Learning to Reassemble Shredded Documents

Fabian Richter, Christian X. Ries, Nicolas Cebron, and Rainer Lienhart

***Abstract*—In this paper, we address the problem of automatically assembling shredded documents. We propose a two-step algorithmic framework. First, we digitize each fragment of a given document and extract shape- and content-based local features. Based on these multimodal features, we identify pairs of corresponding points on all pairs of fragments using an SVM classifier. Each pair is considered a point of attachment for aligning the respective fragments. In order to restore the layout of the document, we create a document graph in which nodes represent fragments and edges correspond to alignments. We assign weights to the edges by evaluating the alignments using a set of inter-fragment constraints which take into account shape- and content-based information. Finally, we use an iterative algorithm that chooses the edge having the highest weight during each iteration. However, since selecting edges corresponds to combining groups of fragments and thus provides new evidence, we reevaluate the edge weights after each iteration. We quantitatively evaluate the effectiveness of our approach by conducting experiments on a novel dataset. It comprises a total of 120 pages taken from two magazines which have been shredded and annotated manually. We thus provide the means for a quantitative evaluation of assembly algorithms which, to the best of our knowledge, has not been done before.**

***Index Terms*—Annotated dataset, document assembly, graph algorithm, supervised learning.**

I. Introduction

In this work we propose an algorithmic framework for the automatic assembly of shredded documents. The problem of having to reconstruct shredded documents is often faced by historians and forensic investigators. For example, when the Historical Archive of the City of Cologne was destroyed in 2009 as the ground beneath it collapsed, a large amount of valuable historical documents got severely damaged. Still, some of those documents might be reconstructable.

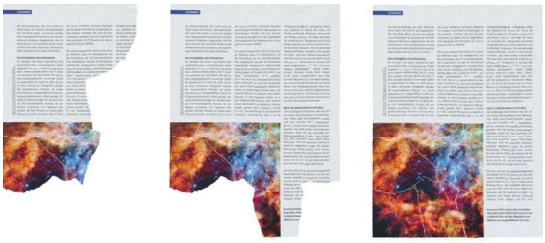
Another example is the currently ongoing work on reassembling documents related to the Stasi which was the secret police of the GDR. Shortly before the end of the Socialist regime of the GDR in 1989 members of the Stasi destroyed millions of documents containing evidence about their activities. Many of those files were simply shredded by hand and thus can be reassembled.

Manuscript received September 26, 2011; revised February 23, 2012; accepted August 23, 2012. Date of publication December 19, 2012; date of current version March 13, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Xian-Sheng Hua.

The authors are with the Multimedia Computing and Computer Vision Lab, University of Augsburg, D-86159 Augsburg, Germany (e-mail: richter@informatik.uni-augsburg.de; ries@informatik.uni-augsburg.de; cebron@informatik.uni-augsburg.de; lienhart@informatik.uni-augsburg.de).

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/TMM.2012.2235415



(a) (b) (c)

Fig. 1. Examples for intermediate results of the assembly algorithm after 6 (left) and 12 (center) iterations, respectively. The document is entirely reconstructed (right) after 15 iterations. The page depicted here has been manually shredded into 16 pieces.

However, reconstructing such documents manually is a difficult and tedious task due to the large number of possible permutations of fragment arrangements. Therefore, a system capable of assembling shredded documents automatically is of great use. The major challenge introduced by the reassembling task, however, also applies to automatic systems—the number of potential fragment arrangements is huge.

In theory we have to exhaustively search among all possible fragment permutations which renders a naive reassembly intractable. Besides, in contrast to the closely related problems of solving traditional jigsaw puzzles or edge-matching puzzles, manually shredded document fragments may assume almost arbitrary shapes. At first glance this might seem to facilitate the task by introducing additional edge shape information. However, it also significantly increases the number of possible fragment alignments and rotational fragment angles. It is also not straightforward to exploit edge shape information.

Thus, one of the main challenges we focus on is the efficient reduction of the number of potential fragment matches while preserving a sufficient number of correct matches. Another important issue that must be addressed is devising an algorithm capable of reliably aligning matching groups of fragments.

We tackle these problems by devising a framework consisting of two stages: We first use a SVM classifier to identify locations on fragment pairs which might belong together using both, shape- and content-based local features. In the second stage we iteratively align groups of fragments until the document has been restored. Thereby we construct a spanning tree which is less complex than an exhaustive search. Figs. 1 and 2 give examples of intermediate results of our iterative algorithm.

We evaluate our system on a novel, fully annotated dataset which is publicly available for download on our website.[1] We also introduce a geometric quality measure for reassembled pages. This provides the means to evaluate how much an assembly result deviates from the intact document.

[1]http://www.multimedia-computing.de

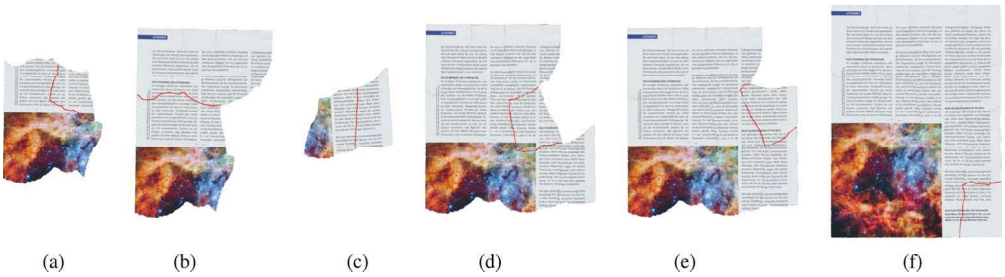


Fig. 2. Examples for aligned groups of fragments. (a)-(e) show intermediate results after iteration t. (f) depicts the entirely reassembled document. The curves (highlighted red) are contour intersections which form the coincident border between groups of fragments. This information is utilized for alignment as detailed in Section V-B. (a) $t = 3$ (b) $t = 6$ (c) $t = 7$ (d) $t = 9$ (e) $t = 10$ (f) $t = 15$.

Our paper is organized as follows: In Section II we give a brief overview of related work. In Section III we explain how we digitize our dataset of manually shredded documents. We also describe how we determine support points of fragments which are used as points of attachment. Section IV deals with our pre-selection step for support points. This is an important part of our approach since it makes our assembly algorithm feasible. In Section V we explain our main algorithm which iteratively reassembles shredded documents. Here we provide a detailed description of the algorithm and explain the constraints which are used to determine pairs of fragments that fit together well. Afterwards, we describe our dataset and how it has been annotated in Section VI. Finally, Section VII presents experiments and a thorough quantitative evaluation of our approach. Section VIII concludes the paper.

II. Related Work

As mentioned in the introduction, the problem of reassembling shredded documents is closely related to the problem of automatically solving jigsaw puzzles. As early as 1964, Freeman *et al.* [1] for instance defined shape-related features for matching pieces of jigsaw puzzles. In 1982 Radack *et al.* [2] described the borders of puzzle pieces using a log-polar representation in order to efficiently match them.

There are also various approaches that apply the paradigms of jigsaw puzzle solving to real-world applications. For instance, Sagiroglu *et al.* [3] propose a method for reassembling broken objects. They expand the boundaries of fragments using inpainting and texture synthesis. The problem of reconstructing wall paintings is also closely related to the reassembly of jigsaw puzzles. Papaodysseus *et al.* [4] approach this problem by using shape-based contour matching. Zhu *et al.* [5] present an approach for the reconstruction of ripped-up manuscripts based on partial curve matching. Their method iteratively disambiguates candidate matches by maximizing a global compatibility criterion. Based on a partial curve matching strategy, Cao *et al.* [6] use inter-fragment constraints to identify possible matches between pieces from multiple photos. Therefore, they augment their geometric shape features by color information. In order to assemble the photos, a graph-based approach is proposed which first identifies subgraphs corresponding to separate photos and then searches for a spanning tree of each subgraph.

In our previous work [7] we adapt the preprocessing strategy of Cao *et al.* for fragments of shredded documents. We also iteratively determine a spanning tree, however, we consider information obtained by evaluating shape- and content-based constraints on partial solutions. For more examples of document and artifact reconstruction methods refer to the survey of Kleber *et al.* [8].

Since the above mentioned approaches are designed for real-world applications, they deal with randomly shaped pieces as created by natural processes. For this reason, the assembly problem is two-fold: On the one hand fragments have to be aligned properly onto each other, and on the other hand, finding a globally consistent solution poses a combinatorial problem. However, by considering more regularly shaped fragments the inherent challenges are shifted towards the combinatorial optimization problem.

For instance, Chung *et al.* [9] map the corners of each puzzle piece into a unit square to describe the shape. Furthermore, all pieces are categorized into frame pieces and interior pieces. They interpret the problem of positioning the frame pieces as a traveling salesman problem, which is solved heuristically. Afterwards, the inner pieces are placed by finding the globally best permutation. Goldberg *et al.* [10] address this problem in a similar manner. However, they also apply an optimization step to refine the embedding once a new piece is placed.

Cho *et al.* [11] focus on the general case of rectangular image patches. They use a probabilistic model and a belief propagation algorithm to determine the optimal patch configuration. Since there is no shape information available, they use a similarity measure based on the pixels along the respective patch borders. They also evaluate various compatibility metrics.

For a discussion on the combinatorial aspects of jigsaw puzzles and other types of puzzles we refer to the work of Demain *et al.* [12], who show that finding a solution in general is NP-complete.

III. Preprocessing

Our dataset, which is described in detail in Section VI, consists of 120 pages taken from two scientific magazines. All pages have been manually shredded into 16 pieces of different sizes and shapes. A subset of 48 pages has been further divided into 24 and 32 pieces, respectively. As in our previous work [7], each piece has been scanned, front and back, against a uniformly colored background.

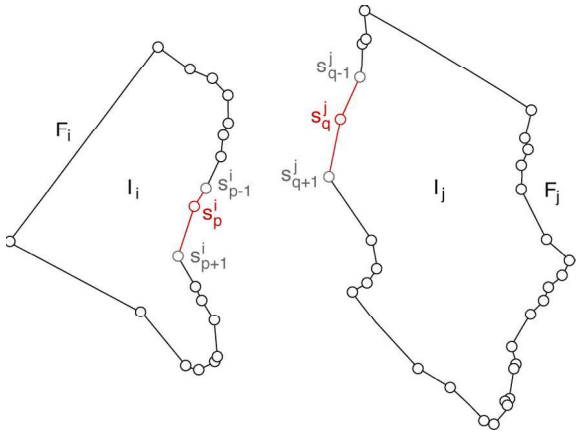


Fig. 3. Line segments constituting the approximate contour of two fragments F_i and F_j. Note that each support point is associated with exactly two line segments.

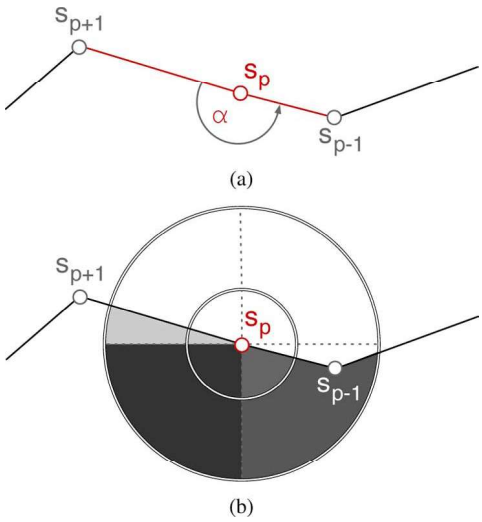


Fig. 4. (a) Angle and line segments features. (b) Shape feature with 2 radial and 4 angular bins. Bin weights are assigned proportionally to the number of fragment pixels intersecting with the image region covered by the respective bin. A dark shading indicates a high number of foreground pixels in the respective bin.

Afterwards we subtract the background from the resulting image. Therefore, we use a flood fill algorithm to distinguish background from foreground pixels. The implementation of this algorithm uses an edge map of the raw scanned image to avoid cutting into the fragments. We use a standard implementation of the Canny edge detector [13] for creating the edge map. The preliminary segmentation results are refined along the fragment borders by morphological openings. Eventually, the entire process yields a binary foreground-background segmentation mask.

We finally apply the algorithm of Suzuki *et al.* [14] to this segmentation mask to determine the contour of each fragment. As the exact contour of the i-th fragment of each page is given by a set of pixels P_i that tends to be large in practice, we use the Douglas-Peucker algorithm [15] to determine a subset of *support points* $S_i = \{s_1^i, s_2^i, \ldots, s_{n_i}^i\} \subset P_i$ that allows an accurate yet less complex description of each fragment. By connecting each consecutive pair of support points we obtain a *contour approximation* that makes our approach feasible while maintaining fair performance.

We define fragments by $F_i = (S_i, I_i)$, where I_i refers to the image content and S_i denotes the set of support points as illustrated in Fig. 3. For each support point we also extract features as described in detail in Section IV.

IV. Support Point Selection

In regular puzzles pieces often have tabs and corresponding blanks cut into their respective sides. However, instead of puzzle pieces our dataset consists of fragments without interlocking pairs of tabs and blanks. In fact, each support point on an approximate contour can be considered a candidate for the alignment with another support point of a second fragment. A naive selection strategy would make our approach prohibitively expensive in practice. Thus we restrict the pairs of support points to a small subset of points that are likely to fit together. To accomplish this, each support point is first described by 5 complementary local features. These features model the shape and the content of the respective fragment in the vicinity of the support point.

For each pair of support points we use the feature descriptors to compute feature dissimilarities, individually for each of the 5 feature channels. By concatenating the resulting dissimilarity values we form a descriptor of the respective support point pair. Finally, we train a SVM on these dissimilarity vectors, using true and false correspondences from our annotated dataset.

A. Features

In order to identify true correspondences between fragments, we extract a set of features for each support point. Despite the fact that local features, such as SIFT [16], are commonly used to describe image patches, they are not appropriate in our scenario as fragments do not share meaningful content overlap.

Instead, we extract 5 features around each support point s of each fragment. These features describe the shape and the content of a small foreground region. For each feature channel k we obtain a descriptor $\boldsymbol{x}_k^s \in \mathbb{R}^{d_k}$.

Note that our approach requires all features to be invariant towards rotation since we do not know the true orientation of our document fragments.

Line Segments Feature. Our first feature is the absolute length of the line segments associated with a support point. As illustrated in Fig. 3, support points of fragments are traversed in either clockwise- or counterclockwise direction. We thus know which line segments from different support points are to be associated with each other. Intuitively, two support points match if their associated line segments are very similar in length.

Angle Feature. Our second geometrical feature is the angle α which lies on the inside of the polygon. As depicted in Fig. 4(a), it is enclosed by the two line segments adjacent to the support point. Except for noise, the angles of two matching support points are conjugate angles and thus sum to 2π. In general, the deviation from this ideal angle sum is large for non-matching support points.

Shape Feature. Since the line segments and angle take into account only a narrow region around each support point we propose a contextual descriptor that captures the spatial distribution of content. Motivated by the shape context feature [17] we create a log-polar-like coordinate system defined by the number of angular and radial bins. After identifying foreground and background pixels, we assign a weight to each bin. Each weight is simply given by the relative number of foreground pixels intersecting with the image region covered by the respective bin. As illustrated in Fig. 4(a), bins which almost exclusively cover foreground are assigned large weights (indicated by dark gray), whereas sparsely populated bins obtain small weights.

We need to ensure that our feature is invariant towards rotation. Thus we rectify each shape feature with regards to a distinctive bin. We choose the bin which covers the boundary from background to foreground of the fragment. Note that we have to search for the distinct bin in different directions (clockwise or counterclockwise) for support points on opposing fragments.

Color Histogram. We use color histograms in CIE Lab color space that are extracted over circular regions around each support point. Color histograms yield a rotation-invariant description of the content around the respective support point. However, they also omit the spatial color layout.

Color Layers Feature. Since regular histograms do not encode any spatial information, we group pixels nearby the outer contour of the fragment into multiple color layers. As detailed in Section III, we use the algorithm of Suzuki *et al.* [14] on the foreground-background segmentation mask to find the fragment contour. All pixels on this contour within a maximum distance to the support point constitute the outmost color layer. Repeating this process on a modified binary mask in which all previously identified contour pixels have been removed yields contours that are located further on the inside of the fragment. Each content layer is represented by a fixed-size vector which contains pixel values in CIE Lab color space. The color values are ordered canonically in clockwise- and counterclockwise direction for features to be matched. Finally we concatenate all layer vectors into one single feature descriptor.

All of our features depend on various parameters which influence their discriminativeness. Thus we experimentally determine the best configuration for each feature as detailed in Section VII-A.

B. SVM on Feature Dissimilarities

We train a SVM to efficiently distinguish matching pairs of support points of adjacent fragments from false matches.

Given the set of features described in the previous section we determine the dissimilarity between pairs of support points for each feature channel. By design, the pairwise distance between two feature descriptors of non-matching support points tends to be large in practice. Accordingly, it is small for support points which belong together.

It is worth mentioning that in some cases the vicinity around support points provides insufficient evidence to disambiguate false from true matches. Thus, the distance between feature descriptors is a necessary but no sufficient condition.

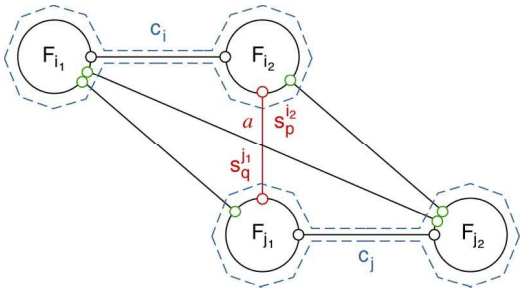


Fig. 5. In this example, clusters c_i and c_j contain two fragments each. They are combined by aligning fragment F_{i_2} to F_{j_1} using support points $(s_p^{i_2}, s_q^{j_1})$. The resulting transformation is then applied to all remaining fragments, i.e., F_{i_1}.

Throughout the rest of this paper we refer to matching pairs as *true correspondences* and non-matching pairs as *false correspondences*. For each feature k we compute the dissimilarity d_k between two support points s_p^i and s_q^j of fragment pair (F_i, F_j) by

$$d_k(s_p^i, s_q^j) = \eta^{-1} \left\| \boldsymbol{x}_k^{s_p^i} - \boldsymbol{x}_k^{s_q^j} \right\|_1 , \tag{1}$$

where η is the value of the median of all pairwise distances in feature channel k of fragments F_i and F_j. Finally, we concatenate the five resulting dissimilarities over all features in one dissimilarity vector $\boldsymbol{d}^{s_p^i, s_q^j} = [d_k(s_p^i, s_q^j)]_{k=1\ldots5} \in \mathbb{R}^5$.

Our dataset, as described in Section VI, contains annotations for all pages and provides examples of true and false correspondences, which are used as positive and negative examples for training. Since there are significantly more negative than positive examples we randomly sample negative examples to maintain a balanced training set.

We use LIBSVM [18] to train a binary SVM using a RBF kernel. We perform a coarse-to-fine grid-search over the (C, γ) parameter space and use 5-fold cross-validation. It should be emphasized that the quality of our SVM classifier depends on the feature parameters which are evaluated in Section VII-A.

V. Approach

In this section we provide a thorough explanation of our main algorithm and its components. We first explain our concept of fragment alignment by providing an illustrating example in Section V-A. Then we describe the constraints used to determine whether two fragments or groups of fragments fit together. We explain how we evaluate these constraints and how we translate them into an alignment score. Finally, we give a formal step-by-step explanation of our assembly algorithm in Section V-C.

A. Fragment Alignment

Assuming no prior knowledge about fragments one can align two fragments based on any pair of support points. In practice, however, a naive approach that considers each pair of support points is prohibitively expensive, since the number of possible combinations grows quadratically in the number of support points of the respective fragments. As described in the previous

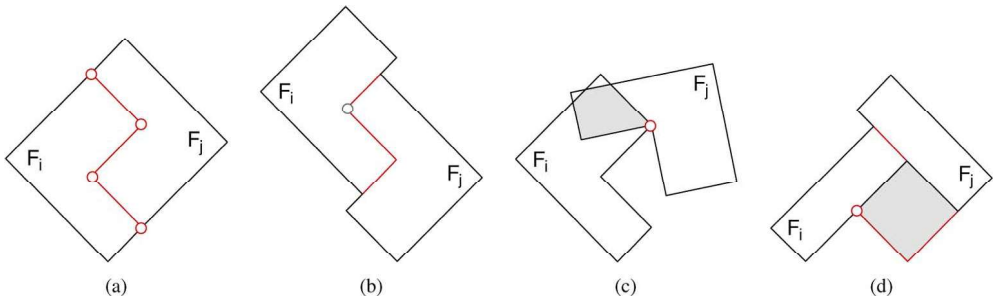


Fig. 6. Schematic representation of four distinct alignments between two fragments F_i and F_j. The red lines illustrate the coincident border between F_i and F_j. Red circles symbolize true positive correspondences and gray circles indicate false positive correspondences (both contribute to the consistency). Gray regions visualize the intersection between fragments. Examples show (a) long coincident border together with high consistency and no intersection, (b) low consistency, (c) low consistency with intersection, and (d) low consistency with long coincident border and intersection.

section we use a SVM to discard most of the false correspondences *a priori* to the actual algorithm.

To exemplify the alignment between a single pair of fragments, assume that fragment F_i is to be aligned onto F_j according to a pair of support points $(s_p^i, s_q^j) \in S_i \times S_j$. First, we translate all support points S_i of fragment F_i by an offset vector $\boldsymbol{v}$ such that s_p^i coincides with s_q^j. Second, we rotate all support points S_i by angle θ enclosed by the pair of line segments given by (s_p^i, s_{p+1}^i) and (s_q^j, s_{q+1}^j) respectively (see Fig. 3). Finally, the same translation and rotation is applied to the image content I_i.

However, as our algorithm iteratively creates clusters of aligned fragments, we need a strategy to align entire groups of fragments. As illustrated in Fig. 5, an $alignment\ a = (\boldsymbol{v}, \theta)$ between groups of fragments c_i and c_j is defined by an affine transformation between two fragments and a single corresponding pair of support points. Without loss of generality, assume that $(s_p^{i_2}, s_q^{j_1})$ from their respective fragments $(F_{i_2}, F_{j_1}) \in c_i \times c_j$ are chosen for the alignment. In order to align both clusters we first compute the affine transformation that locally aligns F_{i_2} to F_{j_1}. Afterwards, this alignment is applied to each fragment within c_i.

B. Constraints on Alignments

In this section, we introduce an alignment score which reflects our confidence in the correctness of a given alignment. Our score is based on a set of geometric and content-based constraints. These constraints are evaluated for each alignment a between any pair of clusters.

Our first constraint is the *coincident border* between two groups of fragments, which is illustrated in Fig. 2 as red curves. It is defined as the number of intersecting contour pixels between the fragments that are to be aligned. Second, to model the fact that properly aligned fragments do not have any content overlap, we also determine the *absolute intersection* between them.

Since fragments often vary in size, we also take into account the *relative intersection*, which is computed by normalizing the absolute intersection with respect to the minimal area of both fragment groups. Finally, we use the SVM introduced in Section IV-B to classify each pair of support points along the coincident border. The number of all pairs that are classified as positive is referred to as the *consistency* of the given alignment.

For instance, consider the four situations shown in Fig. 6. Note that aligning the two fragments F_i and F_j as depicted in 6a complies with all constraints optimally because of a long coincident border and a high consistency as well as no intersection. The remaining figures give examples for violated constraints. First, in situations where shape information is present, geometric constraints often provide sufficient evidence for aligning fragments correctly. However, in cases where shape information is insufficient, one also requires content-based information to resolve such ambiguities. For example the fragments in Fig. 6(b) seemingly match with regard to their coincident border and their lack of intersection. This alignment, however, conflicts with the last constraint due to a low consistency along the coincident border. Second, the examples in Fig. 6(c) and (d) both violate the intersection constraint.

As illustrated by these examples, an alignment is correct only if it fulfills all constraints at the same time. Therefore, to each alignment a we assign scores ζ_k^a, $k \in \{1, \ldots, K\}$, for each of the $K = 4$ above mentioned constraints. We normalize the scores to $[0, 1]$, individually for each constraint. Finally, we compute the *alignment score* of an alignment a as

$$\psi(a) = \sum_{k=1}^{K} \zeta_k^a. \tag{2}$$

Since on the one hand, all individual constraint scores are restricted to $[0, 1]$ and on the other hand, all summands are weighted equally, large values for $\psi(a)$ can only be obtained if the alignment complies with all constraints at the same time.

It should be emphasized that in practice the angle θ that is computed for each alignment is only approximately correct as it is inferred from the contour approximation of fragments. Therefore, we use a heuristic that allows for small variations of the alignment angle. To make this computationally feasible in practice we apply a coarse-to-fine search in a small interval around θ. We then choose the locally optimal solution that yields the highest alignment score. Note that this local optimum corresponds to a global optimum in most cases since θ usually is a good estimation for the correct alignment angle.

The alignment score can be used to determine the best alignment $\hat{a}_{ij}$ between two groups of fragments (c_i, c_j) by

$$\hat{a}_{ij} = \underset{a \in \delta(i,j)}{\arg\max} \psi(a), \tag{3}$$

where $\delta(i,j)$ is the set of alignments between the two clusters. Since alignments are based on pairs of support points we can use our SVM to discard unpromising candidates in advance. As a consequence, we significantly reduce the computational complexity of finding the optimal alignment.

It is now straightforward to choose the overall best alignment between any pair of clusters by

$$\hat{a} = \arg\max_{i<j} \hat{a}_{ij}. \tag{4}$$

Algorithm

Given the fragments of a shredded document that are positioned and oriented randomly, our algorithm is capable of repositioning all fragments to recover the layout of the intact document. We aim to determine a sequence of translations and rotations for each fragment that yield the best assembly result with respect to the aforementioned constraints.

As in our previous work [7] we create a *document graph* $\mathcal{G} = (V, E)$ in which each fragment corresponds to a vertex and edges between them are associated with alignments. Since each pair of support points provides one alignment we obtain a large number of edges between each pair of vertices. However, we do not have to consider the subset of alignments that are rejected by our SVM. For all remaining alignments we weight their edges with respect to the alignment score. Also, edge weights are updated after each iteration. As a consequence, alignments that were seemingly correct in prior iterations due to insufficient contextual evidence may later cause inconsistencies and vice versa.

By iteratively choosing the best alignment to combine clusters within each iteration, we greedily determine a spanning tree of $\mathcal{G}$ as in Kruskal's algorithm [19]. Provided as input the positive correspondences between each pair of fragments, the algorithm returns an alignment $\hat{Q}_i$ for each fragment F_i as well as a spanning tree represented by edge set $\hat{E}$. Applying each alignment to its respective fragment yields the assembly of the document. The spanning tree is required to evaluate the quality of the assembly result as discussed in Section VII-B.

In the following we describe each step of our proposed algorithm in detail.

Algorithm 1

Step 1: Initialization. Initially at time $t = 0$, each fragment F_i is considered a cluster by itself, i.e., $c_i = \{F_i\}$ and $C^{(0)} = \{c_0, c_1, \ldots, c_{|V|}\}$. Each edge of the document graph is weighted according to its alignment score (see (2)) and each Q_i is initialized as an empty sequence of alignments. Also, $\hat{E}$ is an empty set of edges.

Step 2: Combining Clusters. Let t be the current iteration. In order to find the best alignment $\hat{a}$ between any pair of clusters, we simply determine the edge with the largest weight that connects two distinct clusters. Let e be this edge and, without loss of generality, let it connect clusters c_i and c_j, $i < j$. We add e to the spanning tree:

$$\hat{E} = \hat{E} \cup \{e\}.$$

We combine both clusters into $\hat{c}_i = c_i \cup c_j$ by aligning their fragments according to $\hat{a}$ as described in Section V-A. As only fragments of cluster c_i are aligned in this step, we append $\hat{a}$ to their sequence of alignments, i.e.,

$$\forall\, k\ F_k \in c_i : \text{append } \hat{a} \text{ to } Q_k.$$

We obtain the set of clusters for iteration t by replacing the clusters that were combined, i.e.,

$$C^{(t)} = \left\{ \frac{C^{(t-1)}}{\{c_i \cup c_j\}} \right\} \cup \hat{c}_i.$$

Since we reduce the number of clusters by one during each iteration, the algorithm terminates after iteration $t = |V| - 1$. Fig. 2 shows some intermediate results.

Step 3: Removing Unpromising Alignments. To speed up the computation during the remaining iterations, we aim to reduce the number of alignments to be considered. For this purpose we apply two heuristics that remove any pair of support points from $\hat{c}_i$ that became obsolete due to alignment $\hat{a}$. First, we disable support points along coincident border of the new cluster (illustrated as red curves in Fig. 2). Intuitively, aligning another fragment to one of these points naturally produces high intersections. Second, we remove all pairs of support points which are evidently incorrect due to their clearly insufficient alignment score.

Step 4: Updating Edges. As mentioned before, combining two clusters provides additional evidence about the document at hand. Therefore the weight of all edges originating from the combined cluster $\hat{c}_i$ need to be updated as described in Section V-B.

Step 5: Return. After $|V| - 1$ iterations we accumulate the sequence of affine transformations for each fragment F_i into $\hat{Q}_i$.

VI. Dataset

Our dataset comprises sheets taken from two magazines. The former is a scientific journal (*bdw082010*)[2], consisting of 96 pages. As the examples depicted in Figs. 7 and 8 suggest, its pages show a wide variety of content including text, illustrations, and layout elements such as tables and diagrams. The second magazine is an information brochure (*booklet*) consisting of 24 pages, and its pages differ from those of *bdw082010*, both in terms of content and also page format.

All pages have been manually shredded into 16 pieces. A subset of 24 pages from each magazine has been further divided into 24 and 32 pieces, respectively. Thus, in total 48 pages are available in three different degrees of fragmentation. The number of pieces per page is indicative for the theoretical and computational complexity. Therefore, different numbers of pieces present different levels of complexity on which we evalute our approach.

[2]*Bild der Wissenschaft*, 08/2010.

Fig. 7. Examples for pages in the *bdw082010* dataset.

Fig. 8. Examples for pages in the *booklet* dataset.

A. Partitions

We aim to show that our approach performs well in the following two scenarios. First we show that, despite using only 16 pieces per page during training, our approach generalizes well to a higher number of pieces. Second, we show that it is capable of reassembling documents for which no training data is readily available.

Therefore, we split the 96 pages of *bdw082010* into three sets $\{\mathrm{train}\}$, $\{\mathrm{val}\}$ and $\{\mathrm{test}\}$, consisting of 32, 16 and 48 pages respectively. We categorized each page into either featuring a picture, text or a combination of both. Finally we proportionally distributed the pages of all three categories among $\{\mathrm{train}\}$, $\{\mathrm{val}\}$ and $\{\mathrm{test}\}$ with respect to their sizes in order to make each partition representative for the whole magazine. Pages taken from *booklet* are used for testing only to ensure that our approach is not biased towards characteristics of *bdw082010*.

B. Annotation

Since we want to evaluate our approach quantitatively, we need a ground truth for the correct layout of the shredded documents. Note that it is sufficient to know all pairs of support points among adjacent fragments that constitute a true correspondence. However, it is not straightforward to manually annotate these pairs since all fragments were scanned at different orientations.

Therefore, we created an annotation tool which allowed a human user to manually reconstruct the shredded documents. The user had to correctly arrange the digitized fragments of a given document by translating and rotating them individually. After the user finished reassembling a document the annotation tool automatically determined true correspondences for all adjacent fragments.

More precisely, let (F_i, F_j) be a pair of fragments that are adjacent in the manually reconstructed document. Given these fragments we aim to determine the set of support points that are mutually closest to each other. That is, we create an annotated subset $\mathcal{A}_{ij} \subset S_i \times Sj$ that contains a pair of support points $(p, q) \in \mathcal{A}_{ij}$ if and only if

$$d(p,q) = \left\{ \min_{p',q' \in \{p\} \times S_j \cup S_i \times \{q\}} d(p',q') \right\} < d_{\max} \qquad (5)$$

holds, where d specifies the distance between two points in image coordinates and $d_{\max}$ is a constant for their maximum distance. The support points satisfying (5) constitute the set of true correspondences that induce a bijection between the respective fragments. Note that only approximately 0.2 percent of all pairs of support points in our dataset are annotated as true correspondences.

VII. Evaluation

In this section we first describe how we optimize the parameters of the local features representing support points. Afterwards we introduce a quality measure for quantitative evaluation of the assembly results. We then shortly address the issue of computational complexity. Finally, we conduct experiments that illustrate the trade-off between quality of assembly results and computational complexity.

A. Feature Parameter Learning

To find the optimal parameter configuration for the SVM introduced in Section IV-B, we experimentally evaluate different configurations. Since the dimensionalities of the descriptors depend on the parameters, we choose to concatenate the feature dissimilarities instead of concatenating descriptors directly. We

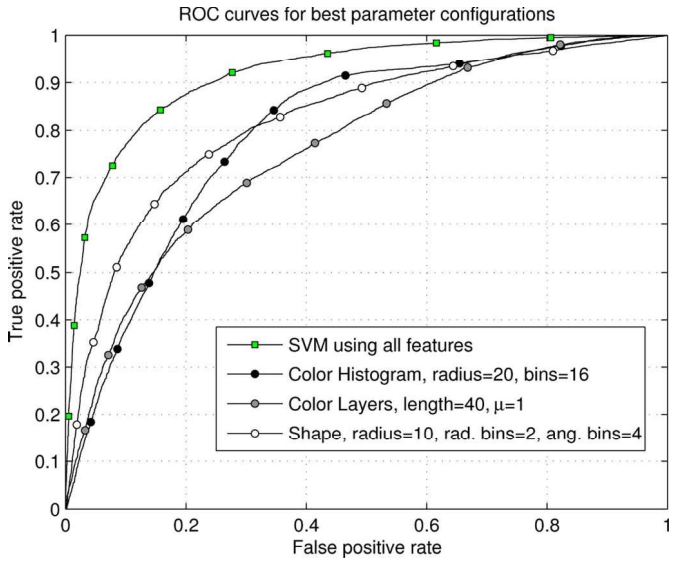


Fig. 9. The ROC curves for the best parameter configuration of each feature. The curve labeled with green square markers shows the performance of our SVM which uses a combination of all features. The AUC for the SVM using all features is 0.9184.

argue that otherwise, high-dimensional features would tend to outweigh low-dimensional features.

Furthermore, we assume to maximize the classification capability of our SVM by maximizing the discriminativeness within each feature channel separately. Our training data comprises pairwise dissimilarities computed on a subset of positive and negative examples from $\{\text{train}\}$. For each feature individually, we then train a SVM, separately for each combination of parameters, by performing grid search.

For the color histograms the radius of the circular area and the number of bins are chosen from $\{5, 10, 20, 40\}$ and $\{4, 8, 16, 32\}$, respectively.

Regarding the color layers feature we set the number of layers to 5 and use different lengths from $\{5, 10, 20, 40\}$ pixels. Also, we vary the weighting scheme which either assigns uniform weights to all layers or chooses the weights according to a normal distribution with $\mu \in \{1, 2, 3, 4, 5\}$ and $\sigma = 2$, where the weight for layer i is given by $\mathcal{N}(i; \mu, \sigma^2)$.

The parameters for the shape feature define the log-polar-like grid. We use an inner radius r, which is chosen from $\{6, 8, 10\}$. Based on a fixed expansion factor γ, empirically set to 1.5, we compute the size of the k-th radial bin as $\gamma^i r$. For the number of radial and angular bins we choose values from $\{2, 3, 4\}$ and $\{4, 8, 16\}$ respectively.

Based on the performance of all SVMs on $\{\text{val}\}$ we determine the ROC curves of all configurations, individually for each feature. We then choose the configuration which yields the largest AUC. In Fig. 9 we plot the results for the best parameter configuration of each feature. We also show the performance of our SVM using the final descriptor, defined in (1), which we obtain from concatenating all optimally configured feature dissimilarities.

Note that we also use the angle and line segments features for the final SVM. However, their respective ROC curves are omitted in Fig. 9 as they do not involve any parameters.

B. Adjustment Cost

Recall that our algorithm provides an accumulated alignment for each fragment, as well as a spanning tree of the document graph. Applying each alignment to its respective fragment yields the assembly result.

However, fragments that have been aligned accordingly may still deviate from their position in the ground truth. Thus we introduce a quantitative evaluation scheme that allows to rate the quality of each assembly result. Intuitively, our quality measure reflects the cost of transforming the assembly result to the correct layout defined by the ground truth.

Let (F_i, F_j) be a pair of fragments and let them be connected by an edge $e = (i, j)$ in the spanning tree. By applying alignment $\hat{Q}_i$ to F_i and $\hat{Q}_j$ to F_j, respectively, we position both fragments relative to each other. Since the optimal relative position for each pair of adjacent fragments is given by the ground truth, we can determine an adjustment that corrects the relative position of F_i with regard to F_j. For this purpose we determine a translational and a rotational component, represented by vector $\boldsymbol{v}_{ij}$ and angle $\vartheta_{ij} \in [0, \pi]$, respectively. Finally, we compute distinct costs—one for each component. The translational cost is defined as

$$\mathcal{C}_T(\boldsymbol{v}_{ij}) = 1 - \exp\left(-\eta \|\boldsymbol{v}_{ij}\|\right), \tag{6}$$

where η is some normalization constant that has been set empirically. The cost for the rotational component is defined by

$$\mathcal{C}_R(\vartheta_{ij}) = \pi^{-1} \vartheta_{ij}. \tag{7}$$

Intuitively, if two fragments are aligned as defined in the ground truth, both costs equal zero. For a given pair of adjustment parameters $(\boldsymbol{v}, \vartheta)$ we then define an *adjustment cost* by fusing both cost functions as follows:

$$\mathcal{C}(\boldsymbol{v}, \vartheta) = \mathcal{C}_T(\boldsymbol{v}) + (1 - \mathcal{C}_T(\boldsymbol{v})) \cdot \mathcal{C}_R(\vartheta). \tag{8}$$

Note that each edge in the spanning tree is associated with exactly one iteration t. Thus we use the equation above to assign a score in $[0, 1]$ to each of the intermediate results. The cost function defined in (8) reflects our intuition that, if the length of the offset vector is zero, only the rotation needs to be corrected and thus is the only parameter having impact on the overall cost. On the other hand, if the offset becomes larger, the rotation is a less significant factor to measure the effort for realigning the fragments.

We accumulate the costs over all edges in the spanning tree to determine an overall cost for correcting the layout of a reassembled document. Accordingly, we define the *accumulated cost* over spanning tree $\hat{E}$ by

$$\mathcal{C}_{span} = \sum_{(i,j) \in \hat{E}} \mathcal{C}(\boldsymbol{v}_{ij}, \vartheta_{ij}). \tag{9}$$

Assuming a page shredded into N pieces we thus obtain an accumulated cost $\mathcal{C}_{span} \in [0, N-1]$ for each reassembled page. Consequently, averaging $\mathcal{C}_{span}$ over all iterations yields a *mean accumulated cost* $\mu_{\mathcal{C}_{span}}$.

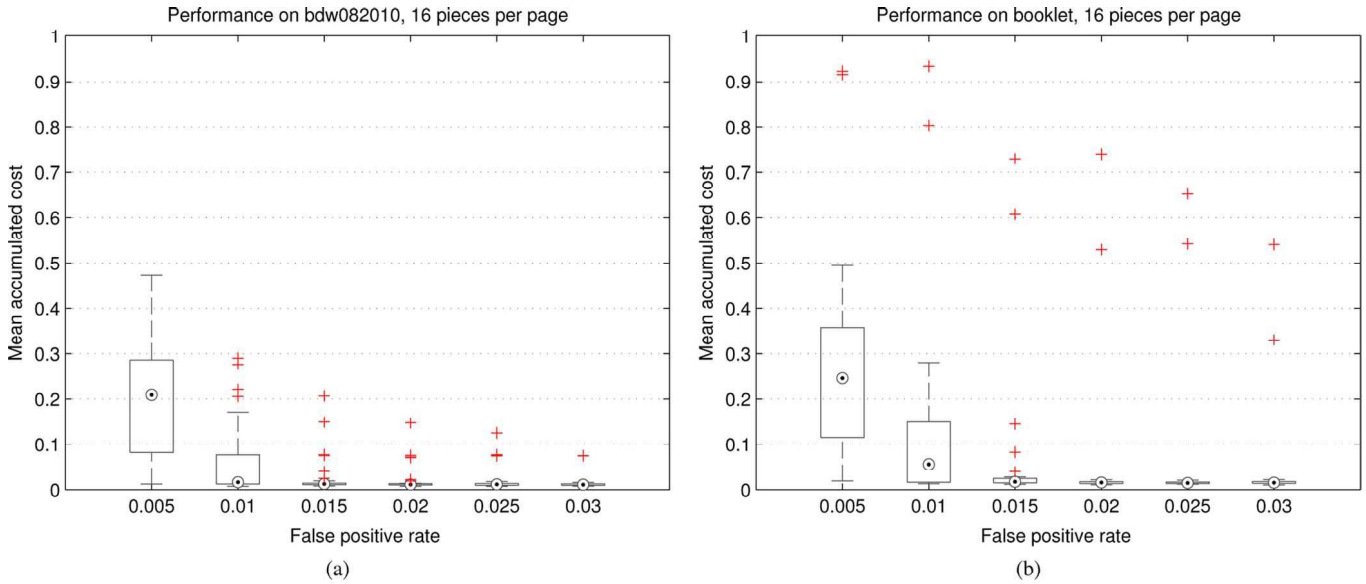


Fig. 10. Mean accumulated costs $\mu_{c_{span}}$ averaged over all pages of $\{\mathrm{test}\}$ on *bdw082010* (a) and *booklet* (b), for various SVM thresholds corresponding to different false positive rates. The circle is the median of costs, boxes show the 25% and 75% quantiles, respectively. Whiskers are indicated by dashed lines and red crosses represent outliers.

C. Computational Complexity

As mentioned before, the feasibility of our approach depends on the number of alignments. Using the SVM introduced in Section IV-B we control this number by choosing a threshold with respect to the distance from hyperplane. In the following we will refer to this threshold as *SVM threshold*. Note that the choice of this threshold is based on the ROC curve depicted in Fig. 9 and corresponds to selecting a fixed false positive and true positive rate on $\{\mathrm{val}\}$. Conversely, we can choose a false positive rate on $\{\mathrm{val}\}$ and determine the according SVM threshold.

Since the absolute number of true positives is significantly smaller than the number of false positives, the false positive rate of the SVM is the dominating factor regarding computational complexity of our approach. Intuitively, a more restrictive SVM threshold reduces the number of false positives (i.e., number of alignments) that need to be considered. Because limiting the number of alignments by this threshold also reduces the number of true positives, it controls the trade-off between computational complexity and quality of the result.

To keep our experiments feasible we empirically limit the false positive rate to $[0.005, 0.110]$. Note that the corresponding true positive rates range from 0.177 to 0.784. For unbalanced problems, however, relative frequencies tend to be misleading. It should thus be emphasized that for each fragment pair in our training set, we observe on average 5210 false correspondences. Using our SVM with a false positive rate as low as 0.005 reduces this number to less than 10 incorrect alignments. The following experiments display this trade-off between computational complexity and quality of the proposed method.

D. Experiments

The first experiment is conducted on $\{\mathrm{test}\}$ of both magazines. We choose $N = 16$ pieces for each page and assign the adjustment cost to the results as detailed in Section VII-B. In Fig. 10 the mean accumulated cost is plotted against the false positive rate of our SVM, separately for *bdw082010* and *booklet*. As can be seen, the mean accumulated cost, averaged over all pages, is below 0.1 for all false positive rates except the first one. Note that this false positive rate of 0.005 corresponds to merely 17.7% true positives. Also, a cost of 0.1 means that on average all fragments are positioned very precisely with respect to the ground truth. Furthermore, one can observe that despite slightly higher cost, we are able to reassemble almost all pages of *booklet*, except for a few extreme outliers. The outliers stem from two pages that are almost entirely colored uniformly and thus offer no content information. Fig. 13(a) and (c) show example results for *bdw082010* and *booklet*, respectively.

In our second experiment we increase the number of pieces to $N = 24$ and use representative subsets of 24 pages of both magazines. As Fig. 11 suggests, our approach is still capable of reassembling the pages correctly. In this setting, however, we require a higher number of true positives to maintain our desired level of quality. Still, the according false positive rates are as low as 0.11.

We suppose there are two main reasons for the increasing demand for true positives. First, fragmenting pages into a higher number of pieces naturally results in smaller fragments. Since we keep the accuracy for contour approximation fixed, there are less support points per fragment, and thus we need to increase the true positive rate to maintain the same absolute number of true positives. Second, increasing the number of pieces from 16 to 24 increases the complexity of the problem considerably. As the number of potential fragment pairs grows quadratically in N, we need more true positives to make our greedy selection strategy work.

Finally, we repeat the second experiment with $N = 32$ pieces and plot the result in Fig. 12. Even though the performance slightly declines, the overall result is still satisfactory given the inherent complexity of reassembling 32 pieces. We show results for the same pages as in the first experiment in Fig. 13(b) and (d).

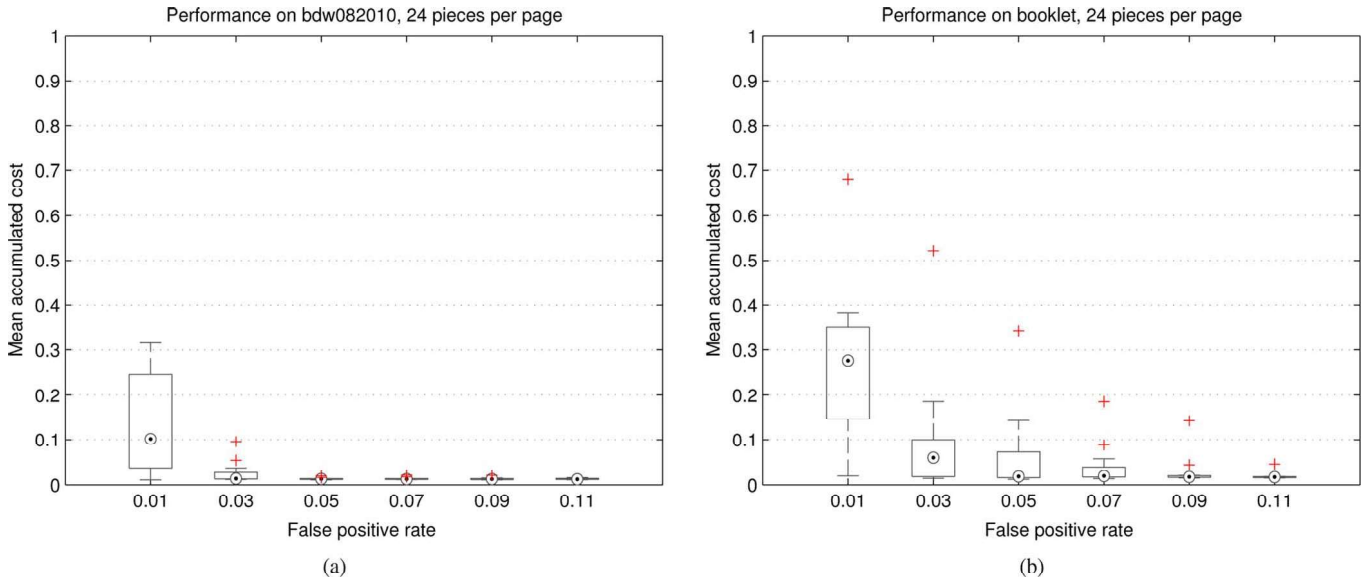


Fig. 11. Performance for 24 pieces per page, conducted on representative sets of 24 pages each, for *bdw082010* (a) and *booklet* (b), respectively.

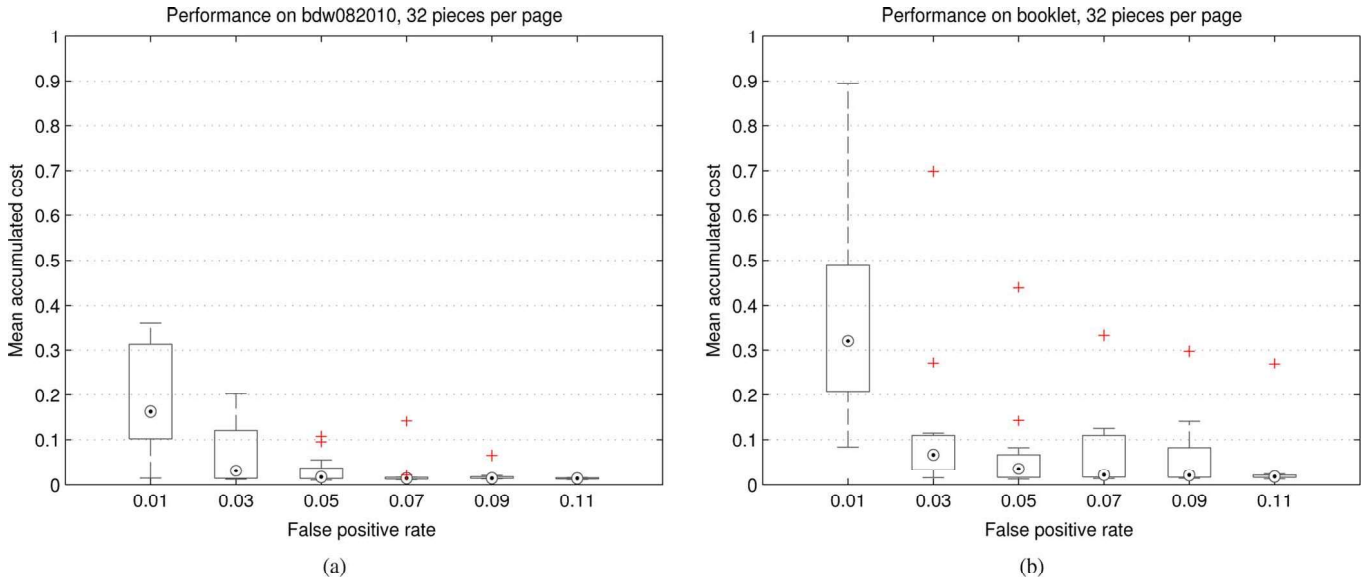


Fig. 12. Performance for 32 pieces per page, conducted on representative sets of 24 pages each, for *bdw082010* (a) and *booklet* (b), respectively.

In real world applications, however, one is often faced with the problem of missing pieces. For this reason we conduct a final experiment to evaluate the robustness of our algorithm in situations where fragments are lost. We choose the test set of *bdw082010* consisting of 32 pieces. For each page we discard a fixed number of randomly sampled fragments. As in previous experiments we determine the performance of our algorithm by averaging the mean accumulated cost over all pages and iterations. The results are plotted in Fig. 14.

We choose a false positive rate of 0.11. Therefore, the rightmost box in Fig. 12(a) is equal to the leftmost box in Fig. 14. Not surprisingly, one observes a decrease in performance when facing a higher number of missing pieces.

Most importantly, when removing too many pieces, the page at some point becomes disconnected and can not be reassembled entirely. In such cases one would need to stop the assembly process since no more correct alignments are available. If one would assume prior knowledge about the expected result, it would be possible to deduce straightforward stopping criteria, i.e., violation of the page format. However, such prior knowledge is usually not readily accessible in practice. In most scenarios we have only very limited information about properties of the intact document.

At first glance one might suppose that the alignment score also provides the means for such a stopping criterion. According to our observations, a high alignment score always corresponds to a correct decision. However, even if the globally optimal alignment score (chosen according to (4)) is low, it may still produce a correct alignment. Because of that, it is no sufficient condition for omitting the remaining iterations.

Fig. 15 depicts two example results for pages where 4 pieces are missing. It should be emphasized that due to the greedy selection strategy, intermediate results stemming from early iterations are predominantly correct. For instance, the wrongly

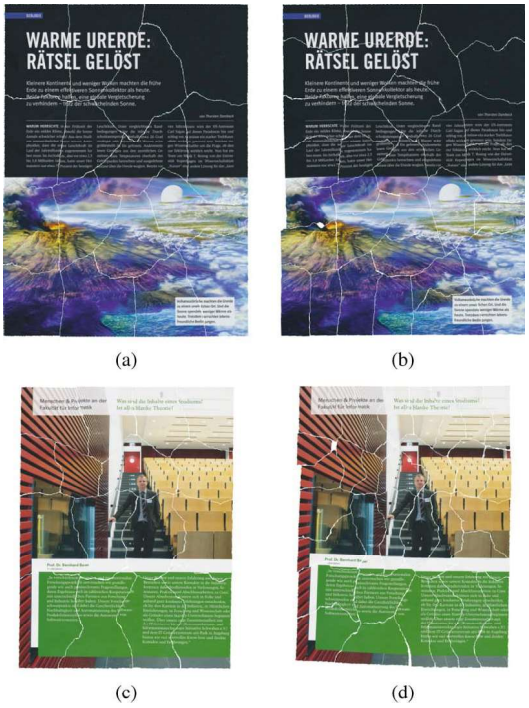


Fig. 13. Example results for one reassembled page from *bdw082010* shredded into 16 pieces (a) and 32 pieces (b). Analogous results for one page from *booklet* (c), (d).

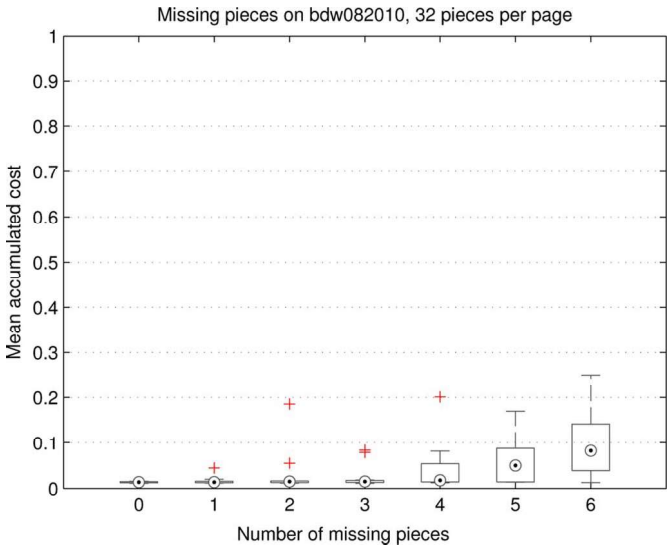


Fig. 14. Performance plotted against the number missing pieces per page. Results have been computed on the pages of *bdw08210* consiting of 32 pieces each.

placed piece in Fig. 15(b) has been positioned in the final iteration, and the intermediate results of all preceding iterations were entirely correct.

VIII. Conclusion

In this paper we proposed an algorithmic framework for the reassembly of shredded documents. In the first step of our framework we used a SVM to find pairs of support points between fragments which are suitable for aligning their respective fragments. The SVM enabled us to distinguish matching points of attachment from false matches based on feature dissimilarities which utilized shape- and content-based information.

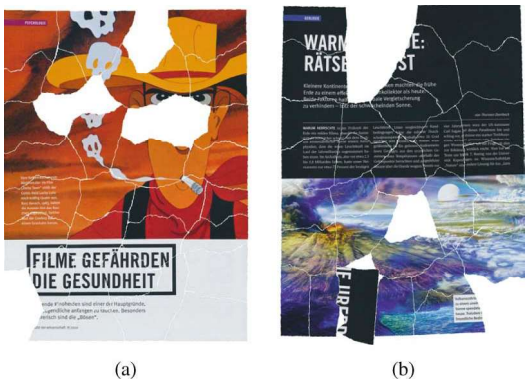


Fig. 15. Two example results for reassembled pages from *bdw082010*, featuring 32 pieces. In this experiment, 4 pieces have been randomly removed beforehand.

After identifying these points of attachment, we iteratively aligned all fragments into groups, which constitutes the second step of our framework. Therefore, we had to find the optimal alignment between all groups of fragments. For this purpose we introduced a set of geometric and content-based constraints which let us rate the quality of each alignment. By greedily selecting the best alignment in each iteration we combined groups of fragments until the document was entirely reassembled.

We quantitatively evaluated our approach on a novel annotated dataset which is publically available on our website. It consists of manually shredded pages from two magazines. For evaluation we created a ground truth and introduced a cost function that rates the quality of each assembly result.

We showed that our algorithm is capable of reassembling pages consisting of up to 32 pieces. It also yielded satisfactory results in the face of multiple missing pieces.

There are several interesting directions for future work. For instance we will consider a more general setting in which pieces of multiple sheets will be presented to the assembly process. This requires a more sophisticated strategy as in the most general scenario one does not know the exact number of pieces each sheet consists of. Furthermore, it will be necessary to develop a robust criterion to decide whether or not certain pieces belong to a document at hand. In particular, this would be helpful for rejecting noisy pieces. In this context we will also investigate how learning a non-uniform weighting scheme for the constraint weights may be beneficial for improving our results.

Acknowledgment

We thank the editorial staff of *Bild der Wissenschaft* and the publisher *Konradin Medien GmbH* for their permission to use the magazine and to make the dataset publicly available for research.

References

[1] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," *IEEE Trans. Electron. Comput.*, vol. 13, no. 2, pp. 118–127, 1964.

[2] G. M. Radack and N. I. Badler, "Jigsaw puzzle matching using a boundary-centered polar encoding," *Comput. Graphics Image Process.*, vol. 19, pp. 1–17, May 1982.

[3] M. Sagiroglu and A. Ercil, "A texture based matching approach for automated assembly of puzzles," in *Proc. 18th Int. Conf. Pattern Recogn.*, 2006, vol. 3, pp. 1036–1041.

[4] C. Papaodysseus, T. Panagopoulos, M. Exarhos, C. Triantafillou, D. Fragoulis, and C. Doumas, "Contour-shape based reconstruction of fragmented, 1600 bc wall paintings," *IEEE Trans. Signal Process.*, vol. 50, no. 6, pp. 1277–1288, Jun. 2002.
[5] L. Zhu, Z. Zhou, and D. Hu, "Globally consistent reconstruction of ripped-up documents," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 1–13, Jan. 2008.
[6] S. Cao, H. Liu, and S. Yan, "Automated assembly of shredded pieces from multiple photos," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2010, pp. 358–363.
[7] F. Richter, C. X. Ries, and R. Lienhart, "A graph algorithmic framework for the assembly of shredded documents," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2011, pp. 1–6.
[8] F. Kleber and R. Sablatnig, "A survey of techniques for document and archaeology artefact reconstruction," in *Proc. Int. Conf. Document Anal. Recogn.*, Los Alamitos, CA, USA, 2009, vol. 0, pp. 1061–1065.
[9] M. G. Chung, M. Fleck, and D. Forsyth, "Jigsaw puzzle solver using shape and color," in *Proc. 4th Int. Conf. Signal Process.*, 1998, vol. 2, pp. 877–880.
[10] D. Goldberg, C. Malon, and M. Bern, "A global approach to automatic solution of jigsaw puzzles," in *Proc. 18th Annu. ACM Symp. Computational Geometry*, 2002, pp. 82–87.
[11] T. S. Cho, S. Avidan, and W. T. Freeman, "A probabilistic image jigsaw puzzle solver," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2010, pp. 183–190.
[12] E. D. Demaine and M. L. Demaine, "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity," *Graphs Combinatorics*, vol. 23, pp. 195–208, Jun. 2007.
[13] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Jun. 1986.
[14] S. Suzuki and K. Abe, "Topological structural analysis of digital binary images by border following," *Comput. Vis., Graphics, Image Process.*, vol. 30, no. 1, pp. 32–46, 1985.
[15] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Can. Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.
[16] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Comput. Vis.*, 1999, pp. 1150–1157.
[17] S. Belongie and J. Malik, "Matching with shape contexts," in *Proc. IEEE Workshop Content-based Access of Image and Video Libraries*, 2000, pp. 20–26.
[18] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," 2001 [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[19] J. Kruskal, "On the shortest spanning subtree and the traveling salesman problem," *Proc. Amer. Math. Soc.*, vol. 7, no. 1, pp. 48–50, 1956.

Fabian Richter is currently working toward the Ph.D. degree at the Multimedia Computing and Computer Vision Lab, University of Augsburg, Augsburg, Germany. He received his diploma degree in Computer Science from the University of Augsburg, in November 2009. His research interests include Machine Learning, Image Analysis, and Image Processing.

Christian X. Ries is a Ph.D. student at the Multimedia Computing and Computer Vision Lab of the University of Augsburg. He acquired the Master degree in Computer Science from the University of Augsburg in 2009. His research interests are in the area of Computer Vision, Machine Learning, and Image Content Analysis.

Nicolas Cebron obtained his diploma in Computer Science from the Ostfalia University of Applied Sciences in Braunschweig, Germany and his Ph.D. degree in Computer Science from the University of Konstanz, Germany. He spent two years as a postdoctoral fellow in the EU research project 'Bisociation Networks for Creative Information Discovery' and at the International Computer Science Institute at the University of California, Berkeley. Dr. Cebron is currently working as a post-doctoral researcher in the field of active machine learning and image classification at the University of Augsburg, Germany.

Rainer Lienhart is a full professor in the computer science department of the University of Augsburg where he heads the Multimedia Computing and Computer Vision Lab. His group is focusing on all aspects of very large-scale image, video, and audio mining algorithms including feature extraction and image/video retrieval.

From August 1998 to July 2004 he was a Staff Researcher at Intel's Microprocessor Research Lab in Santa Clara, California, where he worked on transforming a network of heterogeneous, distributed computing platforms into an array of audio/video sensors and actuators capable of performing complex DSP tasks such as distributed beamforming, audio rendering, audio/visual tracking, and camera array processing. In particular, this requires putting distributed heterogeneous computing platforms with audio-visual sensors into a common time and space coordinate system. At the same time, he was also continuing his work on media mining, where he is well-known for his work in video content analysis with contributions in text detection/recognition, commercial detection, face detection, shot and scene detection, and automatic video abstraction.

He received his Ph.D. in Computer Science from the University of Mannheim, Germany, in 1998, where he was a member of the Movie Content Analysis Project (MoCA).

The scientific work of Prof. Lienhart covers more than 80 refereed publications and more than 20 patents. He was a general co-chair of ACM Multimedia 2007 and SPIE Storage and Retrieval of Media Databases 2004 & 2005. He serves in the editorial boards of 3 international journals. For more than a decade he is a committee member of ACM Multimedia, IEEE ICME, SPIE Storage and Retrieval of Media Databases, and many more conferences. Since July 2009 he is the vice chair of SIGMM. He is also the executive director of the Institute for Computer Science at the University of Augsburg since April 2010.