Software Report
and
Specifications

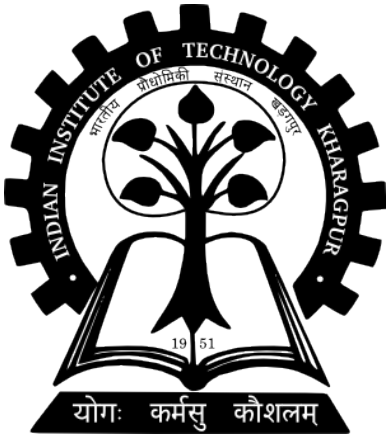
(intentionally left blank)

Automated Mosaicing of Torn Paper Documents

by

TEAM 10

Indian Institute of Technology Kharagpur



IIT Kharagpur

(intentionally left blank)

Abstract

Automated torn paper mosaicing, is a well known problem that arises in the field of investigation science and archival studies. The papers can either be shredded using a machine or manually ripped up. The latter one is often more challenging than the previous one because of complex edge features and introduction of lot of noise. The problem statement required us to approach in a very careful way with precise yet fast collection of important image features, finding the best matches and then properly aligning the images along the matches borders. Large amount of our work is based on research already done in this field. In this document we discuss the algorithm, performance, limitations and future scope of the developed tool.

References:

1. *Reconstruction of Torn Documents Using Contour Maps, Arindam Biswas, Partha Bhowmick*
2. *Reconstructing shredded documents through feature matching, Edson Justinoa, Luiz S. Oliveira , Cinthia Freitas*
3. *Edge Envelope based reconstruction of torn documents, S. A. Santosh Kumar and B. K. Shreyamsha Kumar*
4. *Image Alignment and Stitching: A Tutorial, Richard Szeliski*

Contents

Problem Definition

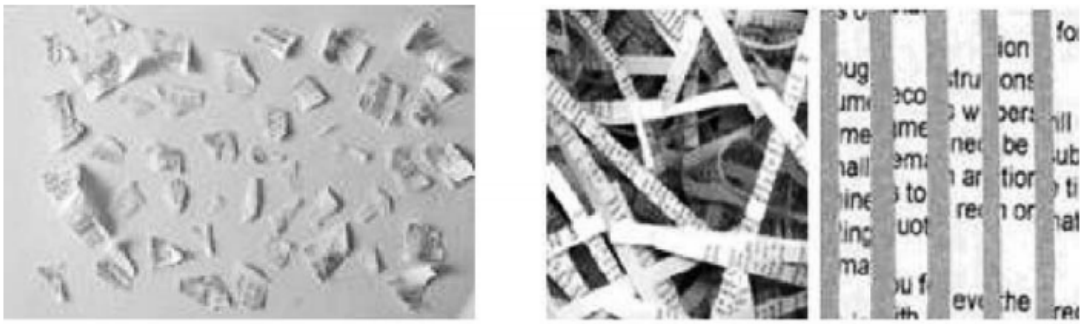
The official problem statement for the Opensoft competition was given as:

The goal is to seamlessly stitch randomly torn paper documents in an automated fashion. The state of the art can handle up to 50 torn pieces. The challenge is to come up with an innovative algorithm that could potentially handle more pieces, or implement existing algorithms in a faster way. The training dataset would be provided, and the test dataset would be used to evaluate the entries. The training dataset would contain the original document as well as the torn pieces.

The reconstruction of a ripped up document is a puzzling, complex and painstaking task to be performed by a human operator. Often, in case of documents with important historical value, we can not directly handle the papers every time. This makes the problem not only difficult but also intractable in many cases just for a fewer number of fragment pieces. Torn papers are more difficult to analyse because of complex edge features, shear on surface thereby causing loss of crucial data. Indeed, it requires innovative thinking, careful implementation and execution besides the hard work involved in actually writing the program. Novel techniques have been developed in this field which are suitable for a particular type of fragmented images. The problem thus, is open end with much scope for further improvement.

The problem statement does not mentions specific constraints related to target system. The final deliverables thus are program with the supporting report.

—



(a) (b)

Fig. 1. Different kinds of shredding.

In case of shredding of papers, each edge has more or less straight edge and hence the problem may be considered as a variety of Jigsaw puzzle. Torn pieces on the other hand have uneven edges bearing undesired shears (containing paper fibres) which give rise to ill-defined physical contours compared to jigsaw puzzle which have smooth and well defined corners. It was therefore necessary to look closer into unique features of each fragmented piece and employ a novel technique for fast and easy reconstruction from several such pieces.

Fundamental Steps

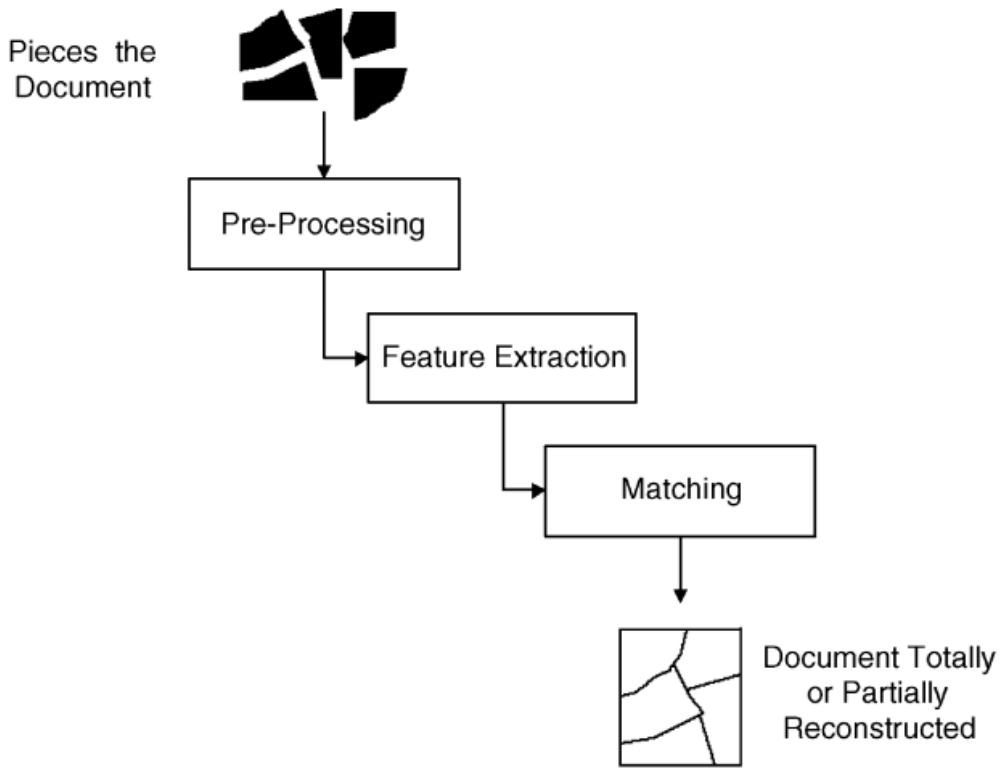


Fig. 2. The block diagram of the proposed methodology.

Detailed Solution

Employed Technique

Our main source code is organized in several layers where the input images are processed one by one and passed on from one layer to the other for further processing.

- Pre-processing:

We have no previous information about the nature of input images whether they are colored, black and white, pieces of a newspaper or entirely a painting. Pre processing thus become an important part of our algorithm. We first load the input image into GRAYSCALE mode and apply thresholding to obtain a binary image. This binary image is then processed by CannyDetector which takes out the largest closed contour (our actual image) and discards all the other contours.

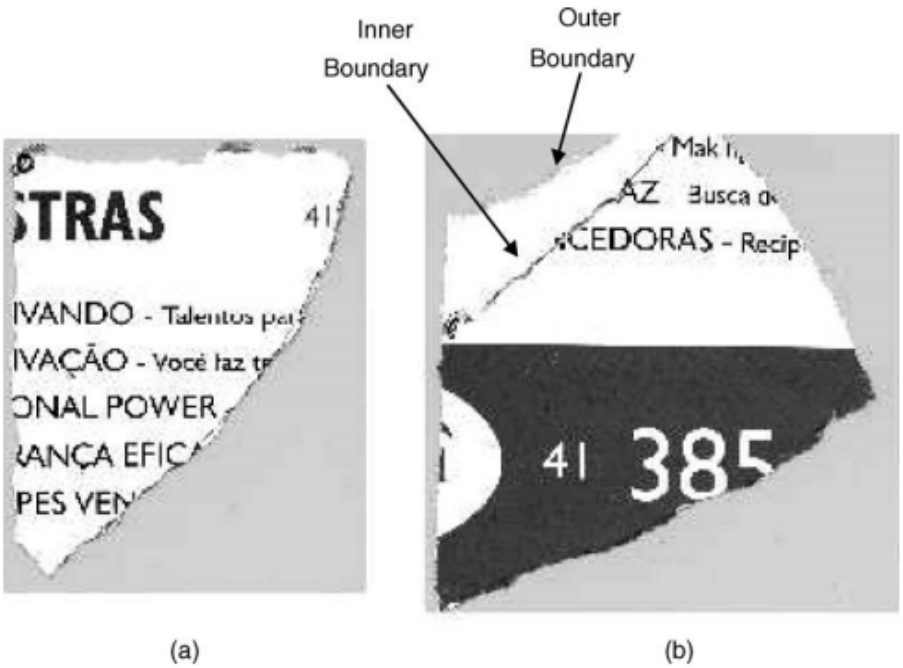


Fig. 3. Inner and outer boundaries produced by shredding.

- Corner Detection:

The contour obtained from previous step is approximated to a nearest polynomial by a particular threshold determined with trial and error technique. This would return us an array of small number of corners.

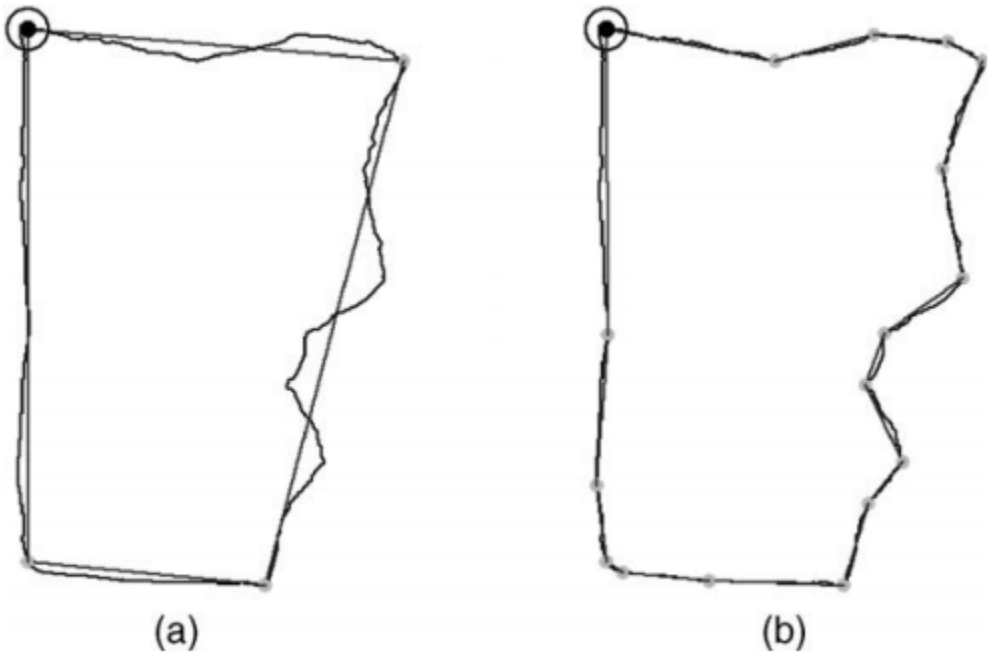


Fig. 4. Inner and outer boundaries produced by shredding.

- Feature Extraction:

To understand which Torn piece matches with the other one, we extracted unique features of every piece by calculating three values for each corner point - next corner point, previous corner point and the angle it forms with the two. A particular convention (clockwise or anticlockwise) is followed all along.

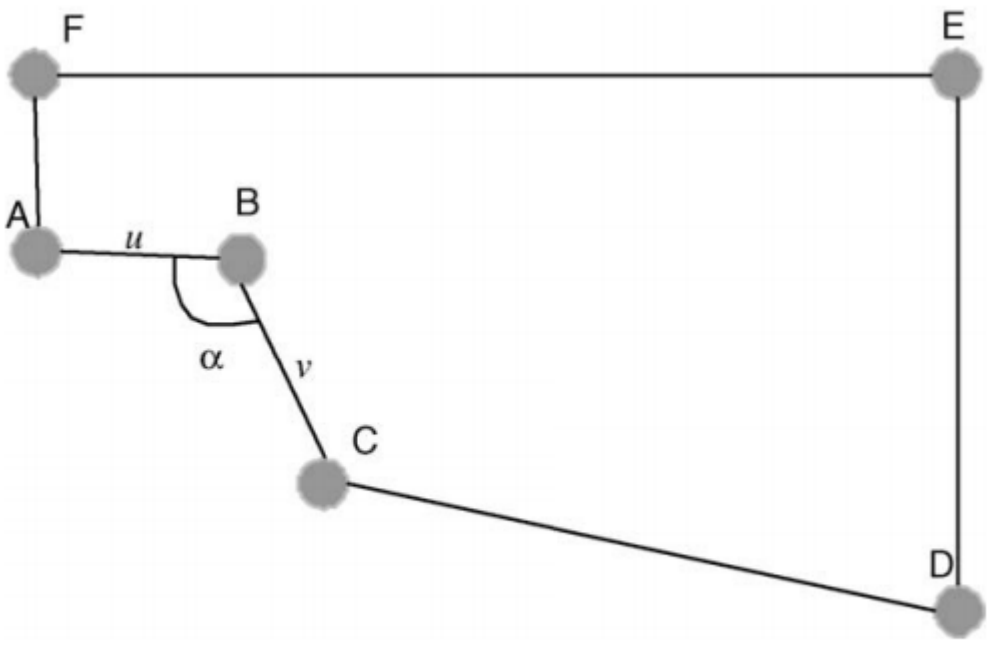


Fig. 5. Angle features extracted from the polygon.

- Finding Best Match:

After obtaining the features for each Torn piece, we find the best possible match among all the pieces based upon these features. Two fragments match the best if for some pair of vertices on the fragments, the sum of their angles is close to 360 and arm lengths are close to each other. These matchings are weighted according to both of these criteria and used for stitching purposes. This process is repeated many times to get the final image.

- Stitching Images:

Best matching images obtained from the previous step are stitched using appropriate ROTATION and TRANSLATION transforms.

The new image obtained is again sent back to the original image pool and the whole process is repeated until all the images are stitched with at least one other image. The image finally obtained is our final

reconstructed image.

System Requirements

The program was tested on Linux based platform running Ubuntu 12.04 LTS. The program must meet following requirements to run successfully:

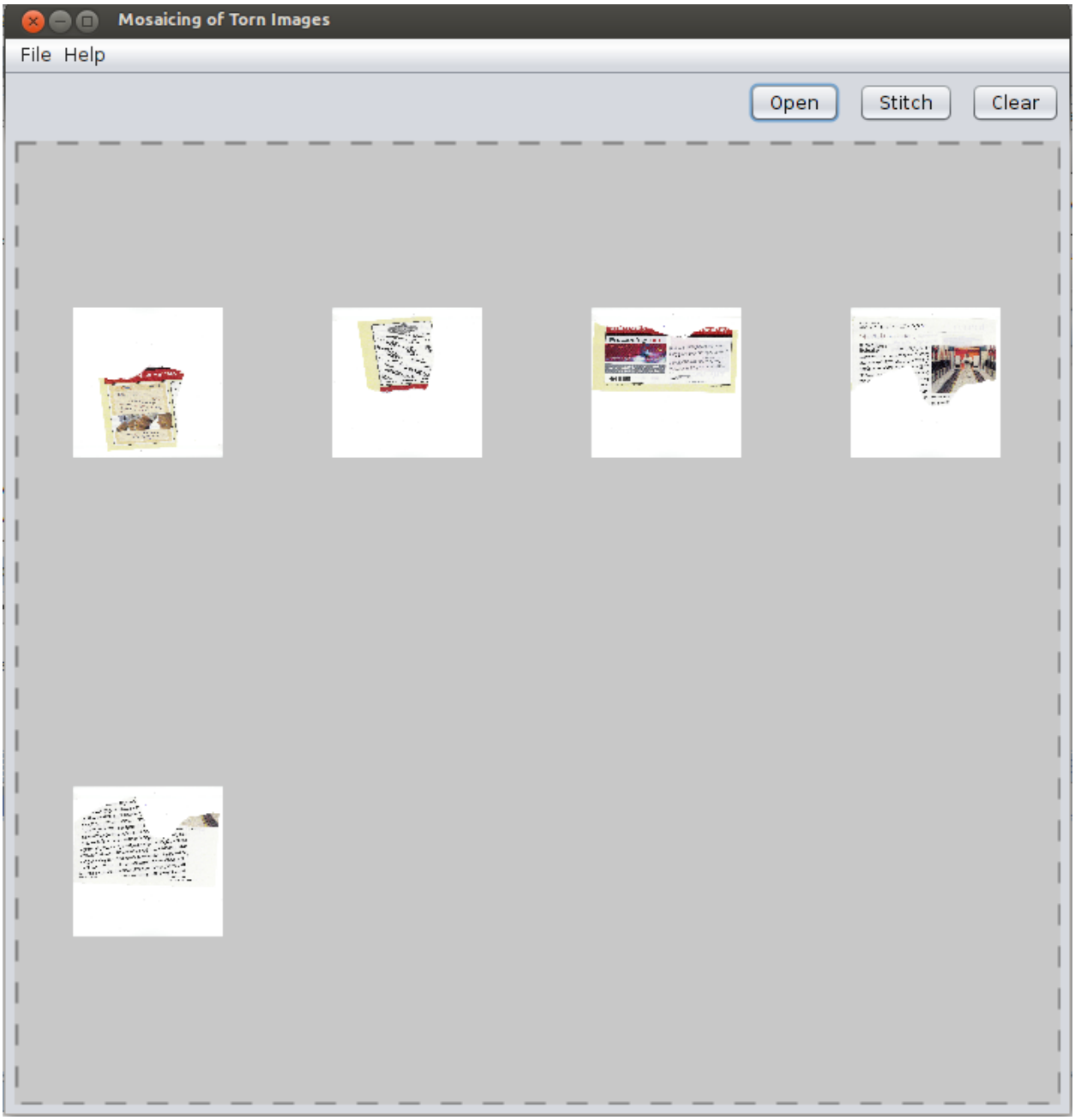
- OpenCV v2.4.0 or higher
- Java Runtime Environment v7.0.25u or higher

The hardware must have at least:

- 1.5 Ghz Dual Core or latest processor.
- 100 MB of free Hard Disk Space.

Installation

To run the program Double click the file called TEAM10.jar. If you have specified requirements already met, the following screen would show up :



(In this example, test images are already loaded)

If you want to run the program from command line, open your bash or provided terminal and run :

home@ubuntu~: java -jar TEAM10.jar

Select the input images by Dragging and dropping them directly on the Panel or by manually selecting them through the open button. To obtain the final image, click Stitch.

Please be patient while the final image loads. It is possible that the you may see nothing for few minutes while the program is still working in background. However if you are unsure, stop the program, clear the images using Clear button and then try again.

Testing and Validation

The program was tested rigorously across variety of test data and promising results were obtained. The execution time was largely dependent on number and resolution of input files. The program performs efficiently for over 10 input images. Few sample test results are given below.

INPUT

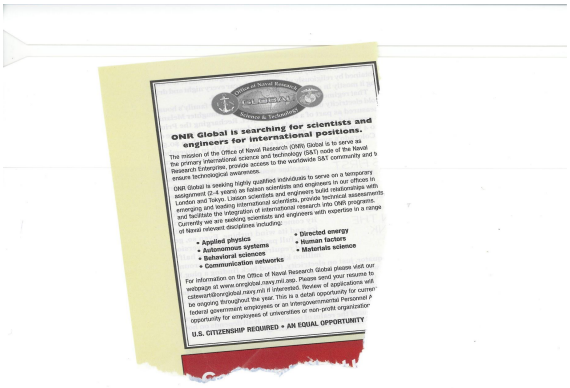


Image 1

Image 2

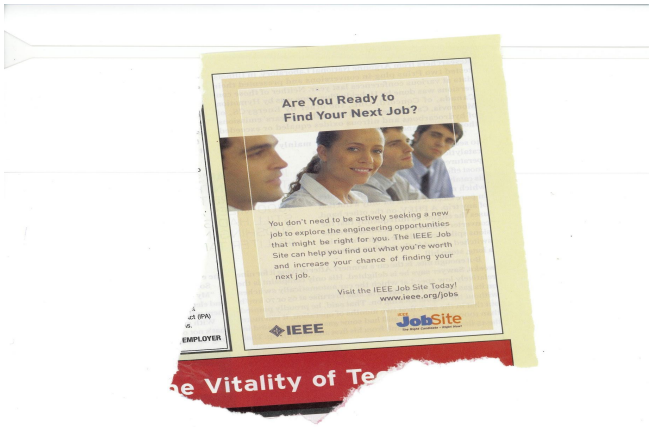


Image 3

OUTPUT


Previous
Next
Are You Ready to Find Your Next Job?
Proceedings of the IEEE
No other publication keeps you in touch with the evolving world of technology better than the Proceedings of the IEEE.
IEEE
1368 × 1805 pixels 906.7 kB 38%
9 / 28

Appendix

—

Unlike the tradition of previous years, we have decided to release all the source code along with the problem statement and documentation in Public Domain and will be hosted on https://github.com.

We also encourage Open source development and the idea of use of Version Control Systems in their active collaborative development.

We thank the problem setters, Hall members, our friends and fellow scientific community for providing us with constant encouragement and knowledge to learn and explore.