

# 1. Abstract

Paper documents are often shredded in an attempt to destroy the information they contain. For forensic and investigative scientists, it is an important task to reconstruct documents for use as evidence. This can be a very time consuming and laborious undertaking if attempted by hand.

This project aims to develop an automated system for the reconstruction of shredded paper documents to make the process faster and easier. The performance of this system will be compared to existing solutions.

# **Shredded document reconstruction**

## **Final report**

Joe Dinn 1724858

CM3203 One Semester Individual Project

Supervisor: Dr Richard Booth

Moderator: Víctor Gutierrez Basulto

## Table of Contents

1. Abstract.....	1
2 Introduction.....	4
2.1 Aims.....	4
3 Background.....	5
3.1 Shredding techniques.....	5
3.2 Context.....	5
3.3 Existing work.....	9
3.4 Specialist Libraries.....	11
4 Specification and design.....	12
4.1 Requirements.....	12
4.2 Architecture.....	12
4.3 Design.....	12
4.4 Previous designs.....	15
5 Implementation.....	17
5.1 Input.....	17
5.2 Data structures and utility functions.....	18
5.3 Discrepancy function.....	19
5.4 Optimiser.....	21
5.5 Output.....	23
6 Evaluation.....	25
6.1 Dataset.....	25
6.1 Metric.....	25
6.3 Methodology.....	26
6.3 Results.....	26
6.3 Comparison with older designs.....	27
6.4 Comparison with existing designs.....	28
6.5 Discussion.....	28
7 Future work.....	29
8 Conclusion.....	31
9 Reflections.....	32
10 References.....	33

## 2 Introduction

Despite the increasing reliance on computers, paper documents still see widespread use. The EPA states that the average U.S. office worker uses 10,000 sheets of copy(printer) paper every year [3]. If a paper document contains sensitive information and is no longer needed, it is common practice to destroy it using a shredder. Reconstructing documents that have been shredded allows for the recovery of the information contained within them. This could be to recover documents that were accidentally lost, or recover documents that were intentionally destroyed. This second application is especially useful in forensic and investigative sciences, as well as for intelligence agencies, for the acquisition of evidence.

Manual reconstruction of shredded documents is possible; shredded documents seized from the American embassy during the Iranian hostage crisis were reconstructed by a team of carpet weavers[1]. This task, however, is time consuming and difficult. The number of different ways to arrange a collection of fragments grows exponentially with the size of the collection. This means that even with only a fairly small number of pieces, there quickly becomes a prohibitively large number of possibilities to try. An automatic document reconstruction system would be desirable to reduce the time and effort required.

### 2.1 Aims

The primary aim of this project is to develop an automatic system to reconstruct a document given a collection of fragments. These fragments will be taken as input in the form of image files produced by a scanner. The output should be an arrangement of the fragments which resembles the original document.

A number of assumptions are made about the problem to reduce the complexity, however these may be relaxed as an extension to the project if it proves too simple. The assumptions are:

- **Vertical strip shredding.** The fragments are created using a strip shredder (see section 3.1) and are vertical.
- **Text based documents.** The content of the original document includes text.
- **Latin alphabet.** The documents are written in a language that uses the Latin alphabet.
- **Completeness.** The collection of fragments is sufficient to complete the original document and does not contain fragments belonging to another document.

In addition, It is desirable that the performance of the system be competitive with other existing solutions. This will be assessed by measuring the effectiveness of this and other systems on the same dataset and comparing the results.

# 3 Background

## 3.1 Shredding techniques

There are three most common methods of document shredding: manual, strip and cross-cut. Manual shredding is tearing of documents by hand. This leads to irregular shapes often with jagged boundaries. Strip and cross-cut shredding are both techniques performed by a mechanical shredder. Strip shredders cut documents into thin vertical ribbons. Cross cut shredders also make horizontal or diagonal cuts leading to smaller fragments. These mechanical approaches produce more regularly shaped fragments.

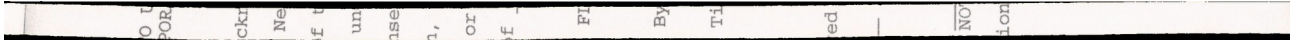


Figure 1: A scanned fragment from a strip shredder (rotated to fit the page). Strips are largely rectangular but there is often some distortion.

## 3.2 Context

In this section a brief explanation of a few important concepts is provided as reference.

### 3.2.1 Combinatorial optimisation

Combinatorial optimisation is a branch of computer science that deals with finding the best solution among a set of candidates[4]. The set of candidates is often prohibitively large to search exhaustively, so heuristic based methods are used that aim to give a good, but not necessarily optimal, solution.

#### **Minimum-weight Hamiltonian path(MWHP) problem**

Given a graph, a Hamiltonian path is one that goes through each node exactly once. This is a common combinatorial optimisation problem. If the graph is weighted, this problem can be extended to the MWHP problem, which finds the Hamiltonian path with the smallest total weight. Similar to this is the travelling salesman problem which finds the minimum weight tour[4].

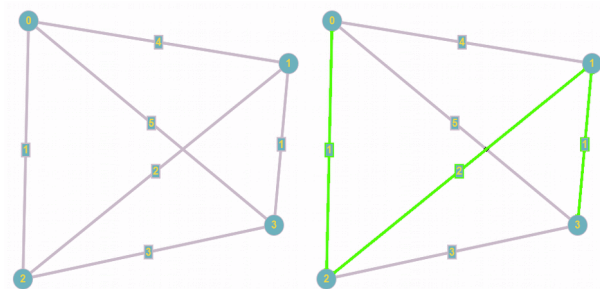


Figure 2: A weighted graph with its corresponding minimum weight Hamiltonian path.

#### **Nearest neighbour algorithm**

This greedy algorithm was designed to give good solutions for the travelling salesman problem[4], but is trivially converted for the MWHP problem. The algorithm starts by

choosing an initial node. In contrast to the travelling salesman problem, the choice of the initial node when constructing a MWHP has a large impact on the quality of the solution found. The algorithm then iteratively adds the unvisited node with the lowest weight edge joining it to the last added one. The algorithm runs in only  $O(n^2)$  but does not guarantee an optimal solution.

```

Input: Weighted graph
Output: minimum weight Hamiltonian path

candidate nodes = all of the nodes in the graph
path = empty list
select initial node
add initial node to path
remove initial node from candidate nodes
while (length(candidate nodes) > 0)
    next node = candidate node with smallest weight edge to last node in path
    add next node to path
    remove next node from candidate nodes

```

Figure 3: Pseudo code for the nearest neighbour algorithm.

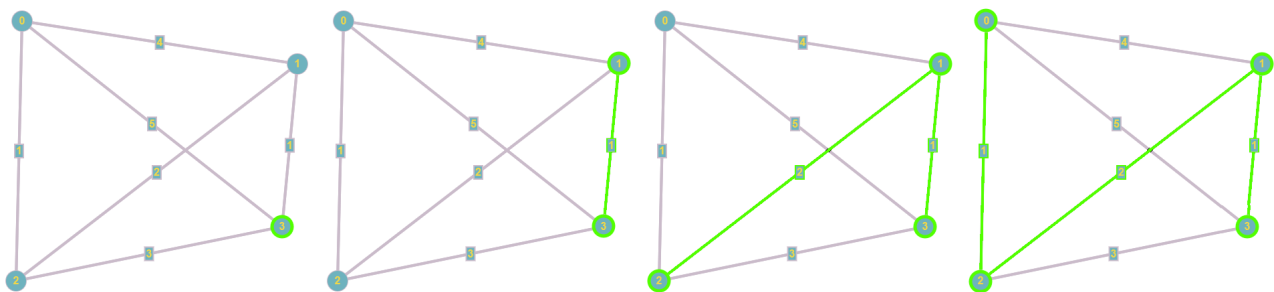


Figure 4: The nearest neighbour algorithm constructing a solution for the graph in Figure 5. Here node 3 was chosen as the start.

## Local search

Local search is a simple heuristic for finding good solutions in a variety of combinatorial optimisation problems[4]. The algorithm is initially provided with a single initial solution. A collection of similar solutions is constructed called the solution's neighbourhood. The best solution in this neighbourhood is chosen to be the next solution. This process is repeated until a solution is found which is the best in its own neighbourhood. This process often gets stuck in local optima because it has no mechanism for exploring worse solutions.

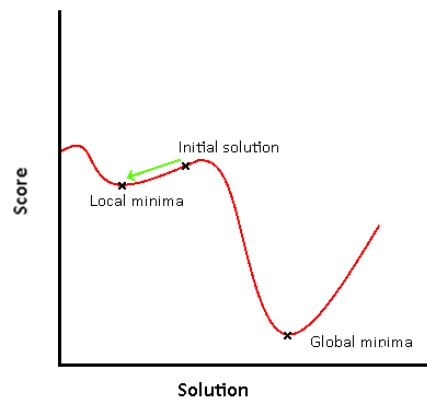


Figure 5: If the local search starts as shown on the diagram, it will go to the local minima.

### **Tabu search**

Tabu search is a metaheuristic that extends upon local search[9]. Like local search, it selects the best solution in the current neighbourhood at each iteration, however it is allowed to select a worse solution if no better ones are available. To prevent the next iteration from returning to the previous, better, solution a memory structure is maintained which stores recent moves. Moves within this structure, called the Tabu list, are forbidden. After a set number of iterations known as the Tabu tenure, the moves are removed. The search ends when a stopping criterion is met. Many of the specifics of the implementation, such as the neighbourhood and stopping criteria, are left to be customised for the specific problem.

Input: initial solution

Output: best solution found

```

best solution = initial solution
current solution = initial solution
tabu list = []
while(stopping criteria not met)
    current neighbourhood = neighbourhood(current solution)
    best neighbour = best solution in current neighbourhood and not in tabu list
    current solution = best neighbour
    if (best neighbour is better than best solution):
        best solution = best neighbour
    push best neighbour to tabu list
    if length (tabu list) > tabu tenure
        remove first item from tabu list
return best solution

```

Figure 6: Pseudo code for Tabu search

### ***Assignment problem and Hungarian algorithm***

The assignment problem is a common combinatorial optimisation problem. Given two sets of equal size, each item in the first set must be matched with a unique one in the second. Each possible pairing is given a weight. The aim is to find an assignment that minimises the total weight.

The Hungarian algorithm is a technique that gives the global optima of an assignment problem in polynomial time[12].

### **3.2.2 Computer vision**

Computer vision is a branch of computer science that deals with manipulation and extraction of information from digital images.

#### ***Otsu's method***

Otsu's method is a technique for finding a threshold value for automatic thresholding. The method starts by constructing a pixel intensity histogram of the target image. For each possible threshold value, the histogram is divided into intensities above and below the value. The inter-class variance between the divisions is then found. The threshold value which creates the divisions which maximises the inter-class variance is chosen. The target image is then thresholded with this value.

```
Input: Image to be threshold
Output: Best intensity to threshold at

build intensity histogram
iterate through all possible thresholds
    compute inter-class variance
    update current best inter-class variance
return threshold that gives best inter-class variance
```

*Illustration 1: Otsu's method for finding a threshold value*

#### ***Floodfill***

This is a technique for visiting every pixel in a connected component. The algorithm starts with a seed coordinate and expands iteratively into each adjacent unvisited pixel in the component. This can be used to find a bounding box of the component by storing values for the left, right, top and bottommost pixels visited so far and updating them each time a new pixel is reached.



```

Input: seed location inside component
output: bounding box of component

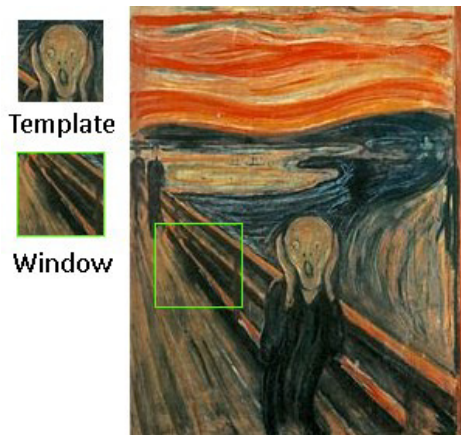
initialise leftmost, rightmost, topmost, bottommost = seed location
Q = empty queue
enqueue seed location to Q.
while Q is not empty:
    set (x,y) equal to the first element of Q
    dequeue first element from Q.
    if (x,y) in component and unvisited:
        mark (x,y) as visited
        update leftmost, rightmost, topmost, bottommost
        enqueue (x+1,y) to Q
        enqueue (x-1,y) to Q
        enqueue (x,y+1) to Q
        enqueue (x,y-1) to Q
return leftmost, rightmost, topmost, bottommost

```

*Figure 7: Pseudo code for using floodfill to find a bounding box.*

### **Template matching**

Template matching is a technique for finding one image, the template, within another, the source[2]. A window is moved around the source image. At each point, the content in the window is compared to the template to give a score. Various methods for comparison exist, the simplest being the sum of absolute pixel differences. By taking the window location that gives the best score, the most likely match within the image can be identified. The value of the best score can also be used to give an indication whether the template image is within the source image at all.



*Figure 8: Template matching.*

### **3.3 Existing work**

A number of papers have been published investigating this problem. However none demonstrate perfect accuracy. Related work often treats the problem of document reconstruction as a combinatorial optimisation problem. This is the approach that this project also takes and is explained more thoroughly in section 4.3.1. This approach is commonly divided into two subproblems:

- A method for scoring an arrangement of fragments. This is achieved by assessing the compatibility between each pair of adjacent fragments in the arrangement.

- A method for finding the arrangement of the fragments which gives the optimal evaluated score.

## Compatibility measures

A variety of methods for evaluating the compatibility or discrepancy between strips have been proposed:

Pimenta et al. [19] and Naiman et al. [14] both use measures that try to find the fragment shapes that fit together best. They are mostly concerned with hand torn documents and have little application to mechanically shredded documents because the shape of the fragments is too regular.

Alhaj et al. [5] Use the absolute edge pixel difference. At each row, the absolute difference is calculated between the pixel at the right edge of the left fragment and the pixel at the left edge of the right fragment. The sum of these differences is the score, a high value indicating a poor match and 0 representing a 'perfect' match i.e. identical edges. It is worth noting that this paper uses coloured documents so the difference between two pixels is the Euclidean distance between the RGB vectors.

Chen et al. [7] Also use the edge difference but with some modification. Each entire edge column is treated as a vector and the Euclidean distance between the fragments' corresponding edges is taken. The paper also introduces a penalty coefficient which is inversely proportional to the combined total sum of the edge pixel values. This is intended to reduce the impact of loss of data which might cause a pair of fragments to appear more similar than they should be.

Ukovich et al. [21] investigated the use of a variety of MPEG-7 content descriptors. An example is "colour layout" - the average colour of each 8×8 image block, encoded with a DCT.

A variety of deep learning methods have been applied to this or similar problems Paixão et al. [17] use a fully-convolutional neural network and Pirrone et al. [20] use a Siamese network. Both of these papers use a programmatic method for fragmenting the documents for the training set due to the difficulty and time consuming nature of generating a large enough collection of physically fragmented ones.

A number of papers use techniques based on OCR. Paixão et al. [18] assess the shapes of boundary characters that are created when two strips are joined together. This is done by taking the smallest Hausdorff distance each boundary character has from a set of characters selected from the interior. Liang et al. [13] use a multi-level similarity measure. First potential matches are found with low level pixel difference. These are refined with first character level and then word level OCR confidence scores which are generated by the OCR tool Tesseract (see section 3.4).

## **Optimisation methods**

The optimisation technique depends upon what combinatorial optimisation problem the original problem is interpreted to.

Much of the existing work interprets the problem as either the travelling salesman problem or the nearly identical minimum-weight Hamiltonian path problem (see section 3.2.1). A large amount of literature has been written on efficient algorithms for solving this problem. The techniques that have been applied to reassembling shredded documents in particular, include ant-colony optimisation[7] and greedy composition[13].

Another common interpretation is as an assignment problem, which can be solved efficiently by the Hungarian algorithm. This is used by Alhaj et al. [5] and Chen et al. [8] among others.

## **3.4 Specialist Libraries**

### **Tesseract**

Tesseract is an open-source Optical character recognition engine. Tesseract has an API allowing it to be integrated with C++.

### **OpenCV**

OpenCV is an open-source computer vision library with APIs in a variety of languages, including C++. OpenCV has a wide variety of inbuilt image processing utilities useful to the project.

# 4 Specification and design

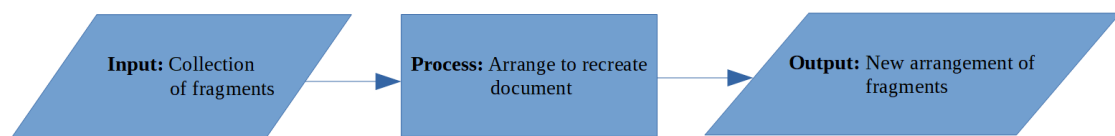
## 4.1 Requirements

The functional requirements for this project are simple:

- The user must be able to input a set of fragments (that meet the constraints outlined in section 2.1)
- The application must order these fragments to best resemble a document
- The application must provide output indicating the new order of the fragments

## 4.2 Architecture

The application can be modelled using the basic IPO (input-process-output) model.



*Figure 9: Flow chart of the overall application*

The objective of the project is not concerned with the nature of the input and output so discussion of these is limited to implementation (section 5). Discussion of the design will focus on the core problem of arranging the fragments.

## 4.3 Design

### 4.3.1 Problem definition

Under the constraints of vertical, strip-shredded fragments, the collection of fragments can be described as a sequence (fragment 1, fragment 2,..., fragment n) arranged from left to right. Any permutation of this sequence gives one of the  $n!$  potential arrangements of fragments, or solutions. To reconstruct the document, it is necessary to find the solution that most resembles a complete document. Instead of assessing the quality of the entire document, this can be evaluated by how discrepant<sup>1\*</sup> each fragment in the solution is with the adjacent fragments.

---

<sup>1</sup> The choice of compatibility vs discrepancy is arbitrary. Discrepancy was chosen to be more consistent with the usual phrasing of the minimum-weight Hamiltonian path problem (see 3.2.1)

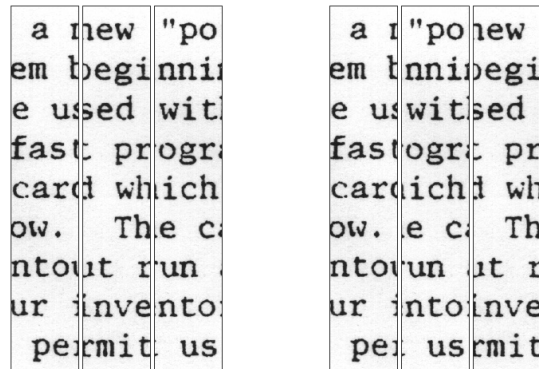


Figure 10: Two possible arrangements of three strips. In the example on the left the strips are arranged so that they are compatible with each other. In the example on the right the strips are arranged to be discrepant.

A value quantifying the discrepancy can be assigned to each pair within the solution. The best solution is the one with the lowest total discrepancy.

### 4.3.2 Approach

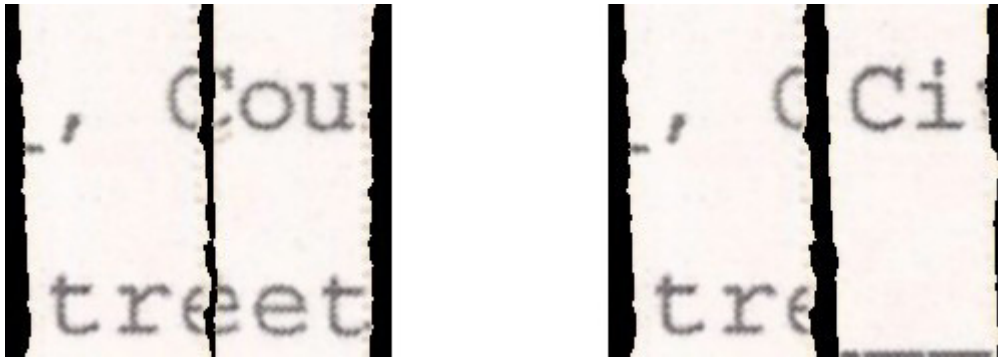
The process can therefore be divided into two parts: the discrepancy function, which scores a pair of fragments, and the optimiser which attempts to find the solution that minimises the total discrepancy.

### 4.3.3 Discrepancy function

A range of different approaches to measuring discrepancy or compatibility have been proposed (see section 3.3). One of the constraints outlined in section 2.1 is that the fragments should belong to a document with textual content. The high frequency changes in intensity in text documents makes them poorly suited to pixel difference based measures such as the method used by Alhaj et al. [5] which rely on high correlation between neighbours. This guarantee of text content, however, makes OCR viable so this is the approach I used. The proposed technique is largely inspired by Paixão et al. [18].

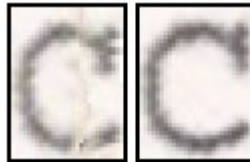
The proposed score is based upon the observation that shredding frequently results in characters being divided over two fragments<sup>2</sup>. If the characters formed when combining two strips appear valid, then the strips are likely compatible. Conversely, if joining the strips produces shapes that are not valid characters, then the strips are likely incompatible.

<sup>2</sup> Note that this assumption fails if the strips are too thin, so that one character can be divided over more than two fragments



*Figure 11: On the left, a section of a compatible pair of strips on the left, showing characters that are split over the two strips. On the right is a discrepant pair of strips.*

The validity of a merged character is assessed by comparing it with a collection of template characters using template matching (see section 3.2.2). The best template match score over all of the templates is the final validity score. This indicates how similar the merged character is to its most similar template character. To reduce the impact of documents with uncommon fonts, The template characters are sourced from within the fragments.



*Figure 12: On the left, the merged 'C' from the first pair in figure []. On the right, an example whole 'C' from within the same document. It can be seen that the two are not identical due to noise produced during the shredding and scanning.*

The overall discrepancy score is calculated from the average character validity.

#### **4.3.4 Optimisation**

With a compatibility score defined, an optimisation meta-heuristic can be employed to find the arrangement of the fragments that minimises the overall score. The proposed system first uses the nearest neighbour algorithm (see section 3.2.1) to quickly get an initial solution. This requires an initial leftmost fragment which is chosen using a heuristic that looks for the largest margin on the left side.

If the discrepancy was perfect and the leftmost fragment was always chosen, this would guarantee a perfect reconstruction. This is often not the case however, so further refinement to the solution is needed. This is achieved using Tabu search (see section 3.2.1).

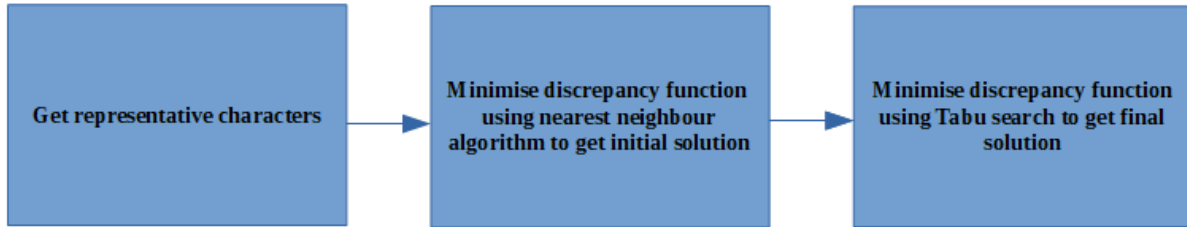


Figure 13: Overall flow of the 'Process' section.

## 4.4 Previous designs

### 4.4.1 Discrepancy functions

The accuracy of the discrepancy function is critical to the success of the overall approach, so this section required the most attention. A variety of techniques were experimented with, however the limited time frame available necessitated that those which did not seem initially promising were quickly rejected. One such approach used a linear regression model, trained on a dataset of shredded documents (see evaluation). For the input variables I tried both Haralick texture descriptors [10] and local binary patterns[15], but neither produced encouraging results.

The following methods showed greater promise so were developed enough to warrant mention in the results (section 6):

#### **Edge difference**

This simple method of finding total absolute difference between adjacent edge pixels initially seemed the most promising. Similar approaches were taken by Alhaj et al. [5] and Chen et al. [7]. To account for variation in height between strips, the difference was calculated with a range of offsets to the starting position of one of the strips and the best result was used. A further refinement replaced the total absolute difference with the normalised correlation value which mitigates the impact of different lighting when scanning the strips. This is achieved by normalising the difference with respect to the mean intensity.

$$\sum |R(y) - L(y)| \qquad \frac{\sum (R(y) - \bar{R})(L(y) - \bar{L})}{\sqrt{\sum (R(y) - \bar{R})^2 \sum (L(y) - \bar{L})^2}}$$

Figure 14: Expressions for absolute difference (left) and normalised correlation (right). At height  $y$ ,  $R(y)$  is the leftmost pixel in the right strip and  $L(y)$  is the rightmost pixel in the left strip.

The main issue with this approach is that text based document have frequent intensity changes from white to black, leading to lower correlation between pixels than would be expected in a more gradually changing document such as an image. This meant that the edge difference could be quite high even if the strips were compatible.

## ***Tesseract***

This largely followed the same steps of the final solution, however the validity of a character was determined using the confidence levels Tesseract gave when classifying it. This is similar to the character level metric used by Liag et al. [13]. This gave fairly poor results, unfortunately, because Tesseract is designed to assume that the input is supposed to be valid. This meant that the confidence scores often appeared unrealistically high for nonsense character shapes. Despite this, Tesseract still finds use in the system for selecting template characters as these are assumed to be valid (see section 5.3).

## **4.4.2 Optimisers**

### ***Simulated annealing***

Simulated annealing is an approach based on the real world annealing process. Like Tabu search, this is based on local search however the next solution is chosen randomly, with probability proportional its score. The results this gave were negligibly different to those achieved with Tabu search so choosing between the two was simply preference.

### ***Hungarian***

A number of papers such as Alhaj et al.[5] utilise a method known as the Hungarian method. This is designed to solve the assignment problem (see section 3.2.1). To reformulate this problem to the assignment problem, the fragments are split into their right and left sides. The problem is then to assign a right hand side for each left hand side. This approach encounters problems however because it is possible for closed loops to be formed in the assignment, leading to multiple separate documents being created. I was unable to discover how this problem was solved in papers that used the Hungarian algorithm so I was forced to abandon the approach.

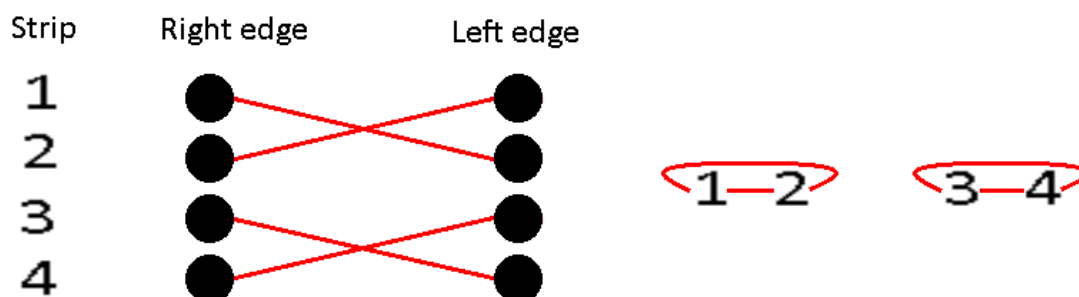


Figure 15: An assignment which causes closed loops.



# 5 Implementation

The Application was developed in C++, making use of the various standard libraries, as well as specialist libraries OpenCV and Tesseract (see section 3.4). Discussion of the implementation is divided into input, data structures and utility functions, the discrepancy function, the optimiser and output.

## 5.1 Input

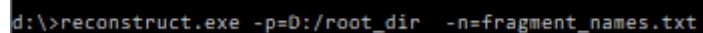
### Interface

To make the development easier the application is called from a command line. User interaction is limited to arguments passed in calling the executable. The arguments available are:

- The path of the top level directory (see input format)
- The name of the text file of fragment names
- The full name of the output file (optional – see section 5.5)

There are also two flags. If these are included the relevant functionality is applied. These are:

- Shuffle the fragments before reconstructing
- Combine the fragments together for output (see section 5.5)



```
d:\>reconstruct.exe -p=D:/root_dir -n=fragment_names.txt
```

*Figure 16: Calling from the command line*

There is little input sanitisation done so inputs must be exact. They must also not include spaces or backslashes.

### Fragment input format

To simplify development I specified a restricted input format. First there is a top level directory. In this directory is a text file specifying the number of fragments, and the name of each one. The top level directory also contains a directory called strips containing the fragment images. Optionally, the top level directory can contain a directory called masks containing binary images that identify the background and the strip for each image. These masks should share the same name with their corresponding strip. All images (strips or masks) should be png files.

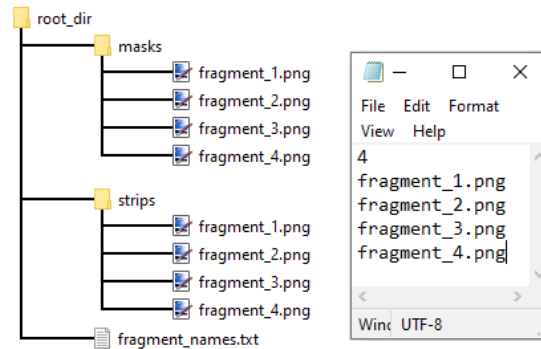


Figure 17: The input format. In this example masks are included.

## 5.2 Data structures and utility functions

### Representing fragments

The process of shredding and scanning often leads to fragments that are not perfectly rectangular. This means that the images often contain background as well as the fragment itself. The background can be identified using the masks provided (If no mask is present it is assumed that the whole image is strip). The index of the first and last non-background pixel in each row is stored so that iteration over the strip can be performed easily.

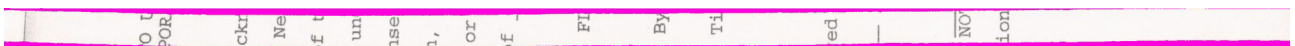


Figure 18: A scanned strip. The background has been coloured pink.

### Representing solutions

Solutions are simply ordered containers of fragments. The container only stores pointers which makes operations like swapping positions and copying solutions quicker.

### Thresholding

Otsu's method (see section 3.2.2) is provided by OpenCV, however it thresholds the entire image including the background. This would impact the quality of the threshold, so I had to implement the algorithm myself so that it would only take a histogram of the pixels in the strip. The thresholded version of each fragment is required a number of times during the process. To avoid needing to threshold repeatedly, a thresholded copy is stored within the fragment data structure.

## 5.3 Discrepancy function

The discrepancy function described in section 4.3.3 can be split into four main subtasks:

- 1.(prerequisite) Select representative character templates
2. Merge the strips
3. Segment the boundary characters
4. Validate each character

### 1. Selecting representative character templates

This step is performed once, before the optimisation stages. Each fragment is passed to Tesseract, which provides functionality for iterating over the individual characters it detects within the image. Tesseract identifies each and assigns a confidence confidence. The image of each character, as well as its confidence, is added to a data structure containing all the other characters of the same type. The characters that stray too far from the mean ( $+ \text{ or } - \frac{1}{2}$  of the height or width) are then removed. The character in each container with the highest confidence is chosen as a representative of that symbol and the rest are discarded. Storing all of the images in order to reject those that are outliers in size is not efficient but improves the quality of the selected templates.

### 2. Strip merging

This is the first step of the function proper. The strips need to be correctly aligned before they can be merged so that corresponding lines of text are matched at the same height. This is achieved with the following algorithm which finds the offset minimising the difference in average row intensities between the strips:

```
Input: left and right fragments
Output: best offset to merge text lines

max offset = height(left fragment)/4
for offset in range (-max offset:max offset):
    total = 0
    for each row in left fragment:
        l = average intensity in left fragment row
        r = average intensity in corresponding right fragment row with offset
        total += |l-r|
    average difference = total/number of rows
return offset with the lowest average difference
```

*Figure 19: Pseudo code for finding the best offset to merge at.*

Once the offset is calculated, the two strips can be merged with it applied. Naive concatenation would often result in a column of background down the centre. To prevent this the rows are combined individually, up to the last pixel in the left strip and starting from the first pixel in the right (see representing fragments).

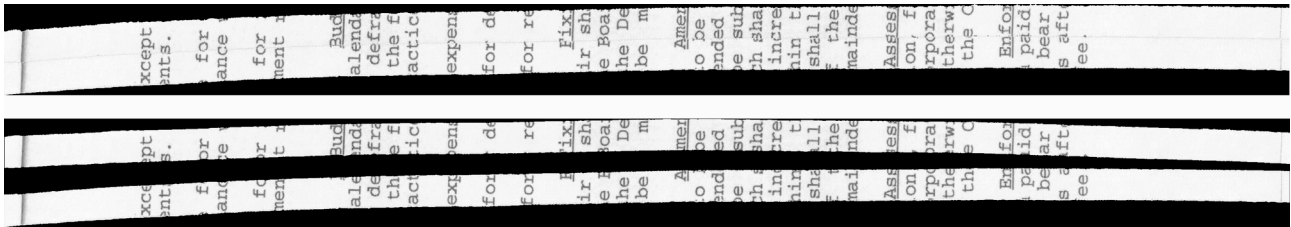


Figure 20: Top: combined strips. Bottom: concatenated strips.

### 3. Boundary character segmentation

Bounding boxes for the characters are found using the following algorithm on a thresholded copy of the image. These bounding boxes are then used to copy rectangular patches from the same position in the original image to be assessed.

The bounding boxes that are too large are rejected because they normally enclose graphics or characters that have become connected due to imperfect thresholding.

```

Input: thresholded fragment
Output: bounding boxes of characters along the boundary

initialise empty list
for each row in the fragment:
    if there is an unvisited black pixel touching the boundary
        use flood fill to get the component's bounding box
        if the box is not too big:
            if the box overlaps with previous one in the list:
                combined box = box + previous box
                if new box is also not too big:
                    overwrite previous box with combined box
            else:
                add box to the end of the list

```

Figure 21: Pseudo code for finding bounding boxes.

Too large is defined in the implementation as greater than  $1\frac{1}{2}$  times the width or height of the largest template character.

There are a couple of reasons that Tesseract was not used for this step. Firstly, calls to the Tesseract API are fairly expensive and this step needs to be done for every possible pair whereas only one call per fragment is needed in for choosing the template characters. Secondly, Tesseract is designed to locate valid characters and may miss invalid ones. These invalid characters are important for assessing the quality of the match.

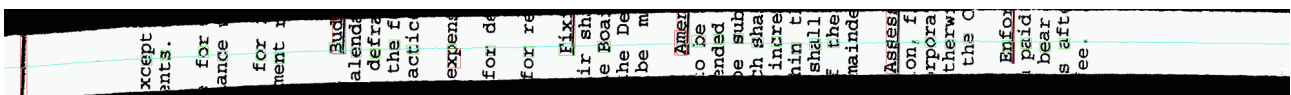


Figure 22: Visualisation of bounding boxes found in a pair of strips. Boxes in green are selected for validity assessment, whereas boxes in red are rejected. The cyan line is the boundary.

## 4. Assessing validity

Characters that are too small (less than 1/2 the smallest template width or 3/4 the height) are given a score of 0.

Characters that don't cross the boundary at any point are also given a score of 0 as this means that a character has not been merged.

The rest of the characters are passed to the template matcher and the score of the best match is found. The characters are padded with a border of white pixels to ensure enough window freedom to move. OpenCV provides inbuilt template matching with a variety of included comparison methods. Normalised cross correlation was determined empirically to be the best method.

$$R(x, y) = \frac{\sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x', y'} T'(x', y')^2 \cdot \sum_{x', y'} I'(x + x', y + y')^2}}$$
$$T'(x', y') = T(x', y') - 1/(w \cdot h) \cdot \sum_{x'', y''} T(x'', y'')$$
$$I'(x + x', y + y') = I(x + x', y + y') - 1/(w \cdot h) \cdot \sum_{x'', y''} I(x + x'', y + y'')$$

Figure 23: Normalised cross correlation at point  $x, y$ .  $T$  is the template image and  $I$  is the source image

This gives a value between 0 and 1, with 0 being entirely dissimilar and 1 showing they are identical. The optimisation implementation is designed to find the minimum score so the negation of the best template match value is returned by the function.

Once these tasks are complete, the final discrepancy is determined by taking the mean validity score (or 0 if no characters are present). This value ranges from -1: all merged characters are identical to the templates, to 0: no valid characters.

## 5.4 Optimiser

### 5.4.1 Finding an initial strip

To begin the nearest neighbour algorithm, an initial strip must be chosen. The heuristic used is based on the assumption that the left most fragment will have an empty white margin on the left hand side. The strip with the thickest margin is assumed to be the first one. This is calculated by finding the average proportion of each row that is left of the first black pixel, on the thresholded fragment. This gives a value between 0 and 1 for each row, 0 meaning that the first pixel is black and 1 meaning they are all white. The rows which are entirely white are discarded as these give no indication of the size of the margin. The strip with the largest margin will give the highest average value.

```

Input: thresholded fragment
Output: Average size of row margin

total = 0
rows_counted = 0
For each row in fragment:
    f = index of first non-background pixel
    l = index after last non-background pixel
    row_length = l-f
    If the row contains a black pixel:
        i = index of first black pixel
        rows_counted++
        total += (i-f)/row_length
return total/rows_counted

```

*Figure 24: Pseudo code for finding margin size in a fragment.*

### 5.4.2 Tabu search

The Tabu meta-heuristic leaves a number of design decisions to be adapted for each specific use.

#### ***Tenure***

The Tabu tenure is the number of iterations before an item in the Tabu list is removed. A value of 60 was chosen empirically.

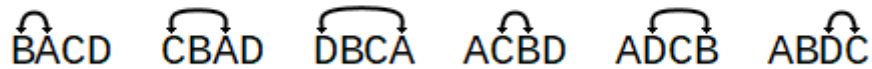
#### ***Stopping criterion***

The stopping criterion the condition under which the search should end. For this I chose to set a number of maximum iterations since an improvement to the overall solution was found. A value of 100000 was determined as a compromise between efficacy and time taken.

#### ***Neighbourhood***

The neighbourhood used is important to the success of the search. If it is too large, too many solutions need to be checked each iteration, making it expensive. If it is too small, not enough solutions will be explored and it becomes more likely to converge on a local minima. A good neighbourhood should also allow calculation of a new solution by a simple modification of the old one. The neighbourhood used is any solution which can be reached by swapping the position of two fragments. This means that each new solution's cost can be calculated from the previous one by subtracting the discrepancy cost between the pairs of fragments that are no longer adjacent and adding the discrepancy cost of those that have just become adjacent.

ABCD



*Figure 25: The neighbourhood of (A,B,C,D)*

### **Tabu list**

For each move to a new solution, the Tabu list stores a pair of identifiers describing which fragments were swapped at that move. The identifier used is the filename of the fragment. To make the Tabu list insensitive to the order of the pair of identifiers, they are stored and queried in lexicographic order. More advanced memory structures exist which differentiate between short, medium and long term memory to promote diversification, for exploring more solutions, and intensification, for narrowing in on promising areas of solutions. However, these were not implemented.

## **5.5 Output**

The default behaviour of the output is to concatenate the fragments in the order of the solution and then show the resulting image.

If an output file is specified, then the final image is saved to that file.

If the combine flag is included, then the fragments are combined without background instead of concatenating. This uses the same method that is used to merge two fragments in the discrepancy function (see section 5.3). Using this method over a large number of strips causes the noise to accumulate, leading to distortion increasing towards the right side of the document.





# 6 Evaluation

The primary aim of the project is to accurately reconstruct documents. This can be assessed by comparing documents recreated by the system with a known ground truth. To assess how competitive the proposed system is with existing solutions, the accuracy of this system can be compared to the accuracy of others on the same data. I chose to compare the performance with Paixão et al. [18] because it provided easy access to the dataset used.

## 6.1 Dataset

Paixão et al. provided two datasets of shredded documents along with ground truths for comparison. The datasets contain documents in English and Spanish. Some documents also contain varying sizes of graphical elements. The two datasets, D1 And D2, are further divided into mechanical and artificial. These contain the same documents that have been shredded with a different technique. The mechanical shredded examples are documents that have been printed, shredded and then scanned. The artificial examples are “shredded” algorithmically. I decided to ignore the D1 mechanical dataset as it is not possible to distinguish between the background and the strips, which makes the proposed system completely ineffective in many cases because it cannot join the edge characters together. D1 contains 60 original documents and D2 contains 20.

## 6.1 Metric

In order to compare my results with those of Paixão et al. on the dataset, I adopted the same comparison metric. Given a solution  $\pi$  of length  $n$ , Paixão et al. define a metric they call accuracy:

$$accuracy(\pi) = \frac{n - ngroups(\pi)}{n - 1}$$

*Figure 27: Formula for accuracy.  $\pi$  is the solution being measured and  $n$  is the length of solution  $\pi$ .*

A group is defined as a maximal correctly ordered subsequence. This is calculated by:

$$ngroups(\pi) = n - \sum_{i=1}^{n-1} \mathbf{1}\{\pi_{i+1} = \pi_i + 1\},$$

Figure 28: Formula for number of groups.  $\pi(i+1)$  represents the next fragment in the solution and  $\pi(i)+1$  represents the correct next fragment in the ground truth.

A perfect solution will have one group containing all of the fragments and a completely incorrect one  $n$  groups containing one fragment.

## 6.3 Methodology

Each collection of fragments is input in the correct order and the ground truth recorded. The fragments are then shuffled and then reconstructed by the system. Finally the rearranged fragments are compared to the ground truth using the metric above.

## 6.3 Results

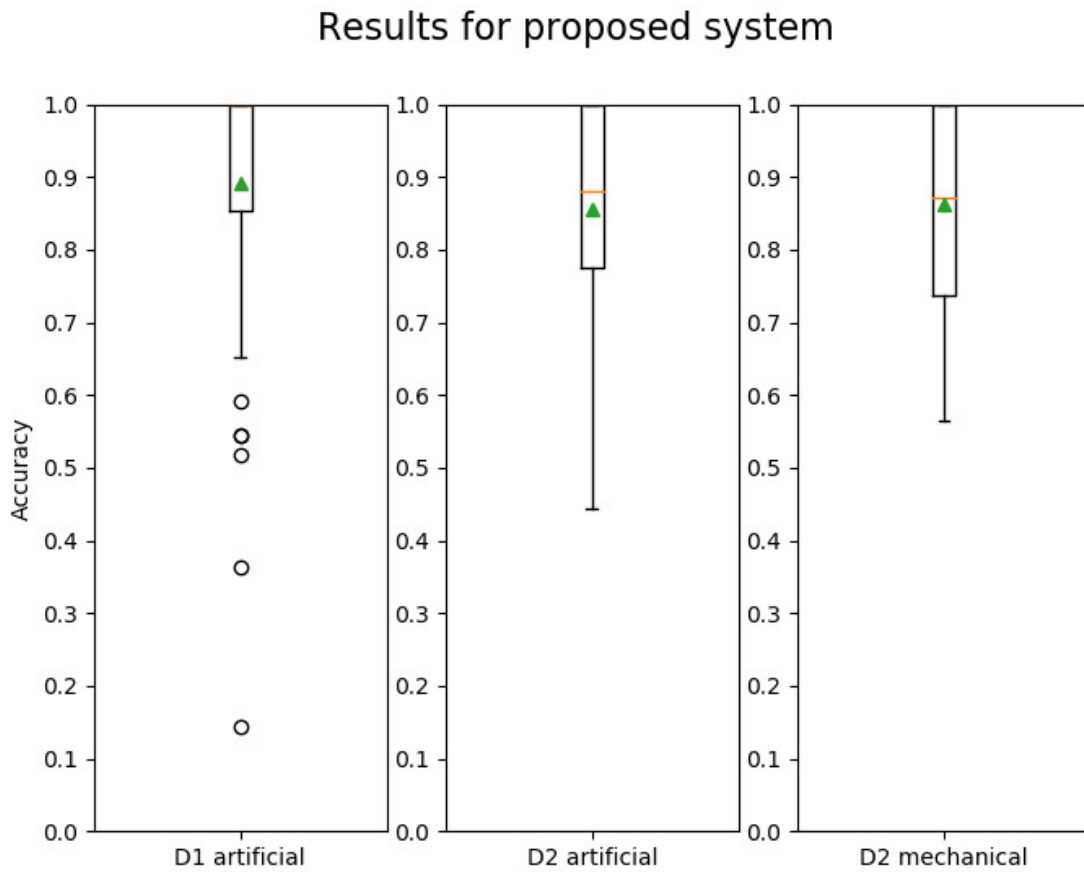
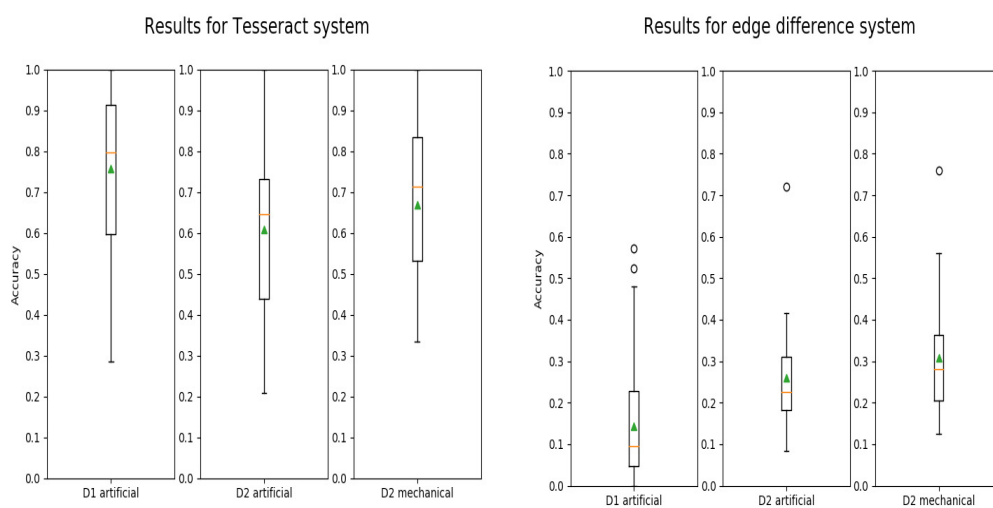


Figure 29: Results for the proposed system. The green triangle points to the mean value and the orange line is on the median value.

The system performed best on the artificial D1 dataset where in the median case the document was reconstructed perfectly. There were, however a few significant outliers. D2 artificial performed slightly worse with a median of 0.88. It is unsurprising that D2 mechanical had the slightly worse performance with a median of 0.87, because the process of shredding and scanning the documents was likely to create noise, however the fact that the difference is small suggests that the technique translates to real world problems fairly well.

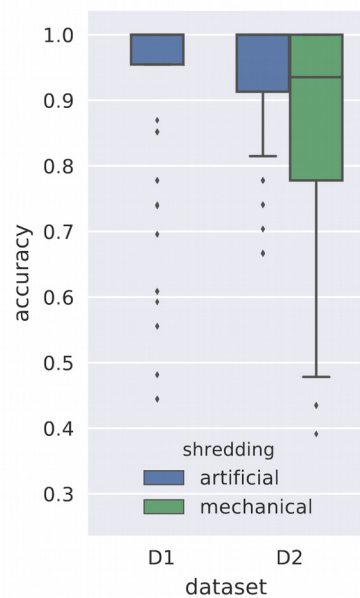
## 6.3 Comparison with older designs

These disused designs were developed during the project and are explained in section 4.4.1. These give some indication of the progress that was made.



*Illustration 2: Left: results for the Tesseract based system. Right: results for the edge difference based system.*

## 6.4 Comparison with existing designs



*Figure 30: Results from [et al.].  
The plot has been edited to  
remove results that are irrelevant  
to the comparison.*

The results from Paixão et al. show higher median scores across all datasets except D1 artificial which was 1 for both. The results also have a smaller interquartile range. These results show that the proposed system did not perform as well as Paixão et al. In order to be considered competitive with contemporary solutions, further refinements to the system would need to be made. Possible improvements are discussed in section 7.

## 6.5 Discussion

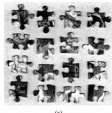
Investigation into the documents that were constructed well and those that were constructed poorly reveals clear patterns. Many of those that were recreated inaccurately had sparse amounts of text content. The worst performing document, which was from the artificial D1 dataset and scored 0.142857, had a large proportion of strips with very few lines of text. It also has a fairly even split between two sizes of text. Some of the poorest results also contained large graphics. Another common issue is documents with text that is very close together. This often results in characters becoming joined during thresholding. Characters that were joined were rejected as graphical elements because they became too large.

Para sistematizar a resolução Yui e Shao criamos uma sequência de características, conforme apresentado na Figura 2.29:

Para tipo	Sequência	Step 1	Step 2	Step 3	Step 4
1	001	0.7	0.7	0.7	0.7
2	002	0.7	0.7	0.7	0.7
3	003	0.7	0.7	0.7	0.7
4	004	0.7	0.7	0.7	0.7
5	005	0.7	0.7	0.7	0.7
6	006	0.7	0.7	0.7	0.7
7	007	0.7	0.7	0.7	0.7
8	008	0.7	0.7	0.7	0.7
9	009	0.7	0.7	0.7	0.7
10	010	0.7	0.7	0.7	0.7
11	011	0.7	0.7	0.7	0.7
12	012	0.7	0.7	0.7	0.7
13	013	0.7	0.7	0.7	0.7
14	014	0.7	0.7	0.7	0.7
15	015	0.7	0.7	0.7	0.7
16	016	0.7	0.7	0.7	0.7
17	017	0.7	0.7	0.7	0.7
18	018	0.7	0.7	0.7	0.7
19	019	0.7	0.7	0.7	0.7
20	020	0.7	0.7	0.7	0.7
21	021	0.7	0.7	0.7	0.7
22	022	0.7	0.7	0.7	0.7
23	023	0.7	0.7	0.7	0.7
24	024	0.7	0.7	0.7	0.7
25	025	0.7	0.7	0.7	0.7
26	026	0.7	0.7	0.7	0.7
27	027	0.7	0.7	0.7	0.7
28	028	0.7	0.7	0.7	0.7
29	029	0.7	0.7	0.7	0.7
30	030	0.7	0.7	0.7	0.7

Figura 2.29 – Características usadas por YUI e SHAO (2003)

Essas características mostradas na Figura 2.29 permitem a categorização das peças para montagem do quadro-cabeça conforme mostrado na Figura 2.29(a) e (b).



(a)

MARLOS ALEX DE OLIVEIRA MARQUES

## RECONSTRUÇÃO DIGITAL DE DOCUMENTOS MUTILADOS COM FORMAS REGULARES

Projeto de Dissertação de Mestrado apresentada ao  
Programa de Pós-Graduação em Informática Aplicada da  
Pontifícia Universidade Católica do Paraná como  
requisito parcial para obtenção do título de Mestre em  
Informática Aplicada.

Área de Concentração: Computação Forense e  
Biotecnologia

Orientador: Prof. Dr. Carlos Otávio de Almeida  
Freitas

CURITIBA  
2008

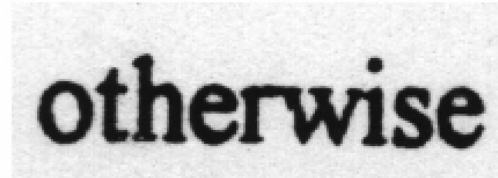


Figure 31: Some common problems. From left: large graphics, sparse text and characters merging when thresholding.

Documents that gave good results usually had dense text content, with mostly regular text sizes and fonts and few if any graphics.

## 7 Future work

Future improvements fall into two main categories: improvements to the performance and improvements to the scope of the system.

### Performance

Various limiting factors to the performance are mentioned in section 6.5.

One of these problems is the inclusion of graphical elements in many documents. An improvement to the system could detect which areas are graphical and which are text and use a different technique for determining compatibility for each. This could also accommodate documents that only consist of graphical elements.

Another problem often encountered was adjacent characters becoming merged during thresholding. A local thresholding approach may improve this but was not investigated due to time constraints.

A solution that may help to improve the results for documents with limited text elements could be to use a natural language processing feature. An approach similar to [ ] could be adopted where a low level method chooses potential solutions and a higher level one finds the best one of this reduced pool.

Finally, modifying the character segmentation to be less sensitive to noise would help reduce false positives when searching for merged characters.

### Scope

The project included a few constraints on the problem with the intention that some or all could be relaxed if the progress was quick enough. An obvious further development would be to extend the system to remove these constraints.

- **Strip shredded documents**

Accommodating hand torn documents could allow the shape of the fragments to become a useful feature for matching. Cross-cut documents would likely be harder although solutions have been proposed such as Chen et al. [7] which clusters fragments together based on the location of the textlines to give rows of fragments. These rows can then be treated as a strip shredded document.

- **Documents that include text.**

A method for handling documents without text is mentioned above in the performance section.

- **Latin alphabet**

Scripts which use a different character set to Latin but are otherwise similar would likely be fairly easy to accommodate. Writing systems such as Chinese, with thousands of characters, would likely be impractical using the current methodology as they would require a prohibitive number of template characters.

## 8 Conclusion

The primary objective of the project was to develop a system with high accuracy recreating shredded documents. Section 6.3 showed that the proposed system is reasonably successful in this aim. However, the success is conditional on certain attributes of the documents. As an extension it was proposed that some of the constraints on the system could be relaxed to better represent the real-world problem. This was not achieved so is suggested as possible future work.

The secondary objective of the project was for the system to be competitive with other existing solutions. Section 6.4 showed that further work would be needed to achieve this aim.

## 9 Reflections

I chose this project because I was interested in gaining some level of insight into the nature of academic computer science. The project necessitated digesting a number of research papers in order to assess existing solutions to the problem. This was challenging initially as I found many of them difficult to read. [sparsity] in particular I found so incomprehensible that I left it out of the existing works section as I felt unable to explain its method. I did find that with practice this became easier however, largely due to better understanding of some of the underlying techniques used.

While there have been many projects during my degree, none have allowed so much freedom as this one in terms of the approach to the problem. This seemed fairly overwhelming at first. Discussion with my supervisor was invaluable for identifying what ideas were worth exploring.

One aspect of the project that I came to appreciate better was how precise the problem definition should be. At the start the project was fairly vague with no specification on what shape of fragments would be used or what sort of content the documents might have. Applying constraints to the problem gave me much more direction in how I was to solve it.

An area I feel could definitely have been improved is planning and structuring the project timeline better. The plan I made for the initial report was fairly vague because I wasn't sure how long certain tasks would take or even exactly what tasks would be needed. In particular I underestimated how long it would take to get a basic initial solution working. This left less time to further improve the design. I had also hoped to be able to extend the problem possibly by accommodating strip shredded documents. Unfortunately I did not get the implementation for the initial problem to a high enough accuracy to justify further developments. I feel this project has given me better perspective on which aspects of the process take more or less time. In addition to poor planning, I did not use a particularly structured approach to the code development. Fortunately this did not present any major difficulties but it is still something that should be addressed for future projects.



# 10 References

## Other

- [1] Bowden. 2006. Guests of the Ayatollah
- [2] Brunelli. 2009. *Template Matching Techniques in Computer Vision: Theory and Practice*
- [3] EPA  
[https://www.epa.gov/sites/production/files/2013-09/documents/fec\\_automatic\\_duplexing.pdf](https://www.epa.gov/sites/production/files/2013-09/documents/fec_automatic_duplexing.pdf)
- [4] Skiena. 2008. The Algorithm Design Manual Second Edition.

## Papers

- [5] Alhaj, Sharieh and Sleit. 2019. Reconstructing Colored Strip-Shredded Documents based on the Hungarians Algorithm. 1-6. 10.1109/ICTCS.2019.8923048.
- [6] Bose and Soltanalian. 2017. Non-convex shredded signal reconstruction via sparsity enhancement. *ICASSP*, New Orleans, LA, 2017, pp. 4691-4695.
- [7] Chen, Tian, Qi, Wang, Liu. 2019. A solution to reconstruct cross-cut shredded text documents based on constrained seed K-means algorithm and ant colony algorithm. *Expert Systems with Applications*, volume 127, 2019, Pages 35-46.
- [8] Chen, Wu, Jia and Zhang. 2017. A pipeline for reconstructing cross-shredded English document. *IC/VC*, Chengdu, 2017, pp. 1034-1039.
- [9] Glover. 1989. *ORSA Journal on Computing* Vol. 1, No. 3 Pages 135-206
- [10] Haralick. 1979. Statistical and structural approaches to texture. *Proc. IEEE*, vol. 67, no. 5, pp. 786-804.
- [11] Kirkpatrick, Gelatt Jr and Vecchi. 1983. *Optimization by Simulated Annealing. Science.* 220 (4598): 671–680
- [12] Kuhn. 1955. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2: 83–97
- [13] Liang and Li 2020. Reassembling Shredded Document Stripes Using Word-Path Metric and Greedy Composition Optimal Matching Solver. *IEEE Transactions on Multimedia*, vol. 22, no. 5, pp. 1168-1181.
- [14] Naiman, Farber and Stein. 2019. Physical Match. *Informatica* 43(2019) 243–252.

- [15] Ojala, Pietikäinen, and Harwood. 1994. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. *Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994)*, vol. 1, pp. 582 - 585.
- [16] Otsu. 1979. *A threshold selection method from gray-level histograms*. *IEEE Trans. Sys. Man. Cyber.* 9 (1): 62–66
- [17] Paixão, Berriel, Boeres, Badue, De Souza and Oliveira-Santos 2018. A Deep Learning-Based Compatibility Score for Reconstruction of Strip-Shredded Text Documents. *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, Parana, 2018, pp. 87-94.
- [18] Paixão, Boeres, Freitas and Oliveira-Santos. 2019. Exploring Character Shapes for Unsupervised Reconstruction of Strip-Shredded Text Documents. *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1744-1754.
- [19] Pimenta, Justino, Oliveira and Sabourin. 2009. Document reconstruction using dynamic programming. *IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, 2009, pp. 1393-1396.
- [20] Pirrone, Aimar and Journet. 2019. Papy-S-Net: A Siamese Network to match papyrus fragments. *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing 2019 Sep 20* (pp. 78-83).
- [21] Ukovich, Ramponi, Doulaverakis, Kompatsiaris and Strintzis. 2004. Shredded document reconstruction using MPEG-7 standard descriptors. *Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology, 2004.*, Rome, 2004, pp. 334-337.