



## BRAIN. Broad Research in Artificial Intelligence and Neuroscience

e-ISSN: 2067-3957 | p-ISSN: 2068-0473

Covered in: Web of Science (ESCI); EBSCO; JERIH PLUS (hkdir.no); IndexCopernicus; Google Scholar; SHERPA/RoMEO; ArticleReach Direct; WorldCat; CrossRef; Peeref; Bridge of Knowledge (mostwiedzy.pl); abcdindex.com; Editage; Ingenta Connect Publication; OALib; scite.ai; Scholar9; Scientific and Technical Information Portal; FID Move; ADVANCED SCIENCES INDEX (European Science Evaluation Center; neredataltics.org); ivySCI; exaly.com; Journal Selector Tool (letpub.com); Citefactor.org; fatcat!; ZDB catalogue; Catalogue SUDOC (abes.fr); OpenAlex; Wikidata; The ISSN Portal; Socolar; KVK-Volltitel (kit.edu) 2025, Volume 16, Issue 2, pages: 409-417.

Submitted: February 17<sup>th</sup>, 2025 | Accepted for publication: April 9<sup>th</sup>, 2025

### Pixel-Wise Method for Enhanced Tesseract OCR Accuracy Using Colour and Spatial Distances

#### Mihai-Lucian Voncilă

Computer Science and Engineering  
Department, Faculty of Automatic Control and  
Computers, National University of Science and  
Technology Politehnica Bucharest, Bucharest,  
Romania.

mihai\_lucian.voncila@stud.acs.upb.ro  
<https://orcid.org/0000-0002-6764-9125>

#### Nicolae Tarbă

Computer Science and Engineering  
Department, Faculty of Automatic Control and  
Computers, National University of Science and  
Technology Politehnica Bucharest, Bucharest,  
Romania.

nicolae.tarba@upb.ro  
<https://orcid.org/0000-0002-7769-8289>

#### Cosmin-Dumitru Oprea

Computer Science and Engineering  
Department, Faculty of Automatic Control and  
Computers, National University of Science and  
Technology Politehnica Bucharest, Bucharest,  
Romania.

cosmin\_dumitru.oprea@upb.ro

#### Costin Anton Boiangiu

Computer Science and Engineering  
Department, Faculty of Automatic Control and  
Computers, National University of Science and  
Technology Politehnica Bucharest, Bucharest,  
Romania.

costin.boiangiu@cs.pub.ro  
<https://orcid.org/0000-0002-2987-4022>

#### Nicolae Goga

The Faculty of Engineering in Foreign Languages, Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Bucharest, Romania.

nicu.goga@upb.ro  
<https://orcid.org/0000-0003-3812-9646>

**Abstract:** *Digital images often contain noise introduced during acquisition, storage, or transmission, which can hinder the performance of Optical Character Recognition systems. Effective noise reduction is essential for improving the accuracy of these systems, as noise can obscure text and reduce recognition rates. The problem of removing noise from images is widely studied in computer vision but remains challenging due to the variety of noise types and the risk of introducing artifacts or blurring. In this work, we propose a new preprocessing algorithm that is used in conjunction with the Tesseract engine, in order to improve its overall accuracy. We test this method against the SmartDoc dataset, which contains images taken from mobile devices, and obtain an improvement over the original accuracy of 6.5%. The method is also compared to several other classical algorithms such as Mean Filter, Median Filter, Bilateral Filter, Adaptive Smoothing, and others showing improved results over each individual one.*

**Keywords:** *OCR; noise reduction; Tesseract; image preprocessing; image enhancement; OCR accuracy improvement; contrast correction..*

**How to cite:** Voncilă, M.-L., Tarbă, N., Oprea, C.-D., Boiangiu, C. A., & Goga, N. (2025). Pixel-wise method for enhanced Tesseract OCR accuracy using colour and spatial distances. *BRAIN: Broad Research in Artificial Intelligence and Neuroscience*, 16(2), 409-417.  
<https://doi.org/10.70594/brain/16.2/29>

## 1. Introduction

Optical Character Recognition (OCR), the process of converting printed or handwritten text in images into machine-readable text, has become increasingly important in various fields such as document digitisation, automated data entry, and archiving. It allows for efficient extraction and processing of information from scanned documents, photographs, or digital images, enabling tasks like searching, editing, and indexing. However, the accuracy of OCR systems is highly dependent on the quality of the input images. Factors such as noise, blur, and poor contrast can significantly impair the system's ability to correctly identify characters. Therefore, preprocessing steps like noise reduction play a critical role in enhancing the performance of OCR by improving the clarity and readability of the text within the image. Traditional noise reduction methods often employ generalised assumptions and broad approaches, which can lead to imprecision in their results. These techniques may fail to adequately address the unique characteristics of different types of noise, resulting in the loss of important image details or the introduction of artifacts, which can further hinder the accuracy of OCR systems.

In the realm of data digitisation, various OCR engines are available, each offering unique strengths tailored to different needs. One of the more popular options tends to be Tesseract (Smith & Google, 2024), which stands out for its open-source nature, offering extensive customisation and flexibility. Originally developed by Hewlett-Packard and now maintained by Google, Tesseract supports multiple languages and scripts. It can also be trained on custom datasets, making it suitable for a wide range of OCR tasks, from simple text extraction to more complex layouts. Users appreciate its ability to recognise text from diverse sources, such as scanned documents and photographs. However, its accuracy can vary depending on image quality and preprocessing steps, often requiring additional noise reduction techniques to improve performance.

In this paper, we introduce a noise reduction method designed to enhance digital images for improved OCR performance, subsequently applying Tesseract to extract the image text. The system employs a kernel-based approach, where a kernel is centered around each pixel in the image. For each pixel, a new value is computed by analysing the values within the kernel and applying a specific formula that combines them. This process reduces noise while preserving key image features, such as edges and sharp transitions, by adjusting pixel values based on local context. The kernel can be tailored to target specific goals, such as minimising random noise or enhancing contrast. By systematically applying this technique across the image, the proposed method improves overall image clarity, making it more suitable for OCR tasks.

This approach tends to be particularly well-suited for real-time mobile OCR applications, where images are captured in uncontrolled environments and processing power is reduced. Thus, the goal of the proposed method is to offer a fast, lightweight, robust algorithm that can improve the image quality without introducing significant overhead, making it ideal for on-device text-recognition tasks such as receipt scanning or document capture.

## 2. Related Work

Image preprocessing is a critical step for enhancing the accuracy of OCR systems by improving the image quality before text extraction. To improve Tesseract's accuracy, various methods have been developed in relation to image preprocessing, addressing a variety of factors, such as noise reduction, resolution enhancement, and feature extraction.

Several classical preprocessing techniques have been shown to improve Tesseract's performance. Some of the more well-known ones, such as the mean filter, median filter, and bilateral filter (Tomasi & Manduchi, 1998), are commonly used for noise reduction. These approaches tend to be lightweight in implementations, making them particularly suitable for real-time or embedded OCR scenarios. Similar techniques such as adaptive smoothing (Saint-Marc, Chen, & Medioni, 1991), conservative smoothing (Sun & Takayama, 1999), and contrast

normalisation are frequently used in order to enhance text visibility and remove background artifacts.

Lu et al. (2017) introduced a shadow removal technique based on a combination of adaptive thresholding, projection, and Median Filtering to improve Tesseract's performance under low-light conditions. Similarly, Kariya, Fujishima, and Zhang (2018) present a way to improve the input via a combination of non-local means denoising and adaptive thresholding. Li et al. (2024) also propose a mechanism for increasing the accuracy of Tesseract based on a combination of Median Filtering and skew removal grounded on the Hough transform.

Recently, more complex methods for denoising, such as those based on deep learning approaches, using autoencoders, have been proposed and used with great results. Wiraatmaja, Gunadi, and Sandjaja (2017) and Alamsyah et al. (2022) implemented denoising autoencoders as preprocessing steps, further increasing Tesseract's ability to process noisy or distorted text. Another approach is that of Zhao et al. (2018), who propose the use of a skip-connected deep convolutional autoencoder to reduce image blur and other forms of noise in an image. Moreover, Hsu et al. (2023) performed a study in which they explored advanced methods of dealing with analytical noise in machine learning, reinforcing the importance of mitigating noise for enhancing text recognition accuracy.

Resolution tends to have a significant impact on an OCR engine's ability to effectively recognise specific characters in low-quality images. Peng and Wang (2020) developed a super-resolution image generator specifically for OCR applications, enabling Tesseract to work more effectively on degraded images. Walha et al. (2018) went in a different direction in order to improve resolution by sparsely representing images and using a combination of both offline and online dictionaries, where the first contains clean images and the other is updated dynamically based on the characteristics of the noisy image. Similarly, Baranov et al. (2023) proposed a super-resolution technique based on Convolutional Neural Networks (CNN) used to improve Tesseract's accuracy, particularly for images that suffer from compression artifacts, blurring, or low contrast.

Galvis et al. (2021) went in the direction of analysing text from video frames, combined with denoising them by using two different architectures. They tested a Denoising CNN designed for deblurring scanned text images, and an encoder-decoder model called FCHardNet, developed for pixel classification.

Lastly, newer studies like those done by Etter, Carpenter, and King (2023) and Emon et al. (2023) explore the application of hybrid models and advanced techniques to handle multilingual OCR and specific script challenges, further extending Tesseract's capabilities in global applications.

Despite the advances in learning-based preprocessing, classical filters remain a lightweight, interpretable, and tunable alternative that is extremely valuable in real-time or resource-constrained environments. Thus, we opt to compare our method with more classical approaches, in order to fit the constraints of these environments better.

### 3. Materials and methods

#### 3.1. Proposed Method

All input images used in our experiments are treated as grayscale images  $I(x, y)$ , where each pixel value lies in the range  $[0, 255]$ . In order to apply localised operations for each filter, we define a square neighbourhood window  $N_w$ , of size  $w$ , centred around each pixel:

$$N_w = \{(i, j) \mid -\lfloor \frac{w}{2} \rfloor \leq i, j \leq \lfloor \frac{w}{2} \rfloor\} \quad (1)$$

The proposed solution uses a sliding window of fixed size, which is applied to all pixels in an image (Oprea, 2019). For each pixel,  $(x, y)$  we compute two distance metrics over its neighbourhood: a spatial distance, Equation 2, and a colour distance, Equation 3.

The spatial distance is defined by:

$$S(i, j) = (i^\alpha + j^\beta)^\gamma \quad (2)$$

where  $\alpha$ ,  $\beta$  represent weights intended for the horizontal and vertical distances between a pixel and its respective neighbours, and the ratio between them can be tweaked if one is deemed more important than the other. Similarly,  $\gamma$  is a control parameter used for distance scaling.

Similarly, our colour distance formula is represented by:

$$D_{x,y}(i, j) = |I(x + i, y + j) - I(x, y)|^\delta \quad (3)$$

where  $\delta$  is a preselected parameter to vary our colour formula.

Thus, our initial filter can be defined as:

$$I_f(x, y) = \sum_{i,j \in N_w} S(i, j) * D_{x,y}(i, j) \quad (4)$$

Notably, the spatial distance kernel  $S(i, j)$  is invariant on image contents and can be precomputed once as an optimisation step.

After computing the new filtered image, we perform two additional operations to improve the results. The first involves removing outliers from the histogram, based on two thresholds, namely a lower and upper bound, where values lower or higher than those bounds are clipped to the interval, as can be seen in Equation 5. Our final step involves normalising the image back to a chosen interval, which can be seen in Equation 6.

We define our image after thresholding as:

$$I_f(x, y) = T_{low} \text{ if } I_f(x, y) < T_{low}, T_{high} \text{ if } I_f(x, y) > T_{high}, I_f(x, y) \text{ otherwise} \quad (5)$$

where  $T_{low}$  represents the lower clipping bound, and  $T_{high}$  represents the upper clipping bound.

Thus, our final computed image can be defined as:

$$I_f(x, y) = \frac{I_f(x, y) - T_{low}}{T_{high} - T_{low}} * (h - l) + l \quad (6)$$

where  $h$  and  $l$  are the desired lower and upper range for the normalisation interval, which in practice we keep in the range  $[0, 255]$ .

It is important to note that, for certain parameter values, our image would get its colours flipped, with the background and text switching colours between each other. In these scenarios, a final flip operation to return the background and text to their original colours is performed by subtracting the max range value from the filtered image.

### 3.2. Dataset Description

We evaluate our method using the SmartDoc (Burie et al., 2015) dataset, which consists of scanned image documents captured via mobile devices. These images feature a variety of distortions, such as skew, uneven illumination, compression noise, as well as potential motion blur.

In order to simulate degraded conditions and increase the difficulty of reading specific characters, images were further downsampled such that their longest edge measured approximately 30-40% of the original, while preserving aspect ratio. This step introduces additional noise, making preprocessing pivotal.

### 3.3. Comparison Filtering Techniques

In this paper, we evaluate the performance of the proposed solution against several denoising techniques. Specifically, we test Mean Filter, Median Filter, Conservative Smoothing, Adaptive Smoothing, Bilateral Smoothing, and Contrast Correction. All algorithms were applied individually to each image before being fed into the OCR engine. By comparing these results, we aimed to

assess the effectiveness of each technique in improving overall accuracy and to highlight the advantages of our proposed method over more traditional approaches.

For our mean filter, we use a window size  $w$  of 3 in order to apply minimum denoising, while aiming to preserve character edges, as larger window sizes can introduce more blur and degrade the quality of characters, which is detrimental for OCR engines. The filter can be defined using:

$$I_f(x, y) = \frac{1}{|N_w|} \sum_{i,j \in N_w} I(x + i, y + j) \quad (7)$$

Similarly, in the case of the median filter, we also use  $w = 3$ , for the same reasons, namely, minimum denoising and preserving of edges. The filter tends to be better at removing specific noises, such as salt and pepper, which are unusual in the images found in this dataset, but it is still kept for comparisons. The filter is defined as:

$$I_f(x, y) = \text{median}\{I(x + i, y + j)\}_{i,j \in N_w} \quad (8)$$

Conservative smoothing (Sun & Takayama, 1999) is a filter that is designed to preserve the structure of an image, and to replace a pixel only if its overall intensity is outside the range of its neighbours, excluding itself:

$$V(x, y) = \{I(x + i, y + j) | (i, j) \in N_w, (i, j) \neq (0, 0)\} \quad (9)$$

Then we define the filter as:

$$I_f(x, y) = \min(V) \text{ if } I(x, y) < \min(V), \max(V) \text{ if } I(x, y) > \max(V), I(x, y) \text{ otherwise} \quad (10)$$

Similarly, we use a small window size  $w = 5$  in order to preserve character edges.

Adaptive smoothing (Saint-Marc, Chen, & Medioni, 1991) is defined as:

$$I_f(x, y) = \frac{1}{|N_w|} \sum_{i,j \in N_w} I(x + i, y + j) \text{ if } \sigma^2 < T, I_f(x, y) \text{ otherwise} \quad (11)$$

where  $\sigma^2$  is the local variance within the window, and  $T$  is a threshold used to determine if a pixel should be replaced by the mean of its window. We use  $w = 7$  and (since images are in the range  $[0, 255]$ )  $T = 50$ . This is done to suppress noise while keeping text details untouched.

In the case of bilateral smoothing (Tomasi & Manduchi, 1998), the method is more complex. As the filter uses two different Gaussian functions in order to smooth edge preservation, similar to our approach, namely a spatial one, and an intensity-based one.

These Gaussians are defined as:

$$G_s(i, j) = e^{-\frac{i^2 + j^2}{2\sigma_s^2}} \quad (12)$$

$$G_r(x, y, i, j) = e^{-\frac{(I(x+i,y+j) - I(x,y))^2}{2\sigma_r^2}} \quad (13)$$

where  $\sigma_s^2$  and  $\sigma_r^2$  represent the standard deviations used by the two kernels.

Thus, the bilateral filter can be defined as:

$$I_f(x, y) = \frac{\sum_{i,j \in N_w} I(x+i,y+j) * G_s(i,j) * G_r(x,y,i,j)}{\sum_{i,j \in N_w} G_s(i,j) * G_r(x,y,i,j)} \quad (14)$$

In the case of this filter, we use a larger window size,  $w = 9$  in order to take advantage of the two kernels, in combination with  $\sigma_s^2 = 2.5$ , and  $\sigma_r^2 = 35$ . This is done to contrast well with our window, reducing noise, maintaining character edges, and avoiding major distortions.

Finally, we implement a global contrast correction, which is aimed at making the image span the entire  $[0, 255]$  range, improving the results for faded documents.

### 3.4. Implementation Details

All filtering techniques were implemented in Python 3.11, with performance optimisations using Numba 0.61.0 for just-in-time compilation. This enabled efficient, parallel execution of windowed filters and allowed for a fair timing comparison between all of them. Support libraries included NumPy 1.26.4 and OpenCV 4.8.1. Experiments were conducted on a system with an AMD Ryzen 9 7900X3D CPU and 64 GB RAM. All images were processed using Tesseract 5.5.

## 4. Tests and Results

The initial dataset used to derive our initial formula parameters consisted of five images with various resolutions and different levels of clarity regarding the text. The reason we chose to employ a brute force search is that we expect certain values to deteriorate the image more than improve it, thus we can limit the search space from the start. Secondly, we are interested in how certain parameter behaviours affect our formula and what insights can be gained from them.

The first step in our brute force approach was varying each individual parameter by itself before performing a more exhaustive search, in order to better limit the variance of our parameters. Thus, we set each of  $\alpha$ ,  $\beta$ ,  $\delta$ ,  $\gamma$  equal to 1, and  $w = 59$ , before starting our search.

The first parameter to consider is  $\delta$ , which tends to behave in various manners for different values. For  $\delta < 0$ , the images end up containing additional noise, or certain artifacts, and the text ends up being blurrier. Similarly, values that are very high end up darkening the whole image a lot more, thus making the text in low-contrast portions of the image even more difficult to read. The best results were observed for values centered around  $\delta = 1$ , which is the value that was set when exploring the rest of the parameters.

In the second step, we test the combination of  $\alpha$ ,  $\beta$ ,  $\gamma$ , as they all have a direct influence over each other since they work distances in the image space. The first thing that can be observed is that varying  $\alpha$  and  $\beta$  separately leads to severe directional blur, either vertical or horizontal, based on the larger parameter. The effect ends up being stomped for closely related or identical values. In a similar manner, if all three parameters are simultaneously positive or negative, the image ends up containing certain ghosting effects. The best results tend to be obtained for values centered around the 2d Euclidean distance, namely  $\alpha = 2$ ,  $\beta = 2$ ,  $\gamma = -0.5$ .

Lastly, we vary the window size (the  $w$  parameter). This parameter, however, is varied on a larger dataset, comprising 200 images. One thing to note about varying the window size is that for small window sizes, all the image characteristics tend to get blurred, both the characters and the overall noise. Whilst this can decrease readability, the algorithm tends to preserve the edges of the characters while removing their filling. Similarly, for large window sizes, certain detrimental characteristics could be amplified alongside certain beneficial ones that would improve overall readability. Additionally, smaller window size reduces the computation time required by a large factor. Thus, a balance needs to be reached between a window size large enough to maintain and amplify important characteristics, but one that doesn't necessarily require a large computational time.

We aimed at measuring three key metrics, namely overall accuracy, the total number of correctly identified words or Word Count (WC), and Longest Common Subsequence (LCS), alongside the average computation time per image. The results of these metrics can be observed in

Table 1. In this scenario, the parameters for each of the filters are those mentioned in Section 3.3, with our proposed method having a  $w = 27$ ,  $T_{low} = 50$ ,  $T_{high} = 205$ , and images are resized to have their longest edge equal to 1600 pixels.

Table 1. Performance comparison between the proposed method and classical approaches,  $W = 51$

| Algorithm              | Accuracy (%) |        |       | WC   |        |      | LCS  |        |      | Time (ms) |
|------------------------|--------------|--------|-------|------|--------|------|------|--------|------|-----------|
|                        | Mean         | Median | Std   | Mean | Median | Std  | Mean | Median | Std  |           |
| Original               | 66.25        | 75.78  | 31.42 | 3366 | 3589   | 1162 | 2519 | 2674   | 1340 | N/A       |
| Mean Filter            | 36.77        | 21.86  | 34.62 | 1573 | 1029   | 1524 | 1216 | 555    | 1317 | 3         |
| Median Filter          | 29.45        | 22.16  | 25.06 | 2059 | 2088   | 1486 | 996  | 666    | 972  | 74        |
| Conservative Smoothing | 66.5         | 76.15  | 31.4  | 3367 | 3588   | 1162 | 2530 | 2673   | 1342 | 5         |
| Adaptive Smoothing     | 67.97        | 77.08  | 30.18 | 3394 | 3636   | 1135 | 2583 | 2701   | 1300 | 57        |
| Bilateral Filter       | 65.55        | 72.43  | 29.68 | 3294 | 3556   | 1186 | 2475 | 2603   | 1278 | 140       |
| Contrast Correction    | 66.9         | 76.08  | 30.88 | 3373 | 3614   | 1160 | 2548 | 2705   | 1324 | 5         |
| Proposed Method        | 70.56        | 77.81  | 21.61 | 3701 | 3767   | 720  | 2802 | 2759   | 947  | 262       |

What we can observe is that while our method tends to outperform all the others and improve the original accuracy of the detections by 6.5%, it tends to be about 4 times slower than the second-best method in this case, while only improving the accuracy by 3.8%. This is mainly due to the large difference in window sizes between the two methods. While this is the case, the time is still on the whole image being processed, so the speed trade-off can be justified in this scenario, as the average compute time is still around 260 milliseconds only.

Similarly, Table 2 presents some of the various window sizes tested on the restricted dataset. Something to note is that past a certain point, increasing the window size does not improve accuracy, while the complexity still increases. This is due in part to adding more unnecessary noise to the window, as some of the characters will start overlapping one another past a certain size.

Table 2. Performance metrics for various window sizes

| Window size | Accuracy (%) |        |       | WC   |        |     | LCS  |        |      | Time (ms) |
|-------------|--------------|--------|-------|------|--------|-----|------|--------|------|-----------|
|             | Mean         | Median | Std   | Mean | Median | Std | Mean | Median | Std  |           |
| 5           | 53.33        | 62.39  | 27.79 | 2384 | 3411   | 865 | 2023 | 2284   | 1142 | 38        |
| 13          | 66.32        | 76.18  | 24.77 | 3530 | 3578   | 811 | 2556 | 2644   | 1086 | 82        |
| 27          | 68.41        | 77.71  | 24.78 | 3575 | 3621   | 750 | 2624 | 2724   | 1061 | 236       |
| 43          | 62.11        | 76.55  | 29.15 | 3566 | 3623   | 728 | 2393 | 2532   | 1219 | 509       |
| 51          | 67.3         | 76.96  | 25.54 | 3553 | 3610   | 716 | 2572 | 2699   | 1063 | 693       |
| 65          | 65.95        | 77.9   | 25.98 | 3549 | 3621   | 716 | 2521 | 2583   | 1083 | 1088      |
| 81          | 63.8         | 74.39  | 28.54 | 3546 | 3593   | 715 | 2456 | 2737   | 1206 | 1647      |
| 113         | 65.43        | 78.9   | 27.01 | 3549 | 3635   | 708 | 2538 | 2640   | 1158 | 3146      |
| 149         | 64.9         | 74.6   | 28.36 | 3545 | 3595   | 702 | 2501 | 2709   | 1164 | 5461      |

## 5. Conclusions

In this paper, we introduced a pixel-wise preprocessing method that incorporates both colour and spatial distances in order to improve the accuracy of Tesseract OCR. The technique we used proved to be efficient in reducing the noise present in the image, while preserving the key features

such as edges, while also changing the contrast. This resulted in an improvement in text clarity, making the images more suitable for OCR.

Even though our method was limited in the parameter search space, we still obtained improvements over the original image accuracy, as well as the other filters used, varying from 4-6.5% in overall improvements. Something to consider as a future improvement would be testing on a variety of other parameters that would also benefit the other filters more.

Another future improvement that could be considered alongside an expanded dataset is testing the validity of the method across multiple other OCR engines, such as ABBYY FineReader or Google Cloud Vision, in order to provide insights into how well this approach generalises beyond Tesseract.

A different potential improvement is related to the optimisation of the kernel parameters to better fit certain image noise profiles. The current implementation uses a fixed kernel size and formula for the entire image, but adaptive techniques could be explored to automatically adjust the parameters based on the characteristics of each image. For instance, deep learning models could be trained to predict optimal parameters for specific image features, allowing for a more dynamic and context-aware preprocessing process.

Additionally, while the current method tends to perform well on certain noise patterns or blurry characteristics, it does not adequately address low-quality or heavily degraded images. In the future, this method could be combined with super-resolution-based methods in order to further increase the accuracy of the OCR in a variety of scenarios.

Finally, the development of a comprehensive framework that does not include only overall accuracy but also measures computational efficiency, such as the processing time and overall resource usage, could be useful for practical deployments. This would ensure that the method could be used efficiently in applications where resources might be limited, such as mobile devices. This is also one of the reasons we choose to use a simpler approach, such as a kernel-based method with fixed parameters, as opposed to a model based on CNN, since the computation can be faster.

## References

- Alamsyah, N., Fauzan, M. N., Putrada, A. G., & Pane, S. F. (2022). Autoencoder Image Denoising to Increase Optical Character Recognition Performance in Text Conversion. 2022 International Conference on Advanced Creative Networks and Intelligent Systems (ICACNIS), Bandung. <https://doi.org/10.1109/ICACNIS57039.2022.10054885>
- Baranov, M., Serhii, I., Shvetsov, D., & Shcherbyna, Y. (2023). Application of super resolution for optical character recognition in low quality images. In *Lecture Notes in Networks and Systems* (pp. 135–145). [https://doi.org/10.1007/978-981-99-3043-2\\_11](https://doi.org/10.1007/978-981-99-3043-2_11)
- Burie, J.-C., Chazalon, J., Coustaty, M., Eskenazi, S., Luqman, M. M., Mehri, M., Nayef, N., Ogier, J.-M., Prum, S., & Rusiñol, M. (2015, August). ICDAR2015 competition on smartphone document capture and OCR (SmartDoc). In *Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR)* (pp. 1161–1165). IEEE. <https://doi.org/10.1109/ICDAR.2015.7333943>
- Emon, M. I. H., Iqbal, K. N., Mehedi, M. H. K., Mahbub, M. J. A., & Rasel, A. A. (2023). A review of optical character recognition (OCR) techniques on Bengali scripts. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* (pp. 85–94). Springer Nature Switzerland. [https://doi.org/10.1007/978-3-031-25161-0\\_6](https://doi.org/10.1007/978-3-031-25161-0_6)
- Etter, D., Carpenter, C., & King, N. (2023). A hybrid model for multilingual OCR. In G. A. Fink, R. Jain, K. Kise, & R. Zanibbi (Eds.), *Document analysis and recognition - ICDAR 2023. ICDAR 2023 (Lecture Notes in Computer Science, Vol. 14187, pp. 326–337)*. Springer, Cham. [https://doi.org/10.1007/978-3-031-41676-7\\_27](https://doi.org/10.1007/978-3-031-41676-7_27)

- Galvis, J., Morales, S., Kasmi, C., & Vega, F. (2021). Denoising of video frames resulting from video interface leakage using deep learning for efficient optical character recognition. *IEEE Letters on Electromagnetic Compatibility Practice and Applications*, 3(2), 82–86. <https://doi.org/10.1109/lemcpa.2021.3073663>
- Hsu, T., Tsai, Y., Chu, W. C., Chen, S., Tsai, H., & Chang, Y. (2023). Exploration of advanced computer technology to address analytical and noise improvement issues in machine learning. *Journal of Systems and Software*, 205, 111820. <https://doi.org/10.1016/j.jss.2023.111820>
- Kariya, K., Fujishima, T., & Zhang, L. (2018). Development of Training Data for Optical Character Recognition using Deformed Printing Characters. In *Proceedings of the 6th IIAE International Conference on Industrial Application Engineering 2018*. The Institute of Industrial Application Engineers.
- Li, X., Zhan, Z., Li, C., Guo, C., Li, Z., & Li, J. (2024). Research on English character recognition technology based on OCR. In *Proceedings of the 2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. <https://doi.org/10.1109/imcec59810.2024.10575683>
- Lu, H., Guo, B., Liu, J., & Yan, X. (2017). A shadow removal method for Tesseract text recognition. In *Proceedings of the 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. <https://doi.org/10.1109/cisp-bmei.2017.8301946>
- Oprea, C.-D. (2019). *Image noise reduction* (Unpublished master's thesis). Faculty of Automatic Control and Computers, Department of Computer Science, Politehnica University of Bucharest.
- Peng, X., & Wang, C. (2020). Building super-resolution image generator for OCR accuracy improvement. In X. Bai, D. Karatzas, & D. Lopresti (Eds.), *Document analysis systems. DAS 2020* (Lecture Notes in Computer Science, Vol. 12116, pp. 137–148). Springer, Cham. [https://doi.org/10.1007/978-3-030-57058-3\\_11](https://doi.org/10.1007/978-3-030-57058-3_11)
- Saint-Marc, P., Chen, J. S., & Medioni, G. (1991). Adaptive smoothing: A general tool for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6), 514–529. <https://doi.org/10.1109/34.87339>
- Smith, R., & Google Inc. (2024). *Tesseract: Open source OCR engine* (Version 5.2.1) [Computer software]. <https://github.com/tesseract-ocr/tesseract> (Last accessed: 2 October 2024)
- Sun, M., & Takayama, K. (1999). Conservative smoothing on an adaptive quadrilateral grid. *Journal of Computational Physics*, 150(1), 143–180. <https://doi.org/10.1006/jcph.1998.6167>
- Tomasi, C., & Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision* (pp. 839–846). IEEE. <https://doi.org/10.1109/ICCV.1998.710815>
- Walha, R., Drira, F., Lebourgeois, F., Garcia, C., & Alimi, A. M. (2018). Handling noise in textual image resolution enhancement using online and offline learned dictionaries. *International Journal on Document Analysis and Recognition (IJDAR)*, 21(1), 137–157. <https://doi.org/10.1007/s10032-017-0294-6>
- Wiraatmaja, C., Gunadi, K., & Sandjaja, I. N. (2017). The application of deep convolutional denoising autoencoder for optical character recognition preprocessing. In *Proceedings of the 2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIT)*. <https://doi.org/10.1109/icsiit.2017.32>
- Zhao, G., Liu, J., Jiang, J., Guan, H., & Wen, J.-R. (2018). Skip-connected deep convolutional autoencoder for restoration of document images. In *Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR)*. <https://doi.org/10.1109/icpr.2018.8546199>