# Evaluation of Discriminative Models for the Reconstruction of Hand-Torn Documents

Fabian Richter, Christian X. Ries, Rainer Lienhart

Multimedia Computing and Computer Vision Lab, University of Augsburg

**Abstract.** This work deals with the reconstruction of hand-torn documents from pairs of aligned fragments. In the first step we use a recent approach to estimate hypotheses for aligning pieces from a set of magazine pages. We then train a structural support vector machine to determine the compatibility of previously aligned pieces along their adjacent contour regions. Based on the output of this discriminative model we induce a ranking among all pairs of pieces, as high compatibility scores often correlate with spatial configurations found in the original document. To evaluate our system's performance we provide a new baseline on a publicly available benchmark dataset in terms of mean average precision (mAP). With the (mean) average precision being widely recognized as de facto standard for evaluation of object detection and retrieval methods, our work is devoted to establish this performance measure for document reconstruction to enable a rigorous comparison of different methods.

## 1 Introduction

Efficient and robust matching of hand-torn document pieces is of great interest in many scientific disciplines, for instance in the fields of forensics, archaeology, and criminal investigation. Successfully aligned pieces can be used in many different scenarios, be it for guiding human users by suggesting partial solutions or for fully automatic reconstruction. For example, according to [1], the winning team of the 2011 DARPA Shredder Challenge partly relied on automatically generated piece-pair recommendations that had to be reviewed by team members for final decision making. While this challenge was focused around extracting information from shredded documents, there is also a famous example for investigators being faced with hand-torn documents: The majority of documents abandoned by the East German secret police in 1989 were simply destroyed by hand, and large efforts have been made to recover their valuable confidential content.

This paper specifically deals with the problem of reconstructing hand-torn documents as a two-stage procedure: In the first part (section 4) we align pairs of pieces based on their outer contours. To do so we extend a recent approach [2] that builds on MSAC [3] (M-estimator SAmple Consensus). We explain how to determine orientation estimates for pieces from the Fourier transform and show how to make use of this prior knowledge in computing alignments. In the second part (section 5) we then focus on the verification of contour regions that become adjacent through alignments. To validate the compatibility of these

regions, we compare their local visual characteristics in terms of shape, color, and texture. Our system therefore uses structured output prediction and dynamic programming to infer optimal sequences of matched contour points.

Our main contributions can be summarized as follows:

- Since our approach employs supervised learning, it provides a well-founded way to enrich geometric contour representations by content-based features. Unlike in unsupervised approaches, we can for example integrate color- and texture information without having to adjust thresholds manually.
- We provide two new baselines on a public dataset [4] in terms of overlap-recall curves for the a priori alignment step, and mean average precision (mAP) for the ranking of piece-pairs induced by our structural model. Using these standard performance measures enables an easily comprehensible evaluation and may facilitate comparison of different methods.

## 2 Related Work

A large number of recent work deals with the reconstruction of two-dimensional objects, for a wide range of applications, e.g., reassembling of jigsaw puzzles [5, 6], hand-torn documents and photos [4, 7–12], or archaeological findings [13, 14].

At the core of most approaches is the feature extraction from polygonal curves that approximate the pieces' outer contours. One popular approach used in many works, e.g., [8, 10, 11], is the turning function [15] proposed by Wolfson. It allows to represent curves by shape signature strings, which are invariant against rotation and translation. Because of that, this signature is well suited for substring matching techniques. For partial contour matching in particular, many approaches [6, 16, 13, 10] use variants of the Smith-Waterman algorithm [17] or similar dynamic programming methods. However, effectively matching contours often requires a priori corner detection [10], as this limits the procedure to subsegments between two consecutive points with high curvature. Since corners can be difficult to identify [13] in some settings, our approach relies on a different partial contour matching approach [2] based on MSAC [3].

To reassemble the intact document layout, pairs of aligned pieces are commonly ranked before being merged. In the approach of da Gama Leitão et al. [13], the authors compare curvature-encoded fragment outlines by dynamic programming. Since their approach uses progressively increasing scales of resolution, the overall computational complexity can be reduced. For each discrete pairing between two matched outline segments, the authors compute a discriminant value to discard likely incorrect candidates. A quite different route is taken by Zhu et al. [11], who aim to find a globally consistent solution to the reconstruction task. After identifying initial candidate matches based on the turning function, the authors disambiguate these candidates by considering the spatial compatibility of neighboring matches. To obtain a consistent solution, they use an iterative procedure that alternates between gradient projection and merging steps. Another approach closely related to our work is that of Stieber et al. [10]. The authors also apply the Smith-Waterman algorithm to align contour points. However,

their reliance on fixed costs for sequence operations seems to be a weak point. In contrast, our system employs supervised learning to obtain the cost model, which also allows to incorporate shape- and content-based local features seamlessly.

## 3 Dataset of Hand-Torn Documents

**Preprocessing.** In this work we use the *bdw082010* dataset [4], which consists of 96 hand-torn document pages that show either pictures, text, or both. Our preprocessing closely follows [4] to obtain an approximation of each fragment's contour: From a binary segmentation mask that identifies the foreground region of each piece we determine the set of outer contour points $P$ using the algorithm of Suzuki et al. [18]. Afterwards we apply the Douglas-Peucker algorithm [19] to find a small subset of *support points* $\hat{P} = \{\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_n\} \subseteq P$ that constitutes a less complex description of the piece's contour. By connecting consecutive pairs of support points through line segments one finally obtains a polygon that approximates the exact contour up to a predefined precision.

**Ground truth.** As explained in [4] the dataset is partitioned into three disjoint sets: {train}, {val}, and {test}. After the data preprocessing step, each page has been put together manually. Hence the layout of each page as well as the upright orientation of each piece is known. Based on this manual reconstruction, we call two fragments $s$ and $t$ *connected* iff four or more support points have an adjacent counterpart on the other piece. Each such pair of support points is called an *inlier*. We found that examples with less than four inliers are only loosely connected and tend to be negligible for document reconstruction – hence these examples were left out in our experiments.

Throughout the rest of this work, adjacent contour regions are represented by intervals of support points, which are denoted by $R(s)$ and $R(t)$. Each of these intervals contains its inliers as well as all intermediate support points from its respective polygon. Thus each interval defines a polyline that approximates a subsegment of the piece's exact outer contour $P$. For example, fig. 5 gives a schematic illustration how inliers and intermediate points between ground truth intervals $R(s) = \{i, \ldots, i{+}5\}$ and $R(t) = \{j, \ldots, j{+}4\}$ are used for the extraction of a training example.

## 4 Partial Contour Matching for Document Pieces

We define partial contour matching between pieces $s$ and $t$ as the task of identifying contour regions $R(s)$ and $R(t)$ along which those pieces were once adjacent in the original document. Our method builds on recent work [2] that introduces a variant of MSAC for aligning pairs of fragments. To determine boundary segments where pieces complement each other, the authors create an initial set of hypotheses (Euclidean transformations) from pairs of candidate inliers. All hypotheses are then verified in a two-step procedure, to identify the one alignment that best recovers the pieces' spatial relationship. The authors show experimentally that incorporating prior knowledge about the pieces' orientations helps to

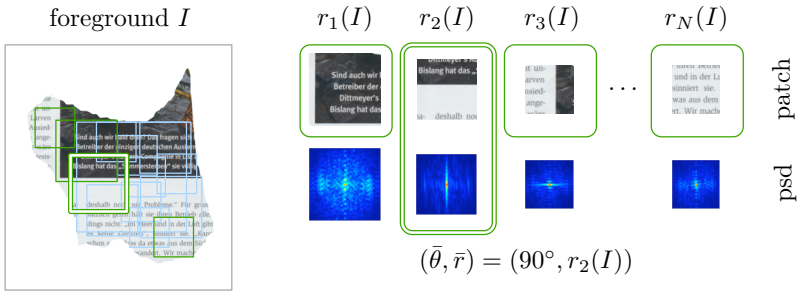


**Fig. 1. Left:** Patches of different sizes are positioned greedily to cover foreground region $I$. **Right:** Below a few highlighted patches we show their power spectral density (psd). Estimate $\bar{\theta}$ for the piece's orientation always stems from a single patch $\bar{r}$, which is chosen according to eq. (2).

improve alignment results. Besides, discarding inconsistent hypotheses effectively reduces the search space and hence speeds up alignment.

Since the studies in [2] are limited to a simulated text detector for orientation assignment, our first contribution is to introduce a method for estimating orientations in practice. For this purpose we implement the recent approach of Hollitt and Deeb [20], who estimate an image's orientation from its Fourier transform. In section 4.1 we first adapt their method for arbitrarily shaped document pieces. Although this approach gives robust estimates in many cases, it is likely to fail when document pieces convey little or no information about their dominant orientation. As discussed in section 4.2 we overcome this issue by using a discriminative model that identifies incorrect estimates. Finally, we explain in section 4.3 how this classifier is integrated into the alignment procedure.

### 4.1 Orientation Estimate from Fourier Transform

We briefly recap the technique of Hollitt and Deeb [20] to estimate the dominant orientation of an image. The idea is to find the direction along which the image shows the greatest variation in intensity values, because usually, its *upright direction* is either equivalent or perpendicular to that direction. Instead of working in the spatial domain, the authors apply a Fourier transform on the image to find its dominant orientation in the frequency domain. Therefore let $I(x, y) \in \mathbb{N}^{n \times n}$ denote an image, and let $(\mathcal{F}I)(\xi_x, \xi_y) \in \mathbb{C}^{n \times n}$ be its Fourier transform. Instead of $\xi_x, \xi_y$, which are conjugate to axes $x$ and $y$, one can equivalently consider the Fourier transform to be a function of polar coordinates $\xi_\rho, \xi_\theta$. To find the direction of strongest spatial variation we sum over the *power spectral density (psd)* along $\theta$:

$$g(\theta; I) = \sum_{\xi_\rho} |(\mathcal{F}I)_m(\xi_\rho, \xi_\theta)|^2 \tag{1}$$

The subscript in $(\mathcal{F}I)_m$ refers to a masked output after applying a bandpass filter. By filtering high frequencies we avoid a bias for diagonal orientations. On

the other hand, ignoring very low frequencies eliminates the impact of illumination changes.

Aside from bandpass filtering we also need to account for the fact that we are dealing with arbitarily shaped pieces. First of all, we need to ignore strong gradients at the piece's outer contour, which are uninformative for its true upright orientation. On the other hand, some foreground regions may be detrimental for orientation estimation, e.g., if the foreground region partially covers a natural image. To choose an optimal foreground subregion we hence use a sliding window approach to position patches of varying size. As shown in fig. 1, each patch covers a square region of interest on foreground $I$. We greedily position patches $[r_i(I)]_{i=1\ldots N}$ one by one, from large to small. Before a new patch is placed, we verify that its mutual overlap with any of the previously positioned patches is not too high. Finally we compute the Fourier transform, separately for each patch, and choose the orientation that shows the strongest spatial variation among all orientations and patches:

$$(\bar{\theta}, \bar{r}) = \underset{\theta, r_i(I)}{\operatorname{argmax}} \left\{ g(\theta; r_i(I))/B_i \right\} \tag{2}$$

In the above equation, bandwidth $B_i$ of the bandpass filter for the $i$-th patch is used for normalization to avoid a systematic bias for larger patches.

### 4.2 A Discriminative Model for Orientation Estimates

Since the method discussed so far always yields an orientation estimate, we now turn to learn a discriminative model (SVM) to decide whether an estimate should be trusted or needs to be invalidated.

We found that a strong peak $g(\bar{\theta}; \bar{r})$ is indicative for a robust estimate and hence should not be discarded. In contrast, we need to reject ambiguous orientation estimates in cases where multiple peaks occur (e.g., for textured image regions). We formalize this idea by sampling sums of spectral densities along different directions $\theta \in [\bar{\theta} - 45°, \bar{\theta} + 45°]$ in steps of $\omega = 0.5°$, using eq. (1). This yields a 181-dimensional descriptor:

$$\boldsymbol{\vartheta}_{ori}[90 \pm i] = g(\bar{\theta} \pm i\omega; \bar{r}) \,/\, g(\bar{\theta}; \bar{r}),\ \forall i \in \{0, \ldots, 90\} \tag{3}$$

By centering the descriptor around $\bar{\theta}$ we make it comparable with descriptors of other patches, because they all have their peak at the same position. Furthermore, we normalize it by $g(\bar{\theta}; \bar{r})$ to obtain a characterization of the relative strength of the peak. To gain partial invariance against shifts we further aggregate descriptor values into blocks of 5°, 15°, and 45°, respectively. For each of these $2 \times (9 + 3 + 1)$ blocks we compute its mean and standard deviation and append these values to $\boldsymbol{\vartheta}_{ori}$. Note that all the information needed is readily available from the spectral densities computed in eq. (2). Finally, to complement this representation from the frequency domain, we extract four Haralick texture features [21] (*angular second moment*, *contrast*, *correlation*, and *entropy*) on patch $\bar{r}$ and add them to our final descriptor of $(181 + 26 + 4)$ values.

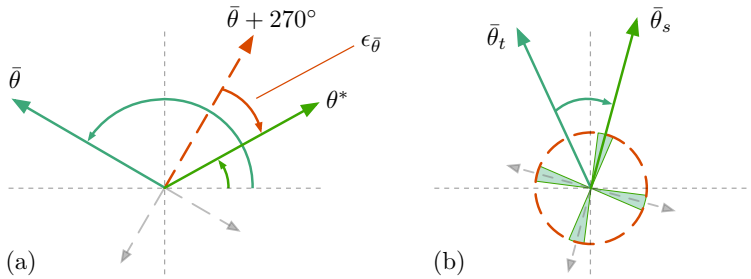


**Fig. 2. Left:** The minimal relative error $\epsilon_{\bar{\theta}}$ (see eq. (4)) is computed from orientation estimate $\bar{\theta}$ (modulo 90°) and true orientation $\theta^*$. **Right:** Orientation estimates for two fragments $s$ and $t$. If both estimates $\bar{\theta}_s$ and $\bar{\theta}_t$ were approximately correct, one could restrict the orientation domain to only four intervals during alignment.

To distinguish correct from incorrect orientation estimates, we then train a linear support vector machine using SVMLight [22]. For this purpose we categorize patches into positive and negative training examples. Therefore we compute orientations $\bar{\theta}$ of randomly oriented pieces, according to eq. (2), and keep track of the error regarding their ideal upright directions $\theta^*$ that are known from the ground truth. Since our approach works in the frequency domain, $\bar{\theta}$ often fails to give a piece's upright direction; however, it may still identify its correct orientation. We note that in some cases, straight lines in vertical direction can also cause the estimate to be correct modulo 90°. For this reason we categorize training examples based on their *minimal relative error*, which we define by:

$$\epsilon_{\bar{\theta}} = \min_k \left\{ \theta^* - (\bar{\theta} + k \cdot 90°) \right\} \in [0°, 45°] \tag{4}$$

It becomes clear from fig. 2a that this relative error is zero only if orientation estimate $\bar{\theta}$ is correct modulo 90°. We found that our orientation estimates are very precise for the majority of examples. Hence if $\epsilon_{\bar{\theta}}$ is less than 2° we consider the descriptor of patch $\bar{r}$ to be a positive example – otherwise it yields a negative example. Next we discuss different strategies how to incorporate our orientation estimates during alignment, and explain how to adjust the decision threshold of our SVM by performing cross-validation on pairwise examples.

### 4.3 Experiments

Given two pieces $s$ and $t$ that were digitized with unknown orientation, we first estimate their orientations $(\bar{\theta}_s, \bar{\theta}_t)$ to bring them in presumably upright direction. We denote these rotated fragments by $\bar{s}$ and $\bar{t}$. To actually align the rectified fragments according to [2], we hold $\bar{s}$ fixed, while computing an Euclidean transformation $h$ that aligns the second piece onto the first. This aligned version of $\bar{t}$ is in the following referred to by $h(\bar{t})$. We now want to discuss different strategies for computing hypothesis $h$:

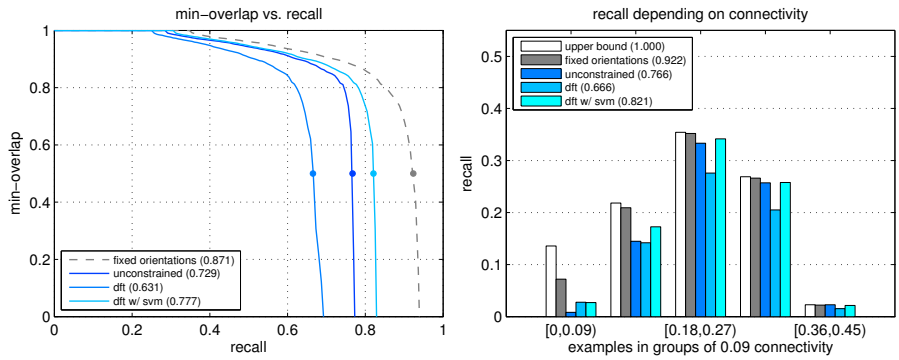


**Fig. 3. Left:** Evaluation in terms of min-overlap vs. recall. Reducing the search space selectively (dft w/ svm) gives the overall best performance. **Right:** Recall evaluated for different levels of connectivity, at a fixed min-overlap of 0.5. The white bar represents all examples within each connectivity group and hence upper bounds the recall to 1.0.

**Baseline (fixed orientations).** As baseline for our experiments we compute the best possible hypothesis by treating the upright orientation of both pieces as known and fixed. Since we use the true orientations from ground truth we only have to determine an optimal translation for $h$.

**Unconstrained.** As illustrated by the broken circle in fig. 2b, not using any orientation estimate comes down to considering all relative orientations between pieces. While this approach ensures that the correct hypothesis can not be falsely discarded, it is also computationally very expensive.

**Use Estimates from DFT.** To accelerate the computation of $h$, the key idea put forward in [2] is to discard transformations for which the relative orientation between $\bar{s}$ and $h(\bar{t})$ is inconsistent with their estimate. Applying this simple rule speeds up the alignment process significantly due to the limited number of hypotheses that need to be evaluated. As illustrated in fig. 2b, trusting the estimates (mod. 90°) narrows the search range to only four small intervals. Obviously, this inevitably yields an incorrect result if either estimate is incorrect.

**Conditionally Use Estimates from DFT.** To get the best of both worlds, we reduce the search space selectively, e.g., when clear lines or text are present on both pieces. Whenever the estimates for both pieces are predicted to be correct by our SVM model, we can safely shrink the hypothesis space as explained above. Otherwise, if either of the two estimates is presumably incorrect (e.g., for pieces showing natural images), we perform an unconstrained search instead.

To optimize prediction performance, we perform cross-validation on all pairwise examples from {train}+{val}. If we encounter pieces with correct estimates, we want the SVM to give a positive prediction for both. We call this a *joint true positive* example. A *joint false positive* on the other hand is if any of the two examples is assigned an incorrect estimate, but the SVM still classifies both as positive. Based on these outcomes we tune the model threshold to optimize the

(a) correct (b) correct (c) correct (d) incorrect

**Fig. 4. Left:** In (a)-(c) we show aligned pieces with increasing connectivity from left to right. Adjacent contour regions (green) have strong mutual overlap with their ground truth segments (min-overlap close to 1). **Right:** The pieces in (d) were not connected in the document and thus have 0% connectivity.

$F_\beta$ score, which is a weighted average of precision and recall. While $\beta=1$ is the harmonic mean of the two, we use $\beta=0.5$ to attach a higher importance to precision than recall. We feel that precision is more important, because joint false positives likely yield incorrect alignments. After adjusting the threshold, we retrain the model on the full {train}+{val} datasplit and report performances on {test}: There we achieve a $F_{0.5}$ score of 0.9155, which corresponds to a precision of 92.0% and recall of 89.7%. Using the so chosen threshold effectively narrows the search from examining all relative orientations (unconstrained) to only four intervals (dft) for 66.9% of all pairwise examples.

**Conclusions and Results.** To assess the quality of alignments we follow the methodology in [2] and report performances in terms of min-overlap vs. recall. The *min-overlap* for two aligned pieces takes on values from $[0, 1]$ that reflect how accurately two regions $R(\bar{s})$ and $R(h(\bar{t}))$ match with their ground truth segments. Formally, the min-overlap is defined as

$$\textit{min-overlap}(\hat{l}_s, \hat{l}_t) = \left\lfloor \frac{\hat{l}_s \cap l_s}{\hat{l}_s \cup l_s}, \frac{\hat{l}_t \cap l_t}{\hat{l}_t \cup l_t} \right\rfloor, \tag{5}$$

where $l_s$ and $l_t$ are the pieces' adjacent line segments stemming from ground truth intervals $R(s)$ and $R(t)$, and $\hat{l}_s$ and $\hat{l}_t$ are the polylines associated with predictions $R(\bar{s})$ and $R(h(\bar{t}))$, respectively. As illustrated in fig. 4(a)-(c), correctly aligned pieces yield adjacent contour regions (green) having high mutual overlap with the annotated segments. On the other hand, figure 4(d) shows that originally disconnected pieces always score min-overlap 0, because $l_s = l_t = \emptyset$.

In fig. 3a we finally plot overlap-recall curves for our different search strategies. We note that an unconstrained search gives better results than blindly relying on estimates from the Fourier transform (dft). However, using conditional estimates (dft w/ svm) improves the area under curve from 72.9% to 77.7%, at a maximum recall of 82.9%. We achieve comparable results to those reported in [2], despite not relying on a simulated text detector. Since the dataset contains a substantial number of pieces that lack text and straight lines, assuming reliable orientation estimates for those pieces would be an overly idealized assumption.

In these situations our SVM proves to be very reliable in deciding whether or not to shrink the search space, depending on the pieces at hand.

In our second experiment we evaluate recall for different levels of connectivity. In [2] the *connectivity* of two pieces is defined as the length of their adjacent boundary segments relative to their overall contour lengths. As common sense suggests, correctly aligning pieces with low connectivity (fig. 4a) is inherently more difficult than others sharing large parts of their boundaries (fig. 4c). This claim is substantiated by the plot in fig. 3b, which shows that our alignment mostly fails when facing low-connectivity examples ($0\% - 18\%$). In this scenario, using the SVM clearly improves the recall over an unconstrained search. For high-connectivity examples ($18\% - 45\%$), the SVM still performs on a par, despite being much faster due to the inherently smaller hypothesis space.

## 5 Ranking Sequences by Structured Output Prediction

To quantify the compatibility of adjacent contour regions we learn a stuctured prediction model that incorporates contour information as well as content-based local features. In sections 5.1 and 5.2 we formulate our prediction problem and give details about the model. Afterwards in section 5.3 we briefly introduce the dynamic programming algorithm used for solving the inference task at training- and test time. Finally in section 5.4 we discuss how to train the structural support vector machine, before concluding the paper with an evaluation in section 5.5.

### 5.1 Structured Output Prediction

As introduced before, we denote by $\bar{s}$, $h(\bar{t})$ two aligned fragments that become adjacent along intervals $R(\bar{s})$ and $R(h(\bar{t}))$. Motivated by the protein alignment model of Yu et al. [23], let us first define a *sequence* $\boldsymbol{y} \in \mathcal{Y}$ between two contour regions as list of subsequent operations $\boldsymbol{y} = (y^1, \ldots, y^{|\boldsymbol{y}|})$. Each of these elements corresponds to exactly one *operation* between the two pieces. In the ideal case, a sequence is only composed of *matches* between support points, i.e., a point from the first polygon becomes associated with one from the second polygon. To score any sequence defined over output space $\mathcal{Y}$ we next define a linear function

$$\Omega_{\boldsymbol{w}}(\boldsymbol{y}; R(\bar{s}), R(h(\bar{t}))) = \boldsymbol{w} \cdot \boldsymbol{\Psi}(\boldsymbol{y}; R(\bar{s}), R(h(\bar{t}))) \;, \tag{6}$$

in which $[\cdot]$ denotes the dot product, $\boldsymbol{w}$ is the cost model that is to be learned, and $\boldsymbol{\Psi}(\boldsymbol{y}; R(\bar{s}), R(h(\bar{t})))$ is a joint feature vector of fixed size that describes the structured output $\boldsymbol{y}$ on intervals $R(\bar{s})$, $R(h(\bar{t}))$. To rank hypothesis $h$ that aligns piece $t$ onto $s$, we aim to find the most promising sequence $\boldsymbol{y}^*$ among all possible sequences along the contour regions:

$$\boldsymbol{y}^* = \underset{\hat{\boldsymbol{y}} \in \mathcal{Y}}{\operatorname{argmax}} \left\{ \Omega_{\boldsymbol{w}}(\hat{\boldsymbol{y}}; R(\bar{s}), R(h(\bar{t}))) \right\} \tag{7}$$

As we will see shortly in section 5.3, we are able to find this optimal sequence regarding model $\boldsymbol{w}$ by dynamic programming.

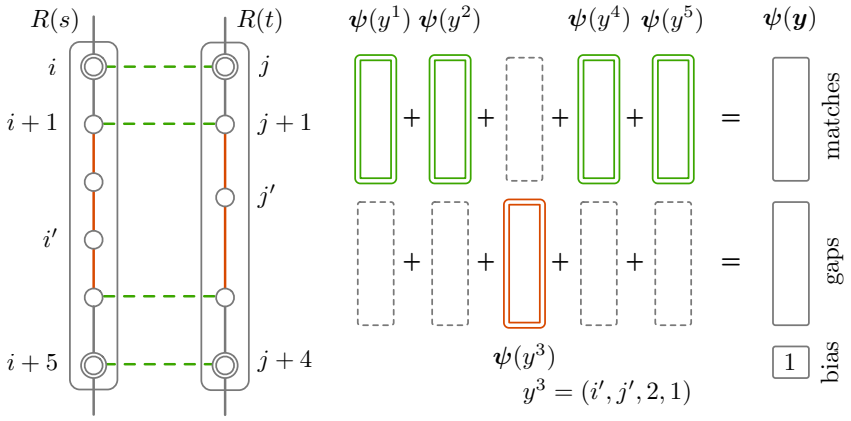


**Fig. 5.** Left: Schematic illustration of a ground truth sequence of length five, with four match operations (green). In between, gap $y^3$ (red) spans over two support points on $R(s)$ and one point on $R(t)$, respectively. According to eq. (8), descriptors for individual operations sum up to the sequence's descriptor, as illustrated on the righthand side.

### 5.2 Decomposition of Sequences into Operations

We begin by motivating the *match* operation, which associates one support point from piece $s$ with another on piece $t$. For instance, we write $y^k = (i', j', 0, 0)$ to associate support point $i' \in R(s)$ with $j' \in R(t)$. The latter two 0's indicate that a match operation does not affect any points other than $i'$ and $j'$. Intuitively, we only want to create a match if $i'$ has been adjacent to $j'$ in the original document (i.e., an inlier). When dealing with real-world documents however, pieces do not always have well aligned support points.

To account for this we complement matches with *gap* operations, which allow to skip (possibly multiple) support points on either of the two fragments. A gap is denoted by $y^k = (i', j', \delta_s, \delta_t)$ to indicate that immediately before $i'$ and $j'$, $\delta_s$ and $\delta_t$ many points are omitted, respectively. Following [23] we define a joint feature descriptor that is linear in its operations. The descriptor for the entire sequence is decomposed into the sum over descriptors $\boldsymbol{\psi}(y^k)$ for individual operations:

$$\boldsymbol{\Psi}(\boldsymbol{y}) = \left[ \sum_{k=1}^{|\boldsymbol{y}|} \boldsymbol{\psi}(y^k), 1 \right] \tag{8}$$

A constant 1 is appended for learning a global bias in our model. For notational simplicity we omit contour ranges, e.g., by writing $\boldsymbol{\Psi}(\boldsymbol{y}) \equiv \boldsymbol{\Psi}(\boldsymbol{y}; R(s), R(t))$ in short. An example for a sequence that is decomposed into its individual operations is given in fig. 5. Since one operation can either be a match or a gap, it is possible to split the descriptor from eq. (8) into blocks. Using our simplified notation we write $\boldsymbol{\psi}(y^k) = \left[\boldsymbol{\psi}_M(y^k), \boldsymbol{\psi}_G(y^k)\right]$, where subscripts in $\boldsymbol{\psi}_M$ and $\boldsymbol{\psi}_G$ refer to match and gap, respectively. Note that depending on the operation,

only one descriptor is set, while the other one is initialized to $\mathbf{0}$. Each descriptor reflects the compatibility of only those parts of $R(s)$ and $R(t)$ that are affected by the respective operation. Analogous to the descriptor, our model decomposes into $\boldsymbol{w} = [\boldsymbol{w}_M, \boldsymbol{w}_G, w_B]$. Due to space limitations we keep the discussion about implementation details short:

**Match Operation.** The descriptor for a match operation $y^k = (i', j', 0, 0)$ describes the compatibility of support point $i'$ on $R(s)$ to $j'$ on $R(t)$. In eq. (9) the first two elements introduce an absolute- and a relative offset. The latter is set to the reciprocal of the minimum over contour lengths $R_s = |R(s)|$ and $R_t = |R(t)|$. This effectively rewards matches between short contour intervals stronger than for long ones.

The remaining component $\boldsymbol{d}_M(y^k)$ introduces a set of dissimilarities computed from a multimodal feature representation. These values can be interpreted as the inherent cost imposed on matching support points. From a geometric perspective, we determine the points' dissimilarity in terms of their local polygonal approximation (line length and enclosed angle), as well as their spatial proximity after alignment through hypothesis $h$. In addition we use three content-based features, introduced in [4], to represent shape, color and texture within the vicinity around each point. For each feature channel we compute a dissimilarity value from the feature descriptors. All dissimilarities [1] are stacked into vector $\boldsymbol{d}_M(y^k)$, which is finally appended to the offsets to give our *match descriptor*:

$$\boldsymbol{\psi}_M(y^k) = \left[1, 1/\lfloor R_s, R_t \rfloor, -\boldsymbol{d}_M(y^k)\right] \tag{9}$$

**Gap Operation.** Gaps allow us to deal with noise in the contour approximation. The principle idea is that each gap operation $y^k = (i', j', \delta_s, \delta_t)$ is delimited by two matches. For example, the gap $y^3$ in fig. 5 is immediately followed by a match $y^{k+1} = (i' + 1, j' + 1, 0, 0)$, and it is preceded by a second match $y^{k-1} = (i' - \delta_s, j' - \delta_t, 0, 0)$. Analogous to a match, the first two values in eq. (10) introduce offsets to learn a flat penalty. Since gaps should always have a negative contribution to the sequence score, those offsets have a negative sign.

In the second component we want to penalize long gaps and those that are uneven in size (in terms of $\delta_s$ and $\delta_t$). Therefore we introduce two dissimilarity values $\delta_s + \delta_t$ and $\lceil \delta_s, \delta_t \rceil - \lfloor \delta_s, \delta_t \rfloor$. Furthermore, we add dissimilarities to encode the divergence of the pieces' polygon approximations along the gap regions. Once again, all dissimilarities [1] are combined into a vector $\boldsymbol{d}_G(y^k)$. Our *gap descriptor* is obtained by stacking the dissimilarity vector onto the gap penalty offsets:

$$\boldsymbol{\psi}_G(y^k) = \left[-1, -1/\lfloor R_s, R_t \rfloor, -\boldsymbol{d}_G(y^k)\right] \tag{10}$$

### 5.3 Inference via Dynamic Programming

Assuming two adjacent contour regions $R(s)$ and $R(t)$ we now introduce a modified variant of the Smith-Waterman algorithm [17] to determine the best sequence along these intervals. We found those regions $R(s) = \{i, \ldots, i + R_s - 1\}$

[1] Dissimilarities are computed from feature-dependent kernel functions that yield positive real numbers within comparable value ranges.

and $R(t) = \{j, \ldots, j + R_t - 1\}$ to provide very reliable estimates for delimiting sequences – hence we restrict each sequence to start with a match in $(i, j)$ and end with a match in $(i + R_s - 1, j + R_t - 1)$.

Since each sequence is decomposable into individual operations, one can incrementally extend sequence prefixes into longer sequences. Therefore we append only one operation at a time to an existing prefix. By $p[i', j']$ we refer to the score of the prefix that starts in $(i, j)$ and ends in $(i', j')$. To compute this score one has to choose the operation that yields the overall highest score for $p[i', j']$. That is, we either append a match operation with score

$$p[i', j'] = p[i' - 1, j' - 1] + \boldsymbol{w}_M \cdot \boldsymbol{\psi}_M((i', j', 0, 0)) \,, \tag{11}$$

or else introduce a gap if

$$p[i', j'] = \max_{\delta_s, \delta_t} \Big\{ p[i' - \delta_s, j' - \delta_t] + \boldsymbol{w}_G \cdot \boldsymbol{\psi}_G((i', j', \delta_s, \delta_t)) \Big\} \tag{12}$$

is a higher score. By performing a traceback from the last to the first match we get a sequence that fully extends over intervals $R(s)$ and $R(t)$. Note that the global bias $w_B$ has to be added only once for the first match in $(i, j)$ as each prefix builds upon this initial operation.

### 5.4 Learning Problem

Learning cost model $\boldsymbol{w}$ is treated as regularized empirical risk minimization problem. Our training set $\{(\boldsymbol{x}_1, \boldsymbol{z}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{z}_N)\}$ consists of pairs of fragments $\boldsymbol{x}_i = (s_i, t_i)$ and annotations $\boldsymbol{z}_i = (l_i, \boldsymbol{y}_i)$. Labels $l_i \in \{-1, +1\}$ are needed to distinguish positive from negative examples. For positive (connected) examples, their correct sequences $\boldsymbol{y}_i$ are part of ground truth data. Pieces from negative examples on the other hand do not share a common border in the original document – thus we align them once in advance to also infer non-empty sequences. Using the hinge loss we obtain the following optimization problem:

$$\begin{aligned} &\underset{\boldsymbol{w}, \xi_i \geq 0}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\| \boldsymbol{w} \right\|^2 + C \sum_i \xi_i \right\} \\ \text{s.t.} \quad &\forall \boldsymbol{x}_i \in P_{pos} : \boldsymbol{w} \cdot \boldsymbol{\Psi}(\boldsymbol{y}_i; R(s_i), R(t_i)) \geq +1 - \xi_i \\ &\forall \boldsymbol{x}_i \in P_{neg}, \forall \hat{\boldsymbol{y}}_i : \boldsymbol{w} \cdot \boldsymbol{\Psi}(\hat{\boldsymbol{y}}_i; R(\bar{s}_i), R(h(\bar{t}_i))) \leq -1 + \xi_i \\ &\boldsymbol{w}_M, \boldsymbol{w}_G \geq \boldsymbol{0} \end{aligned} \tag{13}$$

This formulation establishes one margin constraint on each example, whose violation yields a positive slack value $\xi_i$ that is penalized in the objective function. Parameter $C$ balances these training errors versus the model's capability of generalizing beyond training examples.

To train our model we use the stochastic gradient descent solver from [24]. After obtaining an initial model from a moderate number of examples, we iteratively infer new hard negative examples using the dynamic programming method introduced in section 5.3. We continue to re-train the model from a cache of hard examples until either (i) a fixed number of iterations is reached, or (ii) no new hard negative examples can be found.

| feature evaluation | | ranking performance | | | |
|---|---|---|---|---|---|
| | **16 pieces** | | **16 pieces** | **24 pieces** [2] | **32 pieces** [2] |
| mAP (all features) | **0.7906** | mAP | **0.7906** | **0.6626** | **0.5835** |
| mAP (base) | 0.7252 | mAP (w/ pruning) | 0.9903 | 0.8962 | 0.8740 |
| mAP (base w/ shape) | 0.7258 | mAP (w/ pruning + overlap) | 0.9917 | 0.9601 | 0.9462 |
| mAP (base w/ color) | 0.7754 | mAC (w/ pruning) | 0.0113 | 0.1193 | 0.1092 |
| mAP (base w/ texture) | 0.7895 | mAC (w/ pruning + overlap) | 0.0113 | 0.0493 | 0.0350 |
| | | mAP [4] | **0.7341** | **0.5932** | **0.5374** |
| | | mAC [4] | 0.0109 | 0.0380 | 0.0378 |

**Table 1. Left:** Performance evaluation for different feature combinations. **Right:** Evaluation of the ranking capabilities of our structural SVM in terms of (i) mean average precision (mAP), and (ii) mean accumulated cost (mAC) from [4]. See text for details.

### 5.5 Experiments

Despite the interest of researchers on this topic, related work reporting quantitative results on a standardized dataset is sparse. Thus we provide a reproducible baseline on the publically available bdw082010 dataset (see section 3). The test set of this dataset consists of $M = 48$ magazine pages, which have been torn into $N$ pieces each. Since each page contributes $(N^2 - N)/2$ unique piece pairs, the test set consists of $M \cdot (N^2 - N)/2$ examples in total. We first align each pair of pieces and then assign a compatibility score (see eq. (6)) according to the predicted optimal sequence. An example is positive if its pieces were connected in the original document, and only aligned pieces with min-overlap 0.5 or higher are considered to be true positives. For our experiments we treat each individual page as "*query*", and examples from that very page are ranked according to their scores. From the resulting ranked lists we compute the average precision (AP) for each page, and finally the mean average precision (mAP) over the entire test set with 48 queries in total. We argue that the mAP is indicative for the performance of many approaches [4, 10–12], as they all rely on successively merging the most compatible pieces.

In our first set of experiments we evaluate the importance of content-based features. Using only information from the pieces' polygons in our match descriptor (eq. (9)) yields 72.5% mAP. Next we augment this basic representation with either shape-, color-, or texture information. As can be seen in table 1 (left), we do not gain much by adding shape information. This is because alignments are very accurate and hence shape dissimilarities add only little discriminativeness. However, using color or texture information results in substantial performance improvements to 77.5% and 78.9%, respectively. A combination of all local features gives the overall best performance with 79.1% mAP. As can also be seen from the table, the mAP declines with an increasing number of pieces per page.

[2] For 24 and 32 pieces the bdw082010 test set contains only 12 pages.


(a) 16 pieces (b) 24 pieces (c) 32 pieces

**Fig. 6.** Examples for pages that are reconstructed only from pairwise alignments.

This comes at no surprise, as the number of negative (disconnected) examples grows quadratically, while the number of positive examples only increases approximately linearly with $N$. As summarized in table 1, we outperform previous work [4] on this dataset in terms of mAP, for all degrees of fragmentation.

In the second set of experiments we analyze how mAP relates to the quality of individually reconstructed pages. As explained in [4], the layout of each page can be recovered by constructing a spanning tree from alignments (edges) between pairs of pieces (nodes). Using our compatibility score for the edge weights we create a maximum spanning tree using Kruskal's algorithm [25]. To reconstruct the document we retain only those alignments that correspond to spanning tree edges. This effectively prunes all but $N-1$ alignments, hence why the results are named mAP (w /pruning) in table 1. Again we compute the AP, separately for each page, and report the mAP over all 48 pages. Using all features combined we achieve 99.0% for $N=16$, which means that only very few alignments were false positives. In fig. 6 we give examples for pages that were reconstructed very accurately just from local alignments between pairs of pieces. To further improve our reconstruction results (mAP w/ pruning + overlap), we augment our spanning tree algorithm with a geometric verification that invalidates alignments resulting in overlapping pieces. While its effect is less pronounced for 16 pieces per page, we note that for $N=32$ the mAP increases from 87.4% to 94.6%.

Finally, we report results regarding a complementary performance measure that was put forward with the release of the dataset [4]. The mAC in table 1 corresponds to the median over the mean accumulated costs per page. Intuitively, this value is an indicator for the effort of repositioning pieces within an existing solution, according to the ground truth page layout. Despite that our preliminary postprocessing does not yet account for evidence from previous alignments, our approach gives competitive results compared to those obtained with a revised implementation of [4]. Although differences in terms of mAC are seemingly insignificant, a comparison by mAP clearly proves the superiority of our method.

We conclude that mAP makes for a rigorous and transparent comparison of results, because prediction outcomes (true positives and false positives) have an immediate interpretation, just as for object detection and retrieval methods.

## References


1. Geller, T.: Darpa shredder challenge solved. Communications of the ACM **55** (2012) 16–17
2. Richter, F., Ries, C.X., Romberg, S., Lienhart, R.: Partial contour matching for document pieces with content-based prior. In: IEEE International Conference on Multimedia and Expo. (2014)
3. Torr, P.H.S., Zisserman, A.: Mlesac: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding **78** (2000)
4. Richter, F., Ries, C.X., Cebron, N., Lienhart, R.: Learning to reassemble shredded documents. IEEE Transactions on Multimedia **15** (2012) 582–593
5. Yao, F., Shao, G.: A shape and image merging technique to solve jigsaw puzzles. Pattern Recognition Letters **24** (2003) 1819–1835
6. Bunke, H., Bühler, U.: Applications of approximate string matching to 2d shape recognition. Pattern Recognition **26** (1993) 1797–1812
7. Biswas, A., Bhowmick, P., Bhattacharya, B.B.: Reconstruction of torn documents using contour maps. In: IEEE International Conference on Image Processing. Volume 3. (2005) 517–520
8. Zhu, L., Zhou, Z., Zhang, J., Hu, D.: A partial curve matching method for automatic reassembly of 2d fragments. In: International Conference on Intelligent Computing. Volume 345. (2006) 656–650
9. Justino, E., Oliveira, L.S., Freitas, C.: Reconstructing shredded documents through feature matching. Forensic Science International (2006) 140–147
10. Stieber, A., Schneider, J., Nickolay, B., Krüger, J.: A contour matching algorithm to reconstruct ruptured documents. In: Proceedings of the 32nd DAGM conference on Pattern recognition. (2010) 121–130
11. Zhu, L., Zhou, T., Hu, D.: Globally consistent reconstruction of ripped-up documents. IEEE Transactions on Pattern Analysis and Machine Intelligence **30** (2008) 1–13
12. Cao, S., Liu, H., Yan, S.: Automated assembly of shredded pieces from multiple photos. In: IEEE International Conference on Multimedia and Expo. (2010) 358–363
13. da Gama Leitão, H.C., Stolfi, J.: A multiscale method for the reassembly of two-dimensional fragmented objects. IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002) 1239–1251
14. Papaodysseus, C., Panagopoulos, T., Exarhos, M., Triantafillou, C., Fragoulis, D., Doumas, C.: Contour-shape based reconstruction of fragmented, 1600 bc wall paintings. IEEE Transactions on Signal Processing **50** (2002) 1277–1288
15. Wolfson, H.J.: On curve matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 483–489
16. Chen, L., Feris, R., Turk, M.: Efficient partial shape matching using smith-waterman algorithm. In: CVPR Workshop. (2008)
17. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology **147** (1981) 195–197
18. Suzuki, S., Abe, K.: Topological structural analysis of digital binary images by border following. Computer Vision, Graphics, and Image Processing **30** (1985) 32–46
19. Douglas, D., Peucker, T.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer **10** (1973) 112–122

20. Hollitt, C., Deeb, A.S.: Determining image orientation using the hough and fourier transforms. In: Proceedings of the 27th Conference on Image and Vision Computing. (2012) 346–351
21. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for image classification. In: IEEE Transactions on Systems, Man, and Cybernetics. Volume SMC-3. (1973) 610–621
22. Joachims, T.: Advances in kernel methods - support vector learning. MIT Press, Cambridge, MA (1999) 169–184
23. Yu, C.J., Joachims, T., Elber, R., Pillardy, J.: Support vector training of protein alignment models. Journal of Computational Biology **15** (2008) 867–880
24. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32** (2010) 1627–1645
25. Kruskal, J.: On the shortest spanning subtree and the traveling salesman problem. Proceedings of the American Mathematical Society **7** (1956) 48–50