HyText – a Scene-Text Extraction Method for Video Retrieval

Alexander Theus[0000−0003−3633−5243], Luca Rossetto[0000−0002−5389−9465], and Abraham Bernstein[0000−0002−0128−4602]

Department of Informatics, University of Zurich, Zurich, Switzerland

Abstract. Scene-text has been shown to be an effective query target for video retrieval applications in a known-item search context. While much progress has been made in scene-text extraction from individual pictures, the special case of video has so far received less attention. This paper introduces HyText, a scene-text extraction method for video with a focus on retrieval applications. HyText uses intermittent scene-text detection in combination with bi-directional tracking in order to increase throughput without reducing detection accuracy.

Keywords: Scene-Text Extraction · Video Text Extraction · Video Retrieval.

1 Introduction

As multimedia collections grow larger in terms of size, heterogeneity, and variety of media types, the quest for accessing the knowledge contained within them becomes more onerous. The traditional approach of manually annotating media objects, and retrieving them at a later point based on this metadata has various deficiencies. Firstly, the sheer size of media collections and their progressive growth renders prior annotation unfeasible. Secondly, textual descriptions tend to be subjective due to language, culture, expertise, and personal experience. While several means of querying for video have been proposed, querying for text, which is visible in a video sequence, enables effective and lossless query expression, particularly in a known-item search scenario [15]. This not only works for artificial text overlays but also for text which is an inherent part of the scene.

This type of naturally occurring text is termed *scene-text*. Due to the abundance of textual content in natural scenes and especially in urban settings [13] the task of scene-text extraction (STE) has received a lot of interest from the computer vision community, and has been propelled by advances in deep learning. While STE in still images remains a problem, several approaches have emerged which show a sufficiently high performance to be useful for general purpose applications. STE in videos, however, is still an under-explored field, even though it is of similar importance.

In this paper, we introduce HyText, a scene-text extraction method for video. Rather than applying a costly scene-text detection step on every frame, HyText

uses intermittent detection combined with bi-directional tracking to localize and transcribe text instances in video. This not only substantially reduces end-to-end inference time, it also ensures that reoccurring text instances do not lead to duplicated transcriptions.

The remainder of this paper is structured as follows: Section 2 provides an overview of related scene-text detection, recognition and extraction methods. The proposed method is detailed in Section 3 and its effectiveness is evaluated in Section 4. Section 5 then offers some concluding remarks.

2 Related Work

Current approaches for scene-text extraction primarily focus on individual images. Those methods can be roughly divided into two groups: two-stage methods and end-to-end methods. Two-stage methods use two independent components for scene-text detection and subsequent scene-text recognition, while end-to-end methods have no such clear component division.

One of the more popular methods for scene-text detection is the 'Efficient and Accurate Scene Text Detector' (EAST) [22], which uses a convolutional neural network to predict the position of words in natural scenes. Other methods such as 'Character Region Awareness For Text detection' (CRAFT) [1] even perform such predictions for individual characters.

State-of-the-art methods for scene-text recognition can be divided into segmentation-based and segmentation-free methods. Segmentation-based methods are architecturally suitable for two-dimensional prediction problems. Taking advantage of this, Liao et al. [11] introduce a segmentation-based scene-text recognizer that regards scene-text as a two-dimensional spatial distribution of features. The 'Convolutional Recurrent Neural Network' (CRNN) [19] was one of the early adopters of the segmentation-free method. Even though this method is rather old, it is still widely used as a basis for various methods such as [5,10,12]. However, this method has its limitations when it comes to irregular text.

Comparatively less work has been done on methods, which explicitly focus on video. A method introduced by Wang et al. [20] approached STE in video by first detecting and recognizing text in each frame via a jointly-trained STE method. After extracting text in each individual frame, the method uses a similarity function comprised of the position of the text instance, the recognized text as well as the frame offset to match equivalent text instances. The method then uses majority voting to determine the most likely text for the text stream. Unfortunately, the paper does not mention how they deal with scenarios in which there is no singular majority string.

The aforementioned method has been criticized by Cheng et al. [3]. They argue that reading text in every frame is excessively computationally costly and thus operationally unsuitable. Moreover, recognizing text in every frame also introduces erroneous results due to the abundance of *low quality* (e.g., due to blurring, rotation, perspective distortion, poor illumination, etc.) text regions. To circumvent these problems, Cheng et al. introduce a method called 'You Only

Recognize Once' (YORO), which, as the name already suggests, only recognizes text in one region from a text stream by selecting the most *high quality* one. YORO, like the previously introduced method, follows the tracking-by-detection framework by detecting text in every frame. The detection module is based on the EAST text detector, and its score output map is temporally enhanced via a spatial-temporal aggregation strategy that optimizes the output in the current frame by referring to the output in adjacent frames. Then, a module they call 'Text Recommender' is responsible for three tasks: text quality scoring, text tracking, and text recognition. All of these tasks depend on the same feature map produced by a ResNet-based feature extractor. The text quality scoring is performed on each detected text region and computes scores between 0 and 1, where 1 denotes the highest quality achievable. The text tracking is achieved by comparing the L2 normalized feature maps, and optimal matches are determined via the use of the Hungarian Algorithm [8]. At this point, text streams have been generated, and each text region within the stream has a quality score. The region with the highest quality is selected and an attention-based recognition module produces the final string.

3 HyText

Similar to Cheng et al. [3], we argue that recognizing text in each frame is too computationally expensive and might introduce erroneous recognitions. However, in contrast to Cheng et al., we argue that the computational cost of STE in video is mainly driven by the detection module. This is confirmed by the results presented in Section 4. Thus, unless the video has an unusually high amount of text instances per frame, the detection module is the main driver in computational cost. Moreover, recognized text is a very salient component of a text stream. It is therefore well suited to validate the internal consistency of a text stream, and for associations to other streams. To take advantage of this observation, we introduce the **Hy**brid Bidirectional **T**ext **Ext**ractor (HyText), which can reliably extract scene-text from videos by only detecting text in a subset of frames. Moreover, it can utilize recognized text to validate streams for inter-frame association, without having to rely on per-frame recognition. The following details the three primary components of HyText: text stream formation, text stream recognition, and text stream aggregation.

3.1 Text Stream Formation

A text stream consists of a single text instance which is tracked across a series of consecutive frames. While conventional single-object tracking with a Bayesian framework or template matching cannot handle re-initialization in a multi-object scene, it has a few important advantages over tracking-by-detection. For one, since the relative region of the text in the previous frame is known, these methods seem to be substantially faster than object detection. Moreover, they can handle occlusion a lot better since they are able to estimate the probable region

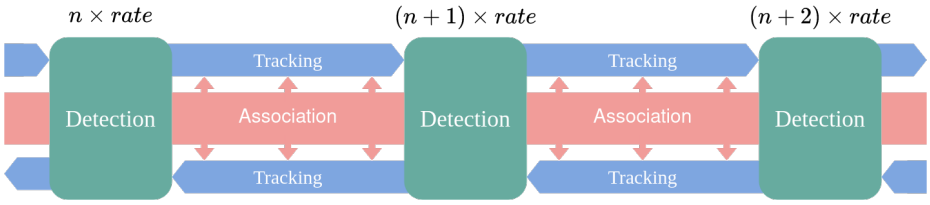

Fig. 1. Abstract illustration of a snippet of the text stream formation procedure of HyText. The labels above the detection box describe the index of the frame at which it was performed.

of the text. Thus, HyText aims to take advantage of the speed and superior occlusion treatment, while still being able to take into account newly appearing text instances. HyText accomplishes this by allowing the user to define a rate at which the detections should occur. For example, if the user of the method specifies the rate to be two, then a detection will be done at every second frame. In order to not miss text occurrences between the frames at which detections were done, HyText uses the discriminative correlation filter method named CSR-DCF [16] to track text instances in the frames at which no detection was performed.

Figure 1 provides an abstract illustration of the text stream formation procedure. First, text is detected in frame $n \times rate$, where $n \in \mathbb{N}$ and $rate$ is the rate at which detection should occur. Then, the detected text instances are tracked via CSR-DCF just before frame $(n+1) \times rate$ is reached. Subsequently, text is detected in frame $(n+1) \times rate$, and tracked backwards until frame $n \times rate$. These tracked text instances are likely going to describe the same text instance. Since bi-directional tracking allows for intra-frame rather than inter-frame association – which is conventional for multi-object tracking – the temporal aspect of association is removed, which radically simplifies the similarity function. In order to find out which text streams describe the same text instance, the average Intersection over Union (IoU) is computed, and the Hungarian Algorithm is applied to find optimal matches. If the average IoU of the optimal match is below a threshold (empirically set to 0.6) the tracked text instances between frame $n \times rate$ and $(n+1) \times rate$ are removed, and the text streams are regarded as separate. If the average IoU is above the threshold, then the text streams are merged, and the average bounding boxes of the tracked text is used. This procedure is then continued for $n \in \{0, 1, 2, \ldots, m \mid m = \lceil \frac{V}{rate} \rceil - 1\}$, where V is the total number of frames in a video. Thus, instead of performing detections on all frames, HyText only applies detection on $\lceil \frac{V}{rate} \rceil$ frames, and applies the much faster CSR-DCF tracker for the remaining ones, which account for $V - \lceil \frac{V}{rate} \rceil$ frames.

3.2 Text Stream Recognition

Once the previously described procedure is complete, we will have a list of text streams describing individual text instances. The next step is to extract the cor-

Fig. 2. Example of the disappearing text phenomenon where quality scoring and unfiltered majority polling produce incorrect results. Image taken from Text in Videos dataset.

rect string from the streams. In order to do this, we introduce a hybrid approach that does neither entirely depend on majority voting, nor on text selection such as in YORO [3]. We argue that unfiltered majority voting may introduce flawed text recognitions, which can be easily filtered out. Moreover, the text scoring system in YORO may reliably select the highest quality bounding box for recognition. However, it neglects the most radical deteriorating factor: disappearing text. Cut text is a particularly abundant characteristic in video, since a text instance may be in the center of the image in one frame, but then, as the camera moves, progressively disappear. In the process, the text gets cut increasingly, and if that text is used for recognition, the recognition will necessarily be incorrect (or incomplete). This phenomenon is exemplified in Figure 2.

In order to address this, we calculate the aspect ratio of each bounding box within a stream and prune those whose aspect ratio is below the median. With this procedure, we effectively mitigate the problem of disappearing text and simultaneously reduce computational cost. For the remaining bounding boxes, the recognition is performed, and the final string is obtained via majority voting. However, there may be cases in which two or more strings appear equally often in the stream. As a means to address this, we prune streams that cannot agree on one or two strings, as they may be indicative of unreliability. If two strings remain after the majority voting procedure, we apply a global pairwise sequence alignment algorithm called *Needleman-Wunsch* [18]. The Needleman-Wunsch algorithm is typically used in bioinformatics to align protein or nucleotide sequences. In our case, we use the algorithm to extract a common substring from the two recognitions. While this may not result in a completely accurate final recognition, it is still of value in a retrieval application, since scene-text search will commonly not look for complete matches, but for relative matches using a string edit distance.

3.3 Text Stream Aggregation

With the aforementioned steps completed, the STE task for video could be regarded as concluded. However, the procedure so far can still be improved to handle text instances that disappear and reappear again. If a text instance disappears at a time at which a detection is performed, and then reappears later, two text streams will be formed for the same text instance, which results

in unwanted duplicity. This can also occur in the case that the detector module simply does not detect a text instance in a certain frame, e.g. due to high motion blur. As a means to overcome this problem, we introduce an additional step that is heavily inspired by the method introduced by Wang et al. [20]. We start with text streams that have the minimal distance between them, which in all cases would be the rate that the user specified. The distance between two text streams is regarded as the difference between the tail and the head of a stream (see Eq. 5). To check whether or not they actually describe the same text instance, the following similarity measure is applied:

$$S_{ab} = k_1 \times S_{dis}^{spatial} + k_2 \times S_{op}^{area} + k_3 \times S_{winkler}^{jaro} + k_4 \times S_{offset}^{frame} \tag{1}$$

where the S_y^x are defined as below and k_1, k_2, k_3, k_4, are weights, empirically set to 1, 1, 10, and 0.2, respectively. S_{ab} stands for the cost of merging two streams a and b, where stream b appears after stream a.

$S_{dis}^{spatial}$ refers to the spatial distance between the bounding boxes and is calculated like the following:

$$S_{dis}^{spatial} = \frac{mean(x_i^a - x_i^b)}{max(W_a, W_b)} + \frac{mean(y_i^a - y_i^b)}{max(H_a, H_b)} \tag{2}$$

Where $(x_i, y_i), i \in \{1, 2, 3, 4\}$ are the coordinates of the quadrilateral bounding box. W and H refer to the width and the height of the boxes, respectively.
S_{op}^{area} refers to the inverse IoU to capture the overlap between the two boxes:

$$S_{op}^{area} = 1 - \frac{Area(a \cap b)}{Area(a \cup b)} \tag{3}$$

$S_{winkler}^{jaro}$ is the inverse Jaro Winkler Distance [21] between the two strings. This metric is particularly informative, hence it is given the highest weight of 10.

$$S_{winkler}^{jaro} = 1 - JW(str_a, str_b) \tag{4}$$

Finally, the distance between the two frames, or in other words the frame offset S_{offset}^{frame}, is also an important characteristic to consider. To include it, we simply take the difference between the head and the tail of the two streams.

$$S_{offset}^{frame} = head_b - tail_a \tag{5}$$

The optimal match between the streams is calculated via the Hungarian Algorithm. If the final score S_{ab} is below the empirically determined threshold of 8, then the streams will be combined and regarded as one. The recognition of the stream which spans the most frames, will be regarded as the recognition for the new combined stream. This procedure will be continued to be done for $distance \in \{rate, 2{\times}rate, 3{\times}rate, \ldots, n{\times}rate \mid (n+1){\times}rate \geq \frac{threshold}{k_4}\}$. Finally, HyText prunes very short text streams that appeared for fewer than 10 consecutive frames. This is done to remove unreliable text streams, and to prune those, which appeared for only a small amount of time. In such cases, the user would probably not have been able read them, which makes such stream uninteresting for a retrieval application.

4 Evaluation

The following shows comparative evaluations of isolated scene-text detection (STD) and recognition (STR) methods as well as the complete HyText pipeline in different configurations. All runtime measurements were performed using an Intel core i7-10700KF and 32GB DDR4-3600 memory. To minimize implementation-specific effects, no GPU acceleration was used, since it was not supported by all available implementations.

4.1 Scene-Text Detection

First, we evaluate several state-of-the-art scene-text detection methods to be able to compare their accuracy and runtime performance on the same data. EAST[1] [22], PixelLink[2] [4], CRAFT[3] [1], and the detection module of EasyOCR[4] are evaluated according to two datasets that carry distinct characteristics. The novel tightness-aware intersection-over-union metric [14] will be used to calculate the correspondence between ground truth bounding boxes and the detected ones. Furthermore, the inference time is captured to put the computed accuracy in perspective.

Two distinct datasets called ICDAR2015 [7] and Text in Videos[5] were chosen to evaluate the STD methods. ICDAR2015 is a dataset based on still images. In contrast to other mainstream detection datasets, ICDAR2015 is characterized by its particular presence of blurred and low-resolution text. Text in Videos is, as the name already suggests, a dataset based on video footage. It contains 15 videos which all in all make up almost 10 000 frames. In order to adapt the dataset to something more manageable for frame-by-frame detection, random frames from each video were selected to create a new and more manageable dataset which is comprised of 538 images. These two datasets were specifically chosen to simulate the detection of scene-text in visual multimedia. Moreover, the groundtruth of both datasets are quadrilateral bounding boxes, which is particularly important in this case since the output of the chosen STD methods are also quadrilateral bounding boxes. If the ground-truth were rectangular boxes or rotated rectangular boxes, the increased tightness of quadrilateral bounding boxes could not be captured. Last but not least, the dataset Text in Videos also serves the function of capturing the generalizability of the STD methods since all of them were trained on ICDAR2015.

Table 1 shows the results with respect to both accuracy and inference time. CRAFT is the most accurate of the compared methods but also the slowest. EAST however, while being by far the fastest of the compared methods still demonstrates only slightly worse accuracy. PixelLink shows comparable accuracy to EAST but with substantially higher inference times while the detection

[1] https://github.com/argman/EAST
[2] https://github.com/ZJULearning/pixel_link
[3] https://github.com/clovaai/CRAFT-pytorch
[4] https://github.com/JaidedAI/EasyOCR
[5] https://rrc.cvc.uab.es/

module of EasyOCR shows the worst accuracy of all compared methods. For other STE methods for video, such as YORO, which apply detection to every frame, the trade-off between accuracy and inference time would probably lead them to use EAST in most circumstances, due to its high frame rate, despite some loss in accuracy. HyText will however be less affected in this case, since it only needs to perform detection on a subset of frames. For this reason, HyText will not only be evaluated with EAST, but also with CRAFT to see whether or not increasing the rate can allow for computationally extensive detectors such as CRAFT to be used for STE in videos.

Table 1. Overview of weighted average STD results. The results on ICDAR2015 have the weight of 0.75, and results on Text in Videos of 0.25. This is done because ICDAR2015 is a more reliable and commonly used dataset for text detection.

Name	FPS (Rank)	TIoU mean (Rank)	IoU mean (Rank)
EAST	2.84 (**1**)	52.7% (2)	76.0% (3)
EasyOCR	1.88 (2)	23.6% (4)	36.73% (4)
PixelLink	1.14 (3)	52.6% (3)	76.2% (2)
CRAFT	0.58 (4)	56.6% (**1**)	79.3% (**1**)

4.2 Scene-Text Recognition

In order to find a suitable STR component for our pipeline, we compare the performance of several methods proposed over the last few years. Specifically, we compare CRNN [19], TPS-ResNet-BiLSTM-CTC (TRBC), TPS-ResNet-BiLSTM-Attn (TRBA), SARN [9], CSTR [2] and the recognition module of EasyOCR (based on CRNN). All these methods except for the EasyOCR recognition module are trained on Synthtext [6] to allow for a fair comparison. EasyOCR does unfortunately not share how and with which dataset their recognition module was trained on, nor does it give the possibility to train the module oneself.

In order to evaluate the performance of STR methods, the widely used word recognition accuracy (WRA) is used which is defined as:

$$WRA = \frac{W_r}{W} \tag{6}$$

where W_r is the number of correctly recognized words and W the number of total words. Inspired by how Liu et al. [14] view the correspondence between detection and ground-truth not as a binary matter (*matching* or *not-matching*), but as a spectrum which can range from *matching* to *not-matching*, we extend the mainstream WRA metric to be tightness-aware. Consequently, instead of classifying a recognition as being "correct" or "incorrect", the Jaro-Winkler Distance [21] will be used to measure the correspondence between the predicted and the ground-truth string. This new metric will be termed Tightness-aware Word Recognition

Accuracy ($TWRA$) and is defined as

$$TWRA = \frac{1}{N}\sum_{i=1}^{N} JW(s_{target_i}, s_{pred_i}) \tag{7}$$

where s_{target} is the ground-truth string and s_{pred} the predicted string. JW is short for Jaro-Winkler Distance, which captures the normalized similarity between strings where 1 describes the case in which s_{target} and s_{pred} are the same and 0 the case in which they are not matching at all. Furthermore, the inference time will be captured.

The datasets used to evaluate the STR methods are ICDAR2015 [7], IIIT5K [17], and Text in Videos. They were specifically chosen because each of them carries distinct characteristics that can affect the performance of STR in visual multimedia. Namely, the images in ICDAR2015 and Text in Videos are often heavily blurred, which is a particular characteristic of natural images and especially frames in videos. IIIT5K is a dataset with more regular text. However, even though the images are rarely blurred, the fonts are often very peculiar, which is also a characteristic of natural images. For example brand names often appear in natural images, and brands often do not choose regular fonts. The company symbol for "Coca Cola" is a primary example of that. Text in Videos also distinguishes itself from the other two datasets by being based on videos.

Table 2. Overview of the weighted average STR results. Results on ICDAR2015 and IIIT5K have weights two times as high as Text in Videos, because the former two are more commonly used and reliable datasets for STR.

Method	FPS (Rank)	WRA (Rank)	TWRA (Rank)
CRNN[6]	158.3 (**1**)	54.4% (5)	81.3% (5)
EasyOCR[4]	93.79 (2)	30.4% (6)	65.0% (6)
TRBC[6]	35.56 (3)	60.7% (4)	84.1% (4)
TRBA[7]	26.25 (4)	66.6% (2)	85.8% (3)
SARN[7]	10.32 (5)	67.9% (**1**)	87.3% (**1**)
CSTR[7]	5.6 (6)	65.9% (3)	86.7% (2)

Table 2 again shows the results of the compared methods for both accuracy and frame rate. CRNN is by far the fastest method, its accuracy is however comparatively low. In contrast, the most accurate of the compared methods SARN has a roughly 15 times lower frame rate. TRBA, while being almost as accurate as SARN is more than twice as fast. Because the inference time of STR is not a primary concern, as most methods have a frame rate which is an order of magnitude higher than any STR method, we select TRBA for the evaluation of the HyText pipeline, since it provides results comparable with the most accurate method in a reasonable time.

[6] https://github.com/clovaai/deep-text-recognition-benchmark
[7] https://github.com/Media-Smart/vedastr

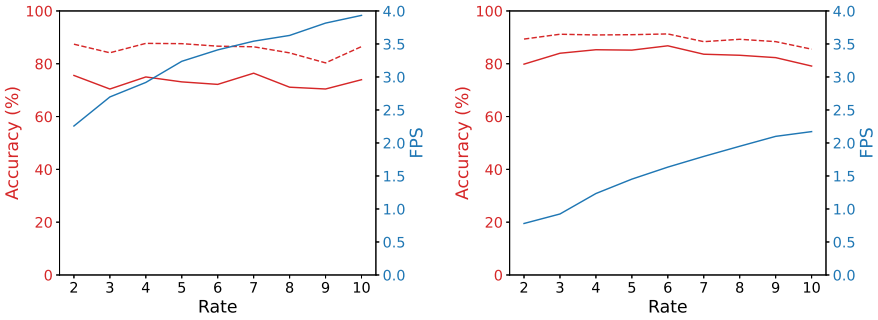

Fig. 3. Accuracy and frame rate for the full HyText pipeline with a detection rate from 2 to 10. The left plot shows results for EAST-TRBA, the right plot shows results for CRAFT-TRBA. The solid line shows WRA, the dotted line shows TWRA.

4.3 HyText

We evaluate the full HyText pipeline, sweeping the detection rate from 2 to 10 for both EAST and CRAFT for STD and TRBA for STR. Accuracy is measured using the F-Score for both WRA and TWRA. For WRA, precision (WRA_p) and recall (WRA_r) are calculated as follows:

$$WRA_p(G, D) = \frac{\sum_{j=1}^{|D|} match_D(D_j)}{|D|} \tag{8}$$

$$WRA_r(G, D) = \frac{\sum_{i=1}^{|G|} match_G(G_i)}{|G|} \tag{9}$$

where G and D represent the ground-truth and prediction set respectively. The *match* operation refers to a complete match between ground-truth and prediction. In contrast, TWRA uses the following definitions:

$$TWRA_p(G, D) = \frac{\sum_{j=1}^{|D|} match^*_D(D_j)}{|D|} \tag{10}$$

$$TWRA_r(G, D) = \frac{\sum_{i=1}^{|G|} match^*_G(G_i)}{|G|} \tag{11}$$

where $match^*$ refers to the optimal similarity matches between two sets of strings, the Jaro-Winkler Distance is used to capture the normalized similarity between two strings, and the Hungarian Algorithm to find optimal matches.

The resulting values for WRA and TWRA as well as the frame rate of the full pipeline are shown in Figure 3.

It can be seen that the accuracy remains consistently high throughout the entire range for the tested detection rate. Decreasing the detection rate from one in 2 frames to one in 10 frames leads to an increase in end-to-end frame

rate of 174.3% when using EAST and 278.6% when using CRAFT for STD. This demonstrates that HyText can substantially increase throughput without sacrificing accuracy. The inference time for other STE methods for videos are bottlenecked by their STD module because they rely on per-frame detection. HyText achieves an end-to-end FPS score that is 375% higher than when using CRAFT for STD alone.

5 Conclusion

In this paper, we introduced HyText, a hybrid bi-directional scene-text extraction method for video. Its intermittent use of scene-text detection and bi-directional tracking can substantially increase throughput without reducing result accuracy. HyText is especially efficient in videos with a low text density, since the time requirements for the tracking component scale with the number of detected text instances. This is in contrast to other methods which apply scene-text detection regularly, since the time required for the detection process does not directly scale with the amount of text present. HyText is agnostic towards the components used for STD and STR and can therefore be easily tailored to a wide range of use cases. The evaluation showed that inference rate can be more than doubled by adjusting the rate of scene-text detection without a noticeable decrease in accuracy.

References

1. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 9357–9366 (2019). https://doi.org/10.1109/CVPR.2019.00959
2. Cai, H., Sun, J., Xiong, Y.: Revisiting classification perspective on scene text recognition. arXiv preprint arXiv:2102.10884 (2021)
3. Cheng, Z., Lu, J., Xie, J., Niu, Y., Pu, S., Wu, F.: Efficient video scene text spotting: Unifying detection, tracking, and recognition. CoRR **abs/1903.03299** (2019), http://arxiv.org/abs/1903.03299
4. Deng, D., Liu, H., Li, X., Cai, D.: Pixellink: Detecting scene text via instance segmentation. CoRR (01 2018)
5. Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q., Wang, H.: Pp-ocr: A practical ultra lightweight ocr system. ArXiv (2020)
6. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2315–2324 (2016). https://doi.org/10.1109/CVPR.2016.254
7. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., Shafait, F., Uchida, S., Valveny, E.: Icdar 2015 competition on robust reading. 2015 13th International Conference on Document Analysis and Recognition (ICDAR) pp. 1156–1160 (2015). https://doi.org/10.1109/ICDAR.2015.7333942
8. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)

9. Lee, J., Park, S., Baek, J., Oh, S.J., Kim, S., Lee, H.: On recognizing texts of arbitrary shapes with 2d self-attention. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 2326–2335 (2019). https://doi.org/10.1109/CVPRW50498.2020.00281
10. Liao, M., Shi, B., Bai, X.: Textboxes++: A single-shot oriented scene text detector. IEEE Transactions on Image Processing **27**(8), 3676–3690 (2018). https://doi.org/10.1109/TIP.2018.2825107
11. Liao, M., Zhang, J., Wan, Z., Xie, F., Liang, J., Lyu, P., Yao, C., Bai, X.: Scene Text Recognition from Two-Dimensional Perspective. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 8714–8721 (2019). https://doi.org/10.1609/aaai.v33i01.33018714
12. Liao, M., Shi, B., Bai, X., Wang, X., Liu, W.: Textboxes: A fast text detector with a single deep neural network (11 2016)
13. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014), http://arxiv.org/abs/1405.0312
14. Liu, Y., Jin, L., Xie, Z., Luo, C., Zhang, S., Xie, L.: Tightness-aware evaluation protocol for scene text detection. CoRR (03 2019)
15. Lokoč, J., Veselý, P., Mejzlík, F., Kovalčík, G., Souček, T., Rossetto, L., Schoeffmann, K., Bailer, W., Gurrin, C., Sauter, L., Song, J., Vrochidis, S., Wu, J., Jónsson, B.t.: Is the reign of interactive search eternal? findings from the video browser showdown 2020. ACM Transactions on Multimedia Computing, Communications, and Applications **17**(3) (Jul 2021). https://doi.org/10.1145/3445031
16. Lukezic, A., Vojír, T., Cehovin, L., Matas, J., Kristan, M.: Discriminative correlation filter with channel and spatial reliability. CoRR **abs/1611.08461** (2016), http://arxiv.org/abs/1611.08461
17. Mishra, A., Alahari, K., Jawahar, C.V.: Scene text recognition using higher order language priors. In: BMVC (2012)
18. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48**(3), 443–453 (1970). https://doi.org/10.1016/0022-2836(70)90057-4
19. Shi, B., Bai, X., Yao, C.: An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2015). https://doi.org/10.1109/TPAMI.2016.2646371
20. Wang, X., Jiang, Y., Yang, S., Zhu, X., Li, W., Fu, P., Wang, H., Luo, Z.: End-to-end scene text recognition in videos based on multi frame tracking. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 01, pp. 1255–1260 (2017). https://doi.org/10.1109/ICDAR.2017.207
21. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. (1990)
22. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: An efficient and accurate scene text detector. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2642–2651 (2017). https://doi.org/10.1109/CVPR.2017.283