Semi-automatic Forensic Reconstruction of Ripped-up Documents

Patrick De Smet
Nationaal Instituut voor Criminalistiek en Criminologie (NICC/INCC)
Vilvoordsesteenweg 100, B-1120 Brussels, Belgium, patrick.desmet@just.fgov.be
Dep. TELIN/IPI, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium.

Abstract

Manual reconstruction of ripped-up documents can be a very difficult and time-consuming task. This paper discusses a semi-automatic toolset that can be used for reconstructing ripped-up documents. First, we present a brief overview of our current fragment scanning, image segmentation and feature computation methods. Then, we discuss how fragments can be matched using their computed features. Next, we report on our global multi-fragment matching strategy and discuss the interactive components of the toolset that can be used to control and iterate through an entire reconstruction process. Finally, we illustrate the efficiency of the proposed approach with experimental results.

1. Introduction

Manual reconstruction of a recovered set of ripped-up documents can be a difficult and time-consuming task. For small scale problems one can often use any manual "random search" strategy for quickly matching and recomposing the fragments. For larger scale problems the human visual system in combination with our relatively small amount of short-term memory, limits our reconstruction capabilities. More specifically, the mathematical complexity is of the order $N(N-1)$ as N fragments will need to be matched to $N-1$ other fragments. Also, the physical space required for spreading out and examining the fragments can grow quickly and inhibit any efficient matching. For these reasons several researchers have turned towards image processing methods for automating the entire reconstruction process; see [1], [2], [3] and references discussed therein. Forensic investigators are interested in such a computer-assisted reconstruction process because it avoids continuous handling of the evidence, as well as the use of any type of adhesive, etc. Additionally, it is important to note that the semi-automatic approach is a desired property; forensic investigators are interested in user controlled tools that optimize their workflow, and that easily allow correction when the algorithms fail (all currently existing algorithms have been shown to have important limitations). Finally, electronic re-editing, high-lighting or annotation, and reproduction of any of the (partial) reconstruction results can be implemented easily.

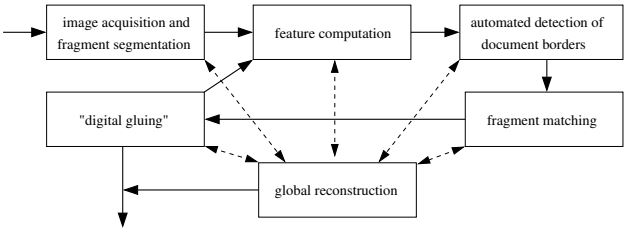

Figure 1. The proposed framework.

Below, we present a framework of techniques and tools, see Fig. 1, that enables semi-automatic reconstruction of a set of ripped-up documents. The individual components are controlled by our global reconstruction approach (dotted lines); i.e., we use an interactive and iterative process of accepting, selecting or fine-tuning the obtained partial reconstruction results. This process consists of first trying to reconstruct the outer frames, followed by filling in the frames with non-border fragments if necessary.

This framework extends and improves upon some of our earlier work reported on in [1]; i.e., a novel set of interactive tools has been implemented, the page border results for merged fragments are computed more efficiently and intelligently (see Sect. 3), a new page border matching method now considers possible orientation information for both matching and visualization (see Sect. 4.1), and the method used for accepting a fragment merging candidate has been simplified so that the list of remaining fragments is updated and/or recomputed correctly (see Sect. 6). Additionally, all important datastructures have been extended so that they can cache the most frequently used data (fragment

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.7

IEEE computer society

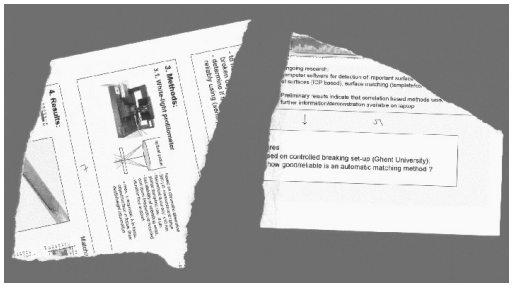

Figure 2. Example flatbed image scan.

images, chaincodes, polygons, bounding boxes etc.) which resulted in important improvements in terms of reduced I/O and computational load.

2. Image Acquisition and Segmentation

A first important step constitutes building a digital database of the given set of real-world paper fragments. We apply the following processing steps. First, we select a subset of the fragments and put them on top of a flatbed scanner. Next, we put a uniformly coloured piece of paper or plastic on top of the fragments and a digital image scan is made. This scanning process is repeated until the entire set of paper fragments has been processed. This results in a set of digital images similar to the one shown in Fig. 2.

To segment the fragments, each of the scans is fed into a background-foreground modeling tool. Figure 4 shows the results obtained for the image shown in Fig. 2. The actual method uses a recursive First In First Out queue flooding algorithm. The segmentation tool then presents the original image, alongside with the bounding box, a contour trace-around and the page border detection results (see below) of the segmented regions. The user can "accept" all of the obtained (per scan) results. Alternatively, clicking on an image area will pop-up a zoom-in window for the region. The user can then again select either "accept" or "fine-tune". The "fine-tune" option allows the user to perform manual segmentation changes, or change the fragments'orientation or page border detection results (see below). We also use an interactive pixel painting tool for solving two additional problems that can occur, i.e., the leaking of background colour into the fragments when they contain similarly coloured pixels, and the appearance of "thick edges" when paper fibres are visible in the tearing edges.

3. Feature and page border computation

For each segmented fragment a chain code is used to trace around its contour. These chain codes are then resampled using a fixed Euclidean distance d. Next, we determine the angles a_i; see Fig. 3. However, as the angles a_i are not rotation invariant, we then use the relative angular changes as our set of contour features, i.e., we compute $c_i = \mid a_{i+1} - a_i \mid_{-\pi}^{+\pi}$.

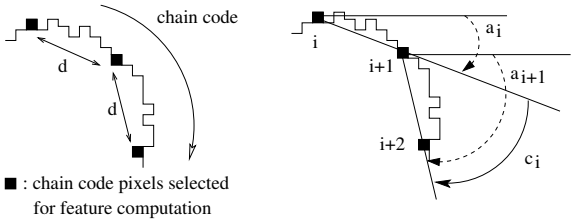

Figure 3. Shape feature computation.

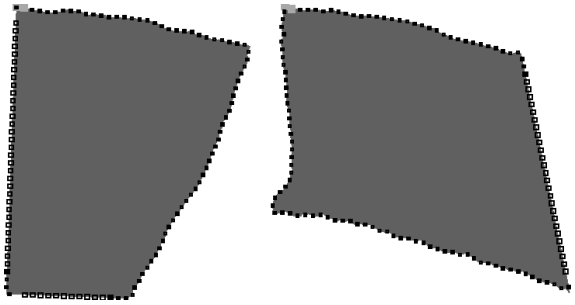

Figure 4. Page border detection results

An interesting technique that is often applied by human jigsaw puzzle solvers is to first look for the border and corner pieces. For document reconstruction tasks it makes sense to also try to use this method because most (multi-page) documents will only be ripped a few times resulting in all or the majority of the fragments containing some part of the page borders. As already indicated above the interactive toolset will present the automatic border detection results to the user. If necessary or desired, the user can alter the algorithmic parameters, i.e.,l, σ_{max}, L_{min}, see below. Additionally, the user can also indicate how the detected page borders should be (re)oriented (e.g., so that the text would appear upright). Currently, our page border detection procedure operates as follows. First, we compute $\mu_i = \sum_j a_j/(l+1)$ and $\sigma_i = \sqrt{(\sum_j (a_j - \mu_i)^2/(l+1)}$ for $j \in [i - l/2, i + l/2]$. Next, for each contour index i we then determine if: $\forall\ j \in [i - l/2, i + l/2],\ \sigma_j < \sigma_{max}$. If so , we set psl_i = TRUE, else we set psl_i = FALSE (psl: possible straight line). Below $l = 3$ and $\sigma_{max} = 3$ unless reported otherwise. Finally, the runs of psl_i all with value TRUE and longer than length L_{min} indicate the detection of a page border area (we use $L_{min} = 6$ unless reported otherwise). Figure 4 illustrates some example results (black squares).

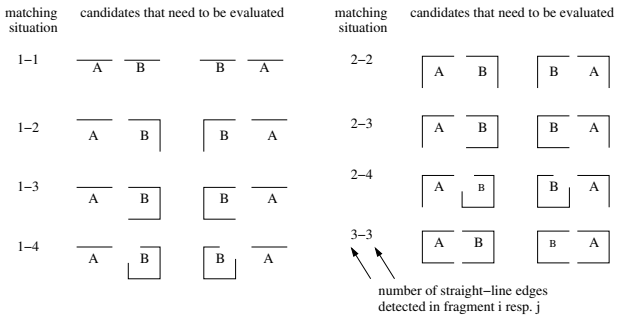

Figure 5. Page border matching possibilities

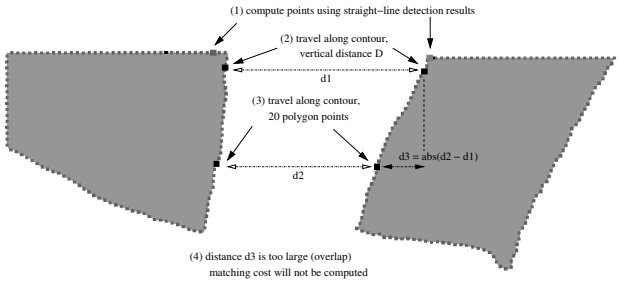

Figure 6. Computation of contour segments used for matching page border fragments.

4. Efficient Fragment Matching Strategies

4.1. Matching of Page Border Fragments

In this section we only consider the subset of fragments selected by using the method described above (Section 3).

First, let us consider matching any page border fragment with all others; i.e., $FM(i,j)$ matches any fragment $i = 1,\ldots,N$, to all fragments $j = i+1,\ldots,N$. If no information about the orientation of the fragments is given, we compute $FM(i,j)$ by considering all possible matching situations as displayed in Fig. 5 in which the fragments are shown as abstract representations that correspond to detected page borders. For each situation two matches should be evaluated, i.e., left-right (or "AB") and right-left ("BA") positioning. If the user has provided information about the orientation of the fragments during the segmentation step, our method will eliminate any matching candidates that cannot occur, e.g., a top and bottom page fragment (determined by their user given orientation and the position of the page border) should not be matched.

Additionally, we do not need to compute all $FM(i,j)$ costs. From the page border detection results we compute the leftmost (resp. rightmost) point on the straight line for the fragment in the right (resp. left) fragment matching position. Next, we trace the fragment contour in clock-wise (resp. anti-clockwise) order and travel a vertical distance D ($D = 50$ for our experiments). We then travel further up until a strong angular change is found or more than 20 contour points have been visited. As a result we find two contour segments. Consider e.g. the fragments shown in Fig. 6; this corresponds to a 1-2 matching situation, BA positioning. Clearly, it is not very useful to compute $FM(i,j)$; we use an additional elimination condition that stipulates that the end points should not be further apart than a distance X (we use $X = 150$). Also, we use additional conditions that ensure that the reconstructed page will be a rectangle. Finally, the user can specify the expected page size (A4, letter, etc.). If the two contour segments are not eliminated by all these conditions, the segment matching cost is computed using the pixel based gap overlap method described in Sect. 5. In the last step, all $FM(i,j)$ costs are sorted and the fragment pairings are presented to the user which can accept or reject them; see also below.

Finally, an alternative approach that closely resembles a human puzzling approach can be based on computing $FMP(i,j)$ for a fixed i with $j = 1,\ldots,N$ and $j \neq i$. This much faster method matches a single chosen fragment, and is particularly interesting when the user wishes to start the interactive process as soon as possible; see also Section 7.

4.2. Matching of Non-border Fragments

To match the non-page-border fragments, we use an adapted "string matching" method that eliminates all page border contour areas. Next, the algorithm tries to find the best matching length $l_{m,C,C'}$ and the corresponding feature positions within all feature strings C and C' that yield a minimal set of differences. To find these optimal substrings, we iteratively evaluate all shifted contour feature strings $C^{(i)} = (c_i, c_{i+1}, \ldots, c_n, c_1, \ldots, c_{i-1})$ and $C'^{(j)} = (c'_j, c'_{j+1}, \ldots, c'_m, c'_1, \ldots c'_{i-1})$ by computing the associated dissimilarity costs $d(i,j,l_x) = \sum_{k=0}^{l_x} \mid c_i - c'_{j+k} \mid$. Example results are given in Section 7.

5. Applying Digital Glue

After a candidate match has been accepted, the fragments have to be repositioned and merged. The merging step (i) translates one of the initial contour points onto the corresponding point of the other fragment, and, (ii) rotates over the angle formed in between the two contour segments used for matching. For the page border fragments we only need to consider translation as rotation was already computed during page border detection and matching.

If available, the orientation information of the fragments will always be used to display them accordingly.

To compute an actual matching cost and reposition the fragments accurately, a search space centered around the initial transformation parameters is evaluated; i.e., the repositioning is optimized using a pixel based gap and overlap computation procedure in the matching contour areas. If the automated results are still not satisfactory, the user can interactively adjust the transformation parameters.

6. Global Document Reconstruction

As already indicated, the global reconstruction of a document is carried out interactively and normally starts by reconstructing the outer frame of page border fragments. This is done by iteratively computing $FM^{(i)}$; see above. Note that if a small number of ripping iterations was used to destroy the document, no non-border matching will be needed.

As soon as the user decides to terminate the page border frame reconstruction process, the remaining set of fragments will be matched using the non-border, shape-based approach discussed in Section 4.2.

Accepting any fragment pairing as valid will merge them together into a single new fragment. Next, we remove any remaining candidate matching pairs that contain one of the fragments. If the matching list becomes empty or the user issue a recomputation command, we only (re)compute the matching costs for all new fragments with the remaining non-matched fragments and each other.

7. Experiments and Results

For illustrating the proper functioning and the performance of our framework we will use three sets of fragments. The first set contains eight fragments obtained by ripping up a single colour page a conference poster. A second set of nine colour fragments is used mainly to demonstrate the non-page-border based fragment matching procedure; see below. A third set of fragments contains 48 fragments obtained by ripping up three pages of a draft scientific paper.

Due to a lack of space we only summarize some of the obtained results. More detailed (reconstruction) results and progress reports about our work can be obtained from: `http://telin.ugent.be/~pds/papers/icdar2009/`.

For the first set of fragments, the progressive build-up results of the outer frame page border fragments are shown in Fig. 7 and Fig. 8. Figure 7 illustrates the first four matching candidates obtained in the first iteration (order left-to-right, top-to-bottom; their resp. matching costs in terms of gap/overlap pixel counts are: 687, 1026, 1291, and 1566).

To obtain the results shown in Fig. 8 seven keys were pressed; (accept, recompute) for the first and second iteration, and (reject, accept, recompute) for $FM^{(3)}$. Without

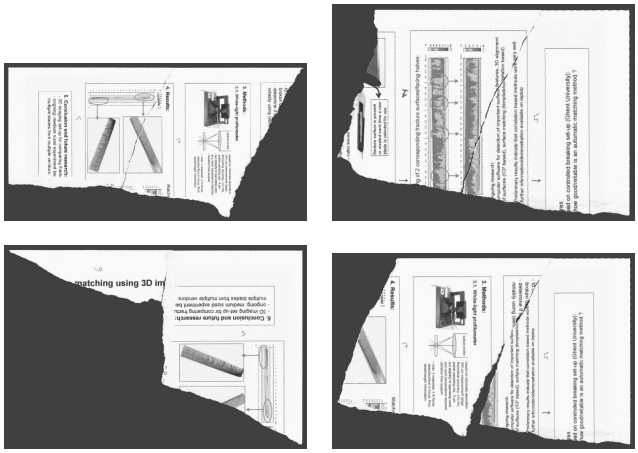

Figure 7. Reconstruction results for the first set of fragments (top 4 results, iteration 1).

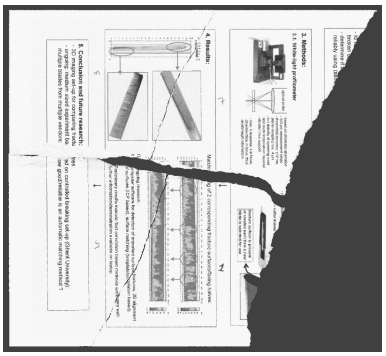

Figure 8. Fourth reconstruction iteration for the first set of fragments.

recomputation commands, the user would also need to press seven keys to obtain an identical result; the user will have to reject a couple of non-matches and the program will automatically recompute as soon as its matching list is empty.

The total processing time needed for obtaining the result shown in Fig. 8 was about 26 s (Intel P9400, 2.4 GHz, 4 GB memory) including all user interaction, and image I/O, etc. Further optimization will surely improve these results. For generating all input of the matching program 2 s were needed (detection of page borders, rotation of images and masks into matching situations, all I/O operations, etc.).

Figure 9 illustrates similar results obtained for a total interaction of seven keys pressed when the thick-edge of the fragment shown in the top left corner of Fig. 8 was removed during initial fragment segmentation; instead of one meta-group of four fragments, six fragments have now been grouped into two meta-fragments.

Using *only* the non-page-border contour segment match-

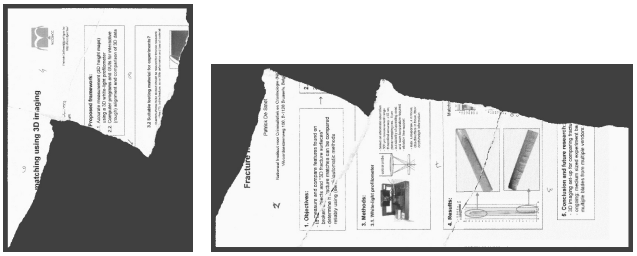

Figure 9. Reconstruction results (first set) when thick edge removal is applied first.

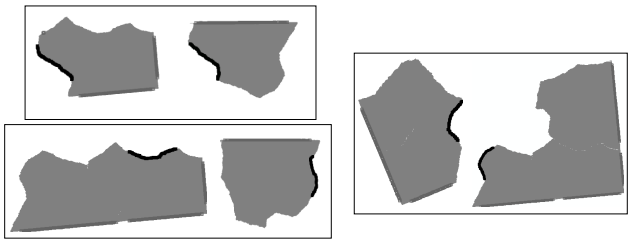

Figure 10. Iterative reconstruction for the second set of fragments.

ing procedure described in Sect. 4.2 we processed the second set of fragments. This illustrates how the approximate shape description can also be used on its own during the interactive matching. The binary image map reconstruction results are illustrated by Fig. 10. The second, third, and fourth iteration results are shown (in the order top-left, bottom-left, right). The page border detection results are highlighted on the contours; they will not be used for shape matching. Black contour points indicate the contour segments computed as yielding the best matches.

For the third set of fragments reconstruction results similar to the ones above were obtained. In this case we set $L_{min} = 10$ and $\sigma = 2$, because most of the fragments had mostly near straight line ripping contours. To give an indication of the efficiency of the FMP method we set up the following experiment. Several persons were asked to manually reassemble the set of forty-eight fragments. The time needed to find the first correct matching pair varied from 20 s to 40 s. All subsequent matching pairs were on average found within a similar time-frame but this was mainly dependent on the persons's "puzzling" approach — i.e., one page contained a small table, two of the other pages contained a list of references; some persons used these characteristics for searching faster. To compute all feature inputs for the matching program about 12 s are needed (same system as mentioned above). Computing and interactively approving the first correct match takes only 2 s using the FMP method. As the reconstruction moves forward in-

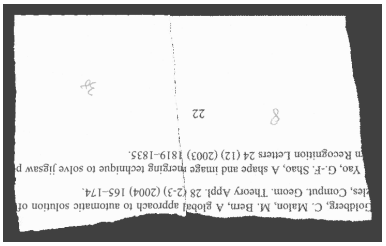

Figure 11. The first correct FMP match obtained for the third set of fragments.

correct matches frequently seem to overtake the correct matches, increasing the need for more disapproval interactions with the user. This is mainly due to the strong shape similarities of the fragments (this is the reason why we are now studying (binary) colour and text-line methods). Fortunately, the on-screen (dis)approval of matching candidates can be done very quickly by pressing a single key. Timing and reconstruction results for subsequent iterations are available through our website mentioned above.

8. Conclusion

In this paper we presented a framework of image processing and computer vision techniques that can be used to automate the reconstruction of (a set of) ripped-up documents. We discussed fragment scanning and segmentation, contour feature computation and matching, and "digital gluing" using gap and overlap computations. Although each of the components and their GUI tools can be and is being improved and extended further, we believe that the proposed tools may offer a time and cost efficient method for reconstructing ripped-up documents. Currently, larger scale experiments are being studied and new methods are being developed for matching using colour and text line detection. We hope this paper may inspire other researchers to discuss or suggest other possibilities for future work, which we believe to be many.

References

[1] P. De Smet. *Towards Automated Reconstruction of Ripped-up Documents (to appear)*, volume Computational Forensics of *Computational Intelligence*. Springer Verlag, 2009.

[2] E. Justino, L. S. Oliveira, and C. Freitas. Reconstructing shredded documents through feature matching. *Forensic Science International*, 160:140–147, 2006.

[3] L. Zhu, A. Zongtan Zhou, and D. Hu. Globally consistent reconstruction of ripped-up documents. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(1):1–13, 2008.