Lettercademy: learn to write letters.
Image comparison using shapes and contours

Marta Pantaleeva
guspantama@student.gu.se

Abstract

This paper explores a method of comparing images. The method consists of finding and extracting contours from the images, and comparing their shapes. This way we could analyze the similarities or differences between the images and provide a feedback to what extent they are similar or not. This method is applied to the simple scenario of learning to write letters or improving one's handwriting.

1 Introduction

The main motivation behind this project is the general interest in innovating and computerizing old methods for the sake of society's progression in a more modern world. Nowadays, research has turned its focus on making everyday tasks automatic thanks to technology. In this project, we are experimenting with the very simple scenario of how people could learn to write letters or even improve their handwriting. We deal with images of letters and, thus, computer vision and image processing techniques will be relevant in our implementation. The main task is to provide feedback about an image of a handwritten letter, by comparing its shape with the shape of a gold standard, model letter.

2 Implementation

For this project, we have built a game-like system that provides feedback or tries to guess which letter the user has written. The implementation consists of three parts: building the gold standard dataset of letters, finding, processing and comparing letter's shapes, and designing the system's rules and feedback.

2.1 Gold standard dataset

The dataset of model letters was build manually with Microsoft's Paint. The letters are white on a black background. This arrangement was chosen because of the need of **thresholding** later in the implementation, which is a segmentation method to separate the object of interest in the image from the background. Calibri font was mainly use, but also Arial (for letter 'y') and Segoe Print (for letters 'a','g' and 'j') all in size 120. The dimensions of the images are all quite small, so that only one letter can fit in one image: 131x151 for the short letters and 122x179 for the tall letters. There are few exceptions - 156x213 for 'g', 119x213 for 'j', 200x144 for 'm' and 'w', and 134x198 for 'y'. The files are all in PNG format and they are annotated after their corresponding letters.

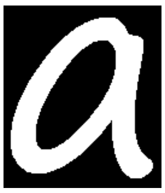
Figure 1: Sample of a gold standard image of letter 'a'

2.2 Comparison method

The main image processing library that is used in the implementation is OpenCV. We start with building a function that reads two images, converts them into gray-scale, then converts the gray-scale images into thresholded images which are then used to find the shapes or **contours** of the letters in the two images. The function returns the original and gray-scale of the two images and their contours.

Next, we build a function to compare the extracted contours from the previous function. **Contours** is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object. For the comparison of letters, only their outer contour is needed, so we select only the first Numpy array list which corresponds to the outer contour of the letter. We use cv2.matchShapes to compare the two contours. This function will return a float value that will represent the similarity or difference between both contours. A perfect match would be 0.0 and the bigger the number gets the more dissimilar the contours are. The matching result from the following model and handwritten letters is 0.07. However, we do not compare the contours only in their totality. In the same function, we split each outer contour of both letters to define four regions: the left part, the right part, the top part and the bottom part of the contour (or letter). Thus, cv2.matchShapes is also applied to compare the left part of one letter with the left part of the other letter, etc. The function returns the all the matching

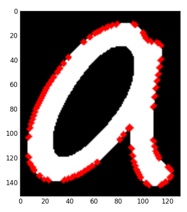
Figure 2: Example of outer contour of model letter ’a’

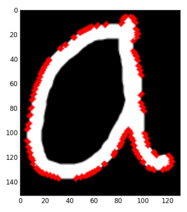
Figure 3: Example of outer contour of handwritten letter ’a’

scores and all the parts of the outer contours of the two letters in a total of 15 elements.

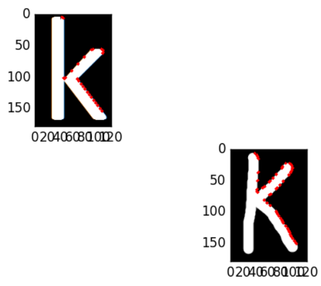
Figure 4: Right part of contour of model and handwritten letters ’k’

Finally, like the images above, we define a plotting function that will plot the images of the two letters and their contours (or parts of contour) in red for visual comparison that would help the user to see where his/her handwritten letter differs from the model letter. It will be used in the game-like program for providing feedback.

2.3 Lettercademy

The final part of the implementation of this project is ’Lettercademy’. It is a game-like program designed to assess users’ handwritten letter or guess it.

2.4 The feedback game

The first part of ’Lettercademy’ is a program function that gives a feedback about a letter. It uses all the functions of the comparison method explained previously. First of all, it asks the user to select a model letter to which he/she wishes to compare his or hers handwritten letter. Then, the system asks about the name of the user’s letter image and finally provides a feedback. There are four kinds of feedback from ”your letter is perfect” to ”you should try again”. The feedback that will be returned depends on the score of the contour comparison between the gold standard letter and the user’s letter. Remember that we are comparing the entire outer contours and also each of the four parts explained above.

For each kind of feedback we set a condition with a threshold (that was decided on after many experiments), which means four conditions in total. The first one is that if the score of the comparison of the contours in their totality is less than 0.1, the user gets the best possible feedback.

You letter ’a’ is perfect! Congratulations!

The second condition is that if the score of the comparison of the contours is between 0.1 and 0.5 **or** if neither of the scores of the comparison between the individual parts (left, right, top, bottom) is above 1 or only one of them is above 1 , the user gets the second feedback which is ”your letter is OK”. However he/she also receives extra feedback concerning the part of the letter with the highest score: ”you should check the left/right/top/bottom part of your letter” and he/she is also provided with the visual aid mentioned above.

You letter ’a’ is OK!
However, you should improve, a bit, the left part of your letter.

The third condition is that if the total score is between 0.1 and 1 **and** more than one of the scores of the individual parts is above 1, the user gets the third feedback which says ”your letter need improvement” and also says which is the part with highest score that needs improvement the most and shows again a plot of the two letters and the contour part in red.

You letter ’a’ needs improvement!
You should check the left part of your letter.

The fourth condition concerns letters that are not good enough and need to be rewritten. If the user gets the fourth feedback the system would say ”you should try again” and show a plot of the user’s letter and the contour shape of the model letter on it.

2.5 The guessing game

The second part of ’Lettercademy’ is a program that guesses which letter the user has written. It is implemented in a similar fashion as the feedback game, using the comparison of the contours method. The user provides the name of the image of his/her letter and the program goes through the dataset of all the images of the model letters comparing each one with the user’s handwritten letter. To guess try and guess, the system chooses either the letter with the smallest score of the comparison between the contours in their total or chooses the letter that shows the smallest sum of the scores of the comparison of the parts. If the sum of all the scores (total and parts) is bigger than a threshold of 15, the system is will not take a guess and return a message saying that the user should try again. The threshold is set based on the development testing.

3 Evaluation

For the evaluation of Lettercademy, two evaluation methods has been implemented: one for each program. A testing dataset of handwritten letters has been created and manually evaluated by a human. This dataset is used in the evaluation of both games. There are four images for each letter in a total of 104 images of letters. Each of the images for one letter is annotated with a number from 4 to 1 or from best(4) to worst(1) which corresponds to the four kinds of feedback that are available in the feedback system.

To evaluate the feedback game, the images would go through the game which will give a feedback and a number from 4 to 1, as we said before, 4 is the best feedback and 1 is the worst, and the evaluation system will count how many times it will give the correct feedback compared to our annotated one. The system achieved a 70% correct output.

To evaluate the guessing game, only the testing images with annotation 1 and 2 (the best and the second best) would go through the game and each one is compared with each one of the images of the model letters. We choose the best and the second best written letters of the testing corpus, because we should select letters that the system should be able to guess. Then the evaluation system counts how many times it guessed correctly, how many times it guessed wrong and how many times it failed to guess, and calculate the percentages. 47% of the times it guessed correctly, 21% of the times it guesses the wrong letter and 32% of the times it does not make a guess.

4 Conclusion

The program described in this paper is good enough to recognize handwritten letters compared to gold standard ones and give feedback about them. This system could be improved, in a future work, and could achieve even better and automatic capabilities to recognize if a letter is good or not by applying machine learning methods. However, just by the method of using the contours and shapes of the letters to build a system, we have given the first steps towards an OCR system. We have explored the OpenCV open library for image processing and applied many methods that were useful for our program. The results were satisfactory enough to consider Lettercademy a useful toy with a lot of potential for future improvement.

5 Gloss

OPENCV, January 2017, http://opencv.org/

OPENCV, Basic Thresholding Operations, January 2017, http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.htm

OPENCV, Contours : More Functions, January 2017, http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_more_functions/py_contours_more_functions.html

OPENCV, Structural Analysis and Shape Descriptors, January 2017, http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html