Design and Implementation of a Paper De-shredder

December 6, 2010

Hei Wang Chan, Evan Gillespie, Delfino Leong

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada

HeiWang@hotmail.com, EvanGillespie1@gmail.com, Delfino.Leong@gmail.com

***Abstract* – A free program, De-Shredder, has been constructed which will reassemble documents which have been shredded. Paper shredding is essentially an insecure cryptographic algorithm, and the De-Shredder project demonstrates this. Documents are shredded to reduce the availability of information, a process which this paper will show to be insecure. Current methods of reassembling shredded documents are not available to most individuals because of time or funding constraints; by releasing De-Shredder freely, any dedicated person has access to shredded information. The De-Shredder reassembly algorithm has been validated by reconstructing several single documents. Recommendations are included to improve the software for more practical use.**

I. Introduction

Many critical documents are not stored electronically, but are instead printed onto paper. Often this paper is shredded as a means of destruction. Paper shredding is a weak cryptographic operation, similar to a block cipher. 'Blocks' of paper are rearranged, but not actually altered. Security in shredded paper is achieved by obscurity, which is not secure at all. Paper shredding is a rare case of computer security which does not live entirely in the virtual world.

In many cases, incriminating documents are shredded by corporate criminals. Legitimate authorities which have claim to the information cannot obtain it. Thus, availability of essential information is compromised. Although there exists services to reconstruct shredded documents, all are very expensive or difficult, making them infeasible to all but the most dedicated individuals. Paper shredding is a practice which the populous incorrectly considers to be secure because of the current difficulty in reversing the act of shredding.

Although the task of rearranging 'blocks' of ciphertext to produce plaintext is possible by hand, it should ideally be automated using a computer. Using software, it is possible to reassemble a shredded document from only the scanned strips of paper which have come out of a shredder. We propose to build a first version of such a piece of software and distribute it freely to anyone who is interested. This software will prove the feasibility of an algorithm, but will not yet be practical for all reverse-shredding purposes in the scope of this project.

II. Related Work

A service currently exists, Unshredder, which offers to rebuild shredded documents. They cite themselves as “the first commercial document reconstruction tool in the world.”[1] A subscription to the Unshredder service costs thousands of dollars per year, will only reconstruct a limited number of documents, and requires special hardware which is only available at addition cost[1]. The De-Shredder project is in a different space than Unshredder because it is a freely available, “open-source” program.

Several guides show readers how to assemble shredded documents by hand[2]. Currently, this

is the most viable solution to assemble a shredder piece of paper. However, this method is inconvenient, very slow, and is limited to only a few pieces of paper.

Photo stitching is a field of software, which is similar to document reconstruction. Examples include Autostitch and CleVR[3]. These programs combine many smaller photos into a large panorama. Photo stitching is similar to shredded document reconstruction in that many smaller pieces are combined to make a larger image[4]. The major difficulty in reassembling shredded documents is that no overlap exists in the images, as it does when combining photos.

Finally, a project is currently underway in Germany to reconstruct intelligence documents which were shredded by the secret police during the final days of the communist regime in East Berlin[5]. This task is much greater than any reconstruction ever attempted with hundreds of thousands of papers being reassembled. The project in Nuremberg is similar to the De-shredder project, but differs greatly in budget, scale, and availability to the public.

III. Our Solution

A. Overview

The program should ideally require little or no human intervention and have a simple user interface. The proposed solution is summarized in Figure 1. The user first scans the shredded strips into the computer on top of a coloured background. A MatLab script then processes the image to isolate each strip and align them vertically. The edges of all the strips are matched in our program, and the matched pairs are displayed for verification. Finally, the assembled image is displayed to the user. The individual steps are described in detail below.

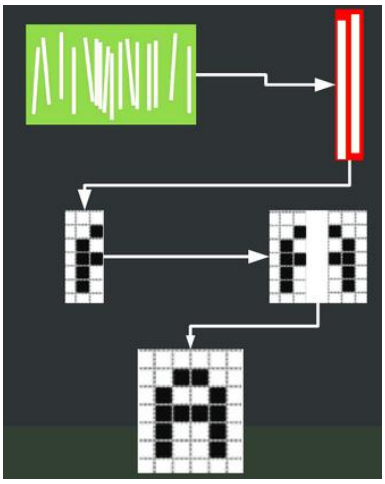


FIG 1. Illustration of the main steps in re-assembling the shredded document

B. Image processing

Once the strips are scanned into the computer, a MatLab script is used to process the image. The contrast of the initial image is first increased to amplify the differences between black text, white paper, and the red background. Using colour thresholding techniques, we assign each pixel as a red, white, or black pixel. The general contours of the strips are then given by the union of the black and white pixels. Next, we label each region (strip) using a MatLab function called bwlabel[6].This function calculates the centroid and the relative lengths of the major and minor axes, which allows each strip to be rotated and aligned vertically. Since the strip is not a perfect rectangle, a minimum bounding box is used to enclose and crop the strip from the larger image. The imaging processing script iterates through the same process for all the remaining strips, resulting in individual strip images which are all aligned vertically, and composed of only white, black, and red pixels. Figure 2 is a screenshot of the strip images after processing.

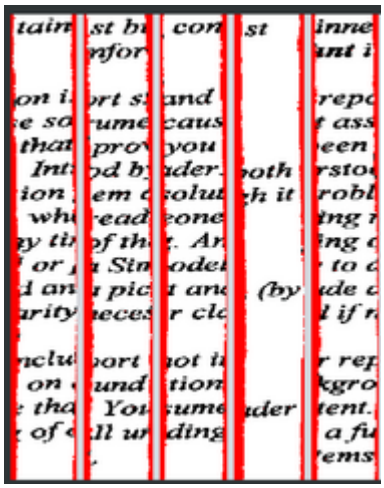


FIG 2. Processed strips

C. Matching Initialization

After the strips are created and processed by the MatLab scripts, the individual strips are fed into the program for matching. The matching algorithm first creates two arrays of pixels for each strip, one for each the left and right edge. A left edge pixel is defined to be the left most non-red pixel in the strip image, and vice versa for the right edge. The edge pixels for the length of each strip form the basis of the matching algorithm, and are used to compare with other strips to find the best possible matches.

D. Matching Algorithm

The matching algorithm initially compares each edge pixel array directly with the corresponding edge pixel of another strip. It looks for matching pairs of pixels between each strip. For example, if an edge array has a black pixel at location index 'i', and another strip also has a black pixel at location index 'i', these two strip's matching coefficient will increase. The increase in the matching coefficient also depends on which colour is matched. If an edge is mostly white, the matching of two black pixels would increase the matching coefficient more than the matching of two white pixels. The pixel comparison is repeated for the entire length of the edge. One of the strips is then offset by one pixel, up or down, and compared again. The final matching coefficient used is the highest calculated coefficient after applying every offset within +/- 0.25% of the total strip length. For a typical letter head sheet of paper, this is an offset of up to ± 4 pixels. This offset helps to alleviate any errors created during the scanning process or during our earlier image processing. This entire process is repeated for each strip. In this manner, the right edge of each strip is compared with the left edge of every other strip to produce a two dimensional matrix of matching coefficients.

Figure 3 shows pairs of strips matched by this algorithm:

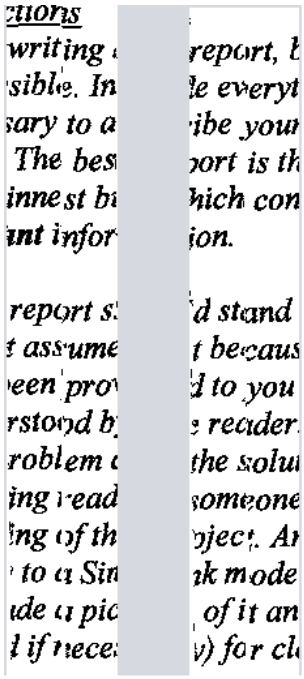


FIG 3. Matched pairs of scanned strips.

It can be seen from the figure that the algorithm provides fairly accurate results for the matched strip pairs. It is possible to read off text and predict words in the document without much difficulty or ambiguity.

E. User Input and Edge Strips

Next, our program will prompt the user to specify the strips corresponding to the left and right edge of the document. This step is necessary to reconstruct the document as it is

highly difficult to create the final image without an initial point of reference. With the user specifying the edge strips, we can gradually build the final document by recursively appending the strip with the highest matching coefficient to its predecessor, starting with an edge strip.

Although our algorithm provides fairly accurate results for most matched strip pairs, it is very difficult to achieve perfect matching for all pair of strips. De-shredder compensates for this by allowing the user to mark a pair of strips as incorrect. Upon refreshing, the program will display the next best matched pair for any incorrect strips.

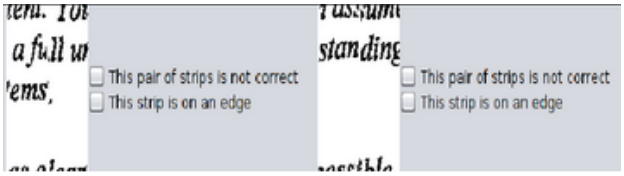
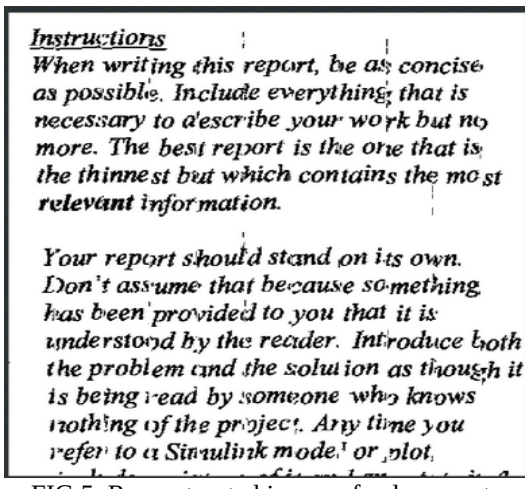


FIG 4. User inputs to verify the matched pairs.

Once the user is satisfied with each pair of strips, he or she may view the final image of the assembled document. Figure 5 shows the final image of a document reconstructed using De-Shredder. As shown in the figure, the user can easily read and extract the information from the document.


Instructions
When writing this report, be as concise as possible. Include everything that is necessary to describe your work but no more. The best report is the one that is the thinnest but which contains the most relevant information.

Your report should stand on its own. Don't assume that because something has been provided to you that it is understood by the reader. Introduce both the problem and the solution as though it is being read by someone who knows nothing of the project. Any time you refer to a Simulink model or plot,
FIG 5. Reconstructed image of a document.

IV. Discussion

De-Shredder manages to piece together shredded documents which have been scanned using the method described above. The matching algorithm has been verified, but the software requires a lot more work to become practical. Some of the current issues with De-shredder are discussed below.

A. Human input

The system currently requires a lot of human input to reassemble the trickiest documents. The human user must repeatedly select which strips which have been matched improperly. The user must carefully examine each matched pair to verify that the strips match; this can take up to 30 seconds for a single pair of strips. On difficult documents the user must perform this check 10-20 times every iteration, for up to 15 iterations. The human verification presents a significant obstacle limiting De-Shredder from being practical for final users. To solve this problem, the algorithm used to compare the strip edges and to arrange the compared strips must be improved.

B. Improving the Matching Algorithm

Practical applications often ask for more robust document reconstruction than De-Shredder currently offers. As noted above, De-shredder currently only uses the direct comparison method to match edges of the strips. We may improve the matching algorithm by comparing the derivatives (slopes) of the black and white lines near the strip edge. It may also be possible to use the curvature of the edges to improve the accuracy of the system. Furthermore, De-shredder could also employ optical character recognition (OCR) to validate the matched pairs before displaying the result to the user. However, this would only be feasible for text-based documents.

C. Improving Strip Arrangement

The strips are currently arranged using the algorithm outlined above and requires the user to specify the edge strips. The problem of arranging strips without a point of reference is similar to the "Travelling Salesman Problem" in mathematics, which has been studied extensively and found to be NP-Hard [7]. After improving the algorithm with the suggestions above, there should be less user input required.

D. Cross Shredding

Some documents are shredded using a "cross-shredder." Cross-shredders cut the paper in two directions. De-Shredder should be able to compare and match all four sides of each paper shred to accommodate for cross shredding. In order to do this, all four sides of each strip need to be compared. The algorithm outlined above would be adequate for comparison and matching if extended.

E. Multiple Pages

Most applications require reconstruction of many documents from many strips of paper. De-Shredder cannot yet accomplish this task and only works for single sheets. The same underlying algorithm can be used to reconstruct multiple documents as are currently used for a single document. One of the most important additions for De-Shredder to be practical is the capability to reconstruct multiple pages and to be able to separate individual pages from each other.

F. Missing Pieces

Often, a user will not have access to every shred of paper in order to reassemble a document. Currently, this will cause produce a single document with a discontinuity, which should remain readable by the astute user. To solve the problem of missing strips, De-Shredder should produce distinct sections of a reconstructed document. The missing piece problem would be solved by accommodating for multiple documents as above.

VI. Conclusion

De-Shredder, a free program to reconstruct shredded documents, has been produced. The software is currently in a first version and is able to reconstruct the documents, but more advanced matching algorithm must be implemented to make the reconstruction procedure more practical for the end-user.

Acknowledgement

We would like to thank Professor Konstantin Beznosov from the Electrical and Computer Engineering department at UBC for providing us with reading material about the many security concepts involved in creating and deciphering secure systems.

References

[1] "Unshredder", *Safe Guard Ltd*. (2010) Available at: http://www.unshredder.com/
[2] "How to Reconstruct Shredded Documents", eHow Journal (2009) Available at: http://www.ehow.com/how_4768399_reconstruct-shredded-documents.html
[3] "Clevr Photo Stitching" (2010), Available at http://www.clevr.com/
[4] M Brown, D Lowe "Automatic Panoramic Image Stitching using Invariant Features" University of British Columbia, 2007. Avaliable at http://cvlab.epfl.ch/~brown/papers/ijcv2007.pdf
[5] "Germany's Effort to Stitch Together Millions of Shredded Secret Documents", *The Wall Street Journal (2007, May)* Avaliable at: http://blogs.wsj.com/numbersguy/germanys-effort-to-stitch-together-millions-of-shredded-secret-documents-103/
[6] "MatLab Image Processing Functions" *MathWorks Inc.* Avaliable at: http://www.mathworks.com/help/toolbox/images/ref/regionprops.html
[7] G Dantzig, R Fulkerson, and S Kohnson. "Solution of a Large-Scale Travelling-Salesmen Problem" The Rand Corporation, Santa Monica, 1954.