International Journal on Document Analysis and Recognition (IJDAR)
https://doi.org/10.1007/s10032-019-00334-z

ORIGINAL PAPER

Total-Text: toward orientation robustness in scene text detection

Chee-Kheng Ch'ng[1] · **Chee Seng Chan**[1] · **Cheng-Lin Liu**[2]

Received: 30 October 2018 / Revised: 3 April 2019 / Accepted: 9 July 2019

Abstract

At present, text orientation is not diverse enough in the existing scene text datasets. Specifically, curve-orientated text is largely out-numbered by horizontal and multi-oriented text, hence, it has received minimal attention from the community so far. Motivated by this phenomenon, we collected a new scene text dataset, *Total-Text*, which emphasized on text orientations diversity. It is the first relatively large scale scene text dataset that features three different text orientations: horizontal, multi-oriented, and curve-oriented. In addition, we also study several other important elements such as the practicality and quality of ground truth, evaluation protocol, and the annotation process. We believe that these elements are as important as the images and ground truth to facilitate a new research direction. Secondly, we propose a new scene text detection model as the baseline for Total-Text, namely *Polygon-Faster-RCNN*, and demonstrated its ability to detect text of all orientations. Images of Total-Text and its annotation are available at https://github.com/cs-chan/Total-Text-Dataset.

Keywords Curved text · Scene text detection

1 Introduction

Scene text detection is one of the most active computer vision topics due to the growing demands of applications such as multimedia retrieval, industrial automation, assistive devices for the visually impaired. Given a natural scene image, the goal of text detection is to determine the existence of text, and return the location if it is present.

Well-known public datasets such as ICDAR'03, '11, '13 [1] (ICDARs from here on), and MSRA-TD500 [2] have played significant roles in initiating the momentum of scene text-related research. For instance, almost all texts in the ICDARs are in horizontal orientation [3], and it has inspired researchers to incorporate the horizontal assumption [4–8] in solving the scene text detection problem. In 2012, Yao et al. [2] introduced a new scene text dataset, namely MSRA-TD500, that challenged the community with text arranged in multiple orientations. The popularity of it in turn defined the convention of 'multi-oriented' text. However, a closer look into the MSRA-TD500 dataset revealed that most if not all the text instances are still arranged in a straight line manner as to ICDARs, as detailed in Sect. 3. From then on, multiple datasets [9–11] were launched, with several noticeable transitions: (i) multi-oriented text has become a norm, (ii) scene texts are no longer centered in the image, (iii) ever-increasing dataset scale. However, curve-oriented text (curved text from here onwards), despite its commonness (depicted in Fig. 1), is missing from the context of study. Before the recent publications of Total-Text [12] and CTW1500 [13], CUTE80 [14] was the only available scene text dataset with curved text. However, its scale is too small with only 80 images and it has very minimal scene diversity.

Without the motivation of a proper dataset, effort in solving the curved text detection problem is rarely seen. This phenomenon brings us to our primary contribution of this paper: **Total-Text**, a scene text dataset collected with curved text as priority, filling up the gap in scene text datasets in terms of text orientations. It has 1555 scene images, 11,459 annotated words with 3 different text orientations including horizontal, multi-oriented, and curve-oriented. Ground truths made available to the public include: spatial location, tran-

✉ Chee Seng Chan
cs.chan@um.edu.my

Chee-Kheng Ch'ng
chngcheekheng@siswa.um.edu.my

Cheng-Lin Liu
liucl@nlpr.ia.ac.cn

[1] Faculty of Computer Science and Information Technology, Center of Image and Signal Processing, University of Malaya, 50603 Kuala Lumpur, Malaysia

[2] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Published online: 01 August 2019

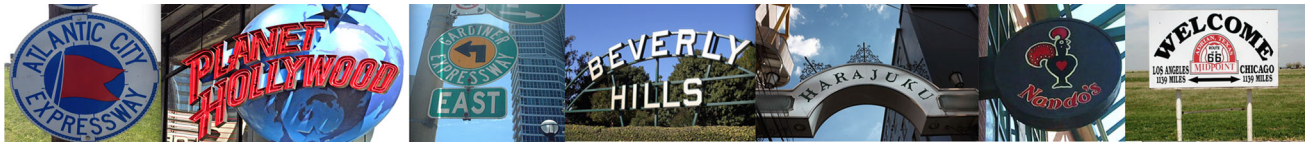


Fig. 1 Curved text is commonly seen in real-world scenery

(a) (b) (c)

	x-axis	y-axis	description	orientation
1	[166,196,228,259,288,300,273,244,215,186]	[214,183,164,146,135,171,182,197,219,242]	'BORDER'	'c'
2	[299,325,351,377,402,400,376,352,329,304]	[124,119,113,111,110,153,154,160,164,170]	'ROADS'	'c'
3	[407,456,505,553,601,583,541,495,452,408]	[115,121,136,163,202,232,196,173,157,156]	'ORGANISATION'	'c'
4	[228,279,328,380,433,432,381,330,277,225]	[264,263,263,263,262,306,315,311,314,311]	'SIMPLICITY'	'm'
5	[441,446,467,481,478,480,478,460,444,444]	[263,264,264,265,263,307,309,310,310,309]	'IS'	'm'
⋮				
14	[402,412,424,435,446,448,437,425,413,401]	[441,440,441,440,439,467,468,470,471,472]	'RCC'	'm'
15	[500,512,524,536,546,544,534,522,510,498]	[427,427,426,425,425,471,473,473,474,475]	'16'	'm'
16	[561,574,587,600,612,611,600,587,575,563]	[425,425,423,423,424,470,468,468,468,467]	'TF'	'm'

(d)

Fig. 2 Annotation details of Total-Text. **a** Image of Total-Text, **b** text region binary mask, **c** character-level binary mask, **d** fixed length polygon vertices, transcription (case-sensitive), and orientation annotations ('c': curved, 'm': multi-oriented)

scription, and pixel level for text detection, recognition and segmentation task (as shown in Fig. 2).

The secondary contribution of this paper is the proposal of a new scene text detection model, **Polygon-Faster-RCNN** (Poly-FRCNN from here on). It is a refined Faster-RCNN [15], to regress polygon instead of box parameters. It is capable of detecting text of all orientations, and binding them in a precise manner. Our proposed model achieves 0.85, 0.72, and 0.7 in terms of F-measure on ICDAR2013, ICDAR2015, and Total-Text; demonstrating its effectiveness across datasets with different attributes.

A preliminary version of the proposed dataset was presented earlier in [12]. This paper adds to the initial version in significant ways as follows:

1.1 Improved ground truth

Firstly, we address the inconsistency of the polygon annotation employed in our preliminary version [12] in Sect. 3.1. The polygon annotation of our previous work was collected with the mindset of covering the text region as tight as possible. As a result, the number of polygon vertices varies from one text instance to another. This poses a practical problem for detection frameworks such as Faster-RCNN, SSD, and YOLO (all of them have inspired many scene text detection works), which require a fixed number of vertices in the regression target. Hence, we have refined Total-Text's annotation with a series of controls, providing the community with a higher quality and less biased ground truth to work with.

1.2 Optimized evaluation protocol for Total-Text

Secondly, our experiments in Sect. 4 show that the current recommended thresholds in DetEval [16] are not optimized with the inclusion of curved text. A series of experiments were conducted to determine a new set of threshold for a fairer evaluation.

1.3 Scene text detection annotation tool

Thirdly, it is known that ground truth annotation is the biggest bottleneck when scaling up a dataset. Karatzas et al. [17] introduced an online annotation platform that stress on quality control and database management. However, the mundane and painstaking annotation task alone has much room for improvements. Hence, we introduce Total-Text-Tool (*T3* from here on) in Sect. 5, an aided annotation framework that is capable of reducing annotation time yet attaining quality ground truth.

1.4 Cross dataset experiment

Next, we have conducted an investigation to determine the impact of Total-Text on scene text detection model. The experiment results in Sect. 6.5 show that the model trained on Total-Text demonstrates good generalization across other scene text datasets.

1.5 State-of-the-art analysis

Last but not least, numerous works started to address the curved text detection problem since the emergence of Total-Text [12]. We discuss about these works in length and provide a complete table for future references in Sect. 6.6.

The refined ground truth, *T*3, and all the baseline models are available at https://github.com/cschan/Total-Text-Dataset for future reproduction and benchmarking.

2 Related works

This section will discuss closely related works, specifically scene text datasets and text detection system. For completeness, readers are recommended to read [3].

2.1 Scene text datasets

2.1.1 ICDAR2003-ICDAR2015

ICDAR2003 started with 509 camera taken scene text images. All the scene text instances in the dataset appear in horizontal orientation. In ICDAR2011, the total number of images were reduced to 484 to eliminate duplication in the previous version. ICDAR2013 [1] further trimmed down the 2011 version to 462 images in total. Improvement was done to increase its text categories and tasks. In 2015, ICDAR launched a new challenge, named as the 'Incidental Scene Text' [9] (also known as the ICDAR2015), which has 1670 images captured using wearable devices. It is more challenging than previous datasets as it has included text with arbitrary orientation and most of them are out of focus. In addition, ICDAR2015 is the first known dataset to utilize quadrilateral as its ground truth format. This challenges scene text algorithms to be more precise in detecting text.

2.1.2 MSRA-TD500

Yao et al. [2] introduced MSRA-TD500 in 2012 to address the lack of arbitrary orientated text in scene text datasets. It has 300 training and 200 testing images; annotated with rotated bounding box.

2.1.3 USTB-SV1K

Yin et al. [18] constructed the USTB-SV1K dataset, which has 1000 images collected from the street of six USA cities. Again it features mostly multi-oriented text, annotated with rotated bounding box.

2.1.4 COCO-text

Veit et al. [10] released COCO-text in the early 2016. It is the largest scene text dataset to date with 63,686 images and 173,589 labeled text regions. It consists of mainly horizontal and multi-oriented text, and a small amount of curved text. However, it uses the axis-aligned rectangle as ground truth, which is more suitable for linearly aligned text.

2.1.5 MLT

Nayef et al. [19] introduced the MLT dataset, which is one of the latest multi script datasets collected for scene text detection, recognition, and script identification task. It consists of 18,000 images for both training and validation, featuring 9 languages with 6 different scripts. The main motivation of this dataset is to challenge the community to design or train a scene text detector that is robust to different scripts that exist in today's world.

2.1.6 CTW-12k

Shi et al. [20] introduced CTW-12k in the ICDAR2017 Robust Reading challenge, RCTW-17. Its collection was motivated by the lack of Chinese scripts in existing datasets. It has more than 12,000 images, featuring both Chinese and English languages.

2.1.7 MTWI

The MTWI dataset introduced by He et al. [21] also includes both Chinese and English languages in their dataset, which was collected for the ICPR2018 Contest on Robust Reading competition. It is one of the largest multi-lingual datasets to date with 20,000 images.

2.1.8 SynthText

Gupta et al. [11] released SynthText in 2016. It is a large-scale synthetically generated scene text dataset with 800,000 scene text images. Its annotation consists of word-level and character-level axis-aligned bounding box, along with its transcription. It was motivated by the fact that existing scene text datasets are generally small in scale. The downside of it is that it uses axis-aligned bounding box like COCO-text, which does not fit multi-oriented text tightly.

2.1.9 CUTE80

Risnumawan et al. [14] introduced the first scene text dataset that highlights curved text, namely CUTE80. Unfortunately, it has only 80 images and limited sceneries.

2.1.10 CTW1500

CTW1500 collected by Liu et al. [13] is in principle the closest dataset to Total-Text. Both datasets are similar in terms of scale, polygon bounding region annotation, and was collected with at least one curved text per image. Despite having similar motivation, both datasets are different in a number of ways. First of all, Total-Text considers only Latin scripts as text instances, while CTW1500 includes both the Latin and Chinese scripts in their annotation. Such difference leads to the second dissimilarity of both datasets, which is the granularity of annotation. Total-Text's text instances were annotated at word level, while CTW1500's were annotated at line level. The third difference is the polygon ground truth between them. CTW1500 annotation was fixed to have fourteen vertices each while Total-Text (initial version) has a range from four to twelve vertices. Lastly, CTW15-00 was annotated with only the spatial location to facilitate the text detection task while Total-Text's annotation consists of spatial location, transcription, orientation, and pixel-level mask for text detection, recognition, and segmentation tasks.

2.2 Scene text detection

2.2.1 Scene text inspired handcrafted feature era

Scene text detection has seen significant progress after the seminal work by Epshtein et al. [22] and Neumann and Matas [23]. Epshtein's proposal, stroke width transform (SWT) was based on the observation that characters in a text have similar stroke width. In the latter, maximally stable extremal regions (MSER) were exploited to extract character components. They used geometrical properties of the components and a classifier to form text candidates. Empirically, both approaches represent characters better than classic feature extractors like color, edge, texture and etc.

2.2.2 The emergence of CNN

Similar to many other computer vision tasks, the incorporation of convolutional neural network (CNN) in localizing text is a very active research area at the moment. For instance, Huang et al. [7] trained a character classifier to examine components generated by MSER to improve the robustness of the feature extraction process. Alongside this work, [24–26] also trained a CNN to classify text components from non-text. This line of work has demonstrated the high discriminative power of CNN as a feature extractor.

2.2.3 Segmentation-based scene text detection

Interestingly, Zhang et al. [27] argued that leveraging on CNN as a character detector has restricted the CNN's potential due to the local nature of characters. Zhang et al. trained two fully convolutional networks (FCN) [28]: (i) a Text-Block FCN that considers both of the local and global contextual info at the same time to identify text regions in an image, (ii) a character-centroid FCN to eliminate the false text line candidates. He et al. [29] trained a FCN to infer text line candidates. By cascading a text region and a text line supervised FCN, cascaded convolution text network (CCTN) achieved generalization in terms of text orientations. Tang et al. [30] proposed another cascaded convolution networks, demonstrated the robustness of segmentation-based scene text detection.

2.2.4 Proposal-based scene text detection

R-CNN [31], Fast-RCNN [32] to Faster-RCNN [15] are a series of groundbreaking works for object detection. The key aspects that contributed to the success of Faster-RCNN are region proposal network (RPN), anchor boxes, and the end-to-end training scheme from feature extraction to region proposal and finally the predictor. Inspired by them, R^2CNN [33] integrated a new regression head to the original Faster-RCNN framework to learn inclined box coordinates. Ma et al. [34] argued that axis-aligned anchor boxes are not the best fit for scene text that appear in wild orientations. They trained their RPN to propose rotational bounding box, which is essentially bounding box with an extra bit for angle information.

2.2.5 Single network scene text detection

SSD [35] and YOLO [36] on the other hand have no proposal module in their design. It is capable of running at 59 and 122 FPS, respectively, the fastest object detection framework we have today. Works like [37–40] have built their algorithm based on SSD. He et al. [41] demonstrated the potential edge of direct regression over offset prediction with a single fully convolutional network.

2.2.6 Curved text detection

Risnumawan et al. [14] used handcrafted features to identify character candidates, then leveraged on the eclipse growing algorithm to look for potential grouping candidates in the feature space. Their growing algorithm is unique in such a way that it has no horizontal orientation assumption that was conventional at the time. Such feature has proven its effectiveness against text of all orientations with the cost of slow running time (15.8 s per image). CTD [13] is the first modern scene text detector that was designed with curved text in consideration. It was built on top of the Faster-RCNN architecture, trained to regress 30 values ($x_{\min}$, $y_{\min}$ and fourteen pairs of $w$idth and $h$eight) that will form a polygon-shaped output with fourteen vertices. Instead of regressing the polygon vertices directly, the work followed the convention of Faster-RCNN [15] and DMPNet [37], agreed that relative information (width and height) are much easier targets to be optimized. The study also showed that LSTM can be used to model the relationship between each w and h to refine the polygon-shaped output. More latest scene text detection works can be found in Sect. 6.6.

3 The Total-Text dataset

3.1 Dataset attributes

3.1.1 Curved text is an overlooked problem

The effort of collecting Total-Text is motivated by the lack of curved text in existing scene text datasets. As depicted in Fig. 1, curved text can be easily found in real life sceneries such as: business logos, signs, entrances. Surprisingly, such data has close to zero existence in the current datasets [1,2,9]. The most popular scene text dataset over the decade, ICDARs have only horizontal text [3]. Consequently, vast majority of algorithms assumed text linearity to tackle the problem effectively. As a result of overwhelming attentions, performances of text detections in ICDARs are saturated at quite a high point (0.9 in terms of F-measure). Meanwhile, multi-oriented text also received a certain amount of attentions from the scene text community. MSRA-TD500 is the well known dataset that introduced this challenge to the field; algorithms like [18,27] were designed to cater multi-oriented text. To the best of our knowledge, scene text detection algorithms designed with curve orientation [14] in consideration is relatively unpopular. We believe that the lack of such dataset is the reason why the community has overlooked it. Hence, we propose Total-Text with 4,907 curved text out of 11,459 text instances, hoping to spur an interest in the community to address curved text detection and recognition problem.

3.1.2 Curved text observation

Geometrically speaking, a straight line has no angle variation along the line, and thus can be described as a linear function, $y = mx + c$. A curved line is not a straight line. It is free of angle variation restriction throughout the line. Shifting to the scene text perspective, it is observed that horizontal oriented text or word is a series of characters that can be connected by a straight line; their bottom alignment in particular for most cases. At the same time, multi-oriented text, in scene text convention, can also be connected by a straight line, given an offset with respect to a horizontal line. Meanwhile, characters in a curved word do not have an unified angle offset, in which deemed to fit a polynomial line in text level (see Fig. 3 for examples). In our dataset collection, we found that curved text in natural images could vary from slightly curved to extremely curved. Also, it is not surprising to find that most of them are in the shape of a symmetric arc due to the symmetrical preferences in human vision [42].

3.1.3 Detection ground truth annotation

Text instances in Total-Text were annotated at word-level granularity. Adopted from the COCO-text dataset, word-level text instances are *an uninterrupted sequence of characters separated by a space*. Unlike generic object, text detection demands tighter bounding region. For instance, ICDAR2015 is the first ever scene text detection dataset that employs quadrilateral ground truth format. The main motive behind that is to have a ground truth region that is capable of binding text region tightly. Total-Text uses polygon bounding region for the same reason. Figure 4 illustrates the apparent benefits of polygon bounding region over axis-aligned box. More details of the polygon format employed in the proposed dataset are detailed in Sect. 3.1 (**Regulated polygon ground truth**). Similar to ICDAR2015, only Latin scripts were annotated with proper transcriptions; other languages, digital watermarks and unreadable text were labeled as *do not care* in the ground truth (marked as '#'). *Do not care* area picked up by detection algorithms should be

Fig. 3 (Top) examples from ICDAR2013, ICDAR2015 and MSRA-TD500 datasets. (Bottom) examples from Total-Text, curved text with different curvatures, slightly curved to extremely curved text

Fig. 4 Comparison between axis-aligned rectangular bounding box (red color) and the proposed polygon bounding region (green color) in Total-Text. It shows that polygon appears to be the better candidate for ground truth (color figure online)

removed before evaluation. We recommend Eq. 1 for filtering:

$$\text{Area}_\text{o} = \frac{\text{Area}_\text{g} \cap \text{Area}_\text{d}}{\text{Area}_\text{d}} \tag{1}$$

where detection candidates with Area_o higher than 0.5 should be removed (Area_g is the ground truth area and Area_d is the detection area).

Besides, text region binary mask (Fig. 2b) is provided as well to handle the recent trend in scene text detection research that cast detection as a segmentation problem [27,29,43].

3.1.4 Recognition ground truth annotation

Ground truth for the word recognition challenge, the transcriptions (case-sensitive) of every text instance, are provided as well (Forth column in Fig. 2d).

3.1.5 Segmentation ground truth annotation

The finest level of Total-Text's annotation is the character-level pixel binary mask (Fig. 2c). Inspired by ICDAR2013, such ground truth is provided to cater segmentation-based challenges. Its annotation process is the most time-consuming process of all annotations. In order to ease the effort in annotating such ground truth, multiple preprocessing schemes are used in the workflow. The entire workflow can be described as follows:

(i) From the original image, crop an image patch that contains text instance based on the annotated detection ground truth (Fig. 5a).
(ii) Prompt annotator to adjust the threshold for all three color channels.[1] This is based on the observation that

[1] This is achieved by 'colorThreshold' function in MATLAB.

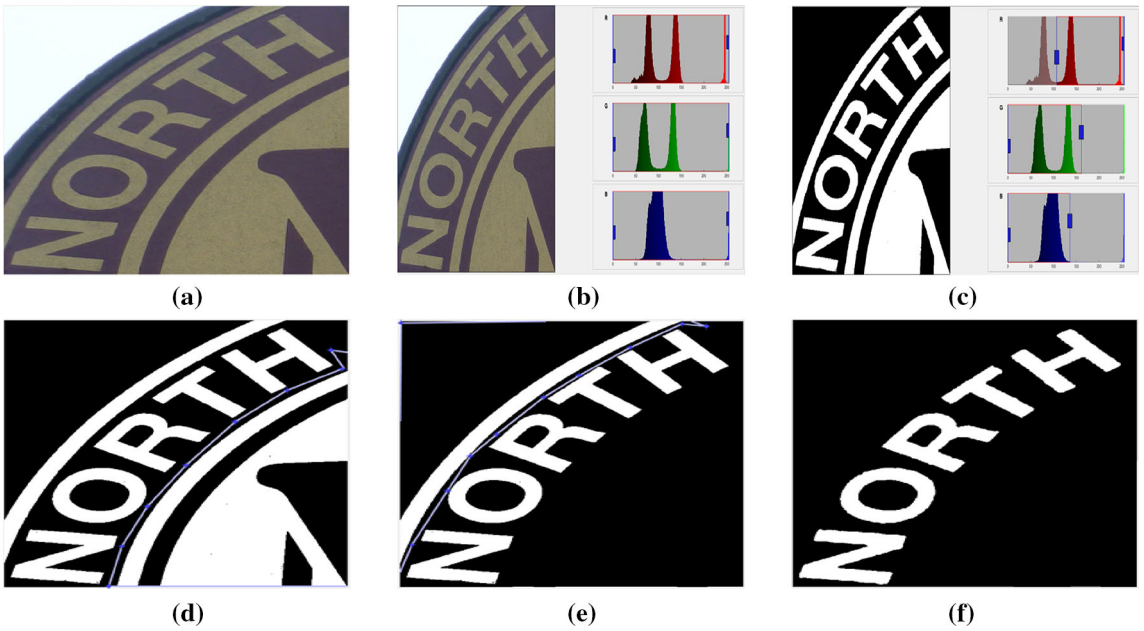


Fig. 5 Pixel-level annotation process. **a** Input image patch. **b**, **c** Adjust the color thresholds to separate text from background region. **d**–**f** Remove 'non-text' region. **g** Final result

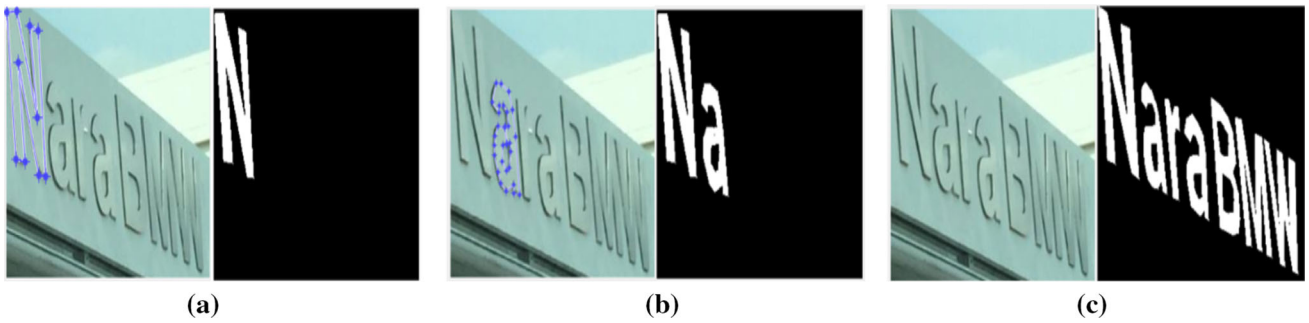


Fig. 6 Pixel-level annotation from scratch. **a** And **b** shows the process of binding the character region, while **c** depicts the final result

the color of the same text instances is usually in contrast to the background region. Adjusting the thresholds would be able to 'zero-out' the pixel (which is the background pixels in this context) below the adjusted threshold (Fig. 5b, c).

(iii) Prompt annotator to bind the regions with the remaining background pixels in order to 'zero-out' them (Fig. 5d, e).

(iv) Repeat step (i) to (iii) until all the text instances are annotated.

However, there are cases where the aforementioned threshold adjustment technique would not be able to help in preprocessing the text patches. Text regions with uneven illumination or color that is similar to the background region are two of such cases. Under such circumstances, annotator has the option to bind the character region from scratch, as visualized in Fig. 6.

At the end of the process, a binary map with text pixel labeled as '1' and background pixel labeled as '0' is produced and saved as a '.png' file.

3.1.6 Orientation annotation

The orientation of every text instance was annotated for modularity convenience. Specifically, the annotation is represented as such: 'h' for horizontal text, 'm' for multi-oriented text, and 'c' for curved text. For example, if one prefers to evaluate the effectiveness of an algorithm to detect curved text only, one could leverage this annotation to filter out instances of other orientations.

Figure 2 depicts all the aforementioned annotations. Considering the scale of this dataset is manageable, we annotated the entire dataset manually and cross checked with three other laboratory members.

Fig. 7 The workflow of our new polygon annotation scheme. (Left) prompt for four vertices. The system will then automatically generate three equidistant guiding lines based on them. (Middle) annotate the interception point (represented by the symbol '*') between yellow guiding lines and the top part of the text. (Right) lastly, annotate the interception point between green guiding lines and bottom part of the text (color figure online)

Algorithm 1 Algorithm to generate the guiding lines in the regulated polygon annotation process.

Let the pair of input vertices be (x_1, y_1) and (x_2, y_2)
$x_{min} = \min(x_1, x_2)$
$x_{max} = \max(x_1, x_2)$
$y_{min} = \min(y_1, y_2)$
$y_{max} = \max(y_1, y_2)$
$width = x_{max} - x_{min}$
$height = y_{max} - y_{min}$
if width > height **then**
  equidistant interval = $(x_{max} - x_{min})\ /4$
  $Line1_{x_1} = x_{min}$ + equidistant interval
  $Line2_{x_2} = Line1_{x_1}$ + equidistant interval
  $Line3_{x_3} = Line2_{x_2}$ + equidistant interval
else if height > width **then**
  equidistant interval = $(y_{max} - y_{min})\ /4$
  $Line1_{y_1} = y_{min}$ + equidistant interval
  $Line2_{y_2} = Line1_{y_1}$ + equidistant interval
  $Line3_{y_3} = Line2_{y_2}$ + equidistant interval
end if

3.1.7 Regulated polygon ground truth

Initial version of Total-Text's polygon annotation was carried out with the mindset of covering text instances tightly with the least amount of vertices. As a result, the uncontrolled length of polygon vertices is not practical to train a regression network. In this paper, we refined Total-Text annotation using the following scheme. Apart from setting the number of polygon vertices to 10 (empirically, 10 vertices are found to be sufficient in covering all the word-level text instances tightly in our dataset), we used a guidance concept inspired by [13], which was introduced to remove human annotators' bias and in turn producing a more consistent ground truth.

The new polygon ground truth annotation steps are illustrated in Fig. 7. First, human annotator is required to manually pick four different vertices that serve as the beginning and ending vertices covering of a word instance. As illustrated in Fig. 7, two vertices at the top corner of the word "MARLEY" (red and green dot) will be employed to generate three equidistant yellow guiding lines. The algorithm that generates the guiding lines is explained in Algorithm 1. The human annotator will then select an interception point (represented as "*") along each yellow guiding line which best binds the top boundary of the word. The same process is repeated for the bottom part of the word (with yellow and cyan dots, and green color guiding lines). Figure 8 shows the comparison between the old and new ground truth. We found that the proposed scheme relief some extend of the cognitive strains during the annotation process as the annotator has to make less decision. Additionally, the described guiding mechanism removes the part where annotators have to decide whether which location to annotate (which could vary largely from one to another), and hence removing potential bias in the produced ground truth.

(a) (b)

Fig. 8 **a** Original ground truth with uncontrolled length of polygon vertices. **b** Our new regulated ground truth with a fixed length of 10 vertices

Besides, the new polygon ground truth was collected with a new condition: the first point of the polygon starts from the starting reading direction of the word, as illustrated in Fig. 9. Such information is useful for the text recognition task.

3.2 Dataset statistics

3.2.1 Strength in numbers

Total-Text is split into two groups: training and testing set with 1255 and 300 images, respectively. Figure 10 shows a series of statistical information of Total-Text. It has a total of 11,459 annotated text instances, 7.37 instances per image on average. More than half of the images in Total-Text have two different orientations and above, yielding 1.8 orientations per image on average. Apart from these, the dataset is also collected with quality in mind, including scene complexity

Fig. 9 New information in the polygon ground truth of Total-Text. Red points are the starting point of each text region (color figure online)

such as text-like and low contrast background, different font types and sizes as illustrated in Fig. 11.

3.2.2 Orientation diversity

Approximately half of the text instances are curved, and the other half is split almost equally between horizontal and multi-oriented. Although all the images were collected with curved text in mind, other orientations still occupy half of the total instances. A closer look into Total-Text revealed that curved text usually appear with either horizontal or multi-oriented text. The mixture of text orientations in an image challenges text detection algorithms to achieve robustness and generalization in terms of text orientations.

3.2.3 Scenery diversity

Scenery in Total-Text's images is well diversified too. Business-related places like restaurant, branding logos, and merchant stores (e.g., Nando's, Starbucks) occupy 61.9% of the curved text instances in Total-Text. 20.5% of them appear in tourist spots such as park, museums, and landmarks (e.g., Beverly Hills in America, Harajuku in Japan) as illustrated in Fig. 1. The rest of curved text instances appear in places like sport club logo, advertising material, alarm devices, etc.

4 Evaluation protocol

4.1 DetEval

Total-Text was first introduced with the DetEval [16] evaluation protocol. However, we realized that the recommended tp and tr thresholds, 0.4 and 0.8, respectively, are not optimized with the inclusion of curved text and polygon ground truth in Total-Text. Firstly, τ is a metric in DetEval which measures *how precise is a detection*. As we can see in Fig. 12c, Poly-FRCNN (detailed in Sect. 6) outperformed the loosely bounded output from Box-FRCNN significantly by 0.35. However, when the tp threshold is set as low as 0.4, both of them gave the same contribution to the final score. In addition, Fig. 13 shows more comparisons between existing state-of-the-art curved text detection models [13,44,45] and Box-FRCNN. As we can in the figure, Box-FRCNN's prediction box is way looser than all the other state-of-the-art methods, with way lower *tau* score (with the smallest gap of 0.4). However, similar to the

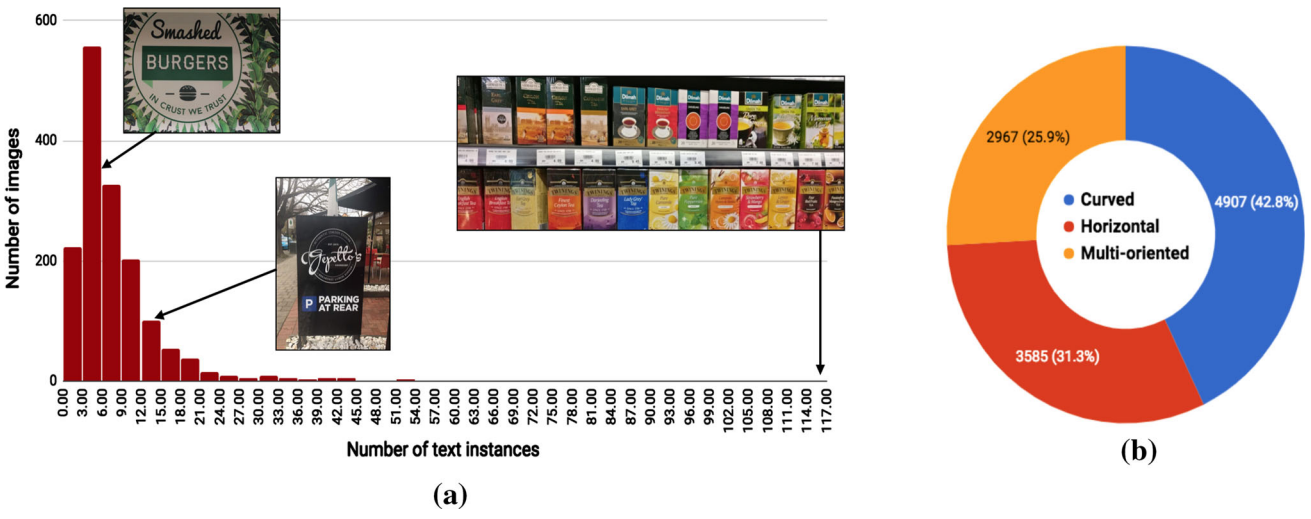


Fig. 10 Statistics of the Total-Text dataset. **a** Distribution of text instances in Total-Text. The images displayed on top of the histogram are the example images with different number of text instances. **b** Distribution of three text orientations on Total-Text

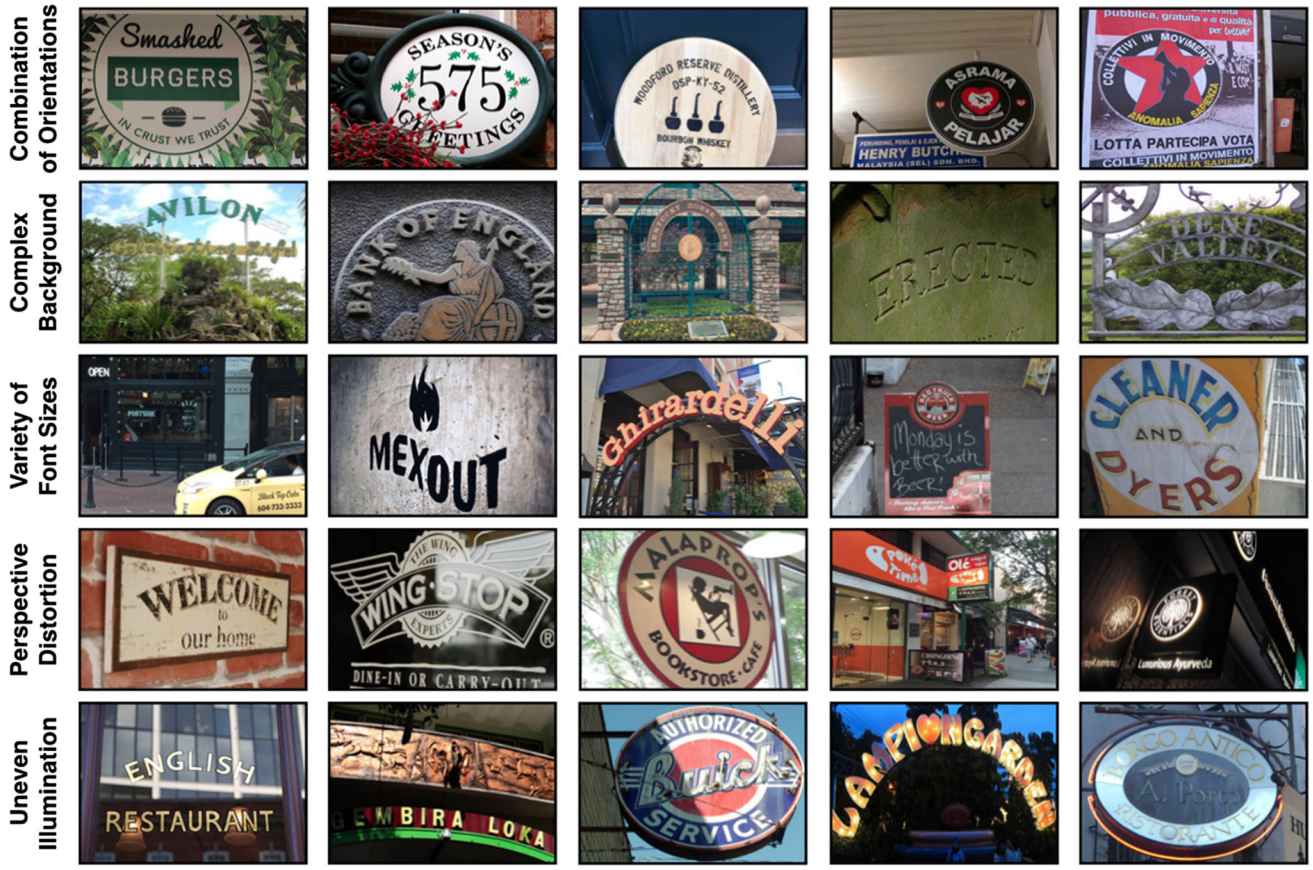
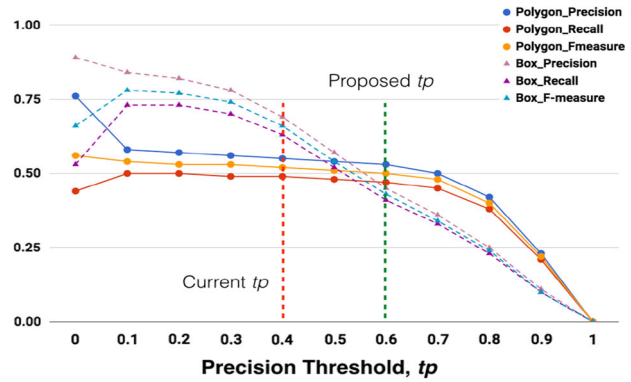


Fig. 11 Challenges in Total-Text. Apart from the commonly seen challenges in the scene text domain such as camouflage background, variations of font size, perspective distortion, and illumination, Total-Text challenges text understanding algorithm to handle a combination of multiple oriented text in a single image

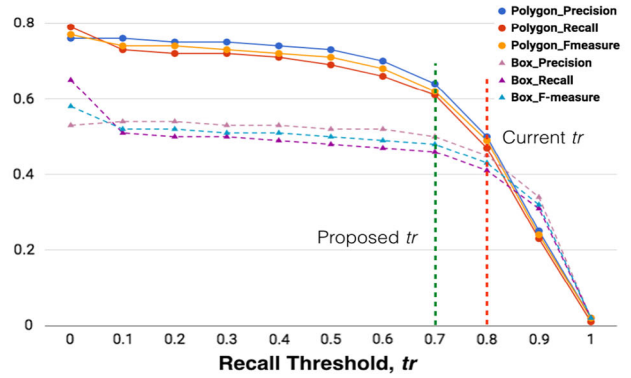
example in Fig. 12c, all of them were classified as True Positive in the evaluation process. Based on these examples, we further establish that the current tp value is not high enough to distinguish the precision of a loose and a tight detection box. Hence, we propose to increase the threshold to 0.6, which as we can see in Fig. 12a, is the point where Box-FRCNN loses its advantage over Poly-FRCNN.

On the other hand, σ measures *how much of the ground truth area is covered*. Figure 12c shows that Box-FRCNN's prediction box scored with its loose prediction almost full marks while Poly-FRCNN scored 0.78 as it tried to match the ground truth precisely. With the tr as high as 0.8, Poly-FRCNN's prediction got forfeited in the final calculation. In order to further validate our point, we again refer to the detection output of existing state-of-the-art methods, which is displayed in Fig. 14. As we can see, all the text regions are bound precisely by the detection outputs, yet they failed to cover more than 80% of the ground truths (i.e., tr = 0.8). This is because polygon ground truth has much more vertices (hence much more offset potential) than an axis-aligned bounding box (which DetEval was designed to address), which is not an easy task to meet at a high level of agreement. Our experiments in Sect. 5 validate this by showing that even human annotation cannot perfectly match each other in multiple attempts. As a result, all of the listed examples were being classified as False Positive during the evaluation process. Based on these findings, we propose to relax the tr, lower it down from 0.8 to 0.7. As observed in Fig. 12a, at our proposed tr, the performance gap between the loose output Box-FRCNN and tight output Poly-FRCNN became significant and much fairer.

On a global scale, Table 1 compares the performance of the latest works which benchmarked on Total-Text with the default threshold values of 0.4 and 0.8 and the proposed threshold values of 0.6 and 0.7. Firstly, we see that the performance of Box-FRCNN ranks 3rd in terms of F-measure with the existing threshold values. Once we configured DetEval to our proposed values, all of Box-FRCNN's scores drop dramatically and ranked last in the table. On the other hand, all the other tight detection box producing methods seen an increment in all metrics mainly due



(a) Sweeping of tp with tr fixed at 0.8.



(b) Sweeping of tr with tp fixed at 0.6.

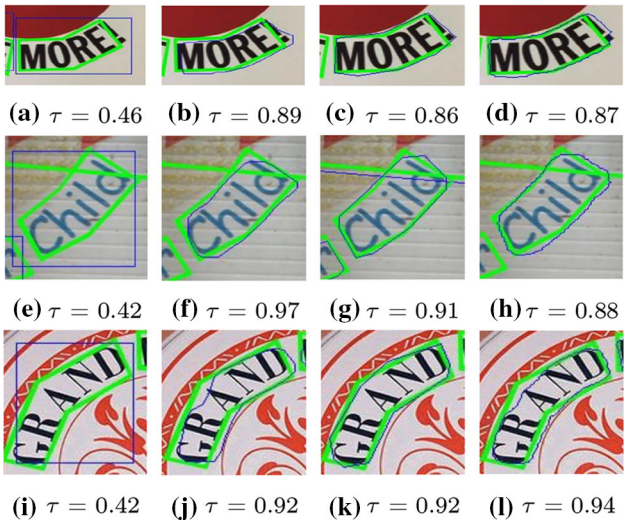
(c) Box-FRCNN : τ : 0.49, σ : 0.97
Poly-FRCNN : τ : 0.84, σ : 0.78

Fig. 12 Performance of Box-FRCNN and Poly-FRCNN across different tr and tp with the DetEval evaluation protocol

to lesser False Positive counts (as a result of the relaxation of tr).

4.2 PASCAL VOC

Similar to CTW1500 and ICDAR2-015, PASCAL VOC evaluation method is made available for Total-Text as well. Our experiment shows that the standard 0.5 Intersection Over Union (IoU) threshold gave Poly-FRCNN a higher score than Box-FRCNN as expected. The mentioned comparison can be seen in Table 2.

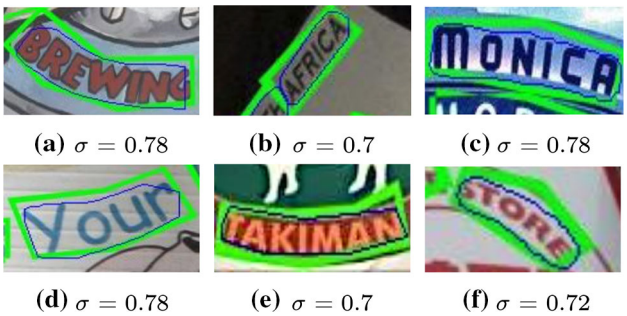


(a) $\tau = 0.46$ **(b)** $\tau = 0.89$ **(c)** $\tau = 0.86$ **(d)** $\tau = 0.87$

(e) $\tau = 0.42$ **(f)** $\tau = 0.97$ **(g)** $\tau = 0.91$ **(h)** $\tau = 0.88$

(i) $\tau = 0.42$ **(j)** $\tau = 0.92$ **(k)** $\tau = 0.92$ **(l)** $\tau = 0.94$

Fig. 13 Detection examples of various methods in compared to Box-FRCNN. All of these detections are counted as a True Positive with the current tp(0.4). Column 1: Box-FRCNN, column 2: CTD [13], column 3: TextField [44], column 4: Mask-TextSpotter [45]. Green: ground truth, blue: detection output (color figure online)



(a) $\sigma = 0.78$ **(b)** $\sigma = 0.7$ **(c)** $\sigma = 0.78$

(d) $\sigma = 0.78$ **(e)** $\sigma = 0.7$ **(f)** $\sigma = 0.72$

Fig. 14 Detection examples of CTD [13] (column 1), Mask-TextSpotter [45] (column 2), and TextField [44] (column 3). All of these detections regions will be discarded by the current tr (0.8). Green: ground truth, blue: detection output (color figure online)

4.3 Intersection area between polygons

The calculation of intersection between the prediction area and the ground truth area is the core of both the DetEval and the Pascal VOC evaluation protocols. The original intersection area calculation algorithm in both the mentioned protocols was designed to address the intersection between rectangles (i.e., axis-aligned bounding box) only. In our implementation, we have replaced that module with the algorithm that is capable of calculating the intersection area between polygons precisely so that they give accurate evaluation. Our implementations of both DetEval and Pascal VOC were written in both MATLAB and Python languages. In our MATLAB implementation, the function 'polybool'[2] is utilized to identify the intersection area, while we use

[2] https://www.mathworks.com/help/map/ref/polybool.html.

Table 1 Comparison between loose bounding box predicting Box-FRCNN and tight bounding box predicting methods with old and new thresholds values of DetEval

DetEval's thresholds	$tp = 0.4, tr = 0.8$				$tp = 0.6, tr = 0.7$			
Methods	Precision	Recall	F-measure	Rank	Precision	Recall	F-measure	Rank
Box-FRCNN	0.74	0.67	0.70	3	0.50	0.46	0.48	6
Poly-FRCNN-3	0.68	0.59	0.63	5	0.78	0.68	0.73	5
CTD [13]	0.61	0.59	0.60	6	0.77	0.74	0.75	4
MSR [46]	0.83	0.68	0.75	2	0.81	0.73	0.77	3
Mask-TextSpotter [45]	0.69	0.63	0.66	4	0.83	0.75	0.79	2
TextField [44]	0.76	0.75	0.76	1	0.83	0.82	0.83	1

Table 2 Evaluation of Poly-FRCNN on Total-Text. All the models were evaluated on both DetEval and Pascal VOC (in the bracket) evaluation protocols

Methods	Wholeset			Curved-set			Non-curved-set		
	P	R	F	P	R	F	P	R	F
Box-FRCNN	0.50 (0.59)	0.46 (0.54)	0.48 (0.56)	0.19 (0.32)	0.21 (0.37)	0.20 (0.34)	0.78 (0.83)	0.63 (0.66)	0.70 (0.74)
Poly-Baseline	0.61 (0.67)	0.53 (0.59)	0.57 (0.63)	0.42 (0.48)	0.42 (0.50)	0.42 (0.49)	0.73 (0.80)	0.60 (0.66)	0.66 (0.73)
Poly-FRCNN-3	**0.78** (**0.80**)	**0.68** (**0.70**)	**0.73** (**0.75**)	**0.66** (**0.69**)	**0.67** (**0.71**)	**0.66** (**0.70**)	**0.84** (**0.86**)	**0.69** (0.70)	**0.76** (**0.77**)
Poly-FRCNN-5	0.72 (0.76)	0.65 (0.69)	0.68 (0.72)	0.56 (0.63)	0.57 (0.65)	0.57 (0.64)	0.8 (0.83)	**0.69** (**0.72**)	0.74 (**0.77**)

The bold values represent the best results in their respectively comparison.
P Precision, *R* Recall, *F* *F*-measure

the 'skimage.draw.polygon'[3] function in Python to generate binary masks for two polygons, and sum them up to obtain their intersection area.

5 Scene text detection annotation tool

The effort of annotating a dataset is usually daunting. Quality annotation format such as our regulated polygon comes with a higher cost than the axis-aligned bounding box and quadrilateral that were commonly used in existing scene text datasets. Inspired by Castrejon et al. [47], we propose a new scene text detection annotation tool to ease the mundane and time-consuming process. Our proposed annotation tool, namely Total-Text-Tool (*T3*), is capable of reducing annotation time by 25% with an agreement rate of 84% with human annotators. The main differences between *T3* and Polygon-RNN [47] are: (i) the latter requires user to input image patches to the system, while *T3* takes the whole image, and (ii) the suggestion (polygon vertices) in *T3* is designed to be aligned with our regulated polygon format as explained in Sect. 3.1, which deemed to fit text instances better.

5.1 Total-Text-Tool

Training a modern scene text detector requires a lot of annotated data and time. In the scene text research domain, most of these invested resources reach the *end of life* when the performance tables are tabulated. The main idea of *T3* is to reuse the well trained and good performing scene text detectors to predict the potential text regions ahead of the annotation process. The predictions will serve as a suggestion for the human annotator.

The ideal flow of a good quality annotation tool should consist of (i) cropping out text region with bounding box, (ii) select a series of vertices for either quadrilateral or polygon that will bind the text instance tightly. This workflow is used in [17] and [13], which will be compared to as a baseline in our experiment. *T3* aids the user with both of the processes in this workflow. The annotation process of *T3* is as follows: (i) firstly, a new scene image will be processed by a scene text detector to output both bounding box and polygon-shaped predictions. (ii) Secondly, *T3* crops the input image with the provided bounding boxes. (iii) Thirdly, *T3* displays the cropped image patches with *interactive* polygon vertices on top for annotator to adjust and fit the text region tightly. Figure 15 illustrates the aforementioned workflow, from left to right of the figure. At the same time, user has the option to discard poorly suggested bounding box or polygon and manually annotate it from scratch using the baseline method.

[3] http://scikit-image.org/docs/dev/api/skimage.draw.html#skimage.draw.polygon.

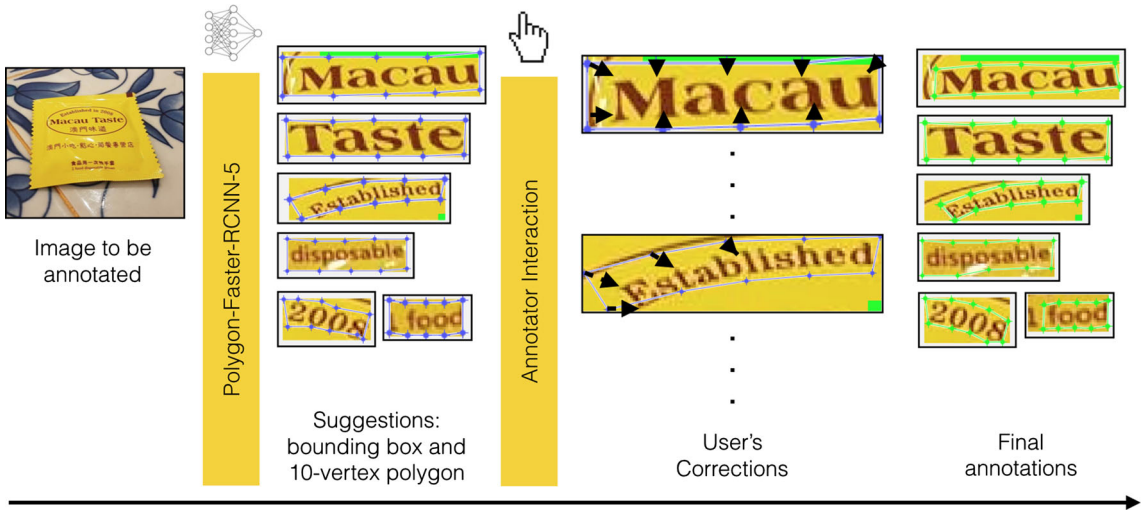


Fig. 15 Overview of the detection ground truth annotation process with the aid of *T3*

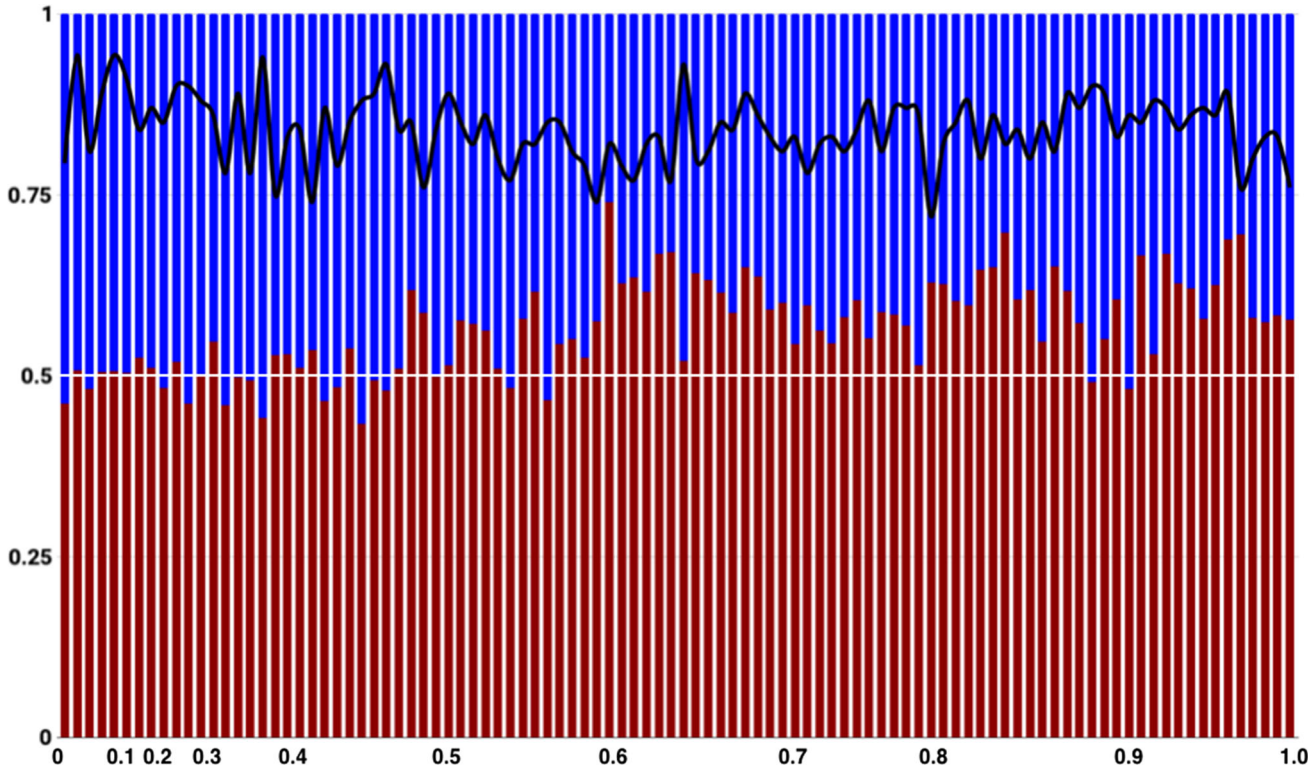


Fig. 16 Comparison of time taken to annotate with *T3* (in blue) and without (in red). The better the performance of Poly-FRCNN on the image, the lesser time it takes for human annotator with the help of *T3*. The agreement of annotation (black line) with the aid of *T3* is remained as high as the one without (left end of the chart), averaged at 84% IoU with a standard deviation of 4% across 100 images we used in this experiment (color figure online)

5.2 Experiment setup

An experiment was carried out in order to measure the efficiency of using *T3*. The scene text detector used in our experiment is Poly-FRCNN-5 (see Sect. 6.1 for details). We selected 100 images from the testing set according to the actual F-measure distribution (depicted by the scale of x-axis of Fig. 16) of Poly-FRCNN-5 to ensure the legitimacy of our experiment. The human annotator was informed and trained on both the baseline and the *T3* annotation tool before

the experiment. The human annotator was given the freedom to take a break whenever he feels like to, ensuring that he will not suffer from fatigue which in turn introduces bias to the experiment. Both time and annotation quality were measured internally (within the script) and individually to each image.

5.3 Performance analysis

In overall, it took the human annotator a total of 8989.58 s (2.5 h) to annotate all the 100 images with the baseline annotation tool. With *T3*, the total time taken is reduced by 25% to 6832.34 s (1.9 h). The quality of both annotation results is in an agreement of 84%, with a standard deviation of 4.6% in IoU terms. Figure 16 illustrates the comparison of time taken (normalized to one) between both methods across all 100 images. As we can see, the better the performance of the detector, the lesser time it takes for the human annotator to finish the annotation process with the aid of *T3*. Note that on those images where the detector scores as low as 0 to 0.4, those suggestions were mostly bad, therefore, the human annotator was required to annotate it from scratch. Hence the time taken for those images is almost identical or sometime worse with *T3* due to the extra decision required to discard the suggestions. Even in such cases, the agreement of annotation between the two methods could not match perfectly. This is mainly due to (i) as pointed out in [47], IoU is a strict metric for small object instances, which is the nature of scene text in general, and (ii) polygon has much more vertices (ten in this case) hence comes with a smaller margin for offset. The *T3* framework is generic; thus, any scene text detector can be part of the workflow. The performance of Poly-FRCNN-5 has much more room for improvement; a better performing detector will in turn further reduce the annotation time with *T3*.

6 Polygon-faster-RCNN

Poly-FRCNN is a novel scene text detection model that is capable of detecting text of all orientations. It was designed to serve as a baseline for future benchmarking on Total-Text. Similar to [13,33,34,37], Poly-FRCNN adopted the Faster-RCNN architecture. In line with our motive to use polygon as the ground truth format, our baseline produces polygon-shaped output, aims to bind text region of all shapes tightly. Specifically, we propose a new text line encoding method to embed every kind of existing text region ground truth formats (i.e., axis-aligned bounding box, quadrilateral, and polygon) into text line parameters. Consequently, Poly-FRCNN is trained to regress the text line location of every text region in an image, which will then be converted to its original text region shape as the final detection output. The following part will discuss all the technical details of Poly-FRCNN.

6.1 Text line encoding method

The conventional regression target (x_{m}, y_{m}, w, h) used in Faster-RCNN, SSD and YOLO can only be used for axis-aligned rectangular boxes, but not polygons. DMPNet [37] used an encoding method for quadrilateral and claimed that relative information is easier to train instead of absolute values of quadrilateral vertices. Following that, [13] trained their CTD with a straightforward polygon encoding method, which convert absolute polygon vertices into a series of relative distance of each vertex to the x_{min}, y_{min} of the particular text region (i.e., h_n,w_n). We argue that such encoding does not embed enough information for the model to learn about text region; hence, we propose a new encoding method, namely 'Text Line Encoding Method', which will convert any text region ground truth into text line parameters. It is worth to note that such encoding method is flexible for all existing text detection ground truth; hence, we are able to carry out experiments on different existing datasets, as detailed in Sect. 6.4.

6.1.1 Variants of Poly-FRCNN

We discovered that six vertices are the minimum amount of vertices required to cover most of the curved texts in Total-Text.[4] Subsequently, Poly-FRCNN-3 is designed to produce six vertices polygon bounding region. Our encoding method splits six vertices into three opposing pairs, hence the notation '3'. The overview of Poly-FRCNN-3 architecture can be seen in Fig. 17. The initial version of Total-Text polygon ground truth was converted to have a fixed length of six vertices for the training process, which can be seen in Fig. 18a. On the other hand, Poly-FRCNN-5 is the scaled-up version of 3, which has 10 more parameters in the regression head. Poly-FRCNN-5 regresses five pairs of opposing vertices. It was trained with the regulated polygon ground truth mentioned in Sect. 3.1. With more vertices, Poly-FRCNN-5 aims to produce tighter detection outputs than Poly-FRCNN-3. Moreover, it is the scene text detector used in the *T3* aided annotation tool as described in Sect. 5. All the following discussions will be based on the Poly-FRCNN-3 model for simplicity.

6.1.2 Encode

The process of encoding converts text region into text line parameters. Consider the text instance in Fig. 18a, bound by a polygon with six vertices ($\{\mathbf{x_0}, \mathbf{y_0}\}$; $\{\mathbf{x_1}, \mathbf{y_1}\}$; $\{\mathbf{x_2}, \mathbf{y_2}\}$; $\{\mathbf{x_3}, \mathbf{y_3}\}$; $\{\mathbf{x_4}, \mathbf{y_4}\}$; $\{\mathbf{x_5}, \mathbf{y_5}\}$), we first split them into three pairs where every opposing vertex form a pair. For instance,

[4] It is sufficient to cover most of the text regions in Total-Text but not texts with larger curvature. Examples in Fig. 21.

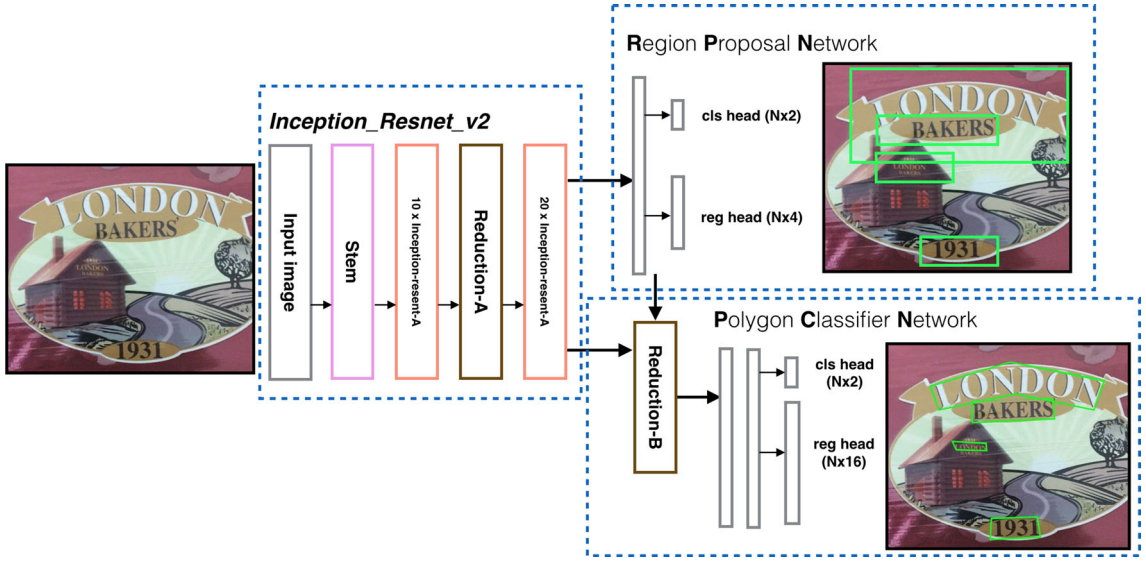


Fig. 17 The overview of Poly-FRCNN. The polygon classifier network (PCN) module is designed to regress polygon-shaped detection region

$\{\mathbf{x_0}, \mathbf{y_0}\}$ is a pair with $\{\mathbf{x_5}, \mathbf{y_5}\}$. Then, for every pair of vertices, we encode them into five values. Taking the $\{\mathbf{x_0}, \mathbf{y_0}\}$ and $\{\mathbf{x_5}, \mathbf{y_5}\}$ pair as example,

(i) $x_{\mathrm{m}(0)}$ is the middle point between x_0 and x_5,
(ii) $y_{\mathrm{m}(0)}$ is the middle point between y_0 and y_5,
(iii) $h_{(0)}$ is the height between y_0 and y_5,
(iv) $w_{(0)}$ is the width between x_0 and x_5, and

Figure 18b visualizes the encoding method, and the formulations are listed in Eq. 2.

$$\begin{aligned} x_{\mathrm{m}(i)} &= (x_{(i)} + x_{((N-1)-i)})/2 \\ y_{\mathrm{m}(i)} &= (y_{(i)} + y_{((N-1)-i)})/2 \\ h_{(i)} &= y_{((N-1)-i)} - y(i) \\ w_{(i)} &= x_{((N-1)-i)} - x(i) \\ \text{For } i &= [0, N/2) \end{aligned} \tag{2}$$

'i' and $((N-1)-i)$ translates to the opposing pair of vertices, where N is the total number of vertices. We replace $y_{\mathrm{m}(i)}$ and $x_{\mathrm{m}(i)}$ from the second point onwards ($i >= 1$) with the *distance from its previous middle point*, denoted as $dy_{\mathrm{m}(i)}$ and $dx_{\mathrm{m}(i)}$ (formulated with Eq. 3) to further reduce the need for absolute information [37].

$$\begin{aligned} dx_{\mathrm{m}(i)} &= x_{\mathrm{m}(i)} - x_{\mathrm{m}(i-1)} \\ dy_{\mathrm{m}(i)} &= y_{\mathrm{m}(i)} - y_{\mathrm{m}(i-1)} \\ \text{For } i &= [1, N/2) \end{aligned} \tag{3}$$

Figure 18c visualizes the aforementioned encoding. At the end of the encoding, we obtain twelve parameters ($x_{\mathrm{m}(0)}$, $y_{\mathrm{m}(0)}$, $h_{(0)}$, $w_{(0)}$, $dx_{\mathrm{m}(1)}$, $dy_{\mathrm{m}(1)}$, $h_{(1)}$, $w_{(1)}$, $dx_{\mathrm{m}(2)}$, $dy_{\mathrm{m}(2)}$,

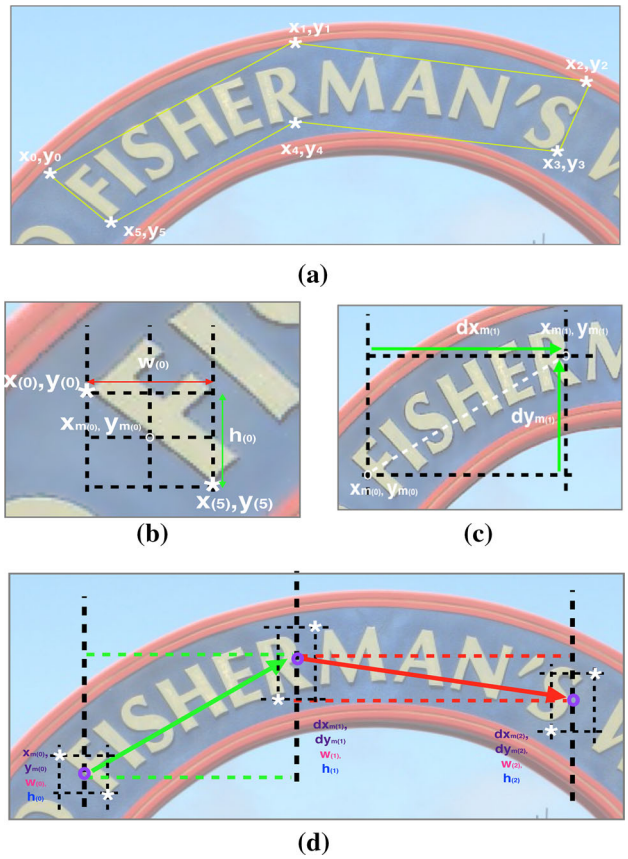


Fig. 18 Text line encoding method. **a** Example of a 6-vertex converted polygon ground truth in Total-Text. **b** Encoding of $x_{\mathrm{m}(i)}$, $y_{\mathrm{m}(i)}$, $h_{(i)}$, $w_{(i)}$. **c** Encoding of $dx_{\mathrm{m}(i)}$ and $dy_{\mathrm{m}(i)}$. **d** Twelve parameters at the end of the encoding process

$h_{(2)}$, $w_{(2)}$) which resemble a text line as our training target, which can be seen in Fig. 18. Note that we refer these parameters as 'encoded polygon' from here on.

6.1.3 Decode

The process of decoding converts the regressed text line values back to the text region detection output. During decoding, we will first determine all the middle points from second points onwards, ($x_{\mathrm{m}(i)}$ and $y_{\mathrm{m}(i)}$) using Eq. 4:

$$\begin{aligned} x_{\mathrm{m}(i)} &= x_{\mathrm{m}(i-1)} + dx_{\mathrm{m}(i)} \\ y_{\mathrm{m}(i)} &= y_{\mathrm{m}(i-1)} + dy_{\mathrm{m}(i)} \\ \text{For } i &= [1, N/2) \end{aligned} \tag{4}$$

Then h_i, $w_{(i)1}$, and $w_{(i)2}$ will be used to transform these middle points back to six vertices using Eqs. 5-6.

$$\begin{aligned} x_{(i)} &= x_{\mathrm{m}(i)} - w_{(i)}/2 \\ y_{(i)} &= y_{\mathrm{m}(i)} - h_{(i)}/2 \\ \text{For } i &= [0, N/2) \end{aligned} \tag{5}$$

$$\begin{aligned} x_{(i)} &= x_{\mathrm{m}(i)} + w_{(i)}/2 \\ y_{(i)} &= y_{\mathrm{m}(i)} + h_{(i)}/2 \\ \text{For } i &= [N/2, N) \end{aligned} \tag{6}$$

The transformation is straightforward, $y_{(0)}$, $y_{(1)}$, and $y_{(2)}$ will always be the opposite of $y_{(5)}$, $y_{(4)}$, and $y_{(3)}$, respectively, with the height values: $h_{(0)}$, $h_{(1)}$, and $h_{(2)}$ (likewise for $x_{(i)}$ with $w_{(i)}$).

6.2 Anchor polygons parameterization

In line with Faster-RCNN, our encoded polygon ground truth and predictions were further parameterized with the encoded anchor polygons. We use the term anchor polygons because we turn four vertices rectangular anchor box into six vertices by adding middle points between the corresponding vertices. The network essentially regresses the offset value between the encoded anchor polygons and the nearby encoded ground truth (both encoded with the Text Line encoding method in Sect. 6.1). Note that our model regresses both a rectangular bounding box and an encoded polygon for each proposal. While the parameterization of the rectangular bounding box is the same with Faster-RCNN [15], we adopted Eq. 7 for our encoded polygon.

$$\begin{aligned} t_{x_{\mathrm{m}(i)}} &= (x_{\mathrm{m}(i)} - xa_{\mathrm{m}(i)})/Wa \\ t_{y_{\mathrm{m}(i)}} &= (y_{\mathrm{m}(i)} - ya_{\mathrm{m}(i)})/Ha \\ t_{w_{(i)}} &= (w_{(i)} - wa_{(i)})/Wa \\ t_{h_{(i)}} &= (h_{(i)} - ha_{(i)})/Ha \\ \text{For } i &= [0, N/2) \\ t_{dx_{\mathrm{m}(j)}} &= (dx_{\mathrm{m}(j)} - dxa_{\mathrm{m}(j)})/Wa \\ t_{dy_{\mathrm{m}(j)}} &= (dy_{\mathrm{m}(j)} - dya_{\mathrm{m}(j)})/Ha \\ \text{For } j &= [1, N/2) \\ \text{where } Wa &= (xa_{\max} - xa_{\min}) \\ Ha &= (ya_{\max} - ya_{\min}) \end{aligned} \tag{7}$$

where y and ya represent the encoded predicted polygon (also the encoded ground truth polygon) and encoded anchor polygon, respectively (same applies to x, h, w, dy, and dx).

6.3 Implementation details

All of our experiments and evaluations were ran on a machine with Intel 6-core Xeon chip, 64 GB of RAM, Nvidia Titan X Pascal Architecture, and Ubuntu OS v16.04. All of our experiments and models were implemented with the Tensorflow (version 1.9) object detection API [48]. Parameters or variables that are not discussed in this section are consistent with [48].

6.3.1 Feature extractor

Our model adopted Inception-Resnet-V2 [49] as the feature extractor.

6.3.2 Anchor boxes

As suggested in [33], text in the wild is generally smaller and has wider aspect ratio; hence, we define the scales and aspect ratio of our anchor boxes as (0.0625, 0.125, 0.25, 0.5, 1.0) and (0.5, 1.0, 2.0, and 3.0), respectively. As mentioned in Sect. 6.2, we converted the anchor boxes in the second stage (which are proposal boxes essentially) to anchor polygons by merely adding a middle point between the corresponding vertices. This is essential so that it can be parameterized with our ground truth in the polygon format.

6.3.3 Loss function

The optimization of Poly-FRCNN is a multi-task problem. For the region proposal network (RPN) module, the loss function is consistent with Faster-RCNN's [48], while the loss function of the polygon classifier network (PCN) module is

formulated as Eq. 8.

$$L(p_i, t_i) = \frac{\alpha}{N} \sum_i L_{\text{cls}}(p_i, p_i^*) + \frac{\beta}{N} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*) \quad (8)$$

where i is the index of anchor in a mini-batch, while p_i and p_i^* are the predicted probability and the ground truth label of anchor i being a text instance. On the other hand, t_i represents the predicted parameterized target as described in Eq. 7 and t_i^* is its ground truth counterpart. N is the mini-batch number, which acts as a normalizer for both of the loss functions. α and β are the regularization terms to control the weightage of both loss terms, which are set to 1 and 2, respectively, as studied in [48]. Meanwhile, L_{cls} and L_{reg} are formulated in Eqs. 9 and 10, respectively.

$$L_{\text{cls}}(p, p^*) = -(p^* \log(p) + (1 - p^*) \log(1 - p)) \quad (9)$$

$$L_{\text{Reg}}(t_i, t_i^*) = \sum\nolimits_i \text{smooth}_{L_1}(t_i - t_i^*)$$
$$\text{For } i \in x_{\text{m}(0)}, y_{\text{m}(0)}, w_{(0)}, h_{(0)},$$
$$dx_{\text{m}(1)}, dy_{\text{m}(1)}, w_{(1)}, h_{(1)},$$
$$dx_{\text{m}(2)}, dy_{\text{m}(2)}, w_{(2)}, h_{(2)} \quad (10)$$

in which,

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \quad (11) \\ |x| - 0.5 & \text{otherwise,} \quad (12) \end{cases}$$

6.3.4 Training

All the models went through the same training schedule. They were first initialized with the ImageNet-pre-trained weights. The training schedule then starts with 100 K iterations on SynthText, followed by another 100 K iterations on the real-world data from COCO-Text. Finally, we fine-tuned them with the targeted training set[5] for another 50 K iterations. The initial learning rate on the largest dataset (SynthText) in this training schedule was set to 0.003, then it was reduced 0.0003 since the starting of the training on COCO-Text and remained the same for the rest of the training.

100 K images from SynthText were randomly chosen for the first stage of the training. Then, approximately 13 K training images from COCO-text that have at least one legible text instance were used during the second training stage. Lastly, 1255 images from the Total-Text training set were used for the fine-tuning stage.

[5] Apart from CUTE80 and CTW1500, which we used the model fine-tuned on Total-Text only.

Table 3 Evaluation results on ICDAR2013

Methods	Precision	Recall	F-measure
FOTS [50]	**0.95**	**0.90**	**0.93**
R^2CNN [33]	0.94	0.83	0.88
DIRECT [41]	0.92	0.81	0.86
TextBoxes[39]	0.89	0.83	0.86
Seglink [38]	0.88	0.83	0.85
RRPN [34]	0.90	0.72	0.80
Poly-FRCNN-3	0.90	0.83	0.86

The bold values represent the best results in their respectively comparison

Table 4 Evaluation results on ICDAR2015

Methods	Precision	Recall	F-measure
FOTS [50]	**0.88**	**0.92**	**0.90**
R^2CNN [33]	0.86	0.80	0.83
DIRECT [41]	0.82	0.80	0.81
RRPN [34]	0.73	0.82	0.77
Seglink [38]	0.73	0.77	0.75
DMPNet[37]	0.73	0.68	0.70
Poly-FRCNN-3	0.80	0.66	0.73

The bold values represent the best results in their respectively comparison

6.3.5 Testing

This process was kept as simple as possible. No post-processing or multi-scale were used apart from the standard non-maximum suppression (NMS).

6.4 Evaluation

6.4.1 Dataset

We evaluate the performance of Poly-FRCNN on ICDAR2013, ICDAR2015, and Total-Text. ICDAR2-013 and ICDAR2015 were chosen to demonstrate the performance of Poly-FRCNN on horizontal and multi-oriented text, respectively.

6.4.2 Evaluation Protocol

The performance reported on ICDAR2013 was evaluated using the recommendations (i.e., $tp = 0.4$ and $tr = 0.8$) in DetEval protocol [16] for a fair comparison with the state-of-the-art solutions. For Total-Text, we used $tp = 0.6$ and $tr = 0.7$ as described in Sect. 4. Meanwhile, standard 0.5 IoU threshold for Pascal VOC evaluation method was used for the results on both ICDAR2015 and Total-Text.

Fig. 19 Detection results of Poly-FRCNN-3 on ICDAR2013 (1st row), ICDAR2015 (2nd row), and Total-Text (3rd row)

6.4.3 Performance analysis

Tables 3 and 4 show that the performance of Poly-FRCNN-3 (our best performing model) is on par with contemporary state-of-the-art works like [33,34,37–39]. Example of Poly-FRCNN-3 performance can be seen in Fig 19.

Table 2 tabulates the performance of Poly-FRCNN on Total-Text. In order to measure the effectiveness Poly-FRCNN against of text of different orientations, especially curved text, we created two subsets of Total-Text with the orientation annotation. The 'curved-set' in Table 2 consists of only curved text; while the 'non-curved-set' includes only horizontal and multi-oriented text.

6.4.4 Box-FRCNN versus Poly-FRCNN-3

Box-FRCNN is basically a Faster-RCNN (it produces axis-aligned box as output, hence the 'box' notion), which we built Poly-FRCNN on. This comparison is made to demonstrate the effectiveness of our polygon predicting model against the axis-aligned predicting model in detecting text of different orientations. Poly-FRCNN-3 outperformed its box predicting counterpart by 0.25 (0.19) in terms of F-measure in general. Their performance difference became much more significant on the curved text only subset of Total-Text, with a gap of 0.46 (0.36). This result demonstrates the effectiveness our polygon extension. Such difference is also displayed in Fig. 20, where the detection output of Box-FRCNN appeared to be loosely binding the text region; while Poly-FRCNN-3 is capable of binding all text regions tightly regardless of its orientation.

6.4.5 Poly-Baseline versus Poly-FRCNN-3

Poly-Baseline on the other hand is the model integrated with the polygon encoding method as mentioned in Sect. 6.1 by [13]'s CTD model. Poly-Baseline was implemented to have the exact same architecture as our proposed Poly-FRCNN

Fig. 20 Comparison between Box-FRCNN (1st row), Poly-Baseline (2nd row), and Poly-FRCNN-3 (3rd row)

except for the encoding module, hence making it a fair comparison for the encoding method. As we can see in Table 2, the performance of Poly-FRCNN-3 is superior to Poly-Baseline in all metrics on Total-Text. The largest difference between the two can be observed in the curved-set category, with a gap of 0.24 (0.21) in terms of F-measure. As observed in Fig. 20, Poly-Baseline did not bind the text regions as precise as our proposed Poly-FRCNN-3, proving the superiority of our proposed Text Line encoding method.

6.4.6 Poly-FRCNN-5

The performance of Poly-FRCNN-5 drops slightly by 0.05 (0.03) in terms of F-measure (Table 2) in compared to Poly-FRCNN-3 in the wholeset category. As we can see in Fig. 21, the detection outputs of Poly-FRCNN-5 are smoother with more vertices. However, it comes with the cost of smaller error margin than Poly-FRCNN-3 (i.e., more prone to error due to the higher number of vertices to be regressed), which we attribute as the cause of the loss in performance.

6.4.7 Inference time

The time performance of our proposed models is recorded as well. Across the 300 testing images of Total-Text, Poly-FRCNN-3 runs at 3.3 FPS; while Poly-FRCNN-5 runs at 3.257 FPS.

Fig. 21 1st row: Poly-FRCNN-3; 2nd row: Poly-FRCNN-5. Poly-FRCNN-5's output has more vertices, hence, is able to bind curved text with larger curvature more tightly

Table 5 Evaluation Result of Poly-FRCNN-3 on CUTE80 and CTW1500

Dataset	Precision	Recall	F-measure
CUTE80	0.66	0.64	0.65
CTW1500	0.86	0.62	0.72

6.4.8 Performance on other curved text datasets

Apart from Total-Text, we have also evaluated our proposed model on other curved text datasets—CUTE80 and CTW1500. Poly-FRCNN-3 achieves 0.65 and 0.72 in terms of F-measure on the mentioned datasets, respectively (Table 5). Note that there was an inconsistency in the annotation granularity in CUTE80 (i.e., mixture of word level and line level), hence we reannotate them to word level[6] before evaluation. Besides, we found that there are some overlapping images between the training set of Total-Text and CUTE80. Hence, we fine-tuned Poly-FRCNN-3 on a new subset of Total-Text, with those duplicated images removed from the training data. The result of our proposed model on CUTE80 is similar to the curved-set subset of Total-Text (0.65 vs 0.67 in terms of F-measure with DetEval), mainly due to the fact that most of the text instances in CUTE80 are curved. As for CTW1500, the authors provided us a new set of ground truth[7] with Latin script being annotated in word level, which we used to evaluate our model. Since both of the ground truths are newly obtained and introduced in this paper, our method can serve as a baseline for both of the datasets.

6.5 Cross datasets experiment

In order to determine the impact of Total-Text on detection models, we performed a cross dataset experiment. The datasets involved in this experiment are SynthText, COCO-Text, ICDAR2013, ICDAR2015, and Total-Text. Poly-FRCNN-3 is the model used in this investigation. The training details of this experiment is the same as Sect. 6.3. Table 6 tabulates the finding of this experiment. Below are some observations we obtained from this experiment.

6.5.1 Pretraining on SynthText and COCO-Text only

The model trained on SynthText and COCO-Text only (1st row in Table 6) generally performs worse than the other models with a fine-tuning process on the respective datasets. While the model's performance is still competitive on ICDAR2013 and ICDAR2015 even without fine-tuning on them; its performance on Total-Text is the worst, with a large gap of 0.24 in terms of F-measure in compared to the best performing model (4th row). The main reason is that SynthText and COCO-Text use axis-aligned box as ground

[6] The new ground truths will be released in the same GitHub page as well.

[7] Credit to Baidu Inc. who helped in re-annotating the ground truth in such format. We (the authors of CTW1500 and us) reached a common ground that Latin scripts should be annotated in word level while Chinese scripts should be annotated in line level due to the nature of both languages.

Table 6 Cross dataset experiment with Poly-FRCNN-3

Datasets		ICDAR2013			ICDAR2015			Total-Text		
Pretraining	Fine-tuning	P	R	F	P	R	F	P	R	F
SynthText –> COCO-Text	–	0.81	**0.83**	0.82	0.70	0.59	0.64	0.51	0.47	0.49
SynthText –> COCO-Text	ICDAR2013	0.86	0.82	0.84	0.79	0.52	0.62	0.56	0.42	0.48
SynthText –> COCO-Text	ICDAR2015	0.89	0.75	0.81	**0.80**	**0.66**	**0.72**	0.67	0.52	0.58
SynthText –> COCO-Text	Total-Text	**0.90**	**0.83**	**0.86**	**0.80**	0.6	0.68	**0.78**	**0.68**	**0.73**

The bold values represent the best results in their respectively comparison

truth, hence the model has no data to learn in predicting polygon-shaped detection region.

6.5.2 Fine-tuning on ICDAR2013 and ICDAR2015

The second and third models (2nd row and 3rd row, respectively, in Table 6), after being fine-tuned on ICDAR2013 and ICDAR2015, have seen improvements on their target datasets. However, only the third model seen an improvement when being evaluated on Total-Text, with an increment of 0.09 in terms of F-measure in compared to the first model. ICDAR2013 consists of mostly horizontal text and uses axis-aligned box as ground truth, it is not a surprise to see that fine-tuning on it does not help in improving its performance on Total-Text. ICDAR2015 on the other hand has more data, with multi-oriented text included in their collection, did manage to boost the performance of the model on the proposed dataset. However, it uses quadrilateral as the detection ground truth, which again is not capable of handling the orientation diversity of Total-Text, especially curved text instances.

6.5.3 Fine-tuning on Total-Text

The fourth model (4th row in Table 6), after being fine-tuned on Total-Text, is the highest scoring model on ICDAR2013 and Total-Text, and ranked 2nd on ICDAR2015. While the model was fine tuned only on Total-Text, it achieves good result on the other two datasets, which demonstrates that Total-Text's data is diversified enough for the generalization of the model.

6.6 State-of-the-art analysis

Since the publication of [12], several works have attempted to address the curved text detection problem [13,44,45,52]. This section aims to compare these latest works and provide readers a general idea about the direction that the community is currently moving toward.

Lyu et al. [45] proposed an end-to-end trainable text spotting framework that is capable of handling text of arbitrary orientation. They built their model on top of the Mask-RCNN framework [54] and leveraged its strong segmentation ability to segment text region from background to produce a binary mask (as illustrated in Fig. 2b). Consequently, they calculate the contour region of the mentioned binary mask to produce the spatial coordinates of polygon vertices as final detection result. Similarly, Long et al. [52] leveraged FCN in their proposed model. The novelty of their work lies in the geometry properties maps prediction on top of the text region segmentation maps. The authors then proposed several heuristics algorithms which make use of these geometry properties (i.e., central axis points, radii, and orientations) to generate final polygon vertices prediction. Segmentation-based methods usually have issues in separating different text instances in their binary map output. Motivated by this problem, [44] model their network to learn 'direction field' which provides the information on how far away is one particular text pixel to non-text pixel. The proposed method, TextField, utilizes this information to refine the binary output mask and produce final detection result. Dai et al. [53]'s proposed framework is also a segmentation-based method. The framework places emphasize on fusing lower-level feature maps into top-level part of the network, which has proven to be effective across various datasets. However, we notice that the last part of [53]'s method is supposed to produce a minimum quadrilateral from the segmentation mask as the final detection output. The method was not presented clearly, and we are unsure whether did the authors alter their algorithm (i.e., to produce polygon-shaped output) when evaluating on Total-Text. Sun et al. [51] proposed an end-to-end trainable text spotting framework which is robust against text of all orientations. We have evaluated most of these methods with our proposed threshold values indicated in Sect. 4 and tabulated them in Table 7. Apart from the result from the whole testing set, we have evaluated them on the 'curved' and 'non-curved' subsets of the proposed dataset for a better comparison.

7 Conclusion

This paper introduces a new scene text dataset, Total-Text, featuring the missing element in current scene text datasets—curved text. We believe that curved text should be included as part of the 'multi-oriented' text detection problem. While

Table 7 Compilation of latest works on Total-Text, evaluated with DetEval (our proposed thresholds) and Pascal VOC (in bracket)

Methods	Wholeset			Curved-set			Non-curved-set		
	P	R	F	P	R	F	P	R	F
TextNet*[51]	0.68 (–)	0.60 (–)	0.64 (–)	–	–	–	–	–	–
TextSnake*[52]	0.83 (–)	0.75 (–)	0.78 (–)	–	–	–	–	–	–
FTSN*[53]	– (0.85)	– (0.78)	– (0.81)	–	–	–	–	–	–
CTD [13]	0.77 (0.81)	0.74 (0.78)	0.75 (0.79)	0.70 (0.72)	0.79 (0.83)	0.74 (0.77)	0.71 (0.79)	0.69 (0.74)	0.7 (0.77)
MSR [46]	0.81 (0.73)	0.73 (0.51)	0.77 (0.61)	0.66 (0.59)	0.76 (0.56)	0.70 (0.57)	**0.79** (0.74)	0.66 (0.48)	0.72 (0.58)
Mask-TextSpotter [45]	**0.83 (0.87)**	0.75 (0.80)	0.79 **(0.84)**	**0.83 (0.86)**	0.85 **(0.87)**	**0.84 (0.86)**	0.77 **(0.84)**	0.67 (0.74)	0.72 **(0.79)**
TextField [44]	**0.83** (0.80)	**0.82 (0.82)**	**0.83** (0.81)	0.77 (0.73)	**0.87 (0.87)**	0.82 (0.79)	**0.79** (0.79)	**0.77 (0.78)**	**0.78 (0.79)**

*Did not receive author's model's detection output
This table tabulates methods which were designed with curved text in consideration only

it is under-researched at the moment, we hope the availability of Total-Text, with all other supporting elements we have presented in this paper, could change this scenario. Moving forward, we believe that the regulated polygon ground truth template and the proposed aided scene text detection annotation tool, *T*3, could help in providing a better scene text dataset in terms of quality and scale. Last but not least, we have presented a text detection baseline model for Total-Text, Poly-FRCNN, which has demonstrated its effectiveness in detecting text of all orientations.

Acknowledgements Funding was provided by Fundamental Research Grant Scheme (FRGS) MoHE (Grant No. FP004-2016) and Postgraduate Research Grant (PPP) (Grant No. PG350-2016A). The authors acknowledge all the authors who provided their results for our experiments. Also, we would like to thank Chun Chet Ng for his contribution in aiding the annotation process of Total-Text.

References

1. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., i Bigorda, L.G., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., De Las Heras, L.P.: ICDAR 2013 robust reading competition. In: 12th International Conference on Document Analysis and Recognition (ICDAR). 37(7), pp. 1484–1493 (2013)
2. Yao, C., Bai, X., Liu, W., Ma, Y., Tu, Z.: Detecting texts of arbitrary orientations in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1083–1090 (2012)
3. Ye, Q., Doermann, D.: Text detection and recognition in imagery: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **37**(7), 1480–1500 (2015)
4. Zhang, Z., Shen, W., Yao, C., Bai, X.: Symmetry-based text line detection in natural scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2558–2567 (2015)
5. Huang, W., Lin, Z., Yang, J., Wang, J.: Text localization in natural images using stroke feature transform and text covariance descriptors. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1241–1248 (2013)
6. Neumann, L., Matas, J.: Scene text localization and recognition with oriented stroke detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 97–104 (2013)
7. Huang, W., Qiao, Y., Tang, X.: Robust scene text detection with convolution neural network induced mser trees. In: European Conference on Computer Vision, pp. 497–511 (2014)
8. Pan, Y.-F., Hou, X., Liu, C.-L.: A hybrid approach to detect and localize texts in natural scene images. IEEE Trans. Image Process. **20**(3), 800–813 (2011)
9. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., Shafait, F. (2015) ICDAR 2015 competition on robust reading. In: 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 1156–1160
10. Veit, A., Matera, T., Neumann, L., Matas, J., Belongie, S.: Coco-text: dataset and benchmark for text detection and recognition in natural images (2016). arXiv preprint arXiv:1601.07140
11. Gupta, A., Vedaldi, A., Zisserman, A.: Building a perception based model for reading cursive script. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2315–2324 (2016)
12. Ch'ng, C.K., Chan, C.S.: Total-Text: a comprehensive dataset for scene text detection and recognition. In: IEEE 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 935–942 (2017)
13. Liu, Y., Jin, L., Zhang, S., Luo, C., Zhang, S.: Curved scene text detection via transverse and longitudinal sequence connection. In: Pattern Recognition (2019)
14. Risnumawan, A., Shivakumara, P., Chan, C.S., Tan, C.L.: A robust arbitrary text detection system for natural scene images. Expert Syst. Appl. **41**(18), 8027–8048 (2014)
15. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
16. Wolf, C., Jolion, J.M.: Object count/area graphs for the evaluation of object detection and segmentation algorithms. Int. J. Doc. Anal. Recognit. (IJDAR). **8**(4), 280–296 (2006)
17. Karatzas, D., Gómez, L., Nicolaou, A., Rusiñol, M.: The robust reading competition annotation and evaluation platform. In: 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 61–66 (2018)
18. Yin, X.C., Pei, W.Y., Zhang, J., Hao, H.W.: Multi-orientation scene text detection with adaptive clustering. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1930–1937 (2015)
19. Nayef, N., Yin, F., Bizid, I., Choi, H., Feng, Y., Karatzas, D., Luo, Z, et al.: ICDAR2017 robust reading challenge on multi-lingual scene text detection and script identification-RRC-MLT. In: IEEE 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1 pp. 1454–1459 (2017)

20. Shi, B., Yao, C., Liao, M., Yang, M., Xu, P., Cui, L., Belongie, S., Lu, S., Bai, X.: ICDAR2017 competition on reading Chinese text in the wild (RCTW-17). In: IEEE 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1 pp. 1429–1434 (2017)
21. He, M., Liu, Y., Yang, Z., Zhang, S., Luo, C., Gao, F., Zheng, Q., Wang, Y., Zhang, X., Jin, L.: ICPR2018 contest on robust reading for multi-type web images. In: 2018 24th International Conference on Pattern Recognition (ICPR), pp. 7–12 (2018)
22. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2963–2970 (2010)
23. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. In: Image and Vision Computing, pp. 761–767 (2004)
24. Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: International Conference in Pattern Recognition, pp. 3304–3308 (2012)
25. Jaderberg, M., Vedaldi, A., Zisserman, A.: Deep features for text spotting. In: European Conference on Computer Vision, pp. 512–528 (2014)
26. He, T., Huang, W., Qiao, Y., Yao, J.: Text-attentional convolutional neural network for scene text detection. IEEE Trans. Image Process. **25**(6), 2529–2541 (2016)
27. Zhang, Z., Zhang, C., Shen, W., Yao, C., Liu, W., Bai, X.: Multi-oriented text detection with fully convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4159–4167 (2016)
28. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440 (2015)
29. He, T., Huang, W., Qiao, Y., Yao, J.: Accurate text localization in natural image with cascaded convolutional text network (2016). arXiv preprint arXiv:1603.09423
30. Tang, Y., Wu, X.: Scene text detection and segmentation based on cascaded convolution neural networks. IEEE Trans. Image Process. **26**(3), 1509–1520 (2017)
31. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
32. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
33. Jiang, Y., Zhu, X., Wang, X., Yang, S., Li, W., Wang, H., Fu, P., Luo, Z.: R2CNN: rotational region CNN for orientation robust scene text detection (2017). arXiv preprint arXiv:1706.09579
34. Ma, J., Shao, W., Ye, H., Wang, L., Wang, H., Zheng, Y., Xue, X.: Arbitrary-oriented scene text detection via rotation proposals. IEEE Trans. Multimed. **20**, 3111–3122 (2018)
35. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37 (2016)
36. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
37. Liu, Y., Jin, L.: Deep matching prior network: toward tighter multi-oriented text detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3454–3461 (2017)
38. Shi, B., Bai, X., Belongie, S.: Detecting oriented text in natural images by linking segments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
39. Liao, M., Shi, B., Bai, X., Wang, X., Liu, W.: TextBoxes: a fast text detector with a single deep neural network. In: AAAI, pp. 4161–4167 (2017)
40. Liao, Minghui, Shi, Baoguang, Bai, Xiang, and and: TextBoxes++: A Single-Shot Oriented Scene Text Detector. IEEE Transactions on Image Processing **27**(8), 3676–3690 (2018)
41. He, W., Zhang, X.Y., Yin, F., Liu, C.L.: Deep direct regression for multi-oriented scene text detection. In: Proceedings of the IEEE International Conference on Computer Vision (2017)
42. Adams Jr., R.B., Adams, R.B., Ambady, N., Shimojo, S., Nakayama, K. (eds.): The Science of Social Vision: The Science of Social Vision, vol. 7. Oxford University Press, Oxford (2011)
43. Yao, C., Bai, X., Sang, N., Zhou, X., Zhou, S., Cao, Z.: Scene text detection via holistic, multi-channel prediction (2016). arXiv preprint arXiv:1606.09002
44. Xu, Y., Wang, Y., Zhou, W., Wang, Y., Yang, Z., Bai, X.: TextField: learning a deep direction field for irregular scene text detection. Trans. Image Process. (2019)
45. Lyu, P., Liao, M., Yao, C., Wu, W., Bai, X.: Mask textspotter: an end-to-end trainable neural network for spotting text with arbitrary shapes. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 67–83 (2018)
46. Xue, C., Lu, S., Zhang, W.: MSR: multi-scale shape regression for scene text detection (2019). arXiv preprint arXiv:1901.02596
47. Castrejón, L., Kundu, K., Urtasun, R., Fidler, S.: Annotating object instances with a polygon-RNN. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, p. 2 (2017)
48. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K.: Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 4 (2017)
49. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. AAAI **4**, 12 (2017)
50. Liu, X., Liang, D., Yan, S., Chen, D., Qiao, Y., Yan, J.: Fots: fast oriented text spotting with a unified network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5676–5685 (2018)
51. Sun, Y., Zhang, C., Huang, Z., Liu, J., Han, J., Ding, E.: TextNet: irregular text reading from images with an end-to-end trainable network. In: Asian Conference on Computer Vision (2018)
52. Long, S., Ruan, J., Zhang, W., He, X., Wu, W., Yao, C.: Textsnake: a flexible representation for detecting text of arbitrary shapes In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 20–36 (2018)
53. Dai, Y., Huang, Z., Gao, Y., Xu, Y., Chen, K., Guo, J., Qiu, W.: Fused text segmentation networks for multi-oriented scene text detection. In: Proceedings of the IEEE International Conference on Pattern Recognition (ICPR), pp. 3604–3609 (2018)
54. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2961–2969 (2018)