Mapping online data on offline documents

Bachelor's Thesis

Stefan Kennedie
s.kennedie@student.ru.nl
0223409
Supervisors: R. Niels & L. Vuurpijl

February 1, 2008

Contents

1 Introduction **2**

2 Method **5**

2.1 Data 5

2.2 Preprocessing data 8

2.3 Searching for the query 11

2.4 Investigating the interesting locations in detail 16

3 Experiment **18**

3.1 Retrieval experiment 18

3.2 Mapping on pixel level experiment 18

4 Results **19**

4.1 Retrieval experiment 19

4.2 Mapping on pixel level experiment 20

5 Discussion and conclusion **24**

6 Acknowledgments **28**

A Appendix **30**

A.1 SPSS output experiment 1 30

A.2 SPSS output experiment 2 31

A.3 Command-line arguments 33

A.4 Javadoc 34

Abstract

This thesis describes a method to find a part of online data in an offline document. This method is able to find the offline document that belongs to the online data from a set of offline documents, or vice versa. In order to optimize the mapping between the online and the offline data, an optimal rotation and resizing of the online data is calculated. This is useful since it produces a better mapping between online and offline data, which makes several methods that are only applicable for online data available for offline data, and vice versa.

Results show that this method can be used for finding the offline document that belongs to certain online data, since it succeeded in 98.07% of the cases for the used dataset. The results also show that computing the optimal rotation and resize factor significantly improves the mapping between online and offline data. This improvement is 6.56% for the used dataset.

1 Introduction

The reading of human handwriting by a computer is called (artificial) handwriting recognition. The data of the handwritten text can be obtained by a computer in two different ways. The first is by writing on a tablet, the second is by scanning a piece of written paper with a digital scanner.

The data obtained by writing on a tablet is called online data. Online data consist of a sequence of coordinates, with a corresponding pressure. These data can easily be visualized (see Figure 1).

Figure 1: A visualized example of online data.

The data obtained by scanning a piece of handwritten paper is called offline data. Offline data is a bitmap which consists of pixels with a grey-value or color (see Figure 2).

Figure 2: An example of a part of offline data.

Both online data and offline data have advantages and disadvantages over one other. An advantage of online data is that the order of writing is known, which is useful for segmentation [9]. Segmentation is an operation that divides the handwritten signal in lines, words or characters [3]. There are techniques for segmenting offline data (see [3] and [8]), but segmenting online data is easier [2]. Besides that, since the number of coordinate-pressure pairs per second, or sample rate, and the spatial resolution are known, the velocity and acceleration of the pentip can be computed [8]. Both are not possible

with offline data. Another advantage of online data is that the location of the pen is also known when it is not on the paper, but hovering above.

An advantage of offline data is that the width of the ink is exactly known [2]. Since online trajectories have zero width, this information can only be calculated from the pressure coordinate form the online data (see [5]).

Mapping online data on offline data allows us to use techniques that are only applicable to online data for offline data, and vice versa. For example, we can use the temporal information of online data [9] to help segmenting offline data [3]. Besides that, the information of online data can be used for preprocessing techniques (see Section 2.2) of offline data such as thresholding and noise removal [8]. Another example is that the pressure coordinate of the online data can explain the ink-distribution in the offline data. Since the results on online handwriting recognition are better than on offline handwriting recognition, it is useful to know the mapping between these different types of data in order to use the results of online handwriting recognition to label the offline data. In other words, when we are able to map online data on offline data, we can use the advantages of both data-types. As Vinciarelli and Perrone [10] describe, recognition improves significantly when the advantages of both modalities are combined, even when the offline data is generated from online data.

Unfortunately, as Franke [5] describes, it is not possible to superimpose the online pen trajectories and offline ink traces. There are several problems that arise here. The first problem is the difference in resolution between the used tablet and the digital scanner. This problem can easily be solved by scaling with a constant factor since this difference is mostly constant within a dataset (see Equation 1).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} c * x \\ c * y \end{bmatrix}, \text{ where } c = \frac{resolution'}{resolution} \tag{1}$$

The second problem is that the alignment of the sheet of paper on the tablet is not always the same, it can differ slightly per page. The result is that there can be a rotation and a translation of the paper with respect to the

online data (see Equations 2 and 3). The third problem is that this alignment problem also occurs with the digital scanner. In other words, the different pieces of paper are not aligned exactly the same (also see Equations 2 and 3).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} d_x + x \\ d_y + y \end{bmatrix} \tag{3}$$

The fourth problem is that people do not hold their pen vertical when they write. The consequence of this is that the pen does not write on the same place on the sheet of paper as that it produces pressure on the tablet (see Figure 3 and [5]). Besides that, Franke [5] also describes that the magnetic field of the pen that is used for writing on the tablet becomes asymmetric when it is tilted. This leads to a displacement of the maximal activation of this signal [5]. Unfortunately this problem is unsolved.

It is also possible that the the sheet of paper moves while a participant writes on it, but this issue is not further investigated in this thesis.

In Section 2 we explain the method used to create a better mapping between the online and offline data. In Section 3 we explain the two experiments that we have done for this thesis. In the first experiment we check whether or not the method explained in this thesis can be used to retrieve the correct offline data that belongs to the a certain part of online data. In the second experiment we check whether or not rotating and scaling the online data significantly improves the mapping of the online data on the offline data. In Section 4 we describe the results of the experiments. In Section 5 we discuss the weak points of the method and the problems that we encountered. At the end we derive a conclusion.

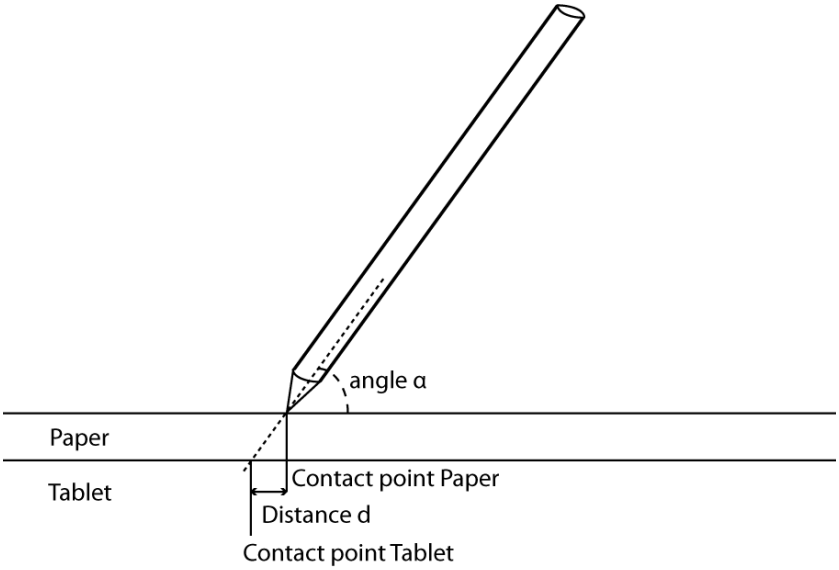


Figure 3: The effect of not holding your pen vertical when writing on paper that is situated on a tablet (problem four, described in Section 1).

2 Method

This section describes the method used for finding the optimal rotation and resizing of the online data with respect to the offline data. Figure 4 gives a visualization of the complete process.

2.1 Data

The NICI[1] owns a database that contains the online and offline data of the same handwritten text of 43 writers. This dataset was created by people that wrote with a pen on a sheet of paper that was attached to a tablet. By writing on the paper, and by that on the tablet, the coordinates and the pressure of the pen were obtained. The handwritten trajectories are sampled at a resolution of 200 dpi. By scanning the sheet of paper with a digital scanner, the offline data became digitally available [2]. The used resolution is 300 dpi. We use the data of three writers for this thesis.

[1]Nijmegen Institute for Cognition and Information (Radboud University Nijmegen).

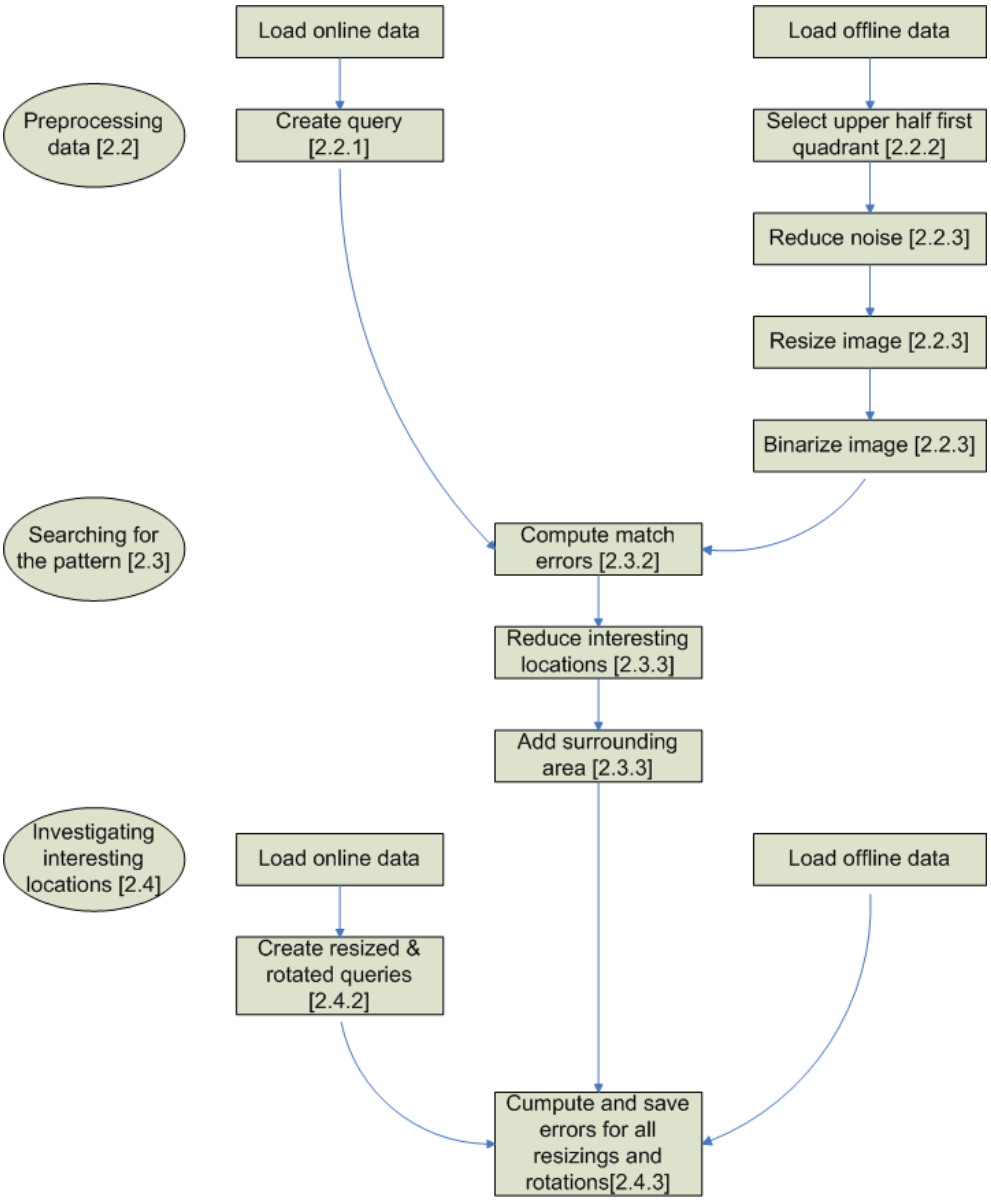


Figure 4: Flowchart of the used method with references to the sections.

A unique identification code is situated in the upper right corner of the document (see 'FAC101s' in Figure 5). This is the only part of the document that we focus on.

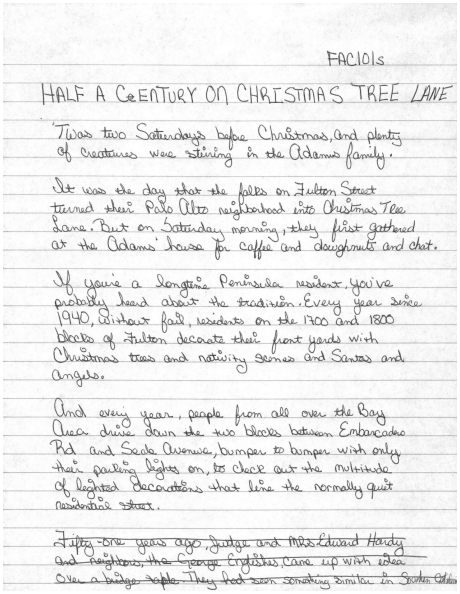


Figure 5: An example of offline data.

The online data-file contains a unique identification code that corresponds with the unique identification code in the offline data, and the number of coordinates in that data-file. On the following lines there is an x-coordinate, an y-coordinate and a pressure coordinate (see Figure 6). The coordinates of this file contain the unique identification code that is found in the first line of the online data-file and the upper right corner of the offline document.

```
AAR-FAC101 523
2754.000000 16241.000000 100.000000
2754.000000 16239.000000 100.000000
2754.000000 16237.000000 100.000000
2754.000000 16235.000000 100.000000
2756.000000 16233.000000 100.000000
...
```

Figure 6: An example of online data (only the first 6 lines).

Every word, or even every part of the document can be used as query[2]. Since the unique identification code is found only once in the handwritten text, and the location of this code in the online data is known, we use this as query.

2.2 Preprocessing data

In the used method, we lay the query over the offline document at all possible locations. The number of locations is huge (see Equation 4). At all these locations the difference between the query and that particular part of the offline document has to be computed, which is a sum that consists of $width(q) * height(q)$ subtractions (see Section 2.3.1).

$$\#locations = (width(off) - width(q)) * (height(off) - height(q)) \quad (4)$$

In this thesis, we use a method that avoids unnecessary calculations. This method is based on first making a coarse scan. It appeared that a resizing to one eight of the original size of the offline document created the best results in reasonable time. When the interesting locations[3] in the offline document were found (see Section 2.3), these locations were further investigated in full detail (see Section 2.4).

2.2.1 Creating a query

The online data consists of points with corresponding pressures (see Figure 6 in Section 2.1). The pen can be on and off the paper. Since the pen only distributes ink when it is on the paper, we are only interested in where the pen is on the paper. To be able to compare the coordinates of the online data with the offline document, a bitmap was created from the online data. We call this bitmap the query. Vinciarelli and Perrone [10] describe a way to create a bitmap from online data. This is done by connecting succeeding

[2]A query is a bitmap created from the online data (see Section 2.2.1).

[3]An interesting location is a location at which the found difference between the query and the part of the offline document is small.

coordinates by drawing a line between them, and subsequently applying a filter several times that grows the foreground areas. This creates a black-white image, but could be adapted to create a grey-value image. In this thesis, the points of the online data are connected using the Bresenham's line algorithm [1], as is done in [10]. The difference is that we adapted the line algorithm such that it is able to draw lines with a width of a certain number of black pixels (see Figure 7). We have chosen this method because it is computational less expensive than the method described in [10]. This algorithm is also able to create a bitmap with given dimensions, such that the resolution of the offline document and the query are approximately the same. At a offline resolution of 300 dpi, the average linewidth is 5 pixels. Since we work with one eight of that resolution, we used a linewidth of one pixel.

Figure 7: Created query from the online data of Figure 6. Also see Figure 1.

2.2.2 Location of the unique identification code

The unique identification code is always found in the upper right corner of the offline document, or, to be more exact, on the upper half of the first quadrant of the offline document. For computational reasons the rest of the offline document is not used. (See Figure 8, and compare it with Figure 5.)

2.2.3 Preprocessing the offline document

Since the query that was created from the online data only contains black and white pixels, we have chosen to create a black-white image from the offline data. First, the noise in the online document is reduced (See Figure 9, and compare it with Figure 8.)

The noise reduction is done by making all pixels with a grey-value under a certain threshold white. This threshold is determined using the Otsu Threshold Algorithm [7].

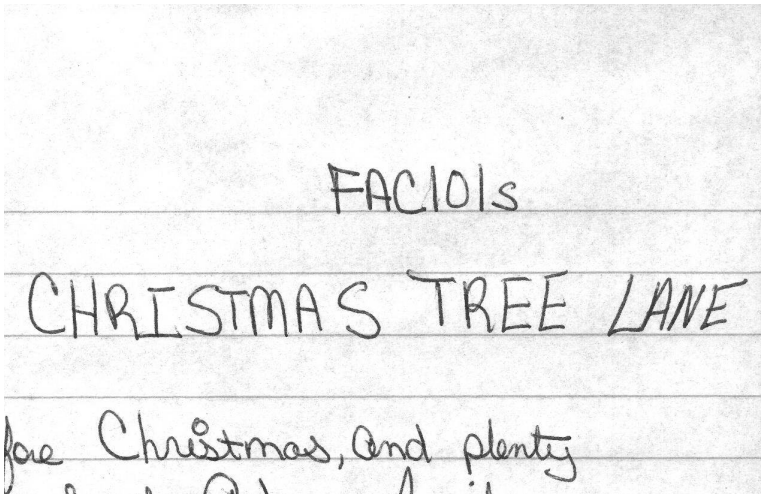


Figure 8: The upper half of the first quadrant of the offline document of Figure 5.

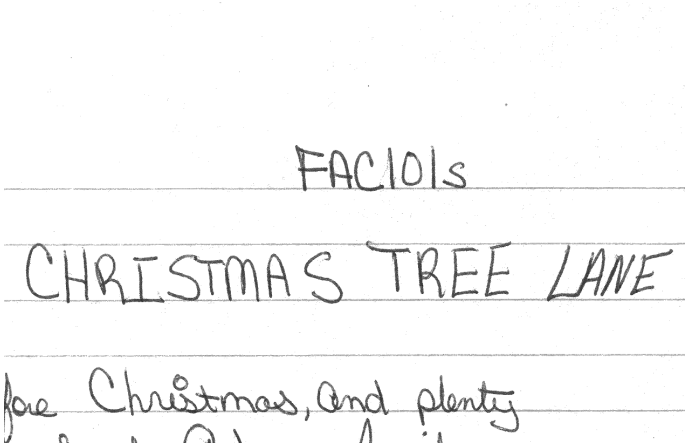


Figure 9: The result of removing the noise from the upper half of the first quadrant of the offline document (compare to Figure 8).

After the noise reduction, the offline document is resized and subsequently binarized (see Figure 10, and compare it with Figure 9). The threshold for binarization is again determined using the Otsu Threshold Algorithm [7]. Binarization makes the offline document black and white.

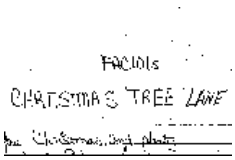


Figure 10: The result of applying the resizing and binarization on the offline data of Figure 9.

2.3 Searching for the query

This section describes the method used to find the location of the unique identification code in the offline document using the query created from the online data.

2.3.1 Comparing two images

Manmatha and Croft [6] describe Euclidean Distance Mapping for Matching [4]. This is a method to compute the difference between two images with the same size. In this thesis, an adjusted version of this Euclidean Distance Mapping for Matching is used to compute the difference between a part of the offline document and the query. A sliding window (with the size of the query) compares all pixels of the offline document with the corresponding pixel of the query (see Equation 5).

$$d(q, off) = \sum_{x_0}^{x_{max}} \sum_{y_0}^{y_{max}} (Intensity(q, x, y) - Intensity(off, x, y)) \qquad (5)$$

Parts of the offline data (with the size of the query) are compared with the query at all possible locations of the offline document. When two pixels are the same (they are both black or both white), the error is 0, otherwise the error is 1. To visualize the difference between the two images, a XOR image is created. An error on the pixel level of 0 is visualized as a black pixel and an error on the pixel level of 1 is visualized as a white pixel (see Figure 11). A match error is computed by counting all errors at the pixel level. In other

words, comparing two images with a higher match error results in a XOR image with more white pixels.

(a) A part of the offline document that is compared with Figure 11(b). The error is visualized in the XOR-image of Figure 11(c).

(b) The query that is compared with Figure 11(a). The error is visualized in the XOR-image of Figure 11(c).

(c) The XOR-image created to visualize the difference between Figure 11(a) and Figure 11(b).

Figure 11: The input and output for a XOR-algorithm.

Since the majority of the pixels of both the offline document and the query are white, only the errors corresponding to the black pixels of the query are counted for the match error. The advantage of this is that the range of possible errors is smaller since the total number of pixels that can produce an error is smaller. Since the majority of the pixels of the offline document is white, they now do not contribute to the match error. The result is that the relative match error is bigger at locations of the offline document

where there are a lot white pixels. A disadvantage is that the match error is zero at large black areas in the offline document (see Figure 23), which does not correspond to the location of the unique identification code. A possible solution for this problem is explained in Section 5.

2.3.2 Computing the match errors

The match error is computed for all possible locations of the query in the offline document. We use an early pruning method to prevent the algorithm from doing irrelevant calculations.

When all match errors are calculated, they can be visualized in an error image. The locations with a large error produce a dark pixel, and the locations with a small error produce a light pixel. See Figure 12 for an example of an error image.

Figure 12: Created error image for visualizing the match errors produced by Figure 10 as offline document and Figure 7 as query, using the black pixels of the query.

Every match error is visualized on the error image at the location that corresponds to the upper left pixel of the part of the offline document that is compared with the query.

Since small errors produce white pixels, a white spot on the error image means that the unique identification code that is being searched could be situated on that location. The locations that correspond with the lightest pixels are called interesting locations.

The dimensions of the error image are smaller than the dimensions of the

offline document. This is because the unique identification code cannot be found on a location that is near the right or lower side of the offline document since a part of the query would then fall off the offline document.

2.3.3 Processing the match errors

Also for computational reasons, the number of interesting locations has to be reduced. In order to do so, the number of locations that do not have the maximum match error value is reduced. This is done by considering only the locations at which the error is smaller than a certain threshold. This threshold is computed using Equation 6.

$$threshold = \frac{biggestMatchError + smallestMatchError}{reduceFactor} \tag{6}$$

The reduceFactor must be bigger than 1.0. The larger this reduceFactor is, the more locations are considered as not interesting. The number of pixels that remain can be adjusted. We chose to leave only three pixels since this produced good results within reasonable time. The result of this operation is visualized in Figure 13. Compare it with Figure 12.

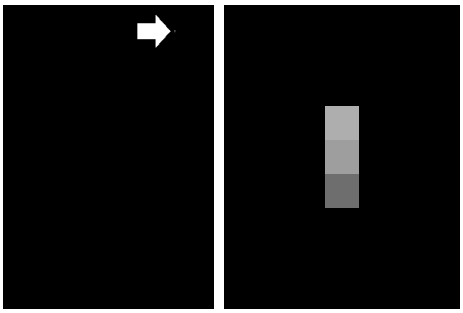


Figure 13: Visualized result of removing the uninteresting locations (left), and the magnification of the most important part of this visualization (right). The most important part of the image is is the part of the image that contains the interesting locations. Compare to Figure 12.

The locations that remain are the locations that possibly contain the searched code, and are marked for further investigation. This is visualized in Figure 14. They will be recalculated on the original size of the offline document.

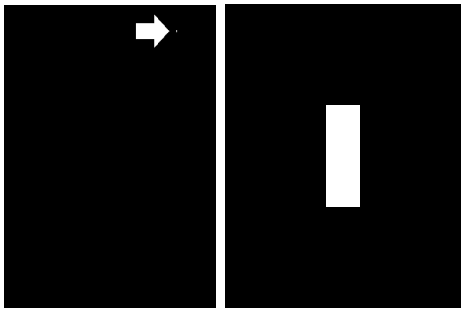


Figure 14: Visualized result of marking the interesting locations (left), and the magnification of the most important part of this visualization (right). Compare to Figure 13.

Since we also want to investigate the locations in the surrounding area of the interesting locations, we add these locations to the list of interesting locations. The effect of this operation is visualized in Figure 15.

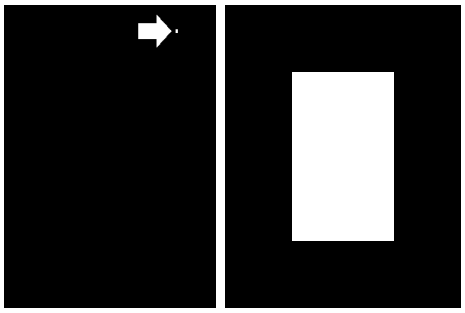


Figure 15: Visualized result of adding the surrounding area of the interesting locations (left), and the magnification of the most important part of this visualization (right). Compare to Figure 14.

2.4 Investigating the interesting locations in detail

In Section 2.3 we explained how interesting locations were found using resized offline data and a query created from the online data. In this section we explain how we further investigate these interesting locations. The locations are recalculated with all given rotations and resizing factors of the query (see Section 2.4.2), and are saved to a file (see Section 2.4.3).

2.4.1 Resizing the coordinates of the interesting locations

The locations that correspond with the white pixels in Figure 15 are the interesting locations. These locations possibly contain the unique identification code, and are therefore recalculated in real size. In Section 2.2 we explained that for computational reasons the offline document and the query were resized to 12.5% of the original size of the offline document. To be able to process the interesting locations in detail with the original offline document, these locations have to be recalculated to the original size of the image. In other words, the coordinates of the interesting locations have to be multiplied with a factor 8.

2.4.2 Resizing and rotating the query

There are two methods for creating a resized and rotated query. The first method is to create a query from the online data, and resize and rotate this image. The second method is to resize and rotate the coordinates of the online data, and create a query from this rotated data.

Resizing and rotating an image is done using affine transform operations. Resizing the coordinates of the online data-file is done by multiplying the coordinates with the resize factor. Rotating the coordinates of the online data is done using the rotation-matrix of Equation 7.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7}$$

The method used in this thesis is resizing and rotating the coordinates of the online data-file since this are operations at the source of the data with a

higher resolution. This produces a better outcome then rotating and resizing the an image.

2.4.3 Computing the errors

Next the offline document is loaded, and the match errors are recalculated for all rotations and resizings of the query at the interesting locations. The linewidth of the query plays a role in computing the match error. When the query consists of more black pixels, there are more pixels that can produce a match error. Since we only use the locations of the black pixels of the query to compare with the corresponding pixels of the offline document, the maximal possible match error is bigger when the linewidth of the query is bigger. The used linewidth is one pixel since this produces the most reliable match errors. The results can be visualized in an error image (see Figure 16 for an example).

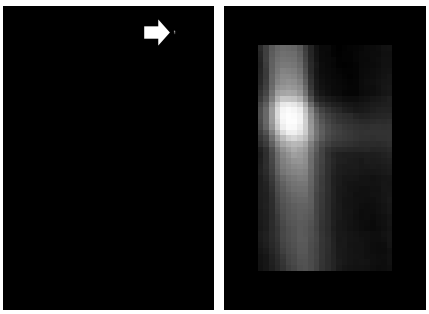


Figure 16: Created error image using the offline data at its original size (left), and the magnification of the most important part of this visualization (right).

The output is written to a file. This file contains the match error, the location of the unique identification code, the used resize factor, the used rotation in degrees, the total number of black pixels of the query and the percentage pixels of it that maps on the ink of the offline document for all calculated resize factors and rotations at all interesting locations. The higher this percentage, the better the match between the document and the query at that location with that resize factor and rotation.

3 Experiment

As noted in the introduction, we have done two experiments. The first experiment is a retrieval experiment or mapping on document level experiment, the second experiment is the mapping on pixel level experiment.

3.1 Retrieval experiment

For each writer, the data is divided in several parts. One part consists of approximately 14 online data samples, and approximately three times as much offline data samples.

In this experiment, all offline data is checked for the occurrence of all online data. This is done by computing the optimal match of all online data for all offline data. In the output-file we find the highest percentage of correct pixels per data-pair. To see whether or not the offline document with the highest percentage of correct pixels indeed was the document that corresponds to the online document, we visually explored the results[4]. In this experiment, the online data was not resized or rotated.

3.2 Mapping on pixel level experiment

When it is known what offline data-file belongs to what online data-file, the mapping on pixel level experiment can be performed. In this experiment the optimal mapping between the online and offline data is calculated by computing the maximum percentage of correct pixels for all rotations and resizings for all online-offline data-couples[5]. The chosen resizing factors are 0.99 till 1.03 with steps of 0.01. The chosen rotations are -0.3 till +1.5 degrees with steps of 0.1 degree. In total 95 $(5 * 19)$ resizing-rotation pairs are calculated per online-offline data-couple. The wider the range of resizing and rotation, and the smaller the step-size is, the better the result of this experiment will be. We have chosen these ranges and step sizes because short tests showed that they provide a good result within reasonable time.

[4] See `http://baseline.ai.ru.nl/OnOff/` for more information.

[5] A data-couple is a offline and online document that belong together.

4 Results

Both experiments were performed on three writers (AAR, ALS and APR). In the next sections you can read the results of these experiments.

4.1 Retrieval experiment

For the first writer, 57 online data-files were compared with 57 offline data-files, from which we knew which online data-file belonged to which offline data-file in advance. In other words, the mapping on document level was already known. These data-files were subdivided in four parts, and 818 comparisons were made. These comparisons can be subdivided in two groups. The first group is the group in which an online data-file was compared with an offline data-file that did not belong to it. This group, that we will call the 'no match'-group, consists of 761 comparisons. The other group is the group in which an online data-file was compared with an offline data-file that belongs to it. This group, that we will call the 'match'-group, consists of 57 comparisons.

The mean percentage of correct pixels found for the 'no match'-group was 40.38%, the mean percentage of correct pixels found for the 'match'-group was 86.44% (see Figure 17 for the distribution of the percentage correct pixels). The mean difference of 46.06% is a significant difference ($p < .001$). The algorithm correctly found 100% of the data-couples for this writer.

For the second writer, 90 online data-files were compared with 274 offline data-files. The data-files were subdivided in five parts, and 4783 comparisons were made. The 'no match'-group consists of 4693 comparisons, and the 'match'-group consists of 90 comparisons.

The mean percentage of correct pixels found for the 'no match'-group was 35.51%, the mean percentage of correct pixels found for the 'match'-group was 75.95% (see Figure 17 for the distribution of the percentage correct pixels). The mean difference of 40.44% is a significant difference ($p < .001$). Since the mapping on document level was not known, we had to visually explore the results. This showed that the algorithm was unable to find the

correct offline data of four online data-files. This are the four peeks in Figure 17 at 29%, 36%, 43% and 50%. The algorithm correctly found 95.56% of the data-couples for this writer.

For the third writer, 60 online data-files were compared with 274 offline data-files. The data-files were subdivided in five parts, and 1882 comparisons were made. The 'no match'-group consists of 1822 comparisons, and the 'match'-group consists of 60 comparisons.

The mean percentage of correct pixels found for the 'no match'-group was 44.01%, the mean percentage of correct pixels found for the 'match'-group was 83.67% (see Figure 17 for the distribution of the percentage correct pixels). The mean difference of 39.65% is a significant difference ($p < .001$). Since the mapping on document level was not known, we had to visually explore the results. This showed that the algorithm was able to find 100% of the data-couples.

The overall result shows that the algorithm was able to find 203 of 207 data-couples. This is a score of 98.07%. The overall mean percentage of correct pixels found for the 'no match'-group was 38.15%, the overall mean percentage of correct pixels found for the 'match'-group was 81.08% (see Figure 17 for the distribution of the percentage correct pixels). The overall mean difference of 42.93% is a significant difference ($p < .001$).

See Table 1 and 2 in Appendix A.1 for SPSS-output of this experiment.

4.2 Mapping on pixel level experiment

For the first writer, the optimal rotation and resize factor of 57 data-pairs were calculated, and subsequently the effect of this rotation and resizing on the percentage correct pixels was calculated. The percentage of correct pixels before the experiment was 86.06%, after the experiment this was increased to 93.21%. This difference of 7.15% is a significant difference ($p < .001$). See Figure 21 for the distribution of optimal rotations, and Figure 22 for the distribution of resize factors.

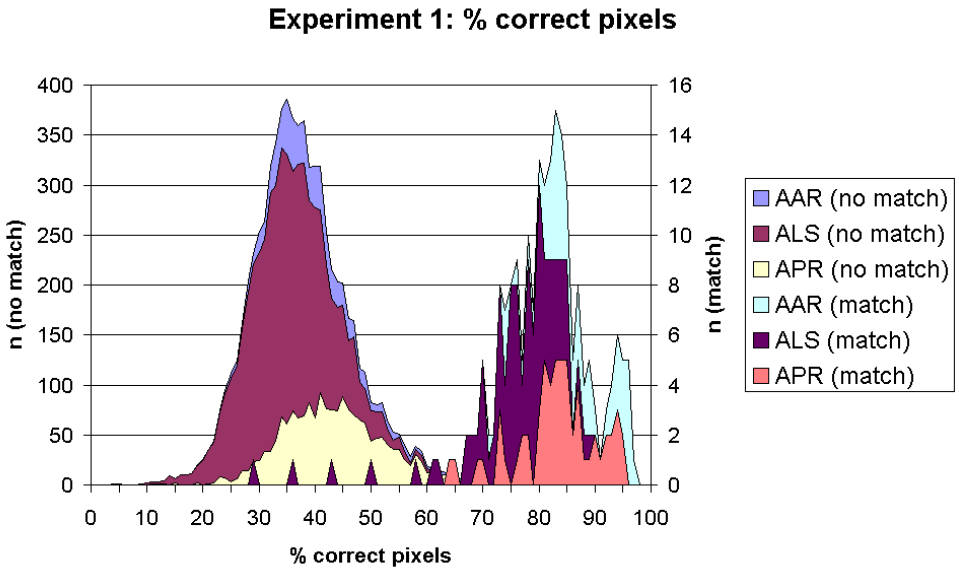


Figure 17: The number of occurrences of correct pixels for the three writers for the 'no match'-group (left y-axis) and the 'match'-group (right y-axis).

For the second writer, the optimal rotation and resize factor were calculated for 86 data-pairs. The first results showed that the chosen range of resizing factors was probably not large enough. In 22 of the 86 data-pairs the optimal resizing factor found was 1.03. For the first writer we found zero data-pairs with an optimal resizing factor of 0.99 or 1.03 (the lower and upper bounds of the resize factor range, see Figure 22.) In other words, the optimal resizing factor might be larger than 1.03. We decided to perform the experiment for this writer with resizing factors 0.99 till 1.05. The effect of this difference in range did not produce a large difference in percentage correct pixels. This percentage rose with 0.03% to 81.91%.

The difference in correct pixels of 6.12% (75.79% before the experiment and 81.91% after the experiment) was significant ($p < .001$). See Figure 21 for the distribution of optimal rotations, and Figure 22 for the distribution of resize factors.

In Section 4.1 we noted that the algorithm was unable to find the correct offline data for four online data-files. These online data-files also produce

a problem in the second experiment. The locations found by the algorithm do not correspond with the location of the unique identification code (see Figure 18).

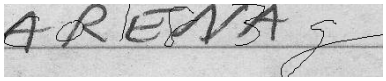


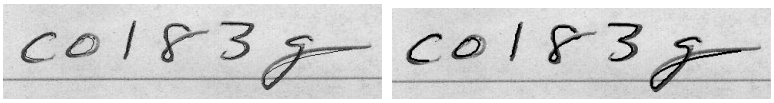
Figure 18: Example of incorrect location found in the mapping on pixel level experiment

When we tried to locate the unique identification code manually, we found that the algorithm was probably unable to locate it since it was impossible to map the online data on the offline data with rotation and resizing as only operations. See the 'o' and the 'g' of Figure 19(a). This problem is probably originate in the orientation of the pen (see [5]), but this remains an unsolved problem.

In attempt to investigate whether or not this was the case, we did the experiment again for this data-pair with linewidth three and five. The idea was that with a linewidth of three or five, relatively more pixels of the query mapped over the offline ink. As visualized in Figure 19(b) and 19(c), this might be an explantation to the problem. Unfortunately the algorithm still found the same location as before (see Figure 18).

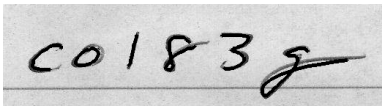
Since this also did not solve the problem, we investigated the images that were created during the experiment. The image of the resized offline data showed that there were only a few pixels left of the unique identification code (see Figure 20(a), and compare it with Figure 20(b)). Since there were only so few pixels left, it was impossible to get a high match at the location of the unique identification code. As a consequence of this, a location with many black pixels has a higher match, and is chosen to investigate in detail. Possible solutions for this problem are discussed in Section 5.

For the third writer, the optimal rotation and resize factor were calculated for 56 data-pairs. The first results showed that the chosen range of rotation factors probably was not large enough. In 15 of the 56 data-pairs the optimal



(a) Linewidth one.

(b) Linewidth three.



(c) Linewidth five.

Figure 19: An example of the problem with four data-pairs of writer two. It is impossible to produce a good mapping of the 'o' and the 'g' with rotation and resizing as only operations. In this image, the image created from the online data (solid black line) was pasted into the offline document.

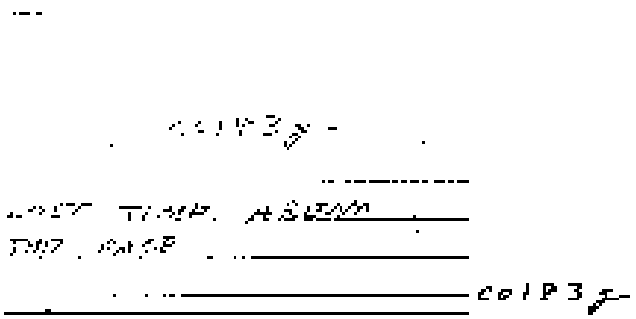


Figure 20: The resized offline document (left) and the created query (right).

rotation found was 1.4. In other words, the optimal rotation might be larger than 1.4. We decided to perform the experiment for this writer with rotating factors -0.3 till 2.3. As a result of this difference in range, the percentage correct pixels increased with 0.18% to 89.50%. An improvement of 2.02%.

The difference in correct pixels of 6.64% (82.86% before the experiment and 89.50% after the experiment) was significant ($p < .001$). See Figure 21 for the distribution of optimal rotations, and Figure 22 for the distribution of resize factors.

The overall mean difference in correct pixels of 6.56% (80.72% before the experiment and 87.28% after the experiment) was also significant ($p < .001$).

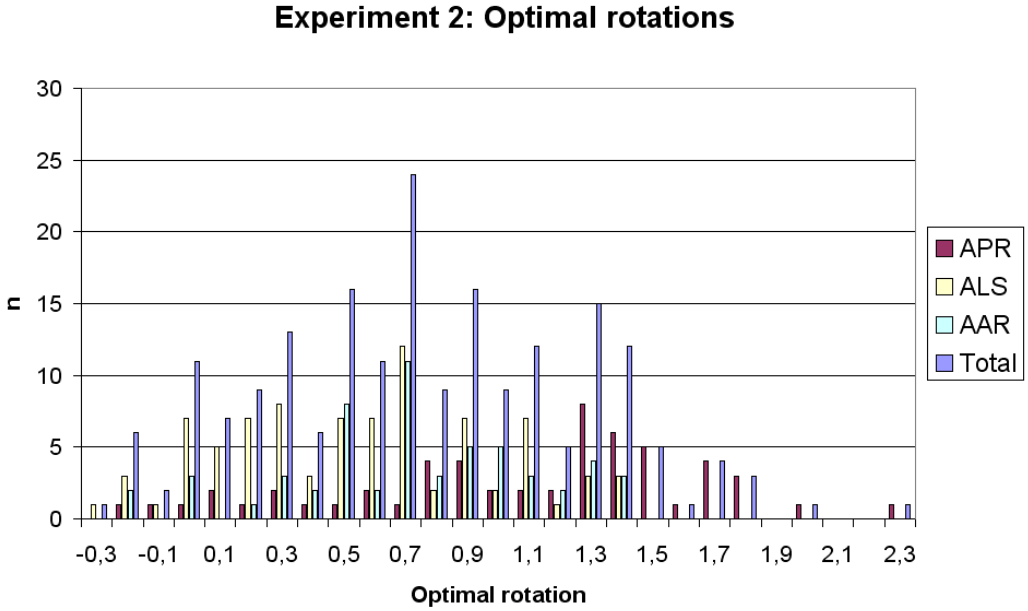


Figure 21: The number of occurrences of the different optimal rotations for the three writers.

It is also worth mentioning that when we only use rotation as operation, the method also produces a significant difference for the three writers and their sum ($p < .001$) for all four data-sets). The difference is also significant ($p < .001$) for all four data-sets) when resizing is the only used operation.

The difference in percentage correct pixels is also significant when the method is used with rotation and resizing instead of only resizing or only rotation (both $p < .001$) for all four data-sets).

See Table 3, 4 and 5 in Appendix A.2 for SPSS-output of this experiment.

5 Discussion and conclusion

Since Java is easier to program than several other languages, it is platform independent and it provides several libraries for processing images, we have

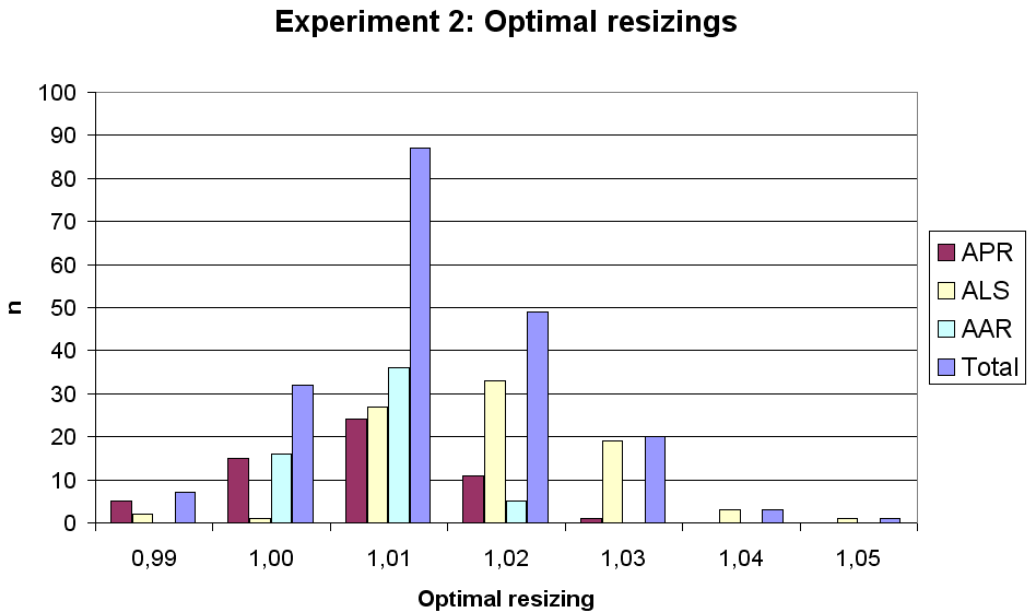


Figure 22: The number of occurrences of the different optimal resizings for the three writers.

chosen to use Java. A disadvantage of Java is that the speed is inferior to the speed of several other languages. To use the method described in this thesis on large scale, it is recommended to write (parts of) the code in a faster language. Besides that, the method used is a computational expensive method. Therefore it is recommended to search for another method.

An alternative approach to locate features in an image is template matching using a Fast Fourier transformation. This method is closely related to Fast Convolution[6]. This works much faster than the method described in this thesis. This method is very promising, but unfortunately we did not have the time to elaborate it.

Another problem is a problem with the offline data. Sometimes the pages where folded in the corner while being scanned. As a result of this, some documents have large black regions. These black regions are problematic

[6]See `http://www.mathworks.com/access/helpdesk/help/toolbox/images/f21-17064.html` for an example.

since the method only compares the black pixels of the query with the offline document, which results in a perfect but false match (see Section 2.3.2). Although this did not happen in the experiment, this problem can probably be solved with an edge detection on the offline document. After the edge detection, the image has to be inverted. This edge detection should take place after the noise reduction that is described in Section 2.2.3. It is worthwhile investigating this in future research. See Figure 23 for an example.

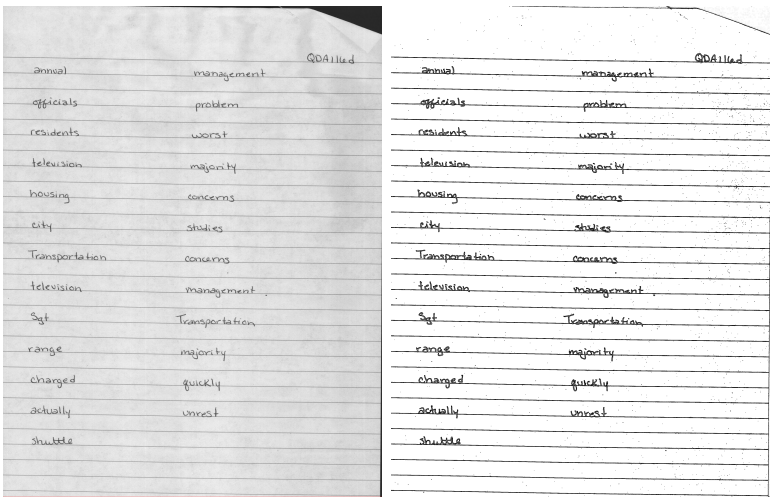


Figure 23: An example of an offline document that was folded while being scanned (left), and the result of an edge detection algorithm and inversion on this document (right).

The significance of the result of experiment 2 (see Section 4.2) depends on the data. If there was no difference in rotation and resizing between the offline and online data, the method would not have found a significant difference.

We think that the problem that occurred for the second writer during the mapping on pixel level experiment (see Section 4.2) can possibly be solved on several ways. The first possible solution is increasing the number of locations that are investigated in detail (see Section 2.3.3). An disadvantage of this method is that the algorithm will become slower as more locations are investigated in detail. The second possible solution is computing the

match error for black and white pixels of the query instead of only using the black pixels. But, as described in Section 2.3.1, this also has disadvantages. The third possible solution is applying an edge detection algorithm and inverting the image as described above. As you can see in Figure 23, the unique identification code is bolder in the right image. As a result there might remain more pixels of the unique identification code after resizing. The last possible solution might be found in the Otsu algorithm since this algorithm produces the threshold for binarization. However, the threshold that was found for this example was comparable with thresholds that were found for other offline documents. It is also possible to combine these possible solutions. Unfortunately we did not have time to focus on these possible solutions.

As written in Section 2.1, every part of the handwritten text can be used to compare online data with the offline data. This makes the method useful for spotting words of online data in offline documents.

Since it is more interesting to provide a good mapping for the complete document instead of only one word, it is worthwhile to investigate whether or not the method described in this thesis is useful to use the complete online data as query. It is probably also interesting to investigate whether or not this result differs from the mean result of three queries at different locations in the document.

The mean percentage correct pixels for the 'match'-group (see Figure 17) in experiment 1 are different for the three writers. This might be originated in the orientation of the pen (see Figure 3 and [5]). There is a possibility that this characteristic can be used in the field of writer identification. It is worthwhile investigating this in future research. Besides this, with an adaptation to the method, it is possible to compare a single written letter with a set of document in order to identify the writer of the letter.

To conclude, the method explained in this thesis is useful for finding the correct offline document given the online data, it succeeded in 98.07% of the cases. The method is also suitable for finding the rotation and resizing of the online data for a significant better mapping with the corresponding offline data, it provided a 6.56% better mapping on the pixel level. This

mapping can be increased further when local affine transformations and the orientation of the pen are added to the used operations.

6 Acknowledgments

We would like to thank F.A. Grootjen for showing the possibility matching using a Fast Fourier transformation.

References

[1] J. Bresenham, "Algorithm for computer control of a digital plotter." *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[2] S. K. C. Viard-Gaudin, P.M. Lallican and P. Binter, "The ireste on/off (ironoff) dual handwriting database," *Document Analysis and Recognition*, p. 455, 1999.

[3] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, vol. 18, no. 7, pp. 690–706, july 1996.

[4] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 227–248, february 1980.

[5] K. Franke, "The influence of physical and biomechanical processes on the ink trace - methodological foundations for the forensic analysis of signatures," Ph.D. dissertation, Artifical Instelligence Institute, University of Groningen, 2005.

[6] R. Manmatha and W. Croft, "Word spotting: Indexing handwritten archives." 1997. [Online]. Available: citeseer.comp.nus.edu.sg/manmatha97word.html

[7] N. Otsu, "A threshold selectiion method from gray-level histograms." *Transactions on Systems, Man and Cybernetics*, vol. SMC-9, no. 1, pp. 62–66, Januari 1979.

[8] E. P. R. Plamondon, S. N. Srihari and Q. Montreal, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, vol. 22, no. 1, pp. 63–84, january 2000.

[9] E. H. Ratzlaff, "Inter-line distance estimation and text line extraction for unconstrained online handwriting," *Workshop on Frontiers in Handwriting Recognition*, pp. 33–42, 2000.

[10] A. Vinciarelli and M. Perrone, "combining online and offline handwriting recognition." *IEEE Computer Society*, 2003.

A Appendix

A.1 SPSS output experiment 1

For the retrieval experiment (see Section 3.1) we have done four Independent-Samples T Tests using SPSS 15.0. The test variables were the three writers separately (AAR, ALS and APR) and the data of the three writers together (Sum), the grouping variables were "Match" and "No Match".

The results are shown in Table 1 and 2. For an explanation of the results see Section 4.1.

Table 1: Experiment 1: Group samples

Writer	Type	N	Mean	Std. Deviation	Std. Error Mean
AAR	No Match	761	40.3792	8.03877	.29141
AAR	Match	57	86.4398	6.59679	.87399
ALS	No Match	4693	35.5094	7.39322	.10792
ALS	Match	90	75.9541	9.90381	1.04395
APR	No Match	1822	44.0147	9.28853	.21761
APR	Match	60	83.6665	7.08138	.91420
Sum	No Match	7276	38.1485	8.78815	.10303
Sum	Match	207	81.0770	9.48397	.65918

Table 2: Experiment 1: Independent Samples Test

	Mean		t-test for Equality of Means						
								95%Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Mean	Lower	Upper
AAR	1.450	.229	-42.200	816	.000	-46.06065	1.09148	-48.20308	-43.91822
ALS	.847	.357	-51.031	4781	.000	-40.44476	.79255	-41.99852	-38.89100
APR	7.271	.007	-32.751	1880	.000	-39.65177	1.21069	-42.02621	-37.27732
Sum	.362	.548	-69.145	7481	.000	-42.92842	.62085	-44.14546	-41.71138

A.2 SPSS output experiment 2

For the mapping on pixel level experiment (see Section 3.2) we have done an Paired-Samples T Test using SPSS 15.0. The paired variables were the variables of Pair 1 through Pair 20 of Table 3. "none" means that neither rotation nor resizing was applied, "rot" means that rotation was applied, "res" means that resizing was applied, and "both" means that both rotation and resizing were applied.

The results are shown in Table 3, 4, and 5. For an explanation of the results see Section 4.2.

Table 3: Experiment 2: Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	AAR_none	86.0584	57	6.54878	.86741
	AAR_both	93.2125	57	5.43403	.71975
Pair 2	ALS_none	75.7876	86	10.05858	1.08464
	ALS_both	81.9103	86	10.11166	1.09037
Pair 3	APR_none	82.8609	56	7.02309	.93850
	APR_both	89.4959	56	6.07564	.81189
Pair 4	Sum_none	80.7199	199	9.45155	.67000
	Sum_both	87.2823	199	9.30282	.65946
Pair 5	AAR_none	86.0584	57	6.54878	.86741
	AAR_rot	91.1614	57	6.45671	.85521
Pair 6	AAR_none	86.0584	57	6.54878	.86741
	AAR_res	89.1926	57	5.60702	.74267
Pair 7	ALS_none	75.7876	86	10.05858	1.08464
	ALS_rot	77.4664	86	9.84432	1.06154
Pair 8	ALS_none	75.7876	86	10.05858	1.08464
	ALS_res	78.5324	86	10.00988	1.07939
Pair 9	APR_none	82.8609	56	7.02309	.93850
	APR_rot	88.1973	56	6.38847	.85370
Pair 10	APR_none	82.8609	56	7.02309	.93850
	APR_res	84.3568	56	6.55327	.87572
Pair 11	Sum_none	80.7199	199	9.45155	.67000
	Sum_rot	84.4088	199	10.14154	.71891
Pair 12	Sum_none	80.7199	199	9.45155	.67000
	Sum_res	83.2249	199	9.16591	.64975
Pair 13	AAR_rot	91.1614	57	6.45671	.85521
	AAR_both	93.2125	57	5.43403	.71975
Pair 14	AAR_res	89.1926	57	5.60702	.74267
	AAR_both	93.2125	57	5.43403	.71975
Pair 15	ALS_rot	77.4664	86	9.84432	1.06154
	ALS_both	81.9103	86	10.11166	1.09037
Pair 16	ALS_res	78.5324	86	10.00998	1.07939
	ALS_both	81.9103	86	10.11166	1.09037
Pair 17	APR_rot	88.1973	56	6.38847	.58370
	APR_both	89.4959	56	6.07564	.81189
Pair 18	APR_res	84.3568	56	6.55327	.87572
	APR_both	89.4959	56	6.07564	.81189
Pair 19	Sum_rot	84.4088	199	10.14154	.71891
	Sum_both	87.2823	199	9.30282	.65946
Pair 20	Sum_res	83.2249	199	9.16591	.64975
	Sum_both	87.2823	199	9.30282	.65946

Table 4: Experiment 2: Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	AAR_none & AAR_both	57	.727	.000
Pair 2	ALS_none & ALS_both	86	.958	.000
Pair 3	APR_none & APR_both	56	.760	.000
Pair 4	Sum_none & Sum_both	199	.911	.000
Pair 5	AAR_none & AAR_rot	57	.795	.000
Pair 6	AAR_none & AAR_res	57	.899	.000
Pair 7	ALS_none & ALS_rot	86	.990	.000
Pair 8	ALS_none & ALS_res	86	.968	.000
Pair 9	APR_none & APR_rot	56	.751	.000
Pair 10	APR_none & APR_res	56	.955	.000
Pair 11	Sum_none & Sum_rot	199	.923	.000
Pair 12	Sum_none & Sum_res	199	.962	.000
Pair 13	AAR_rot & AAR_both	57	.918	.000
Pair 14	AAR_res & AAR_both	57	.803	.000
Pair 15	ALS_rot & ALS_both	86	.969	.000
Pair 16	ALS_res & ALS_both	86	.961	.000
Pair 17	APR_rot & APR_both	56	.973	.000
Pair 18	APR_res & APR_both	56	.853	.000
Pair 19	Sum_rot & Sum_both	199	.965	.000
Pair 20	Sum_res & Sum_both	199	.938	.000

Table 5: Experiment 2: Paired Samples Test

		Paired Difference					t	df	Sig. (2-tailed)
		Mean	Std.	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	AAR_none & AAR_both	-7.15404	4.54895	.60252	-8.36103	-5.94704	-11.873	56	.000
Pair 2	ALS_none & ALS_both	-6.12279	2.91192	.31400	-6.74711	-5.49847	-19.499	85	.000
Pair 3	APR_none & APR_both	-6.63500	4.62670	.61827	-7.87404	-5.39596	-10.732	55	.000
Pair 4	Sum_none & Sum_both	-6.56231	3.95286	.28021	-7.11489	-6.00973	-23.419	198	.000
Pair 5	AAR_none & AAR_rot	-5.10298	4.16410	.55155	-6.20780	-3.99810	-9.252	56	.000
Pair 6	AAR_none & AAR_res	-3.13421	2.88741	.38245	-3.90034	-2.36808	-8.195	56	.000
Pair 7	ALS_none & ALS_rot	-1.67884	1.42375	.15353	-1.98409	-1.37358	-10.935	85	.000
Pair 8	ALS_none & ALS_res	-2.74488	2.53426	.27328	-3.28823	-2.20154	-10.044	85	.000
Pair 9	APR_none & APR_rot	-5.33643	4.76946	.63735	-6.61370	-4.05916	-8.373	55	.000
Pair 10	APR_none & APR_res	-1.49589	2.07976	.27792	-2.05286	-.93893	-5.382	55	.000
Pair 11	Sum_none & Sum_rot	-3.68889	3.89762	.27629	-4.23375	-3.14404	-13.351	198	.000
Pair 12	Sum_none & Sum_res	-2.50492	2.59684	.18408	-2.86794	-2.14191	-13.607	198	.000
Pair 13	AAR_rot & AAR_both	-2.05106	2.60834	.34548	-2.74314	-1.35897	-5.937	56	.000
Pair 14	AAR_res & AAR_both	-4.01982	3.47256	.45995	-4.94122	-3.09843	-8.740	56	.000
Pair 15	ALS_rot & ALS_both	-4.44395	2.51819	.27154	-4.98385	-3.90405	-16.366	85	.000
Pair 16	ALS_res & ALS_both	-3.37791	2.80762	.30275	-3.97986	-2.77595	-11.157	85	.000
Pair 17	APR_rot & APR_both	-1.29857	1.48526	.19848	-1.69633	-.90082	-6.543	55	.000
Pair 18	APR_res & APR_both	-5.13911	3.45501	.46169	-6.06436	-4.21385	-11.131	55	.000
Pair 19	Sum_rot & Sum_both	-2.87342	2.68826	.19057	-3.24992	-2.49762	-15.078	198	.000
Pair 20	Sum_res & Sum_both	-4.05739	3.26255	.23125	-4.51343	-3.60135	-17.545	198	.000

A.3 Command-line arguments

- -od: Name of the output-directory (default: output)
- -oe: Filename of the error-file (default: errorFile.txt)
- -df: The de-scale factor (should be 2, 4, 5, 8 or 10 default: 8)
- -nil: The number of interesting locations to recalculate (default: 3)
- -cfc: The compress factor changer (default: 0.00001, only change when program asks for it.)
- -l: The linewidth of the online data (must be an odd number, default: 1)
- -s: The number of surrounding pixels (default: 0)
- -remin: The minimum resize value (must be >0 and <=1, default: 1)
- -remax: The maximum resize value (must be >=1, default: 1)
- -romix: The minimum rotate value (in degrees, default 0)
- -romax: The maximum rotate value (in degrees, default 0)
- -res: The resize step size (default: 0.5)
- -ros: The rotate step size (default: 0.5)
- -nes: Number of errors to save in the error-file (default: 100, use -1 for all results)
- -nic: Number of output images to create (default: 1)
- -debug: Debug mode on (default: off)
- -bw: Use black and white pixels in comparing online and offline (default: only black)

A.4 Javadoc

A complete Javadoc, including the private methods, can be found in the Internship report that belongs to this thesis, see `http://www.student.ru.nl/s.kennedie/OnOffMapping/InternshipReport.pdf`

A.4.1 Class Error

The datastructure that contain all information about a error

Declaration

- public class Error
- extends java.lang.Object
- implements java.lang.Comparable

Fields

- private int x
  - The x-coordinate of the error
- private int y
  - The y-coordinate of the error
- private float resizing
  - The used resizing of the error
- private float rotation
  - The used rotation of the error
- private int nPixelsUsed
  - The number of black pixels used of the pattern
- private float pctCorrect
  - The percentage correct pixels
- private int width
  - The widht of the pattern
- private int height
  - The height of the pattern

Constructors

- `public` **Error**`( float` **_error**`, int` **_x**`, int` **_y**`, float` **_resizing**`, float` **_rotation**`, int` **_nPixelsUsed**`, int` **_width**`, int` **_height** `)`
  - **Usage**
    - Error constructor
  - **Parameters**
    - `_error` - The error
    - `_x` - The x-coordinate
    - `_y` - The y-coordinate
    - `_resizing` - The used resizing
    - `_rotation` - The used rotation
    - `_nPixelsUsed` - The number of black pixels of the pattern
    - `_width` - The width of the pattern
    - `_height` - The height of the pattern

Methods

- `public int` **compareTo**`( java.lang.Object` **o** `)`
  - **Usage**
    - Compares errors for sorting
- `public float` **getError**`( )`
  - **Usage**
    - Returns the error
  - **Returns** - The error
- `public int` **getHeight**`( )`
  - **Usage**
    - Returns the height of the pattern
  - **Returns** - The height of the pattern
- `public int` **getNBlackPixels**`( )`
  - **Usage**
    - Returns the number of black pixels of the pattern
  - **Returns** - The number of black pixels of the pattern
- `public float` **getPctCorrect**`( )`

  - **Usage**
    * Returns the percentage correct pixels
  - **Returns** - The percentage correct pixels
- `public float` **getResizing**`( )`
  - **Usage**
    * Returns the used resizing
  - **Returns** - The used resizing
- `public float` **getRotation**`( )`
  - **Usage**
    * Returns the used rotation
  - **Returns** - The used rotation
- `public int` **getWidth**`( )`
  - **Usage**
    * Returns the width of the pattern
  - **Returns** - The width of the pattern
- `public int` **getX**`( )`
  - **Usage**
    * Returns the x-coordinate
  - **Returns** - The x-coordinate
- `public int` **getY**`( )`
  - **Usage**
    * Returns the y-coordinate
  - **Returns** - The y-coordinate
- `public void` **print**`( )`
  - **Usage**
    * Prints the error information
- `public void` **setError**`( float` **_error** `)`
  - **Usage**
    * Sets the error
  - **Parameters**
    * `_error` - The error

- `public void` **setResizing**`( float` **_resizing** `)`
  - **Usage**
    - Sets the used resizing
  - **Parameters**
    - `_resizing` - The used resizing
- `public void` **setRotation**`( float` **_rotation** `)`
  - **Usage**
    - Sets the used rotation
  - **Parameters**
    - `_rotation` - The used rotation
- `public void` **setX**`( int` **_x** `)`
  - **Usage**
    - Sets the x-coordinate
  - **Parameters**
    - `_x` - The x-coordinate
- `public void` **setY**`( int` **_y** `)`
  - **Usage**
    - Sets the y-coordinate
  - **Parameters**
    - `_y` - The y-coordinate

A.4.2 Class HWRsample

The class that contains the information of a HWRfile

Declaration

- public class HWRsample
- extends java.lang.Object

Fields

- private String label
  - The label of the HWRfile
- private int nCoordinates
  - The number of coordinates in the HWRfile
- private double x
  - The array of x-coordinates of the HWRfile
- private double y
  - The array of y-coordinates of the HWRfile
- private double z
  - The array of z-coordinates (pressure coordinates) of the HWRfile
- private double maxX
  - The max x-coordinate of the HWRfile
- private double maxY
  - the max y-coordinate of the HWRfile
- private double minX
  - the min x-coordinate of the HWRfile
- private double minY
  - the min y-coordinate of the HWRfile

Constructors

- HWRsample `public` **HWRsample**`( xor.HWRsample` **fromHWR** `)`
  - **Usage**
    - Copies a HWRsample
  - **Parameters**
    - `fromHWR` - The HWRsample to copy
- `public` **HWRsample**`( java.lang.String` **filename** `)`
  - **Usage**
    - Creates a HWRsample from a certain HWRfile
  - **Parameters**
    - `filename` - The HWRfile to create the HWRsample from

Methods

- `public double` **getHeight**`( )`
  - **Usage**
    - Returns the height of the HWRfile
  - **Returns** - The height of the HWRfile
- `public int` **getNCoordinates**`( )`
  - **Usage**
    - Retruns the number of coordinates of the HWRfile
  - **Returns** - The number of coordinates of the HWRfile
- `public double` **getWidth**`( )`
  - **Usage**
    - Returns the width of the HWRfile
  - **Returns** - The width of the HWRfile
- `public double` **getX**`( int` **i** `)`
  - **Usage**
    - Returns the x-value of the i-th coordinate
  - **Parameters**
    - `i` - The coordinate number

  - **Returns** - The x-value of the i-th coordinate
- `public double` **getY**`( int  i )`
  - **Usage**
    - Returns the y-value of the i-th coordinate
  - **Parameters**
    - `i` - The coordinate number
  - **Returns** - The y-value of the i-th coordinate
- `public double` **getZ**`( int  i )`
  - **Usage**
    - Returns the z-value of the i-th coordinate
  - **Parameters**
    - `i` - The coordinate number
  - **Returns** - The z-value of the i-th coordinate
- `public void` **resize**`( float  resizeFactor )`
  - **Usage**
    - Resizes a HWR file with resizeFactor.
  - **Parameters**
    - `resizeFactor` - The factor to resize the HWR file with
- `public void` **rotate**`( float  degrees )`
  - **Usage**
    - Rotates Rotates an HWR file over n degrees.
  - **Parameters**
    - `degrees` - The rotation in degrees.

A.4.3 Class MatrixImage

The datastructure that is used for calculating the error

Declaration

- public class MatrixImage
- extends java.lang.Object

Fields

- private float matrix
  - The matrix that represents the image or the pattern
- private int width
  - The widht of the matrix
- private int height
  - The height of the matrix

Constructors

- MatrixImage `public` **MatrixImage**`( xor.MyImage` **image** `)`
  - **Usage**
    - Converts a MyImage to a MatrixImage
  - **Parameters**
    - `image` - The image of the type MyImage that is converted to a MatrixImage.

Methods

- `public static void` **createErrorImageBruteForce**`(` `xor.MatrixImage` **online**, `xor.MatrixImage` **offline**, `xor.MyHashMap` **map**, `float` **resizeFactor**, `float` **pruneFactor** `)`
  - **Usage**
    - Tries to find the location of the small image in the big image.
  - **Parameters**
    - `small` - The image to find in the big image. (the online image)

        * `big` - The image that is being searched through. (the offline image)
- `public static void` **createRealSizeErrorImage(** `xor.MatrixImage` **onlineRS,** `xor.MatrixImage` **offlineRS,** `xor.MyHashMap` **mapToProcess,** `xor.MyHashMap` **processedMap,** `boolean` **bw)**
  - **Usage**
    * Tries to find the pattern in the image.
  - **Parameters**
    * `onlineRS` - The pattern.
    * `offlineRS` - The image.
    * `mapToProcess` - The map with the interesting locations.
    * `processedMap` - The map with the interesting locations and their error.
    * `bw` - Whether or not to calculate black and white pixels of the pattern.
- `public int` **getHeight( )**
  - **Usage**
    * Returns the height of the MatrixImage
  - **Returns** - the height of the MatrixImage
- `public float` **getHSV_V(** `int` **x,** `int` **y)**
  - **Usage**
    * Retruns the HSV-value of 'pixel' (x,y)
  - **Parameters**
    * `x` - x-coordinate of the pixel.
    * `y` - y-coordinate of the pixel.
  - **Returns** - HSV-value of the pixel.
- `public int` **getWidth( )**
  - **Usage**
    * Returns the width of the MatrixImage
  - **Returns** - the width of the MatrixImage

A.4.4 Class MyHashMap

The datastructure that contains the errorvalues of the locations

Declarations

- public class MyHashMap
- extends java.lang.Object

Constructors

- MyHashMap `public` **MyHashMap**`( )`
  - **Usage**
    - default constructor, does nothing. Used for creating a MyHashMap that is being filled by a function like "createErrorImage()".
- `public` **MyHashMap**`( java.lang.String` **filename** `)`
  - **Usage**
    - Loads a map from a filename.
  - **Parameters**
    - `filename` - The file to load.

Methods

- binarizeMap `public void` **binarizeMap**`( )`
  - **Usage**
    - Binarizes the map, throws away the values by replacing them with the minErrorValue. All these points will be recalculated in the realsize image.
- `public void` **compressMap**`( float` **compressFactor** `)`
  - **Usage**
    - Deletes uninteresting points (points that hava an error-value that is to big, points that are to dark) form the map.
  - **Parameters**
    - `compressFactor` - The factor to compress the map with. Must be <1.0. 1.70 is a good value for this.
- `public void` **determineImageSize**`( )`

  - **Usage**
    * Determines the size of the map and stores this in mapWidth and mapHeight. These values are used in writing the map to an image.
- `public void` **determineMinAndMaxError**`( )`
  - **Usage**
    * Determines the biggest error found in the map and stores this in maxMapErrorValue. This value is used for writing the map to an image.
- `public float` **get**`( java.awt.Point` **point** `)`
  - **Usage**
    * Returns the value to which this map maps the specified Point.
  - **Parameters**
    * `point` - Point whose associated value is to be returned.
  - **Returns** - The value of the point.
- `public int` **getHeight**`( )`
  - **Usage**
    * Returns the Height of the map.
  - **Returns** - the Height of the map.
- `public float` **getLightestPixel**`( )`
  - **Usage**
    * Returns value of the lightest pixel.
  - **Returns** - The value of the lightest pixel.
- `public int` **getSize**`( )`
  - **Usage**
    * Returns the size of the map, in other words the number of interesting points.
  - **Returns** - The size of the map, in other words the number of interesting points.
- `public int` **getWidth**`( )`
  - **Usage**
    * Returns the width of the map.

  - **Returns** - the width of the map.
- `public void` **load**`( java.lang.String` **filename** `)`
  - **Usage**
    - Loads a map from a filename.
  - **Parameters**
    - `filename` - The file to load.
    - `map` - The map to load the file in.
  - **Exceptions**
    - `java.io.IOException` -
- `public void` **printMap**`( )`
  - **Usage**
    - Prints the map to the schreen: [x] [y] [value]
- `public void` **put**`( java.awt.Point` **point**`, java.lang.Float` **value** `)`
  - **Usage**
    - Associates the specified value with the specified Point in this map.
  - **Parameters**
    - `point` - Point with which the specified value is to be associated.
    - `value` - Value to be associated with the specified key.
- `public void` **resizeMap**`( int` **scale**`, xor.MatrixImage` **offline**`, xor.MatrixImage` **online** `)`
  - **Usage**
    - Resizes a map to a map with the dimensions of the original offline image.
  - **Parameters**
    - `scale` - The scalesize (must be >1).
  - **Returns** - The resized map.
- `public void` **save**`( java.lang.String` **fileName** `)`
  - **Usage**
    - Saves the map in the format [x] [y] [error-value].

- **Parameters**
  - `fileName` - The name of the ouptufile.

- `public void` **saveBest**`( int` **n**, `float` **resizing**, `float` **rotating**, `int` **removedWidth**, `java.lang.String` **OUTPUT_DIR**, `java.lang.String` **OUTPUT_ERROR_FILE**, `int` **l**, `int` **s**, `java.lang.String` **pngFile**, `java.lang.String` **hwrFile** `)`
  - **Usage**
    - Saves the n best locations of the current resizing and rotating to a file
  - **Parameters**
    - `n` - The number of locations that have to be saved
    - `resizing` - The current resize factor
    - `rotating` - The corrent rotate factor
    - `removedWidth` - The widht removed in preprocessing the offline image
    - `OUTPUT_ERROR_FILE` - The filename of the errorFile
    - `l` - Linewidth
    - `s` - Number of surrounding pixels
    - `pngFile` - The offline filename
    - `hwrFile` - The offline filename
  - **Exceptions**
    - `java.io.IOException` -
- `public void` **surroundInterestingPoints**`( int` **bw** `)`
  - **Usage**
    - Add points that surround the interesting points of the map to the map
  - **Parameters**
    - `bw` - The widht of the border that is added around interesting points
- `public void` **writeBinarizedMapToImage**`( java.lang.String` **filename** `)`
  - **Usage**
    - Writes a binarized map to an image.

- **Parameters**
  - `filename` - The filename of the image.

- `public void` **writeMapToImage**`( java.lang.String` **filename** `)`
  - **Usage**
    - Writes a map to an image.
  - **Parameters**
    - `filename` - The filename of the image.

A.4.5 Class MyImage

The class that contains the image and its information.

Declaration

- public class MyImage
- extends java.lang.Object

Fields

- private BufferedImage img
  - The image.
- private static final int BACKGROUND_COLOR
  - The color of the background.
- private static final float BLACK
  - The value of black.
- private static final float GRAY
  - The value of grey.
- private static final float WHITE
  - The value of white.
- private static final int RGB_RED
  - The value of red.
- private int removedWidth
  - The number of removed pixels in the width of the image.

Constructors

- MyImage `public` **MyImage**`( xor.HWRsample` **hwr**`, int` **lineWidth**`, float` **resizeFactor** `)`
  - **Usage**
    - Creates a MyImage from a HWRsample with a certain lineWidth an a certain scaling factor.
  - **Parameters**

        * `hwr` - The HWRsample to create a MyImage from.
        * `lineWidth` - The lineWidth of the image.
        * `resizeFactor` - The scaling factor.
- `public` **MyImage**`( int` **w**`, int` **h** `)`
    - **Usage**
        * Constructs an empty image with the backgroundcolor with a certain width and height.
    - **Parameters**
        * `w` - The width of the new image.
        * `h` - The height of the new image.
- `public` **MyImage**`( xor.MatrixImage` **matrix** `)`
    - **Usage**
        * Converts a MatrixImage to a MyImage.
    - **Parameters**
        * `matrix` - The MatrixImage that is converted to a MyImage.
- `public` **MyImage**`( xor.MatrixImage` **matrix**`, int` **x**`, int` **y**`, int` **width**`, int` **height** `)`
    - **Usage**
        * Creates a MyImage from a part of a MatrixImage
    - **Parameters**
        * `matrix` - The MatrixImage
        * `x` - The x-coordinate of the upperleft corner of the part of the MatrixImage
        * `y` - The y-coordinate of the upperleft corner of the part of the MatrixImage
        * `width` - The width of the part of the MatrixImage
        * `height` - The height of the part of the MatrixImage
- `public` **MyImage**`( xor.MyImage` **fromImage** `)`
    - **Usage**
        * Creates a new image from a certain image.
    - **Parameters**
        * `fromImage` - The image to create a new image from.
- `public` **MyImage**`( java.lang.String` **filename** `)`

- **Usage**
  - Constructs and crops an image from a file.
- **Parameters**
  - `filename` - The file to create an image from.

Methods

- `public void` **binarizeImage**`( float` **thresshold** `)`
  - **Usage**
    - Creates a black-white image.
  - **Parameters**
    - `thresshold` - The higher the thresshold the more pixels become black.
- `public static void` **createErrorImageBruteForce**`( xor.MyImage` **online**`, xor.MyImage` **offline** `)`
  - **Usage**
    - Tries to find the location of the small image in the big image, and writes this to an errorImage.
  - **Parameters**
    - `online` - The image to find in the big image.
    - `offline` - The image that is being searched through.
- `public void` **crop**`( )`
  - **Usage**
    - Creates a sub-image that doesn't have a white framing.
- `public void` **crop**`( int` **minX**`, int` **minY**`, int` **width**`, int` **height** `)`
  - **Usage**
    - Creates a sub-image.
  - **Parameters**
    - `minX` - The x-coordinate of the upper-left corner of the sub-image.
    - `minY` - The y-coordinate of the upper-left corner of the sub-image.
    - `width` - The width of the sub-image.

        * `height` - The height of the sub-image.
- `public void` **drawLine**`( int` **x0**`, int` **y0**`, int` **x1**`, int` **y1**`, int` **imageWidth**`, int` **imageHeight**`, int` **lineWidth** `)`
    - **Usage**
        * Draws a line from (x0,y0) to (x1,y1).
    - **Parameters**
        * `x0` - The X-coordinate to start from.
        * `y0` - The Y-coordinate to start from.
        * `x1` - The X-coordinate to end with.
        * `y1` - The Y-coordinate to end with.
        * `imageWidth` - The widht of the image in which the line comes (needed to prevent drawing lines outside the image).
        * `imageHeight` - The height of the image in which the line comes (needed to prevent drawing lines outside the image).
        * `lineWidth` - The width of the line (must be na odd number).
- `public int` **getBlue**`( int` **x**`, int` **y** `)`
    - **Usage**
        * Returns the Blue-value of a pixel.
    - **Parameters**
        * `x` - x-coordinate of the pixel.
        * `y` - y-coordinate of the pixel.
    - **Returns** - Blue-value of the pixel.
- `public int` **getGreen**`( int` **x**`, int` **y** `)`
    - **Usage**
        * Returns the Green-value of a pixel.
    - **Parameters**
        * `x` - x-coordinate of the pixel.
        * `y` - y-coordinate of the pixel.
    - **Returns** - Green-value of the pixel.
- `public int` **getHeight**`( )`
    - **Usage**
        * Returns the height of the image.

  - **Returns** - The height of the image.
- `public float` **getHSV**`( int x, int y )`
  - **Usage**
    - Returns the HSV value of a pixel.
  - **Parameters**
    - `x` - The x-coordinate of the pixel.
    - `y` - The y-coordinate of the pixel.
  - **Returns** - The HSV value of the pixel.
- `public int` **getRed**`( int x, int y )`
  - **Usage**
    - Returns the Red-value of a pixel.
  - **Parameters**
    - `x` - x-coordinate of the pixel.
    - `y` - y-coordinate of the pixel.
  - **Returns** - Red-value of the pixel.
- `public int` **getRemovedWidth**`( )`
  - **Usage**
    - Returns the removed width
  - **Returns** - The removed widht
- `public int` **getRGB**`( int x, int y )`
  - **Usage**
    - Returns the RGB value of a certain pixel.
  - **Parameters**
    - `x` - x-coordinate of the pixel.
    - `y` - y-coordinate of the pixel.
  - **Returns** - RGB value of the pixel.
- `public int` **getWidth**`( )`
  - **Usage**
    - Returns the width of the image.
  - **Returns** - The width of the image.
- `public int` **HSVtoRGB**`( float h, float s, float b )`

  - **Usage**
    * Converts a HSB-value to a RGB-value
  - **Parameters**
    * `h` - H-value
    * `s` - S-value
    * `b` - B-value
  - **Returns** - RGB-value
- `public void` **mergeImages**`( xor.MyImage` **online** `)`
  - **Usage**
    * Merges the online image in the offline image
  - **Parameters**
    * `online` - The online image to merge in the offline image
- `public float` **otsu**`( )`
  - **Usage**
    * Returns the otsu-value of the image
  - **Returns** - The otsu-value of the image
- `public void` **removeGray**`( float` **threshold** `)`
  - **Usage**
    * Removes pixels that have a gray-value below thresshold
  - **Parameters**
    * `threshold` - thethreshold
- `public void` **resizeTo**`( int` **w**`, int` **h** `)`
  - **Usage**
    * Resizes the image to a certain widht an hight
  - **Parameters**
    * `w` - The widht to resize the image to
    * `h` - The height to resize the image to
- `public void` **resizeTo**`( xor.MyImage` **image** `)`
  - **Usage**
    * Resizes the mimage to the size of the argument image.
  - **Parameters**

        * `image` - The image who's size is used to resize the image to.
- `public void` **`rotate( double  degrees )`**
    - **Usage**
        * Rotates an image with a certain number of degrees
    - **Parameters**
        * `degrees` - Number of degrees to rotate
- `public void` **`rotateRad( double  theta )`**
    - **Usage**
        * Rotates an image with theta Radials.
    - **Parameters**
        * `theta` - The angle of rotation in radians.
- `public void` **`scale( double  scale )`**
    - **Usage**
        * Scales an image with a certain factor.
    - **Parameters**
        * `scale` - Scale-factor.
- `public void` **`scale( int  w, int  h )`**
    - **Usage**
        * Scales an image to a certain width an height.
    - **Parameters**
        * `w` - Number of pixels in the width.
        * `h` - Number of pixels in the height.
- `public void` **`selectFirstQuadrant( )`**
    - **Usage**
        * Crops the image to the upper half of the first quadrant, the regio that contains the unique code
- `public void` **`setRGB( int  x, int  y, int  rgb )`**
    - **Usage**
        * Sets the RGB value of a pixel.
    - **Parameters**
        * `x` - x-coordinate of the pixel.

        * `y` - y-coordinate of the pixel.
        * `rgb` - RGB-value of the pixel.
- `public void` **`write( java.lang.String  name )`**
    - **Usage**
        * Writes the image as a PNF file to "name"
    - **Parameters**
        * `name` - Filename.
- `public void` **`write( java.lang.String  name, java.lang.String  type )`**
    - **Usage**
        * Writes the image to a file, and writes the filename to the screen.
    - **Parameters**
        * `name` - The name of the output file.
        * `type` - The type of the output file.
- `public void` **`writeToFile( )`**
    - **Usage**
        * Writes the image to output.png.
- `public static void` **`writeXorImage( xor.MyImage  input1, xor.MyImage  input2 )`**
    - **Usage**
        * Writes the Xor-image of two images to a file named XorImage.png.
    - **Parameters**
        * `input1` - Image1.
        * `input2` - Image2.

A.4.6 Class Stopwatch

A class to help benchmark code It simulates a real stop watch

Declaration

- public class Stopwatch
- extends java.lang.Object

Fields

- private long startTime
- private long stopTime
- private boolean running

Constructors

- Stopwatch `public` **Stopwatch**`( )`

Methods

- getElapsedTime `public long` **getElapsedTime**`( )`
  - **Usage**
    - returns elapsed time in milliseconds if the watch has never been started then return zero
- `public Stopwatch` **reset**`( )`
  - **Usage**
    - resets the stopwatch
  - **Returns** - The stopwatch
- `public Stopwatch` **start**`( )`
  - **Usage**
    - Starts the Stopwatch
  - **Returns** - The stopwatch
- `public Stopwatch` **stop**`( )`
  - **Usage**
    - Stops the stopwatch
  - **Returns** - The stopwatch

A.4.7 Class Xor

The main programm Finds the optimal rotation en resizing for a image and a pattern.

Declaration

- public class Xor
- extends java.lang.Object

Fields

- private static final int HWR_LINE_WIDTH_RESIZED
  - The linewidth of the resized pattern.
- private static final float BINARIZATION_THRESHOLD
  - The binarization threshold.
- private static final float RESIZE_FACTOR_HWR_REAL_SIZE
  - The relation between the resolution of the scan and the hwr.
- private static final float PRUNE_FACTOR
  - The prune factor, used to throw away uninteresting locations.
- private static float COMPRESS_FACTOR
  - The compress factor used to compress the map (in other words: to delete uninteresting points).
- private static String OFFLINE_FILENAME
  - The filename of the offline data. THis should be a png-file.
- private static String ONLINE_FILENAME
  - The filename of the online data. This should be a hwr-file.
- private static int HWR_LINE_WIDTH_REAL_SIZE
  - The linewidth of the pattern.
- private static float RESIZE_FACTOR_OFFLINE_SHRINK
  - The resize factor for the offline data, used for computational reasons.

- private static int RESIZE_FACTOR_OFFLINE_ENLARGE
  - The resize factor used to scale the map with the errors up.
- private static float RESIZE_FACTOR_HWR_SHRINK
  - The resize factor for the online data, used for computational reasons.
- private static float COMPRESS_FACTOR_CHANGER
  - Factor that adds up to the compressfactor.
- private static int N_LOCATIONS_TO_RECACULATE
  - Number of interesting locations to recalculate.
- private static int N_SURROUNDING_PIXELS
  - The number of surrounding pixels used.
- private static float RESIZE_MAX
  - The default maximum resize factor.
- private static float RESIZE_MIN
  - The default minimum resize factor.
- private static float ROTATE_MAX
  - The default maximum rotation.
- private static float ROTATE_MIN
  - The default minimum rotation.
- private static float RESIZE_STEP_SIZE
  - The default resize step size.
- private static float ROTATE_STEP_SIZE
  - The default rotate step size.
- private static String OUTPUT_DIR
  - The default ouptutdirectory.
- private static String OUTPUT_IMAGE_FILE
  - The default name of the output image.

- private static String OUTPUT_ERROR_FILE
  - The default name of the error file.
- private static boolean DEBUG
  - Whether or not debug mode is on.
- private static boolean EDGE_DETECTION
  - Whether or not edge detection is on.
- private static boolean BLACK_AND_WHITE
  - Wether or not to calculate black and white pixels of the pattern.
- private static int N_ERRORS_TO_SAVE
  - Number of errors to save in the error file.
- private static int N_IMAGES_TO_CREATE
  - Number of output images to create.

Constructors

- Xor `public` **Xor**`( )`

Methods

- `public static void` **main**`( java.lang.String [] args )`
  - **Parameters**
    - `args` - The parameters of the command line
  - **Exceptions**
    - `java.io.IOException` -

A.4.8 Class Merge

Creates an image with both the online as the offline data.

Declaration

- public class Merge
- extends java.lang.Object

Fields

- private static final float RESIZE_FACTOR_HWR_REAL_SIZE
  - The relation between the resolution of the scan and the hwr.
- private static String ERROR_FILENAME
  - The filename of the error file used.
- private static int N_IMAGES_TO_CREATE
  - The number of images to create.
- private static String OFFLINE_FILENAME
  - The filename of the offline data.
- private static String ONLINE_FILENAME
  - The filename of the online data.
- private static String OUTPUT_DIR
  - The default outputdirecory
- private static String OUTPUT_IMAGE_FILE
  - The default filename of the output image.
- private static int LINE_WIDTH
  - The used linewidth
- private static int N_SURROUNDING_PIXELS
  - The used number of surrounding pixels
- private static int N_ERRORS_IN_FILE
  - The number of errors in the file

Constructors

- Merge `public` **Merge**`( )`

Methods

- `public static void` **main**`( java.lang.String [] args )`
  - **Parameters**
    - `args` -
  - **Exceptions**
    - `java.io.IOException` -