Reconstructing Colored Strip-Shredded Documents based on the Hungarians Algorithm

Fatima Alhaj
King Abdullah II School of IT
University of Jordan
Amman, Jordan
fat9170261@fgs.ju.edu.jo

Ahmad Sharieh
King Abdullah II School of IT
University of Jordan
Amman, Jordan
sharieh@ju.edu.jo

Azzam Sleit
King Abdullah II School of IT
Amman, Jordan
azzam.sleit@ju.edu.jo

***Abstract*—One of the common problem in forensic science and investigation science is reconstructing destroyed documents that have been strip-shredded. This work intend to design a strip matching algorithm to resemble edges of the strips, in order to reconstruct the original document. The proposed algorithm is divided into three phases. First is image- based similarity evaluation to produce a score function (which includes building " distance matrix"). The second phase is assignment phase that matches a shred border pixels of the right side to a left side of another shred (using Hungarians algorithm). The third phase is defining the sequence according to the matched strips in order to merge the shreds and reconstruct the document. The proposed work is compared with a nearest neighbor search algorithm in term of accuracy and speed. The Hungarians reassembling algorithm scores better accuracy and run time than nearest neighbor reassembling. The proposed approach scored (96.2 percent) as an average accuracy for reassembling an available online benchmark.**

***Index Terms*—Document Reconstruction, Feature matching, Hungarians Algorithm, Nearest neighbor search, Strip-shredded documents.**

I. Introduction

Automatic shreds reconstruction involves finding a correct spatial arrangement of given shreds in order to reassemble a complete document. This problem is usually handled by historians and forensic investigators [6]. It is used in many domains such as: health informatics, insurance claim analysis [1], and military sector [2]. Also, it can be used in recovery of documents accidentally lost [1]. Manually reconstruction can be used, in which parts are arranged and analyzed as if it is a puzzle [3]. The huge number of possible shreds permutations of arrangements makes manual solution an inefficient, exhausting and time consuming. Many methodologies are followed to provide automated and semi-automated document reconstruction. Whether manual reconstruction or automated is used, the greatest challenge is the shreds identification and matching.

In general, shredding machines produce three categories of shreds: rectangular strips (spaghetti), cross-cut and circular [3]. This work aims to design, implement, and test an algorithm to solve the problem of reconstructing shredded document. This implies finding the correct positioning of a given n shreds, in order to form the original document. Each shred can be presented as a binary bitmap, and it is assume that the shreds are placed in the correct orientation. Few researches have worked on reconstructing strip-cut documents problem [4]. We intend to improve the strip-cut matching algorithm specifically, and outperform the sequential "best match" and "minimum distance" search for each shred. Searching for the nearest neighbor matching for each strip from both sides holds the drawback of being time consuming. Motivated by this drawback, a new approach is proposed to reconstruct strip shredded text documents by firstly specifying the problem as an optimization problem, secondly reformulating the problem as a maximum bipartite matching problem. The Hungarians algorithm was deployed to find the best match with a reduced complexity.

The paper is organized as follows. In Section 2, a brief overview of related work is given. In Section 3, the methodology and how obtaining our shredded document is explored. The naive algorithm used to solve this problem is described, along with its complexity analysis, and the optimized algorithm and how to proceed in its different phases, which put both algorithms in a comparative frame in terms of time complexity. In Section 4, the experiment results thorough quantitative evaluation of the proposed approach are presented. Finally, Section 5 concludes the paper.

II. RELATED WORK

The problem of reconstructing shredded documents is closely related to the problem of automatically solving jigsaw puzzles. Schauer et al. [5] considered the shredded document as a form of jigsaw puzzles. They specified three types of the fragments: the manually torn documents, the cross-cut

978-1-7281-2882-5/19/$31.00 ©2019 IEEE

shredded documents, and the strip shredded documents.

Some work used image-based similarity evaluation, such as Lin and Fan-Chiang [3], where they proposed a reconstruction algorithm for strip shreded document. Their algorithm is based on image feature matching and sorting graph-based representation of the shreds. In [3], they used color-matching-based method to produce an impressive accuracy.

On the other hand, another group of researchers concentrate on the text based document and exploits the character features to match shreds boundaries. Perl et al. [8] proposed an optical character recognition algorithm to match two shreds boundaries using characters histograms. When the lines of the text decrease, the precision of the paper reconstruction changes in non- increasing order.

Sleit et al. [4] proposed a solution for the reconstruction of crosscut shredded text documents (RCCSTD) problem based on iterative building of clusters related to shreds. Biesinger et al. [7] investigated the same problem with an improved genetic algorithm.

Butler and Chakraborty [3] proposed "Deshredder" approach, which provides a visual analysis and make use of user involvement to direct the reconstruction process. The approach represents shredded pieces as time series and uses nearest neighbor matching techniques, which enable matching not just the contours of shredded pieces, but also the content of shreds. Some literature deals with reconstructing strip shredded documents. These implies extracting information from the boundaries of the shredded documents strips [2], [3], [12], but their work dose not concentrate on the function order of the algorithm (time complexity). They focus on finding a solution other than finding a better run time solution.

Justino et al. worked in reconstructing hand shredded documents [10]. The proposed methodology include pre-processed each shred based on polygonal approximation in order to reduce complexity of the boundaries. The next stage is features extraction followed with matching stage. Shreds in hand-shredded documents usually yields to irregular boundaries, which need an extra processing before matching. They applied polyline simplification by using Douglas–Peucker (DP) algorithm. Justino et al. methodology's performance degrades as the number of shreds gets bigger since it affects the polygonal approximation.

III. Proposed Methodology

The result of the shredding process is a set of n shreds Sh = sh_0,...,sh_n, which also represents the input to the algorithm. This work is divided into three subsequent phases. First phase is applying similarity score function. This phase will result in $n \times n$ distance matrix. Second phase uses Hungarians method for bipartite matching to find the borders matches. The Third phase is defining the sequence according to the matched strips in order to merge the shreds and reconstructs the document.

A. Distance matrix

The objective of this stage is to define the differences between boarders of shred. Consider this step as a preprocessing step that includes applying a similarity score function. The score function calculates a score for each pair of shreds to measure the similarity (dissimilarity) between them. Specifically, it measures pixel difference. This step will result in $distance_L$=(sh_x, sh_y) or $distance_R$= (sh_x, sh_y) , where $distance_R(sh_x, sh_y)$ is the similarity score of placing sh_x to the right of sh_y and $distance_L(sh_x, sh_y)$ is the score of placing it to the left side. For n shreds, calculation of order of n^2 scores will be done.

Algorithm 1: Hungarians reassembling algorithm

```
   input : sh[] ← shreds
   output: Ordered sequence of shreds
1  distance[] ← ∞ , counter1 ← 0 , counter2 ← 0
2  while counter1 ≤ numberofstrips(sh) do
3      while counter2 ≤ numberofstrips(sh) do
4          distance[counter1][counter2]=
             strip-Distance(counter1,counter2) ( Total
             distances between band edges)
5      end
6  end
7  Hungarians-Reassembling(distance[],sh[]) returns
     indexes that assigning the best right match for left
     side related to every shred.
8  sequence[] ← 0
9  indexes = Hungarians-Reassembling(distance[])
10 while i,v in indexes do
11     val =distance[i][v]
12     sequence.append((v,val))
13 end
14 return sequence[]
```

The preprocessing steps are defined in Algorithm 1 (line 1-6). The innermost loop calculates the distance between two pixel values as a measure of how similar they are. This process deals with the pixel vector, whatever the color model used. Other loops ensure that comparing each shred with all other shreds. It returns the sum of distances between the rightmost column of pixels in a shred and leftmost column of pixels in another shred. The result matrix "distance" will have sums of distances between edges of each two shreds. The complexity of preprocessing in Algorithm 1 (line 1-6) is $n^2 \times h$, where h is the height of the shreds. Assuming that the height of a document is the number of shreds multiplied by some constant, then $h = C \times n$. The run time complexity is approximately O (n^3). The input used in this step is a shredded document that was shredded using a shredding function. This shredding function generates any number of shreds out of a document,

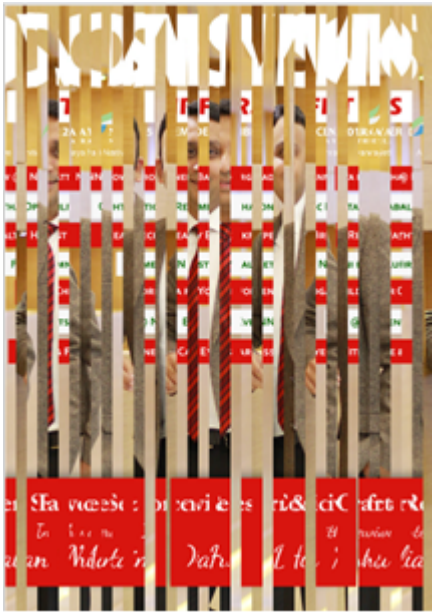

Fig. 1. A strip-shredded document into 40 strips ($n = 40$).

Fig. 2. Algorithm 1 distance matrix process for strip 1 and 28.

by splitting it into strips and shuffle the strips into random positions.Figure 1 shows an example for a shredded document into 40 strips.

The objective of this stage is to define the differences between boarders of shred. Consider this step as a preprocessing step that includes applying a similarity score function. The score function calculates a score for each pair of shreds to measure the similarity (dissimilarity) between them. Specifically, it measures pixel difference. This step will result in $distance_L$=(sh_x, sh_y) or $distance_R$= (sh_x, sh_y) , where $distance_R(sh_x, sh_y)$ is the similarity score of placing sh_x to the right of shy and $distance_L(sh_x, sh_y)$ is the score of placing it to the left side. For n shreds, calculation of order of n^2 scores will be done.

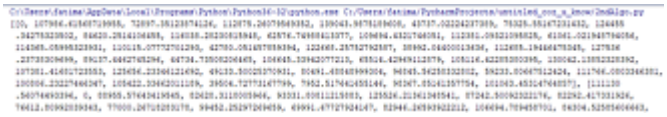

Fig. 3. A part of Algorithm 1 distance matrix for Fig.1 document.

Fig.2 shows an illustration for Algorithm 1 preprocessing. For each point (x, y) in strip 1 right side and strip 28 left side, a long all strip heights, the algorithm finds the difference between their pixels vectors (the absolute value). In distance matrix, distance [1][28] equals the summation of pixels distances for every point(x,y) along the strip height. Similar summation is done for all the points to fill the distance matrix. By doing that, the algorithm compares all shreds lefts sides with all shreds right sides.
e

Fig. 3 shows a part of the 40×40 distance matrix constructing by implementing Algorithm 1 (preprocessing part) on the document shown in Fig 1.

B. Hungarians Method for Bipartite Matching

A strip shredded document reconstructing problem can be defined as an assignment problem (matching) for a bipartite graph, where it includes two disjoint sets. The first step is matching the set of all shreds left-side boarder columns and the second one is the right side boarder columns. The right of a specific shred to be assigned to another shred left side. Thus, Hungarians algorithm suits perfectly for solving this kind of matching according to distances between shreds information, which was collected earlier in Algorithm1.

Hungarians algorithm was chosen because it can achieve a low order polynomial run time, has worst-case run time complexity of $O(n^3)$. Be side, optimality is guaranteed in Hungarians assignment algorithm [9]. The basic rule of the Hungarians is that the number of rows and columns should be equal, which is true about this two-dimensional matrix (distance) introduced in Algorithm 1. This similarity matrix is algorithm equivalent to the cost matrix in the Hungarians method.

In Algorithm 1 , line 7 calls ”Hungarians-Reassembling” as a function. ”Hungarians-Reassembling” as a part of the proposed methodology, was built according to Pilgrim work [11] and his definition to the Munkres’ Assignment algorithm, and was used to achieve the reassembling purpose.

Fig. 4 shows a part of the output of implementing ”Hungarians-Reassembling” using document of Fig. 1. Each pair of shreds are the best assignment found by implementing the Hungarians algorithm. The number after the arrow represent the sum of distances between the two strips. For example, the distance between the right border of strip 4 and

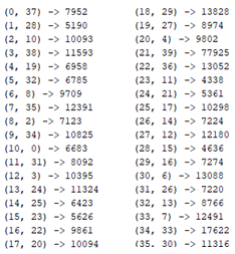

Fig. 4. A part of "Hungarians-Reassembling" output indexes.

left border of strip 19 is the minimum distance between all other strips distances, which equal (6958).

C. Defining Shreds Sequence

The last remaining process employs the Hungarians reassembling algorithm matches or the assigned pair of shreds to define a sequence of shreds. This sequence will be used to attach shreds and reconstruct the original document. Algorithm 1 (line 8-14) shows how to achieve this. This step run time complexity is approximately O(n^3).

Fig.5 shows the steps of reconstruction a strip shredded document using Hungarians algorithm. This fig. summarizes the process proposed in this research along with the resulting reconstructed document, which was obtained by applying Algorithm 1 (lines 8-14) using the indexes in Fig. 4. This part of the algorithm appends the matched strips in order into a sequence list, then attaches document strips in the space required for the document image. Next section will investigates the performance of these processes.

IV. Experiments and Results

The analytical run time complexity of the Hungarians algorithm reassembling calculated in section 3 was used with different number of shreds. Values obtained from assuming different value of n, which is the number of shreds, and using these values in the run time complexity function resulting from algorithm analysis.
The customized constants (C)= 4.17E-10, defined by the frequency of the processor in local machine, was used in the experiments.

The proposed method was implemented using Python v3.6.3, and PyCharm (JetBrains PyCharm - Community Edition 2017.2.4 x64). Shreds were obtained using shredding function that shreds any document image to an assigned number of shreds. This algorithm takes n (number of shreds) as input, along with a document image and splits the image to n shreds. This is followed by a shuffle process to put each shred in a different random position other than its original position.

TABLE I
Experimental run time of the Hungarians reassembling algorithm measured in seconds for each n shreds

Number of shreds(n)	Run time(second)
25	0.856
50	3.219
70	6.221
100	12.9035
120	18.6241
160	33.1682
200	51.8686
300	120.3616

Fig. 6 shows the run time obtained by the implementation constructed for Hungarians reassembling. Fig. 6 also shows the theoretical time complexity of the proposed algorithm. The results show similar behavior of the run time complexity of the theoretical and experimental analysis.

Table 1 shows a sample result of the running time of Hungarians reassembling algorithm, where time increases as the number of shreds increases.

A benchmark called "Caltech and Pasadena Entrances", provided by Caltech University (http://www.vision.caltech.edu"), containing 86 different images was shredded and reassembled using Hungarians reassembling algorithm. The average accuracy obtained was 0.962. Fig. 7 shows the different accuracy score using the mentioned benchmark.

The Nearest Neighbor reassembling algorithm was formulated in Algorithm 2, although it is not provided explicitly, but it is proposed by [1], [12]. Their algorithm depends on searching each row of the matrix sequentially to find the minimum distance. Then, the resulting minimum values are used to join the shreds.

Algorithm 2: Nearest Neighbor reassembling

input : distance[] calculated in Algorithm 1, document shreds (sh[ ])
output: define pairs of matched shreds depending on minimum distance

```
1  distance[] calculated in Algorithm 1
2  sequence[] ← 0
3  sh[] ← shreds
4  min[] ← 0
5  while counter1 ≤ length(sh) do
6  |   sequence[ ].append (sh[counter1])
7  |   while counter2 ≤ length(sh) do
8  |   |   if distance[counter1][counter2] <
   |   |     min[counter1] then min[counter1] =
   |   |     distance[counter1][counter2];
9  |   end
10 end
```

Algorithm 2 finds the minimum distance in each row of

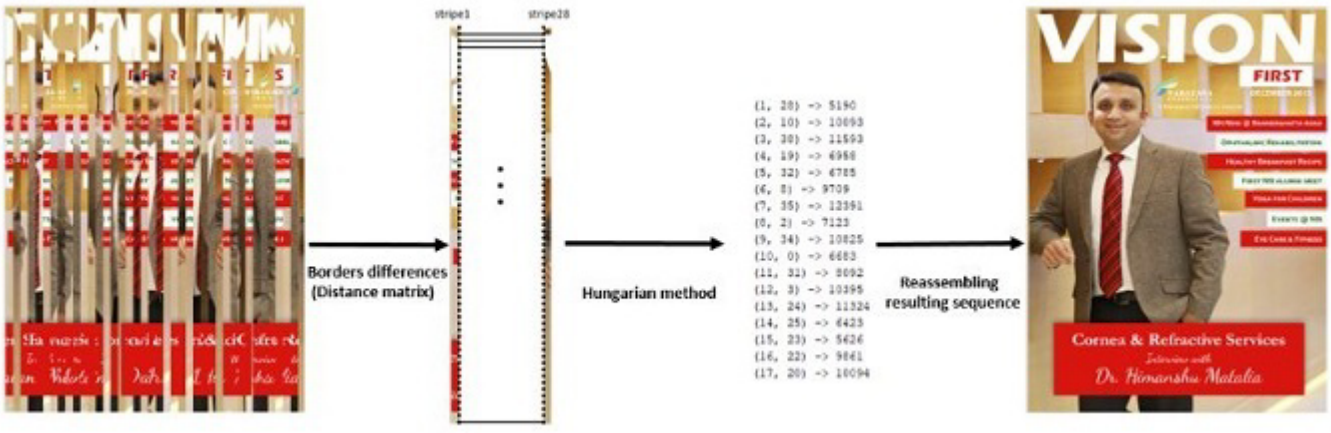

Fig. 5. The reassembled document using Hungarians reassembling algorithm.

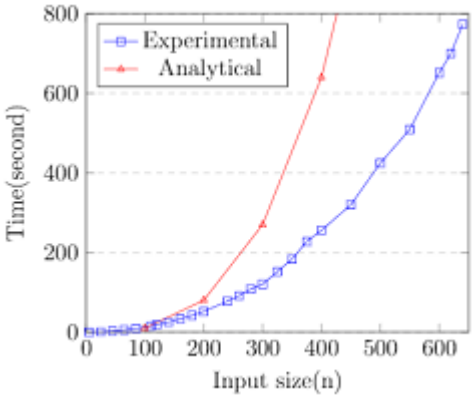

Fig. 6. The analytical run time complexity against the implementation run time of Hungarians reassembling algorithm.

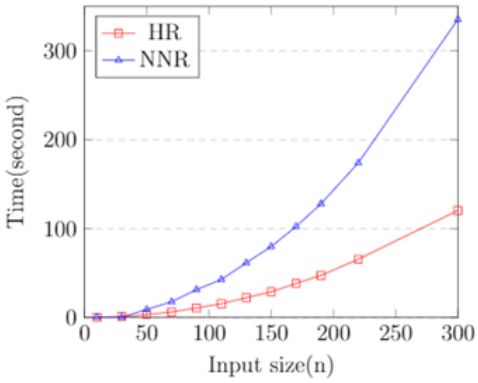

Fig. 7. Nearest Neighbor reassembling (NNR) and Hungarians reassembling (HR) run time.

$n \times n$ matrix that produced by Algorithm1. In every search (each row), the search space decreased by one. When a minimum is assigned as a match, discard the entire column from subsequent minimum searches.

Analyzing the complexity of Algorithm4, the number of searches for minimum equals to $n+(n-1)+(n-2)+.....+1 = \sum_{i=1}^{n} i = (n(n+1))/2 = O(n^2)$. Therefore, adding the first phase of building distance matrix which costs O (n^3), into this phase will results O(n^3). Fig. 8 shows a comparison between the two algorithms: Nearest Neighbor reassembling (NNR) and Hungarians reassembling (HR) in terms of their run time. Since both of them have similar theoretical run time complexity of O(n^3), the shape of cubic function is obvious for both of them. The difference in the constants that was dropped in the HR makes it faster than the NNR.

Accuracy was defined for both algorithms, HR and NNR, by comparing the original document image with the reassembled document results, strip by strip. Accuracy function finds the ratio of the corrected positioned shreds to the number of all shreds. The accuracy depends on the document image and the color distribution of it. Hungarians reassembling shows a more stable performance than the nearest neighbor reassembling. Fig. 9 shows the compared accuracy results after applying both NNR and HR algorithms using different number of shreds (n).

Table II shows both the experimental run time complexity and the accuracy of implementation of both methods: HR and NNR. In general, it is clear that the HR performs better than NNR in both the run time and the accuracy.

V. Conclusion

This paper investigates the power of Hungarians method and its ability to find the best match to provide an algorithmic solution for reassembling colored shredded documents. The algorithm has three phases. The first phase is finding image-based similarity and produces a distance matrix. The matrix defines the distances between the left sides and the right

TABLE II
EXPERIMENTAL RUN TIME OF THE HUNGARIANS REASSEMBLING ALGORITHM MEASURED IN SECONDS FOR EACH N SHREDS

Number of shreds(n)	HR Run time (sec.)	NNR Run time (sec.)	HR Accuracy	NNR Accuracy
50	3.219	9.0668	0.96	0.96
100	15.5388	40.7209	0.95	0.87
150	28.9084	79.8505	0.92	0.84
200	47.3328	127.8999	0.90	0.84
250	65.5816	173.9221	0.88	0.82
300	120.3616	335.283	0.84	0.76

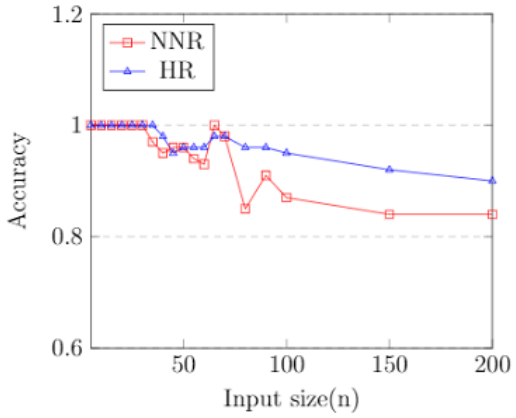

Fig. 8. Accuracy comparison chart between NNR and HR.

side of each strip (shred). In second phase, the Hungarians algorithm was used to match pairs of shreds. Third phase defines the sequence according to the matched strips in order to merge the shreds and reconstructs the document.

The proposed work was compared with the nearest neighbor search algorithm in term of accuracy and run time. The accuracy of the output results by implementing Hungarians reassembling algorithm depends greatly in the image color distribution along with the number of shreds. The proposed algorithm accuracy and run time were evaluated and compared with the Nearest Neighbor reassembling. The proposed algorithm shows better results in terms of run time, and also a more stable accuracy as the number of shreds increases.

Since the matching between each two shreds is an independent process, parallel processing can be applied to the proposed algorithm to get a better speed up. This can be investigated in future work for parallel Hungarians reassembling algorithm.

REFERENCES

[1] Butler, Patrick , Chakraborty, Prithwish ,Ramakrishan, Naren. (2012). The Deshredder: A visual analytic approach to reconstructing shredded documents. IEEE Conference on Visual Analytics Science and Technology 2012, VAST 2012 - Proceedings. 113-122. 10.1109/VAST.2012.6400560.

[2] A. S. Atallah, E. Emary and M. S. El-Mahallawy, "A Step toward Speeding Up Cross-Cut Shredded Document Reconstruction," 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, 2015, pp. 345-349. doi: 10.1109/CSNT.2015.69

[3] Huei-Yung Lin, Wen-Cheng Fan-Chiang, Reconstruction of shredded document based on image feature matching, Expert Systems with Applications, Volume 39, Issue 3, 2012, Pages 3324-3332, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2011.09.019.

[4] Sleit, A., Massad, Y. Musaddaq,An alternative clustering approach for reconstructing cross cut shredded text documents, M. Telecommun Syst (2013) 52: 1491. https://doi.org/10.1007/s11235-011-9626-x

[5] Schauer C., Prandtstetter M., Raidl G.R. (2010) A Memetic Algorithm for Reconstructing Cross-Cut Shredded Text Documents. In: Blesa M.J., Blum C., Raidl G., Roli A., Sampels M. (eds) Hybrid Metaheuristics. HM 2010. Lecture Notes in Computer Science, vol 6373. Springer, Berlin, Heidelberg.

[6] F. Richter, C. X. Ries, N. Cebron and R. Lienhart, "Learning to Reassemble Shredded Documents," in IEEE Transactions on Multimedia, vol. 15, no. 3, pp. 582-593, April 2013. doi: 10.1109/TMM.2012.2235415

[7] Biesinger B., Schauer C., Hu B., Raidl G.R. (2013) Enhancing a Genetic Algorithm with a Solution Archive to Reconstruct Cross Cut Shredded Text Documents. In: Moreno-Díaz R., Pichler F., Quesada-Arencibia A. (eds) Computer Aided Systems Theory - EUROCAST 2013. EUROCAST 2013. Lecture Notes in Computer Science, vol 8111. Springer, Berlin, Heidelberg

[8] J. Perl, M. Diem, F. Kleber and R. Sablatnig, "Strip shredded document reconstruction using optical character recognition," 4th International Conference on Imaging for Crime Detection and Prevention 2011 (ICDP 2011), London, 2011, pp. 1-6. doi: 10.1049/ic.2011.0132

[9] Wong, J.K.: A new implementation of an algorithm for the optimal assignment problem: An improved version of munkres' algorithm. BIT Numerical Mathematics 19(3), 418424 (Sep 1979). https://doi.org/10.1007/BF01930994, https://doi.org/10.1007/BF01930994

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

[11] Pilgrim, R.: Tutorial on implementation of munkres' assignment algorithm (1995)

[12] Marlos A. O. Marques and Cinthia O. A. Freitas. 2009. Reconstructing strip-shredded documents using color as feature matching. In Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09). ACM, New York, NY, USA, 893-894. DOI: https://doi.org/10.1145/1529282.1529475