3D Fragment Reassembly using Integrated Template Guidance and Fracture-Region Matching

Kang Zhang Wuyi Yu
School of Electrical Engineering and Computer Science
Louisiana State University

Mary Manhein
Dept. Geography & Anthropology
Louisiana State University

Warren Waggenspack
Dept. Mechanical & Industrial Engineering
Louisiana State University

Xin Li*
School of Electrical Engineering and Computer Science
Louisiana State University

Abstract

This paper studies matching of fragmented objects to recompose their original geometry. Solving this geometric reassembly problem has direct applications in archaeology and forensic investigation in the computer-aided restoration of damaged artifacts and evidence. We develop a new algorithm to effectively integrate both guidance from a template and from matching of adjacent pieces' fracture-regions. First, we compute partial matchings between fragments and a template, and pairwise matchings among fragments. Many potential matches are obtained and then selected/refined in a multi-piece matching stage to maximize global groupwise matching consistency. This pipeline is effective in composing fragmented thin-shell objects containing small pieces, whose pairwise matching is usually unreliable and ambiguous and hence their reassembly remains challenging to the existing algorithms.

1. Introduction

Geometric restoration that composes 3D fragmented object into its original shape is a difficult matching problem that has many direct applications such as archaeological reconstruction, digital heritage archiving, and forensic evidence analysis, to name a few. A few 3D reassembly algorithms have been developed in graphics and computer vision literature. However, difficulty was reported in handling thin-shell 3D objects and in processing small fragmented pieces [15][34]. In this work, we explore the reassembly of fragmented thin-shell objects with general geometric shapes and small-sized fragments. One application is forensic skull reassembly. In law enforcement investigation cases, to assist with victim identification, forensic anthropologists need to predict the skull's ancestry and perform facial reconstruction/superimposition to help identify the body. The skeletal remains, however, are often incomplete or fragmented due to trauma or environmental exposure. Therefore, forensic specialists need to first perform a manual recomposition of skull fragments before subsequent assessment and analysis. With effective digital reassembly algorithms, after all the fragments are digitized, the existing manual skull reassembly procedure can be automated and augmented.

Considering the geometry of a fragment, its boundary surface consists of two types of regions: the *intact regions* and the *fracture regions*. The *intact regions* are those from the original boundary surface of the complete object before fracturing, and the *fracture regions* are those generated due to fracturing. Fig. 1 (c) illustrates the intact and fracture regions on a fragment.

(a) (b) (c)

Figure 1. (a,b) A ceramic object is fragmented, (c) intact (blue) and fracture (red and gray) regions on a fragment.

Existing 3D fragment reassembly algorithms can be generally classified into two types: (1) reassembly based on fracture-region matching, and (2) reassembly using template guidance. Fracture-region matching approaches exploit similarities in the local fracture geometry of adjacent fragments [8, 33, 15]. The template guidance approaches compose fragmented pieces based on their best match to a complete model [34, 31]. Each approach has advantage and

*Corresponding author. Email: xinli@cct.lsu.edu

limitations, and reassembly algorithms in both categories report difficulty in effectively processing small fragmented pieces. First, with small fragments, it is particularly challenging to differentiate and segment intact and fracture regions. Second, the number of uncertain potential matches tends to be large and their effective pruning is difficult.

In this paper, we develop a new effective reassembly pipeline integrating both template-guidance and fracture-region matching. The idea is to use the information from both intact and fracture regions to construct many potential matching relationships among the fragments and template; then, through a multi-piece matching optimization, we prune and refine these possible matches to obtain globally consistent alignment of the fragments. This approach is formulated in a 3-step pipeline: (1) initial reassembly guided by a template; (2) pairwise fracture matching between fragments; (3) multi-piece matching integrating both intact and fracture information. The **main technical contributions** include (a) reliable pairwise matching algorithms to align fragments with small overlapping regions, and (b) a multi-piece matching and refinement algorithm effectively integrating both template guidance and pairwise fragment matchings, which iteratively optimizes the positioning of fragments while consistently controlling accumulated error and avoiding penetrations.

2. Background and Related Work

2.1. 3D Fragment Reassembly

Early research on data reassembly focused on solving the 2D jigsaw puzzle problem [13]. 3D objects are often projected to 2D [9, 21, 32] and matched by their planar boundaries. In these approaches, planar boundaries are identified manually or detected through the difference of texture/geometry between the intact and fracture regions. These methods work for large thin shell fragments, on which the thickness of the fracture region can be ignored and fracture surfaces can be treated as curves. Recent algorithms solve the reassembly directly in 3D. Cooper et al. [8] assemble 3D pot fragments by matching the break-curves and shard normals. Willis and Cooper [33] improve pottery assembly by applying Bayesian analysis with a semi-automatic matching. Their experiments were shown most effective for axially symmetric and geometrically smooth/simple fragmented pots. Yin et al. [34] use templates to roughly assemble skull fragments, then perform a break-curve analysis to refine the composition. This method also approximates thin shell fragments as surface patches, which is not suitable for small fragments [34]. For general 3D solid models, Papaioannou et al. [23] segment the fragments and extract fracture regions, then use depth maps for matching. Huang et al. [15] propose a 4-step framework for fragmented solid reassembly. Both of these methods require the segmentation of intact and fracture regions on the fragment, which is often difficult on small fragments. Herein, we develop an algorithm to incorporate intact and fracture geometry without explicit segmentation.

2.2. Geometric Partial Matching

3D Partial matching is a key enabling tool for fragment reassembly [19]. Partial matching is related to *geometric feature detection*, *feature description*, and *feature correspondence*. **Feature detection** identifies geometrically salient *keypoints*. [30] evaluates the performance of a few recent feature detectors suitable for partial matching [36, 22, 35, 6]. Specifically related to our problem, desirable feature detectors should identify keypoints consistently under noise with small resolution variations. **Shape descriptors** are designed to evaluate the similarity between extracted keypoints. Descriptors are typically invariant under either rigid transformation [16, 25] or isometric transformations [27, 28, 3]. In this application, we focus on descriptors for rigid transformations. *Curvatures* [25] and *Integral Invariants* [12] are popular local descriptors in object recognition and surface matching. However, these descriptors are sensitive to the local geometric variance in handling template-subject disparity in fragment reassembly. Histogram-based descriptors, such as *spin images* [16] and *shape context* [4] compress geometric structures into bins, hence are more globally discriminate and less sensitive to local geometric variance. [29] improves them by using a unique local reference frame. This descriptor performs well on partial surface matching. **Feature correspondence refinement** is a procedure to establish a bijective map between features on different shapes. Effective feature matching algorithms include *voting* [20], *RANSAC* [11], *graph matching* [7], and *forward search* [15].

3. Template Matching

In some reassembly tasks, complete models with similar geometry to the subject data are available and can be used to guide the reassembly. For example, in forensic skull restoration from fragmented pieces [34], existing skulls models can provide useful guidance. In tasks such as building 3D repository database, models archived in the same category are also suitable templates.

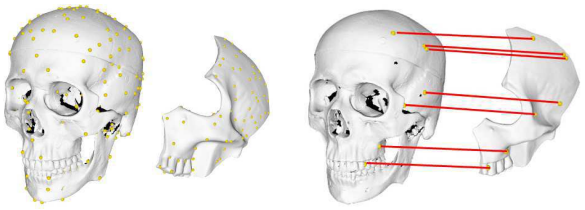
Following a template model M, we can reassemble fragments after solving partial matching between each fragment F_i with M. The matching should align intact regions of F_i with M, despite minor geometric disparity between the subject and template. Our template matching algorithm has two steps: (1) feature extraction and initial correlation; (2) correspondence refinement. The output are ranked 3D transformation lists $\{T_i^j\}$ that align F_i with M.

Feature extraction and initial correlation. According to the comparison in the recent survey [30], among widely used 3D feature detectors, the *Intrinsic Shape Signatures* (ISS) [36] offers great repeatability and efficiency in rigid

partial matching. The ISS constructs the covariance matrix of the support region of each 3D point and whose largest eigenvalues differs much with its second largest eigenvalues are identified as features. We extract features using ISS on F_i and M respectively. Fig. 2(a) illustrates an example of extracted features.

To accommodate geometric disparity between the template and subject, a shape descriptor that is not too sensitive to local geometric variance is preferred. The histogram-based descriptor, such as *spin images* [16] and *shape context* [4], can stably reflect geometric variance. The Signature of Histograms of OrienTations (SHOT) [29] is another effective histogram-based descriptor. SHOT first constructs a unique and unambiguous local reference frame, then calculates the descriptor by compressing the geometric information into bins along these three axes. This descriptor outperforms the spin image in partial matching applications such as object recognition and 3D multi-view reconstruction. Therefore, we use SHOT to describe and compare feature points on M and on F_i.

For each feature p on M, its SHOT descriptor, denoted as $S(p)$, is a 352-dimensional vector, and its top k most similar features on F_i are extracted. Specifically, $p \in M$ and $q \in F_i$ is considered as potential corresponding pair if (i) $D(p,q) = |S(p) - S(q)| < \delta_S$ and (ii) $D(p,q)$ is one of the k smallest $D(p,\cdot)$ values that satisfies (i). Then (p,q) is kept and added into an initial *correspondence set* $\bar{\mathcal{C}}$ ($\delta_S = 0.05$ and $k = 5$ in all our experiments).



(a) (b)

Figure 2. Feature Extraction and Matching. (a) ISS keypoints on the template and a fragment, (b) refined feature correspondences.

Correspondence Refinement. To ensure enough number of features are extracted on smaller fragments, M usually contains many keypoints. This makes finding correct feature correspondence difficult: The size of *innitial correspondence set* $\bar{\mathcal{C}}$ is big, since keypoints from other irrelevant regions on M (that does not correspond with F_i) could introduce irrelevant correspondences into $\bar{\mathcal{C}}$. So the correct correspondence pairs are significantly fewer than the outliers. Due to the big size of $\bar{\mathcal{C}}$, powerful correspondence refinement algorithms such as graph matching [7] and forward search [15] turn out to be too computationally expensive for our pipeline. The RANSAC algorithm [11], instead, is efficient and suitable for correspondence refinement here. To extract correct correspondence pairs from outliers, we evaluate geometric consistency among correspondence pairs. Given two correspondence pairs $c_1 = (p_1, q_1), c_2 = (p_2, q_2)$, where $p_1, p_2 \in M, q_1, q_2 \in F_i, c_1, c_2 \in \bar{\mathcal{C}}$, c_1 and c_2 are geometrically consistent if the Euclidean distance between p_1 and p_2 is similar to the distance between q_1 and q_2, namely, $|\|p_1p_2\| - \|q_1q_2\|| < \delta_t$. δ_t can be chosen according to similarity between the template and subject ($\delta_t = 5$mm in our experiments).

In the RANSAC algorithm, for each extracted transformation model, if the ratio of size of inlier set to size of $\bar{\mathcal{C}}$ is bigger than a threshold δ_r, this transformation is accepted. To estimate a suitable threshold δ_r, we observe that it is mainly affected by (1) the size of F_i and (2) k value in the initial correspondence extraction. We set $\delta_r = \lambda \frac{V(F_i)}{k \times V(M)}$, where $V(F_i), V(M)$ are volumes of the fragment and the template approximated by their bounding box volumes, and λ is a weighting factor. $\frac{V(F_i)}{V(M)}$ roughly estimates the ratio of features from M that have corresponding features in F_i, among which only $1/k$ is correct (since top k matches is preserved). We conservatively use $\lambda = 0.5$ to set δ_r in all our experiments. Fig. 2 (b) shows an accepted correspondence between a skull fragment and a template.

4. Pairwise Fracture Region Matching

Matching with a template object only roughly positions big fragments. And sometimes, no template is available. Therefore, effective matching of shared fracture regions of adjacent fragments is a critical component to successful reassembly. We extracts potential alignments between each pair of fragments F_i and F_j, and each such alignment is referred to as a *relative transformation* T_{ij}.

Computing reliable fracture matching is harder than template matching, because the corresponding fracture regions are significantly smaller than intact regions and possessing less geometric saliency, especially if the subjects are thin-shell objects. Very few effective feature points on fracture regions can be extracted to support reliable matching between fragments. Therefore, rather than using feature points, matching based on *boundary curves* [33, 32] or *feature regions* [15] are used in existing work for aligning adjacent fragments. The *region-based matching* extracts salient geometric areas on fracture regions to compute alignments between fragment pieces. Unfortunately, thin-shell objects often have very simple and flat fracture regions, lacking salient region to help matching computation. In contrast, sharp ridge and valley curves are more salient features for effective partial matching. Thus, solving pairwise alignments using feature curves extracted on fragments is a better strategy here.

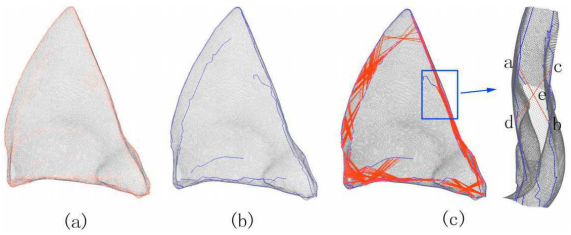


Figure 3. Feature Curve Matching. (a) Extracted feature segments (orange) on a fragment, (b) pruned feature network, (c) sampled wide bases (red crosses).

4.1. Feature Curve Extraction

We extract a set of feature curves on fragments following [24] which was shown to be robust against noise. First, extract points with high mean curvature. Then build a minimum spanning tree (MST) to connect all these points into a curve network. The weights of edges are designed [24] to make MST passing through sharp regions as much as possible. The MST contains many short branches, making the matching of curve network difficult. Then a bottom-up graph pruning is used to eliminate these short branches on the MST tree: (1) sort all leaves of the tree based on their depth; (2) determine the longest path by traversing this tree. (3) remove short branches from the tree. Iterate this pruning until no short branch is left on the tree. Then the final feature curve network, consisting of the remaining salient feature curve paths are extracted. Note that, this algorithm does not rely on the initial root of MST and can remove noisy short feature line segments reliably. Fig. 3(a,b) illustrate an example of feature curve extraction.

4.2. Feature Curve Matching

In existing fragment reassembly work, a few curve matching algorithms [17, 9, 21, 32, 33] have been developed to match adjacent fragments. However, in these curve matching formulations, each fragment's contour is extracted as a simple curve loop, and the matching is performed to find certain "Longest Common Substrings" (LCS) shared by two curve loops. They assume the fragment can be scanned in a way that only intact surfaces are obtained. Unfortunately, such a scan and the clear separation between intact and fracture regions are too difficult when fragments are small. Following our feature extraction, the curve networks obtained from fragments are much more complicated than a curve loop, and their alignment need more sophisticated matching strategy.

To match two feature curve networks C_1 and C_2 reduces to finding a rigid transformation T that $T(C_1)$ and C_2 have the largest overlap. Inspired by [1], we formulate this problem as a Largest Common Point-set (LCP) problem: given two point sets X and Y, the LCP under δ_D-congruence is to find a transformation T and a subset $P = \{p_i\} \subset X$ with the largest cardinality, such that given a distance threshold δ_D, $Dist(T(p_i), Y) < \delta_D$, where $Dist(T(p_i), Y)$ measures the distance from transformed point $T(p_i)$ to point set Y. We also call the cardinality of P the *LCP score*.

We develop a voting algorithm to solve this LCP problem. (1) Extract *4-point congruent sets* (Section 4.2.1) on C_1 and C_2, compute their matchings. (2) Conduct a voting and get the transformations that are top k favorably voted (Section 4.2.2). After the top-k LCP transformations are obtained, we refine each transformation using the iterative closed points (ICP) [5] algorithm and discard alignments that lead to fragment penetration. The penetration is checked by the collision detection package named SWIFT++ [10]. Note that, unlike [1] which conduct a RANSAC for randomly generated points on surfaces, we conduct a thorough extraction of 4-point congruent sets on pruned feature curves. Such a thorough extraction on well pruned samples makes our curve matching much more robust than [1] when there is a high percentage of outliers, which always happens in pairwise fragment matching.

4.2.1 4-Points Wide Base Extraction and Matching

Inspired by [1], we match curve networks using a modified *4 points congruent set* (4PCS) algorithm. A *4-points wide base* is a set of 4 points that are coplanar. The coplanar 4 points set is wide if the length between each two points is larger than a threshold. A *wide base* created by selecting points that are far from each other results in more stable alignments [14]. However, if the points are too far away, the selected points will not all lie in the overlap regions. Therefore, with two controlling parameters d_{min}, d_{max}, we first randomly pick 3 points such that the distance between each two points is in the range of $[d_{min}, d_{max}]$, then select the 4th point that is in the same distance range and is coplanar with these three points. d_{min} and d_{max} are chosen according to the estimated thickness d_t of the fragments: $d_{min} = 0.5 * d_t$ and $d_{max} = 2 * d_t$. Fig. 3(c) shows an example of the 4-points wide bases on the curve network.

Given a 4-points wide base $X = \{a, b, c, d\}$. Let ab and cd be the two lines intersecting at an intermediate point e. We can build a 5-dimensional descriptor vector $v = \{l_1, l_2, \theta, r_1, r_2\}^T$, where $l_1 = \|ab\|$ and $l_2 = \|cd\|$ are the lengths of segments ab and cd, θ is the angle between them, and r_1 and r_2 are computed as $r_1 = \|ae\|/\|ab\|$, $r_2 = \|ce\|/\|cd\|$. Two 4-points wide bases are *congruent* if their descriptors are similar, and matching them introduces a rigid transformation by a least square fit.

4.2.2 Voting and Evaluation of Alignments

We first compute the sets of 4-point wide bases S_1 and S_2 for feature curve networks C_1 and C_2. Then for any wide base $B_i \in S_1$, find its congruent wide base $B_j \in S_2$ and compute the transformation between B_i and B_j. The num-

ber of transformations introduced by congruent wide base is large, and directly evaluating all their effects is computationally expensive. Therefore, based on the observation that *valid transformations are usually supported by many mutually congruent corresponding base pairs*, we do a voting to pre-select potential transformations. First, a transformation inferred by two 4-point wide bases is transformed into a 6D vector (a_x, a_y, a_z, x, y, z), where a_x, a_y, a_z are rotation (Euler) angles and x, y, z are translations. Then the 6D parameter spaces are divided into buckets. Finally, the k buckets with most vectors are selected as the transformation. The total time complexity of this algorithm is $O(n \log n)$. Each pre-selected transformation receives a certain number of votes, but it does not necessarily have high *LCP scores*. Therefore, to re-rank the transformations following the *LCP scores*, for each pre-selected transformation T_i, we compute $T_i(C_1)$ and count points in $T_i(C_1)$ within δ_D-distance to C_2. We use Approximate Nearest Neighbor (ANN) [2] for efficient neighborhood query. Then, the transformations are re-ranked, and the top K_p (in our experiments, $K_p = 200$) transformations are kept.

5. Multi-piece Matching

The template matching (Section 3) suggests a set of possible alignments between the fragments and template; and the pairwise matching (Section 4) suggests a set of possible alignments among the fragment pairs. We perform a graph-based search algorithm to extract globally coherent pairwise matchings from the above possible alignments.

5.1. Notation: Reassembly Graph Representation

We encode the fragments, the template (if used), together with all the extracted potential correspondences in a *reassembly graph* $G = (V, E)$. Each node $n_i \in V$, denoting a fragment F_i, is associated with a transformation matrix X_i that needs to be solved, to get its final position $X_i(F_i)$ in the reassembly. X_i is a 3×4 matrix $[R|t]$ where R is a 3×3 rotation matrix and t indicates a 3D translation. The template model M is denoted as a base node $n_0 \in V$ in G. To get a unique solution, we set template's transformation matrix X_0 to be $[I_3|0_3]$, where I_3 is the 3×3 identity matrix and 0_3 is the 3D zero vector.

A directed edge $e^k_{i,j} \in E$ from node n_i to n_j denotes an alignment from fragment F_i to F_j. This alignment is represented by a relative transformation $T^k_{i,j}$ on each edge $e^k_{i,j}$. Note that between a pair of nodes n_i and n_j, multiple edges ($k = 1, 2, \ldots$) may exist, so G is not a simple graph. These relative transformations $T^k_{i,j}$ are pairwise alignments computed in the previous steps. The matches from each fragment F_i to the template M are also represented as edges $e^k_{i,0}$ and relative transformations $T^k_{i,0}$.

To simplify the formulation in the following, given a directed edge $e^k_{i,j}$, we can also consider it in its opposite direction, as $e^k_{j,i}$, from n_j to n_i. Accordingly, the associated alignment is naturally defined as $T^k_{j,i} = (T^k_{i,j})^{-1}$. With this we can study G as an undirected graph. For example, we can say edges $e^1_{1,2}$ and $e^2_{1,2}$ form a loop, without specifically modifying the order of the subscripts.

5.2. Objective Function and Constraints

We define a few matching scores to measure how well fracture or intact regions overlap. Two regions are considered *overlapped* if their Hausdorff distance is smaller than a threshold ξ_h (based on the precision of the 3D scanner, ξ_h = 3mm in our experiments). Given two nodes n_i and n_j and their associated transformation X_i and X_j, we define a **fracture matching** score $S_f(X_i, X_j)$ as the area of the overlapping regions between these two fragments after applying X_i and X_j. Similarly, we define an **intact matching** score $S_i(X_i, X_0)$ as the area of overlapping regions between $X_i(F_i)$ and the template M. Naturally, $S_i(X_i, X_j) = 0$ between fragment nodes for $i, j \neq 0$, and $S_f(X_i, X_0) = 0$ between each fragment node and the template node. Then, combining them together, between each pair of nodes n_i and n_j, we define a total **region matching** score $S_r(X_i, X_j) = S_f(X_i, X_j) + \alpha S_i(X_i, X_j)$, where α is a weighting factor that can be adjusted according to the availability of a good template. By adding template node n_0 and the intact matching term, our algorithm can make good use of available template to guide multi-piece matching. α and the available template affect the reassembly results. Ideally, when template is similar to the ground truth, α should be set big, while when no good template is available, α should decrease. When a wrong template is used, the reassembly will be bad if α is big. Setting $\alpha = 0$ makes the reassembly to only rely on fracture region matching of adjacent pieces. By default, we set $\alpha = 0.1$, allowing the pairwise fragment matching to play a more important role.

Finally, the multi-piece matching problem reduces to solving an optimal subgraph $G' = (V', E'), V' \subseteq V, E' \subset E$, and a set of transformations $\{X_i | n_i \in V'\}$ maximizing the following accumulated region matching score $\Phi(G')$,

$$\Phi(G') = \sum_{n_i, n_j \in V'} S_r(X_i, X_j), \tag{1}$$

subject to the following *validity constraints*.

Validity Constraints. A subgraph $G' = (V', E')$ and associated transformations infers a valid reassembly if

1) G' is a *simple graph*. At most one edge can exist between two nodes, hence a unique alignment is selected between these two fragments.
2) *Loop closure* constraint: for any loop in G', composing the relative transformations along this loop should yield an identity transformation.
3) *Non-intersection* constraint: reassembled fragments should not spatially penetrate each other.

It is not hard to verify that a graph that satisfies the *loop closure* constraint must be a *simple graph*. Because two different transformations between a fragment pair automatically infer a non-identity loop between these two nodes.

5.3. Optimizing Multi-piece Matching

We develop an iterative algorithm to optimize the multi-piece matching. First, initialize an empty graph G'. Then, iteratively grow it by adding a new node with its incident edges from G, maximizing $\Phi(G')$ subject to the validity constraints. During the growing of G', we simultaneously refine all the alignments added in G'. This algorithm is formulated as follows.

In: $G = (V, E)$ and $\{T_{ij}^k\}$;

Out: $G' = (V', E'), V' \subseteq V, E' \subset E$, and X_i for $n_i \in V'$.

1. $G' = \emptyset$; add n_0 to G' and let X_0=$[I_3|0_3]$;
2. Find an edge $e_{i,j}^k \in E$ where $n_i \in V', n_j \in V \backslash V'$, such that adding $e_{i,j}^k$ to G' does not violate *validity constraints* and leads to maximal increase of $\Phi(G')$. Add this $e_{i,j}^k$ to E' and add n_j to V'; set $X_j = (T_{i,j}^k)^{-1} * X_i$.
3. Perform a graph optimization refinement on G';
4. If $V' = V$ or no further edge $e \in E$ can be added into G', STOP; otherwise, GOTO Step 2.

Evolving G' through Beam Search. In the above Step 2, we add a new edge whose insertion results in a valid new G' with largest increase of F. G' is called the evolution of G. This greedy approach usually stops in local minima. In this problem, between fragment pairs, multiple $T_{i,j}^k$ exist and the actually correct alignment may not be highest ranked. To more robustly circumvent such local minima, we implement a beam search strategy [26]. In each iteration, instead of choosing the single best alignment, we explore m highest-scored $e_{i,j}^k$. First, G' evolves into m different graphs. On each of these graphs, expand its n best subsequent evolutions, which will result in totally $n \times m$ possible graphs. Among these graphs, the m highest-scored graphs are selected and preserved (to avoid exponential space and time expansion). This process repeats until no new edge can be added into G'. Through beam search, G' has a better chance to circumvent small local minima.

5.4. Multi-piece Reassembly Refinement

Transformations X_i are computed through the composition of relative transformations during the growth of G', small errors can accumulate and affect the subsequent X_i computation and penetration test. To minimize this error accumulation, we apply a graph optimization on G' to globally refine existing transformations. We minimize the following least square function defined on G',

$$\phi(\mathbf{X}) = \sum_{e_{i,j} \in E'} ||T_{i,j}^k - X_i \times X_j^{-1}||^2 \times S_r(X_i, X_j), \quad (2)$$

where $\mathbf{X} = \{X_1, \ldots, X_{|V'|}\}$. Node transformation X_i should be coherent with the selected relative transformations $T_{i,j}$ on its incident edges. X_i is first computed by one $T_{i,j}$ by $X_i = T_{i,j} \times X_j$ when n_i was added to G'. But a later added node n_r may introduce another edge $e_{i,r}$ and associated $T_{i,r}$. This consistency $X_i = T_{i,r}^h * X_r$ may not exactly be satisfied due to accumulated noise or numerical errors. Therefore, we globally refine $\mathbf{x}$ following Eq. (2) on G'. We implement this optimization following the algorithm of [18]. With analytic derivative information available, this optimization converges very efficiently.

6. Experimental Results

Experiment Setting. We evaluate our reassembly algorithm on various ceramic models and real forensic skull data. For ceramic models, we first scan the complete models as ground truth for result evaluation. Then break each object into fragments and scan them individually. Our algorithm reassembles the digital fragments and we compare our result with the ground truth. We also use our algorithm to compose real fragmented skull data provided by forensic specialists. In practical law investigation cases, forensic specialists restore the skull geometry from fragments manually, then perform subsequent ancestry analysis, facial reconstruction for body identification. Our digital restoration results on these skulls are visualized. All the 3D objects and fragments are digitized using a NextEngine scanner (with reported accuracy $\pm 0.3mm$). We also estimate and document our scanning accuracy in these experiments by performing multiple scans on each model and computing Hausdorff distances between different scans.

Evaluation of Reassembly Error. To numerically evaluate reassembly accuracy, we define two error metrics: the *Reference Error* ε_R and *Matching Error* ε_M. The completed model $S = \bigcup X_i(F_i)$ (accordingly, also fragments) is scaled to a unit box for consistent error evaluation. If the ground truth model $\hat{S}$ is available, we can measure the **Reference Error** ε_R by comparing S and $\hat{S}$: compute a displacement field $\eta_r(x)$ on $\hat{S}$ to record the distance from each point $x \in \hat{S}$ to its nearest point on S, then use the average, $\varepsilon_R = \sqrt{\frac{\sum_{v \in \hat{S}} f^2(v)}{|\hat{S}|}}$, where $|\hat{S}|$ represents the number of points of $\hat{S}$. When ground truth is not available, we can use a **Matching Error** ε_M to estimate how well the adjacent fragments align with each other after the reassembly. For each point p on the $X_i(F_i)$, if its nearest corresponding point q on an adjacent fragment $X_j(F_j)$ has an opposite normal direction from p's normal, then p and q are paired, and $d(p) = \|pq\|$. Then the average distance of paired points is calculated $\varepsilon_M = \sqrt{\frac{\sum_{p \in X_i(F_i)} d(p)}{|P|}}$, where $|P|$ represents the number of such pairs. Note that, the normal constraint in pairing is used to avoid measuring points

from intact regions.

6.1. Reassembly Results

Our reassembly program was run on a laptop with 2.0GHz Core i7 CPU and 6GB RAM. The Runtime Table 1 documents the efficiency of our experiments. The total computation needs from 4 to 14 minutes, depending on the number of fragments and their geometry. The pairwise matching to collect potential alignment between fragments is the most time-consuming step in this reassembly pipeline.

Table 1. Runtime Table: #V is the total point number of each set of scanned object, #F is the number of fragments, T_t, T_p, and T_m are computational times (in seconds) for template-guided reassembly, pairwise matching, and multi-piece matching, respectively.

Models	# V	# F	T_t	T_p	T_m
Child	1066 k	8	123	89	31
Chicken	882 k	11	183	244	37
Buddha Head	1152 k	12	–	142	43
Skull 1	1294 k	19	146	614	64
Skull 2	427 k	10	93	141	20

Figure 4. Reassembling the Child-Buddha and Chicken models.

We first test a simpler scenario on ceramic objects, using the ground truth model as the template. Fig. 4 illustrates reassembly of the fragmented Child Buddha model and chicken model. The physical dimensions of the smallest pieces here are about $3cm \times 2cm \times 0.4cm$. Note that the scan resolution is about $0.3mm$, and the final matching error ε_M about $0.8mm$. These experiments show that small fragments are reassembled effectively and accurately.

We also examine our algorithm when only *template with relatively big geometric difference* is available. Fig. 5 shows a reassembly example of a gnome model. The broken

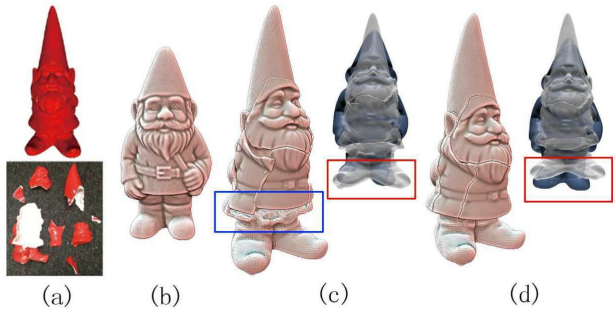


Figure 5. (a) The subject gnome model and its fragments; (b) another gnome model as the template; (c,d) two reassembly results when setting $\alpha = 1$ and $\alpha = 0.01$, respectively.

gnome model (a) is geometrically different from the available template gnome (b). When a big template-guidance weight $\alpha = 1$ is used, the template (b) more strongly affects the composition and leads to the result (c). Coming with a better align the shoes (red box in (c)), there is a bigger gap in the final reassembly (see the region in the blue box). When a small $\alpha = 0.01$ is used, the reassembly, after initial positioning, more relies on inter-fragment alignment, yields a more tightly composed (i.e., smaller matching error) reassembly result (d).

Figure 6. Reassembling a 19-piece fragmented skull model.

Figure 7. Reassembling a fragmented evidence skull in a real law investigation case.

Skull Reassembly. Human skulls possess geometric similarity. Hence, complete human skulls are suitable templates in fragmented skull reassembly. We collected a group of standard skull models from all the major ancestries: {male, female} $\times$ {white, black, and Asian} skulls, using them in guiding skull reassembly. Fig. 6 shows our reassembly on a ceramic skull model. The model was broken into 19 pieces. We randomly used a white male skull model as a template to perform the reassemblies. Despite the geometric disparity between the template and the ceramic models (see the accompanying video for details), the fragmented skull was successfully reassembled, showing that our reassembly is robust against template-fragment disparity. Fig. 7 shows our reassembly on a real fragmented skull from forensic anthropologists in which the victim's skull is severely damaged due to a gun shot. The two sides of the skull are highly fragmented and some are missing. Its manual restoration becomes challenging for two reasons. First, on some small pieces the fracture and intact regions are hard to distinguish. Second, the fracture regions of the fragments are very thin and hard to align. By effectively integrating the guidance from a template skull model and fracture region matching, our algorithm reassembled this damaged skull satisfactorily.

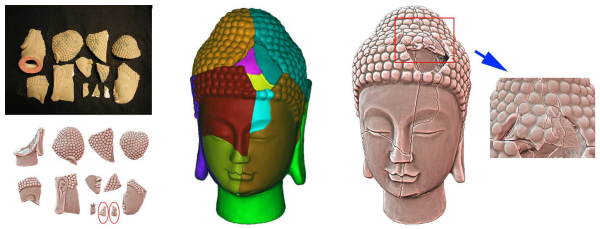
The numerical reassembly error on different models are documented in Table 2. The error distribution is also illustrated in our accompanying video. From the table we can see the algorithm is effective, and in particular, can handle small fragments that are not processed well by existing algorithms such as [34, 15].

Table 2. The Reference Error and Matching Error on different models. $\#F$ is the number of fragments. ε_R and ε_M are the absolute reference error and the absolute matching error (in millimeter). δ_R and δ_M are the reference and the matching error ratios. The error ratio is measured as a percentage of the bounding box diagonal length (i.e., average error / diagonal length $\times 100\%$). Intuitively, for a model with the bounding box diagonal is 1 decimeter, 1% error is 1 millimeter. Some reference errors are not available due to the lack of ground truth model.

Model	$\#F$	ε_R	δ_R	ε_M	δ_M
Child Buddha	8	0.853	0.33%	0.118	0.42%
Chicken	11	0.820	0.27%	0.633	0.21%
Gnome	8	0.575	0.15%	0.697	0.19%
Head	12	n/a	n/a	0.760	0.13%
Skull 1	19	0.435	0.16%	0.595	0.23%
Skull 2	10	n/a	n/a	2.688	0.83%

6.2. Comparison With Existing Methods

Compared with existing algorithms, our algorithms demonstrate better reliability in handling small thin-shell object reassembly. (1) Algorithms using pure pairwise matching, such as [23], cannot not effectively handle ambiguity in pairwise alignment. (2) Algorithms treat fragments as surface patches, such as [33, 32, 31, 34], require that only intact surfaces are scanned and fragment boundaries are simple curve loops. This is practically difficult when fragments are small. (3) Algorithms purely relying on fracture region matching, such as [15], require explicit segmentation of intact and fracture regions which is often difficult for small fragments; and require fracture regions to be big and possess salient geometric variance to allow reliable fracture region alignment, making the approaches more suitable for solid rather than thin-shell objects. Our algorithm effectively tackle the above unsolved challenging issues in thin-shell object reassembly. In Fig. 8, the reassembly of Buddha head model was reported [15] to be difficult due to the existence of small fragments. The algorithm of [15] fails to reassemble the two small pieces (marked in red circles in (a)). In contrast, our algorithm successfully reassembles all the fragments including these small pieces, as shown in (c). In this experiment, no template is available and $\alpha = 0$ is used.



(a) (b) (c)

Figure 8. Reassembling a Buddha Head model. (b) The algorithm of [15] fails to reassemble small pieces (in red circles in (a)). (c) Our algorithm correctly reassemble all the small pieces.

7. Conclusion

We present a new algorithm to integrate template matching and inter-fragment matching for effective reassembly of thin-shell fragmented objects with small pieces. The algorithm extracts many potential alignments, then utilize both template guidance (if available) and groupwise fracture region matching among multiple pieces to prune and refine these alignments and obtain final reassembly. The pipeline is practically effective in composing small pieces that lack geometric saliency and have small overlap regions with adjacent pieces. Our method is demonstrated effective and robust in restoring various fragmented ceramic objects and skull models in forensic cases.

Limitations. In the current algorithm, only one template can be used to guide the reassembly. Incorporating information from multiple template sources may improve the flexibility and accuracy of the reassembly.

Acknowledgements

This research is partly supported by National Science Foundation IIS-1320959 and IIS-1251095.

References

[1] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH'08*, pages 85:1–85:10, 2008.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *JACM*, 1998.

[3] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops, IEEE Intl Conf. on*, pages 1626–1633, 2011.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI, IEEE Trans. on*, 24(4):509–522, 2002.

[5] P. Besl and N. D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, 1992.

[6] U. Castellani, M. Cristani, S. Fantoni, and V. Murino. Sparse points matching by combining 3d mesh saliency with statistical descriptors. *CGF*, 27(2):643–652, 2008.

[7] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[8] D. B. Cooper, A. Willis, S. Andrews, J. Baker, Y. Cao, D. Han, K. Kang, W. Kong, F. F. Leymarie, X. Orriols, S. Velipasalar, E. L. Vote, M. S. Joukowsky, B. B. Kimia, D. H. Laidlaw, and D. Mumford. Assembling virtual pots from 3d measurements of their fragments. In *Virtual reality, archeology, and cultural heritage*, pages 241–254, 2001.

[9] H. da Gama Leitao and J. Stolfi. A multiscale method for the reassembly of two-dimensional fragmented objects. *PAMI, IEEE Trans. on*, 24(9):1239–1251, 2002.

[10] S. A. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum*, 20(3):500–511, 2001.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*.

[12] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Proc. SGP*, 2005.

[13] D. Goldberg, C. Malon, and M. Bern. A global approach to automatic solution of jigsaw puzzles. In *In Proc. Conf. Computational Geometry*, pages 82–87, 2002.

[14] M. T. Goodrich, J. S. Mitchell, and M. W. Orletsky. Practical methods for approximate geometric pattern matching under rigid motions:(preliminary version). In *Proc. of 10th symp. on Computational geometry*, pages 103–112. ACM, 1994.

[15] Q.-X. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. In *Proc. ACM SIGGRAPH*, pages 569–578, 2006.

[16] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *PAMI, IEEE Trans. on*, 21(5):433–449, 1999.

[17] W. Kong and B. Kimia. On solving 2d and 3d puzzles using curve matching. In *CVPR, IEEE Computer Society Conference on*, volume 2, pages II–583–II–590 vol.2, 2001.

[18] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Proc. ICRA*, pages 3607–3613, 2011.

[19] X. Li and S. S. Iyengar. On computing mapping of 3d objects: A survey. *ACM Comput. Surv.*, 47(2):34:1–34:45, 2014.

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

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

[22] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *Int. J. Comput. Vision*, 89(2-3):348–361, 2010.

[23] G. Papaioannou and E. aggeliki Karabassi. On the automatic assemblage of arbitrary broken solid artefacts. *Image & Vision Computing*, 21:401–412, 2003.

[24] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Comp. Graph. Forum*, 22(3):281–289, 2003.

[25] S. Ruiz-Correa, L. Shapiro, and M. Melia. A new signature-based method for efficient 3-d object recognition. In *CVPR*, volume 1, pages 769 – 776, 2001.

[26] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[27] R. M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proc. of the fifth Eurographics SGP*, pages 225–233, 2007.

[28] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proc. SGP*, pages 1383–1392, 2009.

[29] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Proc. of the 11th ECCV: Part III*, pages 356–369, 2010.

[30] F. Tombari, S. Salti, and L. DiStefano. Performance evaluation of 3d keypoint detectors. *International Journal of Computer Vision*, 102(1-3):198–220, 2013.

[31] L. Wei, W. Yu, M. Li, and X. Li. Skull assembly and completion using template-based surface matching. In *3DIMPVT, Intl Conf. on*, pages 413–420, May 2011.

[32] A. Willis and D. Cooper. Estimating a-priori unknown 3d axially symmetric surfaces from noisy measurements of their fragments. In *3D Data Processing, Visualization, and Transmission, Third Intl Symposium on*, pages 334–341, 2006.

[33] A. R. Willis and D. B. Cooper. Bayesian assembly of 3d axially symmetric shapes from fragments. *CVPR*, 2004.

[34] Z. Yin, L. Wei, M. Manhein, and X. Li. An automatic assembly and completion framework for fragmented skulls. In *Int'l Conf. on Computer Vision*, pages 2532–2539, 2011.

[35] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. Surface feature detection and description with applications to mesh matching. In *Computer Vision and Pattern Recognition*, pages 373 –380, 2009.

[36] Y. Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Int'l Conf. on Computer Vision Workshops*, pages 689–696, 2009.