Project 3 documentation

FCV Lab

INTRODUCTION

In this project I developed an algorithm which extracts lines and characters from a handwritten text. I tested the algorithm on 3 different handwriting photos and the results are quite impressive, considering that I only used simple computer vision techniques and opencv. For sure, a neural network would lead to much better results for this specific task.

ALGORITHM

I will present the algorithm used for line extraction and characters extraction respectively.

1. Lines extraction

Lines extraction algorithm	
Step 1	Reading the image with the handwritten text.
Step 2	Transforming the image to grayscale
Step 3	Performing a inverse binary thresholding with a thresh value of 120.
Step 4	Dilating the image in order to unify the characters and the words in a single line. The kernel is (50,1) (height < 1000) and (50,2) (height > 1000) and the number of iterations is 7. The value is bigger for X axis, because I wanted to expand horizontally. The value is smaller on the Y axis because I do not want to unify 2 separate lines verically.
Step 5	I apply findContours function.
Step 6	I filter the contours by size and discard those which have an area smaller than 10000 pixels.
Step 7	I apply minAreaRect function in order to find the coordinates of a bounding box
Step 8	I determine the box points of this rectangle.
Step 9	I perform a perspective transform and dewarp the line. This way I can detect lines of handwriting even if the they are not straight.
Step 10	I check the angle of the detected rectangle and if needed, I perform a counterclockwise 90 degrees rotation.
Step 11	I save the line as a new image to the disk and then I also draw the found line on the original handwritten text.

2. Characters extraction

As far as the characters extraction is concerned, the used algorithm is very similar to the one used for lines extraction. However, in order to identify individual characters, I had to make some tweaks to the original algorithm.

In the table below I will present only those tweaks, for space and simplicity reasons.

Characters extraction algorithm	
Tweak 1	First I tweak the thresholding parameters; I increase the thresh value from 120 to 130. On top of that, after I use binary inverse thresholding, I also apply Otsu's binarization. This is useful because this method finds an optimal global threshold value from the image histogram and it can drastically reduce noise around the characters.
Tweak 2	I add an eroding step, in order to get hold of individual characters. This way thin lines between 2 distinct characters will disappear. I use a (2,1) kernel and 1 iteration.
Tweak 3	I change the dilation kernel to (3,3) (height > 1000) and (1,1) (height < 1000). This way I make sure that the characters will be visible enough, but without touching each other.
Tweak 4	I also change a parameter for contour finding: cv2.RETR_EXTERNAL instead of cv2.RETR_TREE. This way I avoid finding contours in contours.
Tweak 5	I sort the contours by their leftmost coordinate because I want to draw them on the image in the right order, from left to right.
Tweak 6	I draw rectangles around the coordinates of the characters, without using a perspective transform this time.
Tweak 7	I draw rectangles both on line images and on the original handwritten text image.

CONCLUSION

As a conclusion, I would like to say that the method that I created performs better than expected. It manages to detect lines of handwritten text, regardless of how slanted it is. However, it still has some downsides when it comes to characters extraction. Sometimes it fails to extract individual characters and sees a group of them as one. This happens because the characters are still connected, even after the erosion step.

I tried an extra step of erosion or even a bigger kernel in order to solve this problem, but then the algorithms fails to identify other characters that are individual or are not connected to other characters.

I believe that my methods finds the perfect tradeoff between the 2 problems stated above.