The 12th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST (2011)
M. Dellepiane, F. Niccolucci, S. Pena Serna, H. Rushmeier, and L. Van Gool (Editors)

Reassembling Thin Artifacts of Unknown Geometry

G. Oxholm and K. Nishino

Drexel University, Philadelphia PA

Abstract

We introduce a novel reassembly method for fragmented, thin objects that uses minimal user interaction. Unlike past methods, we do not make any restrictive assumptions about the geometry or texture of the object. To do so, we exploit the geometric and photometric similarity along and across the boundaries of matching fragments, and leverage user feedback to tackle the otherwise ill-posed problem. We begin by encoding the scale variability of each fragment's boundary contour in a multi-channel, 2D representation. Using this multi-channel boundary contour representation, we identify matching sub-contours via 2D partial image alignment. 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. By detecting cycles in this graph, we identify subsets of fragments with dependent alignments. We then minimize the error within the subsets to achieve a globally optimal alignment. Using ceramic pottery as the driving example, we demonstrate the accuracy and efficiency of our method on six real-world datasets.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations I.4.9 [Image Processing and Computer Vision]: Applications—Reassembly

1. Introduction

In order to recover the original 3D shape of a fragmented object, matching fragments must be found and aligned with each other. When reassembling a volumetric object from full 3D models of its parts, for example, the points that make up the "break-surface" between two fragments are used to identify the matching fragments, and to bring them into alignment [HFG*06]. Similarly, in conventional range image registration, shared surface information may be used to match and align overlapping scans. An important, but less studied case, is when the fragments do not share any matching surface.

This problem, which is common in archeology, forensics and paleontology, presents a unique set of challenges. Without knowing the final shape of the object, the only way to match and align corresponding fragments is to inspect the fragments' boundary contours. Regardless of the shape, or painted texture of the object, the boundaries, or "break-contours," of matching fragments will be similar, both in geometry and photometetry. Since this thin strip conveys very little information about the global context of the fragment, however, identifying matching fragments, and aligning them are both challenging problems. In order to simplify these problems, past methods have relied on restricting assumptions about the geometry or painted texture of the object.

In this paper, we introduce a three-step method that reassembles objects using only the fragments' boundary contours with minimal user interaction. Fig. 1 outlines our approach. First, we encode the scale-space of each fragment's boundary contour in a multi-channel 2D image representation. We derive an image registration method based on this scale-space boundary contour representation to rapidly identify matching sub-contours. 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. 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 val-

DOI: 10.2312/VAST/VAST11/049-056

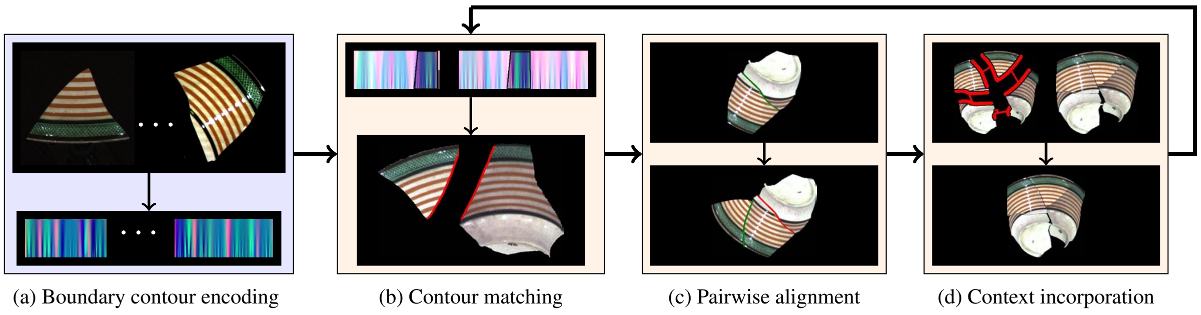


Figure 1: We introduce a three-step method to reassemble real-world objects of arbitrary shape using only the boundary of each fragment's external surface. 1) Using our novel scale-space representation (a), we quickly identify matching boundary regions (b). 2) We then align the fragments by optimizing the geometric continuity across them(c). 3) Finally, we incorporate the additional contextual information each match provides to ensure global accuracy (d).

idate 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. This has significant implications for archeology where manual reassembly is an excessively time consuming task that hampers historical interpretation and dissemination of archaeological studies.

2. Related Work

Reassembling thin objects is particularly challenging because matching fragments share only their boundaries in common. On its own, this thin strip provides insufficient information to reliably validate potential matches and align them. To address this, a common approach is to restrict the class of objects that may be reassembled by imposing a global model. Willis and Cooper [WC04, The01], for example, 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. To determine the profile contour of the vase or bowl, the authors additionally assume that the fragments themselves are able to reliably convey this information. We avoid such limitations by leveraging the similarity matching fragments must exhibit along and across their boundaries.

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 be seen even 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 [WK01, The01] delay the alignment phase until clusters of three matching fragments have been found. While this does improve alignment quality, 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. Sharp et al. [SLW04], for example, divide up the resulting gap or overlap and evenly disperse the required alignment adjustment. Pulli [Pul99] uses the point correspondences of the pairwise alignments as soft constraints on a final global alignment. In object reassembly, however, each additional match provides important contextual clues that can be used, not only to increase overall accuracy, but also to improve subsequent matches. By leveraging the context provided by past matches, we are able to determine the placement of relatively nondescript fragments without relying on a global model.

Some authors focus specifically on the problem of finding matching boundary contours using only their 3D geometry. Although the contours are three dimensional, Wolfson [Wol90] 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 invariant geometric characteristics: curvature and torsion. By encoding contours in this way, matching regions may now be detected using the longest common substring algorithm.

Unfortunately, this method is highly sensitive to noise and other realities as it relies on the calculation of the contour's third derivative. To address this, including scale in the matching process has been quite successful. The standard approach is to incrementally smooth the 3D points of the contour, recalculating the characteristics at each degree of smoothness [Wol90, KHW90, GS02, MM86, Mok88]. 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. A more faithful approach to modeling scale is to smooth the geometric *characteristics* of the contour. 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.

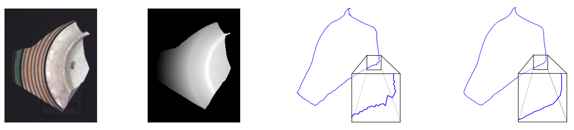


Figure 2: We start with a single scan of each piece. Each scan is comprised of a color image (a) and corresponding 3D point locations shown here as a depth map (b). We use depth discontinuities to locate and extract the 3D boundary of the piece (c), which we then smooth and subsample (d).

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. In order to do so quickly and accurately, 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 Fig. 2, we start by acquiring geometric and photometric information about each fragments using a range sensor. For our datasets, the entire surface of each fragment may be viewed from a single viewpoint. (If this were not the case, multiple viewpoints could be used to form the exterior surface of the fragment.) Using depth discontinuities in the range data we extract the 3D location and color of each point along the boundary. The raw 3D contour (shown in Fig. 2c) is too noisy and its sampling is overly dense to allow for reliable analysis. We therefore smooth the contour slightly and sub-sample the points using an iterative technique described by Leitao and Stolfi [GS02] which ensures that the resulting contour points are uniformly spaced.

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

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

where n is the number of samples along the contour and t is the sample index. The first two values, κ and τ, curvature and torsion, encode the rotationally and translationally invariant geometry of the 3D contour. The second two, c_r and c_g, are the red and green chomaticity that encode the appearance of the contour. These values help locate and refine matches, particularly when the contour geometry is relatively featureless. We choose chromaticity because it provides a measure of invariance to illumination conditions, which is an important consideration in many domains. Note that including the blue channel would be redundant as the three values sum to 1. Finally, observe that $f(\cdot)$ is periodic, i.e., $f(t+n) = f(t)$. This captures the cyclic nature of the contour.

Curvature and torsion may be computed from the contour's coordinates $\boldsymbol{\lambda}(t) = (x(t), y(t), z(t))$ as, [Str50]

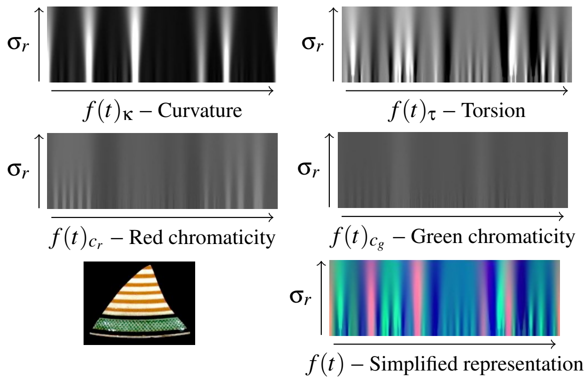


Figure 3: We encode the shape and color of each fragment's boundary contour under various scales. The first two channels (curvature and torsion) encode the shape and the second two (red and green chromaticity) encode the color. Shown at the bottom right is the compact visualization used throughout this paper. Here, the red channel is curvature, green is torsion, and blue is the intensity of each contour point.

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

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

where

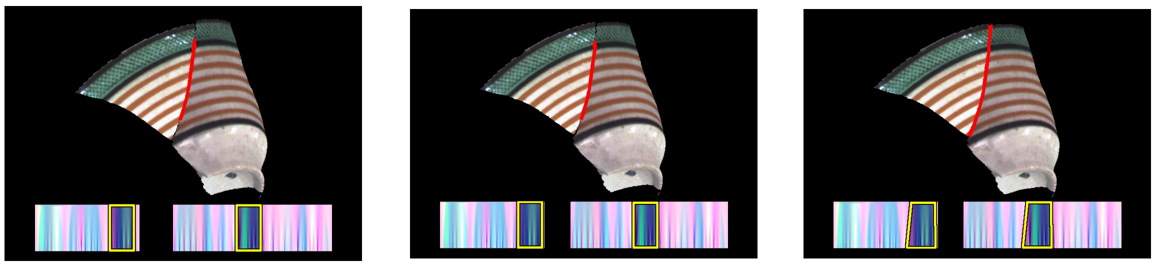
$$A = \begin{vmatrix} \dot{y} & \dot{z} \\ \ddot{y} & \ddot{z} \end{vmatrix},\ B = \begin{vmatrix} \dot{z} & \dot{x} \\ \ddot{z} & \ddot{x} \end{vmatrix},\ C = \begin{vmatrix} \dot{x} & \dot{y} \\ \ddot{x} & \ddot{y} \end{vmatrix}\ . \qquad (4)$$

Here $\|\square\|$ and $|\square|$ denote the $L2$ norm and matrix determinant, respectively; and $\dot{\square}$, $\ddot{\square}$, 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. The chromaticity values c_r and c_g are computed simply by dividing the red and green color channels by the total sum of the three color channels.

3.2. Encoding Scale

Although this string $f(t)$ 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 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 [Wol90, KHW90, GS02, MM86, Mok88] typically encode scale by incrementally smoothing the geometry



(a) We find matching sub-regions using longest-common substring at the coarsest scale. (b) We refine the match using normalized cross-correlation image registration. (c) We expand the base to account for the fan-out of non-matching end-points.

Figure 4: We extract matching boundary contours (red) as matching trapezoidal regions (highlighted boxes) of our scale-space images. Matches are estimated (a), refined (b), and optimized (c) quickly using this scale-space descriptor. Note how the refinement (b) has shifted the matching region right in the scale-space image (clockwise along the boundary), and how the fan-out optimization (c) has expanded the base to the full width of the matching boundary regions.

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 **S**, like the one shown in Fig. 3. Note that as torsion is a signed value, in this channel, a value of 0 is given an intensity of 0.5. 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, which we calculate 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. In our case, we let $\sigma_r = 0.05r$ and build images with 100 rows. The resulting representation is now a four-channel 2D image that encodes the scale-space of the boundary's geometry and photometry.

4. Matching Boundary Contours

By encoding the scale variability of each boundary contour's shape and color, the problem of finding matching sub-contours is now akin to image registration; matching sub-contours correspond to matching image regions. 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. Fig. 4 outlines our matching method.

1) Longest common sub-contour Similar to past work [MM86, Mok88, WK01], we begin our search with the coarsest scale, i.e., the top row. Using these highly smoothed values, we perform a longest-common-substring analysis where we allow for substrings that wrap around the image. A pair of four-valued vectors is considered matching if each their values differ by less than a threshold. The table generated 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. Fig. 4a, shows the top match for the two pieces shown.

2) 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. To address this, we leverage the finer-scaled details contained in the image region below the estimated match. Using one region as a template, we refine the location of the matching region via normalized cross-correlation image registration. This is shown in Fig. 4b. Note that the matching region of the smaller piece has moved right in the scale-space representation and clockwise around the piece.

3) Fan-out estimation The first and last points of every correctly matching contour sit next to a non-matching point. In Fig. 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 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 Fig. 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

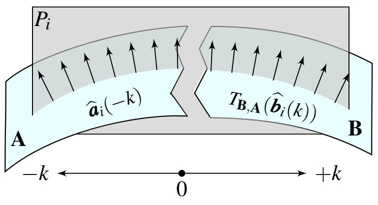

P_i
$\widehat{\boldsymbol{a}}_i(-k)$
$T_{\mathbf{B},\mathbf{A}}(\widehat{\boldsymbol{b}}_i(k))$
A
B
$-k$
0
$+k$

Figure 5: To evaluate the resulting surface continuity at point i, we intersect the two surfaces with a plane perpendicular to the boundary contour. We then extract surface normals at regular intervals. The transformation $T_{\mathbf{B},\mathbf{A}}$ is then evaluated in terms of the gradient of these normals.

boundary contours according to two metrics. The first, which we call "contour error," quantifies the distance between corresponding points along the contour to ensure that the fragments are tightly aligned. The second, which we call "surface error," quantifies the geometric continuity across each pair of corresponding points to ensure that the resulting surface varies smoothly from one fragment to the other.

More precisely, given two fragments **A** and **B**, we seek the transformation $T_{\mathbf{B},\mathbf{A}}$ that brings **B** into alignment with **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 weighting.

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

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

where 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.

Surface error Our second error term evaluates the alignment quality of the fragment surfaces, which we quantify using the orientation of surface-normals across the matching boundary regions of the adjoining surfaces. As shown in Fig. 5, to formulate the surface error e_i^{s} at point i, we begin by computing the plane P_i orthogonal to the boundary contour's tangent vector at this point. We intersect this plane P_i with both surfaces and extract surface normals at regular intervals in the direction perpendicular to the contour. 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 normal vectors across the matching contour as

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

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, and formulate the surface alignment error as

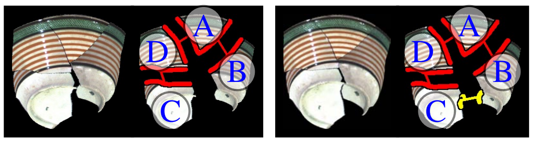
$$e_i^{\mathrm{s}} = \frac{\partial \boldsymbol{q}_i}{\partial k}(0) - \frac{1}{2}\left[\frac{\partial \widehat{\boldsymbol{a}}_i}{\partial j}(\varepsilon) + \frac{\partial \widehat{\boldsymbol{b}}_i}{\partial j}(\varepsilon)\right] \,. \tag{10}$$

To compute the derivative of $\boldsymbol{q}$ at point i, we extract fifteen normals $\{\boldsymbol{q}_i(k) \mid -7 \le k \le 7\}$ 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, we use the same process and evaluate the gradient at $\varepsilon = 4$. We then solve this system with Levenberg-Marquardt minimization [Mar63].

To provide a measure of similarity between the two terms, we set α 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 planes at each point are parallel. 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 Fig. 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.



(a) Four piece chain (b) Four piece cluster

Figure 6: As matching contours are found, edges are added to the assembly graph. When cycles are detected, as in (b), edges may be jointly re-optimized to incorporate the additional information. Note how the gap between pieces ***B*** and ***C*** has been closed with the addition of the yellow edge.

changing the orientation of one fragment in such a subset will impact the alignment quality of the others.

Fig. 6 shows the significance of a cycle in the graph. In the four node chain of Fig. 6a there is only one path between any pair of nodes. The addition of the fourth edge in Fig. 6b has created a cycle. It is plain to see that we cannot adjust the relative alignment between two nodes without impacting the quality of at least one other alignment. In other words, each path between two nodes serves as a set of constraints that can be used to improve the alignment quality between them. 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 ***B*** and ***C*** has been closed by this process, and the overall alignment quality of the subset has been improved.

We formulate the 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 Eq. 6. Each alignment's translation and rotation 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 [BM92]. 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.

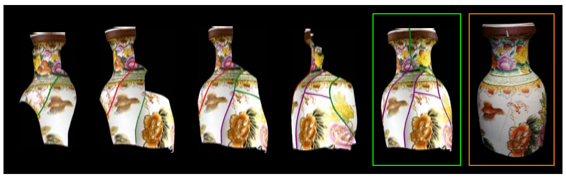


Figure 7: A 6 piece store bought vase is reassembled using our system (green box) and by hand (orange box). At each step the green contour indicates the proposed addition, and purple indicates alignments that have consequently been adjusted. This dataset is labeled VASE in Fig. 13.

7. Grouping Fragments

We introduce one final aspect of our method to aid in the detection of matching contour groups such as the ones shown in the last three steps of Fig. 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. We use a hyper-graph to represent this part of our algorithm. Each connected component subgraph of our alignment graph becomes its own hyper-node with its own scale-space boundary representation. We form the boundary representation by first extracting the unmatched boundary points from all constituent pieces.

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 addition to the examples in Fig. 7, a strong example of this is found in Fig. 10. The left-most step in the second row shows a matching contour that spans 3 fragments on the right, and 2 on the left. Since we prioritize candidate matches based on their length, the small matching contours that make up this long match would not otherwise have been found in a reasonable time-frame.

8. Experimental Results

To validate our algorithm we augmented it with an interactive user interface. The most likely matching contour is presented to the user, who may then accept or reject it. When a match is accepted, it is added to the graph, and any cyclically dependent alignments are re-optimized. Except where indicated, the resulting reassemblies contain every fragment in the dataset. Unfortunately, the exact alignment of the fragments cannot be compared to any ground truth.

In Fig. 7 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.

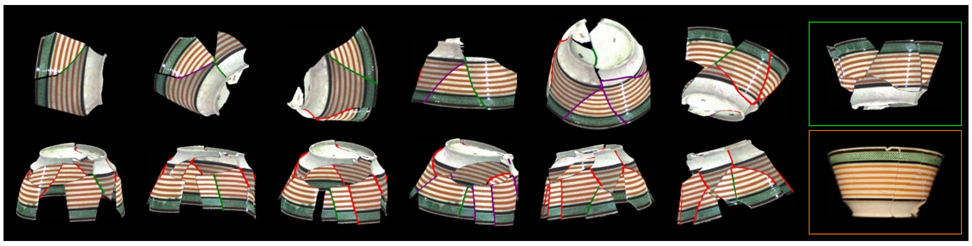


Figure 8: A 16 piece vessel is reassembled using our system (green box) and by hand (orange box). The green contours indicate our method's proposed addition, purple contours indicate cyclically dependent alignments that have been re-optimized to include the green contour's additional information. This re-optimization is best illustrated in the transition between the middle two images of the bottom row where a large gap is closed. This result is labeled BOWL1 in the graph of Fig. 13.

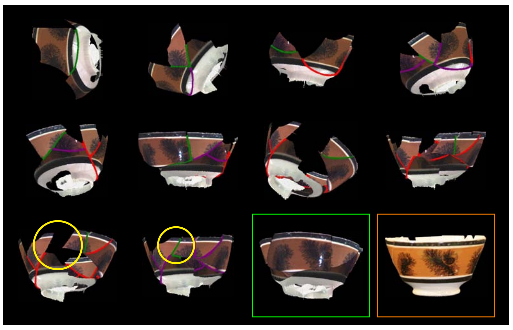


Figure 9: A 10 piece vessel is reassembled using our system (green box) and by hand (orange box). Note the large gap that is closed in the last step. Shown in purple, nearly every alignment is adjusted to correct this accumulated error. This dataset is labeled BOWL2 in the graph of Fig. 13.

In Fig. 13 we compare total reassembly times using our method with manual reassembly of the actual artifacts, here this object is referred to as VASE. The remaining five objects are recently excavated cultural heritage artifacts, and as such exhibit increased chipping and signs of wear. Although these datasets describe axially symmetric objects, the fragments themselves are quite small. Consequently, past work that rely on estimating the profile contour of symmetric objects [WC04, The01] may not be well suited for this task.

In Fig. 8 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 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 Fig. 13.

In Fig. 9 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 Fig. 13.

Fig. 10 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 segment long matching contours. Because of these two factors many erroneous matches are considered. We discard clearly erroneous matches by inspecting the resulting alignment error. 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 Fig. 13, 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.

In Fig. 11 and Fig. 12 we show two more examples. These datasets, which are respectively labeled PLATE2 and PLATE3 in Fig. 13, 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.

9. Conclusion

Whereas past work rely on simplistic assumptions about the global geometric or photometric structure of the object, our approach is applicable to any reassembly problem where the boundary of each fragment may be extracted. To achieve this, we have introduced a method that bridges the gap between top-down and bottom-up approaches. Since access to cultural heritage objects is restricted, simply achieving alignment of real artifacts is an important step. Although we utilized only a rudimentary user interface to help cull the exponential search space, we found that our system outperforms a purely manual reassembly. In future work we will investigate more engaging user experiences that more fully leverage the important contributions both human and algorithm may have on this critical task.

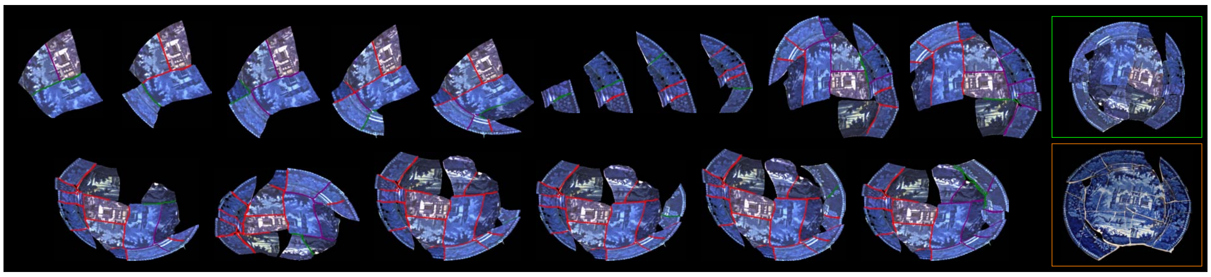


Figure 10: Steps taken to reassemble a 21 piece artifact. Note how the primary cluster is abandoned after the eighth piece. At this point the strongest matches were part of the rim. After completing this 4 piece section, a connection was made between the two components, enabling the completion of the rest of the vase. The final result (green) is compared to the hand reassembly (orange). Note that a few steps have been omitted for compactness. This vessel is referred to as PLATE1 in the graph of Fig. 13.

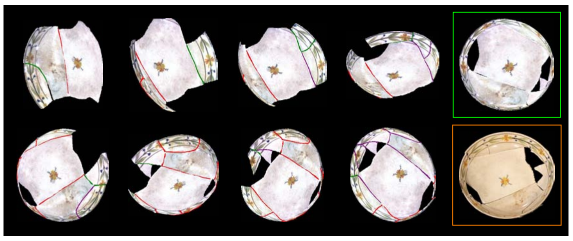


Figure 11: This 11 piece vessel is labeled PLATE2 in Fig. 13.

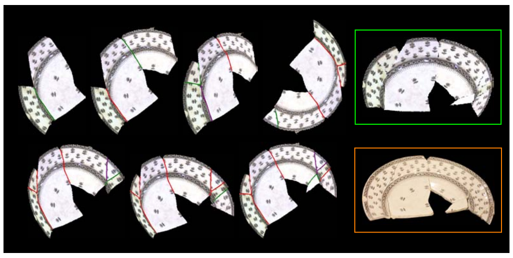


Figure 12: This 8 piece vessel is labeled PLATE3 in Fig. 13.

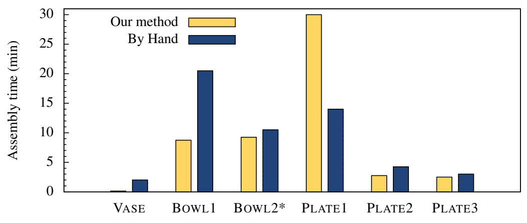


Figure 13: Our method (yellow) enables rapid surface alignment. Note that when reassembling BOWL2 by hand, our test subject was unable to complete the task and gave up at the reported time. The long reassembly time of PLATE1 (shown in Fig. 10) is due to the strongly connected nature of the reassembly graph. Large scale optimizations are far more frequent when assembling this artifact.

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

[BM92] BESL P., MCKAY H.: A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256. 6

[GS02] GAMA LEITAO H., STOLFI J.: A Multiscale Method for the Reassembly of Two-Dimensional Fragmented Objects. *IEEE Trans. Pattern Analysis and Machine Intelligence 24*, 9 (Sept. 2002), 1239–1251. 2, 3

[HFG*06] HUANG Q., FLÖRY S., GELFAND N., HOFER M., POTTMANN H.: Reassembling Fractured Objects by Geometric Matching. *ACM Trans. Graphics 25*, 3 (July 2006), 578. 1

[KHW90] KISHON E., HASTIE T., WOLFSON H.: 3-D Curve Matching Using Splines. In *European Conf. Computer Vision* (Berlin/Heidelberg, 1990), vol. 427 of *Lecture Notes in Computer Science*, pp. 589–591. 2, 3

[Mar63] MARQUARDT D. W.: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics 11*, 2 (June 1963), 431–441. 5

[MM86] MOKHTARIAN F., MACKWORTH A.: Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1 (1986), 34–43. 2, 3, 4

[Mok88] MOKHTARIAN F.: Multi-scale Description of Space Curves and Three-Dimensional Objects. *IEEE Conf. Computer Vision and Pattern Recognition* (1988), 298–303. 2, 3, 4

[Pul99] PULLI K.: Multiview registration for large data sets. *Int'l Conf. 3-D Digital Imaging and Modeling* (1999), 160–168. 2

[SLW04] SHARP G. C., LEE S. W., WEHE D. K.: Multiview registration of 3D scenes by minimizing error between coordinate frames. *IEEE Trans. Pattern Analysis and Machine Intelligence 26*, 8 (Aug. 2004), 1037–50. 2

[Str50] STRUIK D.: *Lectures on Classical Differential Geometry*. Addison-Wesley, Cambridge, Mass., 1950. 3

[The01] THE SHAPE LAB: Assembling Virtual Pots from 3D Measurements of Their Fragments. *Virtual Reality, Archeology, and Cultural Heritage* (2001). 2, 7

[WC04] WILLIS A. R., COOPER D. B.: Bayesian Assembly of 3D Axially Symmetric Shapes from Fragments. In *Proc. Conf. Computer Vision and Patern Recognition* (2004). 2, 7

[WK01] WEIXIN K., KIMIA B.: On Solving 2D and 3D Puzzles Using Curve Matching. In *Proc. Conf. Computer Vision and Pattern Recognition* (2001). 2, 4

[Wol90] WOLFSON H.: On Curve Matching. *IEEE Trans. Pattern Analysis and Machine Intelligence 12*, 5 (1990), 483–489. 2, 3