This feature enable users to adjust the precision and speed of the matching and candidate ordering process, enhancing the efficiency of reassembling stone groups. User control over the tunning parameter of the flake surface matching is pivotal for developing the interactive system, as it allows for a balance between accuracy and computation efficiency.

The experiments conducted demonstrate the effectiveness and efficiency of the proposed methods and the system. The interactive system, leveraging the fine-tuned flake surface matching algorithm, successfully reassembled two groups of stones. It highlights practical application for archaeological research.

## 8.2 Future work

This thesis presents a solution for the reassembly of stone tools. However, for the future research, several improvements are identified:

- **Enhanced matching algorithm:** Future work will focus on further refining the matching algorithm to improve its accuracy and robustness, particularly in cases involving highly irregular and complex flake surfaces.

- **Expanded dataset:** Testing the system's applicability with diverse datasets of stone tool will be crucial. This will help validate the system's versatility and effectiveness across different types of stone tools.

- **User experience evaluation:** Conducting comprehensive user experience evaluations with professional archaeologists to gather feedback and identify areas for improvements. This will ensure the system meets the practical needs and expectations of users.

By addressing these future improvements, the research aims to further advance the field of computational archaeology. It provides more useful tools and techniques for the archaeological study.

# Bibliography


[1] K. Matsufuji, S. Monta, Yoku wakaru koukogaku (Understand archeology), Minerva Shobo, Kyoto, 2010, pp.18.

[2] K. Suzuki, Koukogaku Nyuumon (Archaelogy Introduction), Minerva Shobo, Kyoto, 2010, pp.18.

[3] E. Altantsetseg, Y. Muraki, F. Chiba, K. Konno, 3D Surface Reconstruction of Stone Tools by Using Four-Directional Measurement Machine, International Journal of Virtual Reality, Vol. 0, No.1, pp.37-43, 2011.

[4] Q. Huang, S. Flory, N. Gelfand, M. Hofer, Reassembling Fractured Objects by Geometric Matching, ACM Transactions on Graphics(TOG), 2006, vol. 25, pp.569.

[5] B. Bronn, C. Toler-Franklin, A System for High-Volume Acquisition and Matching of Fresco Fragments: Reassembling Theran Wall Paintings, ACM Transactions on Graphics(TOG), vol.27, pp.84, 2008.

[6] A. Wills, Stochastic 3D Geometric Models for Classification, Deformation, and Estimation. Ph.D. thesis, Brown Univ. Press., 2004.

[7] S.Erdenebayar, K.Murakami, and K.Konno, A Method of Recognizing Flake Surfaces for Noisy Point Cloud of Measuring Stone Tools, NICOGRAPH International 2020, IEEE CPS, pp.1-6, 5th, Jun, 2020.

[8] Mark Pauly, Markus Gross, Leif P.Kobbelt, Efficient Simplication of Point-Sampled Surfaces, IEEE Visualization, pp.163-170, 2002.

[9] Mark Pauly, Richard Keiser, Markus Gross, Multi-scale Feature Extraction on Point-Sampled Surfaces, Eurographics, Vol.22, No.3, 2003.

[10] Liu Ran, Wan Wanggen, Zhou Yiyuan, Lu Libing, Zhang Ximin, Normal Estimation algorithm for point cloud using kd-tree, International Conference on Smart and Sustainable City (ICSSC), pp.334–337, 2013.

[11] Julia Sanchez, Florence Denis, David Coerurjolly, Florent Dupont, Robust normal vector estimation in 3D point clouds through iterative principal component analysis, ISPRS Journal of Photogrammetry and Remote Sensing, Vol.163, pp.18-35, 2020.

[12] Taisuke Hashimoto, Masaki Saito, Normal Estimation for Accurate 3D Mesh Reconstruction with Point Cloud Model Incorporating Spatial Structure, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp.54-63, 2019.

[13] A.Bienert, C.Hess, H.-G. Maas, G. von Oheimb, A Voxel-based technique to estimate the volume of trees from terrestrial laser scanner data, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-5, 2014

[14] F.Chiba, S.Yokoyama, New Method to Generate Excavation Charts by Openness Operators, 22nd Internationl Symposium CIPA2009, 2009.

[15] P. Besl, N. McKay, A method for Registration of 3-d Shapes, IEEE Trans Pattern Anal Mach Intel, vol.14, no.2, pp.239–256, 1992

[16] N. Mellodo, N. Mitra, D. Aiger, Super 4PCS Fast Global Pointcloud Registration via Smart Indexing, Computer Graphics Forum, vol.33, no.5, pp.205–2015, 2014.

[17] R.B. Rusu, N. Blodow, M. Beetz, Fast Point Feature Histograms (FPFH) for 3D registration, 2009 IEEE International Conference on Robotics and Automation, pp.3212–3217, 2009.

[18] M.A. Fischler, R.C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Readings in Computer Vision, pp.726–740, 1987.

[19] X. Yang, K. Matsuyama, K. Konno, A New Method of Refitting Mixture Lithic Materials by Geometric Matching of Flake Surfaces, The Journal of Art and Science, Vol.15, No.4, pp.167-176, 2016.

[20] Q. Huang, S. Flory, N. Gelfand, M. Hofer, Reassembling Fractured Objects by Geometric Matching, ACM Transactions on Graphics(TOG), vol. 25, pp.569, 2006.

[21] B. Bronn, C. Toler-Franklin, A System for High-Volume Acquisition and Matching of Fresco Fragments: Reassembling Theran Wall Paintings, ACM Transactions on Graphics(TOG), vol.27, pp.84, 2008.

[22] A. Wills, Stochastic 3D Geometric Models for Classification, Deformation, and Estimation. Ph.D. thesis, Brown Univ. Press., 2004.

[23] J. H. Hong, S. J. Yoo, M. A. Zeeshan, Y. M. Kim and J. Kim, Structure-from-Sherds: Incremental 3D Reassembly of Axially Symmetric Pots from Unordered and Mixed Fragment Collections, 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, pp. 5423-5431, 2021.

[24] R. Gregor, D. Bauer, I. Sipiran, P. Perakis, T. Schreck, Automatic 3D Object Fracturing for Evaluation of Partial Retrieval and Object Restoration Tasks - Benchmark and Application to 3D Cultural Heritage Data, 2015.

[25] N. Lamb, S. Banerjee, N. Banerjee, Automated reconstruction of smoothly joining 3D printed restorations to fix broken objects, In Proceedings of the 3rd Annual ACM Symposium on Computational Fabrication (SCF '19), Association for Computing Machinery, New York, NY, USA, 2019, Article 3, 1–12.

[26] N. Lamb, C. Palmer, B. Molloy, S. Banerjee, N. Banerjee, Fantastic Breaks: A Dataset of Paired 3D Scans of Real-World Broken Objects and Their Complete Counterparts, 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 2023 pp. 4681-4691.

[27] M. Zhang, H. You, P. Kadam, S. Liu, C. Kuo, PointHop: An Explainable Machine Learning Method for Point Cloud Classification, IEEE Transactions on Multimedia, pp. 1-1, 2019.

[28] Point Cloud Library, Estimating Surface Normals in a Point Cloud.

[29] R. Osade, T. Funkhouser, B. Chazelle, D. Dobkin, Matching 3D models with shape distributions, Shape Modeling and Applications, SMI 2001 Internationl Conference, pp.154-166, 2001.

[30] K. Pearson, On lines and planes of closet fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal Of Science, 2(11), pp. 559-572, 1901.

[31] R. A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, Information Processing Letters, vol. 2, pp. 18-21, 1973.

[32] H. Edelsbrunner, D. Kirkpatrick, R. Seidel, On The Shape of a Set of Points in The Plane, Information Theory, IEEE Transactions on Information Theory, vol.29, no.4, pp. 551- 559, 1983

[33] A. Igarashi, Hakuhen hakuri genri: Seisei no zengo kankei (Flakes peeling pricinple: The context of generation), Sekki zukuri no ziken koukogaku (Exprimental archeology of the Stone tool Making), Lithic Technology Research Society, Gakuseisha, Tokyo, JP, pp.22-35, 2004.

[34] T. Lin, X. Yang, K. Konno, A Method of Searching Lithic Cores by Average Linkage Clustering, NICOGRAPH International 2018, in press.

[35] X. Yang, K. Matsuyama, K. Konno, Pairwise Matching of Stone Tools Based on Flake-Surface Contour Points and Normals, Eurographics Workshop on Graphics and Cultural Heritage, 2017.

[36] T. Lin, X. Yang, F. Chiba, K. Konno, An Interactive Reassembly Method for Stone Tool Restoration, Nicograph 2018, 2018.

[37] K. Yamahara, K. Konno, F. Chiba, M. Satoh, A Method of Detecting Adjacent Flakes in Stone Tool Restoration by Extracting Peeling Surfaces, Japan Society for Archaeological Information., Vol.17, No.1-2, pp.23-31, 2011.

[38] Inter i7-10700 vs. Intel i7-4790, CPU Benchmark. Accessed on: Mar.17, 2024, Available: https://www.cpubenchmark.net/compare/3747vs2226/Intel-i7-10700-vs-Intel-i7-4790.

# List of Publications

- A. Altansukh, M. You, E. Altantsetseg, O. Khorloo, K. Konno, A Study on Automatic Flake Surface Segmentation of Stone Tools by Calculating Shape Features, IWAIT, Vol. 12592, pp. 123-128. SPIE, 2023.
- A. Altansukh, M. You, E. Altantsetseg, O. Khorloo, F. Chiba, K. Konno, A New Matching Algorithm for Stone Tool Reassembly Based on Contour Points of Flake Surface, The Journal of the Society for Art and Science, Vol. 23, No. 2, pp. 4:1-4:17, 2024.