A Machine Learning Approach to Hypothesis Decoding in Scene Text Recognition

Jindřich Libovický[1], Lukáš Neumann[2], Pavel Pecina[1], Jiří Matas[2]

[1] Charles University in Prague,
Institute of Formal and Applied Linguistics, Czech Republic
[2] Centre for Machine Perception,
Czech Technical University in Prague, Czech Republic

Abstract. Scene Text Recognition (STR) is a task of localizing and transcribing textual information captured in real-word images. With its increasing accuracy, it becomes a new source of textual data for standard Natural Language Processing tasks and poses new problems because of the specific nature of Scene Text. In this paper, we learn a string hypotheses decoding procedure in an STR pipeline using structured prediction methods that proved to be useful in automatic Speech Recognition and Machine Translation. The model allow to employ a wide range of typographical and language features into the decoding process. The proposed method is evaluated on a standard dataset and improves both character and word recognition performance over the baseline.

1 Introduction

Scene Text Recognition (STR) is a computer vision task which aims to automatically localize all text areas in an image and to recognize (transcribe) their textual content. The problem has been receiving significant attention of the scientific community since the textual information is heavily present in real-world images with a large application potential. However, only manually assigned metadata is commonly available for image retrieval or content analysis. Manual annotation is costly or infeasible given the steadily rising data volumes.

STR also poses new problems for Natural Language Processing (NLP) as text in real-world images often consists of very few words or snippets without other textual context (see Figure 1a). Even a plain transcription can be quite difficult and the current state-of-the-art STR methods achieve character accuracy of only about 70% on standard datasets [1]. Additionally, interpretation of Scene Text can heavily depend on visual clues not present in the textual information itself (e.g., a meaning of the stand-alone word "visa" is completely different when written on a direction sign at an embassy and when written on a credit card), so novel joint techniques integrating computer vision and NLP are appropriate for such situations.

In this work, we integrate typographical and language features into the state-of-the-art end-to-end STR pipeline [2] and improve its accuracy by using structured machine learning approach inspired by hypotheses decoding in automatic speech recognition [3, 4] and machine translation [5, 6]. The proposed method is evaluated on the ICDAR 2013 dataset, a standard benchmark for STR evaluation [1].

The rest of this paper is structured as follows. Section 2 summarizes previous work in this field with focus on the hypotheses decoding algorithms. In Section 3, we describe and formalize the decoding problem we aim to solve in this paper. Section 4 presents the features we use in the models and methods we use for training the models. Section 5 describes the process of preparing the training data. The method is evaluated in Section 6 and the results discussed in Section 7. Section 8 concludes the work.

2 Related Work

For a comprehensive survey of text detection and localization in STR we refer the reader to work of Zhang et al. [7].

One criterion for categorization of STR methods is the requirement for prior manual text localization. The methods of Mishra et al. [8] and Novikova et al. [9] omit the localization phase and require a human annotator to first "cut out" all words and then recognize (transcribe) text of each of the manually cropped word images. The requirement for manual text localization makes the methods impractical for processing of larger datasets.

Another criterion is the requirement to know the set words that may appear in an image. The method of Wang et al. [10] is given a *lexicon* (a list of 50 to 500 words) for each processed image and aims at localizing one or more of the lexicon words in the image. The method achieves high precision in text recognition, but its applicability is limited by the requirement of having a fixed lexicon for each image prior to the recognition and therefore it cannot be used to acquire new textual data.

TextSpotter [11, 2] is to our knowledge the only lexicon-free method for STR which operates in an end-to-end setup, i.e., it implements both text localization and recognition and requires no manual annotation or lexicon to transcribe text in images. It first detects image regions corresponding to individual characters, joins them into text line hypotheses, and then recognizes the characters using an Optical Character Recognition (OCR) classifier.

The final stage of STR is usually string decoding, which disambiguates character recognition in the context of the entire strings. The hypotheses typically form a graph with vertices representing the chracter regions (there may be either one vertex with distribution over multiple characters or one vertex per character) and edges between the regions which follow each other in the transcription. The decoding itself can be approached in various ways.

Mishra et al. [8] use the Conditional Random Fields (CRFs) to decide between different segmentations. Each bounding box is represented only by its first-best character recognition and is assigned a unary feature – its classification score. Potentially neighboring character hypotheses are connected by binary factors encoding language model score. The feature weights are estimated empirically.

Roy et al. [12] first detect a line on which characters lie. Then a Hidden Markov Model (HMM) is used to decode the string from a set observed by sliding windows. This approach allows both multiple segmentation and multiple hypotheses per window, however by having most of the windows emitting empty output they loose possibility to use a language model score for adjacent characters.

A similar approach to the one presented in this paper is used by Shi et al. [13]. After a text area is detected, it is segmented into character bounding boxes. For each bounding box, a classifier produces several hypotheses which are then decoded (disambiguated) using a linear chain CRF. However, in this case the character segmentation is decided beforehand, and multiple transcription hypotheses a segment hypotheses are allowed.

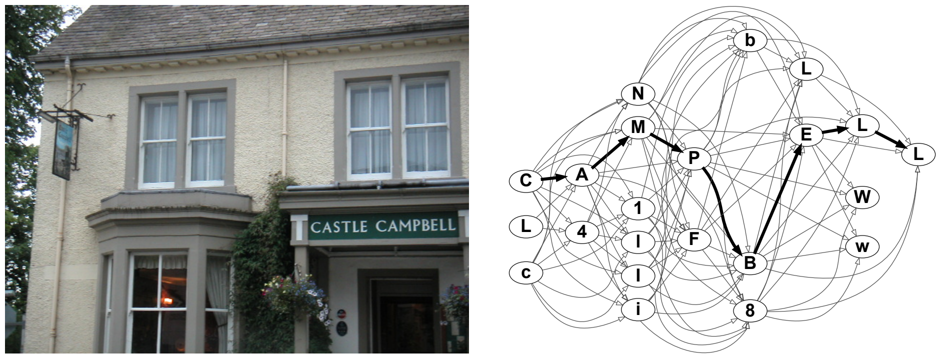
An end-to-end STR pipeline, PhotoOCR [14], uses a machine-translation-like beam search to explore all possible paths with pruning over the least probable. This approach is able to deal with both multiple hypotheses and multiple segmentations. In another end-to-end pipeline [15], similarly to our work, the structured perceptron is used for optimization of the weights in a dynamic programming decoding.

In case of recognition of words from a prior lexicon, the decoding can be constructed such that it could only produce the words listed in the lexicon. Novikova et al. [9] employs a trie-shaped weighted finite state automaton, Wang et al. [10] use dynamic programing for searching areas that may correspond to words from a dictionary. This produces a list of candidate locations which are filtered using a binary classifier.

Feild [16] recently came with two different approaches. One approach is string decoding realized as a parse of a probabilistic context free grammar for English syllables. In the second approach, she first identifies identical characters which necessarily must have the same transcription. The string is decoded using a HMM with a character bigram model for transition probabilities. The inference is done by integer linear programming which allows to ensures the previously detected character' identities.

3 Problem Description

In TextSpotter, each character appearing in an image may be detected several times (each detection corresponds to a different image segmentation) and each such detection may be assigned one or more character labels (transcription hypotheses) by the OCR module. Such an approach is beneficial, because it allows to keep multiple hypotheses for character segmentation and its labels for a later stage of the pipeline, where the final decision can be made by exploiting context of the character in the word (we define a *word* simply as a sequence of characters).



(a) (b)

Fig. 1. Scene text in a real-world image (a). Graph representing transcription hypotheses for the word "CAMPBELL" (the path representing the ground truth transcription is bolded) (b)

In this paper, the space of all word transcription hypotheses is modelled as a directed acyclic graph, where each vertex corresponds to a character segmentation with a character label (assigned by the OCR module). Two vertices are connected by an oriented edge iff the character associated with the first vertex can immediately follow the character associated with the second vertex (see Figure 1b). Therefore, each path unambiguously induces one possible transcription of the whole word. Finding the final word transcription can then be formalized as finding the maximum weighted path where each edge is assigned a weight indicating how likely the two characters follow each other in the character sequence of the word, i.e., we globally optimize values of local cost functions. For solving this problem we need to learn a function that estimates edge weights given a set of features that are assigned to each edge.

Formally, we define a transcription hypotheses graph $G = (V, E, s, t, l, \phi)$ where (V, E) is a Directed Acyclic Graph (DAG), $s \in V$ is the start node with no incoming edges, $t \in V$ is the target node with no outgoing edges, $l : V \setminus \{s, t\} \rightarrow [\mathtt{a} - \mathtt{zA} - \mathtt{Z}]$ is label assignment for the vertices, and $\phi : E \rightarrow \mathbb{R}^m$ is a feature function that assigns an m-dimensional real-valued feature vector to each edge. We denote the set of all the hypotheses graphs as $\mathcal{X}$.

We want find a path $\mathbf{y} = (y_1 = s, y_2, \ldots, y_{k-1}, y_k = t)$ from the start node to the target node such that concatenation of labels of the vertices on the path, $l(y_2), \ldots, l(y_{k-1})$, form the gound truth string. For that purpose we want to learn a function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ assigning each edge a number based on its feature vector such that the ground truth string will be the maximum weighted path $\mathbf{y}$ from s to t.

For each transcription hypothesis graph, we also define a second-order transcription hypothesis graph whose vertices represent edges in the original graphs (i.e., character bigrams) and edges between are pairs of adjacent edges in the original graph (i.e., represent character trigrams). This enable us to use features that describe properties of triplets of potentially adjacent characters. The size of such graphs is quadratic in size of the orignal graphs.

4 Proposed Method

The proposed method extends TextSpotter[1] by using additional features and employing machine learning for parameter training and decoding.

4.1 Features

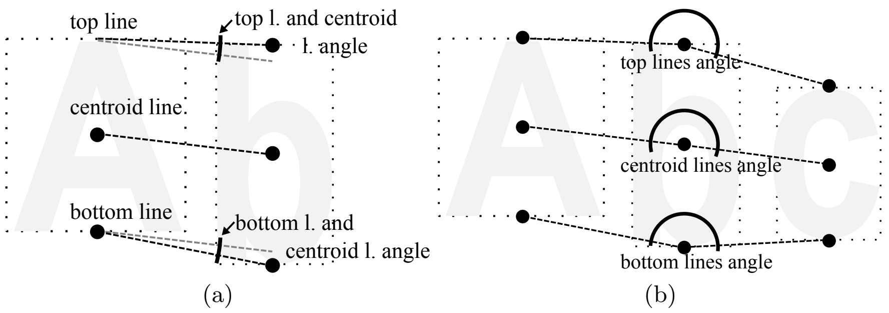


Fig. 2. Typographical features used in the first-order model (a) and the second-order (b) model.

TextSpotter employs a linear combination of four feature functions to infer edge weights (segmentation threshold compatibility, OCR confidence of the second character, confidence of the second character fitting the inferred text line and a heuristic language model score – character bigram probability modified by hand-crafted rules).

Our first-order (bigram) model extends the method by adding the following 20 typographical and language features:

- ratios of width, height, and area of the character on the edge,
- mutual angles of the top line, bottom line, and centroid line (see Figure 2a),
- conditional character bigram probability, and
- binary features coding character patterns (two digits, lower case + uppercase letter, both the same case, lower to upper case).

The second-order (trigram) model then employs the following 9 additional features capturing trigram properties:

- adjacent top lines, bottom lines, and centroid lines angles (see Figure 2b),
- adjacent spaces ratio,

[1] We used the current version of TextSpotter available at http://www.textspotter.org.

- conditional character trigram probability, and
- 4 binary features coding character patterns (digits only, `Xxx`, `xxx|XXX`, `xXX|xXx|xxX|XxX`, where `X` stand for an upper-case charecter and `x` for a lower-case letter).

All the ratios were computed as absolute values of the difference of logarithms. All features are standardized to have a zero mean and unit variance on the training data.

4.2 Model

In TextSpotter, the model parameters (linear combination coefficients) are tuned by a simple grid search. An alternative is to treat the problem as standard classification and train a classifier to predict for each edge how likely it is to lie on the ground truth path. Such classification is called local – the edges are scored independently from each other – those lying on the ground truth paths are used as positive training examples and the others as negative ones (we resampled the training data to have the same proportion of positive and negative examples). We examined several standard machine-learning algorithms implemented in WEKA [17]: Logistic Regression, Support Vector Machine (SVM) with various kernels, Random Forest, and multilayer Perceptron with various hidden layer configurations.

In local classification, the information about the final output is ignored during training. The constraints for the output structure (a graph path) apply in decoding (testing) phase only. A more appropriate solution is structured prediction where the same decoding algorithm is used during training, but the classification is global and allows the parameters to be optimized also with respect to the inference algorithm. To employ the structured prediction we need to formulate the problem as finding a structutre (in this case a path) which is maximal with respect to a function that is a dot product of a weight vector and a feature function of the structure, here the sum of the feature values along the path. This means:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \sum_{e \in \mathbf{y}} \phi(e)$$

where $\mathbf{w}$ is a learned m-dimensional weight vector. While training a prediction model we want to estimate the weight vector $\mathbf{w}$ where $\mathbf{w}$ is an m-dimensional weight vector optimized to maximize the number of training instances correctly classified.

We examined two state-of-the-art techniques for the weights optimization: the structured perceptron [18] and structured SVM approximated by the cutting plane algorithm [19].

The sructured perceptron [18] is a simple modification of the standard perceptron algorithm. The weight vector is iteratively updated by the difference of the feature vector of the currently estimated solution and the ground truth solution.

The structured SVM algorithm aims to optimize the weight vector such that the dot product is an upper bound estimate of a loss function, i.e., unlike the perceptron, it only distinguishes between partially and entirely incorrect solutions. A quadratic programming formulation capturing this requirement would demand exponentially many conditions for each of the training instances and its computation would be intractable. For this reason, we use an iterative approximate algorithm which finds the most violated conditions in each iteration and add them as constraints to the quadratic programming problem.

Based on the results of our preliminary experiments, we also added score from the best performing (the Random Forest) local edge classifier to the to feature vector for the structured learning. It aims to combine an advantage of capturing non-linear relations between the features in the local classifier and the knowledge of the inference algorithm during the learning procedure.

5 Data

The ICDAR 2013 dataset [1] was used to generate the training data for our experiments. The training set consists of 229 real-world images with 849 words in total. For each word, the ground truth annotation is given in the form of a bounding box coordinates and a reference transcription. For our purposes, we ignored all punctuation marks because its annotation is inconsistent across the dataset. The training data was generated by running the initial stages of the TextSpotter pipeline on each image to generate word transcription hypotheses graphs and then in each graph, the ground truth path representing the correct transcription is selected (if it exists) by matching graph vertices to the ground truth annotation of the word. A path is matched with the ground truth annotation iff

- the sequence of the character labels along the path corresponds to the reference transcription or its prefix of a length at least four, and
- all character centroids lie in the bounding box.

This process produced a total of 1607 graphs (including multiple graphs for the same word because the TextSpotter pipeline processes the same image several times using different visual transformations of the original image). We were able to match a subset of 812 graphs with the ground truth, out of which a random sample of 568 graphs was used to train the model and the remaining 244 graphs were used for intrinsic evaluation (see Section 6).

6 Evaluation

In the first experiment, we evaluate how well the model learns to find the ground truth path. A straightforward measure for the correctness of finding the longest path in a graph could be the Hamming loss (number of incorrectly used edges), or precision and recall of correctly selected edges. However, this measures cannot

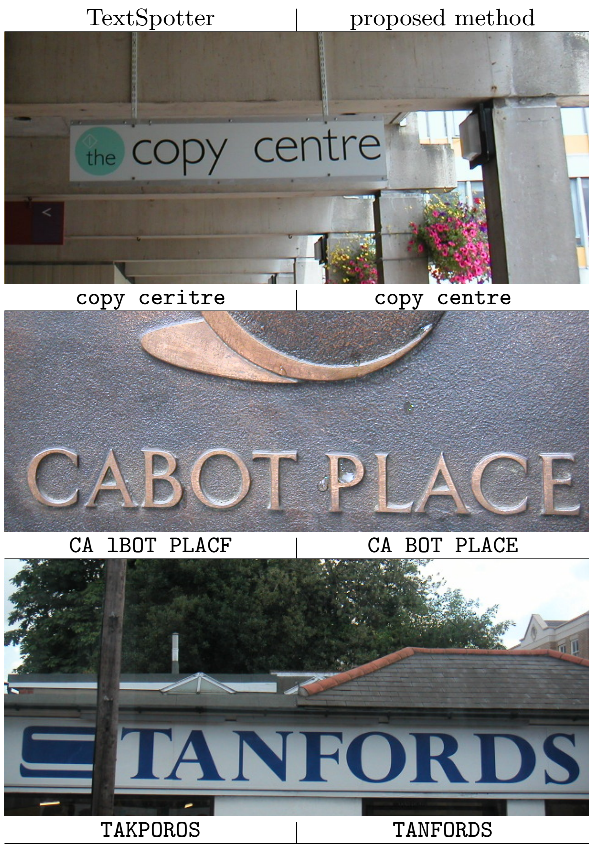


Fig. 3. Samples from the ICDAR 2013 dataset. Note the improvements of the proposed method over TextSpotter [2].

be compared between the graphs with bigram and trigram edges. Therefore, we only use the string based metrics, which were averaged over all graphs in the test set:

– Levenshtein (edit) distance $\bar{d}$ of the output string and the ground truth string;
– relative Levenshtein distance $\bar{d_r}$ of the output and ground truth strings, i.e., the edit distance divided by the length of the ground truth string, saying how likely is a character to be incorrect; and
– full string accuracy $\bar{a}$ – proportion of correctly selected paths.

.

The results of the intrinsic evaluation is provided in Table 1. The evaluated methods brought substantial improvement over the baseline method. The best results were achieved by the Random Forest classifier [20] trained locally on the

model	*order*	$\bar{d}$	$\bar{d_r}$	$\bar{a}$
TextSpotter	–	.647	.134	.685
Logistic Regression	1st	.701	.127	.561
SVM , Gauss. kernel	1st	.480	.094	.737
Multilayer Perceptron	1st	.471	.100	.754
Random Forest	1st	**.332**	**.068**	.807
Structured Perceptron	1st	.463	.092	.738
Structured SVM	1st	.439	.082	.750
Str. Perc + Rand. for.	1st	.377	.080	**.816**
Str. SVM + Rand. for.	1st	.377	.080	**.816**
Logistic Regression	2nd	.660	.121	.631
SVM , Gauss. kernel	2nd	.599	.118	.657
Multilayer Perceptron	2nd	.598	.115	.676
Random Forest	2nd	.398	.075	.779
Structured Perceptron	2nd	.488	.104	.701
Structured SVM	2nd	.402	.077	.775
Str. Perc + Rand. for.	2st	.504	.101	.725
Str. SVM + Rand. for.	2st	.398	.077	.779

Table 1. Results of the intrinsic evaluation (Levenshtein distance $\bar{d}$, relative Levenshtein distance $\bar{d_r}$, full string accuracy $\bar{a}$).

edges and with a structurally trained model that used the Random Forest output as a feature. Bringing more information to the model by using higher order graphs improved only the performance of the local linear regression and structured SVM classifier. The second-order graphs contain quadratically more edges which adds complexity to the learning algorithm, which may be a reason why it lead to a better result with large margin training but worse with Perceptron.

In the second experiment, we used the standard metrics (see Table 2) on the ICDAR 2013 dataset to evaluate the effect of replacing the hypothesis decoding method on the performance of the STR pipeline.

The ICDAR dataset is commonly used to evaluate performance of STR methods, but to our knowledge the only method to report performance in the end-to-end setup is TextSpotter [2] as other methods either focus solely on text localization or on cropped word recognition. The recent ICDAR 2013 Robust Reading competition [1] also only listed participants in these two limited setups.

The *text localization* measures how well the method is able to localize text areas in an image, the *character* and *word retrieval* measure the proportion of correctly transcribed characters (respectively words) in the dataset, when a character (a word) is considered to be correctly transcribed when it is localized correctly and the textual content is identical (using case-sensitive comparison) with the ground truth label. Most of the models substantially improved the recognition precision (the best result was achieved by the structured SVM), while the recall is only marginally worse. The combined model achieved very good results in the character recognition both in precision and recall, however it lowered the whole word recognition performance.

		text localization			character retrieval			word retrieval		
model	*order*	P	R	F_1	P	R	F_1	P	R	F_1
TextSpotter	–	.828	**.629**	**.715**	.786	.625	.696	.421	**.368**	.392
Logistic Regression	1st	.831	.600	.697	.769	.593	.670	.340	.340	.316
SVM , Gauss. kernel	1st	.822	.605	.697	.780	.601	.679	.387	.387	.359
Multilayer Perceptron	1st	.814	.574	.673	.797	.596	.682	.394	.394	.330
Random Forest	1st	.809	.577	.673	.810	.606	.693	.414	.345	.376
Structured Perceptron	1st	**.842**	.617	.710	.800	.626	.703	.424	.360	.389
Structured SVM	1st	.794	.585	.673	.821	.614	.703	**.425**	.364	**.393**
Str. Perc + Rand. for.	1st	.833	.605	.701	**.885**	**.689**	**.775**	.404	.344	.372
Str. SVM + Rand. for.	1st	.833	.605	.701	**.885**	**.689**	**.775**	.404	.344	.372
Logistic Regression	2nd	.843	.606	.705	.784	.604	.682	.387	.329	.356
SVM , Gauss. kernel	2nd	.808	.589	.681	.788	.604	.684	.403	.347	.373
Multilayer Perceptron	2nd	.829	.578	.681	.798	.590	.679	.387	.327	.355
Random Forest	2nd	.804	.570	.667	.808	.602	.690	.425	.353	.386
Structured Perceptron	2nd	.836	. 613	.707	.808	.624	.704	.425	.359	.389
Structured SVM	2nd	.802	.590	.680	.812	.617	.701	.418	.357	.385
Str. Perc + Rand. for.	2nd	.810	.605	.694	.808	.625	.705	.410	.359	.383
Str. SVM + Rand. for.	2nd	.820	.599	.692	.818	.626	.709	.404	.360	.381

Table 2. Results of the pipeline evaluation (precision P, recall R, and F_1 measure).

7 Discussion

The intrinsic evaluation showed a significant improvement over the baseline method, however the biggest effect on the whole STR pipeline is in improving precision of character recognition (8 percentage points improvement for the combined model). This can be attributed to a better language modelling where knowledge of character bigram (trigram) statistics can help to distinguish between similar characters, as seen in Figure 3. The character case pattern features also prevented the differently cased letters to appear within words. Neverthelss, the precision of the word recognition is improved only slightly, because of the relative strictness of the evaluation protocol (a word is considered as correctly recognized only when all its characters match the ground truth, using case-sensitive comparison).

The recall in all three STR metrics (localization, character and word retrieval) remains virtually unchanged compared to the baseline method, which is desired because an improved hypotheses decoding method cannot contribute to detect more characters in the earlier stages of the pipeline but it could incorrectly reject true characters.

The results of classifiers trained on the edges locally show that the classifiers with non-linear decision boundaries performed much better than the linear ones. The gain from the non-linearity is comparable from the gain of using the inference algorithm during the learning. This suggests that even better results could be achieved using a structured prediction method utilizing a non-linear decision boundary.

Using the trigram edges did not lead to any significant improvement in the decoding accuracy. We hypothesize that the additional information from the trigram features did not outweigh the added complexity of finding a path in quadratically bigger graph. However, choosing more informative character trigram features can lead to a different trade-off between the decoding complexity and the informativeness of the features.

8 Conclusions

We proposed a method for hypothesis decoding in a Scene Text Recognition pipeline. Our approach is based on structured prediction and allows to exploit a larger number of features. When plugged-in to an end-to-end STR system together with additional typographical and language features proposed in this work, it achieves a state-of-the-art precision for character and word recognition on the standard ICDAR 2013 dataset and brings a substantial improvement on the character level.

Except the already mentioned non-linear structured prediction methods, another improvements in the word decoding could be achieved by using some global features evaluating the produced words. Additional rescoring of a list word of hypotheses using a syllable based language model or a corpus based spell checking may be a way how to increase also the whole word recognition scores.

A natural follow-up is connecting the recognized words into longer logical segments and thus enabling to use the STR output as an input to Machine Translation and Information Retrieval. The knowledge of which words belong together and how they follow each other can provide more informative input for further processing than just bags of words and moreover it can provide further accuracy improvements to STR methods.

9 Acknowledgements

This research has been funded by the Czech Science Foundation (grant number P103/12/G084). Lukáš would also like to acknowledge the Google PhD Fellowship in Computer Vision and the Google Research Award.

References

1. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., de las Heras, L.P., et al.: Icdar 2013 robust reading competition. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, IEEE (2013) 1484–1493
2. Neumann, L., Matas, J.: On combining multiple segmentations in scene text recognition. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, IEEE (2013) 523–527
3. Ghoshal, A., Jansche, M., Khudanpur, S., Riley, M., Ulinski, M.: Web-derived pronunciations. In: Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on, IEEE (2009) 4289–4292

4. Bilmes, J.A.: Graphical models and automatic speech recognition. In: Mathematical foundations of speech and language processing. Springer (2004) 191–245
5. Daumé Iii, H., Langford, J., Marcu, D.: Search-based structured prediction. Machine learning **75** (2009) 297–325
6. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al.: Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Association for Computational Linguistics (2007) 177–180
7. Zhang, H., Zhao, K., Song, Y.Z., Guo, J.: Text extraction from natural scene image: A survey. Neurocomputing **122** (2013) 310–323
8. Mishra, A., Alahari, K., Jawahar, C.: Top-down and bottom-up cues for scene text recognition. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2687–2694
9. Novikova, T., Barinova, O., Kohli, P., Lempitsky, V.: Large-lexicon attribute-consistent text recognition in natural images. In: Computer Vision–ECCV 2012. Springer (2012) 752–765
10. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 1457–1464
11. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, California, US, IEEE (2012) 3538–3545
12. Roy, S., Roy, P.P., Shivakumara, P., Louloudis, G., Tan, C.L., Pal, U.: Hmm-based multi oriented text recognition in natural scene image. In: Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on, IEEE (2013) 288–292
13. Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S., Zhang, Z.: Scene text recognition using part-based tree-structured character detection. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE (2013) 2961–2968
14. Bissacco, A., Cummins, M., Netzer, Y., Neven, H.: Photoocr: Reading text in uncontrolled conditions. In: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE (2013) 785–792
15. Weinman, J., Butler, Z., Knoll, D., Feild, J.: Toward integrated scene text reading. Pattern Analysis and Machine Intelligence, IEEE Transactions on **36** (2014) 375–387
16. Field, J.: Improving Text Recognition in Images of Natural Scenes. PhD thesis, University Massachusetts Amherst (2014)
17. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter **11** (2009) 10–18
18. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, Association for Computational Linguistics (2002) 1–8
19. Joachims, T., Finley, T., Yu, C.N.J.: Cutting-plane training of structural svms. Machine Learning **77** (2009) 27–59
20. Svetnik, V., Liaw, A., Tong, C., Culberson, J.C., Sheridan, R.P., Feuston, B.P.: Random forest: a classification and regression tool for compound classification and qsar modeling. Journal of chemical information and computer sciences **43** (2003) 1947–1958