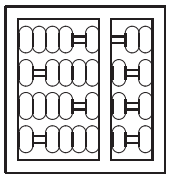

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

A Multi-Scale Technique for Computer Assisted Re-Assembly of Fragmented Objects

Helena Cristina da Gama Leitão
Jorge Stolfi

Technical Report - IC-01-004 - Relatório Técnico

March - 2001 - Março

A Multi-Scale Technique for Computer Assisted Re-Assembly of Fragmented Objects

Helena Cristina da Gama Leitão
Inst. of Computing - Fluminense Federal Univ.
R. Passo da Pátria, 156
24210-240 Niterói, RJ, Brazil
hcgl@pgcc.uff.br

Jorge Stolfi
Inst. of Computing - Univ. of Campinas
Caixa Postal 6176
13083-970 Campinas, SP, Brazil
stolfi@dcc.unicamp.br

May 1, 2001

Abstract

We describe here an efficient algorithm for reassembling one or more unknown objects that have been broken or torn into a large number N of irregular fragments. The algorithm works by comparing the curvature-encoded fragment outlines, using a modified dynamic programming sequence-matching algorithm. By comparing the outlines at progressively increasing scales of resolution, we manage to reduce the cost of the search from $\theta(N^2L^2)$ (where L is the mean number of samples per fragment) to about $O(N^2L)$; which, in principle, allows the method to be used for problems of practical size ($N = 10^3$ to 10^5 fragments, $L = 10^3$ to 10^4 samples). The performance of the algorithm is illustrated with an artificial but realistic example.

---

1 Introduction

Reassembling broken objects from a collection of thousands randomly mixed fragments is major problem for archaeologists and museum curators, which may require years of tedious and delicate work. Since manpower is usually a critical resource, the problem usually goes unsolved: most archaeological fragments are just counted by type, and then stored away in museum basements — or even re-buried on the site.

This problem also arises in other disciplines, such as paleontology, art conservation, forensics, and failure analysis; and is obviously one which could use some computer help. Indeed, computers have already been used to pre-sort the fragments according gross properties such as color, texture, material, profile, etc, so as to reduce the number of pairs to be checked. However, automatic identification of matching pairs will ultimately require geometric comparison of the fragment outlines — which is the problem that we address in this paper.

The main difficulty in this problem is the large number of fragments present in typical instances. If the average fragment contour has L sample points, then finding the best fit between two contours requires $\Theta(L^2)$ operations by standard dynamic programming techniques. Thus, processing a colection with N fragments would require $\Theta(N^2L^2)$ operations. Since computer aid is most needed for large collections, with 10^3 to 10^5 fragments, and accurate matching requires L to be between 10^2 to 10^3 samples, these brute-force methods are too expensive for practical use.

Here we describe an algorithm that performs this task at much lower cost, through the use of *multi-scale* techniques. By repeated filtering and resampling, we build several versions of each fragment outline, each about half as detailed as the previous one. At first, we look for similar outlines at the coarsest possible scale; this search is fast but not very accurate, and results in a large set of potential matches. We then repeatedly select the most promising pairs, and compare them at the next finer scale of detail. In the end, we are left with a small set of fragment pairs that are most likely to be adjacent in the original objects. As we will see, these techniques allow us to reduce the total cost from $O(N^2L^2)$ to about $O(N^2L \log L)$ [10, 11, 5].

In a companion paper [12], we address the question of whether the problem is solvable at all. We show that, for well-preserved ceramic fragments, there is enough information in a couple of centimeters' worth of fracture line to identify the matching fragment, with acceptable accuracy, even among millions of other similar fragments.

1.1 Statement of the problem

Fragments and fracture lines. We assume that the original objects had a well defined smooth surface, which was divided into two or more *ideal fragments* by irregular *ideal fracture lines* of zero width. The fractures also split the original outline of the surface into one or more *border lines*.The point where three or more lines (fracture

or border) meet is called an *ideal corner*. Two fragments are said to be *adjacent* if they share a fracture line. The boundary of an ideal fragment is an *ideal outline*; it is the concatenation of one or more fracture and border lines. See figure 1(a).

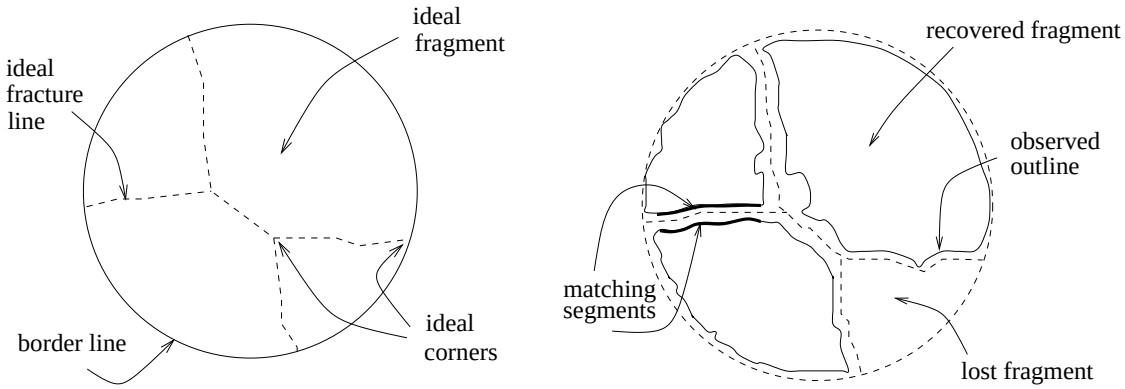

Figure 1: Ideal fracture network (a) and observed outlines (b).

Observed fragment outlines. Ideal fracture lines and outlines are mere abstractions, of course. What we obtain from digitizing the physical fragments are the *observed outlines*; they differ from the ideal outlines for a number of reasons, either real (such as loss of small fragments, dirt, surface irregularities) or artificial (such as parallax, shadowing, quantization, etc.). Because of these perturbations, the single ideal fracture line that separates two adjacent ideal fragments becomes two slightly different *matching segments* on the two observed fragment outlines. See figure 1(b).

The problem. We can now state the *fragment matching* problem as follows: given a set $\mathcal{C}$ of observed fragment outlines, identify those pairs of fragments which were probably adjacent in the original object.

2 Related work.

At present, the reconstruction of archaeological fragments is done largely by hand. Computers are used, if at all, only in the enhancement, classification, and presentation of scanned images of the fragments [2, 8, 26], which are indexed and retrieved based solely on textual descriptions provided by the user.

Computer vision and pattern matching techniques have occasionally been used to automatically extract indexing and matching information from digital images of archaeological artifacts [25, 6, 23, 14]. The specific problem of identifying adjacent ceramic fragments by matching the shapes of their outlines was recently considered

by Üçoluk and Toroslu [25]. Their algorithm considers only a fixed scale of resolution, and therefore has large expected asymptotic cost. (No real-world tests are reported in the article.) Also, Mark Levoy is currently investigating the use of 2D (surface) shape matching techniques for the same purpose [13].

The fragment reassembly problem is similar to that of *automatic assembly of jigsaw puzzles*, which has been addressed before, mainly as a cute exercise in robotics and machine vision [7, 4, 20, 29, 27]. In particular, H. Wolfson and G. C. Burdea developed a program that can find matching pieces in a standard puzzle game [4], and even control a robot arm to assemble the puzzle. However, the techniques used rely on special characteristics of puzzle pieces (smooth borders and well-defined corners), which are not found in archaeological materials. On one hand, in fractured objects, each fragment corner is generally incident to three fracture lines, and two of them usually make an angle close to 180°. Thus, a typical corner will be hardly detectable in one of its incident fragments. On the other hand, physical fracture lines tend to display sharp bends even where there are no ideal corners.

More generally, the problem can be viewed as a special case of *object recognition by approximate outline matching*, an area with extensive bibliography [19, 3]. However, in this field one typically assumes that the given outlines are to be matched against a relatively small set of fixed templates; while, in the applications that interest us, the templates are the fragments outlines themselves, which may number in the thousands. Therefore, we must discard many standard object recognition methods because they rely on extensive preprocessing of the templates.

Multi-Scale techniques have often been used for image-based and outline-based shape matching [15, 22, 21, 1, 16, 28]. However, most existing algorithms begin by identifying certain *key points* of the outlines, such as corners, curvature extrema, or inflection points. As we noted above, this approach is neither feasible nor useful for ceramic fragments. Therefore, our algorithm applies the multi-scale approach directly to the comparison of curvature graphs, without trying to identify any key points.

3 Shape matching

3.1 Outline representation.

Samples. We assume that each observed fragment outline is a circular sequence of *samples* $c_0, c_1, .. c_{n-1}$ with uniform spacing δ, the same for all outlines.

In our tests so far we have used only flat (2-dimensional) fragments. Since each fragment is independently digitized in arbitrary orientation, we define each sample value as the local curvature of the outline at the corresponding sample point — a well-known representation which is invariant under rotations [30, 22, 3, 7, 9].

The curvature can be combined with any other local properties of the fragment — such as color or thickness — that are invariant under rotation and displacement,

and can help to identify matching fragments. Since our algorithm does not depend on the nature of the samples, it can exploit such information whenever it is available.

Our algorithm can be used for non-flat objects, such as ceramic and glass vessels, provided that their surface is smooth enough to posess a well-defined tangent plane at each point along the fracture line. In that case, we could (and probably should) extend each sample value with the local surface curvature, or with the local torsion of the outline curve, as discussed by Üçoluk and Toroslu [25].

Segments and candidates. A *segment* is a sequence of one or more consecutive samples from some fragment outline. A *candidate* is a pair of segments belonging to different outlines. We say that a candidate is *true* it is segments correspond to the same ideal fracture line (or part thereof); otherwise the candidate is *false*. We denote by $\mathcal{T}$ the set of all true candidates among the given fragments. Note that on a set of N outlines with an average of L samples each we can define about NL^2 segments and N^2L^4 candidates, almost all of them false.

3.2 Candidate evaluation

Candidate mismatch. In order to evaluate the likelihood of a candidate to be true, we use a numerical measure of (dis)similarity between the shapes of the two segments. Let a and b be two segments with curvature samples $a_1..a_n$ and $b_1..b_n$. To simplify the exposition, let's suppose at first that there is a perfect pairing between the two segments; that is, the observed samples a_i and b_i derive from the same sample c_i of the ideal fracture line, for all i. (Note that a and b must be enumerated in opposite directions.) Then we define the *quadratic mismatch* of the two segments as

$$S^2(a,b) = \sum_{i=1}^{n} \|a_i, b_i\|^2 \tag{1}$$

where $\|a_i, b_i\|$ is a measure of the difference between the two samples, e.g. the absolute difference between the curvatures at those points.

Critical mismatch. Intuitively, for a fixed n, the smaller the value of $S^2(a,b)$, the more likely it is that (a,b) is a true candidate. This intuition can be supported by the following statistical analysis. Let's assume that the ideal fracture samples c_i and the disturbances that turned c_i into a_i and b_i are independent Gaussian variables, with variances ω^2 and σ^2, respectively. Then the quantity S^2 of formula (1), for randomly chosen candidates, will have a χ^2 distribution with n degrees of freedom [18]. Its mean will be $2n\sigma^2$ for true candidates, and $2n(\omega^2 + \sigma^2)$ for false ones.

Through Bayesian inference, we can compute the probability of a candidate being true, given its discrepancy S^2. More precisely, it can be shown [5] that there exists a

critical value Ξ^2 of S^2 for which the candidate is equally likely to be true or false. If, on the given outlines, we can pick M segments b which could be compared to a, and only one of them is true, then the critical value turns out to be

$$\Xi^2 \;=\; 2\frac{\phi^2\gamma^2}{\phi^2-\gamma^2}\,\ln(\phi/\gamma)\left(n-\frac{\ln(M-1)}{\ln(\phi/\gamma)}\right) \;=\; (n-n_{\min})\xi^2 \tag{2}$$

where $\gamma^2 = 2\sigma^2$ and $\phi^2 = 2(\omega^2+\sigma^2)$, and

$$\xi^2 \;=\; \frac{\phi^2\gamma^2}{\phi^2-\gamma^2}\,\ln(\phi/\gamma) \qquad n_{\min} \;=\; \frac{\ln(M-1)}{\ln(\phi/\gamma)} \tag{3}$$

Thus the candidate is likely to be false if $S^2(a,b) > \Xi^2$, and true if $S^2(a,b) < \Xi^2$.

The parameter ξ^2 is the *critical sample mismatch*, the mean value of $\|a_i, b_i\|^2$ that separates true candidates from false ones, provided they are sufficiently long ($n \gg n_{\min}$). Note that Ξ^2 is negative when the candidate has fewer than $n_{\min}$ samples; in other words, candidates with fewer than $n_{\min}$ samples are more likely to be false than true, no matter how similar the segments are.

3.3 Candidate evaluation with irregular pairing

Discrete pairing. Loss of material and other acquisition errors in the observed outlines may change the length of the matching segments by different amounts, so that we cannot expect a perfect one-to-one pairing of the samples. To address this problem, we must modify formula (1) to allow a more flexible correspondence between the samples of the two segments. We therefore define a *pairing* between the two segments $a = (a_0..a_m)$ and $b = (b_0..b_n)$ — enumerated in opposite directions — as a sequence of index pairs (r_k, s_k), $k \in \{0,..p\}$ such that $r_k \le r_{k+1} \le r_k + 1$, $s_k \le s_{k+1} \le s_k + 1$, and $(r_k, s_k) \ne (r_{k+1}, s_{k+1})$, for all $k \in \{0,..p-1\}$. The pairing is *total* if $(r_1, s_1) = (0,0)$ and $(r_p, s_p) = (m,n)$; and is *partial* otherwise. We call two consecutive pairs $(r_k, s_k) \to (r_{k+1}, s_{k+1})$ a *step* of the pairing.

By definition, a pairing (r,s) for two segments a, b establishes a correspondence between samples a_{r_k} and b_{s_k}, for all $k \in \{0,..p\}$. See figure 2.

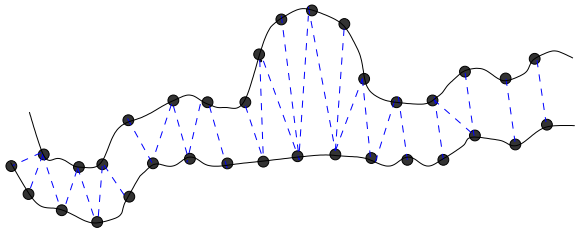

Figure 2: A discrete pairing between segments

Discrepancy with irregular pairing. If two segments a, b are related by a pairing (r, s), we re-define their mismatch as

$$S^2(a,b;r,s) = Y^2(a,b;r,s) + Z^2(r,s) \tag{4}$$

where

$$Y^2(a,b;r,s) = \frac{1}{4}\sum_{k=0}^{p-1}(\epsilon_k + \epsilon_{k+1})(\tau_{k+1} - \tau_k) \tag{5}$$

$$Z^2(r,s) = \frac{\zeta^2}{2}\sum_{k=0}^{p-1}\left|(r_{k+1} - r_k) - (s_{k+1} - s_k)\right| \tag{6}$$

where ζ^2 is a constant, $\epsilon_k = \|a_{r_k}, b_{s_k}\|^2$, $\tau_k = r_k + s_k$, and $\|a_i, b_i\|$ is some distance metric for sample values — e.g. the absolute difference between the curvatures.

The first term Y^2 is a generalization of the mismatch formula (1). The factor $(\epsilon_k + \epsilon_{k+1})/2$ estimates the mean value of $\|a_i, b_j\|^2$ along the step $(r_k, s_k) \to (r_{k+1}, s_{k+1})$ of the pairing; while the factor $(\tau_{k+1} - \tau_k)/2$ is the mean number of sampling steps (either 1 or 1/2) spanned by that pairing step. The second term Z^2 penalizes pairings that are too irregular; note that Z^2 is zero if the pairing is one-to-one. This term is necessary when each sample is a single real number; since, in this case, a sufficiently irregular pairing may yield a very low value of Y^2, even for very different segments.

In practice, of course, the true correspondence between the two segments is not known, so we use instead the pairing that minimizes S^2, among all discrete pairings of the two segments. For this reason, and because consecutive sample values are not completely independent, we are unable to derive the distribution of the mismatch $S^2_*(a,b) = S^2(a,b;r_*,s_*)$ analytically. Fortunately, however, our experiments indicate that the simplified analysis remains valid in qualitative terms. Namely, there is a critical value Ξ^2 of $S^2_*(a,b)$, at which the candidate is equally likely to be true or false, that varies approximately like $\Xi^2 = \xi^2((m+n)/2 - n_{\min})$, where ξ^2 and $n_{\min}$ are constants which depend on the nature of the fragments. Motivated by this analysis, we introduce the *discriminant*

$$\Delta(a,b;r,s) = S^2(a,b;r,s) - \xi^2\left[\frac{m+n}{2} - n_{\min}\right] \tag{7}$$

We can write this formula as

$$\Delta(a,b;r,s) = \sum_{k=0}^{p-1} d(a,b;r_k,s_k,r_{k+1},s_{k+1}) + \xi^2 n_{\min} \tag{8}$$

where

$$d(a,b;i,j,i',j') = y^2(a,b;i,j,i',j') + z^2(i,j,i',j') - x^2(i,j,i',j') \tag{9}$$

with

$$\begin{aligned} y^2(a,b;i,j,i',j') &= \frac{1}{4}(\epsilon_k + \epsilon_{k+1})(\tau_{k+1} - \tau_k) \\ &= \frac{1}{4}\Big(\|a_i, b_j\|^2 + \|a_{i'}, b_{j'}\|^2 \Big)((i'+j') - (i+j)) \qquad (10) \end{aligned}$$

$$z^2(i,j,i',j') = \frac{|i'-i-j'+j|}{2}\zeta^2 = \begin{cases} 0 & \text{if } i'-i = j'-j \\ \zeta^2/2 & \text{otherwise} \end{cases} \qquad (11)$$

$$x^2(i,j,i',j') = \frac{(i'+j')-(i+j)}{2}\xi^2 = \begin{cases} \xi^2 & \text{if } i'-i = j'-j \\ \xi^2/2 & \text{otherwise} \end{cases} \qquad (12)$$

We also define $\Delta_*(a,b) = \Delta(a,b;r_*,s_*)$ where (r_*,s_*) is the optimum pairing of a and b. Then a candidate (a,b) is probably true if $\Delta_* < 0$, and probably false if $\Delta_* > 0$.

3.4 Obtaining the parameters.

In practice, we cannot use formulas (3) to compute the values of ξ^2 and $n_{\min}$. For one thing, we do not know the values of ϕ and γ. Moreover, the outlines are usually oversampled to avoid aliasing, and therefore the samples are not independent—which affects both the average value and the number of degrees of freedom of the quadratic mismatch S^2. Moreover, the use of more flexible pairings, as described above, will have an unknown effect on the distribution of S_*^2. Therefore, the parameters ξ^2 and $n_{\min}$ have to be determined empirically, for each instance of the problem. Our method is to choose a value for ζ; compute the mismatch S_*^2 for known samples of true and false candidates, of various lengths $l = (m+n)/2$; plot the values of S_*^2 against l; and look for values of ξ and $n_{\min}$ such that the line $S_*^2 = \xi^2(l - n_{\min})$ best separates the two samples. We then repeat this analysis for several values of ζ. (In our experiments with ceramic fragments, we got the best discrimination with $\zeta \approx 2.5\xi$, at all scales.)

4 The multi-scale algorithm

The comparison of two segments is expensive because the most efficient algorithm we know to compute the optimum pairing (r,s) — a variant of *dynamic programming* [24] – requires $\Theta(n^2)$ operations for segments with n samples.

To reduce this cost, we use *multi-scale* approach [28]. Let $L_{\min}$ be the minimum length of outline that we need to compare in order ensure a reliable decision [12]. We begin by solving the problem for coarse versions of the contours, sampled with the largest possible step δ, namely $\delta^{(K)} \approx L_{\min}$. At this scale we have only a few tens of samples per outline, so we can afford to check all possible alignments of two segments by exhaustive enumeration. Of course, the limited resolution of the outlines does not

allow us to distinguish true from false candidates, and therefore we are left with a large set of possible candidates.

That done, we solve again the problem at a finer scale, with contours sampled with a smaller step $\delta^{(K-1)} = \delta^{(K)}/2$; but now we *compare only the most promising candidates found at the previous scale.* The additional details available at each new stage usually allow us to eliminate a large fraction of the false candidates. We repeat this process at finer and finer scales, with sampling steps $\delta^{(k)}$ decreasing in geometric progression.

Algorithm 1 *Multi-Scale matching of irregular fragments*

Inputs:
- *the raw outlines* $\mathcal{C} = \{\mathcal{C}_0, \mathcal{C}_1, .. \mathcal{C}_{N-1}\}$*;*
- *the minimum sampling step* $\delta^{(1)}$*;*
- *the minimum length* $L_{\min}$ *for reliable matching;*
- *the parameters* $\xi^{(k)}$*,* $\zeta^{(k)}$*, and* $n_{\min}^{(k)}$*, for each scale* k*;*
- *the corner blurring factor* α*.*

Output:
- *a set* $\mathcal{S}$ *of probably true candidates.*

1. $r \leftarrow 0$; $\mathcal{C}^{(0)} \leftarrow \mathcal{C}$.
2. While $\delta^{(k+1)} \leq L_{\min} - 8\alpha\delta^{(k+1)}$, do:
   2.1. $k \leftarrow k + 1$.
   2.2. Filter and resample the outlines $\mathcal{C}^{(k-1)}$ with step $\delta^{(k)}$, obtaining outlines $\mathcal{C}^{(k)}$.
   2.3. $\delta^{(k+1)} \leftarrow 2\delta^{(k)}$.
3. $K \leftarrow k$. Determine the initial candidate set $\mathcal{S}^{(K)}$ among the outlines $\mathcal{C}^{(K)}$.
4. For $k = K, K-1, \ldots, 1$, do:
   4.1. Refine the raw candidates $\mathcal{R}^{(k)}$ with dynamic programming, obtaining the refined candidates $\mathcal{S}^{(k)}$.
   4.2. Remove from the set $\mathcal{S}^{(k)}$ all candidates with positive discriminant Δ_*.
   4.3. For each pair of outlines, collect all their candidates in $\mathcal{S}^{(k)}$, and eliminate all but 2^k candidates with smallest (most negative) Δ_*.
   4.4. Merge any overlapping candidates among the remaining elements of $\mathcal{S}^{(k)}$.
   4.5. Map the candidates $\mathcal{S}^{(k)}$ from the outlines $\mathcal{C}^{(k)}$ to the outlines $\mathcal{C}^{(k-1)}$, obtaining the raw candidates $\mathcal{R}^{(k-1)}$.
5. Refine $\mathcal{R}^{(0)}$ obtaining the final candidate set $\mathcal{S}^{(0)}$.
6. Return $\mathcal{S} \leftarrow \mathcal{S}^{(0)}$.

Some points of the algorithm deserve further explanation:

Convergence. If the thresholds $\xi^{(k)}$ and $\zeta^{(k)}$ are properly chosen, the candidates that survive to the end of the last stage are likely to contain a large fraction of the true candidates present in the given outlines, and relatively few false ones. This assumption is supported by our experiments (see section 5).

Filtering and resampling. Steps 2.1–2.3 construct for each input fragment outline $\mathcal{C}_i^{(0)}$ a set of coarsened outlines $\mathcal{C}_i^{(k)}$, $k = 1, 2, \ldots, K$, whose sampling steps $\delta^{(k)}$ increase in geometric progression. In order to avoid aliasing artifacts, before resampling each outline we smooth it out by convolution with a Gaussian filter of width $\lambda^{(k)} = 4\delta^{(k)}$. It can be shown that the curve thus filtered can be reconstructed from its samples with negligible error. (The convolution actually changes the length of the curve, and hence the curve's parameterization, which in turns affects the filtering. Therefore, the filtering is performed by an iterative procedure which we call *geometric filtering* [5].)

Corner blurring. A side effect of filtering the outlines is a blurring of the corners (points where fracture lines meet). This effectcauses the ends of matching segments to pull away from each other, so that only the middle part of a true candidate can be recognized as such at the coarser scales. The distance $h^{(k)}$ where this blurring is significant depends on the comparison threshold ξ and the angle between the two fracture lines incident at the corner. In practice, we can assume that $h^{(k)} < \alpha\lambda^{(k)} = 4\alpha\delta^{(k)}$, where α is a small constant.

Coarsest scale. Outline comparison starts with the curves $\mathcal{R}^{(K)}$, the coarsest versions of the outlines where any true candidates with length $\geq L_{\min}$ haven't yet been obliterated by corner blurring. The stopping criterion of the filtering loop (step 2) expresses this condition, and implies that $\delta^{(K)} \leq L_{\min}/(1 + 8\alpha)$, i.e. $K = \log_2(L_{\min}/\delta^{(1)}) + \log_2(2/(1 + 8\alpha))$.

Initial candidates. In step 3 we solve the problem for the coarsest versions of the outlines, sampled with step $\delta^{(K)} \approx L_{\min}$. At this scale, each outline will have only some $O(1)$ samples, so we can afford an exhaustive comparison of all pairs of segments from all outlines, and consider only one-to-one sample pairings between the two outlines. Specifically, for each pair of fragment outlines a, b, we list all segments $(a_i, a_{i+1}, \ldots, a_{i+n})$ and $(b_j, b_{j+1}, \ldots, b_{j+n})$ which have $n \geq L_{\min}/\delta^{(K)} - 8\alpha$ and $\Delta < 0$ (with one-to-one sample pairing); and, of each list, we keep only the 2^K candidates with smallest (most negative) Δ. The set $\mathcal{S}^{(K)}$ of *initial candidates* is the union of these trimmed lists.

Candidate refinement. In steps 4.1 and 5, we *refine* each candidate: meaning that we adjust the endpoints of its segments, and recompute the optimal pairing (r_*, s_*) so as to minimize its discriminant Δ, for the higher-resolution outlines $\mathcal{C}^{(k)}$. The refinement procedure is described in section 4.1.

Selection, pruning, and merging. In steps 4.3 and 4.4, we again discard those refined candidates that have $\Delta_* \geq 0$, and keep only the 2^k best candidates for each pair of fragments; then we look for overlapping candidates and merge them. Note that a true candidate (a, b) may give rise to two or more candidates in the initial set $\mathcal{S}^{(K)}$, covering different parts of a and b with slightly different alignments. These partial candidates are likely to overlap once they get refined and mapped to finer scales.

Candidate mapping. In step 4.5 we map each candidate in the set $\mathcal{S}^{(k+1)}$, — a pair of segments on the outlines $\mathcal{C}^{(k+1)}$ — to the corresponding candidate on the outlines $\mathcal{C}^{(k)}$. To account for corner blurring, we also extend each segment in both directions by $\alpha(\lambda^{(k+1)} - \lambda^{(k)})$.

Final scale. We repeat the refine-prune-merge cycle at finer and finer scales, with sampling steps $\delta^{(k)}$ decreasing in geometric progression, until we reach a specified maximum resolution scale $\lambda^{(1)} = 4\delta^{(1)}$, comparable to the magnitude of the perturbations inherent in the observed outlines.

4.1 Refining a candidate

The most expensive step in the algorithm above is the refinement of each candidate (a, b), which entails finding a pairing (r_*, s_*) that minimizes $S^2(a, b; r, s)$. We find this pairing using a modified version of the *dynamic programming* (DP) algorithm commonly used in sequence matching [24].

Dynamic programming. The problem can be formulated as finding the minimum cost path in a certain directed acyclic graph G, whose vertices are all pairs of indices $(i, j) \in \{0..m\} \times \{0..n\}$ of samples a_i and b_j. Each edge of G corresponds to two index pairs that could occur in consecutive positions in a valid pairing, namely

$$\begin{array}{l} (i, j) \to (i + 1, j), \\ (i, j) \to (i, j + 1), \\ (i, j) \to (i + 1, j + 1) \end{array} \tag{13}$$

for $0 \leq i \leq m - 1$ and $0 \leq j \leq n - 1$. See figure 3(a). Therefore, any discrete pairing (r, s) of a and b, as defined in section 3.3, corresponds to a path in G — and vice

versa. (Total pairings, in particular, correspond to a paths from vertex (r_0, s_0) to vertex (r_p, s_p).)

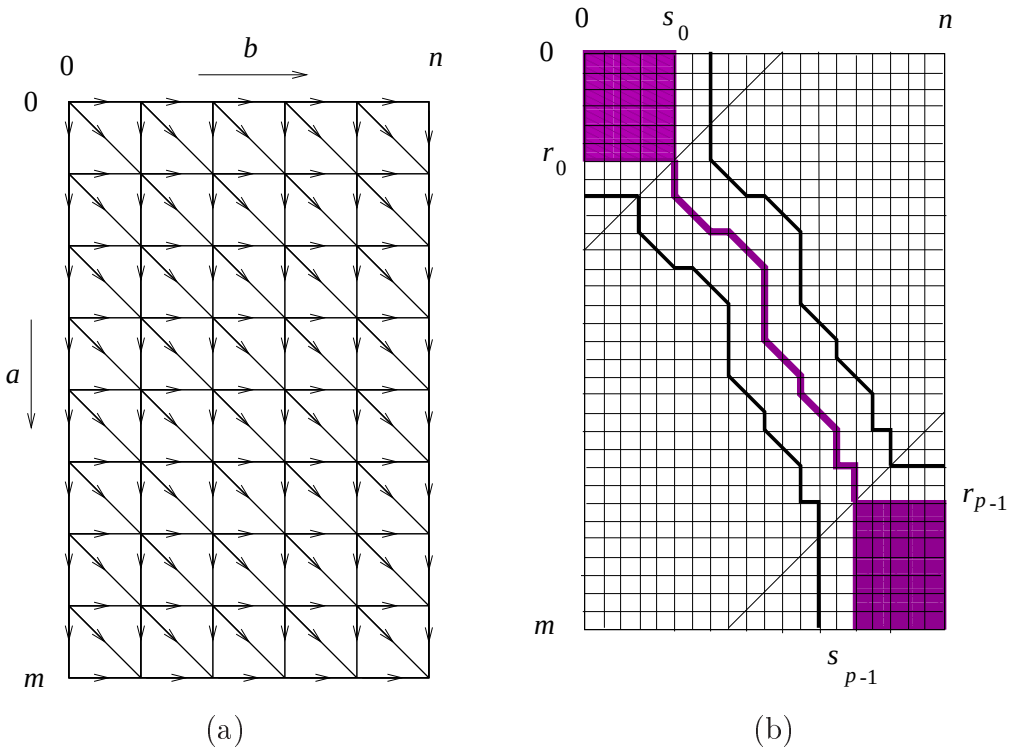

Figure 3: (a) The dynamic programming graph G. (b) The incremental matching principle, showing the approximate pairing (thick line), the set V (thick lines and dark shading), and the valid pairs $\hat{V}$ (medium lines and medium shading).

To each edge of this graph, with endpoints $(i, j) \to (i', j')$, we associate the cost $d(a, b; i, j, i', j')$ according to formula (9). It follows that the total edge cost of the path is

$$\sum_{k=0}^{p-1} d(a, b; r_k, s_k, r_{k+1}, s_{k+1}) \;=\; \Delta(a, b; r, s) - n_{\min}\xi^2 \tag{14}$$

See formula (8). Note that $n_{\min}\xi^2$ is a constant, so minimizing the cost of a path in G is equivalent to minimizing the discriminant $\Delta(a, b; r, s)$.

Since the endpoints of the segment (a, b) are not precisely known a priori, we do not require the optimum pairing (r_*, s_*) to begin at $(0, 0)$ and end at (m, n). Thus the refined candidate actually consists of sub-sements a' and b' of a and b, where $a'_0 = a_{r_{*0}}$, $a'_{m'} = a_{r_{*p}}$, and similarly for b'. Note that true candidates are expected to have $\Delta < 0$, and therefore $d < 0$ on the average. It follows that, for true candidates, the value of $\Delta(a, b; r, s)$ is usually minimized when the paired length is maximum;

while the reverse is usually true for for false candidates.

Incremental dynamic programming. To reduce the cost of the dynamic programming algorithm, we assume that the desired pairing for a given candidate will not deviate excessively from the best pairing found for the same candidate at the previous (coarser) scale. Thus we search for the optimum path in the graph G within a narrow band surrounding the previous optimum path, mapped to the new scale.

More precisely, let $(r', s') = (r'_0, s'_0)..(r'_{p-1}, s'_{p-1})$ be an initial, approximate pairing for the candidate (a, b) (which need not cover the whole of the segments a and b). We define a set V of *initial pairs* as consisting of all pairs (r'_k, s'_k) from this pairing, plus all pairs (i, j) with $0 \le i \le r_0$ and $0 \le j \le s_0$, plus all pairs (i, j) with $r_{p-1} \le i \le m$ and $s_{p-1} \le j \le n$. In other words,

$$\begin{aligned} V \;=\; & \{(r'_k, s'_k) \,:\, 0 \le k \le p-1\} \\ & \cup \{(i,j) \,:\, 0 \le i \le r'_0 \wedge 0 \le j \le s'_0\} \\ & \cup \{(i,j) \,:\, r'_{p-1} \le i \le m \wedge s'_{p-1} \le j \le n\} \end{aligned} \tag{15}$$

We then define the set of *valid pairs* $\hat{V}$ as all index pairs within some tolerance q of the set V, namely

$$\hat{V} = \{(i+\alpha, j+\beta) \,:\, (i,j) \in V \wedge\ \alpha, \beta \in [-q..+q]\} \tag{16}$$

If the tolerance q is sufficiently large, and (r', s') is close enough to the optimum pairing, then the optimum pairing (r_*, s_*) between a and b will be contained in the set $\hat{V}$. The refined pairing that we seek corresponds to a path of minimum cost that uses only the vertices and edges in the set $\hat{V}$. It is easy to see that the boundary of the set $\hat{V}$ consists basically of two copies of the set (r', s'), displaced diagonally by $+q$ and $-q$. See figure 3(b). Therefore, the region $\hat{V}$, and its induced subgraph of G, can be represented in a very compact and efficient way.

4.2 Performance analysis

The standard dynamic programming algorithm requires $\Theta(n^2)$ operations for segments with n samples. With the incremental matching modification, the cost gets reduced to $\Theta(qn)$. Therefore, the total cost of each iteration of the main loop (steps 4.5–4.4) is proportional to the total length of all candidates in $\mathcal{R}^{(k)}$. The length of each candidate is bounded by $c'2^{K-k}$, for some conctant c'; and the number of candidates in $\mathcal{R}^{(k)}$ is bounded in step 4.3 by $2^k N(N-1)/2$. Therefore, the cost of each iteration is $O(2^K N^2)$. Since the number of iterations K is $\log_2 L_{\min} + O(1)$, the algorithm runs in $O(N^2 L_{\min} \log L_{\min})$ time — i.e. faster than the single-scale algorithm by a factor of $\Theta(L^2/(L_{\min} \log L_{\min}))$.

Actually, the bound of $2^k N(N-1)/2$ for $\left|\mathcal{R}^{(k)}\right|$ is rather pessimistic. By Euler's theorem, the number of maximal true candidates, which is the number of edges in the graph-theoretic dual of the fracture network, is only $O(N)$. Therefore, if the parameters $\xi^{(k)}$, $\zeta^{(k)}$, and $n_{\min}^{(k)}$ are appropriately chosen, the number of candidates $\left|\mathcal{R}^{(k)}\right|$ should decrease much faster than $2^k N(N-1)/2$. In that case, the total cost will be dominated by that of the first few iterations, namely $O(N^2 2^K) = O(N^2 L_{\min})$.

Compared to the cost $\Theta(N^2 L^2)$ of single-scale comparison, we obtained a speedup factor of at least $\Theta(L^2/(L_{\min} \log L_{\min}))$. Note that in typical instances of the problem we may have $L \approx 10^3$ and $L_{\min} \approx L/10$, implying $L^2/(L_{\min} \log L_{\min}) \approx 10^3$. This speedup is therefore quite significant, even allowing for any large constants which may be hidden in the Θ notation.

5 Experiments

We coded this algorithm (in Modula3 [17], for the Unix platform) and tested it with an artificial but fairly realistic sample of ceramic fragments. The original objects were five rectangular unglazed ceramic tiles, about 25.0 cm by 6.0 cm, which were shattered into 112 major pieces. See figure 4.

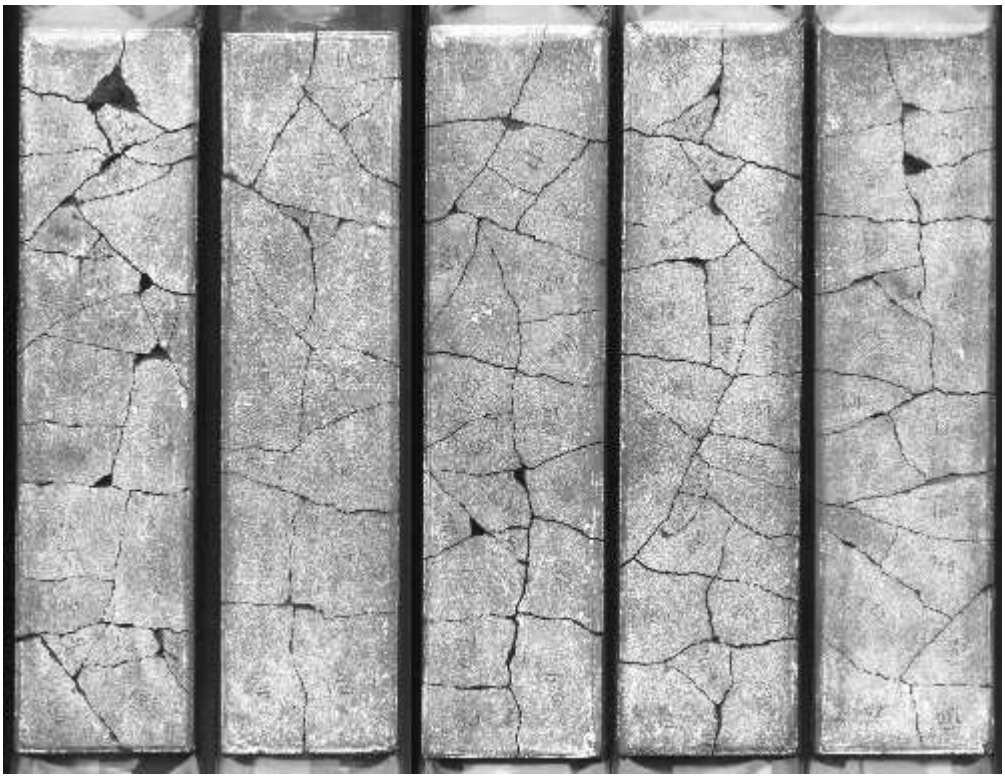

Figure 4: Test fragments, manually reassembled.

Digitization of the fragments. The fragments were directly digitized with a standard flatbed scanner (UMAX model UC630 Maxcolor) at 300 pixels per inch, after lightly rubbing the flat side of each fragment with chalk to enhance contrast. The resulting images are shown in figure 5.

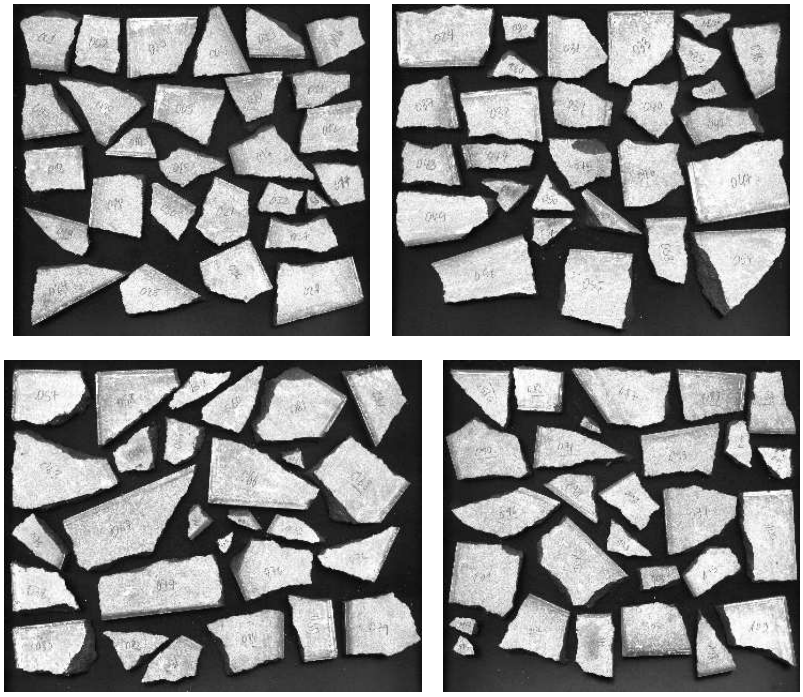

Figure 5: Input images for the test.

The outlines extracted from figure 5 (whose average length L was about 1500 pixels, or 127 mm) were filtered and converted to curvature values.

Parameters and results. The multi-scale matching algorithm was then applied to these curvature-encoded outlines, beginning with scale $\lambda^{(5)} = 32$ pixels (2.7 mm) and ending with scale $\lambda^{(1)} = 2$ pixels (0.17 mm). The corner blurring factor was set to $\alpha = 6$. We report here two runs of the program, denoted below as A e B, with slightly different parameters. For each test run, and each filtering scale $\lambda^{(k)}$, we show:

- the sample mismatch threshold $\xi^{(k)}$ used in refinement,
- the half-step pairing penalty $\zeta^{(k)}$,
- the maximum number $m^{(k)}$ of candidates retained per fragment pair,
- the cost $t_{ref}^{(k)}$ of the refinement (in seconds),
- the number of candidates $\left|\mathcal{S}^{(k)}\right|$ pruning and merging, and
- the number $\left|\mathcal{S}^{(k)} \cap \mathcal{T}\right|$ of true candidates in that set.

Solution quality measures. Furthermore, in order to estimate the quality of each candidate set $\mathcal{S}$, we give two metrics borrowed from information retrieval theory: the *relevance* $\rho(\mathcal{S})$ — which measures the program's ability to reject false candidates; and the *sensitivity* $\nu(\mathcal{S})$ — which measures its ability to recognize the true candidates:

$$\rho(\mathcal{S}) = \frac{|\mathcal{S} \cap \mathcal{T}|}{|\mathcal{S}|} \qquad \nu(\mathcal{S}) = \frac{|\mathcal{S} \cap \mathcal{R}|}{|\mathcal{R}|} \tag{17}$$

Here $\mathcal{R}$ is the set of *recognizable candidates* — the subset of $\mathcal{T}$ that could be reliably indentified by a careful and patient human who compared only the digitalized contours, without any time constraints. (We use $\mathcal{R}$ rather than $\mathcal{T}$ in the formula for $\nu(\mathcal{S})$ because some true candidates may be impossible to identify, by any method and at any cost, due to of excessive material loss or acquisition errors. Counting such candidates as program failures would only reduce the usefulness of the sensitivity index $\nu(\mathcal{S})$.)

In the ideal case — the program finds all the true candidates that could possibly be found, and only those — we would have $\rho(\mathcal{S}) = \nu(\mathcal{S}) = 1$. By and large, there is a tradeoff between sensitivity and relevance. If we relax the candidate selection criteria (for instance, if we increase the threshold parameters $\xi^{(k)}$, or the maximum number $m^{(k)}$ of candidates accepted for each curve pair), we will generally obtain solutions with greater sensitivity, but decreased relevance.

Test A. In this run, the minimum segment length $L_{\min}$ of candidates to look for was set to 250 pixels (20.8 mm). Note that at the coarsest scale ($\lambda^{(5)} = 32$ pixels, $\delta^{(5)} = 8$ pixels) these candidates were reduced to $250 - 6 \cdot 32 = 58$ pixels (≈ 8 samples).

The number of initial candidates found by the program (before refinement and pruning) was 87773. Table 1 shows the parameters and summarizes the results of the multi-scale matching algorithm, for each stage $\lambda^{(k)}$.

$\lambda^{(k)}$	$\xi^{(k)}$	$\zeta^{(k)}$	$m^{(k)}$	$t_{ref}^{(k)}$	$\lvert\mathcal{S}^{(k)}\rvert$	$\lvert\mathcal{S}^{(k)} \cap T\rvert$	$\rho^{(k)}$	$\nu^{(k)}$
32	0.0007	0.0021	16	14200	12950	45	0.0035	0.8491
16	0.0022	0.0044	8	7484	342	35	0.1023	0.6604
8	0.0080	0.0140	4	668	67	27	0.4030	0.5094
4	0.0240	0.0400	2	321	28	23	0.8214	0.4340
2	0.0650	0.1000	1	419	22	19	0.8636	0.3585

Table 1: Parameters and results for run A of the multi-scale algorithm.

Figure 6 below shows the 22 surviving candidates at scale $\lambda^{(1)} = 2$ pixels.

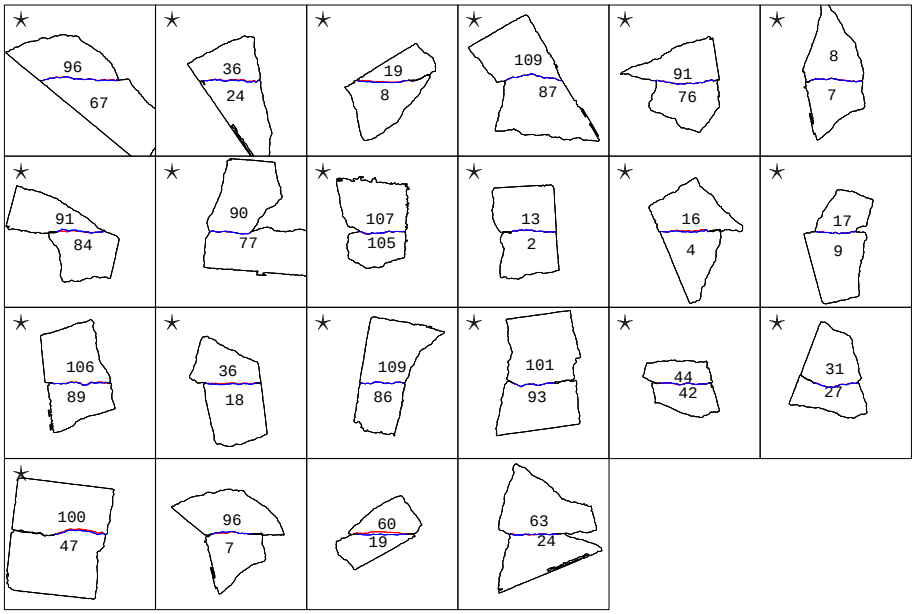

Figure 6: The 22 candidates found by the multi-scale matching algorithm for test A at the scale $\lambda^{(1)} = 2$ pixels. The stars ($\star$) identify true candidates.

Test B. Table 2 below shows the parameters and results of the second run, in which the comparison criteria were relaxed by lowering the threshholds $\xi^{(k)}$. For this run, the minimum length $L_{\min}$ of candidates to look for was set at 210 pixels (17.8 mm). Note that at the coarsest scale ($\lambda^{(5)} = 32$ pixels, $\delta^{(5)} = 8$ pixels) these candidates were reduced to $210 - 6 \cdot 32 = 18$ pixels (≈ 3 samples). The number of initial candidates was 166626.

$\lambda^{(k)}$	$\xi^{(k)}$	$\zeta^{(k)}$	$m^{(k)}$	$t_{ref}^{(k)}$	$\lvert\mathcal{S}^{(k)}\rvert$	$\lvert\mathcal{S}^{(k)} \cap T\rvert$	ρ	ν
32	0.0008	0.0021	16	31091	64743	57	0.0009	0.7703
16	0.0026	0.0044	8	52551	7797	70	0.0090	0.9459
8	0.0090	0.0140	4	10095	1814	64	0.0353	0.8649
4	0.0260	0.0400	2	4058	442	50	0.1131	0.6756
2	0.0705	0.1000	1	2542	277	46	0.1661	0.6217

Table 2: Parameters and results for run B of the multi-scale algorithm.

Note that the sensitivity ν of the final set increased (from 0.3585 to 0.6217) in comparison with run A, while the relevance ρ decreased (from 0.8636 to 0.1661). Figure 7 shows the first 60 candidates surviving after the last stage, and figure 8

shows the corresponding adjacency graph.

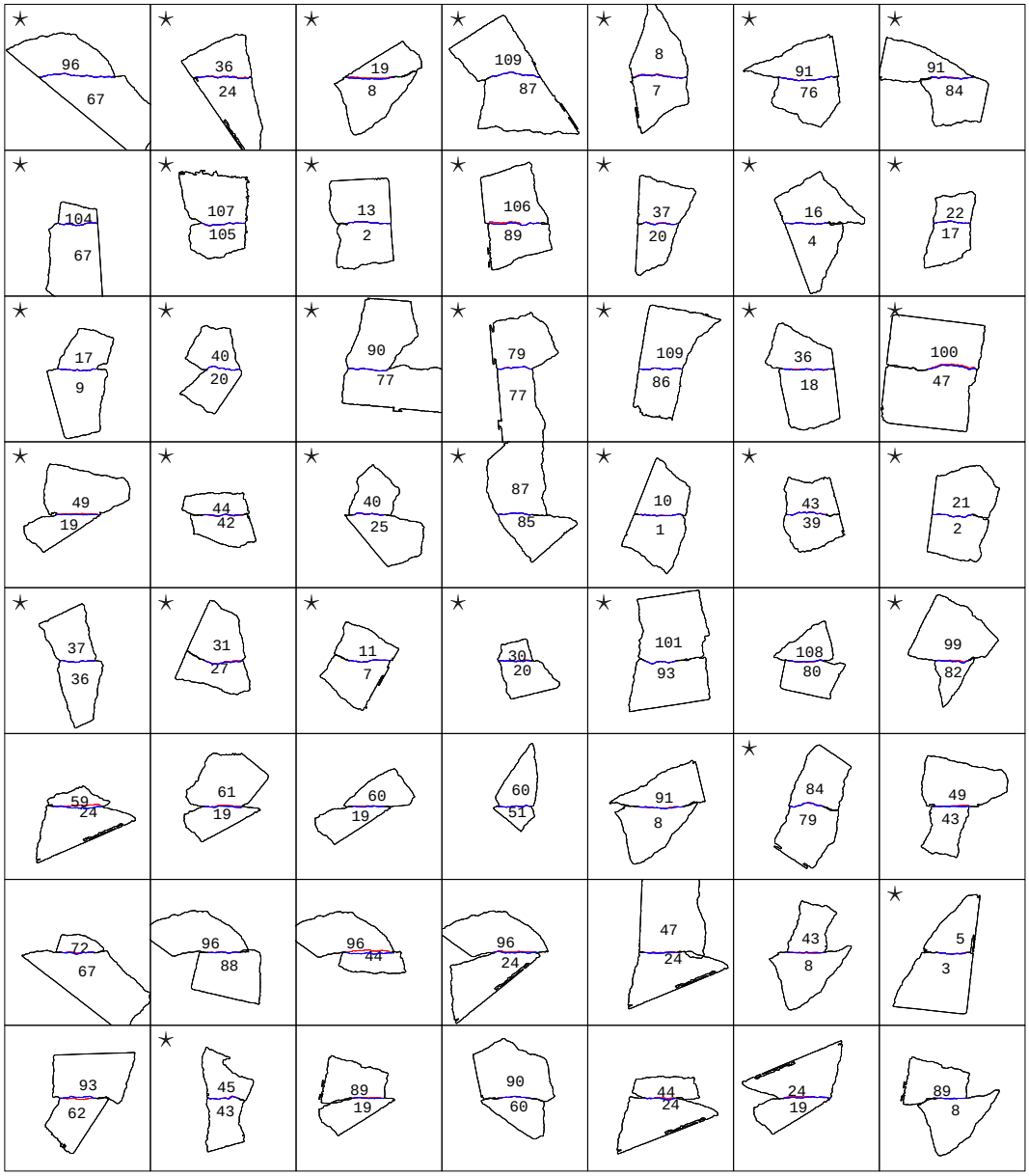

Figure 7: The first 56 candidates returned by the multi-scale matching algorithm for run B, at the final scale $\lambda^{(1)} = 2$ pixels. The stars ($\star$) identify true candidates.

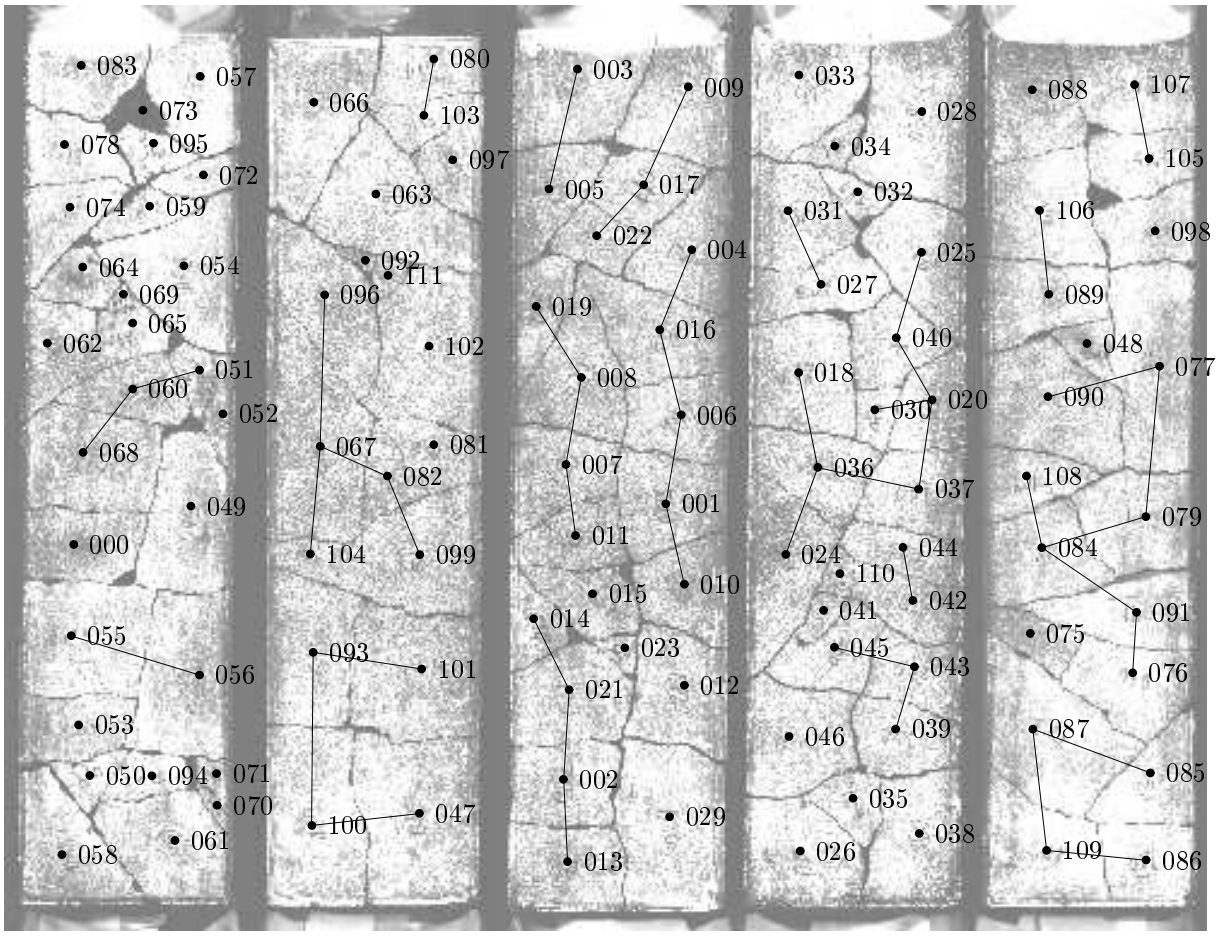

Figure 8: Test fragments, manually reassembled, showing the 46 true candidates among the 277 returned by the program in run B, at the scale $\lambda^{(1)} = 2$ pixels.

5.1 Conclusions

Our experimental results, although modest in scope, demonstrate the possibility of automatically identifying adjacent fragments by matching the shapes of their outlines. Those results also validate the basic premise of the multi-scale matching method, namely that the false candidates are quickly eliminated as they are re-tested with increasing resolution.

Another question that requires further testing is whether the multi-scale method will work for other kinds of fragments, such as glass or glazed ceramics. Our methods depend on the randomness of the fracture lines, and therefore work best for granulated material like unglazed ceramics, stone, stucco, etc.. Measurements show that

fractures in such materials are fractal-like curves with dimension in the range 1.1 to 1.2. (In fact, it is precisely their randomness at all scales that makes our multi-scale approach feasible.) Fractures in glass or glazed ceramic objects tend to be smoother, i.e. their fractal dimension is much closer to 1; and therefore their shapes are less characteristic. On the other hand, the fracture edges are much sharper, and therefore can be compared with greater precision. Therefore we expect our method to work for these materials, too.

In spite of the large estimated speedup (10^2 to 10^3) provided by the multi-scale method, the algorithm is still somewhat too expensive for practical use — witness the times reported in section 5. We expect another factor of 10 will come from better coding and the use of more compact data structures. Further speedup will require improving the algorithm itself. In particular, by using *geometric hashing* techniques, as proposed by Kalvin and others [7], it would be possible to reduce the $O(N^2)$ term to something closer to $O(N \log N)$.

References

[1] J. Babaud, A. Witkin, M. Baudin, and R. Duda. Uniquess of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):26–33, 1986.

[2] Roger Bagnall. Advanced papyrological information system. WWW document at `//scriptorium.lib.duke.edu/`, file `papyrus/texts/APISgrant.html`, 1994.

[3] F. Boussofiane and G. Bertrand. A new method for recognizing and locating objects by searching longest paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):445–448, 1993.

[4] Grigore C. Burdea and Haim J. Wolfson. Solving jigsaw puzzles by a robot. *IEEE Transactions on Robotics and Automation*, 5(6):752–764, 1989.

[5] Helena Cristina da Gama Leitão. *Reconstrução Automática de Objetos Fragmentados.* PhD thesis, Institute of Computing, University of Campinas, November 1999. (In Portuguese).

[6] R. Halíř and J. Flusser. Estimation of profiles of sherds of archaeological pottery. In *Proceedings of the Czech Pattern Recognition Workshop (CPRW'97)*, pages 126–130, February 1997.

[7] Alan Kalvin, Edith Schonberg, Jacob T. Schwartz, and Micha Sharir. Two-dimensional, model-based, boundary matching using footprints. *The International Journal of Robotics Research*, 5(4):38–55, 1986.

[8] Alan D. Kalvin et al. Using visualization in the archaeological excavations of a pre-Inca temple in Peru. Research Report RC 20518, IBM T. J. Watson Research Center, 1996.

[9] Raymond Legault and Ching Y. Suen. Optimal local weighted averaging methods in contour smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):801–817, 1997.

[10] Helena C. G. Leitão and Jorge Stolfi. Reconstrução de objetos fragmentados. In *Anais do IX Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI'96)*, pages 365–366, Caxambú, MG (Brazil), October 1996.

[11] Helena C. G. Leitão and Jorge Stolfi. Automatic reassembly of irregular fragments. Technical Report IC-98-06, Institute of Computing, Univ. of Campinas, April 1998.

[12] Helena C. G. Leitão and Jorge Stolfi. Information contents of fracture lines. In *Proc. WSCG'2000 - the 8th International Conference in Central Europe on Computer Graphics, Visualization, and Interactive Digital Media*, volume 2, pages 389–395. Univ. of West Bohemia Press, February 2000.

[13] Mark Levoy. Scanning the fragments of the Forma Urbis Romae. WWW document available at `//www.graphics.stanford.edu/`, file `projects/mich/forma-urbis/forma-urbis.html`, May 1999.

[14] C. Menard and R. Sablatnig. On finding archaeological fragment assemblies using a bottom-up design. In *Proceedings of the 21st Workshop of the Austrian Association for Pattern Recognition*, pages 203–207, 1997.

[15] Farzin Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):539–544, 1995.

[16] Farzin Mokhtarian and Alan K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.

[17] Greg Nelson. *Systems Programming with Modula-3*. Prentice-Hall, 1991.

[18] Carl E. Pearson. *Handbook of Applied Mathematics: Selected Results and Methods*. Van Nostrand Reinhold, 2 edition, 1983.

[19] Arthur R. Pope. Model based object recognition: A survey of recent research. Technical Report TR-94-04, Berkeley University, 1994.

[20] G. M. Radack and N. I. Badler. Jigsaw puzzle matching using a boundary centered polar encoding. *Computer Graphics and Image Processing*, 19:1–17, 1982.

[21] Anothai Rattarangsi and Roland T. Chin. Scale-based detection of corners of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(4):430–449, 1992.

[22] Paul Rosin and Svetha Venkatesh. Extracting natural scales using the Fourier description. *Pattern Recogniton*, 26(9):1383–1393, 1993.

[23] R. Sablatnig and C. Menard. Classification and 3d-reconstruction of rotational symmetric pottery. In *Computer Vision - Proc. of the Computer Vision Winter Workshop (CVWW'98)*, pages 186–194, 1998.

[24] João Setubal and João Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing, 1997.

[25] Göktürk Üçoluk and I. Hakki Toroslu. Automatic reconstruction of broken 3-D surface objects. *Computers & Graphics*, 23(4):573–582, August 1999.

[26] R. Vergnieux. Le fac-similé èlectronique ou Les restitutions virtuelles. In X^e *CNRS Table Ronde de Informatique et Égyptologie*, Bordeaux, 1994. WWW document at `//silicon.Montaigne.u-bordeaux.fr/`, file `8001/HTML/ONLINE/IE10/Robert.html`.

[27] Roger W. Webster, Paul S. LaFollete, and Robert L. Stafford. Isthmus critical points for solving jigsaw puzzles in computer vision. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1271–1278, 1991.

[28] Andrew P. Witkin. Scale-space filtering. In *Proceedings of IJCAI'83 - 8th International Joint Conference on Artificial Intelligence*, pages 1019–1021, 1983.

[29] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamda. Solving jigsaw puzzles by computer vision. In *Annals of Operations Research*, volume 1986, pages 51–64, 1988.

[30] Haim J. Wolfson. On curve matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(5):483–488, 1990.