A method for detecting text of arbitrary shapes in natural scenes that improves text spotting

Qitong Wang, Yi Zheng, and Margrit Betke
Boston University
Boston, MA 02215
{wqt1996, yizheng, betke}@bu.edu

Abstract

Understanding the meaning of text in images of natural scenes like highway signs or store front emblems is particularly challenging if the text is foreshortened in the image or the letters are artistically distorted. We introduce a pipeline-based text spotting framework that can both detect and recognize text in various fonts, shapes, and orientations in natural scene images with complicated backgrounds. The main contribution of our work is the text detection component, which we call UHT, *short for UNet, Heatmap, and Textfill. UHT uses a UNet to compute heatmaps for candidate text regions and a textfill algorithm to produce tight polygonal boundaries around each word in the candidate text. Our method trains the UNet with groundtruth heatmaps that we obtain from text bounding polygons provided by groundtruth annotations. Our text spotting framework, called UHTA, combines UHT with the state-of-the-art text recognition system ASTER. Experiments on four challenging and public scene-text-detection datasets (Total-Text, SCUT-CTW1500, MSRA-TD500, and COCO-Text) show the effectiveness and generalization ability of UHT in detecting not only multilingual (potentially rotated) straight but also curved text in scripts of multiple languages. Our experimental results of UHTA on the Total-Text dataset show that UHTA outperforms four state-of-the-art text spotting frameworks by at least 9.1 percent points in the F-measure, which suggests that UHTA may be used as a complete text detection and recognition system in real applications.*

1. Introduction

Scene text detection is an important task in computer vision with application significance such as helping people with visual impairments to understand text in images (e.g., of medicine bottles or supermarket shelves) or helping self-driving cars understand the meaning of traffic and street signs. Building computer vision systems that can detect text is not easy due to the variety of sizes, fonts, styles, sizes, and orientations in which text can occur in natural scene images and their often complex backgrounds (e.g., Fig. 1).

Figure 1. Polygonal text annotations of curved text in images are often so imprecise (bottom left) that heat maps (bottom right), computed as an intermediate step to interpret the text, yield inaccurate results. The proposed UHT method computes and interprets deep learned heat maps (top right) that result in much more accurate polygonal text outlines (top left), which in turn yield better text recognition results.

In the past few years, computer vision researchers have developed methods that identify oriented straight text in natural scene images accurately [14, 26, 45, 47, 48] ("oriented" means not necessarily aligned with the image rows). More recently, detection of arbitrarily-shaped text, such as curved or deformed text, has received attention from computer vision researchers, not only because detecting such text is

more challenging than oriented straight text, but also because it commonly appears in daily life. For arbitrarily-shaped text detection, for example, a weakly supervised learning algorithm [7] was recently proposed to extract character-based pseudo ground truth to help deep learning models effectively extract each character of such a text in a natural scene image. Whether it is detection of oriented straight or curved text, we found that most state-of-the-art methods rely on multiple deep-learned geometric properties of the text, such as angle attributes [24, 48] or text center line regions [46]. Others use multiple output models [46] to produce high evaluation scores on widely-used benchmarks. While state-of-the-art text detection methods can solve many challenging problems with these techniques, as far as we know, there is no method that simply and effectively uses a "text region feature map," even when given a variety of text shapes, sizes, and lengths. Moreover, many words are located so close to each other in the images that detection methods do not separate them correctly but grouped into one consecutive-word text region. These challenges make relying on only the text region feature map to effectively detect text in natural scene images seemingly impossible. But is it really impossible to accurately detect text using only one text region feature map in the scene text detection field? Our work shows that the answer is no. Using only one channel, the text region feature map, our method effectively detects text in images of natural scenes. With the help of new pre-processing and post-processing algorithms, we make accurately detecting text in images possible, relying on a relatively small amount of geometric information.

The contributions of our research work are five-fold:

• We propose a new text detection framework, called UHT, that outputs only one text region heatmap channel. UHT can solve challenging problems in the field of scene text detection, such as accurately detecting and separating multiple text regions that "stick" together.

• We propose a new text region feature map representation, which here is a special kind of heat map (Fig. 1), that enables UHT to detect text in natural scene image.

• We propose a new algorithm called the *Textfill Algorithm* that can accurately extract multi-vertex bounding polygons that tightly define the outline of each word in the scene text region.

• UHT obtained evaluation scores that are higher than most of state-of-the-art scene text detection methods when fine-tuned on specific benchmark datasets. UHT outperforms all state-of-the-art methods in its generalization ability, as shown in one of the experiments.

• "Spotting" text in images means detecting and recognizing it. We introduce a complete pipeline-based text spotting system, called UHTA, showing that our UHT can be used for text spotting as long as an effective text recognition model is given.

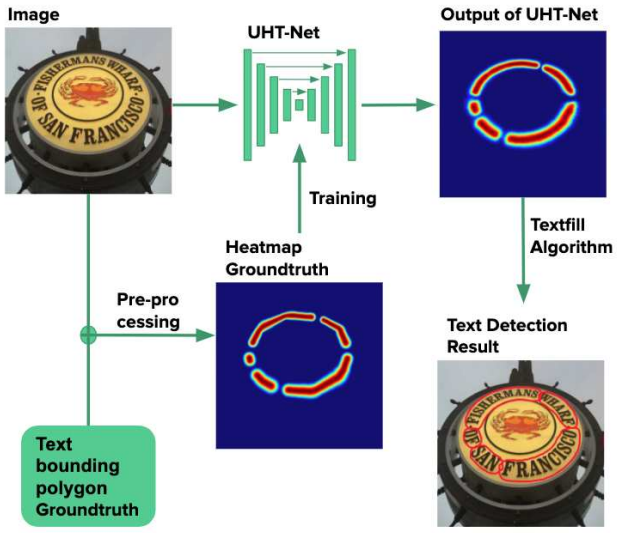

Figure 2. Pipeline of UHT. The process of text detection of UHT can be divided into three steps: 1) Pre-processing is used to generate a heatmap text region ground truth, which is used as a training label of UHT-Net. 2) A trained UHT-Net can output predicted text region heatmaps. 3) In the post-processing step, the Textfill algorithm outputs the final predicted text bounding polygons interpreting the outputs of the UHT-Net.

Our code is available at http://www.cs.bu.edu/faculty/betke/UHT.

2. Related Works

The task of detecting text in images of everyday scenes, also known as "Scene Text Detection" is attracting more and more attention from researchers in the computer vision field. Initially, researchers focused on detecting oriented straight text in scene images [48, 10, 47, 45, 14]. However, detecting text with arbitrary shapes is more and more popular recently [24, 40, 41, 46, 7, 49, 42].

Before methodologies in deep learning field are widely used in text detection field, SWT [11] and MSER [27] were two eye-catching algorithms which had influenced many text detection methodologies. In recent years, modern methodologies, which make use of deep learning backbones, can be coarsely classified into two categories: regression-based methodologies and segmentation-based methodologies.

Regression-based methodologies are largely influenced by some popular general object detection frameworks such as Faster-RCNN [30]. TextBoxes [18] was inspired by SSD [20] and included "long" default boxes that had large aspect ratios to better detect text with different variation in natural scene images. In the text detection branch of Mask-TextSpotter [25], many text proposals were firstly generated by region proposal network to get text candidate boxes, then the RoI features of the text proposals were sent into the Fast

R-CNN module.

Segmentation-based methodologies are mainly inspired by FCN [23], The FCN classifies the image at the pixel level, thus solving the problem of image segmentation at the semantic level. In the text detection field, people see text regions in natural scene images as positive samples and background as negative samples. TextSnake [24] was proposed to detect text in the natural scene by predicting the text region and various geometry attributes of text to detect oriented straight and curve text effectively. Recently, instead of detecting whole text in images, CRAFT [7] was proposed to detect individual characters, connecting them to get each text bounding polygon. The proposed method provides the character region score and the character affinity score that, together, effectively cover various kinds of text shapes. In this method, a weakly-supervised framework was implemented to generate character-level pseudo annotations.

As we can see, state-of-the-art frameworks make full use of a large volume of geometric information to effectively detect text in natural scene images. Our methodology, however, is based on only using text region information to effectively extract text bounding polygons from images.

3. Methodology

The pipeline of our model is shown in Figure 2. We now introduce our methodology in detail.

3.1. Pre-processing: Heatmap Text Region Groundtruth Generation

Our method represents each word or set of words in an image as an arbitrary-length text skeleton surrounded by a fixed-width region whose pixels have values defined by their distance ("radius") to the skeleton (Figure 3e). This "heatmap" representation for text is sufficiently flexible to represent both straight and curved text.

The way we generate heatmaps was inspired by previous work [7, 28]. Instead of simply marking the pixels of the text region as 1 and the background pixels as 0 (e.g., [24, 48]), our method assigns a probability to each pixel position in the feature map, indicating the probability that this pixel belongs to the text region (Figure 3e). Naturally, the closer the pixel is to the center of the text, the closer the probability is to 1, and the farther the pixel is to the center of the text, the closer the probability is to 0.

Text Skeleton and Radius. Each annotated text polygon is defined by K vertices, where K is an even number. First, we use a skeleton to represent each polygon. We expand the original number of center points on the skeleton to $\sigma = K + (m-1) \times (K-2)$ points, where m is a positive integer (see Figure 3b and 3c for more details). In our experiments, m is set to 5. We then pair the

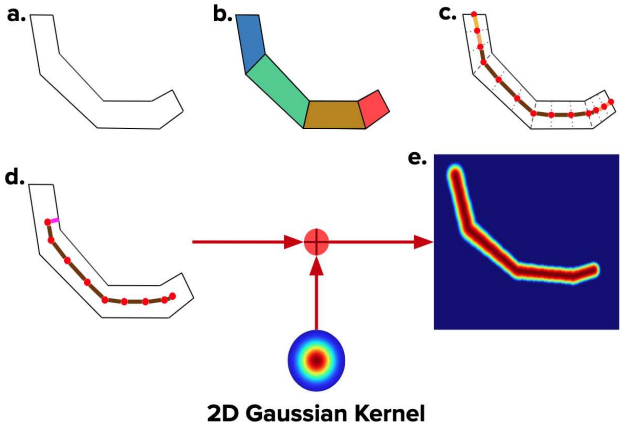

Figure 3. Process of creating the heatmap groundtruth. (a) Input: Text bounding polygon annotation, defined here by $K = 10$ vertices. (b) This polygon consists of $\frac{K-2}{2} = 4$ quadrilaterals (shown in different colors). (c) Each quadrilateral is divided into m equal parts, here $m = 3$. The Text Center Points (TCP) are marked as red dots, and the Text Skeleton (TS) is drawn in orange and coffee colors Using knowledge from mathematical geometry, we can get original text annotation points expanded to $\sigma = K + (m-1)(K-2)$ points, here 26. (d) To focus on the text center region, we delete the two ends of the TS (orange lines). This yields the final TS, here drawn in coffee color. The pink line exemplifies the radius R in our text representation method (Equation 2). (e) The final heatmap ground truth. The range of the 2D Gaussian kernel is set to $[0.0, 1.0]$.

coordinates of the upper part of the vertices of the polygon with the coordinates of the lower part of the vertices of the polygon. This yields the following pairs of points: $(P_{up}^{(1)}, P_{down}^{(1)}), (P_{up}^{(2)}, P_{down}^{(2)}), ..., (P_{up}^{(\frac{\sigma}{2})}, P_{down}^{(\frac{\sigma}{2})})$. For the ith pair of points, we compute the center points as follows:

$$P_{center}^{(i)} = \frac{P_{up}^{(i)} + P_{down}^{(i)}}{2}. \quad (1)$$

Then, $P_{center}^{(1)}, P_{center}^{(2)}, ..., P_{center}^{(\frac{\sigma}{2})}$ are defined as "Text Center Points (TCP)." The TCPs are essential for building the "Text Skeleton (TS)." We delete the two end groups of TCPs, so that the TCP set is changed from $\left\{P_{center}^{(1)}, P_{center}^{(2)}, ..., P_{center}^{(\frac{\sigma}{2})}\right\}$ to $\left\{P_{center}^{(3)}, P_{center}^{(4)}, ..., P_{center}^{(\frac{\sigma}{2}-2)}\right\}$. Connecting the center points in the TCP set, then we compute the final polygon skeleton. For each point in the TCP set, we also need their "Radius (R)." For the ith pair of points, $R^{(i)}$ is defined as follows:

$$R^{(i)} = \frac{dis(P_{center}^{(i)}, P_{up}^{(i)}) + dis(P_{center}^{(i)}, P_{down}^{(i)})}{2}, \quad (2)$$

where function $dis(A, B)$ is the Euclidean distance between A and B.

2D Gaussian Heatmap Ground Truth Representation. First we need to compute every point of TS using the Bresenham algorithm [1], which yields the "All Text Skeleton Points Set (ATSPS)". Given the set R, which is $\left\{R^{(3)}, R^{(4)}, ...R^{(\frac{\sigma}{2}-2)}\right\}$, and a 2D Gaussian kernel, we can compute the heatmap representation for each text bounding polygon. In addition, the range of the values of the generated Heatmap Text Region Groundtruth is set to $[0.0, 1.0]$. These are then used as training labels for the UHT-Net.

Since the generated heatmap groundtruth is the text skeleton convolved with several 2D Gaussian kernels, each text region is proportional to the length of its text skeleton. So due to our pre-processing algorithm, we suggest that UHT has the potential to be more accurate than other methods when detecting long text regions.

3.2. UHT-Net Architecture and Training Objectives

UHT-Net is a UNet-based [31] network that predicts score heatmaps of text regions. First, images are contracted to different feature maps. In the expanding process, our method employs either VGG-16 [34] or ResNet-50 [13] as backbone networks. Then these feature maps are gradually bilinearly expanded to the original size and mixed with the corresponding output of the previous stage in order to accurately detect text of different sizes.

UHT-Net uses an end-to-end training strategy. The definition of the loss function is

$$L = L_{reg} + \lambda_1 L_{center} + \lambda_2 L_{region}, \tag{3}$$

where λ_1 and λ_2 are both set to 1.0.

We define L_{reg} as the weighted MSE-Loss (because the ratio between positive and negative samples is unbalanced in the scene text detection datasets), which is defined for an input image χ to be:

$$\begin{aligned} L_{reg} = & \frac{\sum_{text}}{\sum_{text}+\sum_{BG}} \left(Y_{BG} - f_\theta(\chi_{BG})\right)^2 \\ & + \frac{\sum_{BG}}{\sum_{text}+\sum_{BG}} \left(Y_{text} - f_\theta(\chi_{text})\right)^2, \end{aligned} \tag{4}$$

where *text* denotes the positive pixels in the heatmap, *BG* denotes the negative pixels in the heatmap, $\sum_{text}$ denotes the total number of positive pixels in the heatmap, $\sum_{BG}$ denotes the total number of negative pixels in the heatmap, Y means pixels in the groundtruth heatmap generated by the pre-process, and $f_\theta(\chi)$ means pixels in the output of the UHT-Net, where θ are parameters in the UHT-Net.

The dice loss [36] for the *text center* and *text region* is denoted by L_{center} and L_{region} respectively. The *text center* is defined as the text region pixels in the output of the UHT-Net and generated groundtruth heatmap pixels that are higher than 0.9. The *text region* is defined as the text region pixels in the output of the UHT-Net and generated groundtruth heatmap pixels that are higher than 0.05, which are:

$$L_{center} = 1 - \frac{2\left|f_\theta(\chi_{center}) \bigcap Y_{center}\right|}{\left|f_\theta(\chi_{center})\right| + \left|Y_{center}\right|}, \tag{5}$$

$$L_{region} = 1 - \frac{2\left|f_\theta(\chi_{region}) \bigcap Y_{region}\right|}{\left|f_\theta(\chi_{region})\right| + \left|Y_{region}\right|}. \tag{6}$$

Algorithm 1 Textfill Algorithm

1: **Input:** Output heatmap H from UHT-Net; thresholds T_{top}, T_{end}.
2: **// Extract center points for each text region:**
3: Set pixel values in regions where heatmap H pixel values $> T_{top}$ to 1.0, otherwise to 0.0. These regions are defined as *CR*.
4: Find center points *CP* for each *CR*.
5: **// Extract text region:**
6: V = []
7: **for** each *CP* **do**
8: Initialize zero-valued canvas A with same shape as heatmap H.
9: stack=set($A[x][y]$).
10: **while** stack **do**
11: x, y = stack.pop()
12: **if** $judgeFlow(x-1, y, x, y)$ **then**
13: stack.add($(x-1, y)$)
14: **if** $judgeFlow(x+1, y, x, y)$ **then**
15: stack.add($(x+1, y)$)
16: **if** $judgeFlow(x, y-1, x, y)$ **then**
17: stack.add($(x, y-1)$)
18: **if** $judgeFlow(x, y+1, x, y)$ **then**
19: stack.add($(x, y+1)$)
20: A[stack] = 1.0
21: $C = findCoutour(A)$
22: V.append($contourExpand(C)$)
23: **Output:** Polygon vertices V

3.3. Post-processing: Textfill Algorithm

Extracting the final predicted text bounding polygons from the output of the UHT-Net is accomplished by our novel post-processing method, the Textfill Algorithm, which is inspired by the floodfill algorithm [2]. Details are shown in Algorithm 1, which uses computer vision tools that can be found in OpenCV. The function $judgeFlow(x_1, y_1, x_2, y_2)$ is used to expand *CR* in Algorithm 1 to compute the complete text bounding polygon. The definition of CR is at Line 3 of Algorithm 1. Function $judgeFlow(x_1, y_1, x_2, y_2)$ returns *true* if $H[x_1][y_1] <= H[x_2][y_2]$ and $H[x_1][y_1] > T_{end}$, or return true if $H[x_1][y_1] >= T_{end}/2$, where the $H[x][y]$ denotes the pixel in the output from the UHT-Net. Function

$contourExpand$ is defined as a dilating process (see morphology tools in OpenCV) with the following kernel:

$$k = \begin{cases} 8 + \frac{S}{750} & S \in (0, 2 \times 10^4] \\ 35 & S > 2 \times 10^4, \end{cases}$$

where S is the pixel area of the polygonal region A.

4. Experiments

In this section[1], we introduce details of our experiments, including the datasets we use and our training strategy, and provide experimental results and their analysis.

4.1. Text Detection Datasets Used in Experiments

SynthText [12] is a large scale dataset with 800k synthetic images that are created by adding English oriented straight text with random fonts, sizes, colors, and orientations to natural images. These synthetic images are quite similar to natural scene images with text.

Total-Text [8] is a dataset with images that contain oriented straight and curved text and whose labels are annotated by bounding polygons. The image backgrounds are quite similar to real scenes. This dataset contains 1,255 training and 300 testing images.

SCUT-CTW1500 [21] is another text detection dataset which includes both English and Chinese scripts. SCUT-CTW1500 contains 1,000 training and 500 testing images in which text shape is arbitrary (as for Total-Text). Each text annotation is marked as polygon containing 14 points.

MSRA-TD500 [44] focuses on multilingual oriented straight text in natural scenes. It contains 500 images with English and Chinese scripts, which are split into 300 training and 200 testing images. Text region annotations are marked as rotated rectangles.

COCO-Text [39] is one of the challenges of ICDAR 2017 Robust Reading Competition. Its text instances in the images are English straight text regions distributed in various orientations. It contains 63,686 images in total.

4.2. Implementation Details

Data Augmentation. The process of training UHT-Net can be divided into two steps: 1) Pretraining with the SynthText dataset, and 2) fine-tuning using the Total-Text, SCUT-CTW1500, MSRA-TD500 or COCO-Text datasets respectively. To further improve training, we randomly rotated the training images and cropped with areas ranging from 0.24 to 1.0 and aspect ratios ranging from 0.33 to 3. Data augmentation is implemented in both pretraining and fine-tuning processes.

Training Strategy of UHT-Net. Our methodology was implemented in Pytorch 1.0.1 [29]. UHT-Net was pre-trained on SynthText with one epoch and then fine-tuned using Total-Text, SCUT-CTW1500, MSRA-TD500, or COCO-Text. We adopted the Adam optimizer [17] as the learning rate scheme. In the pretraining process, inspired by Smith [35], we set the initial learning rate to 3×10^{-5} for VGG-16 based UHT and 10^{-4} for ResNet-50 based UHT. We did not change it during the pretraining process. In the fine-tuning process, except for COCO-Text, we set the initial learning rate to 10^{-4} (the fine-tuning learning rate for COCO-Text is set to 5×10^{-4}). The decay rate was 0.8 every 10 epochs. Single-scale training was used. In the pretraining and fine-tuning training processes, we set the batch size to 8 on a single RTX-2080Ti GPU. In the evaluation process, the batch size was set to 1 on a single RTX-2080Ti GPU.

Hyperparameters of Textfill Algorithm. To show general results across datasets, two sets of thresholds T_{top}, T_{end} were tested: (0.7, 0.2) for Total-Text & SCUT-CTW1500 and (0.75, 0.2) for MSRA-TD500 & COCO-Text.

4.3. Experimental Results

4.3.1 Results on Curved Text Detection

Our results on the benchmarks Total-Text [8] and SCUT-CTW1500 [21] are given in Tables 1 and 2, respectively. We found that some state-of-the-art methods [42, 46] included multi-scale testing. To conduct a peer comparison, we ran experiments on curved text detection datasets with single-scale and multi-scale testing (abbreviated as "MS" below) separately. Except for the baselines [8, 21], listed methods without $^+$ used the same pre-training and fine-tuning data as we did, otherwise were different.

Results on Total-Text Dataset (Table 1). Fine-tuning on Total-Text stops at 300 epochs. During the testing process, each image is set to 700 $\times$ 700. In single-scale testing, our UHT beats all of the state-of-the-art methodologies and keeps the same F-measure score with the newest and most competitive model, CharNet H-88 [42]. UHT V16 even yields a higher recall rate than all the other state-of-the-art methods, 85.6%. This indicates that UHT is able to detect text that is missed by other methods.

Results on SCUT-CTW1500 Dataset (Table 2) Fine-tuning on SCUT-CTW1500 stops at 307 epochs for UHT V16 and 300 epochs for UHT R50. During the testing process, each image is set to 512 $\times$ 512 because the average size of images in SCUT-CTW1500 is relatively smaller than that of Total-Text. Experimental results show that UHT also performs well on SCUT-CTW1500. Surprisingly, UHT found image text that did not appear in the ground truth annotation (see Figure 4). We fixed the SCUT-CTW1500 ground truth to include missed words. To ensure fairness in evaluation, we ran experiments on two different versions of text annotations on the SCUT-CTW1500 dataset, with and without updated ground truth (Table 2). After the ground

[1] In tables of this section, **UHT V16** denotes UHT with VGG-16 backbone; **UHT R50** denotes UHT with ResNet-50 backbone.

Methodology	Venue	P (%)	R (%)	F (%)
Single-scale Testing				
Poly-FRCNN-3 [8]	IJDAR-2019	78.0	68.0	73.0
TextSnake [24]	ECCV-2018	82.7	74.5	78.4
CSE$^+$ [22]	CVPR-2019	81.4	79.7	80.2
TextField [43]	TIP-2019	81.2	79.9	80.6
PSENet-1s$^+$ [40]	CVPR-2019	84.02	77.96	80.87
FTSN [9]	ICPR-2018	84.7	78.0	81.3
ICG [38]	PR-2019	82.9	80.9	81.5
LOMO [46]	CVPR-2019	88.6	75.7	81.6
CRAFT$^+$ [7]	CVPR-2019	87.6	79.9	83.6
PSENet_v2 [41]	ICCV-2019	89.3	81.0	85.0
CharNet H-88 [42]	ICCV-2019	**89.9**	81.7	**85.6**
UHT V16 (Ours)	-	88.8	**82.6**	**85.6**
UHT R50 (Ours)	-	88.2	81.8	84.9
Multi-scale Testing				
LOMO MS [46]	CVPR-2019	87.6	79.3	83.3
CharNet H-88 MS [42]	ICCV-2019	**88.0**	85.0	**86.5**
UHT V16 MS (Ours)	-	85.0	**85.6**	85.3
UHT R50 MS (Ours)	-	85.4	84.2	84.8

Table 1. Experimental results on the Total-Text dataset. "P" means Precision, "R" Recall, "F" F-measure, * denotes results on updated groundtruth annotations, and "MS" multi-scale testing.

Methodology	Venue	P (%)	R (%)	F (%)
Single-scale Testing				
CTD [21]	PR-2019	74.3	65.2	69.5
CTD+TLOC [21]	PR-2019	74.3	69.8	73.4
TextSnake [24]	ECCV-2018	67.9	**85.3**	75.6
CSE$^+$ [22]	CVPR-2019	78.7	76.1	77.4
LOMO [46]	CVPR-2019	**89.2**	69.6	78.4
ICG [38]	PR-2019	82.8	79.8	81.3
TextField [43]	TIP-2019	83.0	79.8	81.4
CRAFT [7]	CVPR-2019	86.0	81.1	83.5
PSENet_v2 [41]	ICCV-2019	86.4	81.2	83.7
PAN Mask R-CNN$^+$ [15]	WACV-2019	86.8	83.2	85.0
UHT V16 (Ours)	-	84.3	84.8	84.5
UHT V16* (Ours)	-	86.2	84.1	**85.2**
UHT R50 (Ours)	-	85.9	83.3	84.6
UHT R50* (Ours)	-	87.4	82.3	84.8
Multi-scale Testing				
LOMO MS [46]	CVPR-2019	**85.7**	76.5	80.8
UHT V16 MS (Ours)	-	83.3	85.4	84.4
UHT V16 MS* (Ours)	-	85.2	84.7	**85.0**
UHT R50 MS (Ours)	-	81.9	**86.1**	84.0
UHT R50 MS* (Ours)	-	84.0	85.5	84.7

Table 2. Experimental results on the SCUT-CTW1500 dataset: "P" means Precision, "R" Recall, "F" F-measure, * denotes results on updated groundtruth annotations, and "MS" multi-scale testing.

truth was updated, the recall score of UHT is almost unchanged but the precision score improves a little. We welcome other researchers to run experiments on the updated SCUT-CTW1500 and therefore make it publicly available, see http://www.cs.bu.edu/faculty/betke/UHT.

Analysis on Multi-scale Testing. We tested our model with images of different sizes (500×500, 700×700, and 900×900), relying on the Fast NMS algorithm to screen out excess text bounding polygons. A reason for the increase of the recall score of multi-scale compared to single-scale

Figure 4. Examples of updated ground truth annotations: Left: The groundtruth annotation in SCUT-CTW1500 missed "89" (top) and "©Roberto Herrett" (bottom). Right: Our updated annotations.

testing may be that multi-scale testing combines image information of different sizes, making it easier for UHT to detect text regions that are difficult to detect in single-scale testing. However, false-positive samples are more likely to appear in multi-scale testing. We think that might be caused by the effort of the Fast NMS algorithm or text detection effect on very large images, all of which cause decrease of precision score of multi-scale testing (see Tables 1 and 2).

4.3.2 Results for Oriented Straight Text Detection

Results on MSRA-TD500 Dataset (Table 3). Fine-tuning on MSRA-TD500 stops at 200 epochs. During the testing process, each image is set to 512 × 512. For UHT, only single-scale texting is implemented in this experiment. The results show that our UHT beats most of the state-of-the-art methods on MSRA-TD500.

Results on COCO-Text Dataset (Table 4). With the help of the official COCO-Text API, we extracted 15,124 training images and 3,346 validation images, all of which contain at least one text region. However, testing images cannot be extracted. So in the experiments with COCO-Text, experimental results are given based on tests on the validation images for state-of-the-art frameworks and UHT. Fine-tuning on COCO-Text stops at 300 epochs. During the testing process, each image is set to 768 × 768. For UHT, only single-scale testing is implemented in this experiment. The results (Table 4) show that our UHT beats state-of-the-art methods when fine-tuned on COCO-Text training images. For more details of the fine-tuning experiments on

Methodology	Venue	P (%)	R (%)	F (%)
Zhang et al. [47]	CVPR-2016	83	67	74
He et al. [14]	CVPR-2017	77	70	74
EAST† [48]	CVPR-2017	87.3	67.4	76.1
SegLink [32]	CVPR-2017	86	70	77
PixelLink† [10]	AAAI-2018	83.0	73.2	77.8
TextSnake [24]	ECCV-2018	83.2	73.9	78.3
RRD† [19]	CVPR-2018	87	73	79
Lyu et al.† [26]	CVPR-2018	87.6	76.2	81.5
CRAFT [7]	CVPR-2019	**88.2**	**78.2**	**82.9**
UHT V16 (Ours)	-	84.2	76.2	80.0
UHT R50 (Ours)	-	83.2	77.0	80.0

Table 3. Experimental results on MSRA-TD500; "P" means Precision, "R" Recall, "F" F-measure. † denotes multi-scale testing. In this table, training data and testing scale of different methods may not be the same, which inadvertently hinders comparison.

Methodology	Venue	P (%)	R (%)	F (%)
Fine-tuned using COCO-Text				
TextSnake[24]	ECCV-2018	54.7	36.3	43.6
UHT V16 (Ours)	-	**62.2**	47.7	54.0
UHT R50 (Ours)	-	60.8	**49.0**	**54.2**
Fine-tuned using IC13 and IC17-MLT				
TextSnake[24]	ECCV-2018	35.3	33.7	34.5
CRAFT[7]	CVPR-2019	44.0	28.9	34.9
UHT V16 (Ours)	-	43.8	**43.1**	43.4
UHT R50 (Ours)	-	**46.4**	41.5	**43.8**

Table 4. Experimental results on COCO-Text; "P" means Precision, "R" Recall, "F" F-measure. As far as we know, no new work conducted experiments on testing datasets of COCO-Text since 2019. So in this table, instead of copying directly from other papers, experimental results from state-of-the-art methodologies are reimplemented by us using their official code [7] and [3].

IC13 and IC17-MLT datasets, please refer to Section 4.3.3.

4.3.3 Generalization Ability

A powerful text detection framework should have good generalization ability instead of just overfitting to a particular dataset and reaching high evaluation scores for that dataset. To further verify the generalization ability of UHT, we conducted two additional experiments: (1) we pre-trained and fine-tuned our model on datasets without curved text, here ICDAR-2015 [16] with 200 epochs, and then evaluated it on the Total-Text and SCUT-CUW1500 datasets. We chose state-of-the-art baselines which were also only fine-tuned on ICDAR-2015 [16]. (2) We fine-tuned TextSnake[24] and UHT using the same fine-tuning data as CRAFT[7].

Experimental results (Table 5) show that UHT method performs well on two curved and one oriented straight text datasets, *even if it is not fine-tuned on them.* It outperforms all listed state-of-the-art baseline methods. We suggest that the powerful generalization ability of UHT is due to its flexibility in text expression as well as the effectiveness of the Textfill Algorithm in extracting text bounding polygons by making full use of the UHT-Net output.

Dataset		Total-Text			SCUT-CTW1500		
Methodology	Venue	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
SegLink [32]	CVPR-2017	35.6	33.2	34.4	33.0	28.4	30.5
EAST [48]	CVPR-2017	49.0	43.1	45.9	46.7	37.2	41.4
PixelLink [10]	AAAI-2018	53.5	52.7	53.1	50.6	42.8	46.4
TextSnake [24]	ECCV-2018	61.5	67.9	64.6	65.4	63.4	64.4
CRAFT [7]	CVPR-2019	63.0	**72.9**	67.6	64.5	62.0	63.3
CRAFT* [7]	CVPR-2019	-	-	-	65.5	61.3	63.3
UHT V16 (Ours)	-	71.3	70.2	**70.7**	73.1	**75.3**	74.2
UHT V16* (Ours)	-	-	-	-	**74.5**	74.7	**74.6**
UHT R50 (Ours)	-	**72.7**	65.8	69.1	73.2	74.3	73.7
UHT R50* (Ours)	-	-	-	-	**74.5**	73.5	74.0

Table 5. Generalization ability on Total-Text and SCUT-CTW1500 datasets Results of Seglink [32], EAST [48], PixelLink [10] and TextSnake [24] were reported by Long et al. [24]. Results of CRAFT [7] are from the official CRAFT model [4], which we fine-tuned on the ICDAR-2015 dataset. * denotes results with respect to the annotations that we modified for SCUT-CTW1500. We used single-scale testing.

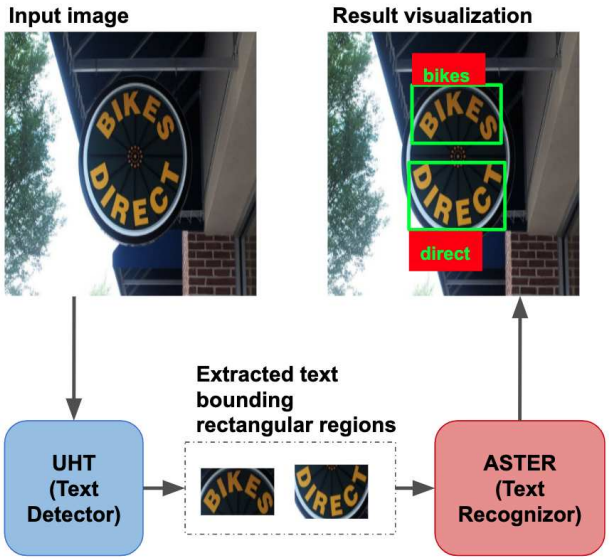

Figure 5. Pipeline of the proposed text spotting model UHTA. The model first calls the proposed UHT Detector and then converts UHT's polygonal output regions into horizontally-aligned rectangular regions of text. These text regions are then passed to the state-of-the-art text recognition model ASTER [33], which can accurately recognize the text in these regions and output text strings. So, like a person, UHTA does not only know where the text regions are, but also recognize the content of each text region.

4.4. Experimental Results on Text Spotting

So far, we have shown that the proposed UHT model can accurately localize text in natural scene images. The application range of UHT can be widen when it is embedded into a text recognition framework (Fig. 5). After all, a sighted person would not stop at the task of localizing text but also aim to identify its content. To provide a computer vision system who can take on both tasks of detecting and recognizing text in images, we here propose the model UHTA (short for UHT + ASTER [33]).

We now present a peer comparison for UHTA, showing

Methodology	Venue	F-measure (%)
Textboxes [18]	AAAI-2017	36.3
Mask TextSpotter [25]	ECCV-2018	52.9
TextNet [37]	ACCV-2018	54.0
CharNet H-88 [42]	ICCV-2019	66.6
TSA [24, 33]	ECCV-2018	58.1
UHTA V16 (Ours) [33]	-	75.7
UHTA R50 (Ours) [33]	-	**77.6**

Table 6. Experimental results of TSA and UHTA on the Total-Text Dataset. Pretrained ASTER [33] model is downloaded from official pytorch reimplementation [5]. Evaluation method for UHTA and TSA is end-to-end recognition from [6]. Annotations are horizontal text-bounding rectangles because UHT and TextSnake outputs horizontal text-bounding rectangles in UHTA and TSA. No distinction between uppercase and lowercase was made when we evaluated UHTA and TSA. The listed F-measures of the prior works were reported in their original papers. UHTA V16 denotes UHTA with VGG-16backbone; UHTA R50 denotes UHTA with ResNet-50 backbone.

the contribution of UHT in a second text spotting system, called **TSA** (TextSnake [24] + ASTER [33]) that applies the pretrained TextSnake model [3] with the same training data as UHT. The reason why we use ASTER as our text recognizor is that it can efficiently recognize curved and straight text and its code is available.

We ran experiments using the Total-Text dataset, where annotations of text spotting are included. Single-scale and lexicon-free testing were implemented in the evaluation for all models. Experimental results are detailed in Table 6, which show that UHTA has a powerful ability to spot text and outperforms state-of-the-art text spotting systems on the Total-Text dataset. Moreover, since the same conditions were applied for UHTA and TSA, the superior results of UHTA shows the effectiveness of UHT as the text detection module of the text spotting pipeline.

4.5. Analysis and Discussion

Framework Features. UHT is robust to different scales and shapes of text from natural scene images. UHT treats the text in the natural scene images directly as positive regions instead of the composition of different geometry attributes, whether it is oriented straight text or curved text. Textfill algorithm can also flexibly and accurately extract the text in the images according to the output of UHT-Net. Even if text regions are very close to each other, UHT can accurately separate words, outperforming most of the state-of-the-art methodologies in the text detection field.

Multilingual Ability. SCUT-CTW1500 and MSRA-TD500 contain English and Chinese scripts. Our results show the effectiveness of UHT in detecting scripts in a Latin language like English and Sino-Tibetan language like Chinese.

Generalization Ability. UHT outperforms state-of-the-art text detection frameworks by at least 1.5 percent points in the F-measure of Total-Text [8] (Table 5), at least 10.4 percent points in the F-measure of SCUT-CTW1500 [21] (Table 5), and 8.5 percent points in the F-measure of COCO-Text [39] (Table 4), even when not fine-tuned on them. This shows that UHT not only performs well when fine-tuned to a specific dataset, but also performs well when not. We thus conclude that UHT is robust and has a strong generalization ability.

Backbone	Total-Text	SCUT-CTW1500	MSRA-TD500	COCO-Text
UHT V16	1.6	2.1	2.6	5.2
UHT R50	1.9	1.8	3.7	4.5

Table 7. Speed of UHT. The unit of measure is FPS.

Speed Analysis. Table 7 reveals that the speed of UHT when dealing with curved text is slower than with oriented straight text. We think this might be caused by the original ground truth representation of the text region. Oriented straight text is represented by a rectangle but curved text by a more complicated multi-vertex polygon.

The backbone UHT-Net. Interestingly, analyzing all experimental results, we found that usually UHT R50 performs slightly better than UHT V16; but sometimes the opposite occurs. We think this might be caused by the hyperparameter setting: our choice of hyperparameters may not allow UHT V16 or UHT R50 to exert their optimal abilities compared with another model for a particular dataset.

Text Spotting Analysis. The strong experimental results of UHTA (Table 6) show the strength of UHT as an application – UHT performs excellent when used as a text detector in a pipeline-based text spotting system.

5. Conclusion

In this paper, we proposed a new text detection model called UHT that, with little information, can effectively detect text in natural scene images. UHT performs well in experiments with publicly available datasets. This includes experiments when UHT is fined-tuned and tested on a specific dataset and when fine-tuned and tested on different datasets. We fixed ground truth annotation errors of the SCUT-CTW1500 dataset and make the corrected ground truth publicly available. We further showed the scope of UHT by implementing a pipeline-based text spotting system that improves the results of other state-of-the-art text spotting frameworks by a range of 9.1–41.3 percent points in the F-measure. In the future, we plan to explore the possibility of detecting scripts of languages other than English and Chinese, such as Korean and Arabic, with UHT.

Acknowledgements

This work has been partially supported by the National Science Foundation, grant 1838193, and the Boston University Hariri Institute for Computing.

References

[1] https://en.wikipedia.org/wiki/Bresenham$%$27s_line_algorithm. 4

[2] https://en.wikipedia.org/wiki/Flood_fill. 4

[3] https://github.com/princewang1994/TextSnake.pytorch. 7, 8

[4] https://github.com/clovaai/CRAFT-pytorch. 7

[5] https://github.com/ayumiymk/aster.pytorch. 8

[6] https://github.com/liuheng92/OCR_EVALUATION. 8

[7] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. *CoRR*, abs/1904.01941, 2019. 2, 3, 6, 7

[8] C. K. Chng and C. S. Chan. Total-text: A comprehensive dataset for scene text detection and recognition. *CoRR*, abs/1710.10400, 2017. 5, 6, 8

[9] Y. Dai, Z. Huang, Y. Gao, and K. Chen. Fused text segmentation networks for multi-oriented scene text detection. *CoRR*, abs/1709.03272, 2017. 6

[10] D. Deng, H. Liu, X. Li, and D. Cai. Pixellink: Detecting scene text via instance segmentation. *CoRR*, abs/1801.01315, 2018. 2, 7

[11] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2963–2970, 2010. 2

[12] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. *CoRR*, abs/1604.06646, 2016. 5

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4

[14] W. He, X. Zhang, F. Yin, and C. Liu. Deep direct regression for multi-oriented scene text detection. *CoRR*, abs/1703.08289, 2017. 1, 2, 7

[15] Z. Huang, Z. Zhong, L. Sun, and Q. Huo. Mask R-CNN with pyramid attention network for scene text detection. *CoRR*, abs/1811.09058, 2018. 6

[16] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015. 7

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[18] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. *CoRR*, abs/1611.06779, 2016. 2, 8

[19] M. Liao, Z. Zhu, B. Shi, G. Xia, and X. Bai. Rotation-sensitive regression for oriented scene text detection. *CoRR*, abs/1803.05265, 2018. 7

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. 2

[21] Y. Liu, L. Jin, S. Zhang, and S. Zhang. Detecting curve text in the wild: New dataset and new solution. *CoRR*, abs/1712.02170, 2017. 5, 6, 8

[22] Z. Liu, G. Lin, S. Yang, F. Liu, W. Lin, and W. L. Goh. Towards robust curve text detection with conditional spatial expansion. *CoRR*, abs/1903.08836, 2019. 6

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 3

[24] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2, 3, 6, 7, 8

[25] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *CoRR*, abs/1807.02242, 2018. 2, 8

[26] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai. Multi-oriented scene text detection via corner localization and region segmentation. *CoRR*, abs/1802.08948, 2018. 1, 7

[27] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. 2

[28] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016. 3

[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 5

[30] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 2

[31] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 4

[32] B. Shi, X. Bai, and S. J. Belongie. Detecting oriented text in natural images by linking segments. *CoRR*, abs/1703.06520, 2017. 7

[33] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai. Aster: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 7, 8

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[35] L. N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015. 5

[36] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *CoRR*, abs/1707.03237, 2017. 4

[37] Y. Sun, C. Zhang, Z. Huang, J. Liu, J. Han, and E. Ding. Textnet: Irregular text reading from images with an end-to-end trainable network. *CoRR*, abs/1812.09900, 2018. 8

[38] J. Tang, Z. Yang, Y. Wang, Q. Zheng, Y. Xu, and X. Bai. Detecting dense and arbitrary-shaped scene text by instance-aware component grouping. *Pattern Recognition*, 2019. 6

[39] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. In *arXiv preprint arXiv:1601.07140*, 2016. 5, 8

[40] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao. Shape robust text detection with progressive scale expansion network. *CoRR*, abs/1903.12473, 2019. 2, 6

[41] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen. Efficient and accurate arbitrary-shaped text detection with pixel aggregation network. *arXiv preprint arXiv:1908.05900*, 2019. 2, 6

[42] L. Xing, Z. Tian, W. Huang, and M. R. Scott. Convolutional character networks. *arXiv preprint arXiv:1910.07954*, 2019. 2, 5, 6, 8

[43] Y. Xu, Y. Wang, W. Zhou, Y. Wang, Z. Yang, and X. Bai. Textfield: Learning A deep direction field for irregular scene text detection. *CoRR*, abs/1812.01393, 2018. 6

[44] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1083–1090, June 2012. 5

[45] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *CoRR*, abs/1606.09002, 2016. 1, 2

[46] C. Zhang, B. Liang, Z. Huang, M. En, J. Han, E. Ding, and X. Ding. Look more than once: An accurate detector for text of arbitrary shapes. *CoRR*, abs/1904.06535, 2019. 2, 5, 6

[47] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. *CoRR*, abs/1604.04018, 2016. 1, 2, 7

[48] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: An efficient and accurate scene text detector. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2, 3, 7

[49] Y. Zhu and J. Du. Textmountain: Accurate scene text detection via instance segmentation. *CoRR*, abs/1811.12786, 2018. 2