For the testing images, we want to divide them into pictures of single words so that we can recognize them individually. We want to keep the information of different words and lines. So we do not want to just get all the components directly. The way is mainly divide each line from a paragraph, then divide each word from a line, and finally divide each letter from a word.

Suppose we just take a picture to the things we want to recognize and save it into the laptop. We load the image and it is a 3D matrix. Then we turn the picture into grayscale. Since we do not need the color information of the picture, we turn the gray image into a binary image. Therefore, we got all the pixels that have words are valued one, and others are valued zero. However, there might be some noises in the picture. So we throw away the connect components that is too small. We set the threshold so that we can remove the noises and keep the parts we want, like the dot of “i”. Then we get a bounding box for all the non zero pixels to unify the format of the image.

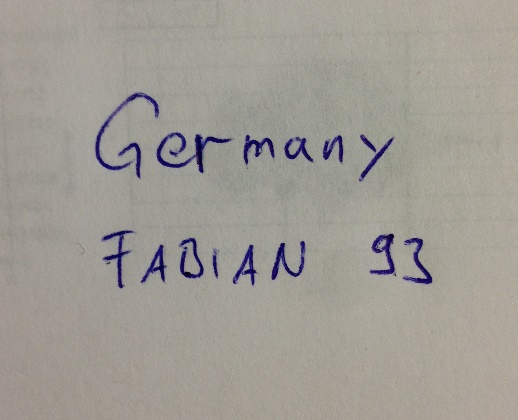


Figure #Unknown: Picture taken from handwriting

With the thresholds, we can divide our images. We want to first separate them letters in lines. When we divide the lines, we get a list of the average position of each component on x-axis. For the letters on the same line, the value should be about the same. So the different should not be large. If the different between two consecutive values exceeds a certain threshold, we will divide them to be separate lines.

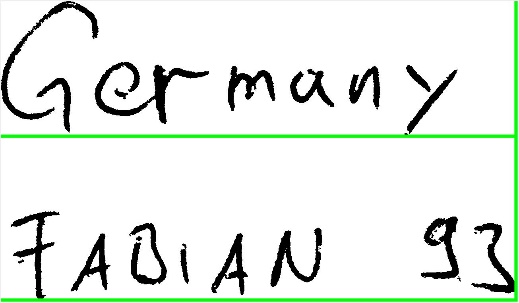


Figure #Unknown: Bounding boxes for separating lines.

After getting a picture of one line, we would like to divide the line into words. We would want to first throw small pixels from the picture again, because it is possible that when we crop the images, we accidental also cropped a small part of other letters not in this line. We do not want to count them. The method is dividing is similar. This time we will get their average position on y-axis. The value should be increase constantly, if it is in the same word. When the interval suddenly have a large increase, and the difference is larger than a certain value, we believe it is the space between words and will treat them as different words.



Figure #Unknown: Bounding boxes for separating words using the result of the second line.

Then we get a picture that only contains one word in it. We do the same morphology to throw out small components. After that, we may just use connect components to figure each word out and crop them by bounding boxes. But not all words are made up by only one component. For example, “i”, “j” are letters that are made by two parts. This time we get the average position on x-axis again for all the pixels that are not 0. We calculate the average of the average values and treat it as a standard line for a letter. If a letter is only made by one component, its average position should be about the standard we set. If we find one component has an average that is really far away from it, it is probably the dot of “i” or “j”. We will combine it with the nearest component according to the position on y-axis.

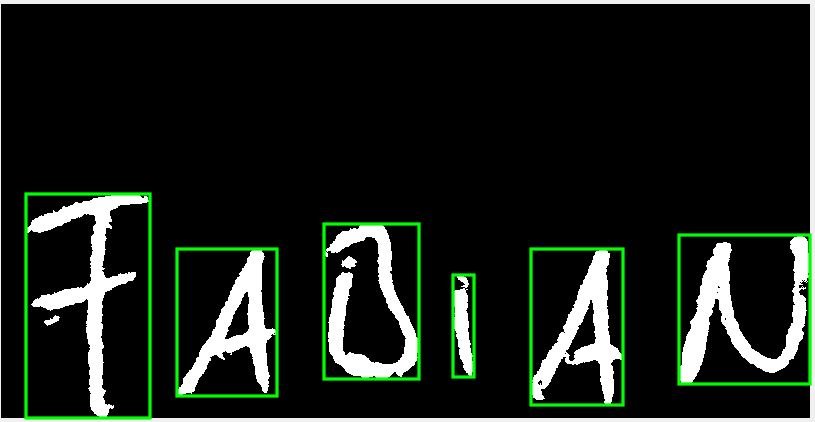


Figure #Unknown: Bounding boxes for separating letters using the result of the first word.

When we store our pictures, we also follow the flow of entire picture, lines, words, and letters. These are stored in subfolders that look like a tree structure. When we get the images, we will know which part they belong to.

Improve proceeding

Currently, our segmentation algorithm works most of the pictures. However, manually change the threshold is often needed to get a good result. The preprocessing of the images could be improved regarding the following specific topics.

**1)   Thin out images, implement hit and miss**

Implementation of existing approaches to be able to reach higher accuracy on a broader variety of handwriting styles.

**2)   Images with different resolution. Find a way to detect areas with characters in the beginning**

How to set flexible thresholds.

What is the benefit of doing so? Robust!

If we want to apply our technique to real use, for example, take some pictures for the words we wrote and turn them into a text file in computer. Our method depend on reasonable thresholds to tell the program the standard of dividing letters. However, it is obvious that using fixed thresholds is not a reasonable way. For each input image, the situation might be different. We really do not want to change the threshold every single time when we are working on a new picture. We want the program can set a reasonable threshold automatically.

We tried some methods to determine our thresholds for lines, words, and letters. We added one input called lines to show the number of rows in the picture. Therefore, we may calculate our threshold for lines by the number of rows and size of the picture. Then we assume that the word threshold is in proportion to the line threshold, because people usually wants to keep the intervals. So we set a coefficient that times the line threshold to calculate it. Then we assume again that the interval between words is in proportion to the size of the letters .So we would like to apply the same technique on the letter threshold.

But it turns out that it is also heavily depend on the style of the writer. The problem is that the coefficients vary on different writing styles. For one style set the word threshold to be 1.3 times the line threshold gives a great result, but for others 2 will be a more reasonable coefficient.

We also tried to resize and pad the input image to make all the inputs have the same size. But this still has a similar problem with the previous method.

A possible method is to include the information about the portion of the black pixels in the picture. If it is small, there may be less word on the picture. The words may be more separated, and we should set a large threshold. But this may not work well on pictures that only have several letters on it, because it might be influenced by the size of each letter.

Another way is to get some pre-defined the styles, and choose the style of the picture to determine the coefficients we want to use.

**3)   Split the connected characters into single ones using specific approaches for that.**

Since we are using the connected components to figure out the boundary box for each letter, if any letters are connected, we will recognize them as one letter. We assume that all the letters should be “slim”. So the row of its bounding box should be larger than its column. If any bounding box we get is not the case, we may divide them.

Another possible way is count the number of non-zero pixels in each row. If the number is really small, we may assume that it is just a connection between two letters, and therefore we can divide them.