Enhancing Tesseract OCR

Laurence Bonat, Giovanni Valer

1 Introduction

Optical Character Recognition, consisting of extracting text from images, has been a task of major relevance in the Computer Vision research community for several decades. Today it is once again a fundamental task for many applications, as we observe an increasing demand in Text Extraction for digitalization and AI systems. Tesseract OCR is a widely used OCR engine. However, it may encounter difficulties when dealing with images with shadows or complex backgrounds.

We try to enhance the performance of Tesseract OCR by preprocessing images. In particular, we aim to replicate the methods proposed by Revathi et al. (2021) [1]. The first one, *trackbar*, allows the user to adjust various parameters to extract the required text (with manual or autonomous mode). The second one, *automatic filtering*, consists in applying edge detection, filtering, and blurring steps to automatically retrieve the text.

Moreover, we further extend our work trying to tackle the problem of particular backgrounds, namely lined or squared paper. This is because from preliminary experiments we noticed how Tesseract struggles in extracting text from such images. We test several techniques, including shadow removal, adaptive thresholding, Otsu's binarization, and Hough transform, among others. Such work is also motivated by the scarce research in the direction of recognizing handwritten text on lined or squared paper, as the community mainly focuses on handwritten text recognition for old documents [2].

We used Python 3.12.0[1] and Tesseract OCR 5.3.3[2].

[1] https://www.python.org/downloads/release/python-3120/
[2] https://github.com/tesseract-ocr/tesseract/releases/tag/5.3.3

2 Methods

We hereby present the methods implemented to preprocess images for enhancing Tesseract OCR. In Section 2.1 and 2.2 we introduce the methods proposed in the aforementioned paper [1]; while in Section 2.3 there are additional methods we propose.

2.1 Trackbar

To isolate the text characters from the image background, a trackbar is used for choosing the thresholds to be applied to the image. The image is converted into HSV color representation so that there are 6 parameters: namely an *upper* and a *lower* threshold for all *hue*, *saturation*, and *value*. The trackbar method uses a mask based on HSV thresholds that can be chosen manually or calculated automatically. The mask is imposed on the original image and acts as a filter for the specific color of the text we want to recognize.

2.1.1 Manual mode

The manual trackbar allows the user to select directly the thresholding values at runtime. This method works quite well, enabling the user to "mask" the background and improve the output quality. Such preliminary result justifies and prompts the need to investigate ways to automate this method.

2.1.2 Autonomous mode

As in the original paper, we ask the user to provide information on the image: whether the background is brighter or darker than the text. A clear explanation of the proceeding is not described in the paper. Therefore, we implement the autonomous mode of the trackbar setting the thresholds of the lower and upper bounds of the HSV values based on the maximum or minimum values of the smoothed image. The image is first smoothed with a Gaussian blur filter and then a median blur filter, to detect the major components present in the image. After this, we mask the image based on the threshold given by the maximum or the minimum values found in the smoothed image. This enhances the image like in the manual trackbar method, but without losing time selecting the thresholds.

2.2 Automatic filtering

To try and automate the process the paper describes a series of steps to create an automatic filtering process to enhance the capabilities of Tesseract OCR. The described steps are the following:

- Convert the image to grayscale
- Apply a canny edge detection filter, followed by a denoising filter
- Apply a threshold according to the requirement of the image
- Apply erosion followed by dilation
- Run Tesseract OCR on the processed image

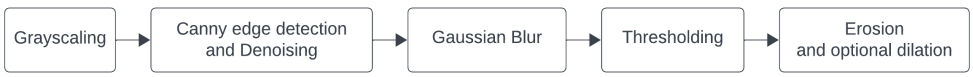


Figure 1: The pipeline for automatic filtering.

This process seems good when the text is large and is elaborated graphically, such as in billboards or boxes. The reason is that photos of a printed book have no particular problems with the Tesseract OCR engine since it is well-trained on printed text.

2.3 Lined or squared paper OCR

While experimenting, we stumbled on some written text on lined or squared paper. Noticing how difficult it is for Tesseract to extract text in such conditions, we deem it interesting to develop *ad hoc* solutions for enhancing OCR on lined and squared paper. In practice, Tesseract cannot extract any character from the major part of images with text on lined paper, so we try to remove lines from the image. After promising preliminary results with text on lined paper, we directly implement a solution for squared paper, which is an extension of the previous.

We first want to threshold the image to make it binary. We experiment with simple thresholding, adaptive thresholding (with both the mean value and the Gaussian-weighted sum approaches), and Otsu's binarization. All these methods are inefficient for images with significant shadows. For such a reason we resort to shadow removal techniques [3]. Specifically, for each channel of the RGB image (we will refer to this as a *plane*), we find the plane shadow by applying a dilation step and blurring with a median filter. Then we subtract the shadow from the original plane and normalize the result (with MinMax normalization). Merging the result for each plane we get the image without shadows, which we can now threshold. We then apply an opening step to remove the remaining noise in the process. This helps to get a clearer image for the next processing.

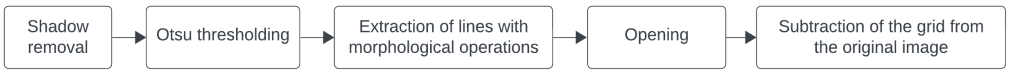


Figure 2: The pipeline for the squared paper OCR process.

Subsequently, we extract vertical and horizontal lines using morphological operations, with single-pixel-width lines as structuring elements. Multiple opening steps are applied to remove noise, and afterward, the vertical (if any) and horizontal lines are combined to get a "grid". Such structure allows us to remove lines from the original image, by computing the bitwise NOT of the two. A foreseeable problem is that, when deleting lines, we risk cutting letters. A solution to reconstruct the original letter is to apply a dilation step; such procedure introduces some artifacts but this is not a problem for Tesseract OCR (see the results in Section 3). Our solution is slightly more elaborate and consists of two opening iterations followed by closing (both using a squared kernel); this allows us to dilate the letters, thus reconstructing those cut by lines without incrementing noise.

Lastly, we try to automatically detect if a text is on lined paper. From preliminary experiments, using the Hough Transform gives promising results. The approach consists of extracting lines with Canny Edge detection followed by Hough Line Transform; then checking if those lines satisfy given properties (length, orientation, relative distance). In alternative to the Standard Hough Transform, the Probabilistic one can be used. This solution proves to be unreliable without any preprocessing of the image, so we apply it to the image after shadow removal and Otsu's binarization (see Section 3.4).

3 Results

In this section we present the results of our methods, providing a qualitative and quantitative assessment of the performance, and an analysis of the error causes.

3.1 Evaluation setup

3.1.1 Metrics

In order to have a measurable way to quantify the performance of the different methods, we use the Levenshtein distance, i.e. the minimum number of single-character edits (insertions or deletions, therefore substitutions are counted as two edits) required to change the extracted text into the ground truth. Before computing such distance, we substitute all spacing characters (`\n`, `\t`, multiple spaces, etc.) with a single white space; this is needed because Tesseract very often extracts a bunch of spaces, which can hugely affect the performance. We can then compute the accuracy from the Levenshtein distance as follows:

$$\frac{(u - d_L)}{u}$$

where u is the number of characters in the image, and d_L is the Levenshtein distance.

3.1.2 Test images

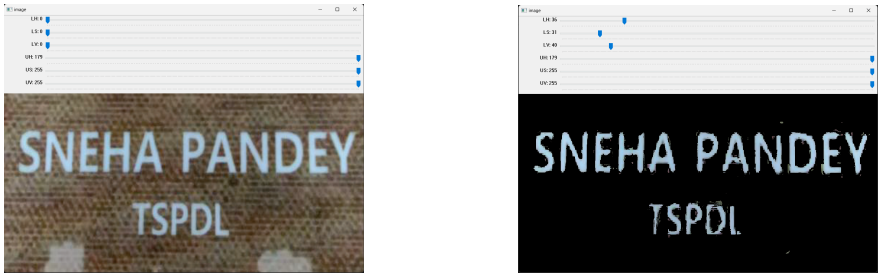
We created a small dataset of images for all methods, including edge cases and difficult images to assess the effectiveness of our methods. The test images are very challenging, so we do not expect our methods to have particularly high accuracy on them.

3.2 Trackbar

The trackbar method performs reasonably well with images having text on a significantly different background (an example is provided in Figure 3), meaning text and background have to be linearly separable in the HSV space. If all the text is of the same color we obtain good results, while separating text with different colors does not yield an impactful result. This is because we extract the text based on its HSV value, so if we have different colors in the text, we cannot be sure that the mask will cut only the background color.

3.2.1 Manual trackbar

In the manual trackbar method, we have 6 user-adjustable sliders, to choose which value of HSV we have to threshold and mask onto the image. The results can be seen in real-



(a) `SNEHA PANDEY SS TSPDL Foe oe` (b) `SNEHA PANDEY TSPDL`

Figure 3: Text extracted with Tesseract OCR, before (a) and after (b) applying thresholding values using the trackbar in manual mode.

time and adjusted to match the text we want to extract. It works well because we can perfectly mask the color of the text in such a way that Tesseract OCR can work flawlessly. We can see an example in Figure 3. Averaging over all test images, the manual trackbar method has an accuracy of 67%, compared to the baseline of Tesseract alone which has

30% of accuracy. The manual approach can be tedious since not all images have the same text color we want to extract. We now analyze the trackbar in autonomous mode, which allows us to choose a color of the background to then automatically search for the text from the images.

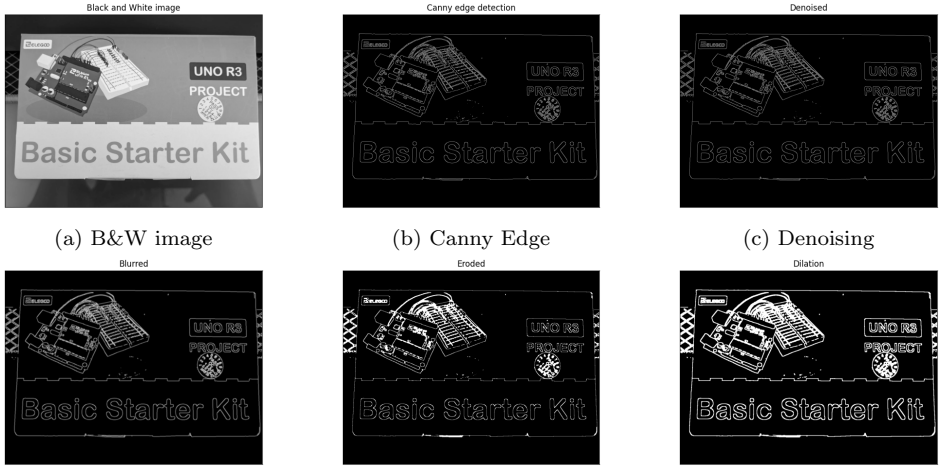
3.2.2 Autonomous mode

The results achieved by the trackbar in autonomous mode could be considered promising, yet unsatisfactory. The algorithm used to find the threshold is not good at separating the values of the text from those of the background and eventually performs well when there is a clear and smooth background. The algorithm's performance is negatively affected when the background or text contains multiple colors. Speaking of evaluation, the autonomous trackbar increases the accuracy from 30% (Tesseract baseline) to 35%.

Figure 4: Result of the autonomous trackbar process

3.3 Automatic filtering

The automatic filtering approach gives mixed results. Tesseract can recognize text in some intermediate steps but the final step does not always yield the best result. The application of the erosion does not help the recognition of the text many times. Another test we made was substituting the erosion and dilation steps with an opening and closing operation on the image, but it did not impact the results significantly. The type of photo used is of fundamental importance. Normal printed text does not require this extensive procedure, since it is the one on which Tesseract is trained.



(a) B&W image (b) Canny Edge (c) Denoising

(d) Gaussian Blur (e) Erosion (f) Dilation

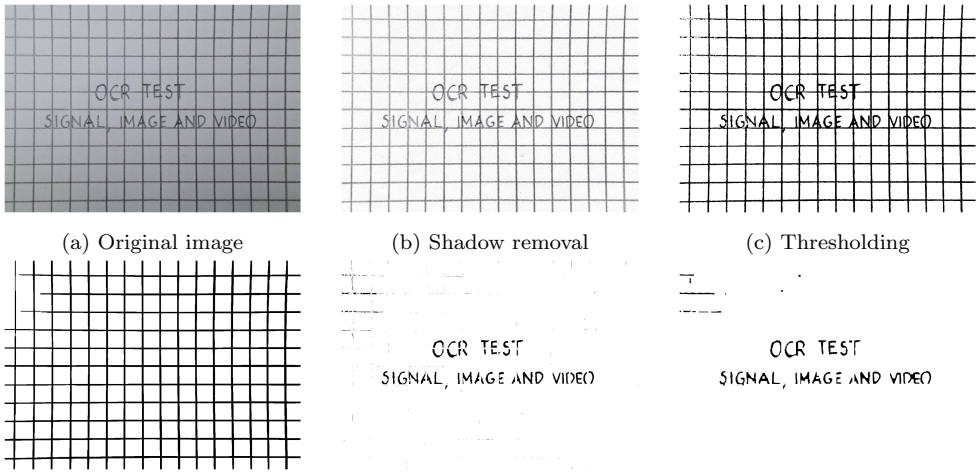
Figure 5: Example of the automatic filtering process. In the original image, Tesseract extracts `Basic Starter Kit`.

The photo in Figure 5 demonstrates this. The process begins with identifying `Basic Starter Kit`. However, after the automatic filtering process, there is no improvement in the results.

Overall, the achieved accuracy on the test images is 20%. Such low value is caused by images with small text, which perform badly with the canny edge detection filter, due to the erosion and dilation process used later on. In addition, shapes present in an image can interfere with the recognition process, as Canny Edge Detection extracts them. This produces a lot of noise, that cannot be smoothed out with an opening or denoising operation, or we will miss the text we want to extract.

3.4 Lined or squared paper OCR

This method can effectively remove lines and grids from images (as shown in Figure 6), resulting in a significantly higher performance of Tesseract OCR. Figure 6 also proves how important the last opening step is, even though in some cases this might bring slightly worse results (see 3.5).



(a) Original image (b) Shadow removal (c) Thresholding

(d) Extracted grid (e) Grid removal (f) Opening-closing

Figure 6: Example of the whole squared-paper-OCR process. In the original image, Tesseract cannot extract any text. After grid removal, it extracts: `Pet ee re`. And at the final stage: `OCR TEST SIGNAL, IMAGE AND VINEO`.

The average accuracy of this method, on the test images, is 63%, while the Tesseract baseline has 0% (not a single correct character is retrieved). Regarding the line detection performance, with the Probabilistic Hough Transform method, we achieve a precision of 100% and a recall of 90% (resulting in 95% of accuracy).

3.5 Error analysis

3.5.1 Trackbar

For the trackbar method described in Section 2.1 there are some cases where it struggles, we analyze these cases and provide explanations. Images that have colorful text are hard to threshold because we could lose information if the text is similar to the background. When we have a stark contrast we can select values easily enough with the manual trackbar, but with the automatic trackbar we have more difficulties, since the method struggles in computing the perfect thresholding values. This because even a small change in the values can have a huge impact on the output.

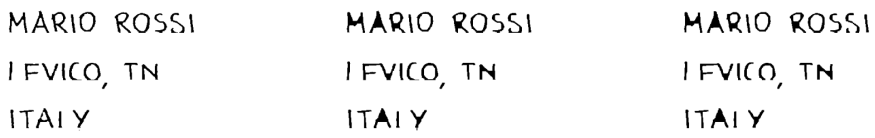
3.5.2 Automatic filtering

In Section 2.2 we mention a method described in the paper that uses only some preprocessing to enhance the OCR of images. This process works in specific cases where the font is larger and bold. For each step of the pipeline, we run Tesseract OCR to see if there is an improvement between steps or a regression. Considering the image in Figure 5, we

can see that with the first canny edge filter we obtain some new text with random characters `[S@ A&A 8 UNO-.R3 BASIC Starter \A`. After the denoising and blurring process we obtain better OCR. The final dilation step extracts the following text: `PROJECT F Basic Starter (Ait`. It finds more text than the original image but is not as accurate. Using another image highlights the same problems: depending on how text is present in the image, we cannot reliably extract text from it. This is because canny edge detection kills the performance of small font text since it can eliminate small details of a character. For example, an "a" could become an "o" if the edge detection fails to recognize a part of the glyph. The successive erosion and then dilation worsens only the problem related to canny edge detection.

3.5.3 Squared paper OCR

As mentioned in Section 2.3, the last opening-closing step might introduce errors. This is due to the final closing operation, which sometimes removes parts of letters if they are too thin. We provide an example in Figure 7. Notice how the `S` letter is affected, with unexpected consequences on the Tesseract output.



(a) After grid removal:
`MARIO ROSSI`
`IEVICO, TN ITALY`

(b) Opening:
`MARIO ROSSI`
`IFVICO, TN ITALY`

(c) Closing:
`MARIO ROSA`
`|FWICO, TN ITALY`

Figure 7: Example of how the last opening-closing step affects the overall performance of Tesseract OCR.

4 Conclusions

We showed several solutions to enhance Tesseract OCR for text on particular backgrounds, with shadows, and on squared paper. We argue a single technique cannot serve as a universal solution, since images containing text can be highly diverse. For this reason, a number of methods are needed, possibly automatically chosen according to a previous image categorization.

We remark that the test images are arguably difficult for OCR, therefore low accuracy in evaluation is expected. However, the automatic filtering method does not give adequate results: it has an accuracy even lower than the baseline. This means the performed processing actually worsens the quality of the image for Tesseract, indicating this approach is not a viable solution for enhancing the OCR engine. The manual and autonomous trackbar methods, on the contrary, enhance the performance of Tesseract OCR, but have an intrinsic limit due to the thresholding procedure. For this reason, we cannot expect to further improve the output. Finally, the results achieved by the squared paper OCR method are remarkable and could be successfully applied in the context of document digitalization; in particular for Handwritten Text Digitalization (HTD), which might happen to deal with lined or squared paper.

4.1 Future work

We are aware of some limitations in our work. In particular, some parameters (e.g., the number of opening step iterations), tuned on our batch of images, might not be the optimal ones. A "fine-tuning" process could be done with a larger dataset to set the best values.

Regarding further work, for the trackbar in autonomous mode, a significant improvement would be to avoid the necessity of asking the user whether the background is brighter or darker than the text. Such a task is far from trivial, as it is strongly influenced by how much space the text occupies in the image.

Moreover, as previously mentioned, we see the automatic categorization of images (e.g., colored text, uniform background, textured/heterogeneous background, squared paper, etc.) as an important step to automatize the whole text extraction process, so that the best method for enhancing Tesseract can be chosen without the need of human intervention. We already worked to this end, with the lines detection method, but more categories should be addressed.

References

[1] AS Revathi and Nishi A Modi. Comparative Analysis of Text Extraction from Color Images using Tesseract and OpenCV. In *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 931–936, 2021.

[2] Wissam AlKendi, Franck Gechter, Laurent Heyberger, and Christophe Guyeux. Advancements and challenges in handwritten text recognition: A comprehensive survey. *Journal of Imaging*, 10(1), 2024.

[3] Huimin Lu, Baofeng Guo, Juntao Liu, and Xijun Yan. A shadow removal method for Tesseract text recognition. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5, 2017.