High-Speed One-Shot Detection and Recognition of Low-Resolution Text Trained on Synthetic Data

Le Duy Huynh, Mihhail Dorosenko, and Bogdan Khomutenko
MCQ-Scan, Lille, France
Emails: {ld.huynh, m.dorosenko, b.khomutenko}@mcq-scan.com

***Abstract*—In this study, we address the challenge of text detection and recognition in low-resolution images. Addressing the dual constraints of limited training data and the necessity for real-time processing, we adopt a twofold strategy. Firstly, we introduce a pipeline for the generation of synthetic datasets that requires minimal manual annotation and is specifically designed for the context of low-resolution real-life text images. Secondly, we employ a streamlined neural network architecture based on the U-Net model, which concurrently executes text detection and recognition across multiple contextual layers. Our method demonstrates superior performance, achieving real-time processing speeds exceeding 120 fps, and is accurate even in challenging conditions where text characters are as small as five pixels in width. Our findings suggest that both the synthetic dataset generation pipeline and the neural network model are highly adaptable and can be easily modified for a broad range of applications.**

I. Introduction

The task of scene text detection and recognition is challenging, due to the variability in environmental factors such as illumination levels, contrast, blurring, geometric distortions, and variation in text content. These challenges are particularly relevant to our system, wherein our autonomous inventory robot captures images of a large section of the store with multiple price tags, therefore, each price tag is in low resolution. These price tags are often captured in sub-optimal conditions such as low light or odd angles. One example of such a scene can be seen in Figure 1. Existing Optical Character Recognition (OCR) methodologies, like those proposed by [1], [2], and [3], often rely on two separate neural networks that do text detection and recognition separately. While these methods are claimed to be effective in terms of accuracy, their two-stage approach incurs computational overhead due to the transfer of data between stages, which impedes their real-time processing capabilities. Furthermore, these methods are not optimized for recognizing very low-resolution text and exhibit suboptimal performance on our data; thus, they cannot be directly applied without further fine-tuning. To the best of our knowledge, there is no dedicated dataset for low-resolution text. This leads to the challenge of collecting a suitable OCR dataset.

This paper introduces our approach designed to overcome these limitations. Our key contributions are twofold: First, a novel synthetic dataset creation process, detailed in Section III-A, with annotations at multiple contextual levels (line, word, and characters), tailored for representing low-resolution text, where some characters' width is as low as five pixels. Second, the application of a streamlined, single-stage deep neural network that performs at the same time text localization and recognition, elaborated in Section III-B, specifically aims at achieving superior OCR accuracy on low-resolution inputs and can run at 120fps. We invite readers to Section IV for comprehensive experimental results.

Fig. 1. Example of a scene captured by our inventory robot (left). An example price tag crop (top right). Zooming in on the price tag (middle right) shows that the characters are in extremely low resolution, for example, the character I in "INS" is represented by about 5 pixels. Our system was trained to ignore certain fields in the price tag (e.g. "Unit=Piece"). As demonstrated in the bottom right crops, it recognizes correctly all the characters in the item description, but it still makes some errors in the very low-resolution text, (e.g., give an additional "2" in the barcode).

II. Related works

A. Scene text detection and recognition

The challenges of scene text reading have attracted increasing attention in the computer vision community. A majority of the approaches are based on a two-stage process involving separate detection and recognition tasks, each currently dominated by deep learning approaches. In this methodology, the detection task identifies text regions, typically a text

This research has been supported by BPI France, the French government, the Eurostars programme, and the European Union

line or a word, and the recognition task is subsequently performed on these regions. This bifurcates the research into two main areas: text localization and text recognition. Early text localization methods relied on engineered features and used sliding windows, or connected components, as seen in Huynh et al. [4]. More recent approaches such as Zhou et al. [5] and Quin et al. [6] have predominantly utilized deep detection models. On the task of text recognition, the works of Zhao et al. [7], Wang et al. [8], Bautista et al. [9], and Liao et al. [10] have demonstrated promising results in benchmark tests. In reviewing current approaches, it's important to acknowledge industry-standard OCR tools such as PaddleOCR [1], and Tesseract [2]. These tools have set benchmarks in text recognition accuracy and processing efficiency, representing key reference points for any new developments in the OCR field. They demonstrate that full image to text solutions can be achieved by combining a detection module with a recognition module.

However, despite its widespread adoption, this two-stage approach presents some limitations. Firstly, this approach prevents training both stages simultaneously, which potentially prevents the recognition stage from leveraging visual information available within the broader image context. Secondly, the recognition stage often has constraints on the input format, which requires cropping the text region and executing certain preprocessing steps, thereby introducing unnecessary overhead.

End-to-end trainable methods have been investigated to overcome the first limitation by integrating two stages through a Region of Interest (RoI) pooling operation. An early example of this approach is presented in Bušta et al. [11], who linked their YOLOv2 detection network with a recognition network via bilinear interpolation, enabling joint training of both networks. A similar paradigm is employed by Liao et al. [10], and Liu et al. [12], which incorporate RoI pooling before the second stage that utilizes a recurrent neural network. Although these methods benefit from end-to-end training, they are not truly single-stage, and the inter-stage pooling layer introduces unnecessary computational overhead. In light of this, we propose a genuinely single-stage model capable of performing both detection and recognition tasks within a single forward pass.

It should be acknowledged that the aforementioned methodologies do not address issues of extremely low resolution text, potentially attributable to a scarcity of datasets in this area of study. Gilbey et al. [13] discuss the challenges associated with low-resolution text within the scope of scanned documents, which is not directly applicable to the scenarios under consideration in our research.

B. Existing dataset

To develop our method, access to a representative training dataset is crucial, particularly one that encapsulates the challenges of low-resolution text that we face. There are publicly available datasets such as CORD [14], and ICDAR2019 [15] which provide insights into scanned documents. Although they represent certain challenges, particularly low-resolution, and corrupted characters due to printing or scanning errors, these datasets lack elements of scene text such as varied lighting and backgrounds. Large-scale real-life datasets like COCO-Text [16] and TextOCR [17] offer diverse texts but differ significantly from low-resolution price tags. These datasets offer a foundation but are not ideal for OCR of low-resolution text.

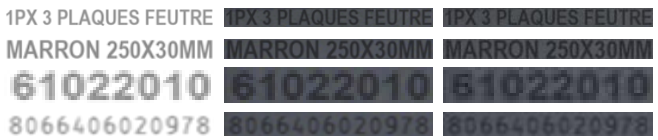

Fig. 2. First column: the rendered string; second column: alpha blend text string on the target patch; third column: after blurring, chromatic distortion, and JPEG compression. The final rendered pricetag is in the first row of Figure 3

Synthetic datasets, such as those introduced by Gupta et al. [18] and Jaderberg et al. [19], contribute to advancements in text localization and recognition within natural scenes. Nevertheless, they may not capture the full spectrum of challenges present in low-resolution text, including unique textures and lighting conditions. While these synthetic datasets cannot be directly applied to our specific problem, they suggest that a carefully constructed synthetic dataset could provide substantial generalization capabilities.

III. Methodology

A. Synthetic dataset for low-resolution text

Creating a robust model for text detection in low-resolution images requires a specialized dataset, particularly when existing datasets diverge significantly from our target domain. To address this, we've synthesized a dataset tailored to our application's specific needs, focusing on text fields in price tags captured by a retail inventory robot. Some examples are shown in Figure 3.

1) Preparations: We selected a set of 25 price tag crops that featured legible text among photos taken by our robot in a field condition. We annotated the position, size, scale, and color of the original text and then performed inpainting over the textual content. This set serves as our template. In addition to the template, we compiled a set of product metadata from the same store chains, which contains information that would appear on the price tags such as product name and short description, price, and internal reference. For missing information, such as barcodes, we dynamically generated plausible strings during the rendering process. To improve the accuracy of our application, we collected a set of free fonts that closely matched the font style used in the target stores and created variations of these fonts to mimic some common printing defects we observed.

2) Rendering and Composition: The rendering and composing of a target image for a text detection dataset involves several steps to simulate real-world variations and distortions that can occur in images. The first step in the process is to introduce variability in the text that will be rendered onto the target image, which in this case is a price tag. This

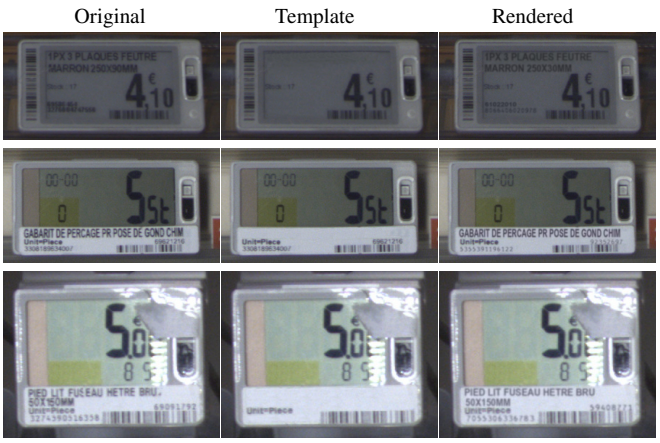

Fig. 3. Examples of synthetic image and the characters level annotations of synthetic text. In this example, we only place synthetic texts in 3 fields: product description, barcode, and internal code.

variability is achieved by randomly choosing different target templates, text strings, fonts, sizes, colors, and transformations. Transformations include blurring, which simulates out-of-focus images or motion blur, JPEG compression artifacts; and chromatic distortion. A level of alpha blending to blend the text patch onto the template is also randomly chosen. Additionally, the text may be stretched or squeezed.

We split the randomly selected strings into multiple lines based on the size of the target fields and the number of possible lines. Each text string is rendered at twice the selected font size, followed by a down-sampling operation using bicubic interpolation. If this procedure is not performed, characters in rendered text will display the same pixelization pattern. For instance, as illustrated in the first panel of the third row in Figure 2, the pair of digit '1's, despite sharing the same font, display several pixel-level disparities. Subsequently, each text is alpha blended onto a cropped section of the template, matching the text field dimensions, to effectuate a very similar text overlay on the price tag, as demonstrated in the second column of Figure 2. To emulate additional complexities encountered with low-resolution text, we apply the previously mentioned augmentations within this segment, resulting in challenging datasets comparable to real-world scenarios, particularly for very low-resolution text, as illustrated in the last column of Figure 2. Post-transformation, the segment is cropped to the text line's extent and then alpha-blended into the original price tag template. Depending on the specific use case, we can generate bounding box annotations for each line of text as well as for individual characters. After all the text fields have been added and blended into the price tag, a final augmentation is applied to the price tag as a whole to transform the text and the price tag coherently to maintain realism in the dataset.

B. A fast one-stage detection and recognition

In this section, we present the design of our Convolutional Neural Network that is adapted for the inference latency requirement of a real-time application on low-resolution text. We utilize a compact neural network designed for speed, precision, and the capability to be trained end-to-end. The segmentation results generated by this network are postprocessed using straightforward yet efficient techniques to yield the final string.

1) Architecture: Our system, depicted in Figure 4, is based on a U-Net architecture [20]. It performs both detection and recognition tasks for low-resolution text. To improve performance on small texts, including characters as narrow as five pixels, we choose to set the prediction resolution higher than that typically used by object detectors. We leverage an ImageNet pretrained backbone for hierarchical feature extraction. The decoder part of the network merges these features through upsampling and convolutional processes, blending detailed spatial information with high-level semantics. We conclude the decoding at half the original image resolution, which is a key factor in speeding up inference while maintaining precise text detection and recognition capabilities.

2) Multiple contextual level prediction: Given $\mathbb{C}$, the set of characters that we aim to detect, we trained a U-Net model to generate $|\mathbb{C}|+1$ prediction maps. In which $|\mathbb{C}|$ outputs are designed to facilitate both the localization and classification of individual characters. It is important to note that the white space is included in the characters set $\mathbb{C}$, therefore simplifying the post-processing task by eliminating the need for word segmentation. The last map is utilized for text line prediction, which is used in the post-processing to group characters into lines.

We construct the ground truth data for character and line segmentation based on the bounding boxes of individual characters. Within each character's bounding box, we place a Gaussian blob centered on the box's center. The ground truth for the line segmentation would be a 2D distribution that is uniform in the x direction and is a Gaussian distribution in the y direction. The standard deviations of these Gaussian blobs are determined by the bounding boxes dimensions: in our test, σ_x, and σ_y are set to 30% of the box's width and height, respectively. We then crop the Gaussian blob using the limits of the text bounding box. These are demonstrated in Figure 4. By adjusting the spread of the Gaussian distribution relative to the character size, we encourage the model to focus on the center of the target box and reduce the overlap between adjacent blobs of the same class, thereby improving the model's ability to distinguish closely situated characters and lines, which is challenging for characters with slender profiles, e.g. "l" or "1".

3) Training loss: In our case, the target distribution is highly imbalanced towards predicting nothing. Utilizing standard loss functions such as binary cross-entropy or mean squared error resulted in the model's predictions converging rapidly to zero. To address this issue, we employ a two-term loss function that is specifically designed to handle the imbalance and to focus the learning process on the relevant regions of the output space.

The first term of our loss function is a weighted mean squared error term *wm* described in (1), which prioritizes regions containing text. This term focuses the model on classifying characters and lines effectively.

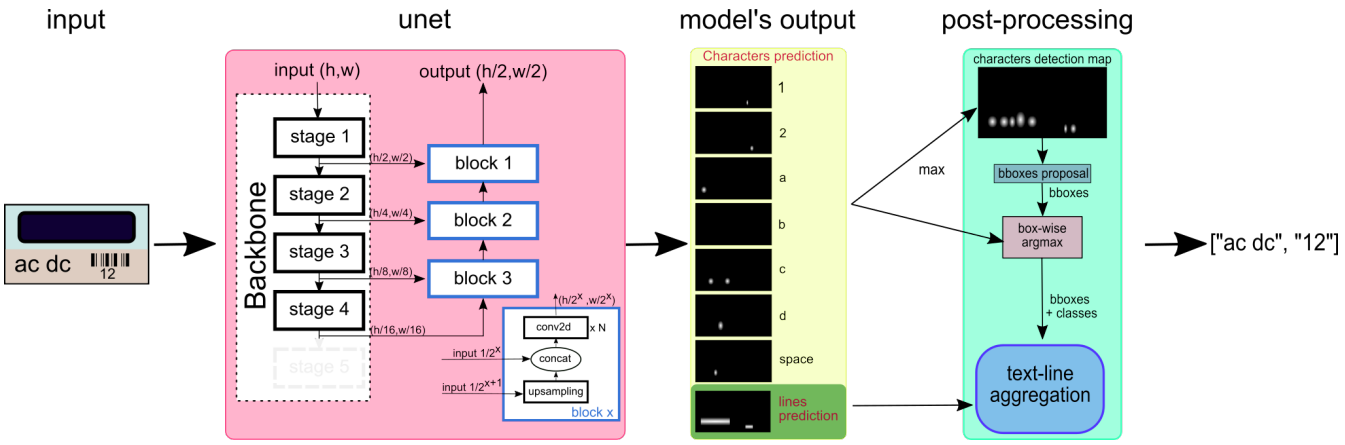

Fig. 4. The overall pipeline of our system. At its core, a U-Net model extracts two contextual layers: individual characters and entire text lines. The models not only localize characters but also classifies them. These characters are then concatenated to form the complete text string with the help of text line prediction. In this example, we construct the U-Net with only 4 feature maps from the backbone. The model is trained on the characters set $\mathbb{C} = \{$"1", "2", "a", "b", "c", "d", " " (white space)$\}$.

$$wm = \frac{1}{HW|\mathbb{C}|} \sum_{i=0}^{H} \sum_{j=0}^{W} \sum_{c=0}^{|\mathbb{C}|} \left(y^*_{ijc} - y_{ijc} \right)^2 \left(\frac{K}{|\mathbb{C}|} m_{ij} + y^*_{ijc} + \mu \right) \quad (1)$$

In this expression, H and W are the height and width of the output; y^* is the ground truth; y is the network output (that has $|\mathbb{C}| + 1$ channels); K is a coefficient (we used $K = 1.3$); μ is the average value of y^* along all dimensions. m is the mask that selects the area of interest around letters. It is a dilated version of the max value of y along the channel axis $m_{ij} = \max_{ij}{}^{k \times k} \max_c y^*_{ijc}$; k is the window size of the max operator used for dilation. We use μ, y^*, and m weights for the squared error at each pixel. They give a certain weight to three distinct areas to balance their presence in training data: pixels outside the text (most common class), channels for wrong characters in pixels inside the text area, and the correct channels at the correct pixel (the least common class).

The second term of our loss function is a regularization term, reg, which aims to penalize false positives and is defined for one image as follows:

$$reg = \frac{1}{HW|\mathbb{C}|} \sum_{c=0}^{|\mathbb{C}|} \text{ReLU} \left(\sum_{i=0}^{H} \sum_{j=0}^{W} (y_{ijc} - \hat{y}) \right)^{\gamma} \quad (2)$$

Here, y represents the model's predictions. The term y_{ijc} represents the prediction for class c in position ij, and $\hat{y}$ is the target value for the predictions, which was set at 0.3 in our test. The exponent γ is a focusing parameter that adjusts the sensitivity of the loss function. The higher the prediction deviates from the target value, the stronger the penalization. The ReLU (rectified linear unit) function ensures that the regularization term only penalizes predictions that exceed the target mean, thereby focusing on reducing false positives.

In the course of our experiments, the final loss function was constituted by an aggregation of these two terms with the regularization term scaled by a factor of α. The coefficient α was empirically set to 10^{-4}.

$$\text{Loss} = \overline{wm} + \alpha \cdot \overline{reg} \quad (3)$$

4) Postprocessing: The model's output is divided into two groups: The first $|\mathbb{C}|$ maps are character predictions, which we denote $pred_c$, and the last output predicts the text line, $pred_l$ The initial step involves the computation of a character detection response map, defined as:

$$Det\_map = \max(pred_c) \quad (4)$$

The operation retains the maximum predicted probability for each spatial location across all character classes. This response map serves as a basis for localizing characters. We can achieve character localization simply by applying a threshold to identify significant responses and subsequently extracting connected components. However, this method is not without its limitations, particularly in the detection of smaller characters such as "1" and "l", which often exhibit a lower response in the detection map. To mitigate this issue, we enhance our approach by identifying local maxima within the response map. For each local maximum, we place a fixed-size bounding box centered on its location. To further refine the results, we employ a non-maximum suppression step to eliminate redundant boxes. The value at the local maximum is utilized as a confidence score for the presence of a character within the box. The final character classification for each bounding box, denoted as $class_{box}$, is obtained by applying a region of interest pooling operation to the predicted class probabilities $pred_c$, followed by an $argmax$ operation. Let R_{box} represent the region of interest defined by the bounding box and $pred_{box}$ the aggregated prediction over the region R_{box}, the character class assigned to each box can be expressed as:

$$pred_{box} = \sum_{(x,y) \in R_{box}} pred_c(x, y) \quad (5)$$

$$class_{box} = \underset{c}{\text{argmax}} \left(pred_{box} \right) \quad (6)$$

Through this post-processing pipeline, we achieve a more robust and accurate segmentation of characters on price tags, which is particularly effective for small-sized text.

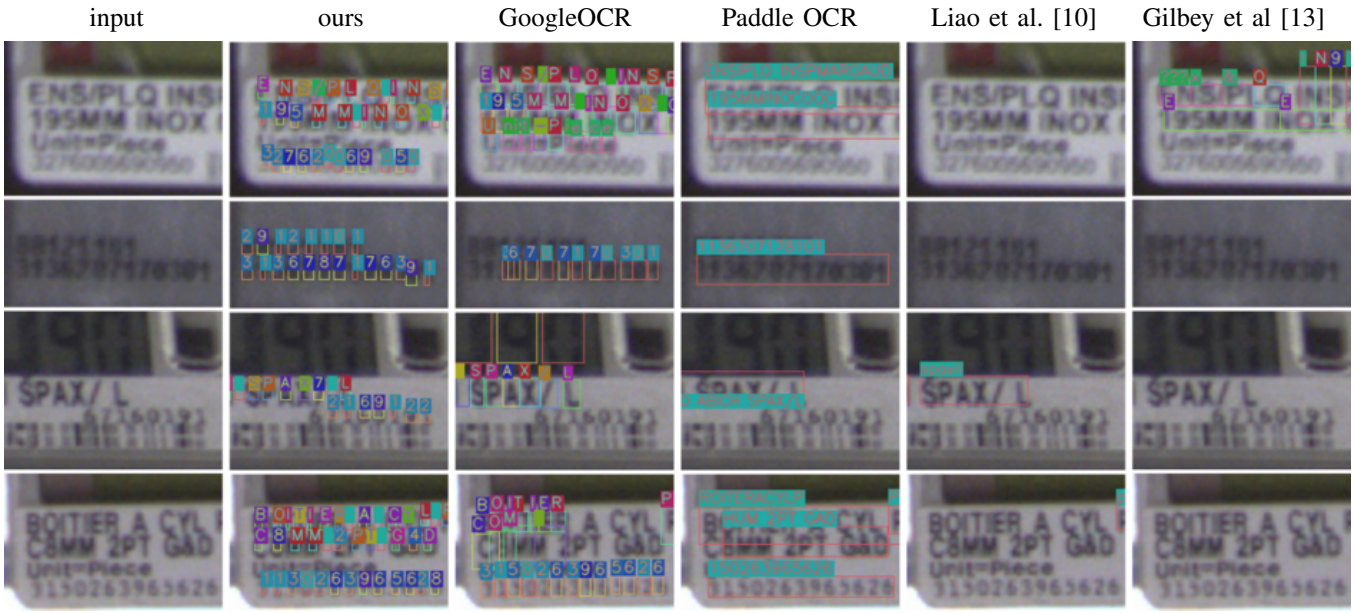

Fig. 5. Qualitative comparison of our method with other approaches. Inference was conducted on the entire cropped pricetag image, akin to the first column in Figure 3. Despite the image being captured automatically by a moving robot approximately 1m from the shelf under low-light conditions, which challenges readability, our method consistently outperforms many other off-the-shelf OCR methods. The last row is one of the few cases our method fails to detect the low-resolution barcode.

From the set of detected characters, we can form words and lines by employing a spatial search akin to [4] which has been proven to be fast and effective in [21]. However, we leverage line and white space predictions of our model to enhance the effectiveness of word and character-level grouping. Firstly, we apply a threshold to the line probability map to delineate our line predictions, which are identified as connected components within this map. Detected characters from the previous step are subsequently segmented based on the connected component that appears the most within their respective bounding boxes. Following this aggregation, the characters are sequentially arranged from left to right within each line. The incorporation of white space prediction in our model provides an additional advantage, as it inherently segments the characters into words.

IV. Experiment and Discussion

We applied our method to the automatic extraction of information from price tags captured by our robot in a retail environment. Our method targets 45 characters, comprising 10 Arabic numerals, 26 uppercase English letters, 8 special characters {“Ø”, “+”, “-”, “.”, “/”, “%”, “(”, “)”}, and the white space. We focused on three primary text fields on the price tags: the product description, the barcode in digits, and the internal code, as necessitated by our application. However, the method can be easily expanded to include any text field. To train the model, we generated a synthetic dataset of 20,000 price tags. After testing various architectures, we empirically choose a 6.46 million parameters U-Net constructed from four stages of ResNet-18 [22]. The model underwent training for 500 epochs with an incrementally enhanced augmentation pipeline. For training and testing, our model resizes the longer side of input images to 320 pixels.

Results from applying different methods to some price tag images from our robot are presented in Figure 5. Qualitatively, while Google Cloud’s text detection [1] and PaddleOCR V4 [1] accurately extract the product description field, they frequently miss the smaller internal codes and barcodes. Specifically, we observed that the detection stage of PaddleOCR V4 often fails to recognize these codes. The specialized Tesseract model by Gilbey et al. [13] for low-resolution text and Mask TextSpotter v3 by Liao et al. [10] generally under-perform in these price tags. We believe that training these models with our approach on a synthetic dataset would likely enhance their methods.

To quantitatively evaluate our detection and recognition results, we annotated at the character level a test set comprising 60 price tags, which were randomly chosen and cropped from images captured by our robot in a real store. We assessed performance using precision and recall metrics. For detection, a true positive is counted if the Intersection over Union with a ground truth box exceeds a predefined threshold. Regarding the recognition task, we first calculated precision and recall for each character class. Subsequently, we derived the mean precision and recall by averaging these values across all character classes. It is important to note that the recognition metrics were computed exclusively on the subset of characters that were correctly detected, i.e., the true positives. The final results for methods that provide character-level output are presented in Table I.

In an experimental setup employing an Intel Core i7-7700K CPU and an NVIDIA GeForce GTX 1080 Ti GPU, our method can process one image in 8 ms on average,

[1]Google Cloud Platform’s Text detection service, model released on December 05, 2023, https://cloud.google.com/vision/docs/release-notes

TABLE I
CHARACTER DETECTION AND CLASSIFICATION QUANTITATIVE RESULTS ON OUR TEST SET.

Method	Detection		Classification	
	Precision	Recall	Mean Prec.	Mean Rec.
Google OCR	**99.85**%	84.40%	88.20%	87.64%
Tesseract [2]	88.04%	2.51%	78.55%	82.41%
Gilbey et al. [13]	61.07%	2.48%	32.91%	23.61%
Ours	98.23%	**97.19**%	**89.99**%	**88.69**%

with 5 ms attributed to the neural network computation and 3 ms to the post-processing. A comparison is presented in Table II, where most of the timing data are derived from their respective papers except PaddleOCR V4 which we measured on our aforementioned hardware and test set. All of the GPUs used to obtain results for this table are comparable in terms of peak FLOPs performance.

TABLE II
COMPARISON OF MODEL INFERENCE TIME PER IMAGE, VALUES COME FROM RESPECTIVE PAPER OR AUTHORS' RELEASED MODELS.
* APPROXIMATION FROM DESCRIPTION OF MODEL.

Model	Time (ms)	GPU	Params ($\times 10^6$)
PaddleOCR V4 [1]	327.2	RTX 1080Ti	**2.6**
Mask TextSpotterV3 [10]	400.0	TITAN Xp	45.5
Deep TextSpotter [11]	111.1	K80	23.9*
FOTS [12]	45.4	TITAN Xp	35.0
Ours	**8.2**	RTX 1080Ti	6.46

V. CONCLUSION

In this study, we detail our methodology for real-time text detection and recognition in low-resolution images. Our proposed method employs a streamlined neural network architecture that is both efficient and capable of end-to-end training, augmented by a lightweight post-processing step to facilitate simultaneous text detection and recognition. We have demonstrated that a synthetic dataset, with minimal manual annotation, can yield a model that performs effectively on our target application. Looking forward, we aim to refine our model to handle text strings of varying sizes and configurations under more complex conditions. Although our current focus is on low-resolution images of price tags, the methodology is designed to be adaptable to a broader range of applications, for example enhancing document digitization processes without the need for high-quality scanners, with only minor adjustments.

REFERENCES

[1] Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang *et al.*, "Pp-ocr: A practical ultra lightweight ocr system," *arXiv preprint arXiv:2009.09941*, 2020.

[2] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, 2007, pp. 629–633.

[3] J. Huang, G. Pang, R. Kovvuri, M. Toh, K. J. Liang, P. Krishnan, X. Yin, and T. Hassner, "A multiplexed network for end-to-end, multilingual ocr," in *Proc. IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4547–4557.

[4] L. D. Huynh, Y. Xu, and T. Géraud, "Morphology-based hierarchical representation with application to text segmentation in natural images," in *Proc. 23st International Conference on Pattern Recognition (ICPR)*. IEEE Computer Society, 2016.

[5] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An efficient and accurate scene text detector."

[6] S. Qin, A. Bissacco, M. Raptis, Y. Fujii, and Y. Xiao, "Towards unconstrained end-to-end text spotting," in *Proc. IEEE/CVF international conference on computer vision*, 2019, pp. 4703–4713.

[7] S. Zhao, X. Wang, L. Zhu, and Y. Yang, "Clip4str: A simple baseline for scene text recognition with pre-trained vision-language model," *arXiv preprint arXiv:2305.14014*, 2023.

[8] P. Wang, C. Da, and C. Yao, "Multi-granularity prediction for scene text recognition," in *European Conference on Computer Vision*. Springer, 2022, pp. 339–355.

[9] D. Bautista and R. Atienza, "Scene text recognition with permuted autoregressive sequence models," in *European Conference on Computer Vision*. Springer, 2022, pp. 178–196.

[10] M. Liao, G. Pang, J. Huang, T. Hassner, and X. Bai, "Mask textspotter v3: Segmentation proposal network for robust scene text spotting," in *Proc. 16th European Conference on Computer Vision (ECCV)*. Springer, 2020.

[11] M. Busta, L. Neumann, and J. Matas, "Deep TextSpotter: An end-to-end trainable scene text localization and recognition framework," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2223–2231.

[12] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "Fots: Fast oriented text spotting with a unified network," in *Proc. IEEE conference on computer vision and pattern recognition*, 2018.

[13] J. D. Gilbey and C.-B. Schönlieb, "An end-to-end optical character recognition approach for ultra-low-resolution printed text images," *arXiv preprint arXiv:2105.04515*, 2021.

[14] S. Park, S. Shin, B. Lee, J. Lee, J. Surh, M. Seo, and H. Lee, "Cord: a consolidated receipt dataset for post-ocr parsing," in *Workshop on Document Intelligence at NeurIPS 2019*, 2019.

[15] L. Gao, Y. Huang, H. Déjean, J.-L. Meunier, Q. Yan, Y. Fang, F. Kleber, and E. Lang, "Icdar 2019 competition on table detection and recognition (ctdar)," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 1510–1515.

[16] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "Coco-text: Dataset and benchmark for text detection and recognition in natural images," *arXiv preprint arXiv:1601.07140*, 2016.

[17] A. Singh, G. Pang, M. Toh, J. Huang, W. Galuba, and T. Hassner, "Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text," in *Proc. IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 8802–8812.

[18] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proc. IEEE conference on computer vision and pattern recognition*, 2016, pp. 2315–2324.

[19] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," *NIPS Deep Learning Workshop*, 2014.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, pp. 234–241.

[21] L. D. Huynh, Y. Xu, and T. Géraud, "Morphological hierarchical image decomposition based on laplacian 0-crossings," in *Proc. 13th International Symposium on Mathematical Morphology (ISMM)*, vol. 10225. Springer, 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.