# Robust Virtual Unrolling of Historical Parchment XMT Images

Chang Liu, Paul L. Rosin, Yu-Kun Lai, and Weiduo Hu

***Abstract*— We develop a framework to virtually unroll fragile historical parchment scrolls, which cannot be physically unfolded via a sequence of X-ray tomographic slices, thus providing easy access to those parchments whose contents have remained hidden for centuries. The first step is to produce a topologically correct segmentation, which is challenging as the parchment layers vary significantly in thickness, contain substantial interior textures and can often stick together in places. For this purpose, our method starts with linking the broken layers in a slice using the topological structure propagated from its previous processed slice. To ensure topological correctness, we identify fused regions by detecting junction sections, and then match them using global optimization efficiently solved by the blossom algorithm, taking into account the shape energy of curves separating fused layers. The fused layers are then separated using as-parallel-as-possible curves connecting junction section pairs. To flatten the segmented parchment, pixels in different frames need to be put into alignment. This is achieved via a dynamic programming-based global optimization, which minimizes the total matching distances and penalizes stretches. Eventually, the text of the parchment is revealed by ink projection. We demonstrate the effectiveness of our approach using challenging real-world data sets, including the water damaged fifteenth century Bressingham scroll.**

***Index Terms*— X-ray, parchment, as parallel as possible, flatten, blossom algorithm, dynamic programming, ink projection.**

## I. Introduction

PARCHMENT has been an important writing medium for recording valuable information throughout history because it is thin and stiff, but yet sufficiently flexible to roll. However, over hundreds of years, parchment can convert to gelatin, so that the layers of the scrolled parchments become brittle and get stuck together. Figure 1 demonstrates a typical parchment scroll, the Bressingham scroll, which is an account from the manor of Bressingham, dated 1408-9 (NRO, PHI 468/5) [1]. The records on the scroll include: the income of the lord from the manor and his expenditure, profits from holding the manor court, sales of underwood, and leasing out the fishing rights. The width of the scroll measures around 270 mm. The total length of the scroll is unknown, as it is impossible to unroll completely: at approximately 100 mm from the start of the parchment the scroll has become fused together. The fusing of the scroll is most likely to have been caused by exposure to moisture and damp storage. Any attempt to physically unfold the parchment document will lead to delamination of the surface of the parchment causing an unacceptable level of damage. Consequently, image processing is being applied to explore new means of accessing such delicate parchments without physically unrolling them.

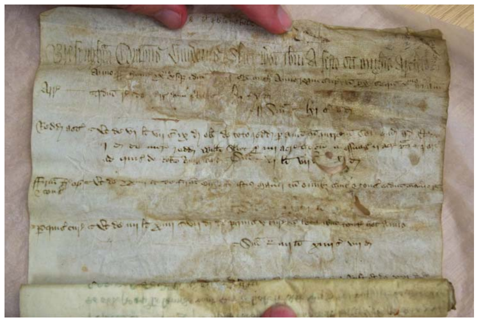


Fig. 1. The Bressingham scroll (1408-9), which cannot be physically fully unrolled.

Digital document restoration technology develops new methods for reconstructing such documents and recovering information which cannot be accessed physically. This technology has been an extremely active area of research in recent years. So far, much attention has been paid to the analysis of regular photographic images of historical documents and 3D reconstruction and virtual flattening of deformed but non-scrolled parchment documents. 3D reconstructions of documents can be grouped into three different classes [2]: single-image methods, which reconstruct the shape of a document based on their geometric and shading information [3], stereo-image methods, which restore the 3D surfaces of documents by stereo image pairs [4], and structured-light scanning methods, which calculate the 3D shape of a document with a structured-light scanner [5]. Based on the restored 3D surface, many surface parameterization methods [6] have been proposed to find a mapping from document surfaces to planar domains with minimum distortion, which will enable the recovery of the original flat shape of distorted documents. A new level of accessibility has been provided for many

Manuscript received January 12, 2017; revised November 10, 2017 and December 11, 2017; accepted December 11, 2017. Date of publication December 18, 2017; date of current version January 23, 2018. This work was supported in part by the EPSRC Project EP/G010110/1 and in part by the China Scholarship Council under Grant 201406020068. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jan Sijbers. *(Corresponding author: Paul L. Rosin.)*

C. Liu and W. Hu are with the School of Astronautics, Beihang University, Beijing 100191, China (e-mail: laodananhang2006@aliyun.com; weiduo.hu@buaa.edu.cn).

P. L. Rosin and Y.-K. Lai are with the School of Computer Science and Informatics, Cardiff University, Cardiff CF24 3AA, U.K. (e-mail: paul.rosin@cs.cf.ac.uk; yukun.lai@cs.cf.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2783626

valuable literary works. However, such methods cannot cope with parchments which cannot be physically unrolled. Virtual unrolling is required for such documents, though only very few results have been reported.

Recovering the information written on a fragile parchment scroll is a difficult task since the characters are wrapped inside the parchment, and thus cannot be photographed by normal cameras. Therefore non-interventive methods have been developed based on X-ray microtomography (XMT) scanning [7], [8] and virtual unrolling [1], [9], [10]. For an XMT-based technique, the separation of fused parchment layers is the major problem for parchment scroll restoration [11], [12]. Some widely used image segmentation methods, such as snakes [13] and max-flow-based graph cut [14], are not effective for this problem because while they can effectively differentiate between the foreground and the background, they cannot recognise fused regions and separate them into individual layers. For a different purpose, Samet et al. presented a method to reconstruct the missing parts of contour lines by automatically detecting the terminal points and then matching and linking the related terminal points by line segments [15]. The matching was based on Euclidean distance and the directional information of the terminal points. Therefore the prerequisite for this method is that the two related terminal points are close enough, which is not generally satisfied by the parchment scroll. The related endpoints of a missing boundary in the fused region may not be close, nor can the missing boundaries be reconstructed by line segments. Allegra *et al.* [16] proposed an approach to semi-automatic virtual unrolling of the papyrus scroll with X-ray tomography. Under the assumption of few differences between adjacent slices, the authors used the skeleton extracted from a single slice to virtually unfold the whole papyrus scroll. However, despite the considerable similarity between adjacent slices, the differences accumulate dramatically over just a small number of slices. This results in relatively large errors in the virtual unrolling. Seales *et al.* [17] presented an algorithm to virtually unwrap part of the En-Gedi parchment scroll. They first propagated several chains through the volume by the local symmetric tensor and a set of inner spring forces, thus tracing out several surfaces of the scroll over time. Then the obtained surfaces were textured, flattened, and manually merged to produce a large reconstructed image for the scroll. The weakness of this method is that the chain used for propagation needs to be frequently manually corrected to prevent it from crossing over itself and the surface boundaries if there exist many large fused regions in the slices. Baum *et al.* [18] attempted to reveal hidden text in rolled papyrus using an interpolation technique. The authors manually initialised several skeletons across the whole volume, and the remaining skeletons were produced by interpolating between their two adjacent initialised skeletons. The scroll was eventually reconstructed by applying texture to the flattened mesh comprised of all the skeletons. However, since the scroll is noticeably distorted along its long axis, substantial slices needed to be initialised to achieve the good final unrolling result.

The most relevant work was reported by Samko *et al.* [11] and [12], who automatically processed several parchment scrolls which have similar local characteristics to the Bressingham scroll. After preprocessing each slice by Coherence-Enhancing Diffusion (CED) filtering [19], a shape prior using the parchment layer thickness was incorporated into graph cut (GC) [14] to robustly segment the parchment layers from the background. Subsequently, the missing boundaries were recreated in the fused region from the boundary of the opposite side of the same layer or the closest preserved boundary. The shape-prior-based graph cut will thin the parchment layers, producing much fewer inter-layer connections between two layers. Nonetheless, the major problems of this graph cut are that it is difficult to choose the weight parameters for the data, smoothness and shape terms, and for those parchments with thin layers containing internal structure, the graph cut is likely to fragment the layers into many small parts (Fig. 2). Furthermore, only a simple strategy was used to match the endpoints of missing boundaries, which may lead to false reconstruction of missing boundaries if the fused region consists of more than three layers. Thus this method cannot cope with parchments like the Bressingham scroll.

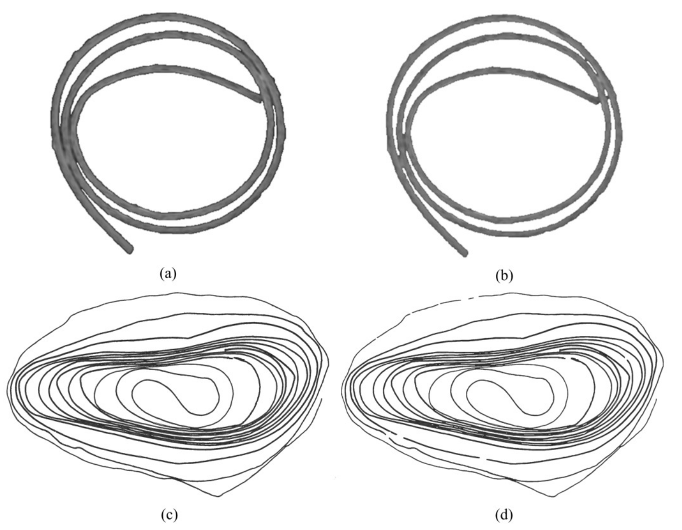


Fig. 2. Image segmentation results. (a) Otsu thresholding algorithm [20] on the small scroll. (b) Shape-prior-based graph cut [12] on the small scroll. (c) Otsu thresholding algorithm [20] on the Bressingham scroll. (d) Shape-prior-based graph cut [12] on the Bressingham scroll.

In this paper, we present a new method to separate parchment layers fused together and to virtually unroll the parchments, revealing the text. We first segment the foreground parchment regions automatically, which is achieved by applying our segmentation algorithm to each individual image frame. Unlike traditional approaches, we explicitly enforce topological constraints, which means a parchment sheet in each image frame forms a continuous (rolled) strip. The skeleton is also extracted from the parchment in each image frame. To flatten the segmented parchment, skeleton pixels in different frames need to be put into alignment to form an interior surface. This is achieved via an efficient dynamic programming based global optimisation which minimises the total matching distances and penalises stretches. Eventually, the text of the parchment is revealed by ink projection. We perform both qualitative and quantitative analysis and demonstrate the effectiveness of our approach using challenging datasets, including the 15th century Bressingham scroll which is difficult to process using previous methods. This is an extended version of our conference paper [21], and we have substantially improved the pipeline with propagation based layer connection, global optimisation for junction section matching for fused region separation, dynamic programming based flattening, and improved ink projection, as well as additional evaluation with more experiments, both qualitative and quantitative.

## II. Parchment Segmentation

Our processing pipeline starts with parchment images obtained from X-ray microtomography, and these are first segmented with the standard Otsu thresholding algorithm [20] since it is simple and efficient. More sophisticated general purpose segmentation algorithms were also considered for this stage, but they are also prone to topological errors. Moreover, we considered following Samko *et al.* [12], and tried applying Coherence-Enhancing Diffusion (CED) filtering [19] to remove noise and fine texture. However, we found this to be unnecessary for our processing pipeline. In most cases, Otsu initialisation produces the best or close to best values for the measures (see Sec. IV for detailed discussion) so we use this by default in our pipeline. Following the initial segmentation the foreground is processed by three main steps: layer connection, refinement of segmentation, and skeleton connection. The first step connects the broken layers of the parchment by a strip with the width of the parchment layer. At the next step, we find the fused regions by detecting junction sections, and match them using global shape energy optimisation effectively solved by the blossom algorithm. We then separate the fused regions into parchment layers by linking those matched junction sections using as-parallel-as-possible connecting curves. Sometimes, the parchment layer is so thin that it is likely to be cut off by our segmentation method. Thus, after separating fused regions, we extract the skeletons of the layers and link the skeletons which should belong to one layer. The extracted skeletons are also useful for flattening and ink projection.

### A. Layer Connection

It is common that some areas of the historic parchments have become scuffed and delaminated, so that in the X-ray slice the parts of the layers corresponding to those areas are missing. A typical example is shown in Fig. 3. In this section, we will link the broken parchment layers by a strip with the same width as the parchment layer. Because in the slice sequence the two adjacent slices are very similar, the broken layers in the current image can be correctly connected according to the topological structure of the previous processed slice [22]. Specifically, we first extract the skeleton of the parchment layer from the previous processed slice using morphological operations, and then obtain the skeleton parts which are included in the background of the current slice. As demonstrated in Fig. 4, the red segment represents a skeleton part included in the background, which touches the

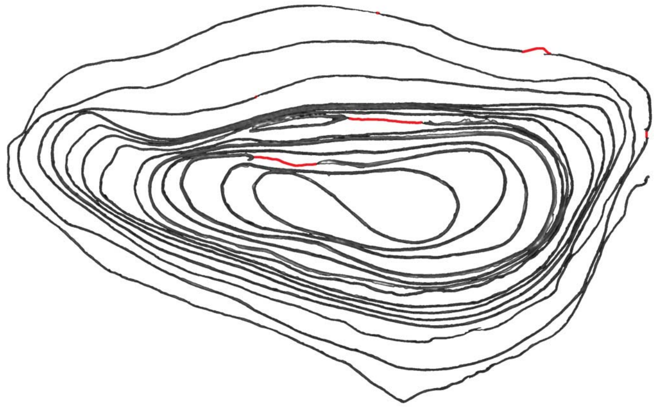


Fig. 3. The broken layers of the parchment are connected by the skeleton segments (red segments) of the previous slice.

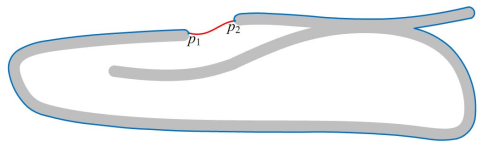


Fig. 4. The description of two broken layers which should be linked.

layer boundary at two points $p_1$ and $p_2$. Such two points are potentially linked by the skeleton part with the width of the parchment layer, if one of the following criteria are satisfied: i) on the layer boundary no path can be found to connect $p_1$ and $p_2$, or ii) the ratio of the skeleton part length to the length of the shortest path between $p_1$ and $p_2$ on the layer boundary (the blue curve in Fig. 4) is less than a threshold value (we set as 0.5). The broken layers in Fig. 3 are linked by the red segments. This approach guarantees the consistency of the topological structure of the parchment layer, and with the use of skeleton segments for linking, works well in our experiments as such layer breaking is usually short.

### B. Global Optimisation Based Segmentation Refinement

In this section we aim to separate those fused regions into several parchment layers to ensure topological correctness. The main steps of our segmentation refinement include detecting junction sections and fused regions, matching junction sections, separating fused regions using as-parallel-as-possible connecting curves, and missing boundary reconstruction based on skeleton.

*1) Junction Section Detection:* A junction point is a demarcation point between the fusing and separating of two adjacent layer boundaries. In the vicinity of such a point exists a set of points with large curvature. All these points constitute a so-called junction section. Due to the property of large curvature, we can use a purely geometric approach to detect each point in a junction section. Given a point $p_0$ on the boundary, we take $s$ pixels $\{p_{-s}, p_{-s+1}, \ldots, p_{-1}\}$ from the left neighborhood of $p_0$ on the boundary, and then take $s$ pixels $\{p_1, p_2, \ldots, p_s\}$ from its right neighborhood on the boundary (we set $s = 15$). If the intersection angle of vectors $p_0 p_s$ and $p_0 p_{-s}$ is less than a threshold value (set as 120° in our algorithm), the point $p_0$ will be considered as a point

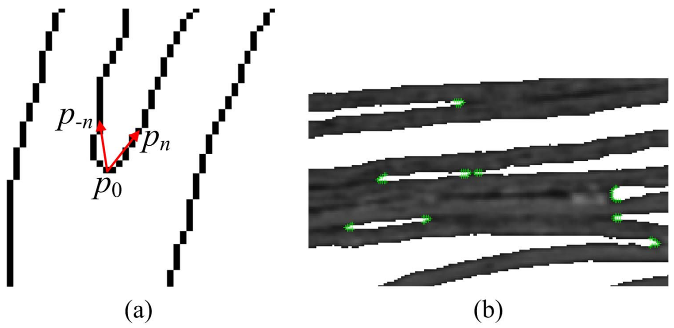


Fig. 5. Junction section detection. (a) Determining the point of a junction section. (b) Detected junction sections (indicated by green crosses).

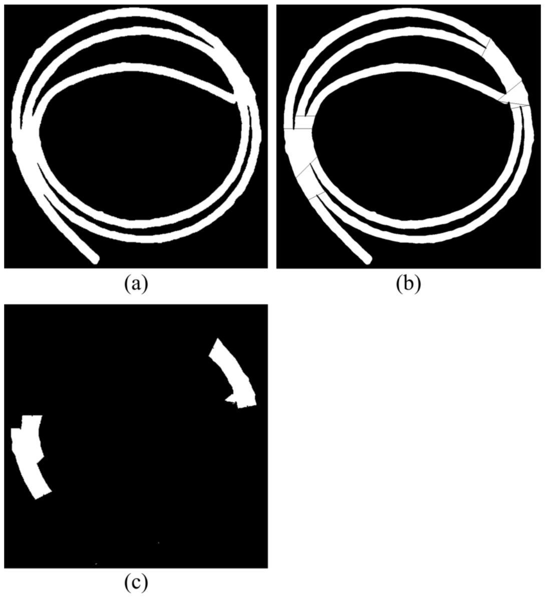


Fig. 6. Detection of fused regions. (a) The binary image of a parchment slice. (b) the separated binary image. (c) The detected fused region.

of a junction section. Figure 5 demonstrates all the detected junction sections. We cannot precisely determine which point is the junction point in a junction section, so the middle point of the junction section is approximately considered as the junction point.

*2) Fused Region Detection:* A fused region is formed by some parchment layers merged together. As long as the fused regions are detected, the stuck layers can be immediately recovered by separating the fused regions. In fact, the junction sections are caused by fused regions, so a fused region must contain the middle point of at least one junction section. Given a binary image of a parchment slice (Fig. 6(a)), we connect the two endpoints of each junction section by a line, and then cut off along this line the region where the junction section is located. Now the binary image has been separated into some small regions, as shown in Fig. 6(b). Consequently, the fused regions are the unions of those small regions which contain the middle points of junction sections (Fig. 6(c)).

*3) As-Parallel-as-Possible Curve Generation:* After detecting all fused regions, we will separate the fused regions by linking two junction sections with a curve which is as parallel as possible to the closest preserved boundaries to the two

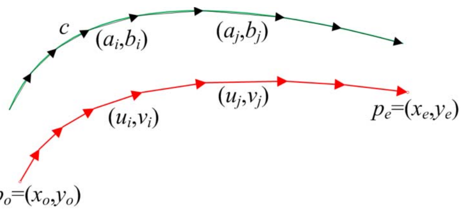


Fig. 7. $c$ and the curve connecting $p_o$ and $p_e$ are approximated by an $n$-sided polygonal curve.

junction sections. Such curves naturally separate fused regions. We first describe an algorithm to generate such curves and will later explain how matching junction sections are identified.

Assume that $c$ is an arbitrary curve and $p_o$, $p_e$ are two arbitrary points in the plane, whose coordinates are $(x_o, y_o)$ and $(x_e, y_e)$, as shown in Fig. 7. Note that we do not require the direction or the distance between the endpoints of $c$ and those of the new curve to be identical. Our aim is to link $p_o$ and $p_e$ by a curve which is as parallel as possible to curve $c$. This distinguishes our goal from the related work in the CAD/CAM literature on offset curves [23]. Not only can the latter only generate truly parallel curves (and hence cannot in general pass through two given points), but moreover they usually operate on parametric curves.

First of all we approximate the curve $c$ by an $n$-sided polygonal curve. The direction vector of each segment of the polygonal curve is represented as $(a_i, b_i)$, $i = 1, 2, \ldots, n$. Suppose that the curve connecting $p_o$ and $p_e$ can be approximated by an $n$-sided polygonal curve too, and the direction vector of each of its segments is denoted as $(u_i, v_i)$, $i$=1, 2, …, $n$, as illustrated in Fig. 7. Therefore, the fact that the curve which links $p_o$ and $p_e$ is the most parallel to curve $c$ is equivalent to the fact that the polygonal curve which links $p_o$ and $p_e$ is the most parallel to the polygonal curve which approximates curve $c$. Consequently, we obtain the cost function:

$$\begin{aligned} &\min \ \sum_{i=1}^{n}(u_i - a_i)^2 + (v_i - b_i)^2 \\ &\text{s.t.} \ \sum_{i=1}^{n} u_i = x_e - x_o, \quad \sum_{i=1}^{n} v_i = y_e - y_o. \end{aligned} \tag{1}$$

Equation 1 can be rewritten in the matrix form

$$\begin{aligned} &\min \ \mathbf{X}^T\mathbf{X} + 2\mathbf{P}^T\mathbf{X} + \mathbf{P}^T\mathbf{P} \\ &\text{s.t.} \ \mathbf{AX} = \mathbf{M} \end{aligned} \tag{2}$$

where, $\mathbf{X} = [u_1\ v_1\ u_2\ v_2\ \cdots\ u_n\ v_n]^T$, $\mathbf{P} = [-a_1\ -b_1\ -a_2\ -b_2\ \cdots -a_n\ -b_n]^T$, $\mathbf{M} = \begin{bmatrix} x_e - x_o \\ y_e - y_o \end{bmatrix}$, and $\mathbf{A} = \begin{bmatrix} 1\ 0\ 1\ 0\ \cdots \\ 0\ 1\ 0\ 1\ \cdots \end{bmatrix}$. Using Lagrange multipliers, we get

$$\begin{bmatrix} \mathbf{X} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} 2\mathbf{I} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} -2\mathbf{P} \\ \mathbf{M} \end{bmatrix} \tag{3}$$

where $\boldsymbol{\beta}$ is the vector consisting of two Lagrange multipliers and $\mathbf{I}$ is the identity matrix. By means of block matrix inversion [24], the coefficient matrix can be further represented

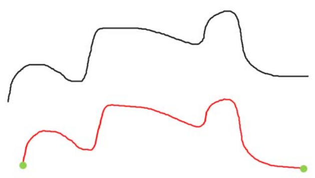


Fig. 8. The red curve connects the two green points, while being as parallel as possible to the black curve.

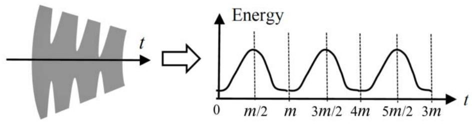


Fig. 9. The form of the shape energy in the interlayer connections between several fused layers.

as

$$\begin{bmatrix} 2\mathbf{I} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} 0.5\mathbf{I} - 0.5\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} & \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1} \\ -(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} & 2(\mathbf{A}\mathbf{A}^T)^{-1} \end{bmatrix}. \tag{4}$$

Noting $\mathbf{A}\mathbf{A}^T = n\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, we eventually obtain

$$\mathbf{X} = \left(-\mathbf{I} + \frac{1}{n}\mathbf{A}^T\mathbf{A}\right)\mathbf{P} + \frac{1}{n}\mathbf{A}^T\mathbf{M}. \tag{5}$$

Figure 8 shows a result of our as-parallel-as-possible curve generation method. Here the black curve is an arbitrary curve, the two green points are two arbitrary points, and the red curve connects the two green points and is as parallel as possible to the black curve.

*4) Shape Energy Calculation:* To evaluate the quality of connecting curves, we introduce a shape energy in this section. The shape energy of a point in a parchment layer reflects the closeness of the point to the estimated layer boundary. The closer the point is to the layer boundary, the lower the shape energy of the point is. Given an estimate of the parchment thickness $m$, we define shape energy by means of a Gaussian function. On the condition that there are interlayer connections between some layers, the shape energy on the transverse direction $t$ of the connected part of those layers should have the form demonstrated in Fig. 9.

Thus, given $d_x$ the perpendicular distance of each pixel $x$ in a layer to its closest boundary, the energy function has the following form

$$E(x) = \exp\left\{-\frac{\left(d_x - \frac{2k-1}{2}m\right)^2}{2\sigma^2}\right\}, \tag{6}$$

where $k$ is the parchment layer counter, $k = 1, 2, 3 \ldots$, chosen to satisfy the following constraint

$$(k-1)m < d_x < km, \tag{7}$$

$\sigma$ is the parameter which we estimate as $\sigma = m/3$, so that almost 99.7% of the energy of Gaussian function will lie within the layer. It can be easily seen from Eq. 6 that $0 < E(x) \leq 1$. A result of shape energy calculation is shown in Fig. 10. It is evident that the shape energy reaches its peak in the middle of layer, and diminishes progressively from the middle of the layer towards the boundary. It works not only for a single layer but also for cases when multiple layers are fused together.

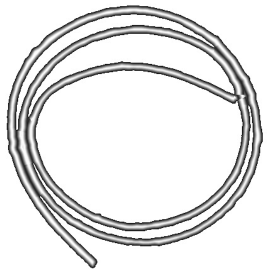


Fig. 10. The shape energy computed only within the segmentation mask.

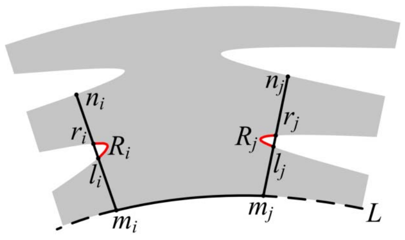


Fig. 11. Two junction sections on different boundaries but in the same fused region have the same closest preserved boundary.

*5) Junction Section Matching and Fused Region Separation:* We now describe our method for recreating the missing boundaries. Our strategy is to first identify all the junction sections that can be potentially matched, without violating topological constraints. We then work out the matching cost. Then we treat junction section matching as a global optimisation that maximises the total matching weights. The matched junction sections are then used to separate fused regions.

Provided there are $\hat{r}$ junction sections in the image, we initialise an $\hat{r} \times \hat{r}$ cost matrix $\mathbf{W}$ with each element as negative infinite and update connectable junction section pairs with a cost reflecting the quality of connecting curves. We find the closest preserved boundaries for each junction section as follows. As depicted in Fig. 11, given a junction section $R_i$, whose two endpoints are $l_i$ and $r_i$, a line which passes through $l_i$ and $r_i$ meets the left closest boundary at $m_i$ and the right closest boundary at $n_i$, then these two closest boundaries on two sides of $R_i$ are the closest preserved boundaries of $R_i$. Topologically, two junction sections may be matched only if they are located on different boundaries but on the same fused region, as illustrated in Fig. 6. If two junction sections in different boundaries and in the same fused region have the same closest preserved boundary, we will check whether their middle points may be linked to reconstruct missing boundaries.

Providing that there exist two junction sections $R_j$ and $R_i$, $i > j$, which are on different boundaries but in the same fused region, and have the same closest preserved boundary (Fig. 11), the two intersection points $m_i$ and $m_j$ respectively corresponding to $R_i$ and $R_j$ separate the boundary $L$ into two parts, which are represented by the solid line and dashed line respectively in Fig. 11. We only take into account the part which completely lies between the two lines $m_i l_i$ and $m_j l_j$, that is, the solid part in Fig. 11.

First of all, we use the solid line part to generate a curve $Q$ which connects the middle points of $R_i$ and $R_j$ using an as-parallel-as-possible curve (section II-B.3), and then check whether $Q$ intersects the existing boundaries at any places other than $R_i$ and $R_j$. If not, there is a possibility that the region between $R_i$ and $R_j$ is an interlayer connection, so we calculate the energy between $R_i$ and $R_j$ along curve $Q$ by the following form.

$$H = \left[1 - \frac{1}{g}\sum_{s=1}^{g} E(x_s)\right] + \lambda \frac{\min(m_i l_i, m_j l_j)}{\max(m_i l_i, m_j l_j)}, \quad (8)$$

where $x_s$ denotes the pixel of $Q$, $g$ is the number of pixels in $Q$, and $E$ is the shape energy calculated by Eq. 6. In Eq. 8, $\frac{1}{g}\sum_{s=1}^{g} E(x_s)$ measures the mean shape energy of $Q$, thus the larger value of the first term means that the curve Q is closer to the missing boundary; the second term $\frac{\min(m_i l_i, m_j l_j)}{\max(m_i l_i, m_j l_j)}$ reflects the similarity of the distances from $R_i$ and $R_j$ to their closest preserved boundary. The more similar the distances from $R_i$ and $R_j$ to their closest preserved boundary, the larger this term. $\lambda$ is a parameter which specifies the significance between these two terms. Both terms have the same scale, and we choose in all experiments to give them equal weight and set $\lambda = 1$. If $H > \mathbf{W}(i, j)$, we will set $\mathbf{W}(i, j) = H$.

After we apply the above method to all possible pairs of junction sections, the fused region can be separated along the parallel curve linking the middle points of two matched junction sections. We match the junction sections by a global graph matching algorithm. Let $G = (V, E)$ be an undirected weighted graph, where $V$ denotes the set of nodes, which consist of all of the junction sections; $e_{ij} \in E$ represents the edge linking two junction sections $R_i$ and $R_j$ with the weight $w(e_{ij}) = \mathbf{W}(i, j)$; negative infinite $\mathbf{W}(i, j)$ means $R_i$ and $R_j$ are disjoint. The required matching is a subset of edges $E' \subseteq E$ such that each node in $V$ has at most one incident edge in $E'$ and the sum of the weights of the edges in $E'$ is maximised. This can be solved efficiently using Edmonds' blossom algorithm [25], [26], which is based on the following linear programming formulation, where $\mathbf{x}$ represents the incidence vector of matching [27]:

$$\begin{aligned} &\max \sum_{i,j=1}^{\hat{r}} w(e_{ij})\mathbf{x}(e_{ij}) \\ &\text{s.t. } 0 \le \mathbf{x}(\mathrm{e_{ij}}) \le 1 \\ &\quad \sum_{i=1}^{\hat{r}} \mathbf{x}(e_{ij}) \le 1, 1 \le j \le \hat{r} \\ &\quad \sum_{i,j=1}^{\hat{r}} \mathbf{x}(e_{ij}) \le (|B|-1)/2, \quad e_{ij} \in E(B),\ \forall B \in \nu_{odd} \end{aligned} \quad (9)$$

where $\nu_{odd}$ is the set of all odd-size subsets of $V$. Edmonds [28] proved that with the third constraint, the basic solutions to the resulting linear programming are integral. We use the implementation in [25] to obtain the matching $E'$. The fused region is then separated along the parallel curve linking the middle points of two matching junction sections $R_u$ and $R_v$ whose edge has the maximum weight among all the edges in $E'$. Subsequently we eliminate the $u$th row, $v$th column, $u$th column and $v$th row of matrix $\mathbf{W}$, then update the already existing fused regions and let the algorithm begin all over again. The algorithm will terminate when the maximum element of $\mathbf{W}$ is negative infinite, which means that there are no junction sections to be matched, or there is a single boundary in the image. Figure 12 illustrates a segmentation result of our algorithm.

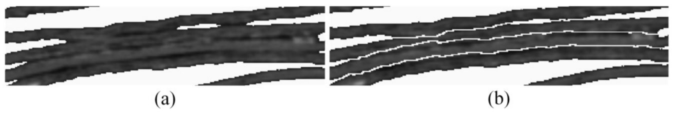


Fig. 12. The result of our segmentation for a parchment slice. (a) A fused region in the original image. (b) The segmentation of the fused region.

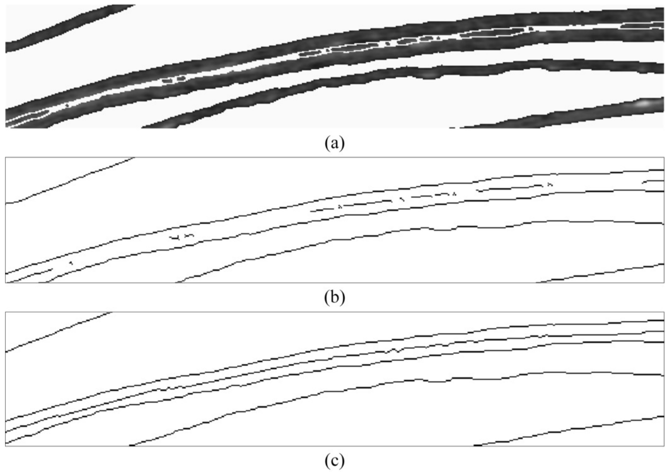


Fig. 13. Skeleton connection. (a) The segmented parchment slice. It can be seen that a layer is mistakenly broken. (b) The skeleton image of the broken layer. (c) The linked skeleton provided by our algorithm.

*6) Skeleton Connection:* After finishing segmentation, we need to extract the skeleton of the parchment layer for virtual unrolling. However, sometimes some parts of the parchment layer are too thin to provide sufficient single pixel edges, and then our segmentation algorithm will eliminate such very thin parts to guarantee the junction section detection and missing boundary reconstruction. The skeletons of these broken layers will mislead the virtual unrolling method to generate a wrong result. Thus before virtual unrolling, we need to link the skeletons of the layers which are mistakenly broken. The linkage method proposed in [22] is adopted to connect such broken skeletons, since this method can effectively ensure the correctness of the topological structure of the skeleton. Figure 13 illustrates the effect of our skeleton connection.

## III. Virtual Unrolling of Parchments

Although Samko *et al.*'s [12] approach can achieve virtual unrolling, the method is based on generating tetrahedral meshes and nonlinear multidimensional scaling (MDS). As a result, it is extremely slow, and takes 6.4 hours for the small scroll in Fig. 2ab, and 4.5 weeks for the Bressingham dataset. Thus in this section, we introduce an efficient high quality virtual unrolling method based on the extracted skeletons, which is 300-1,000 times faster. We first extract a pixel sequence from the skeleton of each slice. Then we formulate the alignment problem of skeletons of adjacent slices as an optimal matching problem, which can be efficiently solved using dynamic programming. Based on the matching, we map each skeleton to a row of pixels in the reconstructed image after virtual unrolling such that the alignment is followed and the overall scaling is minimised. Finally, the pixels in the reconstructed image are recovered by ink projection.

### A. Skeleton Sequence Extraction

We start from the skeleton images extracted in the previous step. Based on 8-way connectivity, the skeleton in each slice can be connected to form a graph with skeleton pixels as nodes and edges connecting adjacent pixels. Such graphs however usually contain branches, which make alignment difficult. Since we know in advance that the skeleton of each slice should contain a single sequence of pixels, we find the longest path in each graph. The longest path problem for a general graph is expensive. Since our graphs are typically graphs with extra small branches, we use a simple heuristic to efficiently find an approximate solution (which takes about 40 seconds per image for the Bressingham scroll). Given a skeleton, starting from an arbitrary pixel $\tilde{p}_0$, we then find the furthest point on the skeleton to it, giving $\tilde{p}_1$, and the furthest to $\tilde{p}_1$, denoted as $\tilde{p}_2$, etc. until convergence. The longest path is obtained as the path connecting the last two points in this process. The resulting sequence of pixels for the $i$-th skeleton $S_i$ is denoted by $\{\mathbf{p}_k^{(i)}\}$, where $k = 1, 2, \ldots, n_i$, and $n_i$ is the number of points on the $i$-th skeleton. Such a method is effective in finding a path that represents each skeleton. However, depending on the iterative process, the sequence may be in reverse orders (clockwise or counterclockwise). Treating virtual unrolling as a matching process, it is essential to choose the order in a consistent way. To achieve this, we work out the total turning angle $\gamma_i$ for the $i$-th skeleton as:

$$\gamma_i = \sum_{k=2}^{n_i-1} \gamma\,(\mathbf{p}_k^{(i)} - \mathbf{p}_{k-1}^{(i)}, \mathbf{p}_{k+1}^{(i)} - \mathbf{p}_k^{(i)}),$$

where $\gamma\,(\mathbf{v}_1, \mathbf{v}_2)$ is the angle between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ with the minimum absolute value, and $\gamma\,(\mathbf{v}_1, \mathbf{v}_2)$ is positive if it turns counterclockwisely from $\mathbf{v}_1$ to $\mathbf{v}_2$ and negative otherwise. If the total turning angle $\gamma_i < 0$, we simply reverse the order of pixels for $S_i$. For simplicity, we hereafter refer to the adjusted skeleton point sequences as $\{\mathbf{p}_k^{(i)}\}$. Using total turning angles makes the approach robust to local fluctuations of orientation. To cope with jaggedness caused by skeleton discretisation as well as inaccuracies in extraction, we apply boxcar smoothing to the skeleton as a polyline, followed by resampling the smoothed polyline with even spacing. The resampled points are denoted as $\tilde{\mathbf{P}}^{(i)} = \{\tilde{\mathbf{p}}_k^{(i)}\}$.

### B. Parchment Slice Alignment

Although in principle it is possible to consider alignment of all the slices simultaneously, for large datasets this can be prohibitively expensive. We thus first consider alignment of

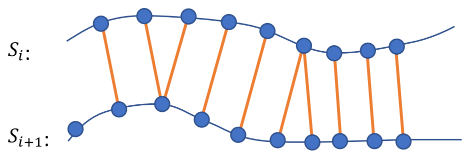


Fig. 14. Illustration of matching between adjacent skeleton sequences. Each point is matched to one or two points on the other curve, with the exception of near end points.

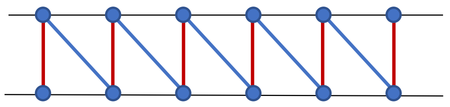


Fig. 15. Illustration of edge lengths used for setting $\tau$.

two adjacent slices, which is computationally manageable, and then align them globally. Doing so also allows parallelisation of matching for speedup. Assuming we are matching $\tilde{\mathbf{P}}^{(i)}$ with $\tilde{\mathbf{P}}^{(i+1)}$ we aim to find a matching that minimises total matching costs. As illustrated in Fig. 14, since we are matching two sequences of points, it is reasonable to further require such matching to be in sequence. Every point $\tilde{\mathbf{p}}_k^{(i)}$ should be matched to at least one and at most two points in $\tilde{\mathbf{P}}^{(i+1)}$. Since sample points are equally spaced, ideally points should be in one-to-one correspondence. In reality, to cope with local stretching and shrinking, e.g. due to skeleton inaccuracies, we allow one to two matching. This is more than adequate, as that allows a skeleton to expand or shrink by a factor of 2 between adjacent slices, which is unlikely to happen. The same rule applies to matching points in $\tilde{\mathbf{P}}^{(i)}$ for points in $\tilde{\mathbf{P}}^{(i+1)}$. There is one exception to this general rule, which happens at the ends of the sequences where a small number of sample points ($\tilde{n}_i$ set to 5% of $n_i$ in our experiments) can be ignored in matching. This is useful to cope with cases where the reconstructed image naturally has a non-rectangular shape. The matching of two sequences can be represented using an edge set $\mathcal{E}$ where each element $(k, t) \in \mathcal{E}$ means $\tilde{\mathbf{p}}_k^{(i)}$ is matched to $\tilde{\mathbf{p}}_t^{(i+1)}$. Given $(k, t) \in \mathcal{E}$, the next matched pair can only be $(k+1, t)$, $(k, t+1)$ or $(k+1, t+1)$. Cases 1 and 2 have only one matching point advanced, whereas case 3 has both matching points advanced. For simplicity, we split $\mathcal{E}$ into two subsets, $\mathcal{E}_1$ and $\mathcal{E}_2$ to refer to both situations. We define the matching energy as the sum of total edge lengths, although we use a different weight to penalise edges in $\mathcal{E}_1$ as they are related to non-isometric scaling.

$$E(\mathcal{E}) = \tau \sum_{(k_1,t_1)\in\mathcal{E}_1} \|\tilde{\mathbf{p}}_{k_1}^{(i)} - \tilde{\mathbf{p}}_{t_1}^{(i+1)}\| + \sum_{(k_2,t_2)\in\mathcal{E}_2} \|\tilde{\mathbf{p}}_{k_2}^{(i)} - \tilde{\mathbf{p}}_{t_2}^{(i+1)}\|. \tag{10}$$

In an ideal scenario where two sequences are well aligned (see Fig. 15), when all the edges are in $\mathcal{E}_2$, assume the total energy is $E_0$. If we instead take both the red and blue edges, making every edge in $\mathcal{E}_1$, the total energy will be $\tau(1+\sqrt{2})E_0$. To penalise $\mathcal{E}_1$ cases, we need to ensure $\tau(1+\sqrt{2})E_0 > E_0$, i.e. $\tau > \frac{1}{1+\sqrt{2}} \approx 0.414$. In practice, we do not penalise this too much to make sure $\mathcal{E}_1$ edges still happen to improve alignment. $\tau = 0.45$ works well and is used in all our experiments.

Global optimisation of $E(\mathcal{E})$ can be efficiently achieved using dynamic programming. To create a nested optimal substructure, we denote by $E^*_{k,t,c}$ the optimal solution to the subproblem of matching two subsequences $\tilde{\mathbf{p}}^{(i)}_1, \tilde{\mathbf{p}}^{(i)}_2, \ldots, \tilde{\mathbf{p}}^{(i)}_k$, and $\tilde{\mathbf{p}}^{(i+1)}_1, \tilde{\mathbf{p}}^{(i+1)}_2, \ldots, \tilde{\mathbf{p}}^{(i+1)}_t$, with the last edge being of case $c$ ($c = 1, 2, 3$ as defined above). This is initialised as:

$$E^*_{1,t,c} = 0 (1 \le t \le \tilde{n}_{i+1}), \quad E^*_{k,1,c} = 0 (1 \le k \le \tilde{n}_i), \ \forall c \tag{11}$$

to allow extensions at endpoints. The remaining $E^*_{k,t,c}$ can be worked out using the following

$$\begin{aligned} E^*_{k,t,1} &= \min(E^*_{k-1,t,2}, E^*_{k-1,t,3}) + \tau \|\tilde{\mathbf{p}}^{(i)}_k - \tilde{\mathbf{p}}^{(i+1)}_t\|, \\ E^*_{k,t,2} &= \min(E^*_{k,t-1,1}, E^*_{k-1,t,3}) + \tau \|\tilde{\mathbf{p}}^{(i)}_k - \tilde{\mathbf{p}}^{(i+1)}_t\|, \\ E^*_{k,t,3} &= \min(E^*_{k-1,t-1,1}, E^*_{k-1,t-1,2}, E^*_{k-1,t-1,3}) \\ &\quad + \tau \|\tilde{\mathbf{p}}^{(i)}_k - \tilde{\mathbf{p}}^{(i+1)}_t\|. \end{aligned} \tag{12}$$

Based on the case $c$ of the last edge, the previous edge can be deduced. If the last edge is of case 3, the previous edge can be of any case; otherwise, the previous edge cannot be of the same case to ensure one point will not be matched to more than two points. The optimal solution is the minimum of $E^*_{n_i,t,c}$ for $n_{i+1} - \tilde{n}_{i+1} + 1 \le t \le n_{i+1}, \forall c$ and $E^*_{k,n_{i+1},c}$ for $n_i - \tilde{n}_i + 1 \le k \le n_i, \forall c$, to allow flexible extension at endpoints. Given the minimum $E^*$, the optimal matching can be obtained by tracing backwards based on optimal values. The time complexity of the globally optimal matching algorithm is $O(n_i n_{i+1})$ which is very efficient.

The above scheme is related to the dynamic time warping (DTW) approach that is commonly applied to find a mapping from one signal to another [29]. However, the standard DTW scheme does not incorporate our preference for one-to-one mapping, restriction beyond one-to-two mappings, or cope with missing data at the ends of the sequences.

## C. Image Formulation and Ink Projection

Once all the pairwise matchings between adjacent slices are performed, we can map them to 2D image space where the $i$-th skeleton is mapped to the $i$-th row in the image. We start with the first skeleton $S_1$ and first assume that no distortion is involved, so the $k$-th point $\tilde{\mathbf{p}}^{(1)}_k$ is mapped to the $k$-th column of the first row. Since in the majority of cases the points have one-to-one correspondence, we can work out the mapping of successive skeletons one by one, by aligning those pixels with known correspondence at the same column and interpolating the mapping in between. More specifically, assuming for the $i$-th skeleton, point $\tilde{\mathbf{p}}^{(i)}_k$ is mapped to column $x^{(i)}_k$. When considering the $(i+1)$-th skeleton, there are three possibilities: 1) point $\tilde{\mathbf{p}}^{(i+1)}_t$ is mapped to a specific point $\tilde{\mathbf{p}}^{(i)}_k$, we set $x^{(i+1)}_t = x^{(i)}_k$. 2) $\tilde{\mathbf{p}}^{(i+1)}_t$ is not mapped to any point in $\tilde{\mathbf{P}}^{(i)}$, and it is between two points $\tilde{\mathbf{p}}^{(i+1)}_{t_1}$ and $\tilde{\mathbf{p}}^{(i+1)}_{t_2}$ which are mapped to $\tilde{\mathbf{p}}^{(i)}_{k_1}$ and $\tilde{\mathbf{p}}^{(i)}_{k_2}$, respectively. We use linear interpolation to work out the 2D image location as

$$x^{(i+1)}_t = \frac{x^{(i)}_{k_1}(t_2 - t) + x^{(i)}_{k_2}(t - t_1)}{t_2 - t_1}. \tag{13}$$

3) If $\tilde{\mathbf{p}}^{(i+1)}_t$ is not mapped to any point in $\tilde{\mathbf{P}}^{(i)}$ and only one side in the sequence has a (nearest) point $\tilde{\mathbf{p}}^{(i+1)}_{t_1}$ mapped to $\tilde{\mathbf{p}}^{(i)}_{k_1}$. This happens when the point being considered is towards one end of the skeleton. We use the mapping of $\tilde{\mathbf{p}}^{(i+1)}_{t_1}$ as reference and assume isometric mapping:

$$x^{(i+1)}_t = x^{(i)}_{k_1} + (t - t_1). \tag{14}$$

This process determines a mapping from the 2D image space to the 3D volume space. For each pixel, assuming the 3D location is $\hat{\mathbf{p}}$ with a normal direction $\hat{\mathbf{n}}$, we start from $\hat{\mathbf{p}}$ and move along $\hat{\mathbf{n}}$ by a maximum of $\hat{d}$ pixels, using a subvoxel step size to avoid aliasing effect. 0.25 is used in our experiments; smaller step size gives almost the same results, with slightly longer running time. We take the maximum intensity of all the sample points, obtained using trilinear interpolation from voxels at neighbouring integer grid positions. The distance $\hat{d}$ ideally can be chosen as $\frac{m}{2}$ where $m$ is the layer thickness. However, the skeleton may not lie exactly in the centre of each parchment layer and the parchment thickness may vary. To cope with this, we use a larger $\hat{d}$: for single-sided parchment, it is safe to set $\hat{d} = m$ without mixing writing on two layers. We also use the segmentation mask that separates layers and stop sampling along the normal direction if the sampling process enters the background.

Since the first skeleton $S_1$ is not distortion free, we further apply a global column-wise rescaling such that overall distortion across all the slices are minimised. For each pair of pixels in the $k$-th and $(k+1)$-th columns, we work out the distance for the corresponding points in the 3D space (using interpolation when needed). The average of such distances provides the column-wise scaling $\bar{s}_i$, i.e. after mapping $\tilde{x}_{k+1} - \tilde{x}_k = \bar{s}_i$. The column is no longer in the integer position, so we apply bilinear interpolation to obtain the intensity values.

The pixels of the obtained image are brighter if there is ink. To make the image look more natural, we invert the pixel intensities so that ink appears dark, and further apply intensity scaling to enhance contrast.

## IV. EXPERIMENTS

We demonstrate the performance of our segmentation method to real parchments, which vary in size and complexity. X-ray images of the parchments were acquired through tomographic development in the School of Medicine and Dentistry at QMUL [7]. Our algorithm is tested on a desktop PC with a 2.9GHz processor. The method presented by Samko *et al.* [11] and [12] is adopted as a reference method, because this work is the most relevant to parchment segmentation and virtual reconstruction. The technique reported by Baum *et al.* [18] is the most recent and effective algorithm for virtually revealing text from rolled scrolls that we found, and was thus used as a further reference method for comparison. In addition, we also include the graph cut [14], snake method [13], and Otsu thresholding algorithm [20] as reference methods since these methods are widely cited in the literature and are often used as baseline methods for comparison.

The first experiment is conducted to test our segmentation method with three parchment scrolls from [11], [12]: the small

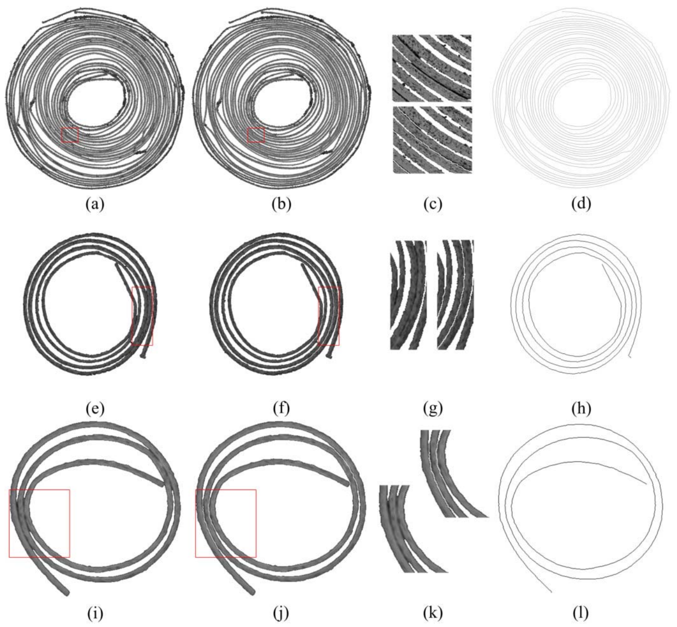


Fig. 16. The results of our segmentation method applied to three different parchments. (a)(e)(i) the foregrounds extracted by Otsu thresholding, (b)(f)(j) the segmented parchments by our method, (c)(g)(k) close-up views of some areas, (d)(h)(l) skeleton images.

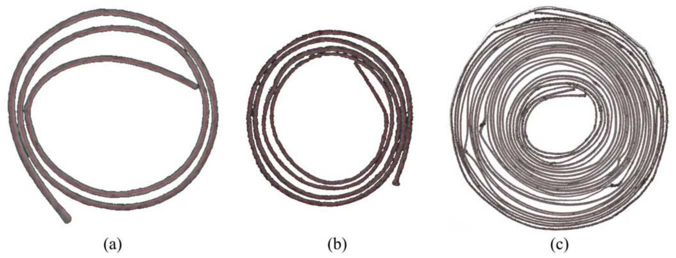


Fig. 17. The skeletons of (a) small scroll, (b) medium scroll, and (c) large scroll estimated by Baum et al's method, which are highlighted by the red lines and superimposed on the foreground of the parchments.

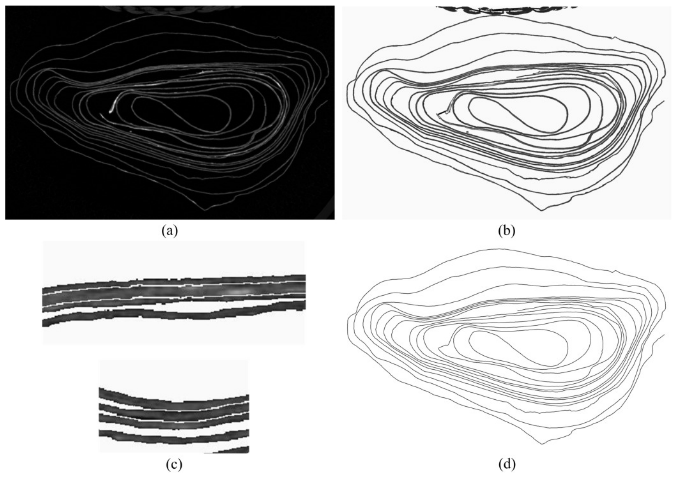


Fig. 18. Segmentation for Bressingham scroll. (a) original image, (b) segmented image, (c) close-ups of some areas, (d) skeleton.

scroll, medium scroll, and large scroll. These scrolls exhibit increasing complexity, and were used by Samko et al. for testing their algorithm development. The first two scrolls are straightforward, being relatively small without tight connections or delamination, and their layers are relatively thick, even and almost complete. The large scroll is also without delamination, but contains more touching layers. However, these are not physically fused, and therefore not as tight or compressed as those in the Bressingham scroll. The sizes of the three parchments are 708 frames with resolution $430 \times 430$, 700 frames with resolution $530 \times 530$, and 423 frames with resolution $1702 \times 1732$ respectively. The largest fused region in those parchments consists of three layers, and each layer is about 14 pixels wide ($m = 14$ in Eq. 6). Figure 16 illustrates the stages of our segmentation method on the three parchments. It can be seen that all the fused regions are correctly separated into several layers. Because the segmentation results for all the slices are quite similar, the example in Fig. 16 represents the performance of our algorithm for the whole data set. It is noteworthy that our segmentation results look similar to Samko *et al.*'s results in [11] and [12] showing that our method can deal with the parchments which Samko et al.'s method can process. Baum *et al.*'s method is applied to each scroll with two slices manually initialised by the initialisation strategy in [18]. Figure 17 demonstrates the segmentation results of Baum *et al.*'s method. It can be seen that Baum *et al*'s method can provide a correct estimation of the skeleton of the parchment layer, which is highlighted by the red lines. The segmentation results of other reference methods can be found in [12].

To provide quantitative evaluation of all the segmentation algorithms, we manually segment twelve images from each set to obtain ground truth segmentations. These are compared using multiple benchmark criteria: Rand Index (RI), which is a measure of similarity of two data clusterings, Variance of Information (VI) [30], which describes the distance of two data clusterings, and three commonly used statistics: Precision (P), Recall (R) and F-measure (F) [31]. Precision, recall and the F-measure are applied directly to the segmentations and provide an indication of the per-pixel classification accuracies of parchment and air (background) segmentation. Low precision indicates that air is misclassified as parchment, while low recall indicates that parchment is misclassified as air. Since topological correctness is critical for the subsequent reconstruction process we also perform a connected component labelling on the segmented images (on both foreground and background), and compare the labellings obtained by the segmentation algorithms against the ground truth segmentation to emphasise errors in connectivity. The Rand Index and Variance of Information are appropriate measures to use since they are invariant to permutations of the labels.

The averages of the five measures for all algorithms in this test are listed in Table I, which shows that Baum et al's method obtains the best segmentation accuracy for the small scroll and medium scroll. This is because these two scrolls perfectly meet the prerequisite of Baum et al's method that the consecutive slices show greatly similar spirals. Nonetheless, our segmentation method results in close-to-the-best values for the five measures of these two scrolls. In contrast, for the large scroll, which is more damaged than the other two scrolls and

TABLE I

QUANTITATIVE COMPARISON OF DIFFERENT SEGMENTATION ALGORITHMS FOR THE THREE SCROLLS. THE BEST MEASURES ARE SHOWN IN BOLD

| | Small Scroll (e.g. Fig. 2a&b) | | | | | | Medium Scroll (e.g. Fig. 18 in [12]) | | | | | | Large Scroll (e.g. Fig. 20 in [12]) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Samko | Graph Cut | Snakes | Otsu | Baum | Our Method | Samko | Graph Cut | Snakes | Otsu | Baum | Our Method | Samko | Graph Cut | Snakes | Otsu | Baum | Our Method |
| RI | 0.9599 | 0.5987 | 0.5851 | 0.6015 | **0.9923** | 0.9903 | 0.8830 | 0.6695 | 0.6651 | 0.6770 | **0.9717** | 0.9701 | 0.8843 | 0.6629 | 0.6233 | 0.6438 | 0.9467 | **0.9617** |
| VI | 0.2392 | 1.4200 | 1.6272 | 1.3724 | **0.0668** | 0.0807 | 0.4642 | 1.3458 | 1.3935 | 1.2214 | **0.1206** | 0.1329 | 0.8694 | 2.8714 | 3.4087 | 3.0474 | 0.4425 | **0.3619** |
| P | **0.9978** | 0.9959 | 0.9455 | 0.9953 | 0.9975 | 0.9969 | **0.9946** | 0.9899 | 0.9282 | 0.9903 | 0.9907 | 0.9853 | 0.9902 | **0.9927** | 0.9228 | 0.9756 | 0.9888 | 0.9728 |
| R | 0.9016 | 0.9622 | 0.9183 | **0.9855** | 0.9835 | 0.9793 | 0.8638 | 0.8697 | 0.8985 | 0.9558 | 0.9899 | **0.9907** | 0.9140 | 0.9152 | 0.9050 | 0.9581 | 0.9321 | **0.9707** |
| F | 0.9473 | 0.9787 | 0.9316 | 0.9904 | **0.9905** | 0.9880 | 0.9246 | 0.9259 | 0.9130 | 0.9727 | **0.9903** | 0.9880 | 0.9505 | 0.9521 | 0.9137 | 0.9666 | 0.9595 | **0.9717** |

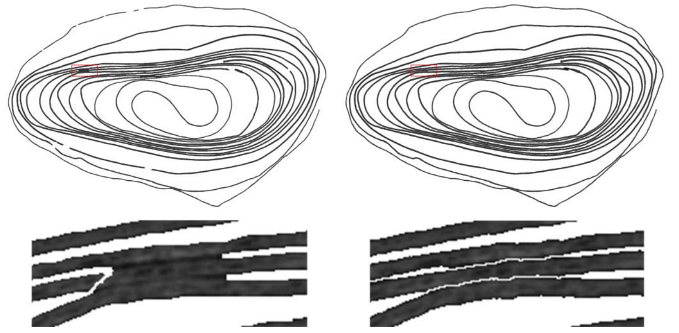

(a) (b)

Fig. 19. Our method compared with Samko et al.'s method. (a) The segmentation of Samko et al.'s method, with the region containing faulty segmentation highlighted. (b) The segmentation of our method.

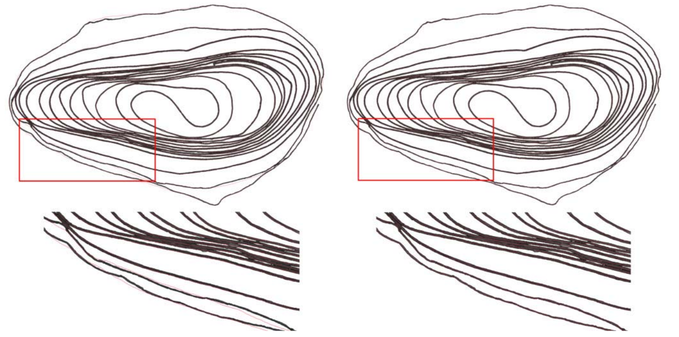

(a) (b)

Fig. 20. Our method compared with Baum et al.'s method. (a) The skeleton (the red line) from Baum et al.'s method, which is not completely contained in the parchment layer. (b) The skeleton from our method. Note that Baum et al.'s method needs substantial effort to manually initialise 35 key frames.

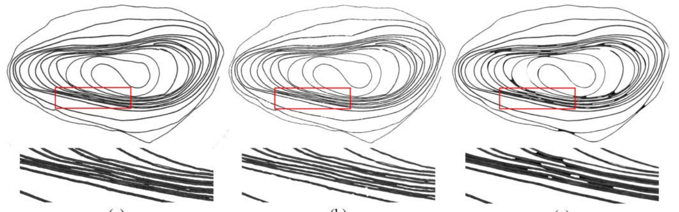

(a) (b) (c)

Fig. 21. The results of (a) Otsu thresholding, (b) graph cut, (c) snakes.

more tightly wound, our method achieves the most satisfactory segmentation.

The next experiment deals with the Bressingham scroll, which is much more challenging than the three scrolls used above. Because of mistreatment, there are several areas of the skin which have become scuffed and delaminated. In addition, many layers have become physically fused. The part of the parchment we are processing consists of 3070 frames, with $1256 \times 816$ pixels in each frame. The largest fused region is comprised of more than six layers. In addition, not only are the layers uneven, but also they are split into many parts. The average thickness of the layers is only about six pixels. Furthermore, there exist artifacts in some frames. All of these pose a serious challenge to the segmentation for this parchment. Because the layers are so thin, we resize the image to double the original width and height before processing. We set $m = 11$ in Eq. 6 for this data. Figure 18 demonstrates an example of the segmentation of a slice of the Bressingham scroll by our method. This figure indicates that our method can correctly divide the fused regions into several layers and then obtain the complete skeleton. The comparison of our method and Samko et al.'s method is illustrated in Fig. 19. It is clear in Fig. 19 that our method can correctly match the junction sections and separate the fused regions, while keeping the parchment complete; by contrast, not only does Samko et al.'s method break the layer which should be

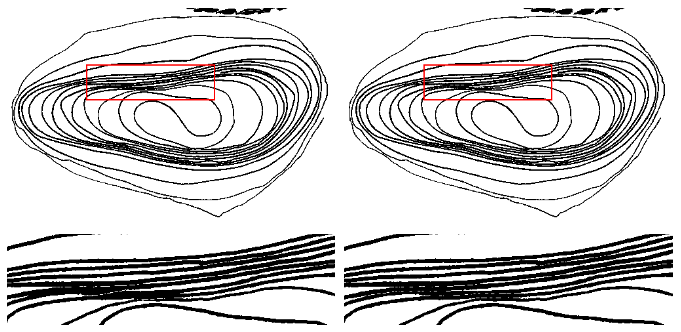

(a) (b)

Fig. 22. Our method compared with Liu *et al.*'s method [21]. (a) The segmentation of Liu *et al.*'s method, with the region containing faulty segmentation highlighted. (b) The segmentation of our proposed method.

TABLE II

QUANTITATIVE COMPARISON OF DIFFERENT SEGMENTATION ALGORITHMS FOR THE BRESSINGHAM SCROLL. THE BEST MEASURES ARE SHOWN IN BOLD

| | Samko | Graph Cut | Snakes | Otsu | Baum | Our Method |
|---|---|---|---|---|---|---|
| RI | 0.7338 | 0.4263 | 0.4591 | 0.4529 | 0.9696 | **0.9744** |
| VI | 1.0230 | 2.7136 | 2.6605 | 2.5483 | 0.2139 | **0.1883** |
| P | 0.8796 | 0.8798 | 0.8845 | 0.9331 | 0.9312 | **0.9442** |
| R | 0.9748 | **0.9866** | 0.8334 | 0.9638 | 0.9559 | 0.9617 |
| F | 0.9247 | 0.9297 | 0.8580 | 0.9481 | 0.9432 | **0.9528** |

TABLE III

QUANTITATIVE EVALUATION FOR DIFFERENT SEGMENTATION INITIALISATION ALGORITHMS. NOTE: UNLIKE OTHER MEASURES, FOR VI, SMALLER IS BETTER. THE BEST MEASURES ARE SHOWN IN BOLD

| | Small Scroll (e.g. Fig. 2a&b) | | | | Medium Scroll (e.g. Fig. 16e&f) | | | | Large Scroll (e.g. Fig. 16a&b) | | | | Bressingham Scroll (e.g. Fig. 2c&d) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Otsu+ Our Method | Otsu+ CED+ Our Method | GC+ Our Method | GC+ Shape+ Our Method | Otsu+ Our Method | Otsu+ CED+ Our Method | GC+ Our Method | GC+ Shape+ Our Method | Otsu+ Our Method | Otsu+ CED+ Our Method | GC+ Our Method | GC+ Shape+ Our Method | Otsu+ Our Method | Otsu+ CED+ Our Method | GC+ Our Method | GC+ Shape+ Our Method |
| RI | **0.9903** | 0.9903 | 0.9828 | 0.9598 | **0.9701** | 0.9640 | 0.9636 | 0.9496 | **0.9617** | 0.9607 | 0.9469 | 0.9243 | **0.9744** | 0.9717 | 0.9647 | 0.9648 |
| VI | **0.0807** | 0.0810 | 0.1245 | 0.2417 | **0.1329** | 0.1785 | 0.1783 | 0.2657 | 0.3619 | **0.3565** | 0.4229 | 0.5400 | **0.1883** | 0.1984 | 0.2307 | 0.2330 |
| P | 0.9969 | 0.9968 | **0.9972** | 0.9967 | **0.9853** | 0.9773 | 0.9682 | 0.9708 | 0.9728 | 0.9794 | 0.9924 | **0.9950** | **0.9362** | 0.9225 | 0.8988 | 0.9025 |
| R | **0.9793** | 0.9793 | 0.9603 | 0.9024 | **0.9907** | 0.9811 | 0.9893 | 0.9457 | **0.9707** | 0.9623 | 0.9288 | 0.8919 | 0.9571 | 0.9765 | **0.9781** | 0.9736 |
| F | **0.9880** | 0.9880 | 0.9784 | 0.9472 | **0.9880** | 0.9792 | 0.9787 | 0.9580 | **0.9717** | 0.9705 | 0.9593 | 0.9404 | 0.9465 | **0.9487** | 0.9365 | 0.9365 |

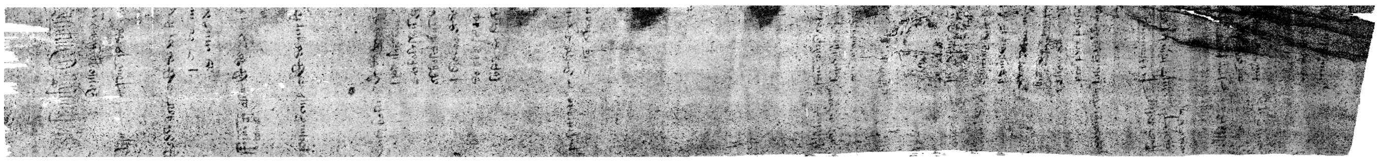


Fig. 23. The virtual unrolling of the Bressingham scroll using our proposed method.

complete, but it creates two false connections. The main reason for these false connections is that the boundary linking approach of Samko et al.'s method is based on the prerequisite that few interlayer connections are left after the graph cut with shape prior. However that precondition is not satisfied in the case of the Bressingham scroll since if we choose the parameters such that most of interlayer connections are removed, the layers will be divided into many small parts. Hence we have to make a trade-off between removing interlayer connections and keeping the layers complete. As a result, some of the junction points are relatively far away from each other, which has a detrimental influence on the postprocessing of Samko et al.'s method. In comparison, our shape-based cost function Eq. 8 and matching using the blossom algorithm guarantee the robustness of our segmentation method to the positions of the junction sections. Therefore, it can be concluded from Fig. 19 that our algorithm is effective to deal with the complicated parchment which Samko et al.'s method cannot cope with. In order to compare our method against Baum et al's method, 35 slices are manually initialised for Baum *et al.*'s method following the initialisation strategy in [18], which is a time-consuming task because the layer of this parchment is considerably long, complicated, and non-spiral. The skeleton of a cross section of the parchment obtained respectively by our method and Baum et al.'s method is exhibited by the red curve in Fig. 20. It can be observed that the skeleton estimated by Baum et al.'s method is not completely contained in the parchment layer. However, to improve the result of Baum et al.'s method will inevitably cost much longer time on additional manual initialisation. Therefore Baum et al.'s method becomes less practical as the parchment complexity increases. Figure 21 shows the segmentation results by Otsu thresholding, graph cut, and snakes. As observed, all these reference methods fail to separate the fused regions into several layers.

We also compare our proposed global optimisation based segmentation refinement with [21]. Liu *et al.* [21] are able to produce correct segmentation for most frames, but fail to produce topologically correct segmentation for a small number of slices in the Bressingham dataset. An example is shown in Fig. 22, where the method [21] has layers stuck in the highlighted region (left) whereas our proposed global optimisation produces topologically correct segmentation (right). Our method is able to produce topologically correct segmentation for the entire scan with $3,070$ slices.

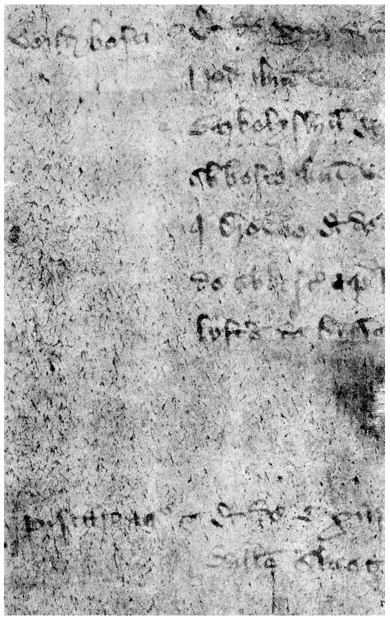


Fig. 24. An unseen section of the Bressingham scroll (that cannot be unrolled) which is visualised after virtual unrolling.

We manually segmented 14 slices of the Bressingham scroll for a quantitative evaluation of all the algorithms in this experiment, and the average values for each of the five measures are listed in Table II. All methods have relatively similar P, R, F values, but there are dramatic differences in RI and VI, as these measures are sensitive to topological errors. In terms of RI, VI, P, and F scores, our segmentation results

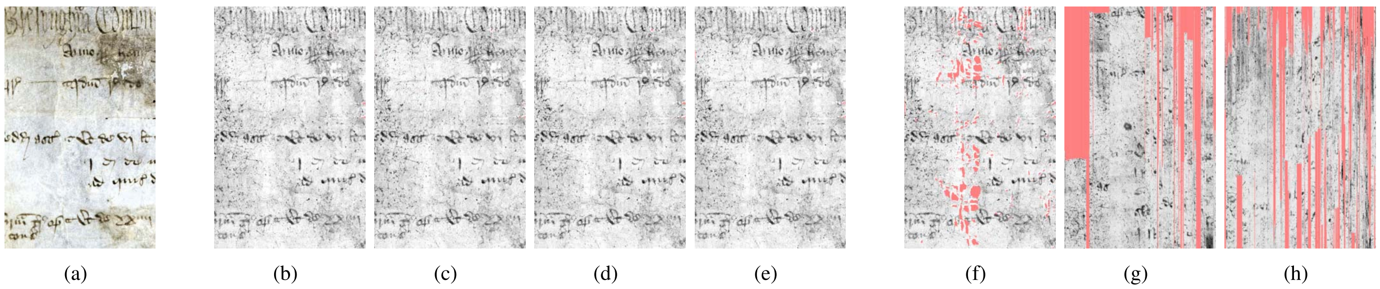


Fig. 25. A visible part of the Bressingham scroll (a) and its recovered image (after warping and histogram matching) from the XMT scan using the following methods: (b) Otsu + Our Method (c) Otsu + CED + Our Method (d) GC + Our Method (e) GC + Shape + Our Method (f) Baum (g) Samko (h) Snakes. Missing values due to holes in the reconstruction are coloured red. Note that Baum et al.'s method needs substantial effort to manually initialise 35 key frames.

TABLE IV

CORRELATION VALUES BETWEEN A PHOTOGRAPH AND RECONSTRUCTIONS FROM DIFFERENT SEGMENTATION ALGORITHMS FOR PART OF THE BRESSINGHAM SCROLL

| Otsu + Our Method | Otsu + CED + Our Method | GC + Our Method | GC + Shape + Our Method | Baum |
|---|---|---|---|---|
| 0.3584 | 0.3553 | 0.3550 | 0.3577 | 0.3539 |

outperform those obtained by the reference methods, which shows that the Bressingham scroll segmented by our method is the most similar to the ground truth.

Table III provides a quantitative comparison of the alternative approaches we have considered for initialising our segmentation algorithm. As can be seen, in most cases the Otsu initialisation without Coherence-Enhancing Diffusion (CED) filtering was most effective and was therefore selected as the default initialisation for our pipeline.

In order to verify the correctness of our segmentation and ink projection methods, we extracted the skeleton line from each segmentation result, and then use them to flatten the parchment scroll by the surface modelling and ink projection method to recover the text written on the parchment. Our virtual flattening and ink projection method is much more efficient than the method proposed in [12]. For the small scroll (Fig. 2a&b), [12] takes 6.4 hours, whereas our method only takes 22.8 seconds, which is over $1,000$ times faster. For the large Bressingham dataset with $3,070$ slices, our method takes 2.56 hours, whereas [12] takes 4.5 weeks. The flattened result of the Bressingham scroll after contrast enhancement is illustrated in Fig. 23. A representative part is illustrated in Fig. 24 which exhibits the recovery of an unseen section of the Bressingham scroll. Fig. 25a shows a photograph of a visible portion of the Bressingham scroll along with corresponding reconstructions obtained by applying several algorithms to the X-ray data. To facilitate quantitative comparison the reconstructions have been warped to align with the photograph (where possible) and their histograms have been matched to the photograph. Missing values in the reconstructions are coloured red, and it can be seen that segmentations from both Samko *et al.* [12] and snakes [13] produce poor reconstructions, making alignment with the photograph impractical. The method of Baum *et al.* [18] also results in many large holes. Table IV provides the Pearson correlation values between the reconstructions and the photograph of the visible section. The scores are generally low since there are inevitable differences in appearance between a photograph and a reconstruction based on X-ray density values. We note that although the versions of our proposed method achieve a slighter better score than that of Baum *et al.* [18] the difference is small despite the latter's holes. This is because the holes, while perceptually significant, cover less than 5% of the image, and so do not substantially reduce Baum *et al.*'s correlation value. Moreover, the holes fortuitously mostly occur in areas which do not contain text.

There are many blocks of text with clear visible letters on the virtually unrolled parchment, despite the parchment having many layers broken and stuck together. Thus Figs. 23–25 are strong evidence that our method correctly segments the parchment scroll for virtual unrolling.

## V. CONCLUSION

We have presented a novel method to virtually restore information from those parchments that cannot be manually read by processing their X-ray images. Our method segments images in five steps. First, we connect the layers of the parchment which are broken. Second, the junction sections are detected from the boundaries of the parchment. Then, the detected junction sections are matched by the blossom algorithm. Subsequently the fused regions are separated into several layers by means of the missing boundary reconstruction and parallel curve connection, and finally skeletons are connected. To virtually unroll parchments, the extracted skeletons are aligned using dynamic programming based global optimisation and parchment images are reconstructed using ink projection. Our method is tested on four different real scrolls: a small test scroll, a medium scroll, a large scroll, and the Bressingham scroll. The experimental results indicate that not only can our method process the parchments which have been processed previously, but it is capable of dealing with a more challenging historical parchment – the Bressingham scroll – and can make the physically unopenable scroll readable.

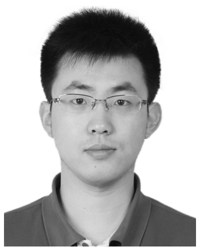
## REFERENCES

[1] D. Mills, A. Curtis, G. Davis, P. L. Rosin, and Y.-K. Lai, "Apocalypto: Revealing the Bressingham roll," *J. Paper Conservation*, vol. 15, no. 3, pp. 14–19, 2014.

[2] K. Pal, M. Terras, and T. Weyrich, "3D reconstruction for damaged documents: Imaging of the great parchment book," in *Proc. Int. Workshop Hist. Document Imag. Process.*, 2013, pp. 14–21.

[3] T. Wada, H. Ukida, and T. Matsuyama, "Shape from shading with interreflections under a proximal light source: Distortion-free copying of an unfolded book," *Int. J. Comput. Vis.*, vol. 24, no. 2, pp. 125–135, 1997.

[4] H. I. Koo, J. Kim, and N. I. Cho, "Composition of a dewarped and enhanced document image from two view images," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1551–1562, Jul. 2009.
[5] M. S. Brown, M. Sun, R. Yang, L. Yun, and W. B. Seales, "Restoring 2D content from distorted documents," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1904–1916, Nov. 2007.
[6] K. Pal, C. Schüller, D. Panozzo, O. Sorkine-Hornung, and T. Weyrich, "Content-aware surface parameterization for interactive restoration of historical documents," *Comput. Graph. Forum*, vol. 33, no. 2, pp. 401–409, 2014.
[7] G. R. Davis and J. C. Elliott, "High definition X-ray microtomography using a conventional impact X-ray source ," *J. Phys. IV France*, vol. 104, pp. 131–134, Mar. 2003.
[8] V. Mocella, E. Brun, C. Ferrero, and D. Delattre, "Revealing letters in rolled Herculaneum papyri by X-ray phase-contrast imaging," *Nature Commun.*, vol. 6, Jan. 2015, Art. no. 5895.
[9] D. Mills, G. R. Davis, Y.-K. Lai, and P. L. Rosin, "Apocalypto: Revealing lost text with XMT," *Proc. SPIE*, vol. 9212, Sep. 2014.
[10] F. Albertin *et al.*, "X-ray spectrometry and imaging for ancient administrative handwritten documents," *X-Ray Spectrometry*, vol. 44, no. 3, pp. 93–98, 2015.
[11] O. Samko, Y.-K. Lai, D. Marshall, and P. L. Rosin, "Segmentation of parchment scrolls for virtual unrolling," in *Proc. Brit. Mach. Vis. Conf.*, 2011, pp. 1–11.
[12] O. Samko, Y.-K. Lai, D. Marshall, and P. L. Rosin, "Virtual unrolling and information recovery from scanned scrolled historical documents," *Pattern Recognit.*, vol. 47, no. 1, pp. 248–259, 2014.
[13] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.
[14] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proc. Int. Conf. Comput. Vis.*, vol. 1. Jul. 2001, pp. 105–112.
[15] R. Samet, I. A. A. Askerbeyli, and C. Varol, "An implementation of automatic contour line extraction from scanned digital topographic maps," *Appl. Comput. Math*, vol. 9, no. 1, pp. 116–127, 2010.
[16] D. Allegra *et al.*, "Virtual unrolling using X-ray computed tomography," in *Proc. Eur. Signal Process. Conf.*, Aug./Sep. 2015, pp. 2864–2868.
[17] W. B. Seales, C. S. Parker, M. Segal, E. Tov, P. Shor, and Y. Porath, "From damage to discovery via virtual unwrapping: Reading the scroll from En-Gedi," *Sci. Adv.*, vol. 2, no. 9, p. e1601247, 2016.
[18] D. Baum *et al.*, "Revealing hidden text in rolled and folded papyri," *Appl. Phys. A, Solids Surf.*, vol. 123, p. 171, Mar. 2017.
[19] J. Weickert, "Coherence-enhancing diffusion filtering," *Int. J. Comput. Vis.*, vol. 31, nos. 2–3, pp. 111–127, 1999.
[20] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.
[21] C. Liu, P. L. Rosin, Y.-K. Lai, and W. Hu, "Robust segmentation of historical parchment XMT images for virtual unrolling," in *Proc. Digit. Heritage*, Sep./Oct. 2015, pp. 11–18.
[22] C. Liu, P. L. Rosin, Y.-K. Lai, G. R. Davis, D. Mills, and C. Norton, "Recovering historical film footage by processing microtomographic images," in *Proc. Int. Conf. Digit. Heritage*, 2016, pp. 219–231.
[23] G. Elber, I.-K. Lee, and M.-S. Kim, "Comparing offset curve approximation methods," *IEEE Comput. Graph. Appl.*, vol. 17, no. 3, pp. 62–71, May 1997.
[24] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas*, 2nd ed. Princeton, NJ, USA: Princeton Univ. Press, 2009, ch. 2, p. 108.
[25] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," *ACM Comput. Surv.*, vol. 18, no. 1, pp. 23–38, 1986.
[26] V. Kolmogorov, "Blossom V: A new implementation of a minimum cost perfect matching algorithm," *Math. Program. Comput.*, vol. 1, no. 1, pp. 43–67, 2009.
[27] R. Duan and S. Pettie, "Linear-time approximation for maximum weight matching," *J. ACM*, vol. 61, no. 1, 2014, Art. no. 1.
[28] J. Edmonds, "Maximum matching and a polyhedron with 0, l-vertices," *J. Res. Nat. Bur. Standard B, Math. Math. Phys.*, vol. 69B, nos. 1–2, pp. 125–130, 1965.
[29] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.
[30] M. Meilă, "Comparing clusterings: An axiomatic view," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 577–584.
[31] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, May 2004.



**Chang Liu** was born in Bejing, China, 1987. He received the bachelor's degree in guidance, navigation, and control from the Nanjing University of Aeronautics and Astronautics, and the Ph.D. degree in guidance, navigation, and control from Beihang University, in 2009 and 2016, respectively. He holds a post-doctoral position in aeronautical and astronautical science and technology at Beihang University. His research interest is guidance and navigation of spacecraft based on computer vision.

**Paul L. Rosin** was with Brunel University, the Joint Research Centre, Italy, and the Curtin University of Technology, Australia. He is currently a Professor with the School of Computer Science and Informatics, Cardiff University. His research interests include the representation, segmentation, and grouping of curves, knowledge-based vision systems, early image representations, low level image processing, machine vision approaches to remote sensing, methods for evaluation of approximations, algorithms, medical and biological image analysis, mesh processing, and the analysis of shape in art and architecture.

**Yu-Kun Lai** received the bachelor's and Ph.D. degrees in computer science from Tsinghua University, China, in 2003 and 2008, respectively. He is currently a Senior Lecturer with the School of Computer Science and Informatics, Cardiff University. His research interests include computer graphics, geometry processing, computer-aided geometric design, computer vision, and image processing. He is on the Editorial Boards of *The Visual Computer*.

**Weiduo Hu** was born in Heihe, China, 1965. He received the bachelor's, master's, and Ph.D. degrees from Beihang University, in 1986, 1989, and 1993, respectively, the Ph.D. degree from the University of Michigan in 2002. He was an Engineer with the Beijing Institute of Control Engineering. He is currently a Professor with Beihang University, China. His major research interests are navigation, control and dynamics, especially for spacecraft.