Open access • Proceedings Article • DOI:10.1109/ICASSP.2009.4959853

# Document reconstruction using dynamic programming — Source link

Andre Pimenta, Edson J. R. Justino, Luiz S. Oliveira, Robert Sabourin

**Institutions:** Pontifícia Universidade Católica do Paraná

**Published on:** 19 Apr 2009 - International Conference on Acoustics, Speech, and Signal Processing

**Topics:** Search algorithm, Prim's algorithm, Approximation algorithm, Dynamic programming and Computational complexity theory

# DOCUMENT RECONSTRUCTION USING DYNAMIC PROGRAMMING

*Andre Pimenta, Edson Justino, Luiz S. Oliveira, and Robert Sabourin*

Pontifícia Universidade Católica do Paraná, Curitiba, PR, Brazil
École de Technologie Supérieure, Montreal, QC, Canada

## ABSTRACT

In this work we propose a methodology for document reconstruction based on dynamic programming and a modified version of the Prim's algorithm. Firstly, we use polygonal approximation to reduce the complexity of the boundaries and extract features from them. Thereafter, these features are used to feed the LCS dynamic programming algorithm. The scores yielded by the LCS algorithm are then used into a modified Prim's algorithm to find the best match among all pieces. Comprehensive experiments on a database composed of 100 shredded documents support the efficiency of the proposed methodology. When compared to global search algorithms, this approach brings an improvement of 18% in the number of fragments reconstructed.

***Index Terms***— Forensics, Document Reconstruction, Dynamic Programming

## 1. INTRODUCTION

Reconstruction of shredded document is a tedious and laborious task that should be performed by forensic document examiners quite often. The amount of time necessary to reconstruct a document depends on the size and the number of fragments, and it can be measured in days or even weeks. In order to alleviate the manual effort of the forensic examiner, some methods for reducing the complexity of reconstruction and reassembly problems using digital images have been proposed in the literature. Most of these methods were developed for solving related problems, such as the jigsaw puzzles [1]. In general, they are based on specific shape and color features as well as the relationships that may exist between several jigsaw puzzle pieces.

An efficient algorithm for puzzle solving was proposed by Wolfson in [2]. In this work, the author presents two curve matching algorithms where the boundaries are represented by shape feature strings which are obtained by polygonal approximation. It fails when the number of puzzle pieces become larger, though. Another interesting strategy is proposed by Kong and Kimia [3]. They re-sample the boundaries by using a polygonal approximation in order to reduce the complexity of the curve matching. Dynamic programming is used to align the pieces.

These techniques have been applied to other fields such as archeology and art restauration. In these cases, the goal is to reconstruct two-dimensional objects that have been broken or torn into a large number of irregular fragments. Willis and Cooper [4] address the problem of artifact reconstruction discussing 2D and 3D approaches. Leitão and Stolfi [5] propose an algorithm based on incremental dynamic programming to reconstruct ceramic tiles. Interesting results also have been reported by Papaodysseus et al [6] where the focus is the reconstruction of archaeological wall-paintings.

Regarding the reconstruction of documents for forensics purposes, few works can be found in the literature. In a more recent work, De Smet [7] discuss a formal analysis of the problem of reconstructing ripped-up documents when the remnants can be recovered as an ordered stack of fragments. Justino et al [8] propose a local reconstruction of shredded documents based on polygonal approximation and feature matching.

In this work we introduce a methodology based on dynamic programming and a modified version of the Prim's algorithm to reconstruct shredded documents. First we apply polygonal approximation to reduce the complexity of the boundaries and overcome specific problems faced in document reconstruction. Works related to jigsaw puzzles explore the fact that ordinary puzzle pieces have smooth edges and well defined corners. However, pieces of paper shredded by hand does not follow this pattern.

After polygonal approximation, we extract relevant features from each piece of the document and then perform the matching using the LCS dynamic programming algorithm. The scores yielded by the LCS algorithm are then used into a modified Prim's algorithm to find the best match among all pieces. We demonstrated by comprehensive experiments that the proposed procedure produces interesting results for the problem of document reconstruction. Regarding our previous work [8], we have addressed some key points which enable us to improve the reconstruction rate in about 18%.

This research has been supported by The National Council for Scientific and Technological Development (CNPq) grant 471496/2007-3

## 2. METHODOLOGY

Our methodology is composed of four major steps: Pre-processing, feature extraction, matching, and reconstruction. Initially, each piece of the document is pre-processed through polygonal approximation so that the complexity of the boundaries can be reduced. Thereafter, we extract relevant features from each piece of the document and then perform the matching using the LCS dynamic programming algorithm. The scores yielded by the LCS algorithm are then used into a modified Prim's algorithm to find the best match among all pieces. In the following sections we describe in details each module of the methodology.

### 2.1. Pre-processing

Traditional puzzle solving algorithms usually take into account smooth edges and well defined corners. However, dealing with shredded documents is quite more complex. The act of shredding a piece of paper by hand often produces some irregularities in the boundaries, which makes it impossible to get a perfect curve matching. To overcome this kind of problem, we have tested different algorithms, and the one that brought the best results was the well-known Douglas-Peucker (DP) algorithm [9]. This algorithm implements a polyline simplification and it is used extensively for both computer graphics and geographic information systems. Figure 1 shows an example of this process using different levels of approximation. For more details about pre-processing, see [8].

**Fig. 1**. Inner and outer boundaries produced by shredding.

### 2.2. Feature Extraction

After the complexity reduction through polygonal approximation, the next step consists in extracting features that should be used during the matching process. The feature extraction can be seen also as a complexity reduction process, since it converts the polygon in a sequence of features. Here, we propose a simple feature set that can be used to carry out the local matching.

The first feature is the angle of each vertex with respect its two neighbors. Consider for example the vertices A and B in the polygon depicted in Figure 2. The angle $\alpha$ is given by

$$cos(\alpha) = \frac{uv}{|u||v|} \tag{1}$$

We also verify whether such an angle is convex or concave. For example, in Figure 2, vertex B has a convex angle while vertex C has a concave one. To complete our feature set, we compute the distances between the vertex and its neighbors (next and previous in a clockwise sense). Such distances are achieved by means of the well-known Euclidean distance. Table 1 describes the feature vector extracted from the polygon depicted in Figure 2. The last two features are the coordinates of the vertex in the image.

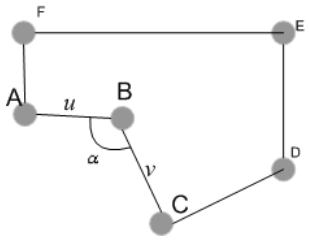


**Fig. 2**. Angle features extracted from the polygon.

**Table 1**. Description of the feature vector.

| Vertex | Angle | Distances | | X | Y |
|---|---|---|---|---|---|
| | | Next | Previous | | |
| A | 270 | 40.0 | 45.0 | 10 | 70 |
| B | 120 | 45.0 | 43.6 | 55 | 67 |
| C | 200 | 43.6 | 115.7 | 67 | 25 |
| D | 245 | 115.7 | 11.0 | 180 | 0 |
| E | 270 | 110.0 | 170.0 | 180 | 110 |
| F | 270 | 170.0 | 40.0 | 10 | 110 |

This table can be read as follows: The angle of the vertex B, which is computed by using vertex A and C, is 120 degrees. The Euclidean distances between B and its neighbors A and C are 45.0 and 43.6, respectively. The coordinates of the vertex B in the image are (55,67).

### 2.3. Matching

Here, the main goal is to compute a degree of similarity between the boundaries of each fragment of the image. To perform this task, we have used the LCS (Longest Common Subsequence) algorithm, which is a dynamic programming algorithm devoted to find the longest subsequence common to all sequences in a set of sequences. In our case, the sequences are the features extracted previously.

The LCS starts with a matrix $E$ of size $(M+1)\times(N+1)$, where $M$ and $N$ are the length of the two sequences ($X$ and $Y$) being analyzed. The first row and column are filled with zeros. The remaining values respect the following definition:

$$E_{ij} = \max \begin{cases} E_{i-1,j-1} + S \\ \max(E_{i,j-1} + P, 0) \\ \max(E_{i-1,j} + P, 0) \end{cases} \tag{2}$$

where $S = 1$ if $X_i = Y_j$ and 0 otherwise. $P$ is the penalty value. Consider for example two sequences $X =$

$\{0223110133\}$ and $Y = \{0223101133\}$. The corresponding matrix $E$ is depicted in Figure 3. In spite of the fact that both sequences do not match completely, we are able to recover partial matches from the LCS matrix. In Figure 3 the main diagonal shows two partial matches. The score is given by the last cell of the match. In the case of the biggest match, the score is 5.

| | | 0 | 2 | 2 | 3 | 1 | 1 | 0 | 1 | 3 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2 | | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 1 | 1 |
| 1 | | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 1 | 0 | 0 |
| 1 | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 2 | 0 | 0 |
| 0 | | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| 1 | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 1 | 0 | 0 |
| 3 | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 1 |
| 3 | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 3 |

**Fig. 3**. (a) Matrix $E$ for sequences $X$ and $Y$ .

In our experiments, a high penalty value ($P = -100$) was considered because the mismatching of any feature means that the entire sequence is not valid. Consider now the following two sequences $Z = \{100, 90, 100\}$ and $W = \{100, 150, 100\}$, representing three features used for reconstruction. If, for instance, the $Z_2$ and $W_2$ represent the angle, the match for 90 would be 270, hence the pair (90,150) makes no sense at all for the reconstruction. This exemplifies why we penalize any mismatch. In the end of the matching process we have a list of all possible matches with its respective scores, which should be used to reconstruct the image of the document. To perform the reconstruction in a more structured way, we have adopted a graph representation, which shows clearly the relationship among all fragments. Figure 4a shows the polygonal approximation for a document composed of seven fragments while Figure 4b presents all the relationships found by the LCS in a graph.

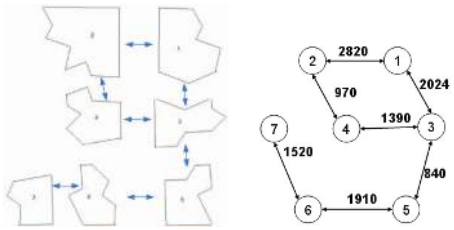


**Fig. 4**. (a) Results of the polygonal approximantion, and (b) The graph representing the relationships among the fragments found by the LCS algorithm

### 2.4. Reconstruction

As we can see in Figure 4, the original graph can have cyclic links, which makes the reconstruction problem more complex. To mitigate such a complexity, we transform the matching graph into a minimum weight spanning tree, using Prim's algorithm [10]. The following modifications were made in the algorithm to make it suitable for document reconstruction: **i)** Instead of choosing a random vertex to start the algorithm, we select the vertex with the highest score, **ii)** The scores represent how favorably are the connections. In fact, we could say we are looking for the maximum weight spanning tree, because the higher the score the best is the matching. **iii)** For each node added into the tree we align the corresponding fragments by translating and rotating the respective fragment with respect to its neighbor. Rotation and translation are saved for the image reconstruction. Very often, we can have a matching of several features, but a part of the fragment can occlude its neighbor. If that happens (false positives), the node is removed from the tree. This problem is depicted in Figure 5, where the matched vertex and the occluded area are highlighted.


**Fig. 5**. (a) Two candidate fragments (b) Occlusion produced after alignment.

Note that the last modification can result in several trees since the elimination of one node can broke the sequence making it impossible to find a path from the first to the last node. When that happens, the document will be reconstructed partially only, because we consider just the biggest resulting tree. After building the Prim's tree, the reconstruction of the document is straightforward.

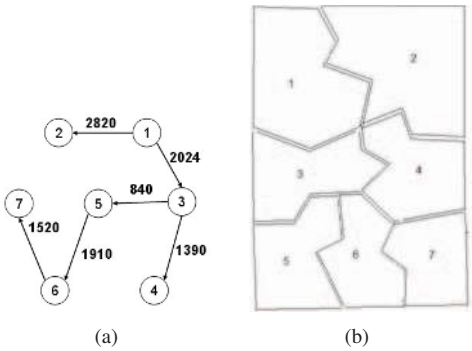


**Fig. 6**. (a) The Prim's tree and (b) the document reconstructed.

We start from node 1 and visit all other nodes using the information about rotation and translation found previously.

Figure 6 shows the Prim's tree for the graph presented in Figure 4 and the respective document reconstructed. We can notice, for example, that some links such as 2-4 were removed, but did not impact in the final reconstruction.

## 3. EXPERIMENTS

To validate the proposed methodology we have used a database composed of 100 documents. Those documents were shredded into 3-16 fragments and their size range from 1cm $\times$ 1 cm to 5cm $\times$ 5 cm. It is worth of remark that the documents were randomly shredded, in other words, we have not used any criteria to perform this task. The same database was used in [8].

The proposed approach was able to put together, in average, 75% of the fragments for each document. In 61% of the documents, the reconstruction was complete or just one fragment was lost. Even when the reconstruction is not total, this is a important tool that can be used to help forensic experts in this laborious task.

Table 2 compares our results to those reported in [8], which considers the same database. As we can notice, the methodology proposed in this work brings an considerable boost in the performance of the algorithm. Note that 75% does not mean 75% of the document totally reconstructed, but rather that in average, the algorithm is able to reconstruct 75% of each document. The same holds for False Positive and Error rates reported in Table 2. In average, the algorithm misplaces 14% of the fragments during reconstruction and did not use 11% of the fragments because of the lack of matching.

The False Positive column represents those fragments that have good features for matching but that are clearly misplaced during reconstruction. This is one of the limitations of the proposed approach since only features yielded by the poligonal approximation have been considered. This kind of problem could be mitigated by using some contextual features. Finally, the column error represents those fragments that were not used because the LCS did not find any plausible matching. A common source of error in this case is the lack of features due to linear cuts, which did not produced vertex and angles that we use as features.

**Table 2**. Comparative results.

| Strategy | Correct Reconstruction (%) | False Positives (%) | Error (%) |
|---|---|---|---|
| Proposed | 75 | 14 | 11 |
| Justino et al [8] | 57 | 24 | 19 |

## 4. CONCLUSION AND FUTURE WORKS

In this paper, we have proposed a methodology for document reconstruction based on polygonal approximation and dynamic programming. The polygonal approximation reduces complexity and produces the features that feed the dynamic programming algorithm. Thereafter, a strategy based on graphs is used to perform the image reconstruction, giving to the forensic expert a clear idea about the document, even when it is not totally reconstructed.

The results reported in the paper show a important boost in the reconstruction rate. However, there is a lot of room for improvement. Still considering the proposed strategy, we believe that the use of other sequence chains produced by the LCS matrix can be used to solve some false positive problems. Besides, we plan to add other kind of features such as texture and color.

## 5. REFERENCES


[1] F.-H. Yao and G.-F. Shao, "A shape and image merging technique to solve jigsaw puzzles," *Pattern Recognition Letters*, vol. 24, pp. 1819–1835, 2003.

[2] H. Wolfson, "On curve matching," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 12, pp. 483–489, 1990.

[3] W. Kong and B. Kimia, "On solving 2D and 3D puzzles under curve matching," in *CVPR*, 2001, pp. 583–590.

[4] A. R. Willis and D. B. Cooper, "Computational reconstruction of ancient artifacts," *IEEE Signal Processing Magazine*, vol. 25, pp. 65–83, 2008.

[5] H. C. G. Leitao and J. Stolfi, "A multiscale method for the reassembly of two-dimensional fragmented objects," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 24, pp. 1239–1251, 2002.

[6] C. Papaodysseus, T. Panagopoulos, M. Exarhos, C. Triantafillou, D. Fragoulis, and C. Doumas, "Contour-shape based reconstruction of fragmented," *IEEE Trans. Signal Processing*, vol. 50, pp. 1277–1288, 2002.

[7] P. De Smet, "Reconstruction of ripped-up documents using fragment stack analysis procedures," *Forensic Science Intern.*, vol. 176, pp. 124–136, 2008.

[8] E. Justino, L. S. Oliveira, and C. Freitas, "Reconstructing shredded documents through feature matching," *Forensic Science Intern.*, vol. 160, 2005.

[9] David Douglas and Thomas Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, pp. 112–122, 1973.

[10] R. C. Prim, "Shortest connection networks and some generalisations," *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.