Structure-from-Sherds: Incremental 3D Reassembly of Axially Symmetric Pots from Unordered and Mixed Fragment Collections

Je Hyeong Hong*†
KIST, Hanyang University
jhh37@hanyang.ac.kr

Seong Jong Yoo*
KIST
yoosj@kist.re.kr

Muhammad Zeeshan Arshad
KIST
zeeshan@kist.re.kr

Young Min Kim
Seoul National University
youngmin.kim@snu.ac.kr

Jinwook Kim†
KIST
zinook@kist.re.kr

Abstract

Re-assembling multiple pots accurately from numerous 3D scanned fragments remains a challenging task to this date. Previous methods extract all potential matching pairs of pot sherds and considers them simultaneously to search for an optimal global pot configuration. In this work, we empirically show such global approach greatly suffers from false positive matches between sherds inflicted by indistinctive sharp fracture surfaces in pot fragments. To mitigate this problem, we take inspirations from the field of structure-from-motion (SfM), where many pipelines have matured in reconstructing a 3D scene from multiple images. Motivated by the success of the incremental approach in robust SfM, we present an efficient reassembly method for axially symmetric pots based on iterative registration of one sherd at a time. Our method goes beyond replicating incremental SfM and addresses indistinguishable false matches by embracing beam search to explore multitudes of registration possibilities. Additionally, we utilize multiple roots in each step to allow simultaneous reassembly of multiple pots. The proposed approach shows above 80% reassembly accuracy on a dataset of real 80 fragments mixed from 5 pots, pushing the state-of-the-art and paving the way towards the goal of large-scale pot reassembly. Our code and preprocessed data is available at https://github.com/SeongJong-Yoo/structure-from-sherds.

1. Introduction

Ceramic pots are fundamental cultural relics in human history. Most of them are made on a spinning wheel, encompassing axial symmetry. While they provide meaningful information to understanding ancestral lifestyles, many pots are excavated as broken pieces (known as sherds), necessitating accurate reassembly from these sherds for the purpose of recovering historical records.

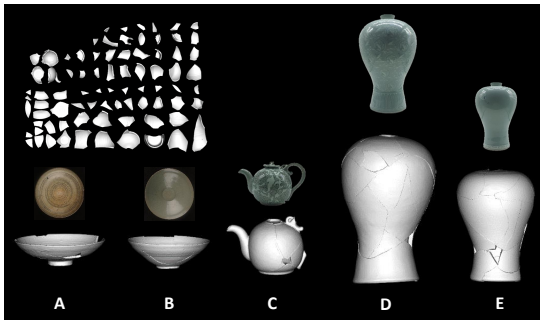


Figure 1. An illustration of 5 real pots simultaneously reassembled from 80 unclustered sherds (3D point clouds) using our pipeline. It can handle some degree of non-axially symmetric decorations on the pot surface as shown in pot C. Ground truth (manually restored) pot images are shown above each reassembly result.

Currently, the process of reassembly is relied upon extensive manual efforts from restoration experts. Despite years of experience, however, reconstructing each pot can often require hours of concentration even for the qualified professionals as fragments are mixed from multiple pots. Furthermore, pieces are fitted on a trial-and-error basis, potentially incurring undesired abrasions on the break surfaces of sherds. These motivate strong need for an efficient framework that can virtually reassemble axially symmetric pots.

Recent studies have proposed pipelines [18, 13] majorly comprising 3 stages, first extracting features from each pot sherd, second computing potential matches between each pair of sherds, and last performing global combinatorial optimization to deduce the most likely 3D pot configuration from these matches. These are known as *global* approaches since the last stage can be viewed as finding true positive sherd matches from the global pot configuration graph. Nevertheless, our effort of implementing such pipeline raises an inherent issue illustrated in Fig. 2, that a

*Both authors contributed equally to this work.
†Co-corresponding authors

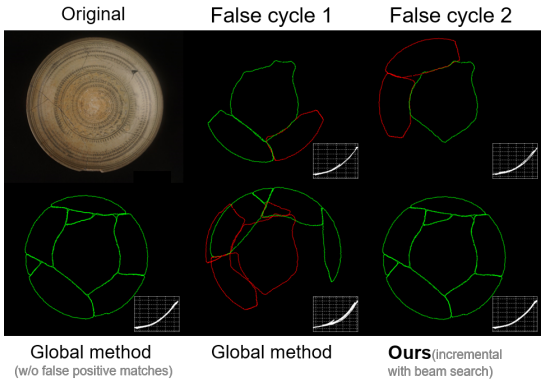


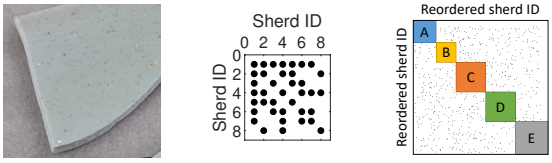
Figure 2. The top row shows some exemplary false positive loops formed in pot A from Fig 1. (True configurations in green and false in red.) Some pairwise matches are difficult to detect even at the cycle level just by looking at the break lines and the axis profile curve (in white). The global method only succeeds when false positive matches are removed. Our incremental method with beam search outputs the correct result in spite of false matches.

global method is sensitive to false positive matches between pot sherds as it directly derives each sherd's configuration from these matches. Unfortunately, such false positives are widespread due to lack of distinctive features on the fracture surfaces as the result of sharply broken edges (see Fig. 3).

In addressing above issue, we turn our attention to structure-from-motion (SfM), which have enjoyed tremendous success over the last few decades in reconstructing 3D scenes from thousands of images [17, 1, 22, 16]. Behind this achievement lies the *incremental* method, which mains robustness to numerous false positive matches by iteratively registering one camera view at a time in descending order of connectivity. The essence of the incremental method lies in its ability to *iteratively improve pairwise matches and discover initially undetected (false negative) matches* as the model grows, which can be tricky for the global approaches that relies on pruning initial pairwise matches only.

A naturally following question is, can we apply the findings from incremental SfM to devise a reliable method for reassembling axially symmetric pots from fragments? As will be illustrated in Sec. 2.2, we find both tasks share a great deal of analogy in terms of the problem and (global) pipeline structures, potentially allowing a smooth transfer of the incremental method from the domain of SfM to pot reassembly. Yet, pot reassembly requires handling additional challenges such as mitigating incorrect sherd registration triggered by many indistinguishable false positive sherd matches (due to ambiguous geometric features) and enabling reassembly of multiple pot models.

To this end, we propose *structure-from-sherds* (SfS), an efficient incremental method for reassembling multiple pots from fragments. Our method follows the iterative sherd registration scheme to grow the reassembly model like in incremental SfM, but some extensions are without which the



(a) (b) (c)

Figure 3. Illustration of some challenges in reassembling multiple axially symmetric pots. (a) shows sharp, thin fracture surfaces with little distinct features triggering false positive sherd matches. (b) shows the sparse nature of connectivity between sherds from the same pot (59% filled across 8 sherds of pot A in Table 2). (c) illustrates the noisy block-diagonal sparsity structure of the problem when sherds are mixed from different fragments (See Sec. 2.1).

reassembly performance is limited (see Table 3). These include adopting beam search to explore multiple registration possibilities in each step to compensate for indistinguishable false matches, and employing multiple sherd roots in each beam state to reassemble multiple pots simultaneously.

Our main technical contributions are as follows:

+ a **new pot reassembly pipeline** derived from our unification of structure-from-motion and pot reassembly (Sec. 2.2) and based on multi-root beam search, yielding more complete pots from more fragments [18],
+ a **new axis-based geometric descriptor** for the break line points, removing a need to use surface point cloud for efficient initial sherd matching(Sec. 3.4)
+ **simple algorithms** for extracting and utilizing rim, base and thickness to reduce false matches and further improve registration accuracy (Table 3), and
+ a **new challenging dataset** of 80 real pot sherds from 5 different pots made for evaluation (Table 2).

Conversely, our implementation has some limitations: currently, it requires at least part of the base fragment for each pot to be correctly reassembled (see Sec. 5.4). Also, we do not utilize fragment texture (like in [18]) because the 3D scanner often fails in stitching the texture correctly and suffers from light saturation. Nonetheless, we believe this work would serve as the basis for further improvements.

1.1. Related work

The problem of virtually reassembling archaeological pots has received consistent attention in computer vision and graphics [12, 8, 20, 18, 24] over the last two decades.

In the early days, several works showed ways to extract geometric properties of the axially symmetric pot from individual fragments such as the axis of symmetry [14, 3, 11] or the profile shape (i.e. axis profile curve) [19]. Mcbride an Kimia [12] devised a method for matching pairs of sherds, and noted this process is much more involved than solving a jigsaw puzzle due to the non-unique break lines.

It was Willis and Cooper [20] who first pioneered a reassembly pipeline for axially symmetric pots. Their method follows an incremental registration procedure like ours,

	Structure-from-motion (SfM)	Pot reassembly
Input	RGB images	3D point cloud of pot sherds
Outputs	camera poses & sparse 3D scene	sherd poses & axially symmetric pot models
Extracted features	SIFT [10] keypoints	axis, edge line descriptor, rim, thickness, base
Pairwise matching criteria	SIFT feature distance	Weighted sum of above features
Geometric verification	inliers from fundamental (or essential) matrix or homography estimation	inliers from iterative closest point (ICP)
Sanity check	cheirality constraint	overlap, thickness & profile curve constraints
Registered quantity	camera views	sherds
Triangulated model	3D scene points	3D pot model (axis of symmetry & profile curve)
Joint estimation	bundle adjustment	global sherd and axis alignment via ICP

Table 1. A list of analogies formed between an incremental structure-from-motion pipeline [16] and our pot reassembly method.

adding each fragment one by one to gradually grow the global consensus of the outputted model. In each step, the best candidate sherd is chosen by considering its matching degree of break line points, the corresponding surface normals and the axis profile curve. Nevertheless, it requires manual effort to extract features known as the T-junctions on the break lines, and has only been tested on a single pot case of 10 sherds. Furthermore, there is no backtracking procedure that avoids failures from incorrect registration.

In later years, Son et al. [18] presented a type of global approach to tackling the reassembly problem with 48 fragments from 3 pots. The method considers all potential pairwise matches across all pot sherds jointly and solves a combinatorial optimization problem of finding true positive fragment pairs to directly retrieve the 3D pot model. This is carried out by minimizing algebraic costs promoting consistency of the axis of symmetry and the axis profile curve using a spectral method [9]. While the results have demonstrated state-of-the-art performance, it is unclear as to how the false positive matches are handled—they are sometimes indistinguishable from ground truth [20], which can be detrimental for global approaches as shown in Fig. 2.

Other studies include the work of Huang et al. [8], which proposes an incremental method for reassembling geometric objects, but this relies on existence of rich unique features on the break surface which is often not visible in pot sherds (see Fig. 3a). Also, the method has only been tested for reassembling single objects. Zhang et al. [24] presented a template-based matching technique, but this is also limited to single object cases with known pot templates.

2. Devising a robust reassembly pipeline

We now review the challenging aspects of axially symmetric pot reassembly, compare this problem to well-established SfM to gain intuitions for a successful methodology and propose our pipeline for the pot reassembly task.

2.1. Review of problem characteristics

Some main problem characteristics exhibited in reassembling axially symmetric pots are summarized below:

Existence of numerous false positive matches Fig. 3 shows the fracture surface (often used for matching [8, 24]) is thin and sharply broken for ceramic pots. It is therefore difficult to detect features and distinguish between break lines, raising the number of false positive matches.

Sparse intra-pot connectivity For pot reassembly, each sherd pair must be physically adjacent, resulting in a sparse connectivity graph between sherds. This is shown in Fig. 3, where pot A (Fig. 2) contains only 59% filled graph.

Sparse but noisy inter-pot connectivity For multiple pot reassembly, fragments end up in different pots, yielding a block-diagonal pot configuration graph plagued by false positive matches as shown in Fig. 3. This further raises the problem difficulty, and subsequently a reassembly pipeline needs to handle noisy but originally-disconnected graphs. (otherwise one may recover only one pot at best.)

2.2. Comparing axially symmetric pot reassembly with structure-from-motion

We now present an analogy between the task of pot reassembly and structure-from-motion (SfM) that allows us to intuitively derive a similar pipeline for pot reassembly.

Problem analogy Structure-from-motion (SfM) is a multi-stage pipeline jointly estimating 3D scene points and camera poses from a set of input images. We can also think of pot reassembly as jointly estimating a 3D pot model (defined by the axis of symmetry and the axis profile curve) and sherd transformations from 3D point cloud data of sherds. In this regards, both tasks amount to solving an optimization problem over a sparsely connected graph (see Fig. 3).

Procedural similarities There are noticeable similarities between the SfM pipeline and pot reassembly methods.

Regarding the global methods, global SfM proceeds by first extracting features from individual images, second searching for pairwise matches between images, and last pruning these matches via triplet or loop filtering [23] to yield a global model of camera poses and 3D points (involving rotation averaging [4] and translation averaging [21] followed by bundle adjustment [5]). Similarly, Son et al.'s approach [18], which is a type of global pot reassembly method, comprises similar steps, extracting axis-based and

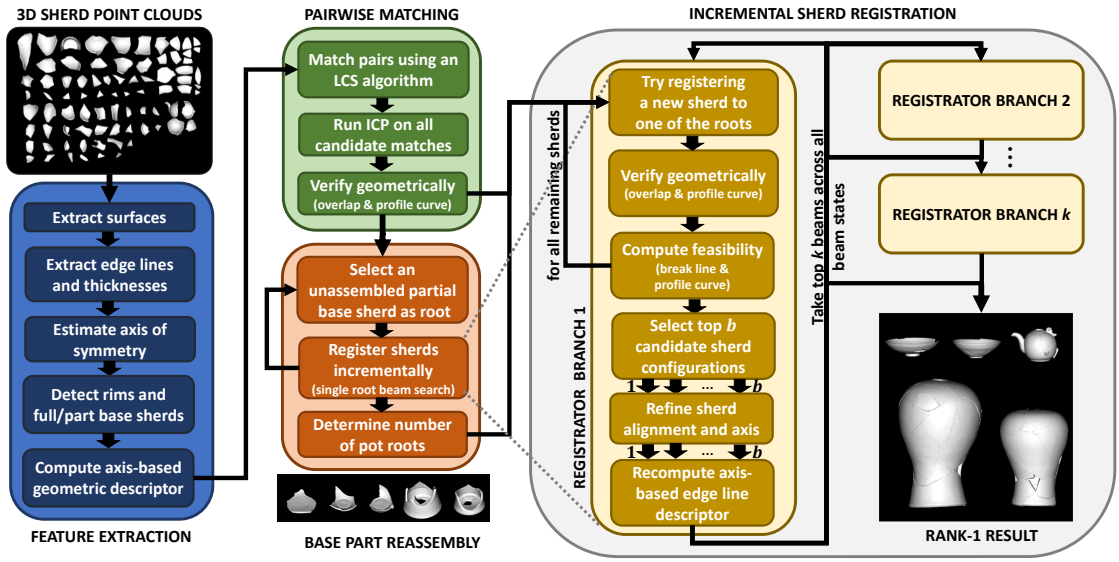


Figure 4. A detailed overview of our pipeline. Multiple arrows $(1, \ldots, b)$ in registrator branch 1 represent multiple branches formed. The grey dotted lines indicate the incremental registration step during base part reassembly has a similar structure to registrator branch 1.

break line-based features for each sherd, searching for pairwise matches between sherds and last solving a graph optimization problem (via spectral method) to find a global pot model (defined by the axis of symmetry and axis profile curve) and respective sherd transformations.

Analogical connections in procedures From the procedural similarities outlined above, we can yield a full list of analogies (see Table 1) formed between SfM and axially symmetric pot reassembly. This enables us to effectively import the architecture of the well-established incremental SfM pipeline to reassembling axially symmetric pots.

Empirical differences As shown in Fig. 2, we have empirically found false positive pairwise matches are highly problematic in pot reassembly as some are almost indistinguishable from the true positives even from the human eye perspective (even after applying false cycle filtering as in [23]). Also, pot reassembly needs to build multiple disconnected models whereas SfM often builds a single scene.

2.3. An overview of the proposed pipeline

Our proposed pipeline for reassembling axially symmetric pots is founded upon the findings from Sec. 2.2.

As shown in Fig. 4, our pipeline is modified and extended from a standard incremental pipeline to achieve more accurate reassembly results in spite of large number of false matches and existence of multiple disconnected pots.

First, the method incorporates beam search during the incremental sherd registration phase, exploring many more paths of possibilites in each step thereby increasing the chance of arriving at the correct solution.

Second, the pipeline efficiently reassembles the base regions first to determine the number of pots to be reconstructed as well as yield a good initial starting point. (A base is the flat region at the bottom of each pot.)

Each pipeline stage is illustrated in the next sections.

3. Feature extraction from individual sherds

We extract features from each sherd, including the inner and outer surfaces and edge line, estimating the axis of symmetry and detecting base and rim parts in each fragment.

3.1. Extracting surfaces and edge lines

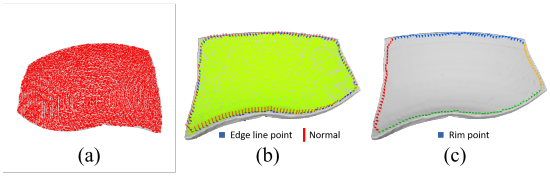
We first extract two surfaces corresponding to inner and outer surfaces. (we later use the axis of symmetry to classify between the two surfaces [7].) Then for each of the surfaces, we extract an ordered line of points around the edge of each surface which we defined as an *edge line*. This is essentially a vector of ordered (counter-clockwise) 3D points and respective surface normals on the surface boundary. Rather than storing the actual surface normal of each edge point (which can be noisy due to the fracture surface), we fit a B-spline curve to the surface and compute the estimated normal of the edge point from the curve. This vector is further segmented to different parts, which is later used for detecting rim and break lines (see Fig. 5.)

Additionally, we extract thickness information (t in Fig. 7). Details can be found in our supplementary URL [7].

3.2. Estimating the axis of symmetry

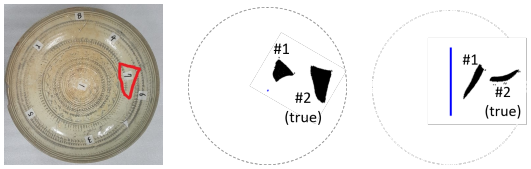
From the analogy in Table 1, estimating the axis of symmetry can be seen as inferring the underlying pot model.

Amongst the algorithms for estimating the axis of symmetry [3, 11, 18, 6], the state-of-the-art performance is achieved by PotSAC [6]. While the original PotSAC is limited to outputting a single axis even for ambiguous sherds,



(a) (b) (c)

Figure 5. Features extracted in Sec. 3.1. Yellow and red denote inner and outer surfaces. Each edge line along the boundary of the surface contains normals computed from its nearest surface point.



(a) Original image (b) Top view (c) Side view

Figure 6. Axis estimation results using modified PotSAC on sherd #7 from Pot A. Original PotSAC outputs incorrect #1 for the axis.

we present a slight modification that can output multiple axes which allows us to avoid failures for these cases.

The basic idea is simple: instead of just taking the winner axis from RANSAC, we take top-10 candidates, refine them as in ordinary PotSAC and prune them to yield a set of distinctive axes candidates. For the pruning purpose, we check the angle made between each pair of axes—if this angle is below 10° for any pair, then those pairs are assumed redundant, and thus the one with higher cost is discarded. To make the algorithm efficient, we use 10% subsampled surface points for the above steps, after which we run refinement on the survived axes with all surface points.

Fig. 6 shows the modified algorithm outputting 2 axes of which #2 is correct, in contrast to the original PotSAC only yielding the (incorrect) #1 axis. This has allowed complete reassembly of pot A as shown in Fig. 2 at the expense of increased number of pairwise sherd matches from two axes.

3.3. Base sherd and rim detection

The axis of symmetry provides a mean to efficient detect sherds with the base region and the rims.

Base sherd detection We use two sets of information to determine that a sherd contains the base. First, we check the proximity of the sherd with respect to the axis of symmetry. If more than 50 surface points *with* surface normals aligned within 10° from the axis of symmetry are found within 20mm radially from the axis, the sherd is considered a base fragment. Second, we check if there is any sudden change in the sherd's profile curve along the z-axis, which is common for the type of Celadon pots. We form bins every 5mm along the z-axis, and for each bin we compute the mean of the angles between the constituent surface points' normal vectors and the axis of symmetry. If any adjacent bins yield more than 30° difference in the mean surface normal angle, the sherd is also considered a base fragment.

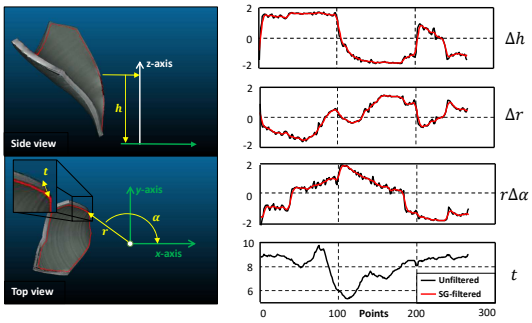


Figure 7. An illustration of our axis-based edge line descriptor. Unfiltered results are computed via finite difference method. The Savitzky-Golay filter is used for smoothing and differentiation followed by a Gaussian filter of width 7 and σ=2.0. Thickness (t) is not filtered as this is sometimes missing in some edge points.

Finally, we align the surface data by the axis of symmetry and project the points onto the xy-plane. We then plot a angular histogram of points that are within 25mm from the axis. If the histogram is evenly spread out, then we regard the sherd as containing the full base. Otherwise, the sherd is flagged as a partial base fragment.

Rim detection For each edge line segment (>25 points) obtained from Sec. 3.1, we check for its consistency in the radial and z-directions— if the radial and vertical standard deviations are less than 3.0mm and 1.5mm respectively, the edge line segment is a rim candidate. We then check if any nearby points along the z-axis have greater radius, which implies the rim is incorrectly segmented. In such case, we drop the edge line from consideration.

3.4. Axis-based edge line descriptor

While previous studies have utilized the axis of symmetry as a physical constraint for incorrect pairwise matching [18], we make one step further and actually encode the information about the axis into each break line (part of the edge line) for efficient matching.

Motivation We note a set of matching break lines from neighboring sherds should ideally meet at the same position from the axis of symmetry. From this observation, we utilize the height (h), radius (r) and angle (α) with respect to the axis of symmetry coordinate system (with the x-axis chosen arbitrarily) as means of providing a description for each of the edge line points (see Fig. 7). Additionally, we incorporate the thickness (t) obtained from Sec. 3.

Need for the derivatives There are two issues with directly using above metrics for matching. First, h and α are computed with respect to some reference defined in each fragment's coordinate system which is almost always different from each other. Second, the absolute value of r can be unreliable for small fragments with high uncertainty in axis translation (see [7] for an illustration). For these reasons, we utilize their derivatives except for t.

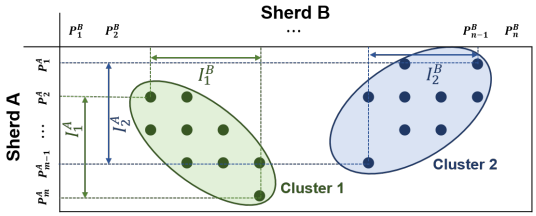


Figure 8. Our LCS matching scheme detects multiple intervals of potential matches between two sherds (<10s for 3160 pairs)

Edge line smoothing Due to noise along the edge lines, simply applying a naive differention technique such as finite differences results in noisy features as shown in Fig. 7. Hence, we apply the Savitzky-Golay digitial differentiator (S-G method) [15] using 7 points, wrapped with a Gaussian filter of width 7 and σ=2.0 to obtain smooth edges.

Descriptor definition Finally, our axis-based descriptor for the edge line point $\mathbf{p}_j \in \mathbb{R}^3$ are defined as

$$\mathbf{f}(\mathbf{p}_j) := [\Delta h(\mathbf{p}_j), \Delta r(\mathbf{p}_j), r\Delta\alpha(\mathbf{p}_j), t(\mathbf{p}_j)]^\top, \quad (1)$$

where Δ involves finite differentiation. (Note r is multiplied to $\Delta\alpha$ to get geometric tangential distance.)

Above can encode all geometric changes regarding each edge line, and it has the benefit of not having to refer to the original point cloud (like in [18]) when checking the axis of symmetry-derived constraints are satisfied.

4. Pairwise matching

Once the geometric edge line descriptors are extracted, we search for pairwise matches across all edge lines in 2 steps: descriptor matching followed by match refinement.

4.1. Descriptor matching

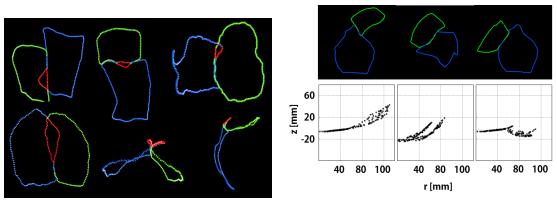
We first match pairs of edge lines by using the longest common subsequence (LCS) algorithm (see [7]). If we denote the matched pair of sherds as A and B respectively, running the algorithm outputs several clusters (known as intervals) of potential matching regions (see Fig. 8). We define this interval as I_k^{AB}, where k is the interval index.

Addressing ambiguities There are 2 folds of ambiguities triggered by the sign ambiguity of the symmetric axis. To account for this, we match each pair of edge lines twice by inverting the descriptor of one edge line.

Rim constraint If the matched interval contains a rim segment, we determine the match is false and discard it.

4.2. Refining matches

While the above descriptor provides useful cues for finding initial correspondences, they are inevitably prone to errors from axis estimation. To refine individual matches, we run iterative closest point (ICP) on the initial pairs of edge lines. The ICP algorithm comprises two stages: correspondence formation followed by correspondence minimization. These steps are repeated until function tolerance is reached.



(a) 3D break line overlap test (b) Axis profile curve check

Figure 9. Geometric verification filters 2 cases (a) physically unrealistic overlaps between sherds and (b) non-smooth profile curves.

Forming correspondences In the very first iteration, correspondences are acquired from descriptor matching, and for later iterations, each pair is formed when both points are the closest to (and within 20mm from) each other (i.e. a 1-to-1 relationship) and their normals are within 30°.

Minimizing correspondence distances In the second step, we iteratively find the transformation matrices of the sherds which minimize the geometric distance between the pairs of correspondences while promoting rim segments to maintain consistent radius r and height z.

We define the transformation of sherd A as $\mathtt{T}^A$, which consists of rotation $\mathtt{R}^A \in SO(3)$ and translation $\mathbf{t}^A \in \mathbb{R}^3$ If use the notation (i, j) to denote the correspondence between the i-th edge line point of sherd A ($\mathbf{p}_i^A \in \mathbb{R}^3$) and the j-th edge line point of sherd B ($\mathbf{p}_j^B \in \mathbb{R}^3$), the correspondence distance term can be expressed as $d_{ij}(\mathtt{T}^A, \mathtt{T}^B) := d(\mathbf{p}_i^A, \hat{\mathbf{n}}_i^A, \mathbf{p}_j^B, \hat{\mathbf{n}}_j^B, \mathtt{T}^A, \mathtt{T}^B)$ the correspondence normal deviation term as $e_{ij}(\mathtt{T}^A, \mathtt{T}^B) := e(\hat{\mathbf{n}}_i^A, \hat{\mathbf{n}}_j^B, \mathtt{T}^A, \mathtt{T}^B)$ and the rim consistency term as $g_i(\mathtt{T}^A, r, h) := g(\mathbf{p}_i^A, \mathtt{T}^A, r, h)$. We essentially solve

$$\min_{\mathtt{T}^A, \mathtt{T}^B, r, h} \sum_{(i,j)\in\Omega_{AB}} \rho_d(d_{ij}^2(\mathtt{T}^A, \mathtt{T}^B)) + \lambda\rho_e(e_{ij}^2(\mathtt{T}^A, \mathtt{T}^B)) + \nu \sum_{E\in A,B} \sum_{i\in\Psi_E} \rho_g(g_i^2(\mathtt{T}^E, r, h)) \quad (2)$$

where ρ_d, ρ_e and ρ_g are robust kernels to suppress outlier correspondences, Ω_{AB} is the set of edge line correspondences between sherds A and B, Ψ_E is the set of edge line points classified as rim in sherd E (A or B). While we solve (2) using the Levenberg-Marquardt (LM) algorithm [2], we use different correspondence distance metrics during optimization for which full details are provided in [7].

4.3. Geometric verification

Above ICP algorithm is deliberately unconstrained to yield better empirical convergence properties. Unfortunately, this leads to many false positive matches, requiring further pruning via geometric verification. Similar to [18], we use two tests, detecting an overlapped region between the pair of sherds and checking the axis profile curve. Some examples are shown in Fig. 9 (see [7] for implementation).

5. Incremental sherd registration

We now illustrate the sherd registration module in Fig. 4. We will assume one or more sherds are reassembled in advance, forming a set-of-sherds $\{C\}$.

5.1. Sherd registration

When registering a new sherd (denoted as sherd D) to the reassembled sherds $\{C\}$, we align the coordinates by the sherd C's axis of symmetry (defined as the z-axis without loss of generality) and keep it fixed. We then adjust sherd D's pose ($\mathtt{T}^D$) to align sherd D to sherd C (or sherds $\{C\}$) considering i) the closeness of the edge line matches between sherd D and sherds $\{C\}$ in terms of 3D points and their surface normals, ii) the consistency of the rim in terms of radius r and height z, and iii) the consistency of the symmetric-axis between $\{C\}$ and D. Noting the notations from (2), we essentially solve

$$\min_{\mathtt{T}^D} \sum_{E\in\{C\}} \sum_{(i,j)\in\Omega_{DE}} \rho_d(d_{ij}^2(\mathtt{T}^D,\mathtt{T}^E)) + \lambda\rho_e(e_{ij}^2(\mathtt{T}^D,\mathtt{T}^E)) + \mu \sum_{i\in\Omega_D} \rho_f(f_i^2(\mathtt{T}^D)) + \nu \sum_{i\in\Psi_D} \rho_g(g_i^2(\mathtt{T}^D,r,h)) \quad (3)$$

where Ω_D is the set of edge line points in sherd D, Ψ_D is a subset of Ω_D classified as rim, and f_i is a measure of the axial consistency in sherd D. Specifically, f_i is the extended Cao and Mumford's objective proposed in eq. (9) of [6], which is essentially a function of the edge line point $\mathbf{p}_i^D$, its normal $\hat{\mathbf{n}}_i^D$ and the axis of symmetry (represented by the axis direction $\hat{\mathbf{v}}$ and the axis offset $\mathbf{u}$ in [6].) Unlike in [6] where the objective is used to adjust the axis of symmetry given fixed surface points and normals, we transform each edge line point and its normal as $\mathtt{R}^D\mathbf{p}_i^D + \mathbf{t}^D$ and $\mathtt{R}^D\hat{\mathbf{n}}_i^D$ respectively under fixed symmetric axis ($\hat{\mathbf{v}} = [0,0,1]^\top$, $\mathbf{u} = \mathbf{0}$). We again utilize the ICP algorithm (i.e. the correspondences are updated every iteration) as in Sec. 4.2, and implementation details are provided in [7].

5.2. Priority list

Above registration step yields a set of pairwise matches between the break lines from 2 or more sherds. We then calculate each pair's score based on the number of inliers from which we decide the ranking of pairs.

Adjusted number of inliers From these, we can calculate the number of inlier correspondences (P), which is defined as pairs with less than 1.5 mm. This number is then multiplied by the factor of proportion of overlapping axis profile curve (Q) to yield $(1+Q)P$ as the adjusted number of inliers. For instance, if a sherd C and sherd D have 100 inlier matches, and they share 70% of the overlap in the profile curve, then the adjusted number of inliers is 170.

5.3. Batch sherd alignment

After a priority list ranking is decided, we further refine some of the higher ranked registration results ($k \times b$ candidates using our beam search in Sec. 5.4) by allowing previously re-assembled sherds to move as well. If we suppose now sherd D becomes part of the set of reassembled sherds $\{C\}$, then by noting notations from (2) and (3), we solve

$$\min_{\{\mathtt{T}^C\},r,h} \sum_{(E,F)\in\Omega_{\{C\}}} \sum_{(i,j)\in\Omega_{EF}} \Big(\rho_d(d_{ij}^2(\mathtt{T}^E,\mathtt{T}^F)) + \lambda\rho_e(e_{ij}^2(\mathtt{T}^E,\mathtt{T}^F))\Big) + \sum_{F\in\Omega_{\{C\}}} \Big(\mu \sum_{i\in\Omega_F} \rho_f(f_i^2(\mathtt{T}^F)) + \nu \sum_{i\in\Psi_F} \rho_g(g_i^2(\mathtt{T}^F,r,h))\Big) \quad (4)$$

where (E, F) denote a pair of sherds in set of sherd pairs $\Omega_{\{C\}}$. (Note $\mathtt{T}^E$ and $\mathtt{T}^F$ are members of the set $\{\mathtt{T}^C\}$.) As in Sec. 5.1, we utilize the ICP algorithm to update correspondences via alternation (see [7]).

5.4. Multi-root beam search

We use beam search to explore multitudes of potential reassembly paths. In each iteration, we make b branches from each of the current branches (initially set to 1) and filter k top-ranked beams to bring to the next iteration. Furthermore, to restore multiple pots from the mixed piles of sherds, we utilize multiple roots in each *state* of the beam search (see [7]). This allows efficient matching as each sherd can only be registered to one root per state.

Merging redundant permutations In each beam search iteration, some paths can end up with same reconstruction (with different reassembly order). We consistently check for these and merge such redundant beams.

Estimating initial number of pots In order to run the multi-root beam search framework in Fig. 4, we need to set the number of roots for reassembling pots (which is ideally equal to the number of pots). We achieve this by first gathering a set of sherds with the base region from Sec. 3.3, merging all broken bases and using this number with the number of full (unbroken) bases for initiating the incremental sherd registration process. The merging process involves iteratively selecting an unmerged partial base fragment as root (for incremental beam search) and reassembling all possible pieces until all partial base fragments are reassembled.

6. Experimental results

We carried out experiments to observe the performance of our pipeline under various conditions. The optimization settings and PC environment can be found in [7].

Datasets We used total of 80 fragments from 5 different pots. Pots A and B are actual ancient pots estimated to have been made between the 14–16th century, and Pots C, D and

Pot ID (# sherds)		A (8)	B (9)	C (4)	D (28)	E (31)	A+B (17)	A+B+C (21)	D+E (59)	All (80)
# base sherds (detected / actual)		1 / 1	1 / 1	1 /1	3 / 3	4 / 4	2 / 2	3 / 3	7 / 7	10 / 10
# rim sherds (detected / actual)		5 / 6	6 / 7	2 / 3	1 / 1	0 / 1	11 / 13	13 / 16	1 / 2	14 / 18
# initial pairwise matches		72	98	51	1742	1063	331	483	4715	6431
Preprocessing runtime (min.)		(3.9)	(3.4)	(10.9)	(47.8)	(37.2)	(7.3)	(18.1)	(85.0)	(103.0)
[b=3, k=5]	Accuracy	8 / 8	9 / 9	4 / 4	22 / 28	19 / 31	15 / 17	19 / 21	27 / 59	43 / 80
	Runtime (min.)	(0.2)	(0.2)	(0.1)	(22.5)	(17.0)	(1.1)	(1.9)	(82.3)	(167.5)
[b=5, k=10]	Accuracy	8 / 8	9 / 9	4 / 4	23 / 28	26 / 31	17 / 17	21 / 21	36 / 59	54 / 80
	Runtime (min.)	(0.4)	(0.4)	(0.1)	(49.7)	(32.8)	(2.8)	(3.7)	(210.2)	(256.8)
[b=10, k=20]	Accuracy	8 / 8	9 / 9	4 / 4	24 / 28	24 / 31	17 / 17	21 / 21	48 / 59	66 / 80
	Runtime (min.)	(0.9)	(1.2)	(0.3)	(127.4)	(172.6)	(4.7)	(8.2)	(440.8)	(592.4)

Table 2. Reassembly results on our data. Pot IDs are assigned in ascending order of difficulty. Data preprocessing was run on a slower PC.

[b=3, k=5]	Baseline	w/o beam search	w/o rim	w/o thickness
# initial matches	483	483	634	943
Accuracy	19/21	11/21	18/21	18/21

Table 3. Ablation study of different components proposed in this work (results from the A+B+C multi-pot reassembly setting).

E are modern pots deliberately broken for the purpose of this task. Each pot was scanned with a Creaform GoSCAN 20 at maximum resolution of 0.1mm. The output is a triangulated mesh, but we only utilized the point cloud data for our pipeline. (mesh was used for visualization only.)

We omitted some very small fragments comprising $<$50 points in the respective surface edge lines (i.e. perimeter $\approx$ 9cm), removing 1 piece in pot C, 3 in pot D and 4 in pot E.

Generating ground truth We were provided with reconstructed pot images with each piece labelled by a group of restoration experts. We used these to manually place sherds in correct locations using CloudCompare, and used this to perform ICP using the fracture surfaces between individual sherds to obtain a set of accurate reassembly models (see [7]). Then, for each pot, we recorded the relative transformation ($\mathtt{T}^E$) of each sherd E from the same pot with respect to the base fragment's coordinates.

Counting success For each reconstruction result, we obtained the transformation of each sherd relative to its constituent base fragment's coordinates and compared it to the semi-ground truth data obtained from above. The threshold values for success are 30° for 3D rotation (not the axis deviation) and 20mm for translation.

6.1. Evaluation of our approach

We tested our algorithm under various settings, namely switching between single pot and multi-pot environments and varying the number of branches (b) and beams (k).

As shown in Table 2, our method shows stable performance across all tested beam search settings for the single pot reassembly cases. For the multi-pot setting, beam search with more branches (b) and ranks (k) overall increases the number of successfully reassembled fragments as predicted, largely due to the increased number of trials allowing the pipeline yield a better solution. Interestingly, in some cases it is the opposite, and we think this is potentially due to earlier-rejected beam paths surviving for larger k and coming back later to negatively influence the result.

While a direct comparison with Son et al.'s method (31/48 sherds reassembled in 648 minutes) is difficult due to difference in datasets, we show over 17.9% improvement in reassembly accuracy with 32 more fragments.

Ablation study We also checked the performance gain brought by the individual components we proposed in this work. Table 3 shows beam search is mandatory for successful reassembly and the new set of constraints using the rim and thickness information helps to reduce the number of initial pairwise matches by 30–50%.

Note on hyperparameters While we made efforts to build a generic pipeline, some hyperparameters need to be tuned correctly. These include parameters for edge line extraction, the threshold for checking sherd overlaps and ICP weights and kernel widths (see [7] for details).

7. Conclusions

In this work, we have addressed the problem of virtually reassembling axially symmetric pots from 3D scanned fragments. We empirically showed a global reassembly pipeline is sensitive to inaccurate pairwise feature matches between pot sherds arising from ambiguous geometric cues. In addressing this intrinsic difficulty, we formed a strong connection between structure-from-motion (SfM) and pot reassembly, inspiring us to utilize incremental registration widely adopted in SfM. Additionally, we proposed several major extensions, i) an axis-based geometric feature descriptor for matching, ii) adoption of beam search to maintain a pool of registration possibilities, and iii) multiple root nodes in each beam state to reduce the number of false matches in reassembling multiple pots simultaneously. We also showed rim and thickness information help in reducing false matches. Through experiments on a larger number of pot fragments and pot types than previously reported in the literature, we showed our method achieves significant improvement in reassembly accuracy over the state-of-the-art.

Acknowledgement This work was supported by South Korea's Ministry of Culture, Sports and Tourism (MCST) and the Korea Creative Content Agency (KOCCA) through the Culture Technology (CT) Research & Development Program (R2018020101).

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, 2011. 2

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org. 6

[3] Yan Cao and David Mumford. Geometric structure estimation of axially symmetric pots from small fragments. In *Proceedings of Signal Processing, Pattern Recognition, and Applications*, 2002. 2, 4

[4] R. Hartley, J. Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International Journal of Computer Vision*, 103:267–305, 2012. 3

[5] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. 3

[6] Je Hyeong Hong, Young Min Kim, Koang-Chul Wi, and Jinwook Kim. Potsac: A robust axis estimator for axially symmetric pot fragments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019. 4, 7

[7] Je Hyeong Hong*, Seong Jong Yoo*, Muhammad Zeeshan Arshard, Young Min Kim, and Jinwook Kim. Supplementary document for structure-from-sherds: Incremental 3d reassembly of axially symmetric pots from unordered and mixed fragment collections. https://github.com/SeongJong-Yoo/structure-from-sherds, 2021. 4, 5, 6, 7, 8

[8] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, July 2006. 2, 3

[9] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1482–1489 Vol. 2, 2005. 3

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 3

[11] Hubert Mara and Robert Sablatnig. Orientation of fragments of rotationally symmetrical 3d-shapes for archaeological documentation. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 3DPVT '06, pages 1064–1071, Washington, DC, USA, 2006. IEEE Computer Society. 2, 4

[12] J. C. McBride and B. B. Kimia. Archaeological fragment reconstruction using curve-matching. In *2003 Conference on Computer Vision and Pattern Recognition Workshop*, volume 1, pages 3–3, 2003. 2

[13] Georgios Papaioannou, Tobias Schreck, Anthousis Andreadis, Pavlos Mavridis, Robert Gregor, Ivan Sipiran, and Konstantinos Vardis. From reassembly to object completion: A complete systems pipeline. *J. Comput. Cult. Herit.*, 10(2), Mar. 2017. 1

[14] Helmut Pottmann, Martin Peternell, and Bahram Ravani. An introduction to line geometry with applications. *Computer-Aided Design*, 31(1):3–16, 1999. 2

[15] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36:1627–1639, 1964. 6

[16] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3

[17] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, page 835–846, New York, NY, USA, 2006. Association for Computing Machinery. 2

[18] Kilho Son, Eduardo B. Almeida, and David B. Cooper. Axially symmetric 3d pots configuration system using axis of symmetry and break curve. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 1, 2, 3, 4, 5, 6

[19] A. Willis, X. Orriols, and D. B. Cooper. Accurately estimating sherd 3d surface geometry with application to pot reconstruction. In *Proceedings of the 2003 Conference on Computer Vision and Pattern Recognition Workshop*, volume 1, pages 5–5, June 2003. 2

[20] A. R. Willis and D. B. Cooper. Bayesian assembly of 3d axially symmetric shapes from fragments. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, June 2004. 2, 3

[21] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 3

[22] C. Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 127–134, 2013. 2

[23] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1426–1433, 2010. 3, 4

[24] K. Zhang, W. Yu, M. Manhein, W. Waggenspack, and X. Li. 3d fragment reassembly using integrated template guidance and fracture-region matching. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2138–2146, 2015. 2, 3

Supplementary Document for Structure-from-Sherds: Incremental 3D Reassembly of Axially Symmetric Pots from Unordered and Mixed Fragment Collections

Je Hyeong Hong*† KIST, Hanyang University jhh37@hanyang.ac.kr

Seong Jong Yoo* KIST yoosj@kist.re.kr

Muhammad Zeeshan Arshad KIST zeeshan@kist.re.kr

Young Min Kim Seoul National University youngmin.kim@snu.ac.kr

Jinwook Kim† KIST zinook@kist.re.kr

Abstract

This document clarifies some statements from the main paper and illustrates further implementation details from the method section. Additionally, a video illustrating our pipeline can be found at supvid-1113.mp4.

0.1. Errata

We would like to correct some careless mistakes.

1. **Base fragment reassembly:** The sentence between l.733-736 should read *We achieve this by first gathering a set of sherds with the base region from Sec. 3.3, merging all broken bases and using this number with the number of full (unbroken) bases for initiating the incremental sherd registration process. The merging process involves iteratively selecting an unmerged partial base fragment as root (for incremental beam search) and reassembling all possible pieces until all partial base fragments are reassembled.* (Please refer to Fig. 4 from the main paper and the supplementary video.)
2. **Fracture surface normal:** the fracture surface normal of an edge-line point (stated in l.641 of the main paper) is different from the surface normal of the edge line point which points towards the axis of symmetry. The fracture surface normal of point j, $\hat{\mathbf{l}}_j \in S^2$, is the vector pointing orthogonal to the local fracture surface of the edge line point $\mathbf{p}_j \in \mathbb{R}^3$. It can be computed by taking the cross product between the point j's surface normal ($\hat{\mathbf{n}}_j \in S^2$) and the vector joining point j and point $j+1$, i.e.

$$\hat{\mathbf{l}}_j := \frac{\hat{\mathbf{n}}_j \times (\mathbf{p}_{j+1} - \mathbf{p}_j)}{\|\hat{\mathbf{n}}_j \times (\mathbf{p}_{j+1} - \mathbf{p}_j)\|_2} \tag{1}$$

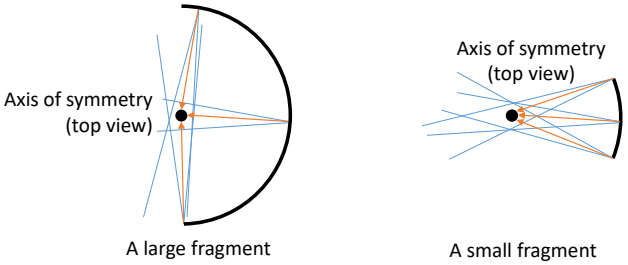


Figure 1. Visualization of uncertainties in the axis location for differently-sized fragments. Large fragments have wide basin of angle formed by the normals of the surface points, yielding a more rounded uncertainty. On the other hand, comparatively small fragments yield narrower basin of angle formed by the surface normals, yielding a stretched uncertainty region and leading to high uncertainty in the axis location.

3. **Joint sherd alignment:** (3) in l.698 of the main paper should be a minimization over $\{\mathtt{R}^i, \mathbf{t}^i\}$, where $\mathtt{R}^i \in SO(3)$ is the rotation matrix of sherd i and $\mathbf{t}^i \in \mathbb{R}^3$ is the translation of sherd i. Additionally, E_i should be E^i and E_j should be E^j to comply with our unified notation regarding the sherd number.
4. **Utilizing rims for match pruning:** During paper trimming, we accidentally left out the fact that we utilize the rim information during LCS matching to prune out false matches (i.e. matches involving a rim segment which is unrealistic). (Sec. 1.2)
5. Experimental settings can be found in Sec. 0.4.

*Both authors contributed equally to this work.
†Co-corresponding authors

0.2. Uncertainty in translation of the symmetric axis

Suppose each surface point $\mathbf{p} \in \mathbb{R}^3$ has some degree of noise (noted in blue) in its surface normal direction (in orange). When the noise distribution is accumulated over the surface points, a large fragment in Fig. 1 shows a well-closed noise distribution. On the other hand, a small fragment shows a longer accumulated noise distribution due to the narrow basin of angle formed by the surface points, leading to high uncertainty in the axis translation.

0.3. Our global method implementation

Many studies on reassembly of axially symmetric pots have not disclosed their codes, making it difficult to compare fairly between the methods.

As the second best choice, we implemented a global method with similar architecture to the ones in structure-from-motion [5, 3], first performing pairwise matches, second pruning these matches by inducing loop constraints (that the overall transformation around a loop should identity for rotation and zero for translation) and geometrically verifying scheme(3D overlap test and profile curve check), third using the maximum spanning tree (MST) algorithm to find initial rotation values and last performing joint sherd alignments (see Algorithm 1).

The global method failed to reconstruct a single pot from the 5 pots even in the single pot reassembly environment. After a series of debugging, we realized this is due to the global method being susceptible to false pairwise matches as shown in Fig. 3 (even after cycle filtering). The same figure shows the global pipeline succeeding when the false positive matches are removed. This implies the reassembly algorithms are encountering more false positive matches than previously anticipated. Some of these are difficult to prune out just by looking at the loop constraints, further demonstrating a need for an incremental method.

0.4. Experimental conditions

Computer specifications We used two different computers for the pipeline. For the preprocessing (feature extraction) part, runtimes were reported from a workstation with i7-7700 (3.6GHz) and 32GB of RAM. For the reassembly part, we used a workstation with a Ryzen 3960X (3.8 GHz) CPU and 128GB RAM.

Optimization settings We used the Ceres solver library [1] for all steps involving nonlinear optimization (e.g. refinement, sherd alignment). We employed the Sparse Schur complement solver (with SuiteSparse) with the function tolerance value set to 10^{-6}.

For all rotation matrix-related computations, we used the axis-angle parameterization.

Number of iterations in ICP For refining pairwise matches using the ICP algorithm, the inner loop of optimizing over the correspondences is solved using the Ceres solver with maximum of 50 iterations. The outer loop of ICP (comprising 1 alternating loop of correspondence search followed by correspondence minimization) is solved for maximum of 100 iterations. In both cases, the function tolerance value is set to 10^{-6}.

For ICP after adding a sherd, the settings are the same as above. For ICP during joint sherd alignment (which is the final step in each iteration of sherd registration), the outer loop iteration is set to maximum of 200 for convergence.

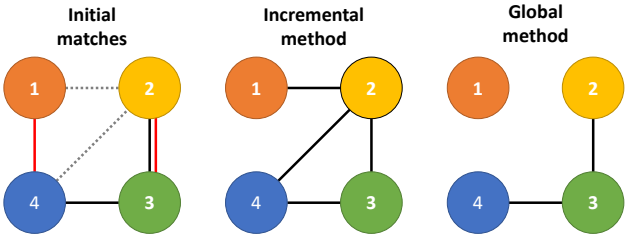


Figure 2. An example showing the potential advantage of an incremental approach. Each node denotes a pot sherd, and each edge is a pairwise match between the connected sherds (true positives are in black, false positives in red and false negatives in dotted grey). (Two different matches are initially formed between fragments 2 and 3.) If the incremental method starts from sherd 3 and registers sherd 2 correctly, its ability to refine the underlying model may lead to detection of previously undiscovered matches 1-2 and 2-4 and subsequently flag 1-4 as false positive. On the other hand, the global approach can only prune initial matches at best, potentially leading to missing or incorrect arrangement of sherd 1.

Algorithm 1 Our global registration method

Input: a graph $G(V, E)$ formed by pairwise matches (edges, $\{E\}$) between sherds (nodes, $\{V\}$)
1: Find all possible unique cycles formed by edges
2: **for** each cycle **do**
3: Compute the overall rotation and translation across the cycle
4: **if** the cycle rotation$\approx \mathtt{I}$ ($< 30°$) and the cycle translation close to $\mathbf{0}$ ($<$ 20mm) **then**
5: perform sherd alignment (Sec. 5.3 of the paper) between the participating sherds
6: Perform geometric verification (Sec. 1.2.3)
7: **end if**
8: **if** cycle is geometrically plausible **then**
9: # inliers is accrued to the participating edges of the cycle
10: **end if**
11: **end for**
12: Find maximum inlier spanning tree from filtered graph to find the absolute transformations $\{\mathtt{R}_i, \mathbf{t}_i\}$.
13: Refine individual sherd transformation by performing point-to-line ICP across all edge lines.
Outputs: sherd transformations $\{\mathtt{R}_i, \mathbf{t}_i\}$

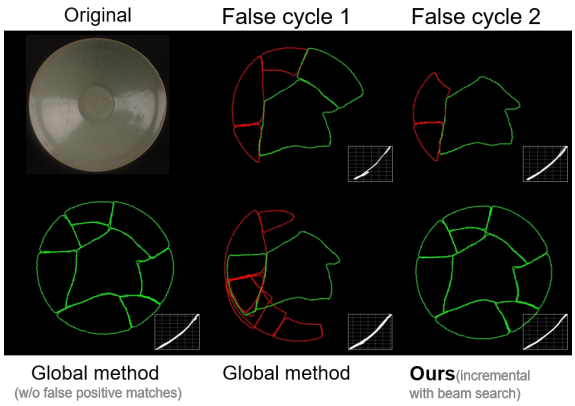
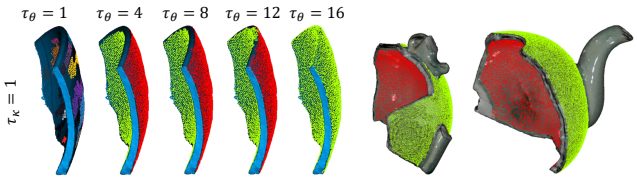


Figure 3. Some more false positive examples (pot B) demonstrating some false positive cycles which are visually indistinguishable. This leads to failures in the global method, necessitating a beam search-based incremental reassembly approach. (True configurations in green and false in red.)



(a) Results for different τ_θ (b) Decorative part removal

Figure 4. Segmentation results using the region growing algorithm in CGAL [4]. Our choice of hyperparameters (τ_θ=4, τ_κ=1) perfectly removes the non-smooth decorative parts in pot C.

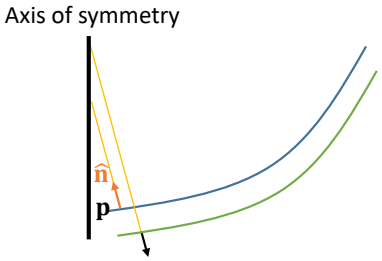


Figure 5. An illustration showing a method for correctly classifying interior and exterior surfaces. We project a ray from each point $\mathbf{p}$ along its surface normal $\hat{\mathbf{n}}$ and check where it meets (or is the closest to) the axis of symmetry. Ideally, the interior surface points will have their rays intersecting the axis along the respective positive normal directions, while the exterior surface points will have them along the respective negative normal directions.

1. Implementation details

1.1. Feature extraction

1.1.1 Interior and exterior surface extraction

We achieve this by segmenting the point cloud using the region growing algorithm [4]. For each point, a neighboring point (within $n_b = 10$ neighbouring points) is considered as part of the same surface if the angle between the normals of the two points is less than τ_θ, and the curvature value of the neighboring point is less than τ_κ. This yields a set of point clusters with each on the same smooth surface.

We set τ_θ=4 and τ_κ=1 such that the inner and outer surfaces are separated from the fractured regions and segmented into their own individual clusters (see Fig. 4). Then, we select the 2 largest clusters, which correspond to the inner and outer surface clusters for large-enough fragments. Segmenting between inner and outer surfaces is carried out after estimating the axis of symmetry.

Decorative part removal For the fragments containing decorative parts or their remains (e.g. a handle or a nose), above parameters also segment those regions, enabling us to exclude these parts from the inner and outer surfaces.

1.1.2 Interior and exterior surface classification

From above, we derive two sets of surfaces one of which is the exterior surface and the other is the interior surface.

To classify each surface correctly, we project a ray from each point on the surface $\mathbf{p} \in \mathbb{R}^3$ along its surface normal direction $\hat{\mathbf{n}} \in S^2$ and check where it meets the axis of symmetry (in practice where it makes the shortest distance between the symmetric axis and the projected ray). Then, we check whether this point of (near-)intersection is along the positive surface normal direction from the point $\mathbf{p}$ or negative surface normal direction (i.e. opposite). As shown in Fig. 5, if the surface is on the interior side, then the point of intersection should be along the positive normal direction, and if the surface is on the exterior side, then the point of intersection is along the negative normal direction.

We test two configurations (i.e. inner & outer vs outer & inner for the two surfaces) and choose the configuration with more number of surface points from both surfaces satisfying above geometry (just need to check if the scalar coefficient of the ray's normal is positive or negative).

1.1.3 Edge line extraction and segmentation

The edge line extraction starts with getting the boundary of the inner point cloud using boundary-estimation from the PCL library [2]. We then sample and sequence these points in order to have an equal distance, d between each point. We set this d = 1.9mm for this work.

Setting the order of the edge line In order to efficiently match edge lines using our LCS descriptor matching algorithm, we need to fix the ordering of the edge line of each sherd's inner surface to either a clockwise or an anticlockwise direction (otherwise, we will have to test each pair of sherds twice more times due to ambiguity in ordering, once clockwise and once anticlockwise).

We achieve this by first computing the mean point $\bar{\mathbf{p}} \in$

Algorithm 2 Our implementation of the LCS algorithm

Inputs: quantized descriptor matrix of sherd A and B

$n \leftarrow$ number of edge line points in sherd A
$m \leftarrow$ number of edge line points in sherd B
Create a 2D array $D \in \mathbb{R}^{N+1 \times M+1}$
for $i = 1, .., N+1$ and $j = 1, .., M+1$ **do**
  $\mathbf{e} \leftarrow f(\mathbf{p}_i^A) - f(\mathbf{p}_j^B)$
  if $i == 1$ or $j == 1$ or $p_i^A \in$ rim or $p_j^B \in$ rim or $\mathbf{e} > \delta_e$ **then**
    $D_{i,j} \leftarrow 0$
  else if $\mathbf{e} < \delta_e$ **then**
    $D_{i,j} \leftarrow D_{i-1,j-1} + 1$
    if $D_{i,j} > l_m$ **then**
      Append $(D_{i,j}, i, j)$ to I^{AB}
      Delete previous entry $(D_{i-1,j-1}, i-1, j-1)$ from I^{AB}
    end if
  end if
end for

Outputs: set of matching intervals I^{AB}

Algorithm 3 Clustering matching intervals

Inputs: set of raw matching intervals I^{AB}

Create an empty set of clusters C and an empty set I^{*AB}
$l \leftarrow 0$
for each interval I_k^{AB} in I^{AB} **do**
  if the midpoint of I_k^{AB} is far from all intervals in C ($<$ 20mm) **then**
    Assign a new cluster by defining a new set C_l and adding C_l to C
    $l \leftarrow l + 1$
    Add I_k^{AB} to C_l
  else
    Add I_k^{AB} to the closest cluster C_m
  end if
end for
for each cluster set $C_m \in C$ **do**
  Find longest matching interval $I_k^{AB} \in C_m$ and add to I^{*AB}
end for

Outputs: pruned set of matching intervals (I^{*AB})

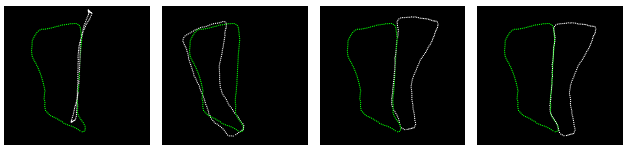
$\mathbb{R}^3$ of the edge line points $\{\mathbf{p}\}$. Then, for each point $\mathbf{p}_j$, we draw vectors $\mathbf{p}_j - \bar{\mathbf{p}}$ and $\mathbf{p}_{j+1} - \bar{\mathbf{p}}$ and check if the cross product vector $(\mathbf{p}_j - \bar{\mathbf{p}}) \times (\mathbf{p}_{j+1} - \bar{\mathbf{p}})$ is in the same direction as its computed normal $\hat{\mathbf{n}}_j$ (i.e. their dot product is greater than 0). If above is the case for the majority of points, then the edge line is aligned in the anti-clockwise direction, and vice versa. We set each edge line to be in the anti-clockwise direction.

Computing surface normals Though we could compute the normals for these points from the original point clouds, however, the normals at the edge of surface are not always exact. Hence, we fit a B-spline surface on the inner point cloud and get the correct normals for these points through the fitted B-spline surface. This sequence of equally spaced points, combined with B-spline surface based normals are from here on referred to as the edge line.

Edge line segmentation The algorithm for segmenting the edge line relies on the detection of corners and sharp curves. For an edge line with points $p_1, p_2, p_3, \ldots, p_n$. For a threshold of $n = 10$, for each point p_j, we draw a line A between points p_{j-10} and p_{j+10}. Then we get the shortest distance d between the point p_j and the line A. The distance d is smaller when the edge line is more straight and larger when the edge line curves. We then find peaks in the value of d and identify them as the endpoints of edge line segments.

1.1.4 Thickness computation

For each point $\mathbf{p} \in \mathbb{R}^3$ on the edge line on the inner surface, we extend the surface normal to project a ray in the opposite direction towards the outer surface. We then find a point $\mathbf{q} \in \mathbb{R}^3$ that lies on the outer surface that is closest to that ray and also satisfies the following two conditions: (a) The distance between the point and the ray is within a distance threshold (1 mm) (b) The direction of $(\mathbf{p} - \mathbf{q})$ is opposite direction of $\hat{\mathbf{n}}^{\mathbf{p}}$ The thickness is then measured as the distance between the points $\mathbf{p}$ and $\mathbf{q}$.



(a) Point-to-point ICP (point-wise distances only) (b) Point-to-point ICP (point-wise normals only) (c) Point-to-point ICP (d) Point-to-line ICP

Figure 6. Pairwise ICP results based on different cost functions. (c) is better than (a) or (b) but is slightly misaligned. (d) shows the optimal matching result.

1.2. Pairwise matching

1.2.1 Descriptor matching

The inputs of the pairwise matching process are i) a descriptor matrix of sherd A formed by column-stacking descriptor vectors $\{\mathbf{f}_j\}$ (each $\mathbf{f}_j \in \mathbb{R}^4$) for each (inner surface) edge line point from sherd A, and ii) a descriptor matrix of the same nature from sherd B. Each descriptor matrix is quantized every 0.15 unit except for the thickness which is quantized per 0.2 due to its practically lower stability.

The output of the matching process is the set of matching intervals I^{AB} between sherds A and B, where the set I^{AB} comprises matching intervals $\{I_k^{AB}\}$ and each matching interval I_k^{AB} is defined as an array of 3 numbers, namely the length of the matching interval, the interval end point index (inclusive) for sherd A and the interval end point index (inclusive) for sherd B.

The process runs in two steps as follows:

1. In order to get the set of matching intervals ($I^{AB} = \{I_k^{AB}\}$), the quantized descriptors are compared on a point-by-point basis, and the interval is defined as the region which continuously satisfies each element of the error vector between the compared descriptors being less than the correspoding element of the threshold vector $\delta_e = [0.45, 0.45, 0.45, 0.8]^\top$ (see Algorithm 2).
2. As step 1 yields multiple similar matching intervals, we cluster these intervals by selecting the longest interval from each cluster. The number of clusters is jointly estimated using a greedy algorithm in Algorithm 3.

Pruning via rim constraints If the LCS outputs a cluster around the rim segment, then we discard the match (since matching along the rim part is infeasible). This results in nearly 1/3 drop in the number of matches, all of which are false positives.

1.2.2 Refining matches

Figure. 6 shows our reason for adopting the point-to-line ICP method rather than point-to-point, that it shows the best matching performance in our application.

We define the transformation of sherd A as $\mathtt{T}^A$, which consists of rotation $\mathtt{R}^A \in SO(3)$ and translation $\mathbf{t}^A \in \mathbb{R}^3$. If use the notation (i, j) to denote the correspondence between the i-th edge line point of sherd A ($\mathbf{p}_i^A \in \mathbb{R}^3$) and the j-th edge line point of sherd B ($\mathbf{p}_j^B \in \mathbb{R}^3$), the correspondence distance term $d_{ij}(\mathtt{T}^A, \mathtt{T}^B)$ can be expressed as (4), the correspondence normal deviation term $e_{ij}(\mathtt{T}^A, \mathtt{T}^B)$ as (5) and the rim consistency term $g_i(\mathtt{T}^A, r, h)$ as (6). Then point-to-line ICP (P2L ICP) can be defined as (2) and point-to-point ICP (P2P ICP) is using $m_{ij}(\mathtt{T}^A, \mathtt{T}^B)$ function(defined as (7)) instead of $d_{ij}(\mathtt{T}^A, \mathtt{T}^B)$ at same the equation which can be defined as (3).

$$\min_{\mathtt{T}^A, \mathtt{T}^B, r, h} \sum_{(i,j)\in\Omega_{AB}} \rho_d(d_{ij}^2(\mathtt{T}^A, \mathtt{T}^B)) + \lambda\rho_e(e_{ij}^2(\mathtt{T}^A, \mathtt{T}^B)) + \nu \sum_{E\in A,B} \sum_{i\in\Psi_E} \rho_g(g_i^2(\mathtt{T}^E, r, h)) \tag{2}$$

$$\min_{\mathtt{T}^A, \mathtt{T}^B, r, h} \sum_{(i,j)\in\Omega_{AB}} \rho_d(||m_{ij}(\mathtt{T}^A, \mathtt{T}^B)||_2^2) + \lambda\rho_e(e_{ij}^2(\mathtt{T}^A, \mathtt{T}^B)) + \nu \sum_{E\in A,B} \sum_{i\in\Psi_E} \rho_g(g_i^2(\mathtt{T}^E, r, h)) \tag{3}$$

Iteration	ICP type	ρ_d	ρ_e	ρ_g	λ	ν
First	P2P	5	2	5	3	1.5
Second	P2L	5	5	5	3	1.5

Table 1. Pairwise ICP implementation details

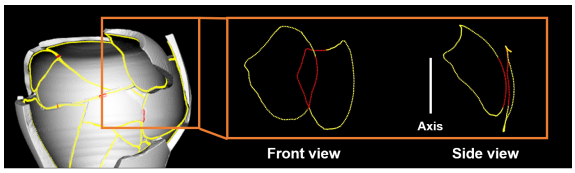


Figure 7. Example of a radial overlap case (which is false and thus discarded) from Sec. 1.2.3. Red line is the overlapped region.

where ρ_d, ρ_e and ρ_g are robust kernels to suppress outlier correspondences, Ω_{AB} is the set of edge line correspondences between sherds A and B, Ψ_E is the set of edge line points classified as rim in sherd E (A or B), $\hat{\mathbf{l}}_i$ is estimated fracture surface normal of i-th point defined as (8). We use median value of radius and height as the initial value of $\bar{r}$ and $\bar{h}$ at (6). In order to converge stably, while minimizing correspondence distances, we solve the equation (2) twice with different settings as shown in Table 1.

$$\begin{aligned} d_{ij}(\mathtt{T}^A, \mathtt{T}^B) &:= d(\mathbf{p}_i^A, \hat{\mathbf{n}}_i^A, \mathbf{p}_j^B, \hat{\mathbf{n}}_j^B, \mathtt{T}^A, \mathtt{T}^B) \\ &= ||\mathtt{R}^A\hat{\mathbf{l}}_i^A \cdot m_{ij}(\mathtt{T}^A, \mathtt{T}^B)||_2^2 + ||\mathtt{R}^B\hat{\mathbf{l}}_j^B \cdot m_{ij}(\mathtt{T}^A, \mathtt{T}^B)||_2^2 \\ &+ ||\mathtt{R}^A\hat{\mathbf{n}}_i^A \cdot m_{ij}(\mathtt{T}^A, \mathtt{T}^B)||_2^2 + ||\mathtt{R}^B\hat{\mathbf{n}}_j^B \cdot m_{ij}(\mathtt{T}^A, \mathtt{T}^B)||_2^2 \end{aligned} \tag{4}$$

$$\begin{aligned} e_{ij}(\mathtt{T}^A, \mathtt{T}^B) &:= e(\hat{\mathbf{n}}_i^A, \hat{\mathbf{n}}_j^B, \mathtt{T}^A, \mathtt{T}^B) \\ &= ||\mathtt{R}^A\hat{\mathbf{n}}_i^A - \mathtt{R}^B\hat{\mathbf{n}}_j^B||_2^2 \end{aligned} \tag{5}$$

$$\begin{aligned} g_i(\mathtt{T}^A, r, h) &:= g(\mathbf{p}_i^A, \mathtt{T}^A, r, h) \\ &= ||\bar{r} - r(\mathtt{R}^A\mathbf{p}_i^A + \mathbf{t}^A)||_2^2 + ||\bar{h} - h(\mathtt{R}^A\mathbf{p}_i^A + \mathbf{t}^A)||_2^2 \end{aligned} \tag{6}$$

$$\begin{aligned} m_{ij}(\mathtt{T}^A, \mathtt{T}^B) &:= m(\mathbf{p}_i^A, \hat{\mathbf{n}}_i^A, \mathbf{p}_j^B, \hat{\mathbf{n}}_j^B, \mathtt{T}^A, \mathtt{T}^B) \\ &= \mathtt{R}^A\mathbf{p}_i^A + \mathbf{t}^A - \mathtt{R}^B\mathbf{p}_j^B + \mathbf{t}^B \end{aligned} \tag{7}$$

$$\hat{\mathbf{l}}_i := (\hat{\mathbf{n}}_i \times (\mathbf{p}_{i+1} - \mathbf{p}_i))/||\hat{\mathbf{n}}_i \times (\mathbf{p}_{i+1} - \mathbf{p}_i)||_2 \tag{8}$$

1.2.3 Geometric verification

Checking potential overlaps

Our overlap test illustratd in Algorithm 4 is designed for detecting intersections in 3D. The two conditions with asterisks (*) in Algorithm 4 are designed to extract potential areas of overlap and detect radial overlap cases, which are:

Algorithm 4 Algorithm for checking overlaps

Inputs: edge lines for sherds A ($\{\mathbf{p}^A\}$) and B ($\{\mathbf{p}^B\}$)
Find a set of correspondence pairs C^{AB} between $\{\mathbf{p}^A\}$ and $\{\mathbf{p}^B\}$, where $C^{AB} := \{C_j^{AB}$ with $C_j^{AB} := (\mathbf{p}_j^A, \mathbf{p}_j^B)\}$)
Select a region (i.e. a range of correspondence pairs) for testing potential overlap between the two edge lines based on one of two conditions (*) listed in Sec. 1.2.3.
for each correspondence pair (C_j^{AB}) in the investigated region **do**
  if C_j^{AB} satisfies at least one of the two conditions (†) **then**
    Area of overlap $\leftarrow$ Area of overlap + $\left\|\mathbf{p}_j^A - \mathbf{p}_j^B\right\|_2$
  end if
end for
Outputs: Area of overlap

Algorithm 5 Checking the consistency of the profile curve

Inputs: edge line points ($\{\mathbf{p}_j^i\}$) across matched sherds
result $\leftarrow$ true
Align the edge points $\{\mathbf{p}_j^i\}$ to the axis of symmetry (z^+ is now in the axis direction)
Compute the radii of edge line points ($\{r_j^i\}$) from the axis-aligned x and y coordinate values.
Sort edge line points in ascending order of z values.
segment line points $\{\mathbf{p}_j^i\}$ into bins every w (5 mm) in z.
for each bin of line points **do**
  Fit a line using orthogonal regression.
  Compute standard deviation (σ) of orthogonal distance errors between the line of best fit and the edge line points.
  if $\sigma > \delta_d$ **then**
    result $\leftarrow$ fail
    break
  end if
end for
Output: result

1. $\left\|\mathbf{p}_j^A - \mathbf{p}_j^B\right\|_2 < d$, and
2. $\mathbf{n}_j^A \cdot \mathbf{n}_j^B > 0$ and $|\frac{(\mathbf{p}_j^A - \mathbf{p}_j^B)}{|\mathbf{p}_j^A||\mathbf{p}_j^B|}| \cdot \mathbf{n}_j^A > \theta$

The first condition extracts the likely correspondence pairs between edge line of sherd A and that of sherd B (we set $d = 5$mm)). The second condition detects the radial direction overlap as shown in Fig. 7.

If the region of interest is detected, we go through two further tests to check for different overlap cases as follows:

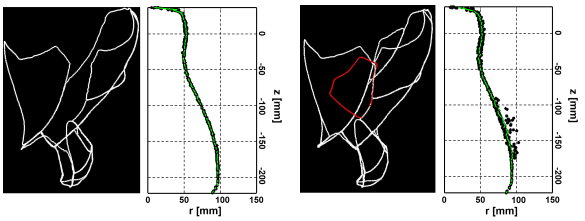
1. $\hat{\mathbf{l}}_j^A \cdot \hat{\mathbf{l}}_j^B > 0$ and $\left\|\mathbf{p}_j^A - \mathbf{p}_j^B\right\|_2 < d$ (this occurs when one piece is almost submerged into another piece),
2. $\hat{\mathbf{l}}_j^A \cdot (\mathbf{p}_j^B - \mathbf{p}_j^A) < 0$ and $\hat{\mathbf{l}}_j^B \cdot (\mathbf{p}_j^B - \mathbf{p}_j^A) > 0$ (this occurs when two pieces match as expected but with some overlaps),

where $\mathbf{l}_j^i$ is the fracture surface normal of point j in sherd i defined in (1).

If the overlapped region is greater than 50 mm^2, then the pair of sherds A and B is removed from the list of matches.

Checking the profile curve Overlap test on its own cannot filter out all false positive matches as shown in Fig. 3. Therefore, we implemented a step checking the consistency of the profile curve. The profile curve is a 2D rz-plane projection of the pot, and each axially symmetric pot should observe a single and continuous curve (see Fig. 8).

Based on the above property we divide the profile curve into segments every 5mm in z, and fit a line onto each segment. We then check the error between the fitted line and the points in each segment. If the standard deviation of this error across all segments is above 5 mm, then the configuration is deemed false due to an inconsistent profile curve. Our algorithm is illustrated in Algorithm 5.



(a) correct configuration (b) incorrect case detected by the profile curve test

Figure 8. The left side of (a) and (b) show the edge lines of matched sherds, and the right side of (a) and (b) show the corresponding profile curves (with locally fitted lines in green). In (b), the overlap test fails to find a false positive configuration but the profile curve test detects it.

1.3. Incremental sherd registration with multi-root beam search

1.3.1 Sherd registration

When registering a new sherd (denoted as sherd D) to the reassembled sherds $\{C\}$, we align the coordinates by the sherd C's axis of symmetry (defined as the z-axis without loss of generality) and keep it fixed. Noting the notations from (2), we essentially solve

$$\min_{\mathtt{T}^D} \sum_{E\in\{C\}} \sum_{(i,j)\in\Omega_{DE}} \rho_d(d_{ij}^2(\mathtt{T}^D, \mathtt{T}^E)) + \lambda\rho_e(e_{ij}^2(\mathtt{T}^D, \mathtt{T}^E)) + \mu \sum_{i\in\Omega_D} \rho_f(f_i^2(\mathtt{T}^D)) + \nu \sum_{i\in\Psi_D} \rho_g(g_i^2(\mathtt{T}^D, r, h)) \quad (9)$$

where Ω_D is the set of edge line points in sherd D, Ψ_D is a subset of Ω_D classified as rim, and f_i is a measure of the

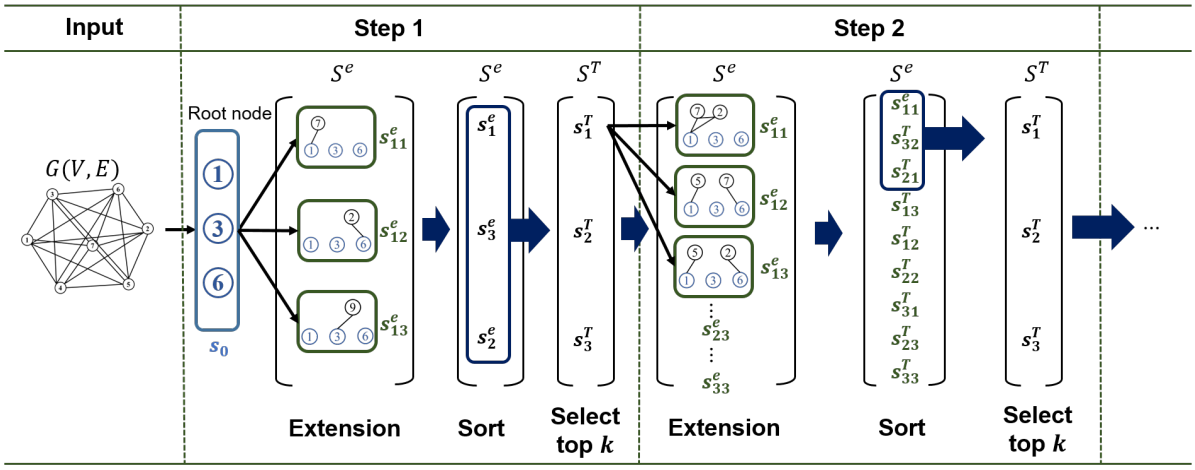


Figure 9. An exemplary illustration of our beam search pipeline for the case of $k = 3$ and $b = 3$.

Iteration	ICP type	ρ_d	ρ_e	ρ_f	ρ_g	λ	ν	μ
First	P2P	5	2	0	1.5	3	1	0
Second	P2L	5	2	0	1.5	3	1	0
Third	P2L	5	2	1.5	1.5	1	1	0.1

Table 2. Registration ICP implementation details

Iteration	ICP type	ρ_d	ρ_e	ρ_f	ρ_g	λ	ν	μ
First	P2L	5	2	0	0	2	0	0
Second	P2P	5	2	1.5	5.0	2	1	0.3

Table 3. Batch sherd alignment ICP implementation details

axial consistency in sherd D. Similar with pairwise matching as mentioned in Sec 1.2.2, we utilize the ICP algorithm and solve the equation (9) three times to make it stably converge with the different settings as shown in Table 2.

1.3.2 Batch sherd alignment

We refine all reassembled sherds by utilizing ICP algorithm. When $\{C\}$ is the set of reassembled sherds, and $\{T^C\}$ is the set of its transformation matrix, then by noting notations from (2) and (9), we solve

$$\min_{\{\mathtt{T}^C\},r,h} \sum_{(E,F)\in\Omega_{\{C\}}} \sum_{(i,j)\in\Omega_{EF}} \Big(\rho_d(d_{ij}^2(\mathtt{T}^E,\mathtt{T}^F)) + \lambda\rho_e(e_{ij}^2(\mathtt{T}^E,\mathtt{T}^F))\Big) + \sum_{F\in\Omega_{\{C\}}} \Big(\mu \sum_{i\in\Omega_F} \rho_f(f_i^2(\mathtt{T}^F)) + \nu \sum_{i\in\Psi_F} \rho_g(g_i^2(\mathtt{T}^F,r,h))\Big) \tag{10}$$

where (E, F) denote a pair of sherds in set of sherd pairs $\Omega_{\{C\}}$, Ψ_F is the set of ege line points classified as rim in sherd F. Again, we solve equation (10) two times to make it stably converge with the different settings as shown in Table 3.

1.3.3 Multi-root beam search

An illustration of our incremental sherd registration algorithm can be found in Fig. 9. This algorithm can also be applied to reassembling base fragments but using a single root (an unused partial base fragment) only.

Generating a priority list for branch extension When deciding which sherds to try out for the "Extension" step in Step 1 of Fig. 9, we rely on the priority ranking of sherds determined by the adjusted number of inliers (defined in l.680 of the main paper).

One important thing to note is that if two pairwise matches yield the same configuration with the newly added sherd, then the two matches are merged as a single candidate configuration in the priority list with the adjusted number of inliers being the sum of these from the two matches (i.e. being more likely). For example, suppose sherds A and B are already merged, and we find a pairwise match between sherd A and new sherd C with V adjusted number of inliers, and another match between sherd B and sherd C with W adjusted number of inliers. If both matches lead to the same transformation matrix for sherd C (i.e. same $\mathtt{R}^C, \mathbf{t}^c$ up to some threshold values), then the two configurations ($A - C$ and $B - C$) are merged as a single configuration with $W + V$ adjusted number of inliers. This encourages fragments with more neighboring sherds to be favored.

2. Reassembly results

Fig. 10 shows single reconstruction result about pot D and E in various top k and extension b values. White mesh

and green break line indicate correct configurations, and red mesh and break line demonstrate false positive configurations. It is difficult to detect false positives without referring to the ground truth. Fig. 11-12 illustrate mixed pots reconstruction results across various settings of ranks (k) and branches (b). Increasing the values of k and b increases the number of correctly configured sherds as anticipated.

References

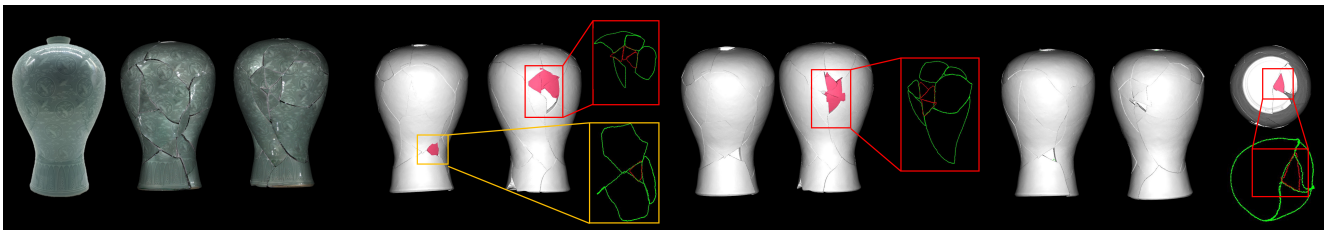
[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org. 2

[2] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. 3

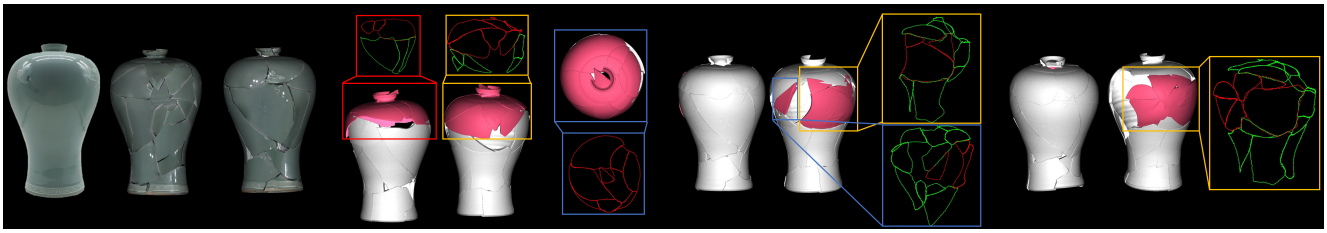
[3] Chris Sweeney. Theia multiview geometry library: Tutorial & reference. http://theia-sfm.org. 2

[4] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.2 edition, 2020. 3

[5] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1426–1433, 2010. 2

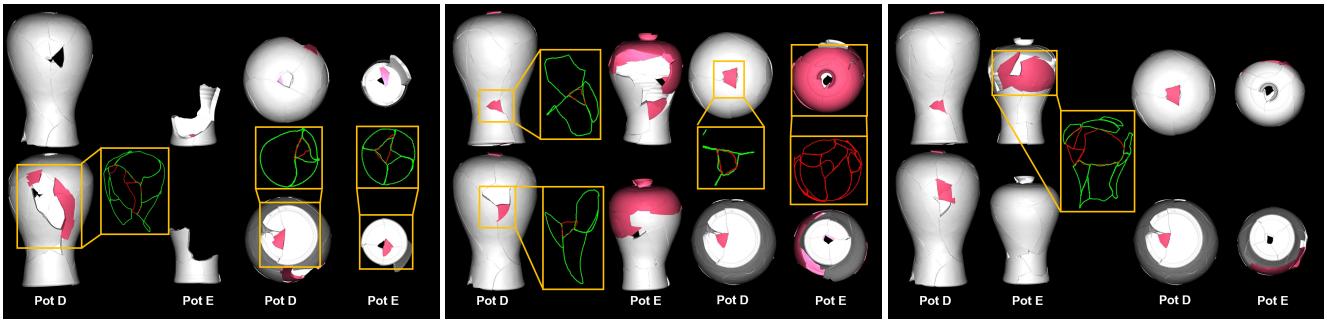


(a) Pot D, Origin (b) Pot D, $k = 5$ and $b = 3$ (c) Pot D, $k = 10$ and $b = 5$ (d) Pot D, $k = 20$ and $b = 10$



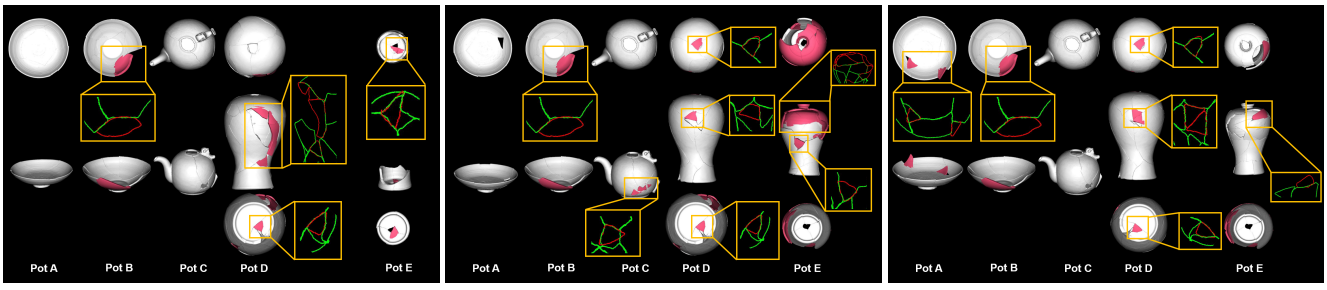
(e) Pot E, Origin (f) Pot E, $k = 5$ and $b = 3$ (g) Pot E, $k = 10$ and $b = 5$ (h) Pot E, $k = 20$ and $b = 10$

Figure 10. The experiment result of single pot D and E reconstruction with with various numbers of ranks (k) and branches (b). Red fragments indicate false positive configurations.



(a) $k = 5$ and $b = 3$ (b) $k = 10$ and $b = 5$ (c) $k = 20$ and $b = 10$

Figure 11. The experiment result of mixed pot E and D reconstruction with various numbers of ranks (k) and branches (b). Red fragments indicate false positive configurations.



(a) $k = 5$ and $b = 3$ (b) $k = 10$ and $b = 5$ (c) $k = 20$ and $b = 10$

Figure 12. Experiment result of mixed 5 different pots reconstruction with various numbers of ranks (k) and branches (b). Red fragments indicate false positive configurations.