


Article

Gradual OCR: An Effective OCR Approach Based on Gradual Detection of Texts

Youngki Park ¹  and Youhyun Shin ^{2,*}

¹ Department of Computer Education, Chuncheon National University of Education, Chuncheon 24328, Republic of Korea; ypark@cnue.ac.kr

² Department of Computer Science and Engineering, Incheon National University, Incheon 22012, Republic of Korea

* Correspondence: yhshin@inu.ac.kr

Abstract: In this paper, we present a novel approach to optical character recognition that incorporates various supplementary techniques, including the gradual detection of texts and gradual filtering of inaccurately recognized texts. To minimize false negatives, we attempt to detect all text by incrementally lowering the relevant thresholds. To mitigate false positives, we implement a novel filtering method that dynamically adjusts based on the confidence levels of recognized texts and their corresponding detection thresholds. Additionally, we use straightforward yet effective strategies to enhance the optical character recognition accuracy and speed, such as upscaling, link refinement, perspective transformation, the merging of cropped images, and simple autoregression. Given our focus on Korean chart data, we compile a mix of real-world and artificial Korean chart datasets for experimentation. Our experimental results show that our approach outperforms Tesseract by approximately 7 to 15 times and EasyOCR by 3 to 5 times in accuracy, as measured using a Jaccard similarity-based error rate on our datasets.

Keywords: optical character recognition; gradual OCR; gradual text detection; gradual low-quality filtering

MSC: 68T01



Citation: Park, Y.; Shin, Y. Gradual OCR: An Effective OCR Approach Based on Gradual Detection of Texts. *Mathematics* **2023**, *11*, 4585. <https://doi.org/10.3390/math11224585>

Academic Editors: Xiangtao Zheng, Jinchang Ren and Ling Wang

Received: 4 October 2023

Revised: 30 October 2023

Accepted: 7 November 2023

Published: 9 November 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Tesseract [1] and EasyOCR [2] are among the most prominent open-source OCR applications. Their multilingual capabilities have played a significant role in their widespread adoption for OCR tasks in languages other than English, in both academia and industry. For instance, Cho et al. [3] proposed a method to enhance the recognition of smaller Korean texts using Tesseract OCR. Kim et al. [4] utilized both Tesseract and EasyOCR for OCR on documents and electronic displays. Similarly, Hyeong et al. [5] employed EasyOCR for OCR on Material Safety Data Sheets (MSDS), while Moon et al. [6] introduced an edutech system that employed EasyOCR for children's handwritten texts. Although many other OCR models exist, to the best of our knowledge, no other open-source OCR approach surpasses these in terms of accuracy, especially in supporting the Korean language.

However, these existing approaches often face difficulties when handling texts that are noisy, small, or presented in a diverse range of fonts and uneven distributions. Korean chart data, in particular, present a challenge due to their numerous text regions, varied font styles, backgrounds, noise levels, and sizes. In this paper, we introduce a new OCR approach to address this challenge. Our main intuition is that text recognition can be enhanced by considering the manner in which its corresponding text regions are detected. Our approach emphasizes two primary strategies: first, the “gradual detection” of texts using different thresholds; second, the “gradual filtering” of inaccurately recognized texts based on their detection thresholds and confidence scores. We also integrate other techniques to further

enhance the OCR performance. Our experiments, conducted on both unique real-world and artificially created data, show the potential of our method, which outperforms leading open-source OCR solutions by approximately four times using the Jaccard similarity-based error rate measure.

Our work stands apart from traditional chart-centric OCR methods. While many studies target OCR for chart data [7–9] or focus on chart comprehension, including question answering [10] and summarization [11], our primary aim is general OCR: extracting text from images. Though we showcase its efficiency with Korean chart data, our method is adaptable to a diverse range of datasets. The main contributions of our research include the following.

- We introduce novel OCR datasets featuring Korean chart data. These datasets comprise (a) real-world noisy chart data sourced from the web, (b) bar charts constructed with various small fonts, and (c) bar charts featuring diverse small, rotated texts (Section 3).
- We unveil a novel and effective OCR approach built on a range of techniques. Specifically, we introduce the “gradual text detection” algorithm to reduce false negatives and a distinct “gradual filtering” process to minimize false positives (Section 4).
- We carry out detailed experiments to evaluate the performance of our approach. Through the systematic application of the techniques that we propose, we gauge the improvements in OCR performance. Our final model considerably outshines the best open-source multilingual OCR applications on our datasets, as evidenced by the Jaccard similarity-based error rate measure (Section 5).

Our research question is as follows:

- Can our approach significantly enhance the final text recognition results by gradually altering the behaviors of the text detection and recognition models?

2. Related Work

At the time of writing, Tesseract [1] and EasyOCR [2] are two of the most popular publicly available OCR software programs. Both support multiple languages, making them suitable for Korean, which is the focus of our paper. It is widely acknowledged that EasyOCR outperforms Tesseract in recognizing Korean texts in various formats, such as documents, electronic displays, and handwritten notes [4,6]. EasyOCR employs the CRAFT [12] text detection algorithm and the CRNN [13] text recognition algorithm. To the best of our knowledge, there are no open-source, widely used approaches based on Vision Transformers [13], such as TrOCR [14].

There are many approaches that attempt OCR for chart data. ChartOCR [7] is one of the most renowned methods for this purpose. As in our experiments, ChartOCR can recognize bar charts, pie charts, and line charts. Chart-RCNN [8] is designed to target line charts, especially from camera images that may not be clean. Chart-Text [9] focuses on pie and bar charts. While each approach utilizes different algorithms for the recognition of various chart types, none of them take into account the recognition of the Korean language.

Rather than merely extracting information from charts, there are alternative approaches that abstractly summarize chart data. For instance, OpenCQA [10] aims to answer open-ended questions based on chart question answering. Similarly, Chart-to-Text [11] offers datasets for chart summarization tasks.

There are various approaches to OCR for the Korean language. Cho et al. [3] addressed the challenge of recognizing small Korean texts by employing the SRCNN [15] in tandem with Tesseract OCR. Kim et al. [4] focused on recognizing text in documents and electronic displays using both Tesseract OCR and EasyOCR. Hyeong et al. [5] used EasyOCR to recognize text from Material Safety Data Sheets (MSDS). Kim et al. [16] aimed to identify text on the side covers of books placed on bookshelves by employing the Google Cloud Vision API. Lastly, Moon et al. [6] developed a solution to recognize handwritten letters penned by children for educational purposes. A common limitation among these approaches is their heavy reliance on EasyOCR and Tesseract OCR. Although one method employs the Google

Vision API, such an API-based approach is not aligned with our research interests. Our primary objective is to introduce a publicly available Korean model that does not depend on APIs.

There are numerous super-resolution approaches that have been proposed. Among these, we employ one of the most effective super-resolution methods to enhance our optical character recognition task. To the best of our knowledge, Real-ESRGAN [17] is among the most efficient and well-regarded techniques. It builds upon the work of ESRGAN [18]. Notably, it is reported to outperform other methods, such as DAN [19], CDC [20], RealSR [21], and BSRGAN [22].

3. Dataset Construction

In this section, we describe the construction of three distinct datasets for our subsequent experiments. Given our focus on Korean chart optical character recognition, we collected and generated a diverse set of chart data. As detailed in Table 1, the first dataset is composed of chart data sourced from the world wide web. In contrast, the second and third datasets are composed of automatically generated chart data. A detailed description of these datasets is provided below.

Table 1. Summary of the datasets constructed for our experiments. Dataset 1 comprises charts collected from the web, while Dataset 2 and Dataset 3 both consist of artificially generated charts.

Dataset	# of Images	Description
Dataset 1	91	Collected Charts (Bar/Pie/Line/Scatterplot) from the Web
Dataset 2	500	Artificially Generated Charts (Bar Charts)
Dataset 3	500	Artificially Generated Charts (Bar Charts with Rotated Labels)

The first dataset was sourced from documents provided by governments or their affiliated departments and agencies. These documents are typically in the hwp or hwpX format, which are commonly used in South Korea. From these, we randomly chose 100 charts. After removing nearly identical charts, we were left with 91 distinct charts. The dataset features a variety of chart types, including pie charts, line charts, bar charts, and labeled scatterplots. While some of the charts are of high quality, a significant number are of very low resolution. A few charts are so noisy that they are nearly indecipherable, even to the human eye. This dataset is intended for the evaluation of the effectiveness of OCR techniques using real-world data.

The second and third datasets were artificially generated using Python's Matplotlib. While these datasets are cleaner than the first, they present unique challenges. Specifically, the font sizes used are minuscule, and the datasets incorporate a wide variety of fonts—48 different types of Nanum fonts (<https://hangeul.naver.com/font>, accessed on 25 September 2023), to be precise. Nanum fonts are widely used free fonts in South Korea. Another challenge stems from the random representation of words. Without any discernible pattern, word recognition becomes notably more difficult.

Further details of the datasets are as follows.

- They comprise bar charts, with each chart showcasing 5 to 10 vertical bars.
- Each vertical bar is assigned a random value ranging from 0 to 100, rounded to two decimal places.
- The Okt package in KoNLPy [23] was employed to extract the top 500 most frequent nouns that formed all the labels in the charts.
- Every vertical bar has a corresponding label composed of 1 to 4 Korean nouns.
- The x and y labels are randomly assigned between 3 and 6 Korean nouns.
- Chart titles are formed from a random assortment of 5 to 10 Korean nouns.
- The y-axis labels are vertically placed, while the x-axis labels are horizontally positioned in Dataset 2 and randomly rotated in Dataset 3.

For clarity, examples from Dataset 2 and Dataset 3 are illustrated in Figures 1 and 2, respectively. It is important to note that the composition of words lacks semantic meaning. Therefore, OCR techniques should identify each Korean letter independently, without relying on post-processing capabilities such as using a language model. In Section 5, we demonstrate how our methods, introduced in Section 4, effectively recognize these datasets.

Note that we chose chart data as our domain because of the challenges associated with its accurate recognition. Furthermore, we specifically focused on Korean data because of the complexity and diversity of its characters, which are often difficult to identify correctly. We believe that if our approach is effective with such intricate datasets, it can likely be generalized to many other domains. Although we did not gather a large dataset, we observed consistent experimental results across different datasets. We discuss this in detail in the experimental section.

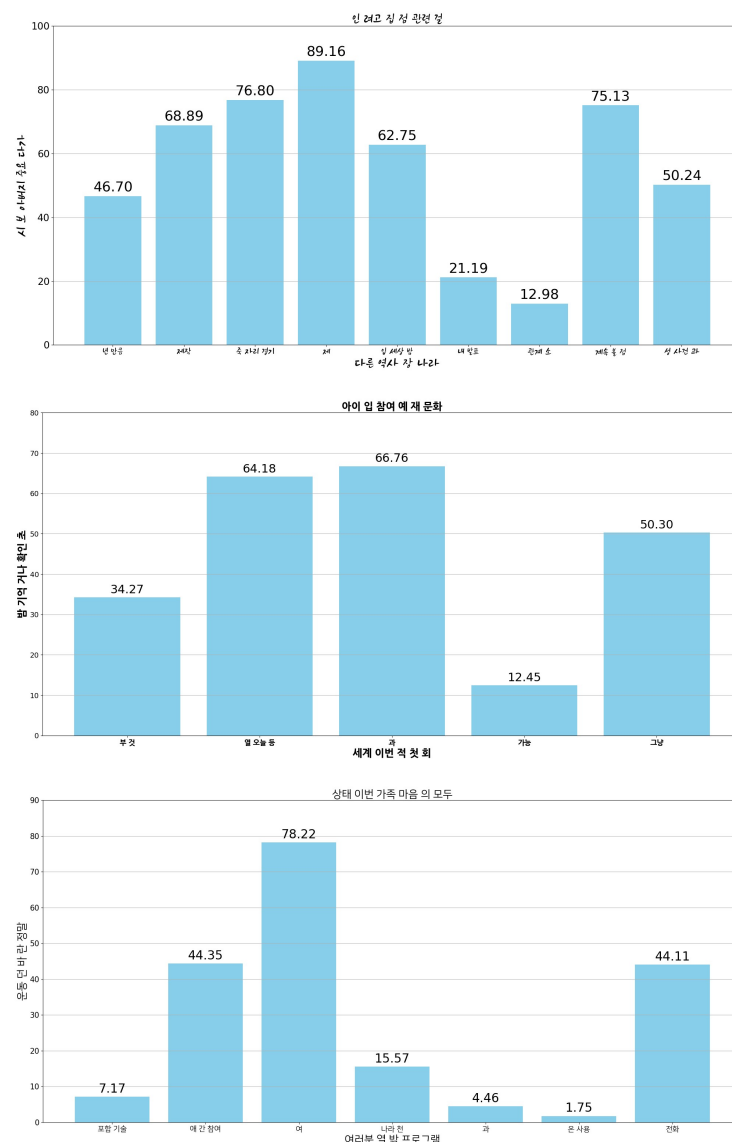


Figure 1. Three bar chart examples from Dataset 2 that we generated. Labels on the bar charts are randomly assigned using numbers and Korean characters. The sizes of the charts, font types, and font sizes vary randomly.

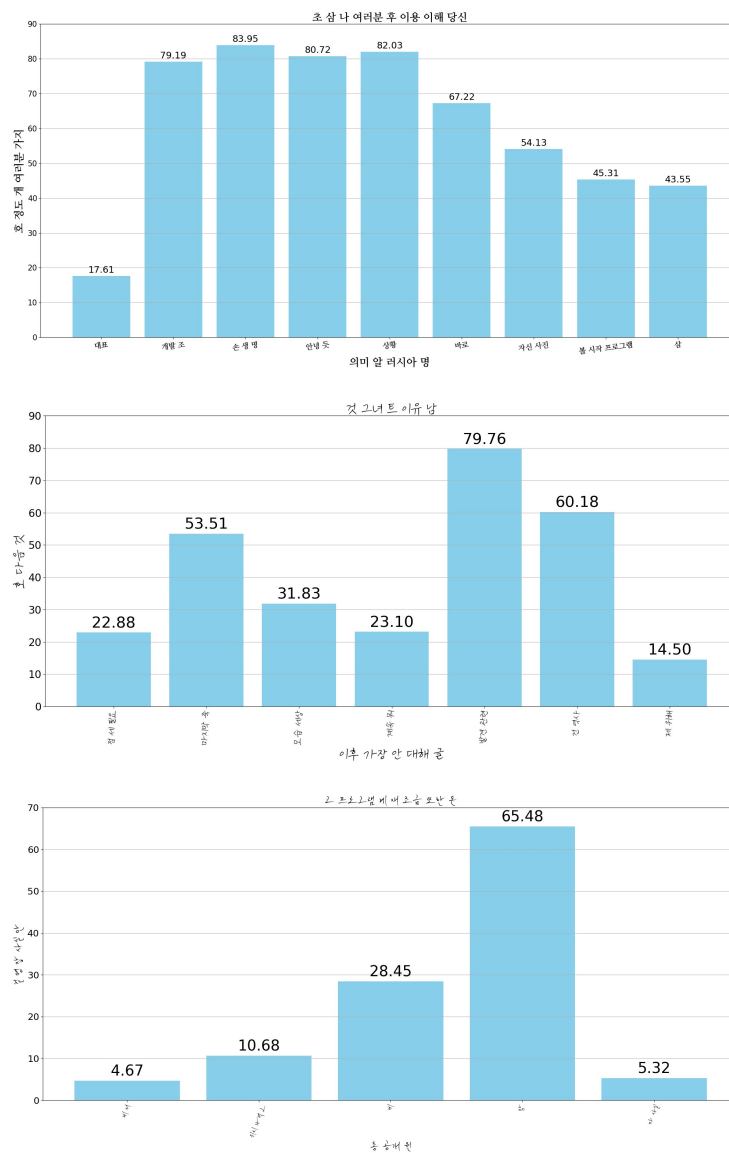


Figure 2. Three bar chart examples from Dataset 3 that we generated. Unlike Dataset 2, in this dataset, all the x-axis labels are rotated randomly.

4. Gradual OCR

4.1. Overview of Our Approach

Figure 3 illustrates the workflow of our proposed method. As depicted, our system encompasses five primary components: (1) image preprocessor, (2) text detection module, (3) text region processor, (4) TrOCR, and (5) filtering Module.

- **Image Preprocessor:** This component, while optional, upscales the input image before the text detection or recognition steps. We anticipate an enhancement in recognizing diminutive text, as found in Datasets 2 and 3, given that the existing literature indicates a performance boost from super-resolution when dealing with small text [3].
- **Text Detection Module:** Before proceeding to text recognition, we employ a text detection module—a strategy commonly adopted in optical character recognition. Similar to EasyOCR [2], we utilize the CRAFT algorithm [12] for text detection. Differing from EasyOCR, we integrate a link refiner to optimize the results. Additionally, we introduce our “gradual text detection” method, which iteratively performs text detection and link refinement using varied thresholds.

- Text Region Processor: Following text detection, the conventional next step is text recognition. However, we incorporate supplementary techniques. Considering that text recognition modules are typically trained on unrotated images, our system transforms detected text regions into unrotated ones to improve the recognition accuracy. As an optional speed-enhancing step, we have designed a “batch decoding” method that merges all text regions into a single region.
- TrOCR (Text Recognition Module): For each identified region, the TrOCR [24] model, based on the Vision Transformer [14], is utilized for text recognition. This model is composed of both the TrOCR encoder and decoder. We discovered that a straightforward “autoregression” method yields optimal results in text recognition.
- Filtering Module: The output might contain false positives for various reasons. We introduce an innovative method to pinpoint these inaccuracies by assessing the decoding results’ quality. A basic filtering technique is employed to remove these errors. Concurrently, we introduce the “gradual low-quality filtering” strategy, which synergizes effectively with our “gradual text detection” method.

4.2. Image Preprocessor

In the image preprocessing phase, we initially apply resizing—either 2 times ($2\times$) or 4 times ($4\times$). This resizing is executed using Python’s Pillow library, where the width is increased by $2\times$ or $4\times$, and the height is adjusted correspondingly. Subsequent to this, we introduce two methods aimed at further enhancing the detection and recognition outcomes: antialiasing and Real-ESRGAN [17]. For antialiasing, we utilize the “Image.ANTIALIAS” parameter from the Pillow library. Optionally, users can select Real-ESRGAN, an advanced version of ESRGAN [18], which has demonstrated superior performance compared to other models, such as DAN [19], CDC [20], RealSR [21], and BSRGAN [22].

While prior research suggests that super-resolution can bolster the recognition results [3], we emphasize that this step is discretionary. This is because some text recognition models have been trained on noisy images, meaning that the super-resolution process may not necessarily enhance the final outcomes.

4.3. Text Detection Module

Text detection typically begins by identifying text regions prior to actual text recognition. We employ the CRAFT algorithm [12], primarily for its multilingual prowess, particularly its demonstrated efficacy for the Korean language. This algorithm operates using three foundational threshold values: text threshold, link threshold, and low text threshold. We have chosen the default values $T_0 = (0.7, 0.4, 0.4)$. In contrast to EasyOCR [2], our algorithm integrates a pre-trained link refiner from <https://github.com/clovaai/CRAFT-pytorch>, accessed on 25 September 2023 to enhance the performance.

Figures 4 and 5 highlight the effects of different thresholds. The top images of Figures 4 and 5 adopt thresholds of T_0 . The second, third, and fourth images in these figures employ thresholds sequentially reduced by a factor of 2:

$$T_{i+1} = \frac{T_i}{2} \quad (1)$$

This leads to

1. $T_0 = (0.7, 0.4, 0.4)$
2. $T_1 = \frac{T_0}{2} = (0.35, 0.2, 0.2)$
3. $T_2 = \frac{T_1}{2} = (0.175, 0.1, 0.1)$
4. $T_3 = \frac{T_2}{2} = (0.0875, 0.05, 0.05)$

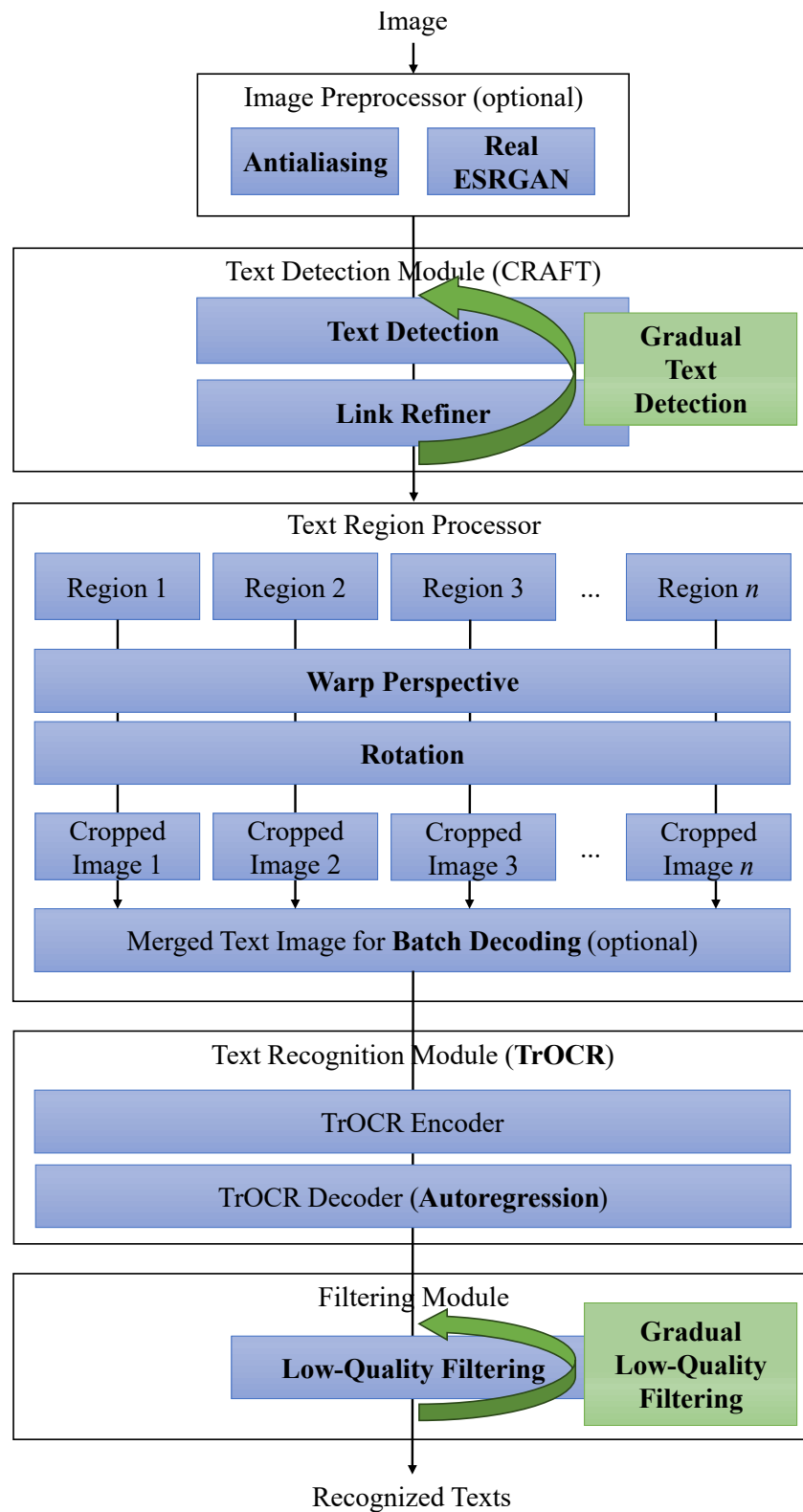


Figure 3. Overview of our gradual OCR framework, which incorporates novel methods like gradual text detection and gradual low-quality filtering to optimize performance.



Figure 4. An illustration of how gradual detection techniques operate internally, varying the parameter S from 0 to 3 (shown in the top four figures), and the corresponding merged results (presented in the bottom figure).



Figure 5. Similar to Figure 5, this illustration presents another example of how gradual detection techniques are applied. In the fourth chart, where S is 3, an empty rectangle is identified. We anticipate that the gradual detection and gradual low-quality filtering will effectively eliminate such irrelevant rectangles.

Elevated threshold values have the propensity to yield false negatives. As an illustration, the top image in Figure 4 using thresholds at T_0 omits the first two x -axis labels and the bottom y -axis label. On the other hand, the fourth image in Figure 5, using thresholds at T_3 , exhibits false positives. A non-text region is mistakenly marked, and the size of the identified text regions is considerably larger than those pinpointed with higher threshold values.

To address these challenges, we introduce the “gradual text detection” strategy.

- Initialize an empty list, L .
- Execute text detection and link refinement with thresholds T_0 and incorporate the resulting boxes into L .
- For a parameter S (indicating steps), repeat for S iterations:
 1. In the i th iteration, apply T_i thresholds.
 2. Reinitiate text detection and the link refiner using the updated thresholds.
 3. For each newly identified text region r , if r has zero overlap with boxes in L , incorporate r into L .

In this algorithm, we prioritize boxes from higher thresholds to minimize false positives. At the same time, we include boxes from lower thresholds to reduce false negatives. For instance, the bottom figures in Figures 4 and 5 show all detected text regions with T_0 , T_1 , T_2 , and T_3 . In our gradual text detection strategy, text regions with T_0 (represented as red boxes) are given the highest priority. The green, blue, and cyan boxes are then considered in descending order of priority. However, a challenge with this approach is determining the optimal parameter S . To address this limitation, we introduce the gradual low-quality filtering technique in Section 4.6.

4.4. Text Region Processor

Each detected text region might be tilted, so we apply “perspective transformation” using OpenCV (<https://opencv.org/>, accessed on 25 September 2023). First, we calculate the width and height of the detected quadrilateral, referred to as a “box”, by measuring the Euclidean distance between the box’s vertices. The width W is determined by the maximum horizontal distance between the opposing vertices of the quadrilateral:

$$W = \max(\|\mathbf{box}_0 - \mathbf{box}_1\|_2, \|\mathbf{box}_2 - \mathbf{box}_3\|_2) \quad (2)$$

Similarly, the height H is determined by

$$H = \max(\|\mathbf{box}_0 - \mathbf{box}_3\|_2, \|\mathbf{box}_1 - \mathbf{box}_2\|_2) \quad (3)$$

Based on these dimensions, we define a new rectangle with vertices at $[0, 0]$, $[W - 1, 0]$, $[W - 1, H - 1]$, and $[0, H - 1]$. Using this rectangle, we calculate the transformation matrix with the OpenCV function `cv2.getPerspectiveTransform()`. With this matrix, we then obtain the corrected text region using the `cv2.warpPerspective()` function.

The next step involves rotating the image 90 degrees clockwise if its width is smaller than its height. This adjustment ensures that vertically oriented text, often found in chart data, is accounted for. We achieve this rotation using OpenCV’s `cv2.transpose()` and `cv2.flip()` functions. Following this, we crop the images based on detected text regions.

While we now have cropped images for each text region, recognizing each cropped image individually could be time-consuming. Thus, we offer an option to merge all the cropped images into a single image for expedited recognition, a process that we refer to as “batch decoding” in this paper. This consolidation means that only one recognition process is necessary, potentially increasing the speed considerably. However, since each cropped image varies in size, we resize smaller images to match the size of the largest image, maintaining their width-to-height ratio. It is worth noting that this merging process does not always ensure optimal recognition quality, as many text recognition models are not trained on such merged images.

4.5. TrOCR

The subsequent step involves recognizing texts from detected text regions. For this task, we employ TrOCR [24], which is built upon the Vision Transformer [14], one of the most effective text recognition methods available. TrOCR comprises a TrOCR encoder and a TrOCR decoder. Clearly, the types of TrOCR encoder and decoder utilized play a pivotal role in achieving a highly accurate model. There are a variety of encoder and decoder types that can be employed, and they can be fine-tuned using different datasets.

At the time of writing, there are three publicly available TrOCR Korean models. We refer to these models as TrOCR1, TrOCR2, and TrOCR3, respectively, and we incorporate them in our experiments. Below are the model specifications.

- TrOCR1: team-lucid/trocr-small-korean (<https://huggingface.co/team-lucid/trocr-small-korean>, accessed on 25 September 2023). This model's TrOCR encoder is based on DeiT, while the TrOCR decoder derives from RoBERTa. The model was trained using six million images generated by synthtiger [25].
- TrOCR2: daekeun-ml/ko-trocr-base-nsmc-news-chatbot (<https://huggingface.co/daekeun-ml/ko-trocr-base-nsmc-news-chatbot>, accessed on 25 September 2023). In this setup, the "facebook/deit-base-distilled-patch16-384" model acts as the TrOCR encoder, while the "klue/roberta-base" model functions as the decoder. The training data encompass a range of sources, including a news summarization dataset.
- TrOCR3: ddoobokki/ko-trocr (<https://huggingface.co/ddobokki/ko-trocr>, accessed on 25 September 2023). For this model, microsoft/trocr-base-stage1 is the chosen TrOCR encoder, and snunlp/KR-BERT-char16424 is the decoder. The OCR training data were sourced from AI-Hub (<https://www.aihub.or.kr/>, accessed on 25 September 2023).

We also employ a naive autoregression technique, instead of TrOCR's default "generate" function, for inference. Although our autoregression approach is slower, it offers superior performance compared to the generate() function. We set the maximum decoding length to 512.

4.6. Filtering Module

In our text detection method, using low threshold values can increase the likelihood of false positives. To address this, we introduce a novel filtering technique that leverages the recognition confidence of each identified text segment. The confidence is computed from the average confidence of its constituent letters.

For a specific recognized text segment, its recognition confidence, RC_{text} , is defined by

$$RC_{\text{text}} = \frac{1}{n} \sum_{i=1}^n c_i \quad (4)$$

Here, n denotes the number of letters in the recognized text, while c_i represents the recognition confidence of the i -th letter.

Our preliminary filtering algorithm, termed "low-quality filtering", operates as follows: for each recognized text, if its recognition confidence falls below a filtering threshold of 0.8, the text is excluded.

Nevertheless, this initial approach does not consider the "detection confidence" of the texts. Intuitively, even if two recognized texts share the same confidence, a text with lower detection confidence is arguably less likely to be accurate compared to the other. Hence, by integrating both the detection and recognition confidence, we introduce the gradual low-quality filtering algorithm as follows. Firstly, we determine the filtering threshold, F , using the text detection threshold, T . Their interrelation is described as

$$F(T_i) = 1 - \left(\frac{1}{2}\right)^{i+1} \quad (5)$$

where i spans from 0 to $m - 1$, and m signifies the highest detection step.

For any detected text with recognition confidence, RC_{text} , and a corresponding detection threshold T_i ,

$$\text{If } RC_{\text{text}} < F(T_i), \text{ the recognized text is omitted.} \quad (6)$$

This improved algorithm ensures the balanced consideration of both the detection and recognition confidence. For illustration, consider the examples in Figures 4 and 5. In the first subfigure of Figure 4, a text segment detected at threshold T_0 has recognition confidence of 0.77. It is retained because its confidence surpasses the filtering criterion for $F(T_0) = 0.5$. On the other hand, in the fourth subfigure of Figure 5, the text identified at threshold T_3 with confidence of 0.65 is excluded since it fails to meet the threshold criterion of $F(T_3) = 0.9375$.

5. Experiments

5.1. Experimental Setup

In our experiments, we utilized the three datasets constructed in Section 3. For each dataset, we compared the performance of our approaches with that of existing methods.

Recognition performance is commonly measured using either the Character Error Rate (CER) or Word Error Rate (WER). However, we argue that these measures are not suitable for a fair evaluation of our approach, especially when dealing with Korean data. In the Korean language, a single noun can consist of multiple sub-nouns, leading to multiple correct answers when calculating the WER. Additionally, we believe that the CER might not be the most appropriate metric since it can penalize misrecognized texts excessively. For instance, consider a reference text $A = \text{"I am a boy"}$. If the recognized text is $B = \text{" "}$ (an empty string) and another recognized text is $C = \text{"I am a boy and and and and"}$, the CER between A and B is 1, while the CER between A and C is 1.4545. This implies that the empty string B is a better-recognized result than string C , even though C closely matches the reference string but fails to terminate correctly. Such discrepancies are even more pronounced when working with Korean recognition in noisy and small texts. To encourage our algorithms to recognize texts without the risk of misrecognition, we propose a new evaluation metric.

We introduce the Jaccard error rate (JER) based on the Jaccard similarity for bags [26]. It is defined as follows: given two multisets A (the reference) and B (the candidate) consisting of letters with all blank spaces ignored, the JER measures the dissimilarity between A and B . It is given by

$$JER(A, B) = 1 - \frac{2|A \cap B|}{|A| + |B|}$$

Using this measure, the JER between A and B is 1, whereas the JER between A and C is approximately 0.4815, making the latter a better recognition result than a blank.

The JER can be computed using either macro-averaging or micro-averaging methods. "Macro JER" is defined as the arithmetic mean of individual JER values. In contrast, "Micro JER" is calculated based on (1) the total multiset of reference characters and (2) the total multiset of candidate characters.

In addition to JER, we also track the elapsed time, reported in seconds, with the total elapsed time provided for the entire dataset. Time measurements are conducted using an 11th Gen Intel[®] Core[™] i7-11800H @ 2.30 GHz CPU, 16 GB RAM, and an NVIDIA GeForce RTX 3080 Laptop GPU, all running on Ubuntu 20.04.6 LTS via WSL2.

We use three popular open-source OCR programs that support the Korean language as comparators: Tesseract [1], EasyOCR [2], and three variations of the TrOCR models detailed in Section 4.5. We introduce nine distinct versions of our approach, each incrementally applying our techniques. Detailed experimental results are discussed in the following subsections.

5.2. Experimental Results

We summarize the experimental results in Table 2. For each type of experiment, the best-performing numbers are highlighted in bold and underlined. A detailed discussion of these results is provided below.

First, the results of existing approaches are reviewed.

- Tesseract v5.3.1 Windows: While Tesseract is a recognized OCR solution for the Korean language, it does not deliver satisfactory results on our datasets. Specifically, for our first dataset, which features real-world noise, the Macro JER exceeds 0.5. Its performance is somewhat better on our second dataset, which uses a variety of small fonts. These results indicate Tesseract’s challenges with noisy datasets. Since our dataset is exclusively in Korean, the Korean-only recognition setting (“ko”) outperforms the combined Korean and English setting (“ko + en”).
- EasyOCR 1.7.0: EasyOCR is among the most popular open-source OCR packages available. Consistent with its reputation, it demonstrated commendable performance across all our datasets. In our tests, it consistently surpassed Tesseract, echoing previous studies that found EasyOCR superior to Tesseract, especially in recognizing Korean text [4,6]. This was observed when the settings were tailored for either exclusively Korean or both Korean and English. Nevertheless, much like Tesseract, EasyOCR encountered challenges with real-world noisy data (Dataset 1) and with datasets containing rotated text (Dataset 3).
- TrOCR Models: We employed three TrOCR models, as introduced in Section 4.5 (TrOCR1, TrOCR2, and TrOCR3). Although TrOCR1 outperforms the other two models, its Macro/Micro JER exceeds even 0.9, indicating that most recognition results are incorrect. These results suggest that using TrOCR by itself, without additional techniques, is challenging when applied to both real-world and artificial datasets.

Table 2. Summary of experimental results across various approaches and techniques.

#) Model Name	Dataset 1			Dataset 2			Dataset 3		
	Macro JER	Micro JER	Time (s)	Macro JER	Micro JER	Time (s)	Macro JER	Micro JER	Time (s)
Existing Approaches									
Tesseract v5.3.1 Windows									
(1) en	0.7662	0.7448	15.3270	0.5419	0.5373	135.0326	0.5349	0.5249	134.9901
(2) ko	<u>0.5227</u>	<u>0.4621</u>	<u>13.7473</u>	<u>0.2480</u>	<u>0.2304</u>	<u>126.7941</u>	<u>0.3577</u>	<u>0.3398</u>	<u>127.7314</u>
(3) en + ko	0.5609	0.5074	23.3183	0.2748	0.2609	169.1845	0.3910	0.3755	170.3775
EasyOCR 1.7.0									
(4) en	0.5096	0.5189	<u>15.2087</u>	0.4924	0.4931	176.6375	0.4631	0.4647	<u>178.3048</u>
(5) ko	0.1795	0.1839	15.2353	<u>0.1039</u>	<u>0.1030</u>	<u>173.0024</u>	<u>0.1843</u>	<u>0.1823</u>	184.9896
(6) en + ko	<u>0.1745</u>	<u>0.1794</u>	15.7230	0.1050	0.1041	176.6355	0.1873	0.1854	184.6961
TrOCR Models									
(7) TrOCR1	<u>0.9487</u>	<u>0.9505</u>	<u>8.6719</u>	<u>0.9559</u>	<u>0.9540</u>	47.7345	<u>0.9674</u>	<u>0.9659</u>	44.4016
(8) TrOCR2	0.9979	0.9975	17.9105	0.9959	0.9961	235.6769	0.9965	0.9968	242.9406
(9) TrOCR3	0.9982	0.9976	10.9051	0.9956	0.9950	<u>44.2932</u>	0.9943	0.9937	<u>41.3378</u>

Table 2. Cont.

(#) Model Name	Dataset 1			Dataset 2			Dataset 3		
	Macro JER	Micro JER	Time (s)	Macro JER	Micro JER	Time (s)	Macro JER	Micro JER	Time (s)
Our Approaches									
<u>Text Detection(T_0) + TrOCR Models</u>									
(10) TrOCR1	0.0552	0.0606	80.3815	0.1162	0.1150	401.6401	0.2237	0.2242	477.5391
(11) TrOCR2	0.6668	0.6689	306.4827	0.5302	0.5282	1430.5761	0.6053	0.6058	1400.2860
(12) TrOCR3	0.0707	0.0757	222.1472	0.1125	0.1107	1024.3570	0.2150	0.2144	995.9418
<u>Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(L)</u>									
(13) L = 20	0.0496	0.0580	73.0700	0.0459	0.0449	379.3420	0.0820	0.0811	366.7101
(14) L = 512	0.0536	0.0794	74.6489	0.0459	0.0449	366.9069	0.0820	0.0811	362.8143
<u>Antialiasing(M) + Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(512)</u>									
(15) M = 2	0.0489	0.0530	81.8446	0.0459	0.0449	543.4094	0.0839	0.0831	516.9085
(16) M = 4	0.0523	0.0561	112.1780	0.0466	0.0456	1204.3357	0.0847	0.0839	1205.7066
<u>Real ESRGAN(M) + Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(512)</u>									
(17) M = 2	0.0597	0.0688	161.6895	0.0531	0.0519	2306.2874	0.0928	0.0915	4119.9295
(18) M = 4	0.0672	0.0734	348.6670	0.0522	0.0508	6734.3283	0.0895	0.0884	12333.8697
<u>Text Detection(T) + Link Refiner + Warp + Rotation + TrOCR1(512)</u>									
(19) T = T_1	0.1027	0.0976	73.3372	0.0459	0.0449	380.7345	0.0604	0.0599	392.1491
(20) T = T_2	0.2469	0.2420	62.7877	0.0612	0.0606	378.5012	0.0721	0.0718	393.4250
(21) T = T_3	0.2910	0.2885	61.2715	0.0928	0.0933	380.1300	0.1102	0.1118	399.6759
<u>Gradual Detection(S) + Link Refiner + Warp + Rotation + TrOCR1(512)</u>									
(22) S = 1	0.0494	0.0743	85.8151	0.0372	0.0366	454.0669	0.0705	0.0701	474.0587
(23) S = 2	0.0498	0.0749	99.8336	0.0401	0.0399	519.2830	0.0734	0.0736	525.2052
(24) S = 3	0.0600	0.0882	115.3246	0.0453	0.0467	599.2412	0.0836	0.0870	620.1675
<u>Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(512) + Autoregression + Batch Decoding</u>									
(25) AR Only	0.0396	0.0475	122.7754	0.0360	0.0352	562.3832	0.0728	0.0721	595.6792
(26) AR+BD	0.2294	0.2910	77.6673	0.0745	0.0753	394.0620	0.1148	0.1153	394.5043
<u>Gradual Detection(S) + Link Refiner + Warp + Rotation + TrOCR1(512) + Autoregression + Low-Quality Filter</u>									
(27) S = 1	0.0432	0.0488	141.3093	0.0288	0.0282	697.1296	0.0575	0.0565	687.0164
(28) S = 2	0.0424	0.0481	156.3434	0.0283	0.0277	780.0101	0.0553	0.0543	795.0077
(29) S = 3	0.0424	0.0481	170.1998	0.0279	0.0275	856.4727	0.0544	0.0535	896.8290
<u>Gradual Detection(S) + Link Refiner + Warp + Rotation + TrOCR1(512) + Autoregression + Gradual Low-Quality Filter(S)</u>									
(30) S = 0	0.0396	0.0474	119.2158	0.0354	0.0346	560.9463	0.0690	0.0680	520.0968
(31) S = 1	0.0352	0.0420	132.7426	0.0244	0.0239	659.6777	0.0538	0.0531	640.4872
(32) S = 2	0.0346	0.0414	144.3790	0.0235	0.0230	777.9616	0.0512	0.0505	744.6360
(33) S = 3	0.0344	0.0412	158.9390	0.0231	0.0226	828.3957	0.0504	0.0497	835.1660

Second, we summarize the experimental results of our various approaches as follows.

- **Text Detection(T_0) + TrOCR Models:** Recall that TrOCR in isolation exhibits suboptimal performance on our datasets. However, when supplemented with the text detection techniques outlined in Section 4.3, even without further strategies such as link refinement or perspective transformation, its performance substantially improves. Specifically, while TrOCR1 alone demonstrates a Macro JER of 0.9487 in Dataset 1, its performance markedly improves to a Macro JER of 0.0552 when using the text detection technique, albeit with a tenfold increase in elapsed time. Through our analyses, TrOCR1 is found to be most effective on Dataset 1, while TrOCR3 excels on Datasets 2 and 3. Given that TrOCR3 demands significantly more elapsed time than TrOCR1

without outperforming it in subsequent experiments, we exclusively report results from the TrOCR1 model in ensuing experiments.

- **Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(L)**: In order to enhance the recognition quality, we employ the link refinement technique, as detailed in Section 4.3, and the perspective transformation and rotation techniques, introduced in Section 4.4. These methods prove particularly effective in Datasets 2 and 3, where images contain various types of text fragments, making accurate text detection crucial. Practically, the Macro JER in Dataset 3 witnessed a substantial decrease from 0.2237 to 0.0820. We also explored the impact of the text generation length limits, adjusting them from 20 (the original setting) to 512. A longer length limit resulted in a higher JER in Dataset 1, particularly increasing the Micro JER, suggesting that the recognized text for some entries did not terminate appropriately, thereby generating excessively long and inaccurate sequences. This issue can be mitigated using our gradual text detection and gradual low-quality filtering techniques, the results of which are demonstrated in subsequent experiments. Therefore, in the following experiments, we utilize a maximum length of 512.
- **Antialiasing(M) + Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(512)**: We additionally employ the image resizing and antialiasing techniques described in Section 4.2. Here, “M” denotes the multiplication factor for image resizing (e.g., $M = 2$ implies $2 \times$ width and $2 \times$ height, while $M = 4$ implies $4 \times$ width and $4 \times$ height). Initially, we resize using factor M and subsequently refine the resized image using the antialiasing technique. Experimental results indicate that when $M = 2$, the Macro JER in Dataset 1 is enhanced from 0.0536 to 0.0489, and the Micro JER is also improved, moving from 0.0794 to 0.0530. However, in artificial datasets with minimal noisy text (Dataset 2 and Dataset 3), the JER does not exhibit an increase. Despite slight improvements, we opt not to utilize this technique in subsequent experiments because the elapsed time significantly increases when the image size is already substantial (as in Dataset 2 and 3).
- **Real ESRGAN(M) + Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(512)**: We additionally apply image resizing and the Real ESRGAN technique introduced in Section 4.2. As previously described, “M” denotes the multiplication factor for image resizing. Surprisingly, Real ESRGAN does not outperform the antialiasing technique. This underperformance may stem from the fact that the TrOCR models were not trained on data refined by Real ESRGAN.
- **Text Detection(T) + Link Refiner + Warp + Rotation + TrOCR1(512)**: In these experiments, we varied the text detection threshold T from the original T_0 to T_1 , T_2 , and T_3 , respectively. The results are somewhat interesting: in Dataset 1, when using T_1 , both the Macro and Micro JER significantly deteriorate compared to using T_0 . Conversely, in Dataset 3, they improve. These outcomes suggest that the optimal parameters can vary depending on the dataset or images, inspiring our concept of gradual text detection and gradual low-quality filtering.
- **Gradual Detection(S) + Link Refiner + Warp + Rotation + TrOCR1(512)**: We use the gradual detection technique from Section 4.3 instead of a fixed threshold. The best JER is achieved when $S = 1$, which is better than when $S = 0, 2$, or 3. Since increasing S does not consistently improve the recognition, selecting the correct S value remains a challenge.
- **Text Detection(T_0) + Link Refiner + Warp + Rotation + TrOCR1(512) + Autoregression + Batch Decoding**: In these experiments, we additionally applied autoregression techniques and batch decoding techniques, as introduced in Sections 4.4 and 4.5. The notation “AR Only” in Table 2 indicates that only the autoregression technique was used, without batch decoding, while “AR + BD” signifies the employment of both autoregression and batch decoding. The experimental results reveal that utilizing autoregression substantially improves the recognition performance. Specifically, without the autoregression technique, the Micro JER is 0.794 in Dataset 1; with the technique, it markedly

decreases to 0.0475. Unfortunately, the batch decoding technique did not demonstrate satisfactory performance. Although it significantly reduced the elapsed time, its recognition performance notably degraded, especially in Dataset 1. In Dataset 1, texts typically had varying background colors and font sizes, factors that likely contributed to the diminished recognition performance.

- ***Gradual Detection(S)*** + Link Refiner + Warp + Rotation + TrOCR1(512) + Autoregression + ***Low-Quality Filter***: Here, we additionally apply both the gradual detection and low-quality filtering techniques. As mentioned in Section 4.6, the low-quality filtering technique employs a threshold of 0.8. Experimental results reveal that optimal recognition performance is achieved when $S = 2$ or $S = 3$. These findings contrast with those obtained using only the gradual detection technique, where $S = 1$ delivers the best performance. This underscores the significance of filtering incorrectly recognized results when employing our gradual detection technique.
- ***Gradual Detection(S)*** + Link Refiner + Warp + Rotation + TrOCR1(512) + Autoregression + ***Gradual Low-Quality Filter(S)***: This marks our final experiment utilizing both the gradual detection and gradual low-quality filtering techniques. As the parameter “S” is varied from 0 to 3, we observe a consistent increase in recognition performance, alleviating concerns over the specific choice of the S parameter. The final results are somewhat surprising: in terms of Micro JER, when $S = 3$, all datasets yield Micro JER values below 0.05. Particularly in Dataset 2, the Micro JER is significantly lower at 0.0226. Considering that the JER between “I am a boy.” and “I am a boy” is 0.0666, it is evident that our recognizer accurately identifies the majority of texts within our complex datasets.

Table 3 illustrates two example recognition results from Dataset 2. In this table, the model number corresponds to the same model number specified in Table 2. To provide concise representations, we report only false negatives (FN) and false positives (FP) for each model, instead of showcasing the full recognized results. For existing approaches, we selected model #2 (Tesseract), #5 (EasyOCR), and #7 (TrOCR). For our approach, we opted for models #10, #14, #19, #20, #21, and #33. In the first image (sample image 97 from Dataset 2), existing approaches produced a substantial number of false negatives and false positives (the quantities of FN + FP for each model are 78, 14, and 101, respectively). Similarly, for the second image (sample image 134 from Dataset 2), they produced 121, 12, and 88 FN + FP, respectively. In stark contrast, our final model (Model #33) yielded an FN + FP count of only 5 for the first image and 2 for the second. Out of the seven FN + FP in our model, four were misrecognitions of 0 as O. A noteworthy strength of our final model is its consistently robust performance across various images. For instance, while Model #20 performs well on the first image, evidenced by an FN + FP of merely 4, its performance dwindles on the second image, where FN + FP increases to 12. Conversely, while Model #14 exhibits strong performance on the second image (with an FN + FP of only 2), it falters on the first image, where the FN + FP ascends to 8.

We previously argued that the Jaccard error rate (JER) offers a more equitable evaluation measure for our approach. Therefore, we consistently use the JER throughout this paper. However, for clarity, we also provide a brief comparison of our method against other techniques using both the Character Error Rate (CER) and the Word Error Rate (WER). Table 4 presents a concise comparison of Tesseract, EasyOCR, and our final model, Gradual OCR. The results indicate a consistent trend across all evaluation measures: our approach surpasses the other methods in terms of macro-averaged JER, CER, and WER.

Table 3. Recognition results for two sample images from Dataset 2. Only false negatives and false positives are reported for concise presentation.

Model	# of FN + FP	FN (False Negatives)	FP (False Positives)
Sample Image 97 of Dataset 2			
Existing Approaches			
Model #2	780000001111122222334444 4556667789999결고과권데뒤란 람만문반부사사상새세약월은 잡제죽진짜큼태테표형화회	서저칸포
Model #5	14	0계권란부새약테형	#관네데작
Model #7	1010000000001112222223334 4444555666778999999 가결계고과권나날데도뒤란람 만문반발버부사사상새세 아약용월은이작잡장제제죽지 진짜큼태테표하형화회	:::전
Our Approaches			
Model #10	14	.006 고권반부새월은테형	2
Model #14	8	006계반월테	레
Model #19	6	0계고	'-레
Model #20	4	0	'-○
Model #21	12	006사잡회	'+○가임
Model #33	5	0계반	○레
Sample Image 134 of Dataset 2			
Existing Approaches			
Model #2	12122333344444455555666777 9999 결경과관길동동뒤뚝뚝런로물 살상서스안알역우운위점정집	((()))000001«===== [[× 흥 ㅠ 고교끄내내너더딱때때때띠 메몰버브비시씨아아에에여울 으으이입학호회
Model #5	12	...0서어위집	,,,꽃
Model #7	8800000000012223333344444 4555556667779999 결경과관길너동동뒤뚝뚝런로 물바살상서스안알어역우운위 이이점정지집	"" ;..... 나나넌는
Our Approaches			
Model #10	7	0서알어위집	'
Model #14	2	0알	
Model #19	5	0길뒤	11
Model #20	12	0길동뒤뚝살운이	○끄빛삼
Model #21	14	03길너동뒤뚝운이	+권너뚝뒤
Model #33	2	0	○

Table 4. Comparison of JER (Jaccard Error Rate), CER (Character Error Rate), and WER (Word Error Rate) across Tesseract (Model #2), EasyOCR (Model #5), and Gradual OCR (Model #33).

Model	Dataset 1			Dataset 2			Dataset 3		
	JER	CER	WER	JER	CER	WER	JER	CER	WER
Model #2 (Tesseract)	0.5227	0.6450	0.8406	0.2480	0.3525	0.5332	0.3577	0.4855	0.6839
Model #5 (EasyOCR)	0.1795	0.2795	0.5767	0.1039	0.1631	0.3367	0.1843	0.2755	0.4936
Model #33 (Gradual OCR)	<u>0.0344</u>	<u>0.0602</u>	<u>0.1803</u>	<u>0.0231</u>	<u>0.0411</u>	<u>0.0921</u>	<u>0.0504</u>	<u>0.0872</u>	<u>0.1698</u>

In conclusion, our final model demonstrates outstanding performance across various datasets. While we introduce numerous techniques to enhance the performance, we believe that the gradual detection and gradual low-quality filtering methods are especially significant. This is because the success of practical OCR relies largely on the quality of text detection models. Notably, these two techniques add substantial value to text detection algorithms when combined with text recognition models. We argue that our proposed methods can be smoothly integrated into a wide range of text recognition models. With a more advanced recognition model, it is likely that our techniques would deliver even better performance.

6. Conclusions

In this paper, we introduce “Gradual OCR”, a novel optical character recognition (OCR) approach focused on Korean chart data and utilizing two pivotal techniques: gradual detection and gradual low-quality filtering. The former involves the incremental detection of texts, while the latter dynamically filters inferior-quality texts. Our approach was validated using three distinct Korean chart datasets. The first dataset encompassed real-world, web-collected data, characterized by high noise levels. The second dataset was crafted with small, varied text fonts, while the third included rotated versions of these text instances, thereby assessing the model’s robustness against various typographical variations and orientations. The experimental results were indeed surprising. In terms of the macro-averaged Jaccard error rate, our method achieved scores of 0.0344, 0.0231, and 0.0504 for Datasets 1, 2, and 3, respectively. This performance was approximately 7 to 15 times better than Tesseract and 3 to 5 times better than EasyOCR. However, a notable drawback of our approach is its elapsed time. Recognizing the entirety of Datasets 1, 2, and 3 took about 167.3675, 887.8956, and 949.0274 s, respectively. This was roughly 7 to 12 times slower than Tesseract and 5 to 10 times slower than EasyOCR. Nonetheless, we expect that the elapsed time could be considerably reduced if a smaller text recognition model were used.

Our future work is twofold. Firstly, we aim to further enhance our approach by training more efficient TrOCR models, recognizing that our results harbor the potential for further refinement with advanced models. Secondly, we aspire to leverage the efficacy of gradual OCR to develop a novel methodology for a chart understanding model, ensuring comprehensive OCR application to chart data.

Author Contributions: Conceptualization, Y.P. and Y.S.; methodology, Y.P. and Y.S.; investigation, Y.P. and Y.S.; data curation, Y.P. and Y.S.; writing—original draft preparation, Y.P. and Y.S.; writing—review and editing, Y.P. and Y.S.; funding acquisition, Y.P. and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00250352).

Data Availability Statement: Some of the data from this study are available at <https://github.com/tooeworld/gradualocr> (accessed on 25 September 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Smith, R. An overview of the Tesseract OCR engine. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 23–26 September 2007; Volume 2, pp. 629–633.
2. EasyOCR. JaiedAI. 2023. Available online: <https://github.com/JaiedAI/EasyOCR> (accessed on 25 September 2023).
3. Cho, W.; Kwon, J.; Kwon, S.; Yoo, J. A Comparative Study on OCR using Super-Resolution for Small Fonts. *Int. J. Adv. Smart Converg.* **2019**, *8*, 95–101.
4. Kim, C.Y.; An, S.Y.; Jeon, E.J.; Cha, B.R. A Study on the Survey and Demonstration Test of OCR based on Open Source for AI OCR. In Proceedings of the Symposium of the Korean Institute of Communications and Information Sciences, Seoul, Republic of Korea, 23–25 August 2023; pp. 851–852.
5. Hyeong, K.K.; Duke, C.W. Development of OCR-based algorithm for information extraction from Material Safety Data Sheets (MSDS). In Proceedings of the Symposium of the Korean Institute of Communications and Information Sciences, Seoul, Republic of Korea, 23–25 August 2023; pp. 986–987.
6. Moon, J.; Kim, D.; Kim, E.; Oh, Y.; Jung, S.K.; Jang, J.; Kim, D. Development of OCR-based Spell check EduTech tool for Handwriting Education for children. In Proceedings of the Korea Computer Congress, Jeju, Republic of Korea, 25–28 June 2023; pp. 1640–1642.
7. Luo, J.; Li, Z.; Wang, J.; Lin, C.Y. Chartocr: Data extraction from charts images via a deep hybrid framework. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikola, HI, USA, 5–9 January 2021; pp. 1917–1925.
8. Li, S.; Lu, C.; Li, L.; Zhou, H. Chart-RCNN: Efficient Line Chart Data Extraction from Camera Images. *arXiv* **2022**, arXiv:2211.14362.
9. Balaji, A.; Ramanathan, T.; Sonathi, V. Chart-text: A fully automated chart image descriptor. *arXiv* **2018**, arXiv:1812.10636.
10. Kantharaj, S.; Do, X.L.; Leong, R.T.K.; Tan, J.Q.; Hoque, E.; Joty, S. Opencqa: Open-ended question answering with charts. *arXiv* **2022**, arXiv:2210.06628.
11. Kantharaj, S.; Leong, R.T.K.; Lin, X.; Masry, A.; Thakkar, M.; Hoque, E.; Joty, S. Chart-to-text: A large-scale benchmark for chart summarization. *arXiv* **2022**, arXiv:2203.06486.
12. Baek, Y.; Lee, B.; Han, D.; Yun, S.; Lee, H. Character region awareness for text detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9365–9374.
13. Shi, B.; Bai, X.; Yao, C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 2298–2304. [[CrossRef](#)] [[PubMed](#)]
14. Dosovitskiy, A.; Beyler, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv* **2021**, arXiv:2010.11929.
15. Dong, C.; Loy, C.C.; He, K.; Tang, X. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 295–307. [[CrossRef](#)] [[PubMed](#)]
16. Kim, S.; Park, J.; Kim, S.; Na, Y.; Jang, Y. Multi-book Label Detection Model using Object Detection and OCR. *J. Korean Inst. Inf. Technol.* **2023**, *21*, 1–8.
17. Wang, X.; Xie, L.; Dong, C.; Shan, Y. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In Proceedings of the IEEE/CVF international Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 1905–1914.
18. Wang, X.; Yu, K.; Wu, S.; Gu, J.; Liu, Y.; Dong, C.; Change Loy, C. Esrgan: Enhanced super-resolution generative adversarial networks. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
19. Huang, Y.; Li, S.; Wang, L.; Tan, T. Unfolding the alternating optimization for blind super resolution. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2020; Volume 33, pp. 5632–5643.
20. Wei, P.; Xie, Z.; Lu, H.; Zhan, Z.; Ye, Q.; Zuo, W.; Lin, L. Component divide-and-conquer for real-world image super-resolution. In *Computer Vision—ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020*; Part VIII 16; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 101–117.
21. Ji, X.; Cao, Y.; Tai, Y.; Wang, C.; Li, J.; Huang, F. Real-world super-resolution via kernel estimation and noise injection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 466–467.
22. Zhang, K.; Liang, J.; Van Gool, L.; Timofte, R. Designing a practical degradation model for deep blind image super-resolution. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 4791–4800.
23. Park, E.L.; Cho, S. KoNLPy: Korean Natural Language Processing in Python. In Proceedings of the 26th Annual Conference on Human & Cognitive Language Technology, Chuncheon, Korea, 10–11 October 2014; Volume 6, pp. 133–136.

24. Li, M.; Lv, T.; Chen, J.; Cui, L.; Lu, Y.; Florencio, D.; Wei, F. Trocr: Transformer-based optical character recognition with pre-trained models. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 13094–13102.
25. Yim, M.; Kim, Y.; Cho, H.-C.; Park, S. SynthTIGER: Synthetic Text Image GEneratorR Towards Better Text Recognition Models. In Proceedings of the International Conference on Document Analysis and Recognition, San Jose, CA, USA, 21–26 August 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 109–124.
26. Rajaraman, A.; Ullman, J. D. *Mining of Massive Datasets*; Cambridge University Press: Cambridge, UK, 2011.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.