E²VTS: Energy-Efficient Video Text Spotting from Unmanned Aerial Vehicles

Zhenyu Hu[1]*, Pengcheng Pi[1]*, Zhenyu Wu[1], Yunhe Xue[1], Jiayi Shen[1], Jianchao Tan[2], Xiangru Lian[2],
Zhangyang Wang[3], and Ji Liu[2] †

[1]Texas A&M University   [2]Kwai Inc.   [3]The University of Texas at Austin

Abstract

Unmanned Aerial Vehicles (UAVs) based video text spotting has been extensively used in civil and military domains. UAV's limited battery capacity motivates us to develop an energy-efficient video text spotting solution. In this paper, we first revisit RCNN's crop & resize training strategy and empirically find that it outperforms aligned RoI sampling on a real-world video text dataset captured by UAV. To reduce energy consumption, we further propose a multi-stage image processor that takes videos' redundancy, continuity, and mixed degradation into account. The model is pruned and quantized before deployed on Raspberry Pi. Our proposed energy-efficient video text spotting solution, dubbed as E²VTS, outperforms all previous methods by achieving a competitive tradeoff between energy efficiency and performance. All our codes and pre-trained models are available at https://github.com/wuzhenyusjtu/LPCVC20-VideoTextSpotting.

1. Introduction

UAV-based video text spotting is broadly applied in assistive navigation, automatic translation, road sign recognition, industrial monitoring, and disaster response, etc. A standard video text spotting model has four components: text detector, text recognizer, text tracker, and post-processing.

Existing video text spotting solutions [37, 10] are purely performance-driven and fail to take energy consumption into account. Multi-frame related features are first obtained in frame-wise detection or tracking. Then, they are aggregated for enhancement in a cross-frame and multi-scale way for text recognition. Therefore, existing performance-driven solutions are high in energy consumption and unsuitable for resource-constrained UAV platforms.

In this paper, we propose an Energy-Efficient Video Text

Spotting solution, dubbed as E²VTS. Our contribution can be summarized as follows:

- **Novel Training & Inference Strategies:** To obtain better text spotting performance, we revisit RCNN and empirically find that crop and resize outperforms aligned RoI Pooling when connecting the text recognizer with the text detector. To further save energy consumption, we propose a multi-stage image processor to select the highest-quality frame in a sliding window, reject text-free frames as well as crop non-text regions, and reject out-of-distribution frames.

- **Experiments:** On a real-world UAV-captured text video dataset deployed on Raspberry Pi, we conducted thorough ablation studies on the proposed training and inference strategies. The evaluation metric takes both energy consumption and text spotting performance into consideration. Models are pruned and quantized before deployment.

2. Related Work

2.1. Text Reading in Images

Text Detection: Object detection-based [21, 48, 24, 47, 36] and sub-text components-based [11, 42, 32, 35, 2] are two streams of solutions to text detection. Based on the observation that any part of a text instance is still text, sub-text components-based methods incorporate the inductive biases of the homogeneity and locality of text instances into model design.

Text Recognition: There are two major strategies in decoding text content from image encoded features from CNN, Connectionist Temporal Classification (CTC) [13] and the encoder-decoder framework [30]. CTC-based methods includes CRNN [29] and Sliding Convolutional Character Models [46]. Encoder-Decoder Methods include R²AM [16] and Edit Probability (EP) [4].

End-to-end Text Spotting: The representative end-to-end text spotting are summarized into two categories, regular-shaped and arbitrary-shaped.

Regular-shaped Text: Li *et al.* [17, 34] proposed the first deep-learning based end-to-end trainable scene text spotting method for horizontal text by incorporating RoI Pooling [27] to join the detection and recognition stage. Deep TextSpotter [5] handled multi-orientation text instances without feature sharing between the detection and recognition stages. End-to-End TextSpotter [14] and FOTS [22] adopted an anchor-free mechanism to improve both the training and inference speed. They use two similar sampling strategies, *i.e.*, Text-Alignment and RoIRotate, to extract feature from arbitrary-oriented quadrilateral detection results.

Arbitrary-shaped Text: Mask TextSpotter [19, 20] used character-level supervision to simultaneously detect and recognize characters and instance masks. Nonetheless, the character-level ground truths are expensive, thus mostly unavailable for real data. RoI Masking [26] cropped out the features from the predicted axis-aligned rectangular bounding boxes and multiplied the features with the corresponding instance segmentation mask. TextDragon [12] proposed RoISlide to transform the whole text features into axis-aligned features indirectly by transforming each local quadrangle sequentially. As the first one-stage text spotting method, CharNet [43] directly outputted bounding boxes of words, with corresponding character labels. ABCNet [23] adaptively fitted arbitrarily-shaped text by a parameterized Bezier curve and used BezierAlign layer to extract accurate convolution features. CRAFTS [3] used the character region feature from the detector as input character attention to the recognizer.

2.2. Text Reading in Videos

Text Detection & Tracking: Wang *et al.* [33] proposed a multi-scale feature sampling and warping network on adjacent frames, and an attention-based multi-frame feature aggregation mechanism to fuse the complementary text features from related frames. Wu *et al.* [41] explored Delaunay triangulation to detect and track texts. The triangular mesh pattern reflects text properties, such as regular spacing between characters and constant stroke width, thus distinguishable from non-text. Yang *et al.* [44] combined single-frame detection with cross-frame motion-based tracking. The text association was formulated into a cost-flow network. Tian *et al.* [31] located character candidates locally and searched text regions globally. Specifically, a multi-strategy tracking based text detection approach [49] was used to globally search and select the best text region with dynamic programming. Wang *et al.* [38] proposed a fully convolutional model based on a novel refine block structure, which refines the low-resolution semantic features with the high-resolution low-level features.

End-to-End Text Spotting Wang *et al.* [37] proposed a multi-frame tracking based method, where text detection and recognition are done on each frame before recognized texts are tracked over the video sequence. FREE [10, 9] proposed a text recommender to select the highest-quality text from text streams for recognizing and released a large scale video text spotting dataset.

3. E^{2}VTS: An Energy-Efficient Video Text Spotting Solution

Overview: The E^{2}VTS two-step text spotting system adopts Efficient and Accurate Scene Text Detector (EAST) as the text detector, and Convolutional Recurrent Neural Network (CRNN) as the text recognizer. The recognizer is connected with the detector via crop & resize. A multi-stage image processor is proposed to further save the energy consumption. It has three stages, selecting the highest-quality frame in a sliding window, rejecting text-free images and cropping non-text regions, and rejecting out-of-distribution images. The pipeline is shown in Figure 1,

3.1. Revisiting RCNN: Crop & Resize vs. Aligned RoI Pooling

We compare two connecting mechanisms for the detector and the recognizer: crop+resize versus aligned RoI pooling. Examples of aligned RoI pooling include BezierAlign [23] for arbitrary-shaped text and RoIRotate [22] for rotated text.

Given the predicted bounding box from the detector, in crop+resize, the input to the recognizer is the cropped box area affinely transformed from the original image and resized to a fixed resolution. The detector and the recognizer are trained independently. In aligned RoI pooling, the input to the recognizer is the cropped box area affinely transformed from the feature map. The detector and the recognizer are trained jointly. Note that the text recognition loss uses the ground truth text regions instead of predicted text regions.

Unlike the benchmarks in image-based text spotting, real-world videos for text spotting are full of small size and poor quality text boxes. Consequently, crop+resize outperforms aligned RoI pooling for two reasons. First, aligned RoI pooling losses the discriminative details for small size text boxes due to the deep convolutions in the detector. In contrast, crop+resize enlarges the input resolution of small size text boxes and preserves their discriminative spatial details [39]. Second, text recognition (*i.e.*, knowing what is the text) is intrinsically more difficult than text detection (*i.e.*, knowing where is the text). Thus, feature sharing and joint training will lead to sub-optimal performance for both tasks [8, 7].

3.2. Multi-Stage Image Processor

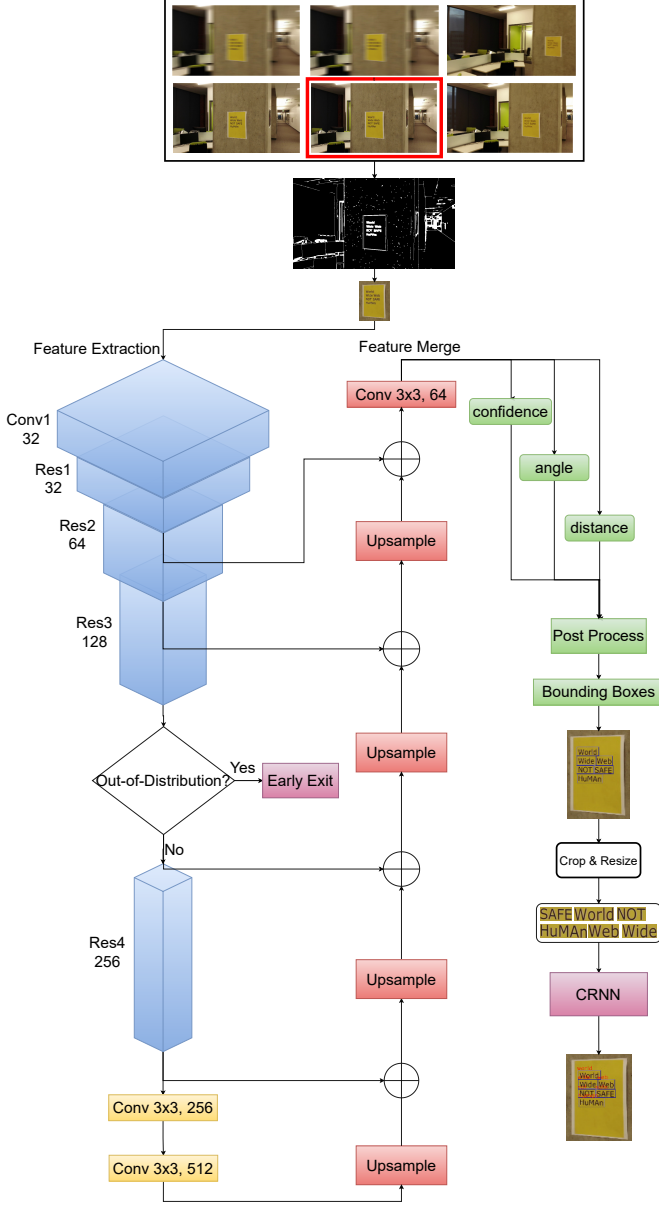Different from a single image, video frames are redundant and continuous in the temporal domain. Comparing

Figure 1: Overview: E²VTS consists of two components. Component one is a multi-stage image processor which selects the best frame within a window size and crops out the background. Component two is a two-step crop & resize text spotting system including an EAST detector and a CRNN recognizer. The EAST detector is based on ResNet34 backbone and outputs confidence, angle, and distance. Out-of-distribution frames are rejected at ResNet Layer3.

one frame with its precedents and successors over certain metrics is a natural filtering process to select the most suitable frame for the later detection task. We also leverage sharp transitions of text regions to remove non-text back-

ground preliminarily to further boost efficiency. All these implementations are based on simple signal processing algorithms which are significantly faster than neural network models.

3.2.1 Stage I: Selecting the Highest-Quality Frame in a Sliding Window

Problem Definition: Blur is the major artifact in UAV captured videos due to camera shake, depth variation, object motion or a combination of them [40, 15]. Among all the frames describing the same visual scene, the clearest image gives the least amount of detector or recognition error. Since blurred frames contain less energy in the high frequency components, in their associated power spectrum [45], the power tends to fall much faster with increasing frequency, compared with clear frames. Therefore, the average of the power spectra of clear frames is higher than these degraded ones, as degraded ones have a steeper slope on their power spectrum.

Implementation: In Fig. 2, we propose a sliding window mechanism and select the highest-quality frame in each window. Given a video containing L frames, the i-th window $\mathcal{W}_i$ is obtained via:

$$\mathcal{W}_i = S_{(i,N)}(I_1, ..., I_L), \tag{1}$$

where S represents the sliding rule and N is the window size. The selected highest-quality frame in $\mathcal{W}_i$ is:

$$I_{HQ} = \underset{I \in W_i}{\mathrm{argmax}}\, \mathcal{G}(I), \tag{2}$$

where $\mathcal{G}$ is the quality measure. We propose two measures in this work: variance of Laplacian[25] and average fast Fourier transform (FFT) magnitude defined as:

$$\mathcal{G}_{FFT} = \frac{1}{hw}\|\mathrm{FFT}(I_0)\|,$$
$$\mathcal{G}_{LV} = \mathrm{Var}(k_L * I_0), \tag{3}$$

where I_0 is a given frame with height h and width w. FFT magnitude measure is an approximation of power spectrum density in frequency domain. Variance of Laplacian stresses spatial information by counting sharp transitions in the frame. These two measure works in a complementary way. Therefore, we integrate the two methods by taking a weighted average of two measures' scores ranking over a certain window. Let $\mathrm{rank}(I, W, \mathcal{G})$ denotes a function that returns the rank of frame I among all the frames in the window W scored by the quality measure $\mathcal{G}$ in ascending order. The selected highest-quality frame is:

$$I_{HQ} = \underset{I \in W_i}{\mathrm{argmax}}[\lambda \cdot \mathrm{rank}(I, W_i, \mathcal{G}_{FFT}) \\ + (1-\lambda) \cdot \mathrm{rank}(I, W_i, \mathcal{G}_{LV})], \tag{4}$$
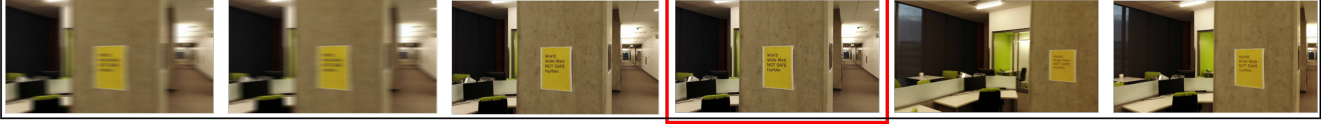
3

Figure 2: Sliding window for highest-quality frame selection. A window iterator is sliding over the temporally sub-sampled frames and quality scoring is conducted on the frames via the proposed measure. The highest ranked frame is selected.

where λ is the relative weight parameter.

In practice, the video sequence is sub-sampled at rate r to further boost efficiency before applying the sliding window filter. As a hyper-parameter, the sub-sample rate r has a great impact on the tradeoff between energy-efficiency and performance. Although setting higher sub-sample rate could save more energy, it has higher chance to miss scenes for text spotting.

3.2.2 Stage II: Rejecting Text-free Images and Cropping Non-Text Regions

Problem Definition: Known for high time complexity and energy consumption, connected component-based text detection depends on maximally stable extremal region (MSER) as character candidates, and stroke width transform (SWT) for filtering and pairing of connected components. Given the observation that cohesive characters compose a word or sentence sharing similar properties such as spatial location, size, and stroke width, we turn to Canny edge detector [6] to locate the edge pixels that build the text's structure (a.k.a. contour).

Implementation: In Fig. 3, we further reject text-free images and crop non-text regions. First, Canny edge detector is applied on the three channels of the input image I_{yuv} represented in YUV color space, and the three channels (Y_c, U_c, V_c) are merged by bitwise OR (denoted as "|") operation to obtain the edge map I_e. Then, morphological closing is applied on the I_e to remove small holes and merge connected components. If any text region is present in the image, a binary image with continued text characters will be returned. Next, the histogram map is obtained by summing up pixels along the x and y axis [1]:

$$H_x[i] = \sum_{k=1}^{h} I_c[i,k], \quad H_y[j] = \sum_{k=1}^{w} I_c[k,j], \quad (5)$$

where w and h are the width and height of the image. After that, all the peaks [2] for these two histogram maps, *i.e.*, P_x and P_y, are found. Text regions are assumed to fall within

the peaks. Finally, Text-free images are rejected based on two preset thresholds (θ, α) on the peak intensities and numbers, respectively. Note that the second-stage selector cannot deal with images with complicated backgrounds, since the peaks value varies along both axes without any identifiable pattern. Therefore, images with complicated background whose peaks are consistently high along the entire x and y axis are accepted.

Cropping text regions improves the SNR [3] in the image. On images with simple background, the text regions are assumed to lie between (x_l, y_b) and (x_r, y_t). The coordinates of the text region are obtained from the peaks via

$$x_l, x_r, y_b, y_t = P_x[1], P_x[-1], P_y[1], P_y[-1]. \quad (6)$$

The details of the second-stage selector is shown in Algorithm 1.

Algorithm 1: Rejecting Text-free Images or Cropping Text Regions

---

1 Initialization: θ, α: predefined thresholds
2 $I_{yuv} \leftarrow \text{RGB2YUV}(I)$
3 $Y_c, U_c, V_c \leftarrow \text{CannyEdge}(I_{yuv})$
4 $I_e \leftarrow Y_c \mid U_c \mid V_c$
5 $I_c \leftarrow \text{MorphClose}(I_e)$
6 $H_x, H_y \leftarrow \text{Histogram}(I_c)$ `// sum up pixels among axis`
7 $P_x, P_y \leftarrow \text{FindPeaks}(H_x, H_y)$
8 $\mu_x, \mu_y \leftarrow \text{Mean}(P_x), \text{Mean}(P_y)$
  `// Whether the number of peaks or the mean of intensity is less than preset thresholds`
9 **if** $Count(P_x) \leq \theta$ *or* $Count(P_y) \leq \theta$ *or* $\mu_x \leq \alpha$ *or* $\mu_y \leq \alpha$ **then**
10 | REJECT
11 **else**
12 | ACCEPT
13 | $x_l, x_r, y_b, y_t \leftarrow P_x[1], P_x[-1], P_y[1], P_y[-1]$
14 | **return** $I[x_l : x_r, y_b : y_t]$

---

[1] With the origin in lower left corner, the x-axis is running from left to right and the y-axis is running from bottom to up.

[2] The peaks are all local maxima by comparing neighboring values in the histogram.

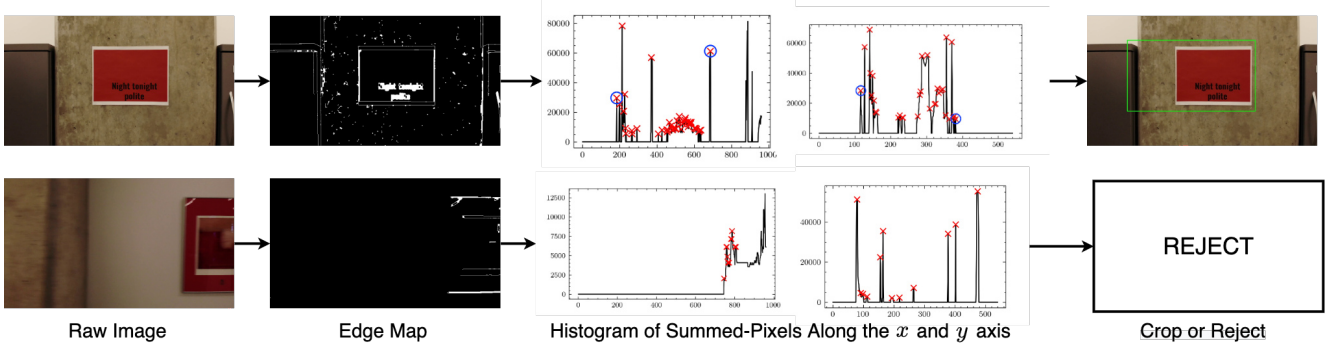[3] We treat text related pixels as signal and all other pixels as noise.

Figure 3: Cropping text foreground: we use the histogram to analyze the edge information of the selected frame. If the number of peaks and the mean of intensity satisfy predefined thresholds, text bounding coordinates will be selected from peaks info. Otherwise, the frame will be discarded.

3.2.3 Stage III: Rejecting Out-of-Distribution Images

Problem Definition: A "trained" E^2VTS model f with fixed parameters is able to fit a distribution $\mathcal{X}_f$ defined on the image space. During inference, rejecting the out-of-distribution images in an early-exit way could greatly reduce energy consumption. The out-of-distribution rejection problem [18] can be formulated as a binary classification. Examples of the positive cases and negative cases used to train the rejector are shown in Fig. 4.

Implementation: Grad-CAM [28], a visual explanations technique via gradient-based localization, is deployed to locate the first text semantic-aware layer l for our model. The outputs of the text semantic-aware layer H_l serve as the high-level features to distinguish the out-of-distribution images from the in-distribution ones. Support Vector Machines (SVM) is used for binary classification on H_l. SVM is preferred over the deep model due to its small size in the number of parameters and low latency.

4. Experiments

4.1. Experiment Settings

4.1.1 Datasets and Evaluation Protocols

We evaluate the proposed E^2VTS approach on the LPCVC-20 video text spotting dataset, abbreviated as **LPCVC-20**. The videos are captured by UAVs flying indoors on the corridors, where tons of posters and board signs with rotated text are presented. Five videos were used for training and one video was reserved for testing. After decomposing videos into frames, we handpick text-related images as our system experiment datasets which include $7,886$ for training and $2,033$ for texting. Furthermore, the text was annotated using the Auto Labeling algorithm described in the coming subsection. LPCVC-20 consists of images of resolution 3840×2160, 1920×1080, and 1280×720.

IoU, IoP, IoG [4] are used for detection and edit-distance is used for recognition. For each ground truth, the predicted bounding box with the maximum IoU is selected and the edit distance is calculated between the ground truth text label and the predicted text.

Auto Labeling: Image Registration-Aided Annotation for Video Text Spotting Given the observation that temporally consecutive frames are describing the same scene, we propose Auto-Labelling in Algorithm 2 to aid the video annotation, which utilizes the videos' temporal redundancy and continuity. It takes advantage of feature matching and perspective transformation to transfer the annotated bounding box from the source frame to the target frame. Figure 6 shows the annotation results produced by Algorithm 2.

Algorithm 2: Auto-Labeling

```
   // Annotate the 1st frame
1  b_s ← Annotate(V[1]) // Number of frames
       describing the same scene
2  N ← Size(V)
3  for i ← 2 to N do
       // Next Adjacent frame
4      I_t ← V[i] // Feature Matching
5      k_1, d_1 ← SIFT(I_s); k_2, d_2 ← SIFT(I_t)
6      m ← LoweRatioTest(BFMatcher(d_1, d_2))
7      p_s, p_t ← FilterKeyPts(m, k_1), FilterKeyPts(m, k_2)
       // Perspective Transformation
8      b_t ← Perspective(b_s, HomographyMatrix(p_s, p_t))
9      I_s ← I_t; b_s ← b_t
```

---

[4]Given a ground truth bounding box area G and predicted bounding box area P the IoU is $(P \cap G)/(P \cup G)$, IoP (a.k.a precision) is $(P \cap G)/P$, and IoG (a.k.a recall) is $(P \cap G)/G$.
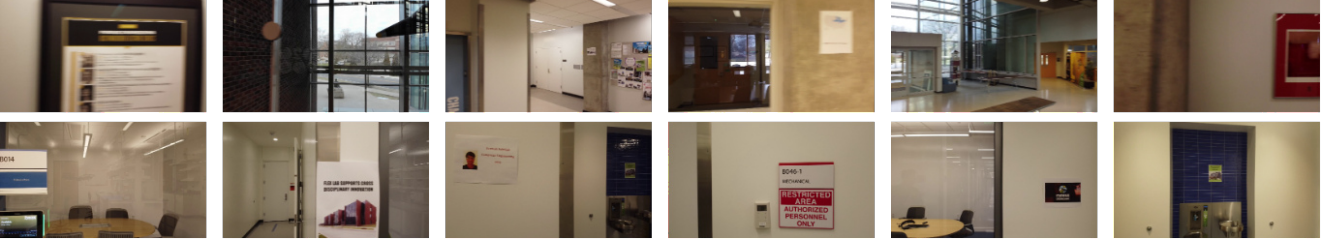
Figure 4: The negative samples in the first row and positive samples in the second row are used to train the out-of-distribution rejector. Heavily-blurred, text-free, are truncated-text are all considered as negative cases to be rejected.

Figure 5: Sample Images from the LPCVC-20 Video Text Spotting Dataset.

Table 1: E^{2}VTS results on LPCVC-20

EditDistance \ Resolution		1200	600	300
BBox Area/Char Count	<=20	N/A	6.86	3.69
	<=60	5.25	2.12	3.60
	>60	1.93	2.00	2.92
Char Count	<=4	1.08	1.31	2.21
	<=8	1.86	2.08	3.56
	>8	4.20	3.79	5.32
Total		1.93	2.04	3.26

Energy Consumption Measurement A USB power meter [5] is used to measure energy consumption. We connect the USB power meter in series to the power supply of the Raspberry Pi. With this setup, the power meter can real-time measure the current through the Raspberry Pi. Since the voltage for Raspberry Pi is constantly 5V, we can calculate the energy consumption by recording the current values. The power meter is connected with a computer through Bluetooth and the energy measurements of the Raspberry Pi are recorded using [1]. The timestamps for model inference are written down to measure the latency for the model.

4.1.2 Model Compression

Pruning Pruning algorithm compresses neural network by removing redundant weights or channels of layers. For a Raspberry Pi, structured pruning is preferred over unstructured pruning, since structured pruning does not require specific hardware support for deployment. For our experiment, we applied ℓ_1 filter Pruner with a one-shot pruning strategy and a sparsity rate of 0.7, which allowed our model to achieve the best trade-off between accuracy and energy efficiency.

Quantization Quantization refers to techniques for using a reduced precision integer representation for weights and activations. For Raspberry Pi, Pytorch provides QN-NPACK backends which supports running quantized opera-

---

tors efficiently on ARMS CPU. For our experiment, we applied static post quantization on all convolutional and fully-connected layers; and applied dynamic post quantization on the LSTM modules in the CRNN model.

4.2. Ablation Studies

4.2.1 Crop & Resize vs. Aligned RoI Pooling

In this section, we conduct ablation studies for the two-step Crop and Resize E^{2}VTS text spotting model and the two-stage Aligned RoIPool text spotting model. From Tables 1 and 2 it can be seen that the E^{2}VTS model performs better than the Aligned RoIPool model at all resolutions. The different factors that influence the performance of the model are also measured. From Tables 1 and 2 it can be concluded that the greater bounding box to character count ratio and the lesser character count improves the recognition performance. Table 3 shows the deployment results of E^{2}VTS on Raspberry Pi.

4.2.2 Multi-Stage Image Processor

We compare the overall performance for our method after incorporating different data level efficiency techniques. As shown in Table. 4, incorporating data level efficiency results in a better performance in both accuracy and efficiency.

Based on the results in Table 4, Stage I data pre-processing greatly improves the accuracy of the model by selecting the best quality frame within a window size as the model's input. The latency and energy decrease slightly
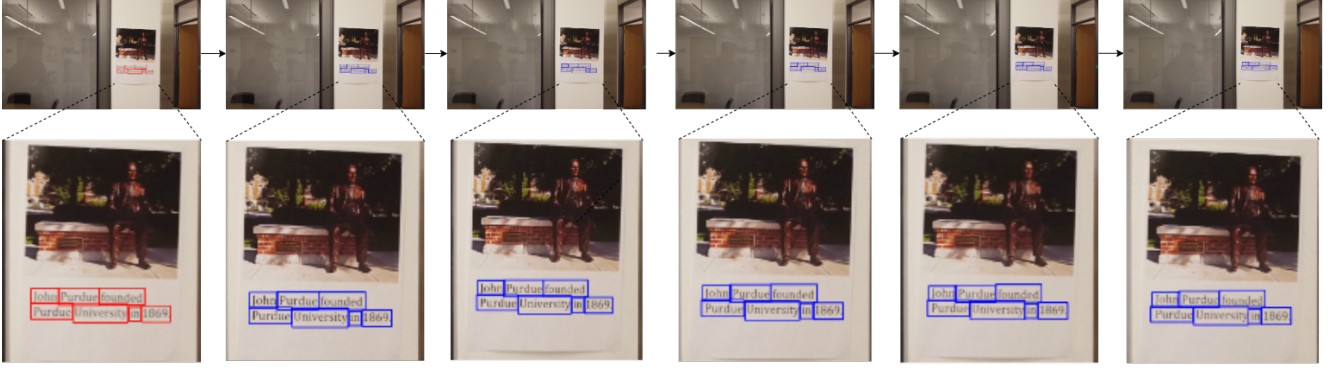
Figure 6: Qualitative Results of Auto Labeling: extract features from the source frame and the target frame. Conduct feature matching and perspective transform to update bounding boxes annotation. Repeat the process until the end of the scene.

Table 2: FOTS results on LPCVC-20

EditDistance \ Resolution		1200	600	300
BBox Area/Char Count	<=20	N/A	6.84	4.19
	<=60	4.56	2.84	5.19
	>60	2.62	3.48	5.97
Char Count	<=4	1.83	2.38	3.28
	<=8	2.98	4.21	6.12
	>8	3.95	4.92	8.61
Total		2.65	3.48	5.25

Table 3: Performance, Latency, and Energy Measurement of E^2VTS on Raspberry Pi

Model	IoU	IoP	IoG	EditDistance	Latency	Avg Energy
E^2VTS	72.21	76.24	93.94	1.39	12.90	31.77

Table 4: Ablation studies on the multi-stage image processor. Performance, latency, and energy consumption are evaluated.

Stage I	Stage II	Stage III	Latency	Energy	EditDistance
✓			627.43	1841.49	**0.78**
	✓		545.20	1349.23	1.14
		✓	571.48	1482.31	1.05
✓	✓	✓	**528.12**	**1267.2**	0.96

due to the extra cost introduced by quality scoring and the decrease of the video's sub-sample rate. Stage II data pre-processing mainly decreases the latency and average energy consumption of the model by improving the SNR in the image and rejecting low-quality and non-text frames. Stage III data pre-processing also decreases latency and energy consumption by rejecting out-of-distribution frames at an early stage of the detection model. The integration of Stage I, II, and III data pre-processing benefits the model from the perspective of speed and energy consumption.

4.2.3 Deployment on Raspberry Pi

In this section, we evaluate the overall performance of our method after incorporating different model level efficiency techniques, which include pruning and quantization.

Table 5: Ablation studies on pruning(**P**) and quantization (**Q**).

P	Q	Latency	Energy	EditDistance
	✓	76.48	195.25	**1.09**
✓		56.67	164.73	1.12
✓	✓	**12.90**	**39.23**	1.14

Based on the results in Table 5, model pruning and quantization significantly decrease latency and average energy consumption respectively. Although implementing model compression results in a sightly drop in accuracy, the trade-off between energy efficiency and accuracy shows that incorporating model level efficiency notably boosts overall performance.

5. Conclusion

In this paper, we proposed an energy-efficient video text spotting solution, dubbed as E^2VTS, for Unmanned Aerial Vehicles. E^2VTS is an energy-efficiency driven model without compromising text spotting performance. The proposed system not only utilizes data level efficiency enhancement techniques but also makes use of model level efficiency boosting methods such as pruning and quantization. Specifically, a sliding window is used to select scene-wise highest quality frame; a Canny edge based algorithm is proposed to reject text-free images and non-text frames; a dynamic routing mechanism emphasizes the in-distribution inputs. Far

from the application on UAV devices, our video text spotting system is competent for any energy-constrained scenario.

6. Acknowledgement

References

[1] Usb power meter measurement github repository. https://github.com/kolinger/rd-usb.git/. 6

[2] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *CVPR*, 2019. 1

[3] Youngmin Baek, Seung Shin, Jeonghun Baek, Sungrae Park, Junyeop Lee, Daehyun Nam, and Hwalsuk Lee. Character region attention for text spotting. In *ECCV*, 2020. 2

[4] Fan Bai, Zhanzhan Cheng, Yi Niu, Shiliang Pu, and Shuigeng Zhou. Edit probability for scene text recognition. In *CVPR*, 2018. 1

[5] Michal Busta, Lukas Neumann, and Jiri Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *ICCV*, 2017. 2

[6] John Canny. A computational approach to edge detection. *PAMI*, 1986. 4

[7] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Decoupled classification refinement: Hard false positive suppression for object detection. *arXiv*, 2018. 2

[8] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Revisiting rcnn: On awakening the classification power of faster rcnn. In *ECCV*, 2018. 2

[9] Zhanzhan Cheng, Jing Lu, Yi Niu, Shiliang Pu, Fei Wu, and Shuigeng Zhou. You only recognize once: Towards fast video text spotting. In *ACMMM*, 2019. 2

[10] Zhanzhan Cheng, Jing Lu, Baorui Zou, Liang Qiao, Yunlu Xu, Shiliang Pu, Yi Niu, Fei Wu, and Shuigeng Zhou. Free: A fast and robust end-to-end video text spotter. *TIP*, 2020. 1, 2

[11] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. Pixellink: Detecting scene text via instance segmentation. *arXiv preprint arXiv:1801.01315*, 2018. 1

[12] Wei Feng, Wenhao He, Fei Yin, Xu-Yao Zhang, and Cheng-Lin Liu. Textdragon: An end-to-end framework for arbitrary shaped text spotting. In *ICCV*, 2019. 2

[13] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006. 1

[14] Tong He, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun. An end-to-end textspotter with explicit alignment and attention. In *CVPR*, 2018. 2

[15] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *ICCV*, 2019. 3

[16] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *CVPR*, 2016. 1

[17] Hui Li, Peng Wang, and Chunhua Shen. Towards end-to-end text spotting with convolutional recurrent neural networks. In *ICCV*, 2017. 2

[18] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018. 5

[19] Minghui Liao, Pengyuan Lyu, Minghang He, Cong Yao, Wenhao Wu, and Xiang Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *PAMI*, 2019. 2

[20] Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, and Xiang Bai. Mask textspotter v3: Segmentation proposal network for robust scene text spotting. In *ECCV*, 2020. 2

[21] Minghui Liao, Baoguang Shi, and Xiang Bai. Textboxes++: A single-shot oriented scene text detector. *TIP*, 2018. 1

[22] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. In *CVPR*, 2018. 2

[23] Yuliang Liu, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Liangwei Wang. Abcnet: Real-time scene text spotting with adaptive bezier-curve network. In *CVPR*, 2020. 2

[24] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. Arbitrary-oriented scene text detection via rotation proposals. *TMM*. 1

[25] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 314–317. IEEE, 2000. 3

[26] Siyang Qin, Alessandro Bissacco, Michalis Raptis, Yasuhisa Fujii, and Ying Xiao. Towards unconstrained end-to-end text spotting. In *ICCV*, 2019. 2

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *TPAMI*, 2016. 2

[28] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 5

[29] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *TPAMI*, 2016. 1

[30] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *NeurIPS*, 2014. 1

[31] Shu Tian, Wei-Yi Pei, Ze-Yu Zuo, and Xu-Cheng Yin. Scene text detection in video by learning locally and globally. In *IJCAI*, 2016. 2

[32] Zhuotao Tian, Michelle Shu, Pengyuan Lyu, Ruiyu Li, Chao Zhou, Xiaoyong Shen, and Jiaya Jia. Learning shape-aware embedding for scene text detection. In *CVPR*, 2019. 1

[33] Lan Wang, Jiahao Shi, Yang Wang, and Feng Su. Video text detection by attentive spatiotemporal fusion of deep convolutional features. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 66–74, 2019. 2

[34] Peng Wang, Hui Li, and Chunhua Shen. Towards end-to-end text spotting in natural scenes. *arXiv*, 2019. 2

[35] Wenhai Wang, Enze Xie, Xiang Li, Wenbo Hou, Tong Lu, Gang Yu, and Shuai Shao. Shape robust text detection with progressive scale expansion network. In *CVPR*, 2019. 1

[36] Xiaobing Wang, Yingying Jiang, Zhenbo Luo, Cheng-Lin Liu, Hyunsoo Choi, and Sungjin Kim. Arbitrary shape scene text detection with adaptive text region representation. In *CVPR*, 2019. 1

[37] Xiaobing Wang, Yingying Jiang, Shuli Yang, Xiangyu Zhu, Wei Li, Pei Fu, Hua Wang, and Zhenbo Luo. End-to-end scene text recognition in videos based on multi frame tracking. In *ICDAR*, 2017. 1, 2

[38] Yang Wang, Lan Wang, Feng Su, and Jiahao Shi. Video text detection with fully convolutional network and tracking. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1738–1743. IEEE, 2019. 2

[39] Jianchao Wu, Zhanghui Kuang, Limin Wang, Wayne Zhang, and Gangshan Wu. Context-aware rcnn: A baseline for action detection in videos. In *ECCV*. Springer, 2020. 2

[40] Junru Wu, Xiang Yu, Ding Liu, Manmohan Chandraker, and Zhangyang Wang. David: Dual-attentional video deblurring. In *WACV*, 2020. 3

[41] Liang Wu, Palaiahnakote Shivakumara, Tong Lu, and Chew Lim Tan. A new technique for multi-oriented scene text line detection and tracking in video. *TMM*, 2015. 2

[42] Yue Wu and Prem Natarajan. Self-organized text detection with minimal post-processing via border learning. In *ICCV*, 2017. 1

[43] Linjie Xing, Zhi Tian, Weilin Huang, and Matthew R Scott. Convolutional character networks. In *ICCV*, 2019. 2

[44] Xue-Hang Yang, Wenhao He, Fei Yin, and Cheng-Lin Liu. A unified video text detection method with network flow. In *ICDAR*, 2017. 2

[45] Xin Yi and Mark Eramian. Lbp-based segmentation of defocus blur. *TIP*, 2016. 3

[46] Fei Yin, Yi-Chao Wu, Xu-Yao Zhang, and Cheng-Lin Liu. Scene text recognition with sliding convolutional character models. *arXiv preprint arXiv:1709.01727*, 2017. 1

[47] Chengquan Zhang, Borong Liang, Zuming Huang, Mengyi En, Junyu Han, Errui Ding, and Xinghao Ding. Look more than once: An accurate detector for text of arbitrary shapes. In *CVPR*, 2019. 1

[48] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017. 1

[49] Ze-Yu Zuo, Shu Tian, Wei-yi Pei, and Xu-Cheng Yin. Multi-strategy tracking based text detection in scene videos. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 66–70. IEEE, 2015. 2