Cover

A Flexible Approach to Reassembling Thin Artifacts of Unknown Geometry

Geoffrey Oxholm (PhD Candidate)* and Ko Nishino (Associate Professor)

{geoff.oxholm,ko.nishino}@drexel.edu

Drexel University

Department of Computer Science

3141 Chestnut Street

Philadelphia, PA 19104

PHONE: +1-215-895-2669

FAX: +1-215-895-0545

A Flexible Approach to Reassembling Thin Artifacts of Unknown Geometry

Abstract

We present a novel 3D reassembly method for fragmented, thin objects with unknown geometry. Unlike past methods, we do not make any restrictive assumptions about the overall shape of the object, or its painted texture. Our key observation is that regardless of the object's shape, matching fragments will have similar geometry and photometry along and across their adjoining regions. We begin by encoding the scale variability of each fragment's boundary contour in a multi-channel, 2D image representation. Using this multi-channel boundary contour representation, we identify matching sub-contours via 2D partial image registration. We then align the fragments by minimizing the distance between their adjoining regions while simultaneously ensuring geometric continuity across them. The configuration of the fragments as they are incrementally matched and aligned form a graph structure that we use to improve subsequent matches. By detecting cycles in this graph, we identify subsets of fragments with interdependent alignments. We then minimize the error within the subsets to achieve a globally optimal alignment. We leverage user feedback to cull the otherwise exponential search space; after each new match is found and aligned, it is presented to a user for confirmation or rejection. Using ceramic pottery as the driving example, we demonstrate the accuracy and efficiency of our method on six real-world datasets.

Preprint submitted to Journal of Cultural Heritage *February 20, 2012*

Research Aims

Thin objects, such as hollow ceramic sculptures, bowls and vases, play an important role in the study of cultural heritage. Unfortunately, reassembling such objects from their fragments is challenging because the exposed "break-surfaces" are so thin. There are two general strategies used to account for this missing information. *Global* approaches assume that the overall shape of the object is known a priori; this greatly simplifies the process of aligning each piece. *Local* approaches, rely on accurately aligning only two or three pieces at a time, building the object up incrementally. Local methods are more general, since they do not rely on restrictive assumptions about the object's overall shape, but suffer from error accumulation. The goal of this work is to reassemble thin objects without assuming a known global model and without suffering from accumulated error. To do so we present a three step method in which accurate local alignments are first estimated and then continually refined as the full model is uncovered.

1. Introduction

Cultural heritage objects, which are frequently broken when they are discovered, must be reassembled before they may be interpreted. This process is so excessively time consuming that many objects are never reassembled. The goal of virtual reassembly is to not only ease the process of reassembling heritage objects, but also to enable wider access to these objects.

For thick objects, such as thick tile work and sculptures, past methods [1, 2, 3] rely on the "break-surfaces" that have been exposed by the fractures. This information is valuable because matching fragments of such objects will have oppositely shaped break-surfaces. Sets of geometric features in these regions (such as sharp points, or valleys) may be compared to determine if two fragments match each other. Once two fragments are determined to match, the entire set of break-surface points can then be used to accurately align the fragments. For thin objects, both the matching and aligning problems become much more challenging. Instead of

sharing oppositely shaped break-surfaces, fragments of thin objects share only a minute strip in common.

This thin strip, which is often relatively featureless, is seldom descriptive enough to accurately determine which fragments align with each other. In addition, since this matching region is only a curve, and not a surface, it is insufficient in itself to correctly align two matching fragments. In order to simplify these problems, past methods have relied on restrictive assumptions about the geometry or painted texture of the object.

In this paper, we present our three-step method that reassembles objects using only the fragments' boundary contours with minimal user interaction. Our method exploits the key observation that regardless of the shape, or painted texture of the object, the matching boundary regions of adjoining fragments will be similar, both in geometry and photometry. Figure 1 outlines our approach. (a) First, we preprocess each fragment to encode the scale variability of its boundary contour as a multi-channel 2D image. (b) We then identify matching subcontours using a novel image registration method based on these scale-space boundary contour representations. (c) Next, we estimate the transformation to align the fragments using a least squares formulation that minimizes the distance between the adjoining regions while simultaneously maximizing the resulting geometric continuity across them. (d) The configuration of the fragments as they are incrementally matched and aligned form a graph structure. By identifying cycles in this graph, we detect subsets of fragments whose alignments are dependent on each other. When a cycle is formed, we jointly re-optimize the alignments of the constituent fragments to ensure a globally optimal configuration, and improve subsequent matches.

We use ceramic pottery as the driving example, and validate our method on several recently excavated real-world historic artifacts. The results show that our method allows for accurate reassemblies to be achieved with minimal human interaction, even when many fragments are missing.

2. Related Work

When reassembling an object from its fragmented parts, the basic approach is to iterate between finding matching fragments and bringing them into alignment. Reassembling thin objects is particularly challenging because matching fragments share only their boundaries in common. In other words, their "break-surfaces" are not surfaces at all, but contours. On their own, these contours provide insufficient information to reliably match candidate fragments since many fragments have similar shapes. This lack of information also makes aligning matching fragments challenging. We cannot simply place the two matching boundary contours next to each other since this does not guarantee that the fragment surfaces themselves will reform the original unbroken shape.

To address the ill-posed problems of finding matching fragments and aligning them, a common approach is to restrict the class of objects that may be reassembled by imposing a global model. Some authors [4, 5, 6] assume the object is axially symmetric. The shape of the fragment can then be compared to the profile contour of the object model to help determine its proper location and orientation. This effectively reduces the problem to a sort of curved 2D puzzle. To determine the profile contour of a vase or bowl, the authors also assume that the fragments themselves are able to reliably convey this information. If the fragments are too small, the profile contour may not be estimated. We avoid relying on simple object models, or assuming sufficiently large fragment sizes by exploiting the geometric continuity that must exist across matching fragments.

Even when a global model is used to simplify the reassembly process, methods that rely on pairwise alignments necessarily suffer from error accumulation. This effect can even be seen when the object has only three pieces. Once the first two pieces have been matched, the space for the third piece may already be too small or too large. To address this, some authors [7, 5] delay the alignment phase until clusters of three matching fragments have been found. While

this does improve alignment accuracy, it is still a local optimization and therefore prone to error accumulation.

In range image registration, where pairwise alignment is much more reliable, accumulated error may be evenly dispersed to finalize a reassembly. Once a series of overlapping fragments is found that connects back to itself, Sharp et al. [8] then divide up the resulting gap or overlap over the whole set of fragments. Pulli [9] uses the point correspondences of the pairwise alignments as soft constraints on a final global alignment. In object reassembly, however, the pairwise alignments are not as reliable. In addition to improving the overall reassembly accuracy, each incremental match provides important contextual clues that can be used to improve subsequent matches. By leveraging the context provided by past matches, we determine the placement of relatively nondescript fragments, and ensure global accuracy.

Sağiroğlu and Erçil [10] focus on the fragments' painted texture. If the object is sufficient patterned, they use texture synthesis and inpainting methods to expand each fragment's surface texture. This can be thought of as generating overlapping surfaces which may then be aligned using standard texture matching approaches. We also use the texture painted onto the surface, but only as part of our representation of the boundary contour and do not assume any a priori knowledge of the pattern or characteristics of the texture.

Some authors focus specifically on the problem of finding matching boundary contours of thin fragments using only their 3D geometry. Although the contours are three dimensional, Wolfson [11] showed that finding matching sub-contours can be recast as a one-dimensional string matching problem. This dimensionality reduction is possible because contours can be uniquely expressed in terms of their rotationally and translationally invariant geometric characteristics: curvature and torsion. By encoding contours in this way, matching regions may be detected as matching series of curvature and torsion tuples using the longest common substring algorithm.

Unfortunately, recasting 3D contour matching as 1D string matching is highly sensitive to noise, chipping and other realities since it relies on the calculation of the contour's third derivative. To address this, including scale in the matching process has been quite successful. Large scale detail can be used to estimate matches, while finer scale detail can be used to refine and validate the match. The standard approach to incorporate scale is to incrementally smooth the 3D points of the contour, recalculating the curvature and torsion values at each degree of smoothness [11, 12, 13, 14, 15]. Although incremental smoothing highlights features of prominence and mitigates the effect of noise, smoothing the *geometry* of the contour introduces shapes that are dramatically different from the original. Our approach is more faithful to the notion of scale. By smoothing the geometric *characteristics* of the contour we induce an effect similar to moving away from the object. In addition to this, we incorporate the photometric characteristics of the boundary contour. This additional information helps inform the contour matching process, and is particularly useful when many boundary contours share similar geometry.

3. Boundary Contour Representation

The first step in our method is to identify which fragments are most likely to align, and where their boundaries match. To do so quickly and accurately, we leverage the scale variability of each fragment's boundary contour. A coarse scale representation, which is robust to noise and subtle detail, may be used to quickly estimate potential matches, while finer scale detail may be used to verify and fine-tune the estimated matches. This graduated relationship naturally lends itself to a hierarchical encoding. To that end, we build a multi-channel image representation that encodes the scale variability of each fragment's shape and color.

3.1. Boundary Extraction

As shown in Figure 2, we start by acquiring geometric and photometric information about each fragment using a range sensor; specifically a light-stripe range scanner Canon VIVID 910. Without loss of generality, we assume the entire surface of each fragment may be viewed from a single viewpoint. Although this assumption holds true for the various objects in our datasets, if it were not the case, multiple scans could be used to form the exterior surface of the fragment. Each scan is comprised of a 640×480 color image (Figure 2a), as well as three-dimensional coordinates for each of the image points (represented in Figure 2b as a depth map). In most cases we use depth and color discontinuities to automatically isolate the fragment in the scan. The result is a 2D mask like the one shown in Figure 2c. Occasionally, part of the thin break-surface is visible and manual intervention is required to correct the mask.

Using Suzuki's border following method [16], we extract the 2D border of the mask (shown in Figure 2d). We then extract the 3D location of each point along the border from the range data. This contour (shown in Figure 2e) is noisy and its sampling is too dense to allow for reliable analysis. To address this, we smooth the contour slightly and sub-sample the points using an iterative technique described by Leitao and Stolfi [13] which ensures that the points along the resulting contour are uniformly spaced. The processed boundary contour (Figure 2f) exhibits a necessary degree of smoothness to reliably evaluate while remaining true to the original shape. Once the geometry of the 3D contour has been processed, we project the contour back to the image plane. Here we extract the color of each contour point using bilinear interpolation.

3.2. Boundary Representation

We describe each fragment's boundary contour as a cyclic string $f(t)$ of four-valued feature vectors

$$f(t) = \left\{ (\kappa, \tau, c_r, c_g)_{t \bmod n} \right\} , \tag{1}$$

where n is the number of samples along the contour and t is the sample index. The first two values, κ and τ, are curvature and torsion. These two values encode the rotationally and translationally invariant geometry of the 3D contour. The second two, c_r and c_g, are the red and green chromaticity; they encode the appearance of the contour. These values help locate and refine matches, particularly when the contour geometry is relatively featureless. We use chromaticity because it provides a measure of invariance to illumination conditions, which is an important consideration in many domains. Also, observe that the string $f(\cdot)$ is periodic, i.e., $f(t+n) = f(t)$. This captures the cyclic nature of the contour.

The curvature and torsion (κ and τ) of a 3D contour $\boldsymbol{\lambda}(t) = (x(t), y(t), z(t))$ can be defined in terms of the derivatives of the three coordinates [17] as,

$$\kappa = \frac{\|\ddot{\boldsymbol{\lambda}} \times \dot{\boldsymbol{\lambda}}\|}{\|\dot{\boldsymbol{\lambda}}\|^3} = \frac{\sqrt{A^2 + B^2 + C^2}}{(\dot{\boldsymbol{x}}^2 + \dot{\boldsymbol{y}}^2 + \dot{\boldsymbol{z}}^2)^{3/2}} \tag{2}$$

and

$$\tau = \frac{(\dot{\boldsymbol{\lambda}} \times \ddot{\boldsymbol{\lambda}}) \cdot \dddot{\boldsymbol{\lambda}}}{\|\dot{\boldsymbol{\lambda}} \times \ddot{\boldsymbol{\lambda}}\|^2} = \frac{\begin{vmatrix} \dot{\boldsymbol{x}} & \dot{\boldsymbol{y}} & \dot{\boldsymbol{z}} \\ \ddot{\boldsymbol{x}} & \ddot{\boldsymbol{y}} & \ddot{\boldsymbol{z}} \\ \dddot{\boldsymbol{x}} & \dddot{\boldsymbol{y}} & \dddot{\boldsymbol{z}} \end{vmatrix}}{A^2 + B^2 + C^2}\,, \tag{3}$$

where

$$A = \begin{vmatrix} \dot{\boldsymbol{y}} & \dot{\boldsymbol{z}} \\ \ddot{\boldsymbol{y}} & \ddot{\boldsymbol{z}} \end{vmatrix}\,,\; B = \begin{vmatrix} \dot{\boldsymbol{z}} & \dot{\boldsymbol{x}} \\ \ddot{\boldsymbol{z}} & \ddot{\boldsymbol{x}} \end{vmatrix}\,,\; C = \begin{vmatrix} \dot{\boldsymbol{x}} & \dot{\boldsymbol{y}} \\ \ddot{\boldsymbol{x}} & \ddot{\boldsymbol{y}} \end{vmatrix}\,. \tag{4}$$

Here $\|\square\|$ and $|\square|$ denote the $L2$ norm and matrix determinant, respectively. $\dot{\square}$, $\ddot{\square}$, and $\dddot{\square}$ denote the first-, second-, and third-order derivatives with respect to the arc length t, i.e., $\frac{\partial}{\partial t}$, $\frac{\partial}{\partial t^2}$ and $\frac{\partial}{\partial t^3}$. We compute these using central numerical differentiation.

These geometric values are used frequently in shape description applications because they provide a unique, rotationally and translationally invariant representation of the contour. We

extend the descriptive formulation beyond past work by additionally encoding the photometric properties of the contour. After white-balancing the color image, we then encode the color of each contour point using its red and green chromaticity values (c_r and c_g respectively). These are computed by simply dividing the color channels by the total intensity of the point. Note that including the blue channel would be redundant since the three values sum to one.

3.3. Encoding Scale

Although this string of values $f(t)$ accurately describes the boundary contour, it encodes small-scale detail and noise that may not precisely align with the matching sub-contour description of an adjoining fragment. In order to reliably locate matching boundary sub-contours, we exploit the scale variability of their geometry and photometry. We use the coarse scale representation, which is robust to noise and subtle detail, to quickly estimate potential matches, and the finer scale detail to verify and fine-tune them.

Past authors [11, 12, 13, 14, 15] typically encode scale by incrementally smoothing the geometry of the contours themselves. This approach, however, introduces shapes that are dramatically different from the original. In order to maintain the authenticity of the underlying geometry and photometry of the contour our approach is to iteratively smooth the string $f(t)$ itself. We then store the smoothed values as rows of a multi-channel image $\mathbf{S}$, like the one shown in Figure 3. For clarity, we have shown each channel separately. Left-to-right, top-to-bottom they are curvature, torsion, red and green chromaticity. The lowest row of pixels $\mathbf{S}_0$ encodes the smallest scale detail, i.e., $\boldsymbol{S}_0 = f(t)$. Moving up the image to row r corresponds to a coarser scale. Each increasing scale is calculated using circular convolution of the base row with a Gaussian smoothing kernel $\mathcal{N}(\sigma_r)$ of standard deviation σ_r proportional to r,

$$\boldsymbol{S}_r = \mathcal{N}(\sigma_r) \circledast \boldsymbol{S}_0 \,, \tag{5}$$

where $\circledast$ is the circular convolution operator (convolution with periodic a boundary). In our

case, we let $\sigma_r = 0.05r$ and build images with 100 rows. Increasing the number of rows, and consequently the maximum value for σ, will increase the number of false positives as discerning detail becomes smoothed away. Decreasing the number of rows, on the other hand, will lead to false negatives and ultimately, incomplete reassemblies. We therefore chose this value empirically to minimize the false positives without allowing for many false negatives.

The resulting representation is now a 4-channel 2D image that encodes the scale variability of the boundary contour's geometry and photometry. In Figure 3, observe how the five sharp corners of the fragment (e) have become bright regions in the curvature channel (a), and how the banded painting yields a similarly striped pattern in the red chromaticity channel (c). Note that as torsion (b) is a signed value, in this channel, a value of 0 is given an intensity of 0.5. Although the image has four channels, we use a more compact rendering throughout this paper. As shown in (f), we replace the two chromaticity channels with a single channel computed as the intensity. This channel is colored blue; curvature is shown in red, and torsion is green.

This 2D formulation is particularly appealing because it describes the photometric and geometric properties of the contour under all scales at once. It is this characteristic that we leverage when solving the matching problem.

4. Matching Boundary Contours

By encoding the scale variability of each boundary contour's shape and color as a multi-channel image, the problem of finding matching sub-contours is now akin to partial image registration. Specifically, matching image regions will correspond to matching sub-contours. Note that because both contours are encoded in counterclockwise ordering, one image must first be horizontally flipped. Although chromaticity has a standard range of $[0, 1]$, curvature and torsion are unbounded. To introduce a degree of comparability across all channels, we limit each range by scaling it. Figure 4 outlines our three step matching method.

a) Longest common sub-contour Similar to past work [14, 15, 7], we begin our search with the coarsest scale, i.e., the top row. Here we perform a longest common substring analysis with two modifications. First, a pair of four-valued feature vectors $r_0^{\boldsymbol{A}}(p)$ and $r_0^{\boldsymbol{B}}(q)$ is considered to be matching if each of their values differ by less than a threshold. Secondly, we allow matches to wrap around the images. The table produced by the dynamic programming algorithm is then traversed, and matches are inserted into a queue where priority is given based on the length of the match. Figure 4a, shows the top match for the two fragments shown.

b) 2D image registration refinement Since this coarse scale involves the most smoothing, there remains some ambiguity about the precise location of the matching sub-contour. Just as archaeologists do by hand, we leverage finer-scale detail to validate the match. Using one region as a template, we refine the location of the matching region via normalized cross-correlation image registration. In Figure 4b we show the result of this step which resembles the "lock-in" effect archaeologists use to describe the certainty that comes from aligning small scale features. Note that the matching region of the smaller piece has moved right in the scale-space representation and clockwise around the piece.

c) Fan-out estimation The first and last points of every correctly matching contour sit next to a non-matching point. In Figure 4, for example, the match stops when the two contours turn sharply away from each other. This results in contrasting values in the two contour description strings encoded at the base of the scale-space representations. As we increase the scale, and move up the scale-space representations, these non-matching values are smoothed with an increasing number of matching values (as the standard deviation of the smoothing function σ_r increases). The result of this smoothing is an iterative decreasing in the number of matching values. To address this effect, which we call *fan-out*, we incrementally expand the base of the matching regions until a threshold is surpassed. As shown in Figure 4c, the resulting trapezoidal regions correctly convey the full match.

5. Pairwise Alignment

Once two matching boundary regions have been identified, the second step of our algorithm is to estimate the transformation that brings the fragments into alignment. We formulate this as a least-squares optimization problem where the error is measured at each sampled point of the matching boundary contours according to two metrics. The first, which we call "contour error," quantifies the distance between the corresponding points. This ensures that the fragments are tightly aligned. The second, which we call "surface error," quantifies the geometric continuity across the points. This ensures that the resulting surface geometry transitions smoothly from one fragment to the other.

More precisely, given two fragments $\mathbf{A}$ and $\mathbf{B}$, we seek the transformation $T_{\mathbf{B},\mathbf{A}}$ that brings $\mathbf{B}$ into alignment with $\mathbf{A}$ such that the sum of squared residuals $\sum_i^m e_i^2$ across the m points of their matching boundary contours is minimized. We formulate the residual at point i as a weighted combination of our two metrics,

$$e_i = e_i^{\mathrm{c}} + \alpha e_i^{\mathrm{s}} \,, \tag{6}$$

where e_i^{c} and e_i^{s} are the contour error and surface error, respectively, and α is a relative weight.

Contour error

To formulate the contour error e^{c}, we let $\{\boldsymbol{a}_i \mid 1 \leq i \leq m\}$ be the m 3D contour points from $\mathbf{A}$, and let $\{\boldsymbol{b}_i \mid 1 \leq i \leq m\}$ be the corresponding points from $\mathbf{B}$. The contour error e_i^{c} at point i is then the Euclidean distance between corresponding points,

$$e_i^{\mathrm{c}} = \|T_{\mathbf{B},\mathbf{A}}(\boldsymbol{b}_i) - \boldsymbol{a}_i\| \,. \tag{7}$$

The transformation $T_{\mathbf{B},\mathbf{A}}$ is defined in terms of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$,

$$T_{\mathbf{B},\mathbf{A}}(\boldsymbol{b}_i) = \boldsymbol{b}_i\mathbf{R} + \mathbf{t} \,. \tag{8}$$

This error metric is insufficient on its own because the contours do not convey any reliable information about the fragment surfaces themselves; the alignment should not only be tight, but should result in a smooth surface.

Surface error

Our second error term evaluates the alignment quality of the fragment surfaces. We evaluate the continuity of the newly-formed surface in terms of the progression of surface-normal vectors across the matching boundary contour points. In Figure 5 we illustrate how the surface error e_i^{s} is computed for a point i. We begin by computing the plane P_i that is orthogonal to the boundary contour's tangent vector at this point. The intersection of this plane with both surfaces forms a curve (shown in red and blue). Along this curve, we extract surface normals from the fragment surfaces at regular intervals. Specifically, we let $\widehat{\boldsymbol{a}}_i(j)$ denote the j^{th} surface normal from point $\boldsymbol{a}_i$, and define $\widehat{\boldsymbol{b}}_i(j)$ analogously. We then form a piecewise function $\boldsymbol{q}_i(k)$ of these normal vectors as

$$\boldsymbol{q}_i(k) = \begin{cases} \widehat{\boldsymbol{a}}_i(-k) & \text{for } k < 0 \\ \mathbf{0} & \text{for } k = 0 \\ T_{\mathbf{B},\mathbf{A}}(\widehat{\boldsymbol{b}}_i(k)) & \text{for } k > 0\ , \end{cases} \tag{9}$$

where $T_{\mathbf{B},\mathbf{A}}$ is the alignment transformation to be optimized. Note that we negate k in the first case since $\widehat{\boldsymbol{a}}(\cdot)$ is only defined for positive inputs.

An optimal alignment will result in a gradient of surface normals $\frac{\partial \boldsymbol{q}_i}{\partial k}$ whose value at the boundary (i.e., $k = 0$) matches the gradient of surface normals on the adjoining fragments. We therefore compute a target gradient value equal to the mean of the gradients from both fragments evaluated a short distance ϵ from the boundary contour. The surface alignment residual

error is then computed as

$$e_i^{\mathrm{s}} = \frac{\partial \boldsymbol{q}_i}{\partial k}(0) - \frac{1}{2}\left[\frac{\partial \widehat{\boldsymbol{a}}_i}{\partial j}(\epsilon) + \frac{\partial \widehat{\boldsymbol{b}}_i}{\partial j}(\epsilon)\right] . \tag{10}$$

To compute the derivative of $\boldsymbol{q}$ at point i, we extract fifteen normals $\{\boldsymbol{q}_i(k) \mid -7 \leq k \leq 7\}$ from a thin strip of the corresponding surfaces. By restricting this calculation to a thin strip we avoid any assumptions about the global geometry of the object. We then weight these 15 normals using a standard discrete derivative kernel. For the target gradients $\partial\widehat{\boldsymbol{a}}_i$ and $\partial\widehat{\boldsymbol{b}}_i$, we use the same process and evaluate the gradient at the midway point $\epsilon = 4$. We then solve this system with Levenberg-Marquardt iterative minimization [18, 19].

To provide a measure of similarity between the two error terms e^{c} and e^{s}, we set the relative weighting term α (from Equation 6) equal to the boundary contours' sub-sampling distance divided by π (the maximum value for e^{c}). To reduce the running time of this optimization, we pre-compute the intersection planes, surface normals, and target gradients for each contour point. By doing so, however, we are making the assumption that the intersection plane computed for fragment $\boldsymbol{A}$ at point i is parallel to the intersection plane computed for fragment $\boldsymbol{B}$. This assumption may be violated, in particular when alignments are re-optimized, as we discuss next. To address this, we add an additional error term that quantifies the orientation error of these planes. Since the range of this error is the same as for e^{s} we also weight it with α.

6. Incorporating Context

As illustrated in Figure 6, the final component of our framework is to incorporate the global geometry of the object as it becomes known. We use each additional match to increase the overall alignment quality of the object, which in turn improves the quality of future matches. We encode the global geometry as a graph in which the nodes represent individual fragments, and the edges correspond to matching contours between them. Cycles in this graph correspond

to subsets of fragments whose alignments depend on each other, i.e. changing the orientation of one fragment in such a subset will impact the alignment quality of the others.

Figure 6 shows the significance of a cycle in the graph. In the four node chain of Figure 6a there is a gap between fragments $\boldsymbol{B}$ and $\boldsymbol{C}$ that has resulted from the accumulation of small alignment errors in the three edges. In Figure 6b, the addition of the edge connecting fragments $\boldsymbol{B}$ and $\boldsymbol{C}$ has created a cycle. It is plain to see that aligning the two fragments using only this newly found matching contour (marked in yellow) will decrease the quality of the neighboring alignments. It would also be inaccurate to simply move fragment $\boldsymbol{B}$ evenly between its neighbors. Instead, this additional edge provides important context information that should be used to improve all of the other alignments.

We detect cycles by computing all of the paths between the fragments involved in the new match. This is done simply with a single depth first traversal of the graph. Using the correspondences embedded in every edge of the cycle to evaluate the overall error in the subset, we jointly re-optimize the constituent alignments. Note how the gap between $\boldsymbol{B}$ and $\boldsymbol{C}$ has been closed by this process, and the overall alignment quality of the subset has been improved.

We formulate the update alignment of the subset as a least squares optimization where the residual error is measured across all points of every matched contour pair according to Equation 6. All alignment translations and rotations are optimized jointly to minimize the error of the entire subset. As the number of cyclically dependent nodes approaches the entire set of fragments, this re-optimization becomes a full global optimization. By building up the global model in a bottom-up fashion, however, each re-optimization is simply a refinement of the previous configuration. Thus, we leverage the accurate estimates of our pairwise alignments to reduce the search-space of the global optimization.

After updating the alignment of the subset, the set of points being used to evaluate an alignment may no longer be an optimal set of correspondences. Recall that the alignment quality

across a pair of matching contours depends on the distance between pairs of corresponding points. When a cycle is formed, we may find that one fragment should slide along the other to improve the overall alignment quality. To that end we utilize the core functionality of Besl and McKay's iterative closest point (ICP) algorithm [20]. Specifically, we iterate between optimizing the transformations of the subset, and updating the correspondences used to evaluate them. At each iteration every point is paired with the nearest point on the corresponding contour. Once the correspondences become stable, the optimal alignment for the object will be found.

7. Grouping Fragments

The final aspect of our method is designed to aid in the detection of matching contour groups such as the one shown in Figure 7. When two or more pieces are bound together by matching sub-contours, they effectively form a single *meta-piece*. Intuitively, this corresponds to virtually gluing pieces together–the matched sub-contours become internal to the meta-piece. Each connected component subgraph of our alignment graph is given its own scale-space boundary representation which is then compared against the remaining unpaired fragments and meta-pices.

The boundary representation for a meta-piece is formed by first extracting the unmatched boundary points from all constituent pieces. These points are concatenated to from a single boundary contour. As shown in Figure 7, there are two important factors to consider during this process. First, the matching boundary contour between the two fragments may have been too short. This will result in small sub-contours that essentially overlap each other as in Figure 7b. To account for this, we extend the boundary contours as shown in Figure 7c. The second factor to consider comes from our preprocessing step. Recall that each fragment's raw boundary contour is initially smoothed and sub-sampled, causing a slight rounding of sharp corners. As shown in Figure 7c, the meeting point of two fragments must be smoothed to account for this.

This approach ensures that contour regions that have previously been matched are no longer considered for future pairings. Additionally, the boundary contour of a meta-piece will typically have a more favorable representation. In Figure 7d, the third fragment is successfully paired with both of the other fragments. Since we prioritize candidate matches based on their length, the small matching contour that makes up the bottom of this match would not otherwise have been found in a reasonable time-frame. Further examples can be seen in the final steps of Figure 8, and in the begging of the third row of Figure 12.

8. Experimental Results

To validate our algorithm we augmented it with an simple, interactive user interface. At each iteration of our algorithm (summarized in Figure 1) we first extract the longest matching boundary contour. We then align the constituent fragments and present the updated reassembly to the user who may then accept or reject it. When a match is accepted, it is added to the reassembly, and the additional context information is propagated to the other alignments. Except where indicated, the resulting reassemblies contain every fragment in the dataset. Each dataset is incomplete, describing only a partial vessel. Despite this, we are able to successfully recover the full 3D geometry of every dataset. Unfortunately, there is no ground truth to which we could compare the exact alignment of the fragments.

In Figure 8 we show the partial reassembly of a store bought vase. At each step, the current candidate match is shown in green, other matches that have been re-optimized are colored in purple, and any other past matches are shown in red. In this artificial example, each of the suggested matches is correct making the full reassembly essentially fully automatic. As such, the full process requires only about 12 seconds.

In Figure 9 we summarize our results and compare total reassembly times using our method with manual reassembly of the actual artifacts. In this graph, the store bought vase of Figure 8

referred to as VASE. The remaining five objects are recently excavated cultural heritage artifacts, and as such exhibit increased chipping, signs of wear, and a greater range of fragment size. Although these datasets describe axially symmetric objects, the fragments themselves are quite small. Consequently, past methods that rely on estimating the profile contour of symmetric objects [4, 5] are not be well suited for this task.

In Figure 10 we show the steps taken by a user to reassemble 15 pieces of one such historical artifact. Note, however, that we have omitted a small triangular piece in the top of the rim. This omission is visible in the final reassembly (shown in a green box) as compared to the ground truth image (shown in an orange box). This piece was too small to be reliably analyzed, due to the fan-out effect discussed in Section 4. This example is referred to as BOWL1 in Figure 9.

In Figure 11 we show the reassembly of an 11 piece historical artifact. Note how a large gap is closed in the last step, as indicated by the yellow circles. Since many pieces are missing on this side of the bowl, a long chain of pieces has accumulated error. When the matching contour is found that closes this gap, the entire rim of the vessel is re-aligned to distribute the error. This example is referred to as BOWL2 in Figure 9.

Figure 12 shows our most challenging dataset. Whereas the painted scene provides useful clues to archaeologists working with the artifact itself, the color information contained at the boundary of each piece is too subtle to be discerning in our scale-space images. Further, there are many small missing fragments that partition long matching contours into smaller matches. Because of these two factors many erroneous matches are considered. We discard clearly erroneous matches by inspecting the resulting alignment error. If too much space exists between matching contours, or the resulting surface is not smooth, the match is discarded. This dramatically reduces the amount of user interaction required in all cases. In this case, however, evaluating alignment error may involve large cycles of fragments. Note that almost every step involves a large scale re-optimization. As shown in Figure 9, this vessel (referred to as

PLATE1) is the only example where assembly by hand took less time than using our method. Nevertheless, despite the challenges in this dataset, our method is able to rapidly align all but one fragment of this object.

The final two datasets, which are respectively labeled PLATE2 and PLATE3 in Figure 9, show the importance of our surface error alignment metric. Many of the matching contours are essentially straight lines. As such, simply minimizing the distance between the contours would result in dramatic surface discontinuity across the matching regions. By additionally optimizing the geometric continuity across the surfaces, the pieces are accurately aligned.

Figure 9 shows the reassembly times for our six datasets using our method (light) and by hand (dark). Although our user interface is rudimentary, it is important to note that with the exception of PLATE1 our system allows for more rapid reassembly than by hand. BOWL2, which is detailed in Figure 11, proved too challenging to reassemble by hand; the time reported is when the test subject abandoned the task. We believe this is due to the object's many similarly shaped fragments. By utilizing the full scale-space of each fragment, we exploit subtle detail to eliminate the guesswork associated with manual reassembly.

9. Conclusion

Our method, which makes no assumptions about the global geometric or photometric structure of the object, is applicable to any reassembly problem where the boundary of each fragment may be extracted. This includes volumetric object reassembly, with the added benefit that a full 3D model of each fragment is not needed. To achieve this, we have presented a method that balances our novel pairwise alignment method with a flexible global approach that leverages the important context information each match provides. By allowing the global model to be uncovered incrementally, we successfully complete full reassemblies even when most of the object's fragments are missing. Since access to cultural heritage objects is greatly restricted,

achieving reassembly of real artifacts in a virtual setting is itself an important step. For future work, we plan to investigate more engaging and efficient user interface designs.

Acknowledgements

This work was supported in part by National Science Foundation CAREER Award IIS-0746717 and IIS-0803670. The authors are grateful to Jed Levine of the National Park Service, and to Patrice Jeppson, Glen Muschio and his students at Drexel University for their time and help with data acquisition.

References

[1] S. Winkelbach, F. M. Wahl, Pairwise Matching of 3D Fragments Using Cluster Trees, International Journal of Computer Vision 78 (2008) 1–13.

[2] B. J. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, T. Weyrich, A System for High-Volume Acquisition and Matching of Fresco Fragments, ACM Transactions on Graphics 27 (2008) 1–9.

[3] Q. Huang, S. Flöry, N. Gelfand, M. Hofer, H. Pottmann, Reassembling Fractured Objects by Geometric Matching, ACM Transactions on Graphics 25 (2006) 569–578.

[4] A. R. Willis, D. B. Cooper, Bayesian Assembly of 3D Axially Symmetric Shapes from Fragments, IEEE Conference on Computer Vision and Pattern Recognition (2004) 82–89.

[5] The SHAPE Lab, Assembling Virtual Pots from 3D Measurements of Their Fragments, Virtual Reality, Archeology, and Cultural Heritage (2001) 241–254.

[6] M. Kampel, R. Sablatnig, On 3D Mosaicing of Rotationally Symmetric Ceramic Fragments, International Conference on Pattern Recognition 2 (2004) 265–268.

[7] K. Weixin, B. Kimia, On Solving 2D and 3D Puzzles Using Curve Matching, IEEE Conference on Computer Vision and Pattern Recognition (2001) 583–590.

[8] G. C. Sharp, S. W. Lee, D. K. Wehe, Multiview Registration of 3D scenes by Minimizing Error Between Coordinate Frames., IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 1037–50.

[9] K. Pulli, Multiview Registration for Large Data Sets, International Conference on 3-D Digital Imaging and Modeling (1999) 160–168.

[10] M. Sağiroğlu, A. Erçil, A Texture Based Approach to Reconstruction of Archaeological Finds, Symposium on Virtual Reality, Archaeology and Cultural Heritage (2005) 1–6.

[11] H. Wolfson, On Curve Matching, IEEE Transactions Pattern Analysis and Machine Intelligence 12 (1990) 483–489.

[12] E. Kishon, T. Hastie, H. Wolfson, 3-D Curve Matching Using Splines, European Conference on Computer Vision 427 (1990) 589–591.

[13] H. Gama Leitao, J. Stolfi, A Multiscale Method for the Reassembly of Two-Dimensional Fragmented Objects, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 1239–1251.

[14] F. Mokhtarian, A. Mackworth, Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes, IEEE Transactions Pattern Analysis and Machine Intelligence (1986) 34–43.

[15] F. Mokhtarian, Multi-scale Description of Space Curves and Three-Dimensional Objects, IEEE Conference on Computer Vision and Pattern Recognition (1988) 298–303.

[16] S. Suzuki, K. Be, Topological Structural Analysis of Digitized Binary Images by Border Following, Computer Vision, Graphics, and Image Processing 30 (1985) 32–46.

[17] D. Struik, Lectures on Classical Differential Geometry, Addison-Wesley, Cambridge, Mass., 1950.

[18] K. Levenberg, A Method for the Solution of Certain Non-linear Problems in Least-Squares, Quarterly of Applied Mathematics 2 (1944) 164–168.

[19] D. W. Marquardt, An Algorithm for Least-Squares Estimation of Nonlinear Parameters, SIAM Journal on Applied Mathematics 11 (1963) 431–441.

[20] P. Besl, H. McKay, A Method for Registration of 3-D Shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992) 239–256.

Captions for figures

Figure 1

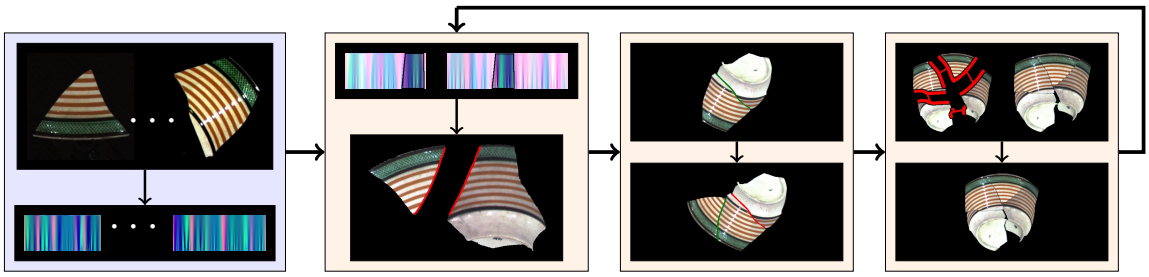


Figure 2

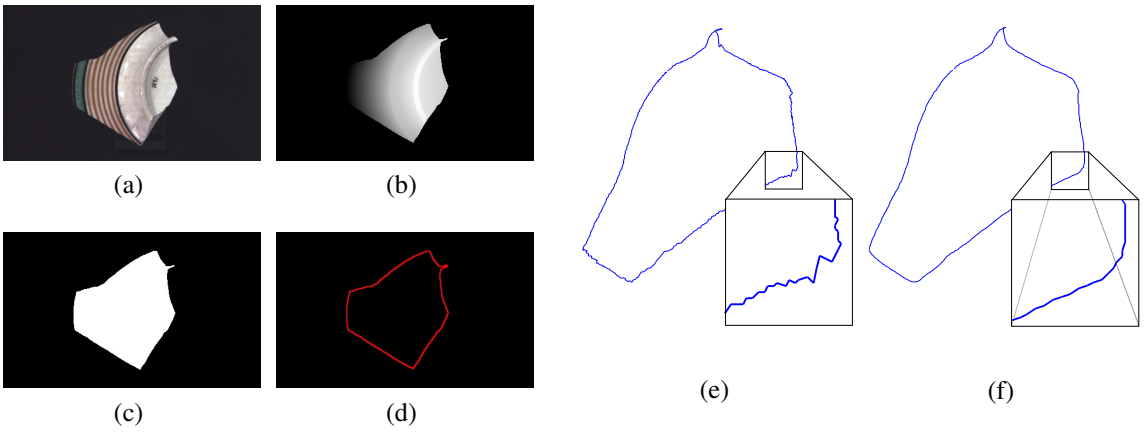


Figure 3

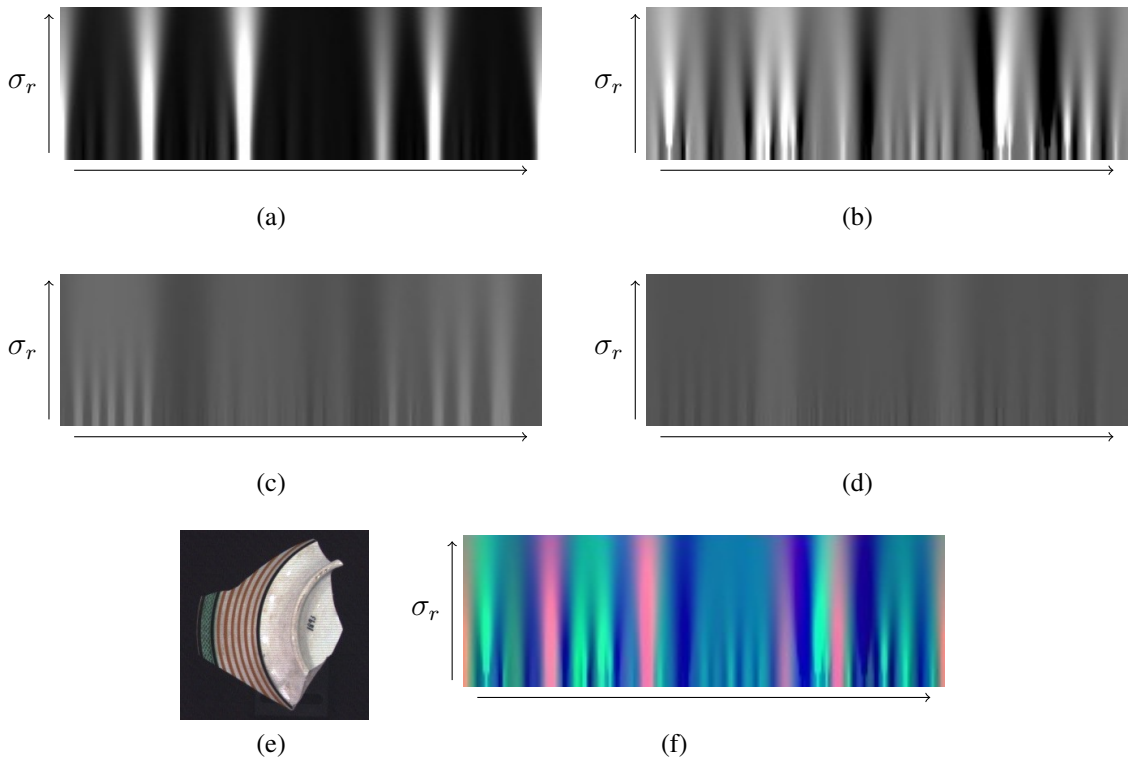
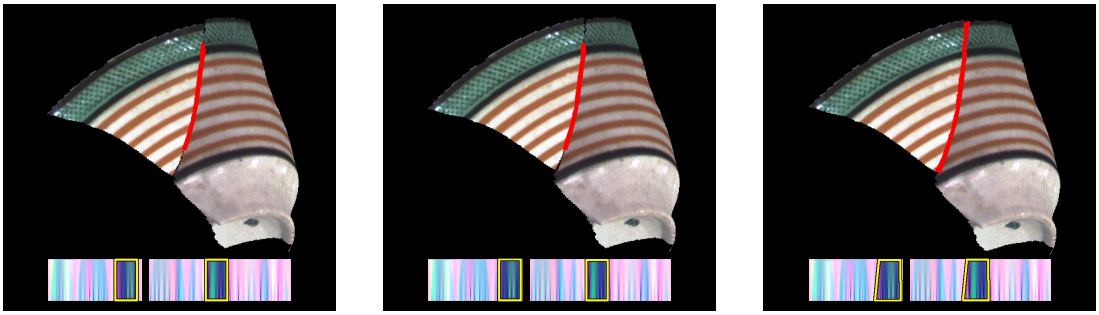


Figure 4



(a) (b) (c)

Figure 5

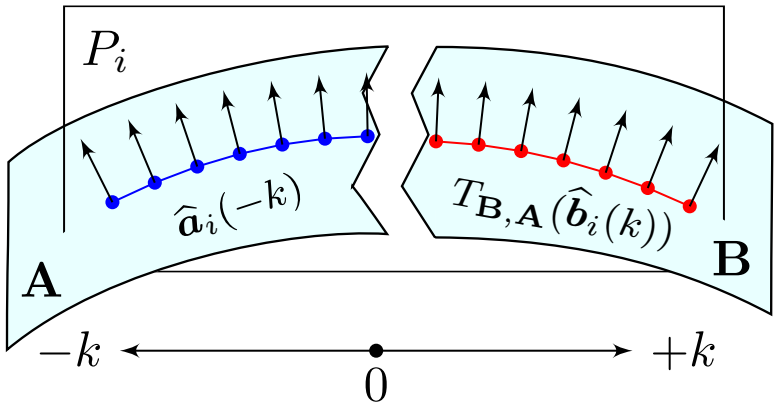
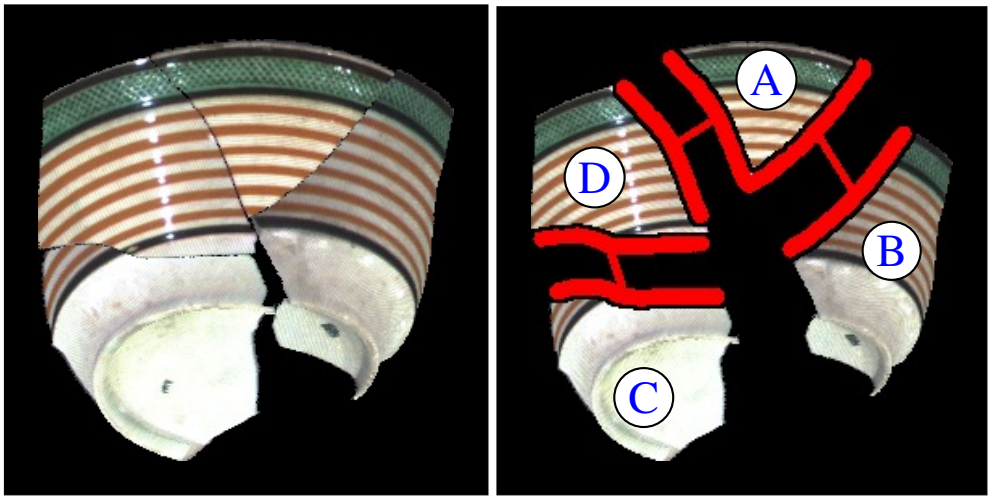
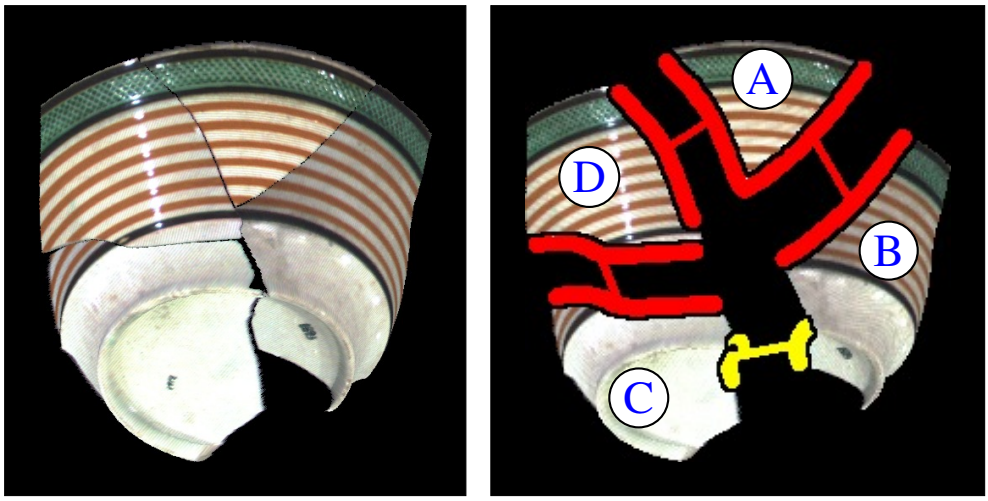

P_i
$\widehat{\mathbf{a}}_i(-k)$
$T_{\mathbf{B},\mathbf{A}}(\widehat{\boldsymbol{b}}_i(k))$
$\mathbf{A}$
$\mathbf{B}$
$-k$
0
$+k$

Figure 6

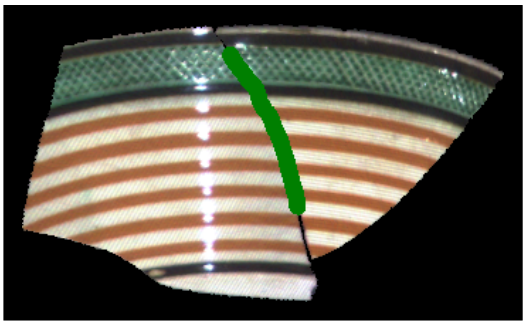


(a)

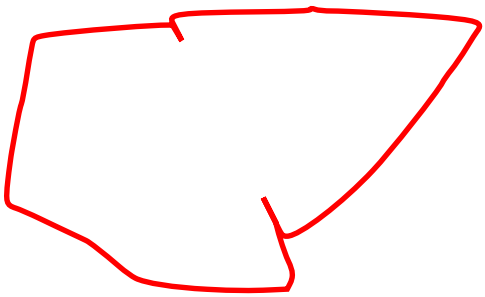


(b)

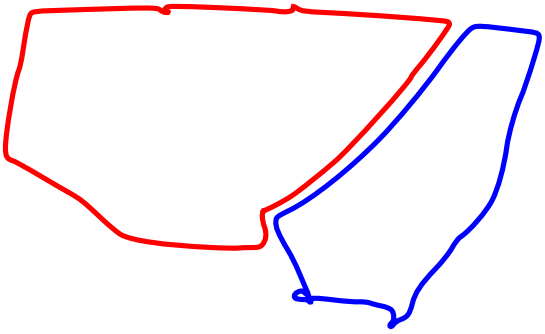
Figure 7



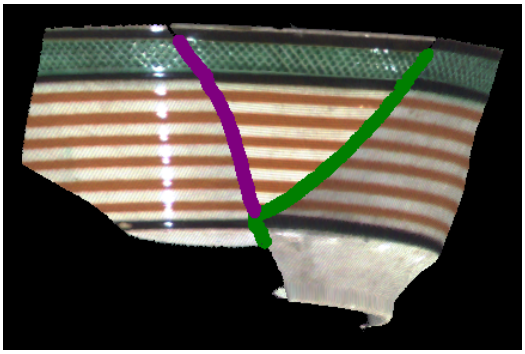
(a)



(b)



(c)



(d)

Figure 8

Figure 9

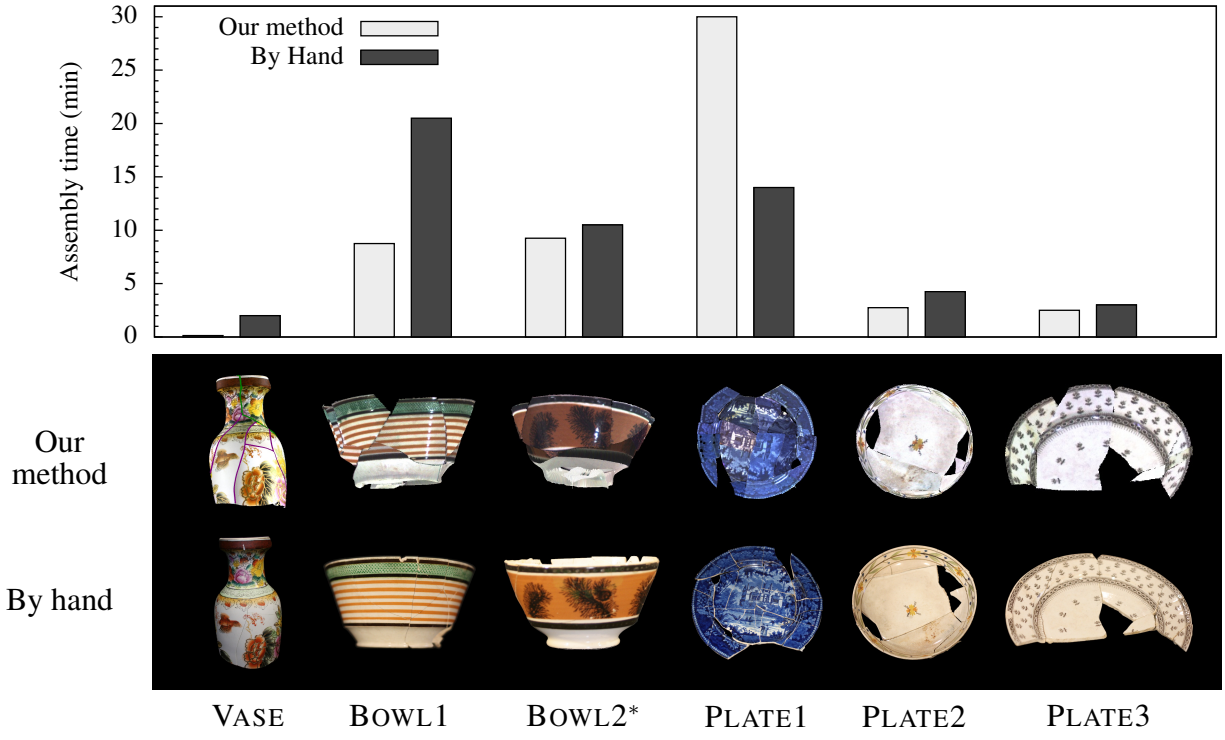

Assembly time (min)
30
25
20
15
10
5
0
Our method
By Hand
Our method
By hand
VASE
BOWL1
BOWL2*
PLATE1
PLATE2
PLATE3

Figure 10

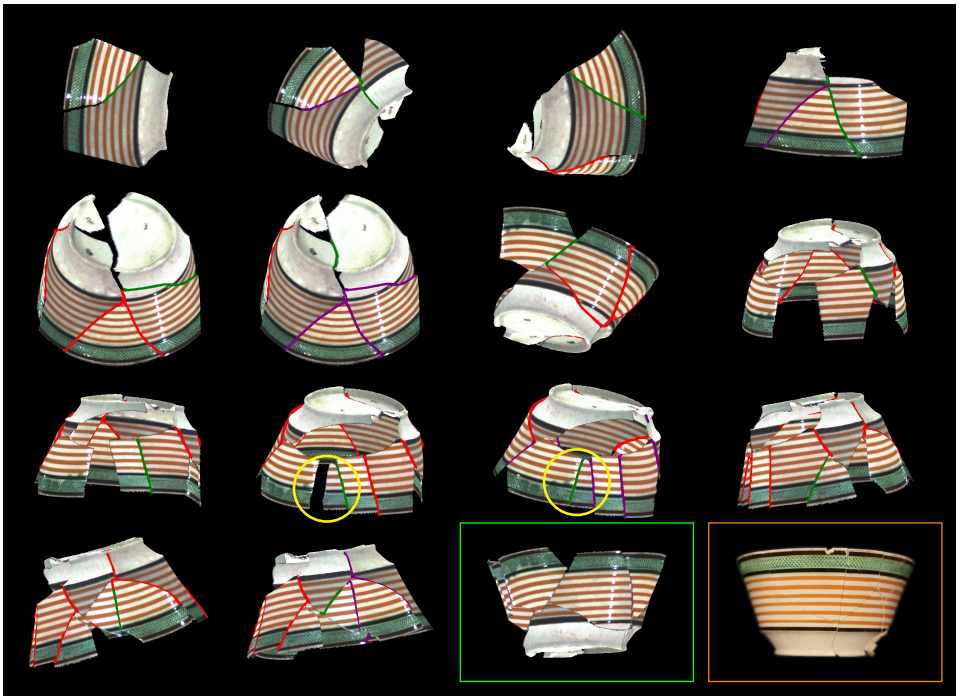


Figure 11

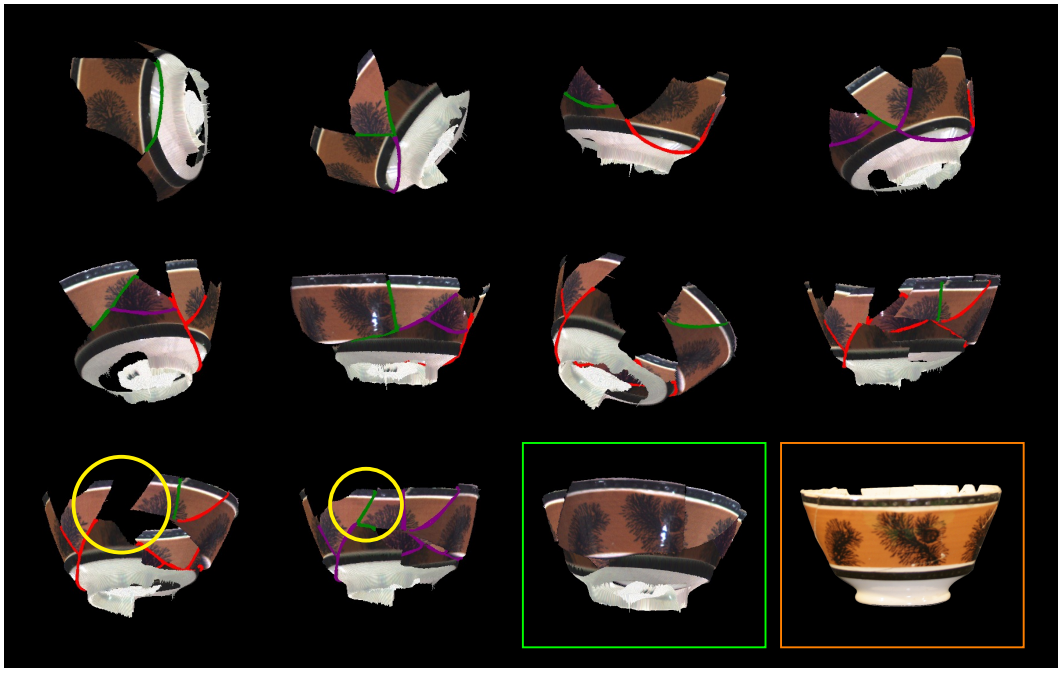


Figure 12