Reassembling Fractured Objects by Geometric Matching

Qi-Xing Huang, Tsinghua University
Simon Flöry, TU Wien
Natasha Gelfand, Stanford University
Michael Hofer, TU Wien
Helmut Pottmann, TU Wien

Abstract

We present a system for automatic reassembly of broken 3D solids. Given as input 3D digital models of the broken fragments, we analyze the geometry of the fracture surfaces to find a globally consistent reconstruction of the original object. Our reconstruction pipeline consists of a graph-cuts based segmentation algorithm for identifying potential fracture surfaces, feature-based robust global registration for pairwise matching of fragments, and simultaneous constrained local registration of multiple fragments. We develop several new techniques in the area of geometry processing, including the novel integral invariants for computing multi-scale surface characteristics, registration based on forward search techniques and surface consistency, and a non-penetrating iterated closest point algorithm. We illustrate the performance of our algorithms on a number of real-world examples.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; Curve, surface, solid, and object representations.

Keywords: geometric matching, integral invariants, feature-based registration, non-penetrating alignment, 3D puzzle.

1 Introduction

In the last few years, the problem of reassembling fractured 3D objects in a fully automatic way has gained an increasing importance, due mainly to the increasingly wide-spread use of shape acquisition devices in field archeology. In many cases, reconstruction has to be based purely on the geometry of the fragments, since information like color and texture has been long lost. This makes the problem closely related to the challenging problems of shape matching and 3D scan alignment in Computer Graphics and Vision. In this paper, we present an algorithm that automatically reassembles fractured 3D objects from digital models of their fragments. Although the examples in this paper are related to applications in the area of archeology, the majority of the algorithms developed here can be applied with only trivial modifications for constructing an object from partial 3D scans and other shape matching problems.

1.1 Problem Statement and Contributions

The geometric entities we start with are 3D digital models of the solid fragments obtained by 3D laser scanning of the fragment boundary surfaces, see Fig. 1 (top left). The goal is a digitally reconstructed model as shown in Fig. 1 (right). The problem of automatic 3D reconstruction consists of several challenging parts. There is an arbitrary number of fracture surfaces per fragment, which can match over any part of their extent. Therefore, we need to identify fracture surfaces on the fragments, and find pairs of matching surfaces. In order to reassemble the entire 3D object, we need to decide which pairwise matches are correct, and find a global position for each fragment relative to other fragments. This is different from many other alignment problems, since the fragments correspond to physical objects and are not allowed to penetrate each other. Fig. 2 shows the outline of our multi-step solution procedure.

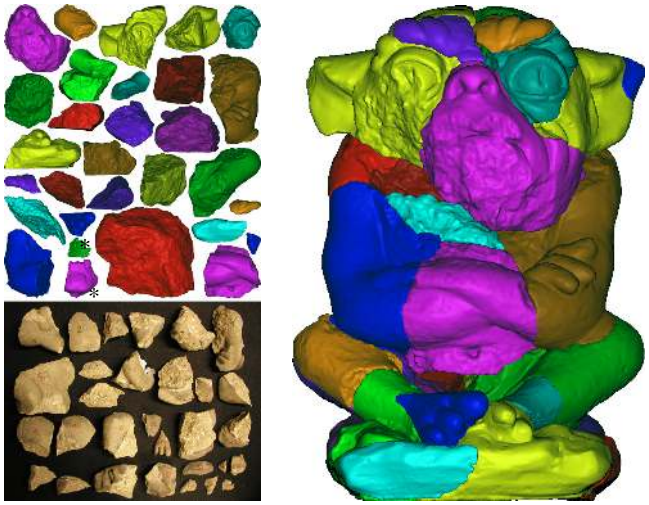


Figure 1: Reassembling a gargoyle statue: photo (bottom left) and 3D models (top left) of the fragments, final assembly (right).

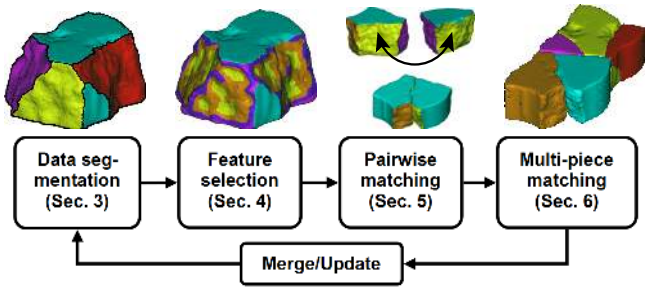


Figure 2: High level overview of our algorithm.

For the main steps of the algorithm our key contributions are:

- novel integral invariants for surfaces and 3D curves that are computed on multiple scales and are used for multi-level data segmentation and feature selection,
- robust pairwise matching using feature clusters that incorporate surface features at different scales,
- graph optimization methods for the multi-piece global matching of fragments,

- constrained optimization for both pairwise and multi-piece local registration without mutual penetration.

1.2 Algorithm Overview

The input to our algorithm is a set of point cloud surfaces representing the pieces of the fractured object. The first step of our algorithm automatically segments the fragments into a set of faces bounded by sharp curves (Sec. 3). By examining the roughness of the face surfaces, we additionally classify the faces into *original* faces, which come from the boundary surface of the unbroken object, and *fracture* faces, which were created when the object broke. This segmentation increases the robustness and efficiency of the subsequent matching algorithms in two ways. First, in the reassembly process of a 3D object only fracture faces can be matched (at least partially) against each other, and each face provides a natural grouping for a set of matching features on the surface of the fragment. Matching pairs of faces instead of considering the entire fragments, therefore, results in a more stable and faster matching algorithm. Second, we can increase the robustness of the matching algorithm by enforcing the consistent alignment of the original faces in the reassembled model. However, even in cases when the classification into original and fracture faces is not possible (e.g. when a fragment lies completely inside the object, or the original and fracture surfaces do not have distinguishing roughness), one can still successfully employ the remaining steps of our algorithm as illustrated in Fig. 2.

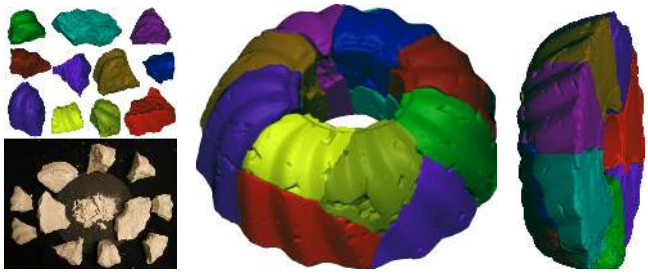


Figure 3: Reassembling a fractured cake model.

After the segmentation step we compute a novel type of patch-based surface features called *feature clusters* for all fracture faces (Sec. 4) and use these features to match all fracture faces pairwise (Sec. 5). There are several differences between our algorithm and prior feature-based matching work: we use patches instead of points as features, we allow the features to overlap, and we exploit the overlap structure to verify potential feature correspondences. We find all potential matches between pairs of faces and then verify each match using several consistency checks.

In many cases, pairwise matching is not enough to reassemble the complete 3D object since some of the matches may be incorrect (see Fig. 9(b)). Additionally, small errors in pairwise matches can accumulate incrementally resulting in the fragments not fitting together, as in Fig. 3 where pairwise matching alone is not enough to guarantee that the beginning and end of the chain of fragments will match without penetration. It is necessary therefore that all matching fragments that have been found so far undergo a simultaneous local registration to find a consistent set of matching faces and mutual fragment positions such that they do not penetrate each other (Sec. 6). The multi-piece matching is also necessary since some pairwise matches have a better chance to be found and verified after previously found matches were precisely registered onto each other and merged together (the merge/update step in Fig. 2), see e.g. Fig. 4. In our examples it suffices to perform the four major steps of our algorithm and the update/merge outlined in Fig. 2 at most 3 times until each object is reassembled.

Figure 4: The sixth piece has a better chance to be matched after the first five have been matched and simultaneously registered.

1.3 Related Previous Work

Reassembly of broken objects. Most of the work in this area is motivated by the challenge of reassembling broken archeological artifacts. Several approaches have been developed for specialized reconstruction problems such as matching planar 2D fragments (e.g. fractured tiles) [Hori et al. 1999; Kong and Kimia 2001; da Gama Leitão and Stolfi 2002; Goldberg et al. 2004] or objects that are roughly surfaces of revolution (sherds of pottery), see [Willis and Cooper 2004] and the references therein. A solution for geometric reconstruction of 3D solids was first developed by [Papaioannou et al. 2001; Papaioannou and Karabassi 2003]. The underlying assumption of this method is that the fracture faces are nearly planar and they match each other completely. After a region growing segmentation step which computed the set of fracture faces, the algorithm projects the points of each face in direction of the average face normal, and uses the resulting depth maps for matching. In contrast, our method can deal with arbitrarily shaped fracture faces, and with partial matches, which also makes it less dependent on precise segmentation. Finally, Stanford's Digital Forma Urbis Romae project [Koller and Levoy 2005] deals with heavily eroded fragments, whose fracture surfaces sometimes do not even touch each other. In this case, instead of using the geometry of the fracture faces, reconstruction is done by matching annotated incisions on the fragments' top surfaces.

Pairwise global and local registration. To reassemble a fractured 3D object we search all fragments for pairs of matching fracture surfaces first. This problem is closely related to pairwise registration (matching, alignment) of geometric models, for which an abundance of literature exists in Computer Graphics, Computer Vision, and other fields. The two major steps of pairwise registration are first *global* and then *local* alignment. Starting from arbitrary initial positions of the two pieces one first looks for a possible coarse matching. If this global registration is successful, then the second step aims at improving the mutual spatial position using local registration. Nearly all *global pairwise matching* algorithms are feature based, where different features are chosen in such a way that they are rare with respect to some descriptor value. Various descriptors have been proposed including curvatures [Li and Guskov 2005; Gal and Cohen-Or 2006] and integral invariants [Gelfand et al. 2005]. Related are also spin images [Johnson and Hebert 1999] that have been used successfully for object recognition. After selection, features are matched according to their descriptor values and a few correct feature correspondences are used to compute a coarse alignment. In current approaches this alignment is computed either by combinatorial optimization [Gelfand et al. 2005; Sara et al. 2005] or by RANSAC algorithms [Li and Guskov 2005; Shan et al. 2004]. We introduce a novel approach based on the *forward search method* [Atkinson et al. 2004], which is more robust to the presence of incorrect correspondences and also more efficient. If a coarse alignment of two pieces can be found, their mutual spatial position is improved using *pairwise local registration*. The pioneering work

in this field is the iterative closest point algorithm (ICP) for which various variants exist, see [Rusinkiewicz and Levoy 2001] for a survey and classification. Since fragments can not interfere with each other, we need a novel kind of constrained local registration that avoids mutual penetration of digital fragments.

Multi-piece global and local registration. While pairwise matching proposes a set of candidate matching pairs of fragments, we need to employ multi-piece global and local matching algorithms to reconstruct the entire original object. *Multi-piece global matching* algorithms are typically graph-based. For a given reassembly problem, a graph is build with the fragments (or scans) as nodes, and the pairwise matches between the fragments (computed by one of the algorithms above) as edges. The goal is to compute the set of edges that results in the best possible reconstruction of the object. To the best of our knowledge, the first work in this direction is [Huber 2002] which finds a global matching by computing the best spanning tree in this graph, and proves also that multi-piece global matching is NP-hard. In addition to the work on multi-piece matching done in the Computer Graphics literature, a similar problem is commonly encountered in protein docking, see e.g. [Inbar et al. 2005]. After the global matching produces the roughly aligned pieces, the role of *multi-piece local registration* is to refine the relative poses. Literature on this topic usually uses the term 'multi-view' registration since the purpose of previous algorithms was the registration of several 3D scans (views) of the same object (see e.g. [Neugebauer 1997]). Prior art for multi-view local registration are the doublet based methods [Pulli 1999; Sharp et al. 2004] that make use of the results from pairwise registration to iteratively merge scans, and numerical optimization methods [Krishnan et al. 2005], that define an energy function to be minimized where the rigid body transformations of all scans are the variables.

2 Preliminaries

2.1 Integral Invariants

Our segmentation and pairwise matching algorithms are based on comparing local curve and surface descriptors computed for points on the fragment surfaces. We are interested in quantities that are related to concepts of differential geometry, such as curvature, but that can be computed robustly and on multiple scales. Simply put, *integral invariants* are defined by integrating spatial functions over moving domains centered at surface points. Here we briefly introduce the concept, a detailed treatment of theory and computation and further references can be found in [Pottmann et al. 2005].

Surface integral invariants. Suppose the surface Φ under consideration (e.g. the face of a fragment) is the boundary of a domain D in $\mathbb{R}^3$. We use the *characteristic function* χ_D, which is 1 for points of D and 0 elsewhere, and the *squared distance function* $d^2(\mathbf{x},\Phi)$ that gives the squared distance between a point $\mathbf{x}$ and the surface Φ. $B_r(\mathbf{p})$ denotes a ball of radius r, centered at the point $\mathbf{p}$, with bounding sphere $S_r(\mathbf{p})$. Integral invariants computed at a specific radius are called *descriptors*. For a point $\mathbf{p} \in \Phi$, we define the *volume descriptor* $V^r(\mathbf{p})$ [Gelfand et al. 2005] and the *volume distance descriptor* $VD^r(\mathbf{p})$ with respect to the kernel radius r as

$$V^r(\mathbf{p}) = \frac{3}{4\pi r^3}\int_{B_r(\mathbf{p})} \chi_D d\mathbf{x}, \quad VD^r(\mathbf{p}) = \frac{15}{4\pi r^5}\int_{B_r(\mathbf{p})} d^2(\mathbf{x},\Phi) d\mathbf{x}. \tag{1}$$

In other words, $V^r(\mathbf{p})$ is the ratio between the volume of the intersection $B_r(\mathbf{p}) \cap D$ and the volume of the entire ball $B_r(\mathbf{p})$, and $VD^r(\mathbf{p})$ is the weighted integral of the squared distance function over the entire ball $B_r(\mathbf{p})$ (see Fig. 5). Note that $V^r(\mathbf{p}) = 1/2$ and

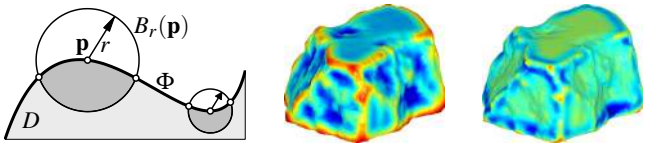


Figure 5: (Left) Intersection of the kernel balls $B_r(\mathbf{p})$ with the domain D bounded by the surface Φ. (Middle) Volume descriptor, and (right) volume distance descriptor, both shown for one scale.

$VD^r(\mathbf{p}) = 1$ if the patch of Φ contained in $B_r(\mathbf{p})$ is planar. $V^r(\mathbf{p})$ is related to the mean curvature H, and $VD^r(\mathbf{p})$ to the difference of the principal curvatures κ_1, κ_2 at $\mathbf{p}$. As $r \to 0$, we can express these relations as follows (for a proof see [Pottmann et al. 2005]):

$$V^r(\mathbf{p}) = \frac{1}{2} - \frac{3}{16}H \cdot r + O(r^2),\; VD^r(\mathbf{p}) = 1 - (\kappa_1 - \kappa_2)^2 \cdot \frac{r^2}{28} + O(r^3).$$

Spatial curve integral invariants. Let $c \subset \mathbb{R}^3$ be a spatial curve, such as an edge of a fragment. We define the *deviation descriptor* $D^r(\mathbf{p})$ at a curve point $\mathbf{p} \in c$ with respect to the kernel radius r as,

$$D^r(\mathbf{p}) = \frac{1}{r^2}\int_{x=0}^{r} \|\mathbf{c}_x(\mathbf{p}) - \mathbf{d}_x(\mathbf{p})\| dx, \tag{2}$$

where $\mathbf{c}_x(\mathbf{p})$ and $\mathbf{d}_x(\mathbf{p})$ are the first two points obtained by intersecting the sphere $S_x(\mathbf{p})$ with the curve c when going left and right from the point $\mathbf{p}$ on c. For a straight line c we have $D^r(\mathbf{p}) = 1$. In general, as $r \to 0$, $D^r(\mathbf{p})$ is related to the curvature κ of c at $\mathbf{p}$,

$$D^r(\mathbf{p}) = 1 - \frac{\kappa^2}{16} \cdot r^2 + O(r^3). \tag{3}$$

2.2 Multi-scale Surface Characteristics

For data segmentation we use the following multi-scale surface characteristics computed using the above integral invariants.

Surface sharpness. Based on the integral invariants $V^{r_i}(\mathbf{p})$ of Equ. (1) we define the *surface sharpness* $s_{vol}(\mathbf{p})$ as,

$$s_{vol}(\mathbf{p}) = \Big[\frac{1}{N}\sum_{i=1}^{N}(V^{r_i}(\mathbf{p}) - \frac{1}{2})^2\Big]^{\frac{1}{2}}. \tag{4}$$

We set $r_i = r_{min} + i \cdot (r_{max} - r_{min})/(N-1)$ for user specified r_{min} and r_{max} at N, typically 6 to 8, equidistant scales. We give details on the choice of r_{min}, r_{max} and on any thresholds and parameters defined below in the discussion of experiments in Sec. 7. Note that via $V^{r_i}(\mathbf{p})$ we incorporate mean curvature information of the surface at multiple scales into the surface sharpness characteristic. For example, s_{vol} vanishes if the neighborhood of the point $\mathbf{p}$ is planar. Surface sharpness will be used in Sec. 3 to segment the fragment into a set of faces, since for points $\mathbf{p}$ on the break curves between faces we have a high value of $s_{vol}(\mathbf{p})$.

Surface roughness. In order to differentiate between original and fracture faces, we introduce the *surface roughness* characteristic. When speaking of a surface Φ in the following, we mean the discrete set of measurement points from 3D scanning. Additionally, we write $|P|$ for the number of points of a subset $P \subset \Phi$. Let $\mathbf{q}_i$ be the k-nearest neighbors of a point $\mathbf{p} \in \Phi$, and let $\mathbf{n}_\mathbf{p}$ and $\mathbf{n}_{\mathbf{q}_i}$ be the surface normal vectors at these points. Then we define the local *bending energy* $e_k(\mathbf{p})$ at $\mathbf{p}$ as

$$e_k(\mathbf{p}) = \frac{1}{k}\sum_{i=1}^{k}\frac{\|\mathbf{n}_\mathbf{p} - \mathbf{n}_{\mathbf{q}_i}\|^2}{\|\mathbf{p} - \mathbf{q}_i\|^2}. \tag{5}$$

Notice that we could incorporate surface curvature information using the integral invariants of Equ. (1). However we found, that at the small scales needed to compute the surface roughness, the integral invariants become expensive to compute and unstable. Therefore, we integrate the above bending energy over the local neighborhood $N_r(\mathbf{p}) = B_r(\mathbf{p}) \cap \Phi$ instead,

$$\overline{e}_{k,r}(\mathbf{p}) = \frac{1}{|N_r(\mathbf{p})|} \sum_{\mathbf{q} \in N_r(\mathbf{p})} e_k(\mathbf{q}). \quad (6)$$

The characteristic value $\overline{e}_{k,r}(\mathbf{p})$ varies according to the number k of nearest neighbors and the kernel radius r. To ensure that $\overline{e}_{k,r}(\mathbf{p})$ reflects the actual kind of surface, we should choose a suitable k based on the correct local structure, and a suitable r based on the noise level. As we do not know the surface structure and noise, we assume that the correct $\overline{e}_{k,r}(\mathbf{p})$ will give a better classification result into original and fracture surfaces. For this purpose we manually select two groups of points from both original and fracture surfaces and build a statistical model of $\overline{e}_{k,r}(\mathbf{p})$ for both surface classes using supervised learning [Duda et al. 2000]. The optimal parameters k_0 and r_0 are determined for each fractured 3D object according to the classification error. The binary classification result $\rho(\mathbf{p})$ is called *surface roughness* characteristic at $\mathbf{p}$; we set $\rho(\mathbf{p}) = 1$ for an original and $\rho(\mathbf{p}) = 0$ for a fracture surface point.

3 Data Segmentation

The first stage of our reassembly algorithm segments the surface of each fragment into a set of faces. We first perform a multi-scale edge extraction on the point sampled surfaces that represent the fragments of a 3D object. By constraining the multi-scale edge extraction to return cycles of edges, we achieve an initial segmentation of each fragment into faces. Once this initial set of faces is computed, we use a graph cut algorithm to partition the set into the *original* faces and *fracture* faces. Additionally, since the multi-scale edge extraction sometimes results in an over-segmentation of a fragment into too many fracture faces, we improve the initial segmentation by merging together some of the adjacent fracture faces as dictated by the graph cut algorithm.

3.1 Multi-scale Edge Extraction

For multi-scale edge extraction we use a modified version of the method presented by [Pauly et al. 2003]. The details of the algorithm can be found in the above paper. Here we just mention the two modifications that we made to the algorithm. In the classification step we replace their 'surface variation' by our surface integral invariants and surface roughness measure. This means essentially that we classify points $\mathbf{p}$ as edge points if they have persistent high curvatures at multiple scales as measured by the integral invariants $V^r(\mathbf{p})$ and $VD^r(\mathbf{p})$ of Equ. (1), and high variance of ρ in the neighborhood of $\mathbf{p}$. The second modification of Pauly's method concerns the reconstruction step of the algorithm. After constructing the minimum spanning graph of the edge points, we specifically extract the long closed cycles from the graph. The cycles are not a by-product of the algorithm for us, but the main goal of our multi-scale edge extraction, since they frame the boundaries of the fragments' faces.

3.2 Final Segmentation

Since we focused on the extraction of edges that form closed cycles we already obtain an initial segmentation of each fragment into a set of faces F_i. We now refine this segmentation to produce a meaningful set of faces for matching, see Fig. 6. We begin with the definition of a weighted graph $G(F,E)$. The set of nodes F of the graph G are the faces F_i. The set E of edges is given by those pairs of faces F_i and F_j with a common border edge. We assign to each edge in E two weights, namely the *surface roughness weight* w_r and the *surface sharpness weight* w_s,

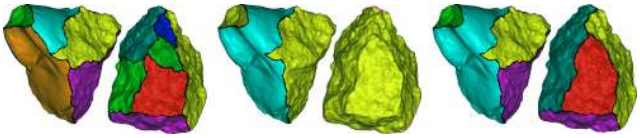


Figure 6: Data segmentation of two fragments. (Left) Initial segmentation. (Middle) Segmentation into original and fracture surfaces. (Right) Final segmentation of fracture surfaces.

$$w_r = |\overline{\rho}(F_i) - \overline{\rho}(F_j)|, \quad w_s = \frac{1}{|F_i \cap F_j|} \sum_{\mathbf{p} \in F_i \cap F_j} s_{vol}(\mathbf{p}).$$

Here we denote by $\overline{\rho}(S)$ the mean surface roughness of all points contained in the set S. These weights have the following meaning: w_r is the difference between the mean surface roughness of adjacent faces F_i and F_j (and thus will be high between original and fracture faces), and w_s is the mean surface sharpness of all points that are shared by faces F_i and F_j (and thus will be high for a sharp edge between two faces). Now we iteratively apply the *normalized cut method* of [Shi and Malik 2000] to the weighted graph $G(F,E)$. With a first series of normalized cuts we partition $G(F,E)$ into the set P_1 of original faces and the set P_2 of fracture faces, as follows. We use the surface roughness weight w_r and terminate the partitioning of the graph $G(F,E)$ if the *surface roughness variance* $\sigma^2_\rho(P_k)$,

$$\sigma^2_\rho(P_k) = \frac{1}{|P_k|} \sum_{\mathbf{p} \in P_k} [\rho(\mathbf{p}) - \overline{\rho}(P_k)]^2, \quad k = 1,2,$$

is less than 0.3 for each part P_1, P_2. The multi-scale edge extraction sometimes results in an over-segmentation of the fragment into too many faces (see Fig. 6 (left)). Using the graph cut method we can merge those faces that are likely to belong to one larger fracture face. Therefore, we perform a second series of normalized cuts of the graph $G(F,E)$. We further partition the set P_2 of fracture faces into meaningful groups, such that the weight w_s is small between adjacent faces in a group, but large for the cuts separating different groups. Using the surface sharpness weight w_s as a criterion, we iteratively subdivide the groups of fracture faces into two subgroups along the cut of highest total w_s. We obtained the final segmentation once the graph-cut threshold w_s falls below 0.1.

4 Feature Selection and Representation

Our pairwise matching algorithm is based on matching similar features of fracture surfaces. Each surface feature is a cluster of points with similar descriptor values (we will use the terms feature and cluster interchangeably). Our feature selection algorithm produces a dense set of such features, computed for a set of multi-scale integral invariants. Two key differences of our algorithm from prior work are first that we use *feature clusters*, and second the property that the feature clusters *overlap* with each other. We keep track of which clusters overlap, and use the structure of the overlaps in the matching algorithm to discard or verify feature correspondences.

4.1 Feature Selection

The key idea of our feature selection for fracture surfaces is very simple. Suppose we have a series of descriptors $\{g_1,\ldots,g_n\}$, which are functions defined on a surface Φ (e.g. a set of integral invariants as defined in Sec. 2.1). We use the level sets of these functions to partition Φ into surface patches. These patches and combinations of them are used as features (also called feature *clusters* henceforth). We first introduce feature selection using a single descriptor g, and then we proceed with multiple descriptor based feature selection.

Feature selection using single descriptor. Given a descriptor g whose range is $I_g = [0,b]$, we partition I_g into equally sized intervals $[l_i,l_{i+1}]$, usually 32 in number. For each pair of levels $l_i < l_j$, we denote the set of surface points $\mathbf{p}$ whose descriptor values $g(\mathbf{p})$ fall into the interval $[l_i,l_j]$ by S_{ij} (see Fig. 7). Starting from an arbitrary point of S_{ij} and using depth-first search on the K-nearest neighbors, we cluster the points of S_{ij} into C_{ij}^k (this clustering is analogous to extracting connected components, but performed in a point cloud setting). Then we remove those clusters C_{ij}^k where either $\min_{\mathbf{p}} g(\mathbf{p}) > l_{i+1}$ or $\max_{\mathbf{p}} g(\mathbf{p}) < l_{j-1}$ holds for a $\mathbf{p} \in C_{ij}^k$ because they are redundant. This selection process produces a dense set of overlapping feature clusters, since clusters corresponding to larger intervals contain clusters corresponding to smaller intervals.

Feature selection using multiple descriptors. We first order the descriptors $g_1,\ldots,g_n$ according to their associated scales (kernel radii) r_i, i.e., $g_i > g_{i+1}$ if $r_i > r_{i+1}$. Later in the matching algorithm, we will use feature correspondences produced by descriptors of small scales to verify feature correspondences produced by descriptors of large scales. We select features using the single descriptor algorithm described above for each g_i. We designate as C_{g_i} a cluster produced by the single descriptor feature selection algorithm for descriptor g_i.

The above process is repeated for computing feature clusters from points that belong to the boundaries between the fragment faces, as computed by the segmentation algorithm in Sec. 3. Therefore, the feature selection algorithm produces a set of fracture *surface features* and a set of fracture *edge features*. The descriptors we use are based on the integral invariants introduced in Sec. 2.1. For selecting fracture surface features we use two volume descriptors $g_1 = V^{r_{\max}}$, $g_3 = V^{r_{\min}}$, and two volume distance descriptors $g_2 = VD^{r_{\max}}$, and $g_4 = VD^{r_{\min}}$ (see Equ. (1)), and we discard descriptors near the surface boundary. For fracture edge features we use deviation descriptors $g_1 = D^{r_{\max}}$ and $g_2 = D^{r_{\min}}$ (see Equ. (2)). The use of different scales makes the descriptors robust to noise.

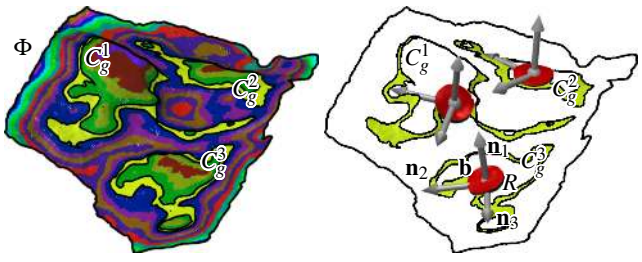


Figure 7: Feature selection and representation. (Left) Color coded sets $S_{i,i+1}$ for a single descriptor g on a fracture surface Φ. (Right) Three feature clusters C_g^k, $k = 1,2,3$ and the representation via barycenter $\mathbf{b}$, vectors $\mathbf{n}_j$, and set R (disc like), see Equ. (7).

Feature cluster topology. The above feature selection process produces a large number of overlapping feature clusters. In order to efficiently process feature correspondences in the matching stage of our algorithm, we will define a topology on the feature clusters. We call two clusters C and D *neighbors* if they have any points in common. Notice that C and D can be clusters that correspond to the same descriptor, or two different descriptors. In the case of feature clusters corresponding to two different descriptors, g_i and g_j, we additionally call C_{g_j} a *parent* of C_{g_i} if the kernel radius $r_j > r_i$.

4.2 Feature Representation

In order to use the selected feature clusters for matching fragment faces, we derive a concise representation of each cluster using a set of characteristic values, vectors, and points. We perform principal component analysis (PCA) on all points in each feature cluster C. The principal components and corresponding principal directions are denoted by $\lambda_1 < \lambda_2 < \lambda_3$ and $\mathbf{n}_1,\mathbf{n}_2,\mathbf{n}_3$, respectively. Moreover, the $\mathbf{n}_j$ are put in a right-handed coordinate system.

Representation of fracture surface features. We store the following information for each cluster C:

$$\mathsf{C} = \{\mathbf{b}(C), \mathbf{n}_j(C), \mathbf{p}_k^{\pm}(C), R(C), \mu_i(C)\}. \tag{7}$$

- The point $\mathbf{b}(C)$ represents the *position* of the feature (for which we choose the barycenter of the point set C). Notice that $\mathbf{b}(C)$ does not necessarily lie on the fracture surface.
- The vectors $\mathbf{n}_j(C)$ are directions associated with C as given by the PCA.
- The points $\mathbf{p}_k^{\pm}(C) = \mathbf{b}(C) \pm l_d\ \mathbf{n}_k(C)$ where $k = 1,2,3$. They are used for rough registration of feature clusters.
- The set $R(C)$ is a collection of representative surface points chosen from C, defined as $R(C) = B_{r_{max}}(\mathbf{b}(C)) \cap \Phi$. This point set is used for the fine registration of surface features.
- Finally, $\mu_i(C)$ contains four signatures of the cluster C. We call the values l_i and l_j of the descriptor level sets corresponding to C *descriptor signatures*. We also compute the *size signature* $\mathrm{sig}_S(C)$ and the *anisotropy signature* $\mathrm{sig}_A(C)$,

$$\mathrm{sig}_S(C) = (\lambda_1+\lambda_2+\lambda_3)^{\frac{1}{2}}, \quad \mathrm{sig}_A(C) = |\lambda_2/\lambda_3|^{\frac{1}{2}}. \tag{8}$$

The relatively low dimension of our features allows a fast retrieval of matching parts and suits better for fracture surfaces (with rather similar local neighborhoods) than other, richer descriptors such as shape contexts or spin images.

Representation of fracture edge features. Edge features are represented similar to surface features. The barycenter, set $R(C)$, descriptor signatures, and size signature are computed analogously to the surface case. The *anisotropy signature* for an edge feature C is defined as $\mathrm{sig}_A(C) = |\lambda_1/\lambda_3|^{1/2}$. The *direction vectors* $\mathbf{n}_j$ for an edge feature are defined as follows. By performing PCA on C we obtain $\mathbf{n}_1$ as that principal direction pointing along the fracture edge. $\mathbf{n}_2$ is given as the principal direction of $R(C)$ corresponding to the smallest eigenvalue. The points $\mathbf{p}_k^{\pm}(C)$ are computed for $k = 1,2$ only. If an edge feature C belongs to the break curve of a fracture surface and an original surface, we additionally assign it an *angle signature* $\mathrm{sig}_a(C) = \angle(\mathbf{n}_1,\overline{\mathbf{n}}_o)$, where $\overline{\mathbf{n}}_o$ is the average surface normal of points in C of the original surface.

5 Pairwise Matching

We use the features introduced in the preceding section to match the fragments pairwise. Let S and T be two faces on two fragments,

produced by the segmentation algorithm. We seek to find the set of common features on S and T and find an aligning transformation that brings S close to T. For each feature on S, we find all corresponding features on T and then extract subsets of consistent feature correspondences. The initial correspondence set is refined by several novel pruning algorithms and the final set of matching features is found robustly using a forward search algorithm. Similar to [Gelfand et al. 2005] we keep all consistent sets of matching features, instead of choosing just one best match. We then use multi-piece matching described in Sec. 6 to select the best match between a pair of faces in the global context.

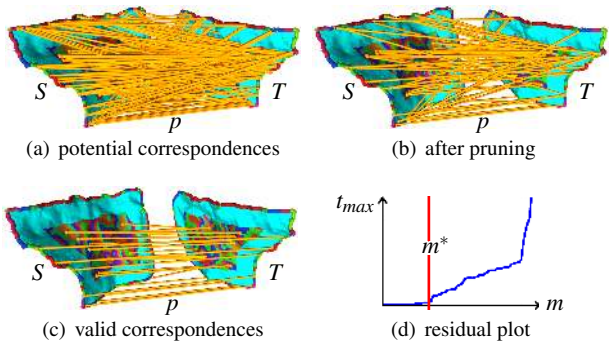
5.1 Feature Correspondences

We build the initial set of correspondences by finding for each feature cluster C in S all clusters D in T which have similar descriptor values. Two features C_{g_i} and D_{g_i} potentially correspond if they were computed using the same descriptor g_i and have the same descriptor signatures (level sets of the descriptor). For edge features, we additionally ensure the match of the angle signature, if it exists. Every edge feature correspondence with angle signatures $\mathrm{sig}_a(C_{g_i})$ and $\mathrm{sig}_a(D_{g_i})$ is required to satisfy $|\mathrm{sig}_a(C_{g_i}) + \mathrm{sig}_a(D_{g_i})| < \varepsilon_\theta$. In the remainder of the matching algorithm, we no longer make any distinction between surface and edge features. As this initial set of correspondences is quite large, we prune the set of potential correspondences by examining the feature properties of C_{g_i} and D_{g_i} and the associated feature topology. This pruning makes the succeeding forward search method (which is of quadratic complexity) run faster, but does not effect the correctness of our algorithm. For this reason, we can be rather conservative on setting any pruning thresholds without much need to tune them (see Sec. 7).

Shape pruning. Let $p = (C, D)$ be a potential feature correspondence between faces S and T respectively. We assess the quality of p by comparing the cluster signatures of C and D. We define $SD(p) = \left|\frac{\mathrm{sig}_S(C)-\mathrm{sig}_S(D)}{\mathrm{sig}_S(C)+\mathrm{sig}_S(D)}\right|$ and $AD(p) = \left|\frac{\mathrm{sig}_A(C)-\mathrm{sig}_A(D)}{\mathrm{sig}_A(C)+\mathrm{sig}_A(D)}\right|$ as the *size deviation* and the *anisotropy deviation* of two clusters. A feature correspondence p is considered further only if $SD(p) \le \varepsilon_s$ and $AD(p) \le \varepsilon_a$ hold, otherwise we discard it.

Topological pruning. We use the topology of the feature clusters to discard redundant and false correspondences and to verify the correct ones. The pruning works in two steps. In the first step, we consider feature correspondence pairs that were generated by *different descriptors*. For each feature correspondence $p = (C_{g_j}, D_{g_j})$ we find all feature correspondences $p_i = (C^i_{g_k}, D^i_{g_k})$ where $C^i_{g_k}$ and $D^i_{g_k}$ are parent clusters of C_{g_j} and D_{g_j}, respectively (this will only happen if the corresponding clusters overlap, and the kernel radius $r_k > r_j$). If the set $\{p_i\}$ is not empty, we mark all feature correspondences in $\{p_i\}$ and 'hide' p. After we perform this operation for all feature cluster correspondences, we remove all hidden and unmarked correspondences. The motivation behind this operation is that correspondences with smaller radius descriptors verify overlapping correspondences with larger radius descriptors, but larger radii are used for matching since these descriptors have better noise tolerance. Additionally, correspondences whose features are not verified by their children are likely to be incorrect, and are thus removed.

In the second step, we consider correspondence pairs with overlapping clusters corresponding to the *same descriptor*. For each pair of feature correspondences $p_1 = (C^1, D^1), p_2 = (C^2, D^2)$ such that C^1 is a neighbor of C^2 and D^1 is a neighbor of D^2, and p_1 and p_2 are geometrically consistent (see Sec. 5.2), we remove the one in p_1, p_2 that has larger average size signature. Having both, p_1 and p_2, is redundant, and always keeping the larger one would reduce the set of potential correspondences too much for successful matching.



(a) potential correspondences (b) after pruning (c) valid correspondences (d) residual plot

Figure 8: Pairwise matching. The potential feature correspondences $\{p\}$ between faces S and T are pruned; valid correspondences are extracted using a forward search. The iteration step m^*, when outliers become included, is clearly seen in the residual plot.

5.2 Potential Matches

After selecting high quality feature correspondences we face the task of finding possible matches between fracture faces S and T. We regard this problem as a fitting problem, where the data points to be fitted are feature correspondences. As incorrect feature correspondences can be seen as masked outliers, we use the forward search method, which has successfully been used in defining point set surfaces by [Fleishman et al. 2005]. We choose a pair of feature correspondences as the starting subset for the search. Since forward search needs a noise-free initial subset, we first perform several consistency tests on pairs of correspondences, and only then build larger correspondence sets.

Geometric consistency. We consider feature correspondences pairwise. As in [Shan et al. 2004], we call a pair of feature correspondences $p_1 = (C^1, D^1)$ and $p_2 = (C^2, D^2)$ *geometrically consistent* if the displacement vectors between the features' position $\mathbf{b}(C^1) - \mathbf{b}(C^2)$ and $\mathbf{b}(D^1) - \mathbf{b}(D^2)$ are of similar length, and the angles between corresponding pairs of vectors $\angle(\mathbf{n}_k(C^1), \mathbf{n}_l(C^2))$ and $\angle(\mathbf{n}_k(D^1), \mathbf{n}_l(D^2))$ do not differ more than a certain threshold ε_θ for all possible pairs (k,l) with $k,l = 1,2,3$.

Registration consistency. All geometrically consistent correspondence pairs are further checked using local registration by minimizing the sum of squared distances between certain corresponding points related to $p_1 = (C^1, D^1)$ and $p_2 = (C^2, D^2)$. Using the quaternion method proposed by [Horn 1987], we find the optimal Euclidean transformation that maps the points $(\mathbf{b}(C^1), \mathbf{p}^{\pm}_k(C^1))$ of feature C^1 to their corresponding points $(\mathbf{b}(D^1), \mathbf{p}^{\pm}_k(D^1))$ of D^1, and similarly for C^2, D^2. The obtained initial position of the features is further improved using local registration as described by [Pottmann et al. 2006] using the whole set of data points contained in the sets $R(C)$ of all four features. We keep only those feature correspondence pairs where this registration process converges and leads to a final positioning with mean deviation below a certain threshold ε_{dev}. At the end of these tests (which are performed in order of increasing complexity so as to discard the most incorrect pairings cheaply), we are left with pairs of consistent feature correspondences. We now concentrate on building a set of consistent feature correspondences from these pairs.

Forward search. Let $P = \{p_i\}$ be the set of edge and surface feature correspondences determined in the previous paragraph. We order the correspondences in P by increasing score $AD(p_i) \cdot SD(p_i)$. We also form the set $\{M_{ij} = (p_i, p_j)\}$ of correspondence pairs that pass the geometric and registration consistency tests. The forward search iteratively builds the set of matching features as follows. Let E^m be the subset of selected feature correspondences at iteration m. We start with choosing an unmarked element of $\{M_{ij}\}$ with the minimum $i+j$ that satisfies $|\mathbf{b}(C^i) - \mathbf{b}(C^j)| > 2 \cdot l_d$ for stability reasons (see [Shan et al. 2004]) to form E^1. To form E^{m+1} we first use the correspondence pairs in E^m to compute a rigid transformation α^m that aligns face S to face T. Similar to the registration consistency step, α^m is computed by the registration of the points $(\mathbf{b}(C^l), \mathbf{p}_k^{\pm}(C^l))$ and their corresponding points $(\mathbf{b}(D^l), \mathbf{p}_k^{\pm}(D^l))$ for each correspondence $p_l \in E^m$. We define the residual of a feature correspondence $p_l = (C^l, D^l)$ as $\|\alpha^m(\mathbf{b}(C^l)) - \mathbf{b}(D^l)\|$ and include the feature correspondences with the $m+1$ smallest residuals in E^{m+1}. We define the critical point of time in the forward search, i.e., when outliers begin to be included, as that index m for which the largest residual t_{max} of E^m exceeds $2 \cdot \varepsilon_{dev}$ and denote it by m^* (see Fig. 8(d)). Finally, we output the rigid body motion specified by E^{m^*} as a potential initial alignment, set all pairs of feature correspondences of E^{m^*} as marked, and continue with the next iteration. This process is repeated until all correspondence pairs in $\{M_{ij}\}$ are marked.

Each final set of correspondences E^{m^*} represents a match between the faces S and T. We define three parameters w^d, w^e and w^f to assess the quality of the match. w^d and w^e are defined as the average deviation of the overlapping regions of fracture faces and fracture edges, respectively. We use the bidirectional closest point search method introduced in [Pauly et al. 2005] to compute overlapping regions. w^f is defined as the integral of the surface sharpness over the fragment surfaces' overlapping regions $S \cap T$, $w^f = \int_{\mathbf{p} \in S \cap T} s_{vol}(\mathbf{p}) d\mathbf{p}$. A good match is supposed to have high values of w^f, indicating a high number of 'features' (or the amount of total curvature), and small deviation errors w^d and w^e. Hence, we define the quality w of a match as

$$w = \log(w^f) - \log(w^e) - \log(w^d). \tag{9}$$

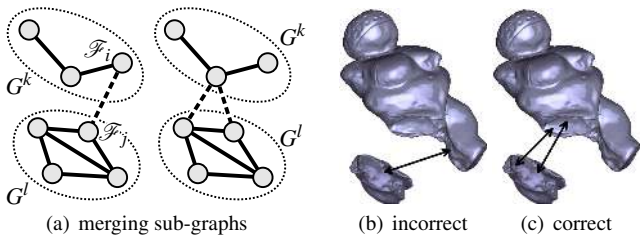
In the case when the matching faces contain edges between original and fracture faces, we enforce a surface consistency, requiring the edges to be within a threshold distance of each other.

6 Multi-piece Matching

The pairwise matching step gives a set of possible matches between the fragments of a broken 3D object along with a quality rating for each match. Based on this information we iteratively compute a global multi-piece matching, perform a local multi-piece registration and merge matched fragments until the object is reassembled.

6.1 Global Multi-piece Matching

Based on the results of the pairwise matching we define a graph $\mathscr{G}(\mathscr{F}, \mathscr{E})$, where each node $\mathscr{F}_i \in \mathscr{F}$ represents a fragment and each edge $e = (\mathscr{F}_i, \mathscr{F}_j) \in \mathscr{E}$ one of (possibly several) matches between two fragments $\mathscr{F}_i$ and $\mathscr{F}_j$. Let $w(e)$ denote the weight of such an edge and $T(e)$ its associated relative transformation. With the help of such a graph [Huber 2002] examines the global matching problem in its most general, NP-hard form. For the same problem, we introduce a well-behaved greedy algorithm, based on two observations specific for our application of reassembling broken objects. First, incorrect matches lead to heavy penetration effects and thus can easily be detected. Second, it is advantageous to design the matching algorithm iteratively as we are typically lacking sufficient correct pairwise matching information for a single step solution.



(a) merging sub-graphs (b) incorrect (c) correct

Figure 9: Different strategies for merging two sub-graphs G^k and G^l. (Left) Using the edge of maximum weight that connects these two sub-graphs. (Right) Using the group of consistent edges of maximum weight that connect these two sub-graphs.

To ease the discussion below, we introduce another weighted graph $M(\Gamma, L)$. Each node $G^k \in \Gamma$ of M is a weighted sub-graph $G^k(F^k, E^k)$ of $\mathscr{G}$ for a group F^k of fragments. Two sub-graphs G^1 and G^2 are connected in M if two fragments $\mathscr{F}_i \in F^1$ and $\mathscr{F}_j \in F^2$ exist such that $(\mathscr{F}_i, \mathscr{F}_j) \in \mathscr{E}$. The weight $w(G^1, G^2)$ of the connecting edge is defined as the sum of weights $w(e)$, see Equ. (9), of all edges $e = (\mathscr{F}_i, \mathscr{F}_j) \in \mathscr{E}, \mathscr{F}_i \in G^1, \mathscr{F}_j \in G^2$,

$$w(G^1, G^2) = \sum\nolimits_e w(e). \tag{10}$$

Similar to the incremental merging algorithm of [Huber 2002] our multi-piece global matching is an iterative process. As long as there exists more than one fragment, the graph $\mathscr{G}$ is built through exhaustive pairwise matching of the current fragments, using the algorithms described in the previous sections. Accordingly, we construct the affiliated model graph $M(\Gamma, L)$ by regarding each fragment as a single sub-graph. At each iteration, we select a group of edges in M (see details below) and merge their corresponding sub-graphs G^k and G^l into a new sub-graph. Then, we check if all the fragments in this new sub-graph do not penetrate each other and continue with the next iteration step if the merging passed this test. Otherwise, we try another group of edges.

Sub-graph merging. Let $E^{k,l}$ denote the set of edges that connect two sub-graphs G^k and G^l. A straightforward strategy [Huber 2002] would be to let every edge in $E^{k,l}$ represent a single group, sort these groups according to their weights and merge the sub-graphs by using the highest weighted edge. However, this strategy is unfavorable in some cases (see Fig. 9) and relies heavily on penetration tests to discard incorrect merges. Our approach is based on a global consistency argument, which states that the associated relative transformations of the edges of a loop in $M(\Gamma, L)$ should approximate the identity motion when applied one after another. Thus, we classify each $E^{k,l}$ into subsets $E_s^{k,l}$ such that the sub-graphs $G_s^{k,l}(F^k \cup F^l,\ E_s^{k,l} \cup E^k \cup E^l)$ are globally consistent and use the highest rated subset of edges for merging.

The edge subsets $E_s^{k,l}$ are computed using forward searches. We order the edges in $E^{k,l}$ according to their weights and select the unclassified edge of $E^{k,l}$ with maximum weight as initial set. Let E^m denote the selected edges in the m-th step of the forward search. We compute the current relative motions $A = \{\alpha_1, \ldots, \alpha_{|F^k|+|F^l|}\}$

of the fragments by minimizing the quadratic function Q that sums over all edges $e = (\mathscr{F}_i, \mathscr{F}_j) \in E^k \cup E^l \cup E^m$,

$$Q = \sum_e w(e) \cdot \left(\|\alpha_i - \alpha_j \circ T(e)\|^2_{\mathscr{F}_i} + \|\alpha_j - \alpha_i \circ T(e)\|^2_{\mathscr{F}_j} \right).$$

Thereby, $\|\cdot\|_{\mathscr{F}_i}$ is a metric for affine maps as defined in [Hofer et al. 2004]. After obtaining A we order all unclassified edges by their residuals $\|\alpha_i - T(e) \circ \alpha_j\|^2_{\mathscr{F}_i} + \|\alpha_j - T(e) \circ \alpha_i\|^2_{\mathscr{F}_j}$ and choose the edges corresponding to the $m+1$ smallest residuals for E^{m+1}. The forward search is repeated until either all unclassified edges in $E^{k,l}$ are classified or the sub-graph becomes disconnected. Again, we find that iteration m^* of the forward search in which outliers start to be included. The edges of E^{m^*} form a new subset $E_s^{k,l}$ of classified edges. Note that we only obtained the optimal affine motions α_i of the fragments in $F^k \cup F^l$. In order to get Euclidean motions, we project α_i onto the rigid body motion manifold using the method introduced in [Krishnan et al. 2005].

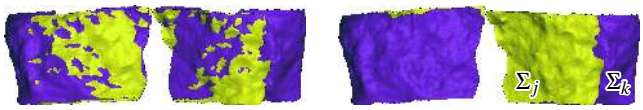
These edge subsets $E_s^{k,l}$ are ordered as follows. The key value is the size of each subset, so that we favor big subsets over smaller ones and select strong connections between G^k and G^l first. Edge subsets of the same size are sorted according to their weights as defined in Equ. (10). In this way, better matches are chosen first.

Penetration test. We employ a collision detection algorithm [Lin and Manocha 2004] between all pairs of fragments in $G_s^{k,l}$. As wrong matches yield severe penetration effects, we terminate the sub-graph merging if fragments are found to intersect with more than a predefined threshold.

Our multi-piece matching as described above is a greedy process. It will fail when the penetration test accepts incorrect pair-wise matches. However, the strong local and global consistency checks prune out incorrect matches and the algorithm achieves good results in practice.

6.2 Multi-piece Local Constrained Registration

After global multi-piece matching of the 3D fragments we move them to their final alignment by simultaneous registration. The nature of reassembling fractured 3D objects imposes an interesting new aspect on registration: as in reality, the objects's fractured parts must not penetrate each other on recomposing, thus asking for a *constrained* registration. We illustrate this aspect on two systems in Fig. 10. The penetration free registration is shown on the brick example in Fig. 11.



(a) standard registration (b) penetration free

Figure 10: Front and back view of two registered systems Σ_j, Σ_k.

We choose the fragments of a sub-graph as the distinct systems of our registration process. Let Σ_i denote the union of fracture faces of the i-th fragment $\mathscr{F}_i$. We fix Σ_0 and simultaneously register those parts that are adjacent after the global matching. In order to be able to express the distances of the elements of two nearby systems Σ_j and Σ_k, we work with the relative linearized motion of Σ_j and Σ_k,

$$\mathbf{v}_{jk}(\mathbf{x}) = \mathbf{v}_{j0}(\mathbf{x}) - \mathbf{v}_{k0}(\mathbf{x}),$$

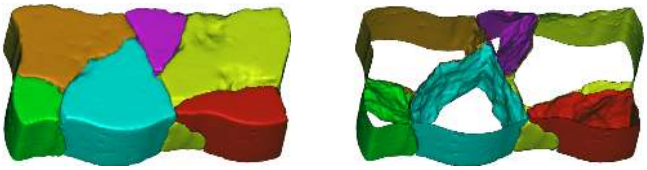


Figure 11: (Left) Final alignment of 6 brick fragments after multi-piece local constrained registration. (Right) After removing the top and bottom faces we see the penetration free fracture faces.

where $\mathbf{v}_{i0}(\mathbf{x}) = \bar{\mathbf{c}}_i + \mathbf{c}_i \times \mathbf{x}$ denotes the velocity vector of the i-th fragment towards Σ_0. Only a subset $X_{jk} \subset \Sigma_j$ of points located sufficiently close to the target point cloud is registered onto Σ_k. For $\mathbf{x}_i \in X_{jk}$, the closest point $\mathbf{y}_i \in \Sigma_k$, the normed residue $\mathbf{r}_i = (\mathbf{x}_i - \mathbf{y}_i)/(\|\mathbf{x}_i - \mathbf{y}_i\|)$ and the distance $d_i = \|\mathbf{x}_i - \mathbf{y}_i\|$ are computed. Then,

$$\tilde{d}^2(\mathbf{x}_i + \mathbf{v}_{jk}(\mathbf{x}_i), \mathbf{y}_i) = [d_i + \mathbf{r}_i^T \cdot \mathbf{v}_{jk}(\mathbf{x}_i)]^2$$

approximates the squared distance of a moved point $\mathbf{x}_i$ to $\mathbf{y}_i \in \Sigma_k$. By performing this initial step for all pairs of adjacent faces, the linearized motions $\mathbf{v}_{j0}$ are found by minimizing

$$F(\mathbf{c}_1, \bar{\mathbf{c}}_1, \ldots, \mathbf{c}_n, \bar{\mathbf{c}}_n) = \sum_{i=1}^{|X|} [d_i + \mathbf{r}_i^T \cdot \mathbf{v}_{jk}(\mathbf{x}_i)]^2. \qquad (11)$$

Here, X denotes the union of all chosen point clouds X_{jk}. Note, that the objective function is quadratic in the unknowns $\mathbf{c}_1, \bar{\mathbf{c}}_1, \ldots, \mathbf{c}_n, \bar{\mathbf{c}}_n$.

On recomposing the fractured pieces, registration has to avoid any penetration of the boundary surfaces. Thus, all elements of Σ_j must be located on the same side of an adjacent system Σ_k. Accordingly, we write $\mathbf{n}_i$ for the outward oriented normal in $\mathbf{y}_i$ and require the distances from Σ_j to Σ_k to have the same sign,

$$\mathbf{n}_i^T \cdot (\mathbf{x}_i - \mathbf{y}_i + \mathbf{v}_{jk}(\mathbf{x}_i)) \geq 0, \qquad i = 1, \ldots, |X|.$$

In total, we obtain a quadratic optimization problem with linear constraints that we solve with an *active set method* (see e.g. [Nocedal and Wright 1999]). As $\mathbf{x} \mapsto \mathbf{x} + \mathbf{v}_{i0}(\mathbf{x})$ is an affine but not a Euclidean motion, we perform the underlying helical motion on Σ_i as described in [Pottmann et al. 2006]. A final alignment without penetration is achieved after 5 to 10 iteration steps.

Fragment merging. After constrained local registration we merge the fragments of each sub-graph into a single 'virtual' fragment for further global matching. We remove the points of all matching fracture faces using the bidirectional closest point search method of [Pauly et al. 2005]. In a second step we fill any holes using a modified method of [Amenta and Kil 2004] such that the virtual fragment is a single closed surface.

7 Results and Discussion

We have created several challenging examples to test our reassembly algorithm (see Table 1 for an overview). For all models except the Forma Urbis Romae (FUR) example [Koller and Levoy 2005] the fracturing process (either dropping on the floor or hammer and chisel) returned several larger fragments, some small fragments, and quite some debris (see e.g. Fig. 3). We scanned all the fragments with a 3D laser scanner, created 3D digital models as dense point set surfaces (available at *http://www.geometrie.tuwien.ac.at/ig/3dpuzzles.html*), and estimated outward oriented surface normal vectors, which are used throughout the whole processing pipeline. For all examples, our algorithm could differ between fracture and

original surfaces. The gargoyle and the FUR example both contain noticeably large and small fragments, which shows the robustness of our method to different sizes of matching features.

Garg.	Cake	Brick	Venus	Sculp.	Head	FUR
stone	mortar	stone	clay	clay	clay	marble
30f	11f	6f	7f	15f	12f	20f
3.54m	1.45m	1.49m	1.84m	1.66m	1.15m	9.45m

Table 1: Example overview: name, material, number of fragments, and total number of fragment data points.

Examples. The gargoyle model consists of 30 fragments from which we could successfully reassemble 28 (Fig. 1). The fragments that could not be matched are marked by a tiny $*$ in Fig. 1 (left top). Some of the gargoyle fragments only possess fracture surfaces since they are completely from the interior of the original model. This additional challenge shows that we do not rely on the presence of original surfaces in our fragments.

The fragments and two views of the reconstructed cake model can be seen in Fig. 3. The reassembled 3D model of the brick and the penetration free fracture surfaces (in the final alignment position) are shown in Fig. 11. For the Venus example (Fig. 12) we could only match the 6 largest fragments. As shown in Fig. 13 we could reassemble the fractured sculpture completely.

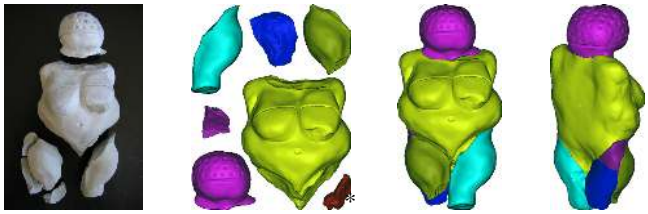


Figure 12: Reassembling a fractured Venus model.

For the FUR example we choose a set of 20 fragments with varying size from the online database *http://formaurbis.stanford.edu/*, including some fragments known to match each other. Our motivation for working with this FUR data was not to find new matches (an ambitious task for future work) but to verify that our method succeeds for heavily eroded fracture faces. We are able to find the correct matching fragments based purely on the geometry of the fracture surfaces, as shown in Fig. 14.

In Fig. 15 we show the original fragments of the head model, their 3D digital counterparts, and the reconstructed head. The head model is not a solid but a thin shell. This poses the challenge of rather small fracturing surfaces that have to be matched against each other. We could automatically match the 10 largest of the 12 fragments. The two small fragments marked by a $*$ in Fig. 15 (left top) are from a hole in the forehead (the location of impact on fracturing) and could not be matched automatically.

Timing results. On a 1.4GHz machine with 512MB RAM, we need on average 1 minute for processing a fragment with 400k points. This includes building the necessary data structures and computing the integral invariants on 8 different scales for every data point. The segmentation, feature selection, and pruning are linear in the number of data points and take around 4s per fragment. The time for pairwise and multi-piece matching depends on the number of faces. For the brick example, the set of potential matches was built in 2s while the multi-piece matching (including the dominating constrained registration) took 5s. We passed through the whole pipeline two times which took 15s.

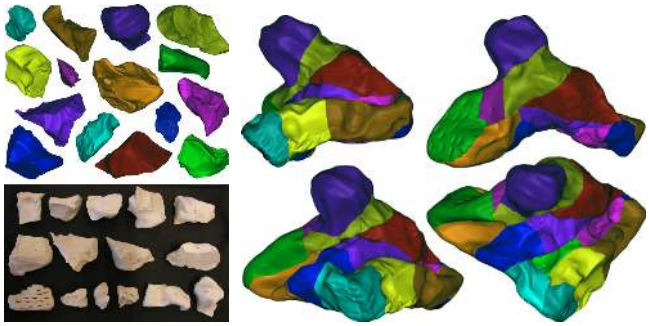


Figure 13: Reassembling a fractured sculpture model.

Choice of thresholds. Different parts of our algorithm are guided by thresholds. In general, the choice of these thresholds does not greatly influence the final results, since most of them are used for pruning, but affects the running time of the algorithm. The surface sharpness measure Equ. (4) is defined for scales ranging from r_{min} to r_{max}. We encounter these thresholds throughout this work and they are set to reflect the fragments' size. We set r_{max} to 0.1 times the average size of all fragments and $r_{min} = r_{max}/2$, whereas a fragment may be composed of several smaller fragments at later iterations of the algorithm. Our feature representation contains auxiliary points $\mathbf{p}_k^{\pm}$ for rough registration. These are defined at distance of $l_d = r_{max}/2$ from the clusters' barycenter. Feature pruning uses three hard thresholds: angle signatures are required to differ by less than $\varepsilon_\theta = 0.2$ and the size and anisotropy deviation must be below 0.1 (thresholds ε_a and ε_s). Finally, two thresholds are set to verify registration results. Registration consistency in pairwise matching requires a mean deviation of less than ε_{dev}. We do not rely on a user defined constant here but use a strategy similar to the method in [Fleishman et al. 2005] which takes into account the noise level of the surface. The penetration test of global multi-piece matching discards any matches with a penetration exceeding r_{min}.

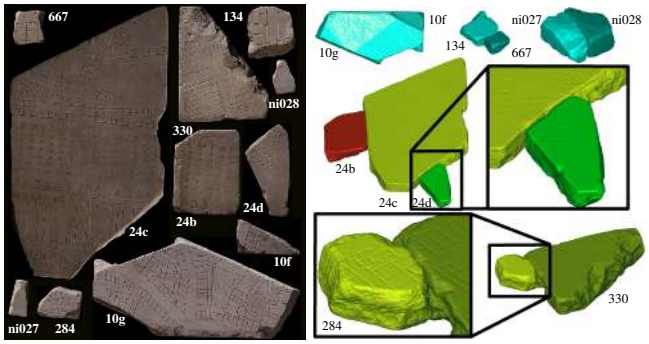


Figure 14: Matching parts of the Forma Urbis Romae project. (Left) The fragments for which we found matches based on the geometry of the fracture surfaces only. (Right) Aligned fragments.

Conclusions. We presented a method for automatic reassembly of fractured 3D solids. Our matching algorithm assembles the broken object by using purely the geometric information contained in the fracture surfaces of the fragments. It is straightforward to include additional information such as 2D and 3D textures. Although the motivating application for this work comes from archeology, we contribute several novel techniques for surface analysis and matching to Computer Graphics, Vision, and Geometry Processing.

Acknowledgements. The authors acknowledge support by a Max Planck Institute Fellowship and grants P16002, S92, and P18865

of the Austrian Science Fund (FWF). We thank the Stanford Digital Forma Urbis Romae Project for the use of the FUR models, Markus Forstner for his help on 3D scanning of the fragments, and the anonymous reviewers for their valuable comments.

References

AMENTA, N., AND KIL, Y. 2004. Defining point-set surfaces. *ACM Trans. Graph. 23*, 3, 264–270. (Proc. SIGGRAPH'04).

ATKINSON, A. C., RIANI, M., AND CERIOLI, A. 2004. *Exploring Multivariate Data With the Forward Search*. Springer.

DA GAMA LEITÃO, H. C., AND STOLFI, J. 2002. A multiscale method for the reassembly of two-dimensional fragmented objects. *IEEE Trans. PAMI 24*, 9, 1239–1251.

DUDA, R. O., HART, P. E., AND STORK, D. G. 2000. *Pattern Classification (2nd Edition)*. Wiley-Interscience.

FLEISHMAN, S., COHEN-OR, D., AND SILVA, C. T. 2005. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3, 544–552. (Proc. SIGGRAPH '05).

GAL, R., AND COHEN-OR, D. 2006. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph. 25*, 1, 130–150.

GELFAND, N., MITRA, N. J., GUIBAS, L. J., AND POTTMANN, H. 2005. Robust global registration. In *SGP'05*, 197–206.

GOLDBERG, D., MALON, C., AND BERN, M. 2004. A global approach to automatic solution of jigsaw puzzles. *Computational Geometry 28*, 2-3, 165–174.

HOFER, M., ET AL. 2004. From curve design algorithms to the design of rigid body motions. *Vis. Comput. 20*, 5, 279–297.

HORI, K., IMAI, M., AND OGASAWARA, T. 1999. Joint detection for potsherds of broken earthenware. In *Proc. CVPR*, vol. 2, 440–445.

HORN, B. K. P. 1987. Closed form solution of absolute orientation using unit quaternions. *J. Optical Society A 4*, 629–642.

HUBER, D. 2002. *Automatic three-dimensional modeling from reality*. PhD thesis, Carnegie Mellon University.

INBAR, Y., WOLFSON, H. J., AND NUSSINOV, R. 2005. Multiple docking for protein structure prediction. *The International Journal of Robotics Research 24*, 2-3, 131–150.

JOHNSON, A. E., AND HEBERT, M. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. PAMI 21*, 5, 433–449.

KOLLER, D., AND LEVOY, M. 2005. Computer-aided reconstruction and new matches in the Forma Urbis Romae. *Bullettino Della Commissione Archeologica Comunale di Roma*. to appear.

KONG, W., AND KIMIA, B. B. 2001. On solving 2D and 3D puzzles using curve matching. In *Proc. CVPR*, vol. 2, 583–590.

KRISHNAN, S., LEE, P. Y., MOORE, J. B., AND VENKATASUBRAMANIAN, S. 2005. Simultaneous registration of multiple 3D point sets via optimization on a manifold. In *SGP'05*, 187–196.

LI, X., AND GUSKOV, I. 2005. Multiscale features for approximate alignment of point-based surfaces. In *SGP'05*, 217–226.

LIN, M. C., AND MANOCHA, D. 2004. Collision and proximity queries. In *Handbook of Discrete and Computational Geometry*.

NEUGEBAUER, P. 1997. Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *Int. Journal of Shape Modeling 3*, 1-2, 71–90.

NOCEDAL, J., AND WRIGHT, S. J. 1999. *Numerical Optimization*.

PAPAIOANNOU, G., AND KARABASSI, E.-A. 2003. On the automatic assemblage of arbitrary broken solid artefacts. *Image and Vision Computing 21*, 401–412.

PAPAIOANNOU, G., KARABASSI, E.-A., AND THEOHARIS, T. 2001. Virtual archaeologist: Assembling the past. *IEEE Computer Graphics and Applications 21*, 2, 53–59.

PAULY, M., KEISER, R., AND GROSS, M. 2003. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum 22*, 3, 281–289.

PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M. H., AND GUIBAS, L. J. 2005. Example-based 3D scan completion. In *SGP'05*, 23–32.

POTTMANN, H., HUANG, Q.-X., KÖLPL, S., AND YANG, Y. 2005. Integral invariants for robust geometry processing. Tech. Rep. 146, Geometry Preprint Series, Vienna Univ. of Techn.

POTTMANN, H., HUANG, Q.-X., YANG, Y.-L., AND HU, S.-M. 2006. Geometry and convergence analysis of algorithms for registration of 3D shapes. *Intl. J. of Comp. Vision 67*, 3, 277–296.

PULLI, K. 1999. Multiview registration for large datasets. In *3DIM'99*, IEEE CS, 160–168.

RUSINKIEWICZ, S., AND LEVOY, M. 2001. Efficient variants of the ICP algorithm. In *3DIM '01*, IEEE CS, 145–152.

SARA, R., OKATANI, I. S., AND SUGIMOTO, A. 2005. Globally convergent range image registration by graph kernel algorithm. In *3DIM '05*, IEEE CS, 377–384.

SHAN, Y., MATEI, B., SAWHNEY, H. S., KUMAR, R., HUBER, D., AND HEBERT, M. 2004. Linear model hashing and batch RANSAC for rapid and accurate object recognition. In *Proc. CVPR*, vol. 2, 121–128.

SHARP, G., LEE, S., AND WEHE, D. 2004. Multiview registration of 3D scenes by minimizing error between coordinate frames. *IEEE Trans. PAMI 26*, 8, 1037–1050.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. PAMI 22*, 8, 888–905.

WILLIS, A., AND COOPER, D. B. 2004. Alignment of multiple non-overlapping axially symmetric 3D data sets. In *Proc. ICPR*, vol. IV, 96–99.

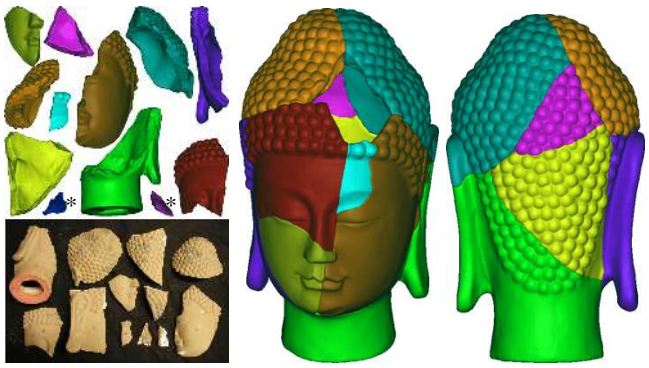


Figure 15: Reassembling a fractured head model.