

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.

- Base fragment reassembly:** The sentence between 1.733-736 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.)
- Fracture surface normal:** the fracture surface normal of an edge-line point (stated in 1.641 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

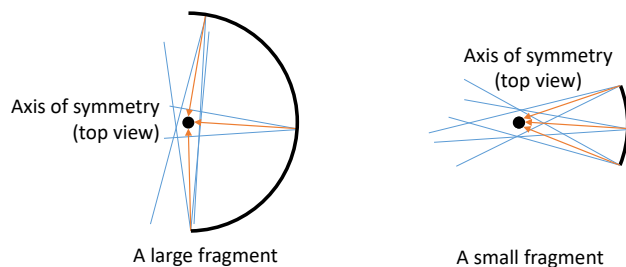


Figure 1. Visualization of uncertainties in the axis location for differently-sized fragments. Large fragments have wide basin of angle formed by the normals of the surface points, yielding a more rounded uncertainty. On the other hand, comparatively small fragments yield narrower basin of angle formed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

point $j + 1$, i.e.

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

- Joint sherd alignment:** (3) in 1.698 of the main paper should be a minimization over $\{\mathbf{R}^i, \mathbf{t}^i\}$, where $\mathbf{R}^i \in SO(3)$ 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.
- 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)
- 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

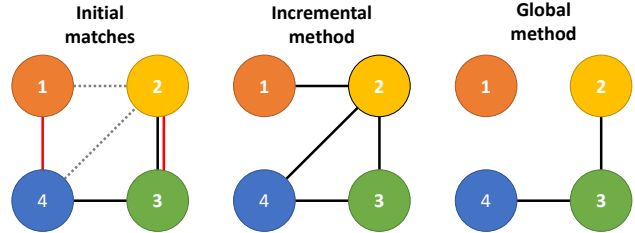


Figure 2. An example showing the potential advantage of an incremental approach. Each node denotes a pot sherd, and each edge is a pairwise match between the connected sherds (true positives are in black, false positives in red and false negatives in dotted grey). (Two different matches are initially 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 $\approx \mathbf{I}$ ($< 30^\circ$) and the cycle translation close to $\mathbf{0}$ ($< 20\text{mm}$) **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 $\{R_i, \mathbf{t}_i\}$.
- 13: Refine individual sherd transformation by performing point-to-line ICP across all edge lines.

Outputs: sherd transformations $\{R_i, \mathbf{t}_i\}$

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.

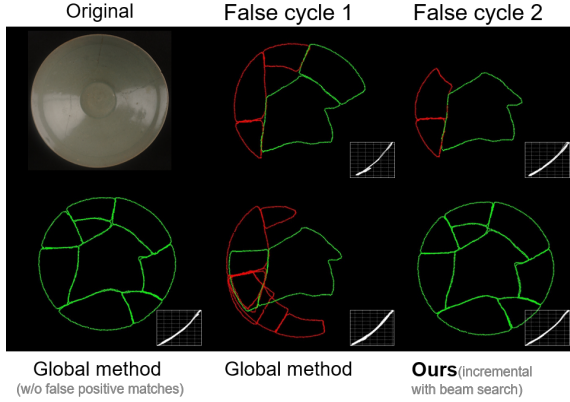


Figure 3. Some more false positive examples (pot B) demonstrating some false positive cycles which are visually indistinguishable. This leads to failures in the global method, necessitating a beam search-based incremental reassembly approach. (True configurations in green and false in red.)

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.

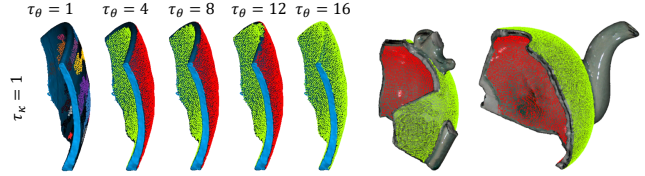
We set $\tau_\theta=4$ and $\tau_\kappa=1$ such that the inner and outer surfaces are separated from the fractured regions and segmented into their own individual clusters (see Fig. 4). Then, we select the 2 largest clusters, which correspond to the inner and outer surface clusters for large-enough fragments. 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



(a) Results for different θ (b) Decorative part removal
Figure 4. Segmentation results using the region growing algorithm in CGAL [4]. Our choice of hyperparameters ($\theta=4$, $\kappa=1$) perfectly removes the non-smooth decorative parts in pot C.

Axis of symmetry

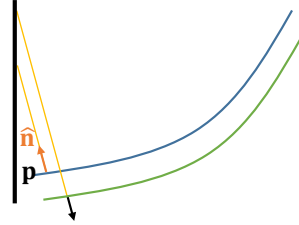


Figure 5. An illustration showing a method for correctly classifying interior and exterior surfaces. We project a ray from each point \mathbf{p} along its surface normal $\hat{\mathbf{n}}$ and check where it meets (or is the closest to) the axis of symmetry. Ideally, the interior surface points will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

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 $d = 1.9\text{mm}$ 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 sherds A and B

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

Outputs: 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 $p_1, p_2, p_3, \dots, p_n$. 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

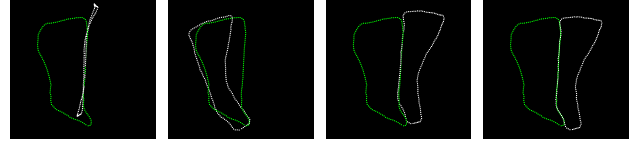
For each point $\mathbf{p} \in \mathbb{R}^3$ on the edge line on the inner surface, we extend the surface normal to project a ray in the

Algorithm 3 Clustering matching intervals

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

- 1: Create an empty set of clusters C and an empty set I^{AB}
- 2: $l \leftarrow 0$
- 3: **for** each interval I_k^{AB} in I^{AB} **do**
- 4: **if** the midpoint of I_k^{AB} is far from all intervals in C ($< 20\text{mm}$) **then**
- 5: Assign a new cluster by defining a new set C_l and adding C_l to C
- 6: $l \leftarrow l + 1$
- 7: Add I_k^{AB} to C_l
- 8: **else**
- 9: Add I_k^{AB} to the closest cluster C_m
- 10: **end if**
- 11: **end for**
- 12: **for** each cluster set $C_m \in C$ **do**
- 13: Find longest matching interval $I_k^{AB} \in C_m$ and add to I^{AB}
- 14: **end for**

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



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

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 $\hat{\mathbf{n}}^P$. The thickness is then measured as the distance between the points \mathbf{p} and \mathbf{q} .

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_k^{AB} between sherds A and B, where the set 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 $I_k^{AB} = \{I_k^{AB} | k \in \mathbb{N}\}$, 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 corresponding element of the threshold vector $\mathbf{e} = [0.45; 0.45; 0.45; 0.8]^T$ (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 T^A , which consists of rotation $R^A \in SO(3)$ and translation $t^A \in R^3$. If use the notation $(i; j)$ to denote the correspondence between the i -th edge line point of sherd A ($p_i^A \in R^3$) and the j -th edge line point of sherd B ($p_j^B \in R^3$), the correspondence distance $d_{ij}(T^A; T^B)$ can be expressed as (4), the correspondence normal deviation $e_{ij}(T^A; T^B)$ as (5) and the rim consistency $g_i(T^A; r; h)$ as (6). Then point-to-line ICP (P2L ICP) can be defined as (2) and point-to-point ICP (P2P ICP) is using $m_{ij}(T^A; T^B)$ function (defined as (7)) instead of $d_{ij}(T^A; T^B)$ at same the equation which can be defined as (3).

$$\min_{T^A; T^B; r; h} \sum_{(i; j) \in \mathcal{I}^{AB}} d(d_{ij}^2(T^A; T^B)) + \sum_{E \in \mathcal{E}^{A; B}} g(g_i^2(T^E; r; h)) \quad (2)$$

$$\min_{T^A; T^B; r; h} \sum_{(i; j) \in \mathcal{I}^{AB}} d((j m_{ij}(T^A; T^B))^2) + \sum_{E \in \mathcal{E}^{A; B}} g(g_i^2(T^E; r; h)) \quad (3)$$

Iteration	ICP type	d	e	g		
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 d , e and g are robust kernels to suppress outlier correspondences. \mathcal{I}_{AB} is the set of edge line correspondences between sherds A and B, \mathcal{E} is the set of edge line points classified as rim in sherd A (or B), \hat{n}_i is estimated fracture surface normal of i th point defined as (8). We use a median value of radius and height as the initial value r and 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.

$$\begin{aligned} d_{ij}(T^A; T^B) &:= d(p_i^A; \hat{n}_i^A; p_j^B; \hat{n}_j^B; T^A; T^B) \\ &= \|j R^A \hat{n}_i^A - m_{ij}(T^A; T^B)\|_2^2 + \|j R^B \hat{n}_j^B - m_{ij}(T^A; T^B)\|_2^2 \\ &\quad + \|j R^A \hat{n}_i^A - m_{ij}(T^A; T^B)\|_2^2 + \|j R^B \hat{n}_j^B - m_{ij}(T^A; T^B)\|_2^2 \end{aligned} \quad (4)$$

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

$$\begin{aligned} g_i(T^A; r; h) &:= g(p_i^A; T^A; r; h) \\ &= \|j r - r(R^A p_i^A + t^A)\|_2^2 + \|j h - h(R^A p_i^A + t^A)\|_2^2 \end{aligned} \quad (6)$$

$$\begin{aligned} m_{ij}(T^A; T^B) &:= m(p_i^A; \hat{n}_i^A; p_j^B; \hat{n}_j^B; T^A; T^B) \\ &= R^A p_i^A + t^A - R^B p_j^B + t^B \end{aligned} \quad (7)$$

$$\hat{n}_i := (\hat{n}_i - (p_{i+1} - p_i)) = \| \hat{n}_i - (p_{i+1} - p_i) \|_2 \quad (8)$$

1.2.3 Geometric verification

Checking potential overlaps

Our overlap test illustrated 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 ($f p^A g$) and B ($f p^B g$)

- 1: Find a set of correspondence pairs C_j^{AB} between $f p^A g$ and $f p^B g$, where $C_j^{AB} := f C_j^{AB}$ with $C_j^{AB} := (p_j^A; p_j^B)g$
- 2: 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.
- 3: for each correspondence pair $(p_j^A; p_j^B)$ in the investigated region do
- 4: if C_j^{AB} satisfies at least one of the two conditions then
- 5: Area of overlap = Area of overlap + $p_j^A p_j^B$
- 6: end if
- 7: end for

Outputs: Area of overlap

Algorithm 5 Checking the consistency of the profile curve

- 1: Inputs: edge line points $(p_j^i g)$ across matched sherds
- 2: result = true
- 3: Align the edge points $(p_j^i g)$ to the axis of symmetry z^i (is now in the axis direction)
- 4: Compute the radii of edge line points (r_j^i) from the axis-aligned x and y coordinate values.
- 5: Sort edge line points in ascending order of r_j^i values.
- 6: segment line points $(p_j^i g)$ into bins every w (5 mm) in z .
- 7: for each bin of line points do
- 8: Fit a line using orthogonal regression.
- 9: Compute standard deviation (σ) of orthogonal distance errors between the line of best fit and the edge line points.
- 10: if $\sigma > d$ then
- 11: result = fail
- 12: break
- 13: end if
- 14: end for
- 15: Output: result

1. $p_j^A p_j^B < d$, and
2. $n_j^A n_j^B > 0$ and $j \frac{(p_j^A p_j^B)}{p_j^A p_j^B} j n_j^A >$

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. $\hat{n}_j^A \hat{n}_j^B > 0$ and $p_j^A p_j^B < d$ (this occurs when one piece is almost submerged into another piece),
2. $\hat{n}_j^A (p_j^B p_j^A) < 0$ and $\hat{n}_j^B (p_j^B p_j^A) > 0$ (this

(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.

occurs when two pieces match as expected but with some overlaps),

where \hat{n}_j^i is the fracture surface normal of point n in sherd i defined in (1).

If the overlapped region is greater than 50% 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-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.

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

$$\min_{T^D} \sum_{E \in f C g(i,j) \in DE} \sum_{i \in D} d_i^2(T^D; T^E) + \sum_{i \in D} e_i^2(T^D; T^E) + \sum_{i \in D} f_i^2(T^D) + \sum_{i \in D} g_i^2(T^D; r; h) \quad (9)$$

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

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

Iteration	ICP type	d	e	f	g			
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	d	e	f	g			
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 A . 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 reassemble all reassembled sherds by utilizing ICP algorithm. When C_g is the set of reassembled sherds, and F is the set of its transformation matrix, then by noting notations from (2) and (9), we solve

$$\begin{aligned}
 & \min_{f \in C_g, r; h} \sum_{(E;F) \in C_g} d_{ij}^2(T^E; T^F) \\
 & + \sum_{(E;F) \in C_g} e(e_{ij}^2(T^E; T^F)) + \sum_{(E;F) \in C_g} f(f_i^2(T^F)) \\
 & + \sum_{i \in F} g(g_i^2(T^F; r; h)) \quad (10)
 \end{aligned}$$

where (E, F) denote a pair of sherds in set of sherd pairs C_g , F is the set of edge 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 1.680 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. $A \rightarrow C$ and $B \rightarrow C$ 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 topk and extension 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 r_{max} (and branches b). Increasing the values of r_{max} 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. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1426–1433, 2010. 2

