Supplementary Material for *Hierarchical Text Spotter for Joint Text Spotting and Layout Analysis*

Shangbang Long, Siyang Qin, Yasuhisa Fujii, Alessandro Bissacco, Michalis Raptis
Google Research
{longshangbang,qinb,yasuhisaf,bissacco,mraptis}@google.com

A. Line Cropping and Projecting Characters and Word Boxes Back to Image Space

A.1 Preliminaries: Bezier curve representation

Liu et al. [5] proposes to use a pair of Bezier curves[1] to represent a polygonal text detection box, denoted by:

$$top(t) = \sum_{i=0}^{n} b_{top,i} B_{i,n}(t), 0 \leq t \leq 1 \quad (1)$$

for the top edge of text, and:

$$bottom(t) = \sum_{i=0}^{n} b_{bottom,i} B_{i,n}(t), 0 \leq t \leq 1 \quad (2)$$

for the bottom edge. In this formulation, both top and bottom edges are represented as parametric Bezier curves of order n, with parameter t, control points $b_{top,i}$ and $b_{bottom,i}$, and Bernstein basis polynomials $B_{i,n}(t)$ defined as:

$$B_{i,n}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i}, i = 0, ..., n \quad (3)$$

where $\begin{pmatrix} n \\ i \end{pmatrix}$ are the binomial coefficients. Specifically, $top(0)$, $top(1)$, $bottom(0)$, and $bottom(1)$ are the 'top-left', 'top-right', 'bottom-left', and 'bottom-right' of a text bounding polygon, as shown in Fig. 1.

A.2 BezierAlign for cropping

In order to crop from the input image, Liu et al. [5] establish a correspondence from the coordinate space of text crops to the space of the input image. Denote the shape of a text crop as $h_o \times w_o$. For the $j-$th pixel with coordinates $p_{crop,j} = (x_j, y_j)$ in the text crop, we first compute $t_j = \frac{x_j}{w_o}$ to determine the t parameter as in Eq. (1) and Eq. (2). Then, the supporting points on top and bottom edges

[1] https://en.wikipedia.org/wiki/Bezier_curve

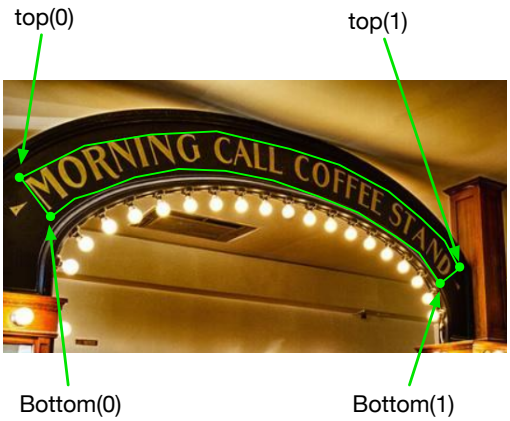

Figure 1. Illustration of Bezier curve polygons.

are computed as $tp_j = top(t_j)$ and $bp_j = bottom(t_j)$. Note that both tp_j and bp_j are in the coordinate space of the input image. Then, the image space coordinates for (x_j, y_j), denoted as p_{image} can be obtained by linearly interpolating between tp_j and bp_j:

$$p_{image,j} = tp_j \times (1 - \frac{y_j}{h_o}) + bp_j \times \frac{y_j}{h_o} \quad (4)$$

In this way, we establish a mapping from text crop space coordinates to full image space coordinates:

$$f_{Bezier} : p_{crop,j} \longrightarrow p_{image,j} \quad (5)$$

We can then easily apply bilinear interpolation to calculate each pixel for the text crop.

A.3 Postprocessing: Projecting Characters and Word Boxes Back to Image Space

Our L2C2W recognizer produces bounding boxes for each recognized characters. After text line is split into words, word bounding boxes are then obtained by merging character bounding boxes. We project each vertex of word and character bounding boxes from line crop spaces to full

image spaces using Eq. 5, and obtain the coordinates in the original input image space.

B. Label Maps

Our recognizer learns to recognize a total number of 708 character classes, excluding special tokens for *Start-of-Sentence*, *End-of-Sentence* and *Padding*. The character list includes digits, punctuation symbols, Latin letters and their variants. We also attach the complete list of character classes to this zip file under the name of *label_maps.txt*.

C. Samples of Synthetic Training Images

Our recognizer is trained on a mixture of real image data and synthetic data. We show and compare text crops from different synthetic datasets in Fig. 3. We use a similar method to [3] to generate our internal synthetic data. Our synthetic data is visually similar to the other two except that ours are mostly text lines containing multiple words sampled from natural text corpus. In contrary to ours, Synth90K [3] is generated by using a fixed word vocabulary list containing about $90K$ words, and each crop only has one word. SynthText [2] is generated from a corpus of natural text[2]. While most previous works train text recognizers on word crops extracted from SynthText, we notice that SynthText actually has line-level labels. Nonetheless, lines in SynthText are much shorter than our synthetic lines, as shown in Fig. 2. To train line recognizers, line-level synthetic data is indispensable. Among the 3 datasets, SynthText and our internal line datasets provide character level bounding boxes. The training of the character localization head is enabled by these labels.

D. Model output samples

In the submission, we are only able to demonstrate the model output on one image due to space limit. Here, we show the outputs of our model on multiple images from HierText [6], ICDAR 2015 [4], and Total-Text [1], in Fig. 4, 5, 6, 7, 8, 9, 10. In addition to character bounding boxes and their grouping into words, lines, and paragraphs, we also visualize the character classes.

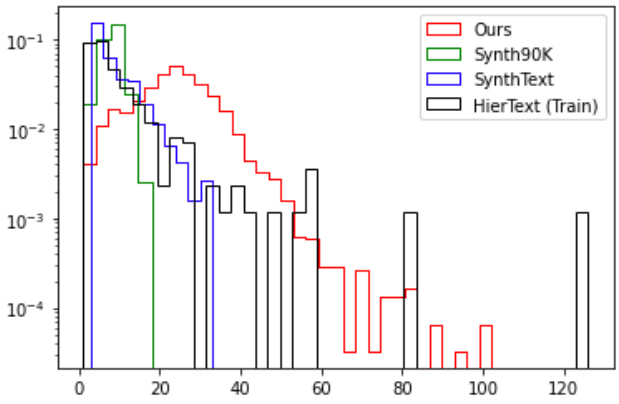

Figure 2. Distribution (Y-axis is log proportion) of numbers of characters in text crops (X-axis).

References

[1] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 935–942. IEEE, 2017. 2

[2] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2315–2324, 2016. 2, 3

[3] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014. 2, 3

[4] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015. 2

[5] Yuliang Liu, Chunhua Shen, Lianwen Jin, Tong He, Peng Chen, Chongyu Liu, and Hao Chen. Abcnet v2: Adaptive bezier-curve network for real-time end-to-end text spotting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8048–8064, 2021. 1

[6] Shangbang Long, Siyang Qin, Dmitry Panteleev, Alessandro Bissacco, Yasuhisa Fujii, and Michalis Raptis. Towards end-to-end unified scene text detection and layout analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1049–1059, 2022. 2

[2] http://qwone.com/~jason/20Newsgroups/

Figure 3. Sample images from 3 synthetic text datasets. **Top**: SynthText [2]. **Middle**: Synth90K [3]. **Bottom**: Our internal synthetic line dataset.

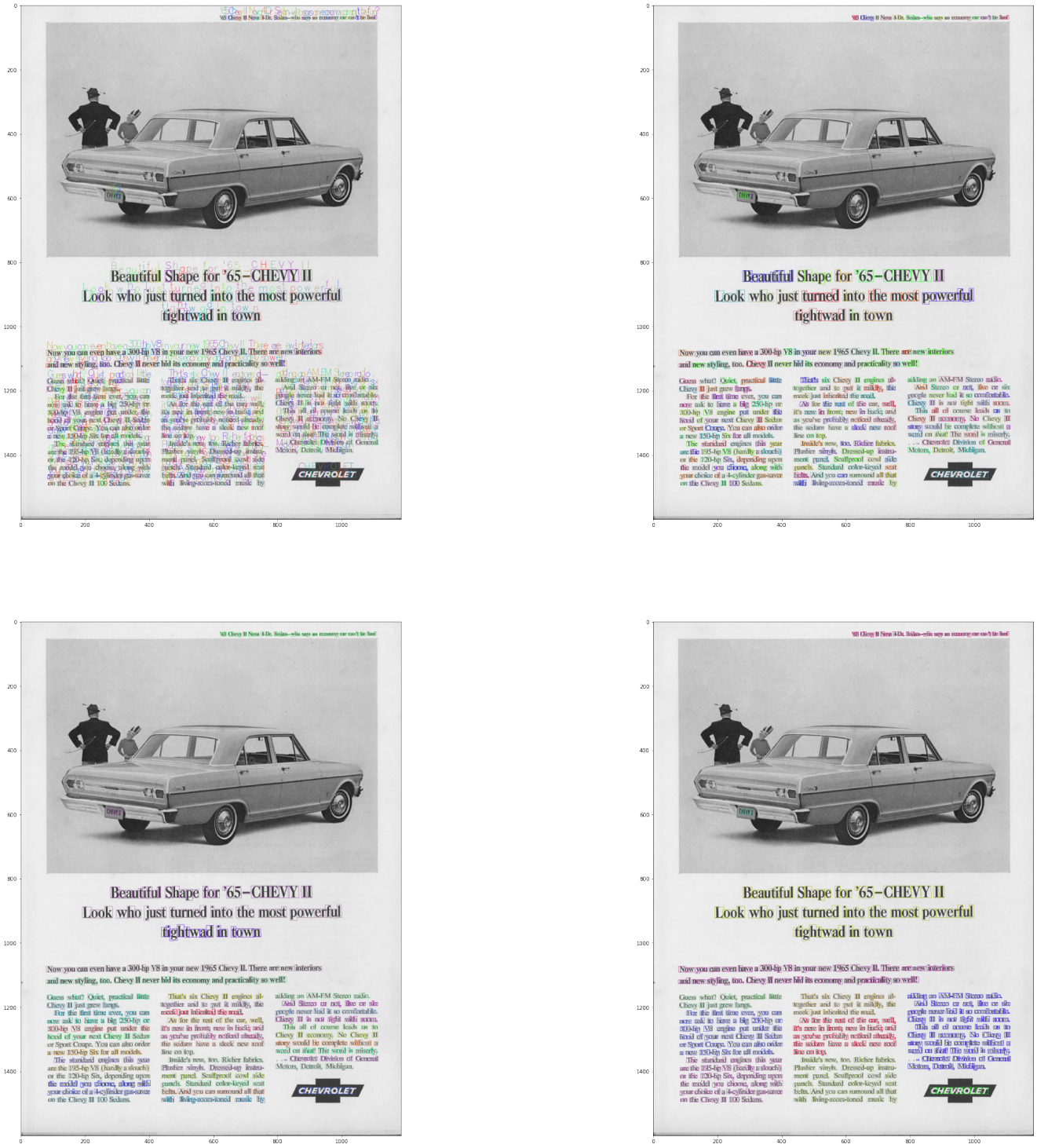

Figure 4. Sample of model outputs. Image is from HierText Val set. **Top left**: Character bounding boxes and transcriptions; **Top right**: Character bounding boxes grouped into words; **Bottom left**: Character bounding boxes grouped into lines; **Bottom right**: Character bounding boxes grouped into paragraphs. Images are converted to gray for better visualization. Zoom-in for clearer views.

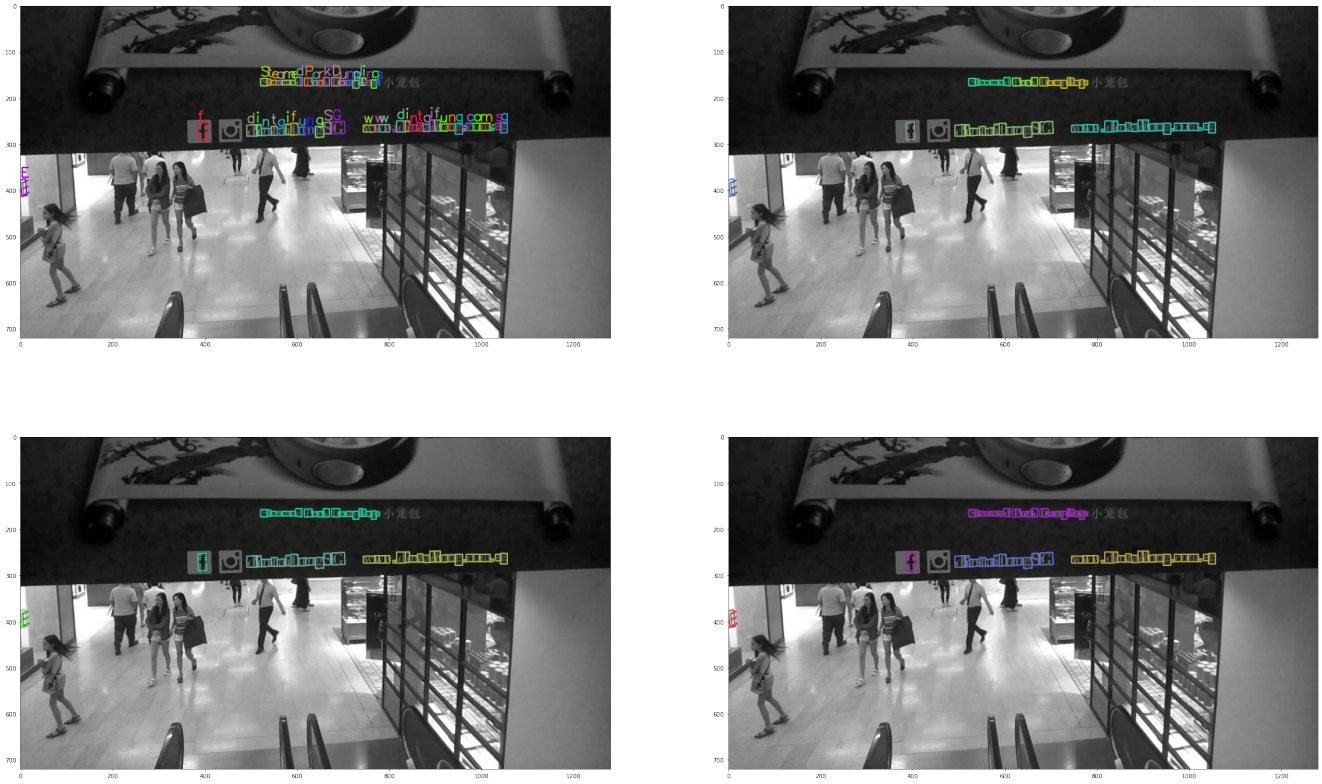

Figure 5. Sample of model outputs. Image is from ICDAR 2015 test set. **Top left**: Character bounding boxes and transcriptions; **Top right**: Character bounding boxes grouped into words; **Bottom left**: Character bounding boxes grouped into lines; **Bottom right**: Character bounding boxes grouped into paragraphs. Images are converted to gray for better visualization. Zoom-in for clearer views.

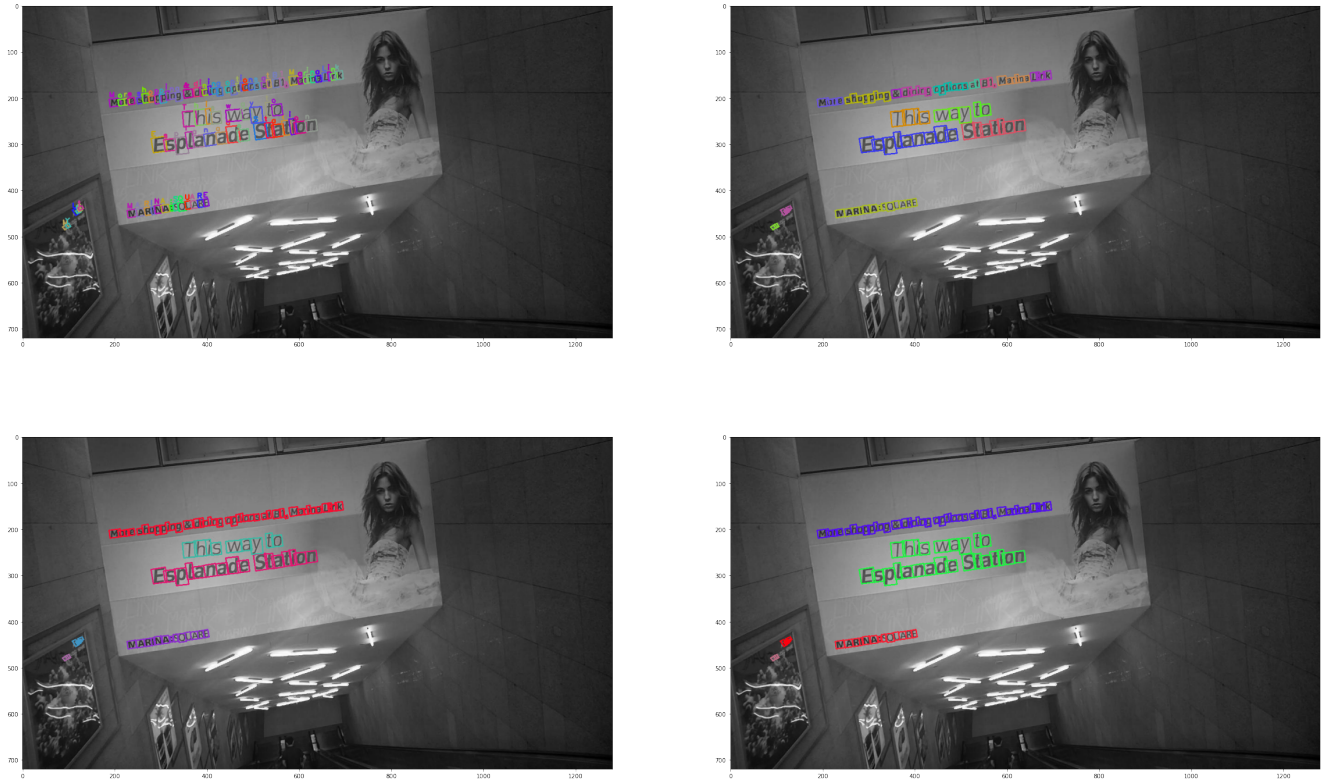

Figure 6. Sample of model outputs. Image is from ICDAR 2015 test set. **Top left**: Character bounding boxes and transcriptions; **Top right**: Character bounding boxes grouped into words; **Bottom left**: Character bounding boxes grouped into lines; **Bottom right**: Character bounding boxes grouped into paragraphs. Images are converted to gray for better visualization. Zoom-in for clearer views.

Figure 7. Sample of model outputs. Image is from Total-Text test set. **Top left**: Character bounding boxes and transcriptions; **Top right**: Character bounding boxes grouped into words; **Bottom left**: Character bounding boxes grouped into lines; **Bottom right**: Character bounding boxes grouped into paragraphs. Images are converted to gray for better visualization. Zoom-in for clearer views.

Figure 8. Sample of model outputs. Image is from Total-Text test set. **Top left**: Character bounding boxes and transcriptions; **Top right**: Character bounding boxes grouped into words; **Bottom left**: Character bounding boxes grouped into lines; **Bottom right**: Character bounding boxes grouped into paragraphs. Images are converted to gray for better visualization. Zoom-in for clearer views.

Figure 9. Sample of model outputs. Image is from Total-Text test set. **Top left**: Character bounding boxes and transcriptions; **Top right**: Character bounding boxes grouped into words; **Bottom left**: Character bounding boxes grouped into lines; **Bottom right**: Character bounding boxes grouped into paragraphs. Images are converted to gray for better visualization. Zoom-in for clearer views.

Figure 10. Sample of model outputs. Image is from Total-Text test set. **Top left**: Character bounding boxes and transcriptions; **Top right**: Character bounding boxes grouped into words; **Bottom left**: Character bounding boxes grouped into lines; **Bottom right**: Character bounding boxes grouped into paragraphs. Images are converted to gray for better visualization. Zoom-in for clearer views. Note that, this image contains some variations of Latin characters, for example 'DÖNE' in the bottom right of the image. Although our recognizer is able to recognize them, the visualization function we used (cv2.putText) fails to print them accordingly, and thus they are replaced with '??'.