MSER-based Real-Time Text Detection and Tracking

Lluís Gómez and Dimosthenis Karatzas
Computer Vision Center
Universitat Autònoma de Barcelona
Email: {lgomez,dimos}@cvc.uab.es

***Abstract*—We present a hybrid algorithm for detection and tracking of text in natural scenes that goes beyond the full-detection approaches in terms of time performance optimization. A state-of-the-art scene text detection module based on Maximally Stable Extremal Regions (MSER) is used to detect text asynchronously, while on a separate thread detected text objects are tracked by MSER propagation. The cooperation of these two modules yields real time video processing at high frame rates even on low-resource devices.**

I. Introduction

Camera based scene text analysis applications targeted specifically for mobile and wearable devices is an interesting area of research receiving increasing attention [1], [2], [3], [4], [5], [6], [7]. Although the newly arrived products in the mobile device market (2013) feature high definition cameras of up to 12 mega-pixel sensors, and powerful quad-core processors, they still have many limitations in comparison with standard desktop computers: e.g. slow memories, limited floating-point support, and very small internal caches. This work attempts to bring real-time scene text detection to low-resource wearable devices. The contribution of this paper is mainly on the methodological aspect, presenting a simple but effective method for text detection and tracking suitable for devices with limited computational power. We demonstrate how a reasonably fast text detection method can be efficiently combined with tracking to give rise to real-time performance on such devices.

Text detection in video sequences differs from still images in many aspects and it is not a straightforward assumption that a method devised and trained on static text images would be applicable to video sequences. A key characteristic of video sequences is the temporal redundancy of text, which calls for tracking based processes taking advantage of past history to increase the stability and quality of detections. Keeping constant track of a text object throughout all the frames where it is visible is desirable for example to ensure a unique response of the system (e.g. translation, or text to speech conversion) for each distinct text, and also to be able to enhance the text regions [8], or to select the best frames in which they appear, before doing the full text recognition. Moreover, one can take advantage of the tracking process in order to obtain a real-time detection system, under the assumption that the scene does not change much from frame to frame. This yields an extra speed-up that can be exploited in see-though applications (e.g. Augmented Reality translation [6], [7] and augmented documents [9]) or street-view navigation [5].

The proposed method combines two separate modules: a text detection module based on a Multi-Script Scene Text Extraction algorithm [10], and a MSER-based tracking module [11]. As both detection and tracking modules are based on MSER they can be integrated symbiotically, improving robustness and providing a speed boost to the system. Real-time text detection is simulated by propagating in time the previously detected text-regions, until a new text detection takes place.

The novelty of the proposed method lies in the ability to effectively track text regions (establishing a pixel level segmentation of constituent text parts in every frame), not merely their bounding boxes as usually done in state-of-the-art tracking-by-detection algorithms [12], while providing a considerable speed-up in comparison to performing a full frame text detection on each frame. Moreover, the proposed method can deal with rotation, translation, scaling, and perspective deformations of detected text.

The system has been implemented for a mobile platform and the obtained results demonstrate state of the art performance at real-time frame rates. Our experiments show that MSER-based text tracking outperforms full-detection approaches, i.e. methods performing a full text detection each frame, in terms of computation cost while achieving similar accuracy rates.

In the next section we review briefly related state of the art. Section III describes the different aspects of the detection and tracking system. In section IV results are given and time performance is measured, before the paper closes with our conclusions in section V.

II. Related Work

Comprehensive surveys on camera based document analysis advances can be found in Jung *et al.* [13], Liang *et al.* [14], or Zhang and Kasturi [15], being the latter dedicated exclusively to text detection and tracking in video.

For the task of scene text detection in still images, the large number of published methods can be divided into texture-based and region-based approaches. To shortly summarize recent leading research on texture-based methods, Coates *et al.* [16] and Wang *et al.* [17] have proposed the use of unsupervised feature learning to generate the features for text versus non-text patch classification. Wang *et al.* [18] have built an end-to-end scene text recognition system based on a sliding window character classifier using Random Ferns, with features originating from the Histogram of Oriented Gradients (HOG) descriptor. On the other hand, among the many recently published region-based methods we can observe an increasing use of the Maximally Stable Extremal Regions (MSER) algorithm for character candidates detection. For example,

Chen *et al.* [19] obtain state-of-the-art performance with a method that determines the stroke width of edge-enhanced MSERs using the Distance Transform. The effectiveness of MSER is also exploited by Novikova *et al.* [20] and Merino *et al.* [3] among others, while Neumann and Matas [21] propose a region representation derived from MSER where character/non-character classification is done for each possible Extremal Region (ER). An interesting observation about state of the art text detection in still images is that there is a whole corpus of methods using MSER-related techniques that can eventually benefit from using MSER-tracking as we propose in this paper.

Relevant work on video based text detection and tracking can be categorized into batch-processing and online-processing methods. Within the former category, Li *et al.* [22] proposed a text tracking scheme where a SSD (Sum of Squared Differences) based correlation module was used to track the detected text between adjacent frames. Crandall *et al.* [23] proposed a method for caption text extraction where the motion vectors encoded in MPEG-compressed videos were used for tracking. Gllavata *et al.* [24] take advantage of temporal redundancy for detecting challenging text displayed against a complex background, by building a multi-frame clustering. Myers and Burns [25] propose a method to track planar regions of scene text with arbitrary 3-D rigid motion by correlating small patches and computing homographies on multi-frame blocks simultaneously. All the aforementioned text tracking methods were designed for off-line video processing, and thus are not directly applicable to a real-time system.

Regarding online-processing, Kim *et al.* [26] proposed a method that analyses the textural properties of text in images using a sliding window based classifier, and then locates and tracks the text regions by applying the continuously adaptive mean shift algorithm (CAMSHIFT) [27] on the texture classification results. Merino and Mirmehdi develop in [2] a real-time probabilistic tracking framework based on particle filtering, where SIFT matching is used to identify text regions from one frame to the next. Their region based text detection method is further extended in [3], where the authors present a head-mounted device for text recognition in natural scenes. A similar wearable camera system for the blind is presented in the work of Goto and Tanaka [4] where text strings are extracted using a DCT-based method [28] and then grouped into temporal chains by a text tracking method also based on particle filter. Minetto *et al.* [5] propose a method combining a region-based text detection algorithm [29] with a particle filter tracking module. Fragoso *et al.* [6] have developed an Augmented Reality (AR) see-through text translator where the initial text detection is done with help of the user, which has to tap on the screen near the center of the word of interest, then the plane in which text stands is tracked in real-time using efficient second-order minimization (ESM). Petter *et al.* [7] have extended this application with automatic text detection (not requiring the user interaction) based on Connected Components analysis on the Canny edge detector output.

From the methods described, only [2] [4] and [7] report comparable frame rates to our method proposal (between 7 fps. and 10 fps.), but in [2] and [4] processing times are calculated in a standard PC and not in low-resource devices. The key difference is that [2] and [4] make use of a full-detection strategy, performing a full text detection on every frame, in order to provide observations to the tracking system, while in our method the detection module is only performed periodically in a subset of the incoming frames. On the other hand, although [22] and [5] use a similar strategy to combine and merge text detections and tracked text regions, in our method the detection algorithm is executed asynchronously in a separate thread, thus not affecting to the overall speed of the system. An analogue multi-threaded tracking solution is also used by Takeda *et al.* [9] in a document retrieval AR system to display relevant information as an image superimposed in real-time over the camera captured frames.

Another important difference between the work presented here and the ones in [5], [22], [26], [6] is that our system is able to obtain an updated segmentation of tracked characters at every frame, despite not doing a full detection. Since we are not tracking just bounding box information but the text-regions themselves, we can take advantage of several segmentations of each character that would eventually lead to improved recognition accuracy.

Finally, while most of the described methods are constrained to detect horizontally aligned text [2] [4] [5] [7], and pure translational movement models [5] [22] [23], our method can deal with multi-oriented text and is able to track it under scale, rotation, and perspective distortions.

III. Text Detection and Tracking Method

In this Section we proceed to explain the text detection and tracking modules comprising our system, as well as the way they are integrated. As we make use of an already published method for the text detection module [10], this section will extend more on the tracking part and in the implementation details for the integration of detection and tracking, while providing a more compact description of the text detection algorithm.

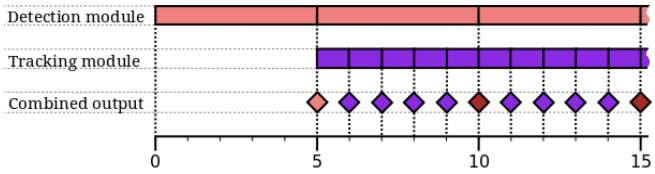

Fig. 1: Timeline view of the text detection and tracking modules' combination. Rectangles represent lasting processes in time and diamonds are the output of the system. The detection module creates the first estimates of text object positions, which are propagated to subsequent frames by the tracking module. Each time the detection module provides new results, a merging mechanism combines the detected and tracked objects in a unique output (maroon diamonds).

Figure 1 shows a timeline representation of the proposed method. The main idea behind our proposal is that even with a slow text detection method it is possible to achieve real-time performance by running it periodically while in parallel a fast tracking module takes care of propagating previous detections for the frames that are not processed by the detection module. The overall speed of the process really only depends

on the tracking module, therefore the system can be real-time. However, the speed of the text detector is nevertheless important, as it is the only source of new information to the system, and long times between consecutive detections could deteriorate substantially the overall performance.

A. Text Detection Module

The first step of the text detection module [10] is to extract Maximally Stable Extremal Regions (MSER) [30] in a given still frame in order to obtain text-region candidates. The extracted MSERs give rise to different possible grouping hypotheses, some groups being more perceptually meaningful than others. In order to create grouping hypotheses we take into account the spatial proximity of regions in relation to different manifestations of similarity, such as colour similarity, stroke width similarity, size similarity etc. For each such similarity definition a dendrogram is built using Single Linkage Clustering, in which each node represents a possible group of regions.

The assessment of the meaningfulness of a group is based on the *a contrario* approach to Hierarchical Clustering validity assessment proposed by F.Cao et al. [31], itself stemming from a more general perceptual organization framework [32]. The combination of different grouping hypotheses produced using distinct feature sets is performed using Evidence Accumulation Clustering [33].

At this stage the algorithm produces relatively pure text-only groups, with almost all text parts clustered together in separate groups. In order to filter non-text groups a combination of two Real Adaboost classifiers is used, one at the region level and the other at the group level. The detection process finishes with a simple post-processing step aiming to obtain text line level bounding boxes, based on an orientation independent collinearity test.

B. Tracking Module

Our tracking module is built upon the framework proposed by Donoser and Bischof in [11] where tracking of single MSERs in successive frames is posed as a correspondence problem within a window surrounding their previous location. The component tree of this small windows can be used as an efficient data structure to solve the correspondence problem: contains all the information needed to search the best matching region, and obviously can be computed much faster than for the whole image.

Figure 2 shows how finding the corresponding MSER between two consecutive frames can be done efficiently by constraining the search in two ways: searching only in the component tree of a small window (Figure 2b), and looking only in a sub-set of the tree levels (Figure 2c). These two constraints define two parameters for the tracking method: the size of the window (with respect to the query region size), and the levels interval to look for (with respect to the level of the query region).

Notice that the search process is done among all the regions in the component tree and would be able to find correct matches even when the target region has lost the stability criteria (e.g. appears blurred) in the consecutive frame.

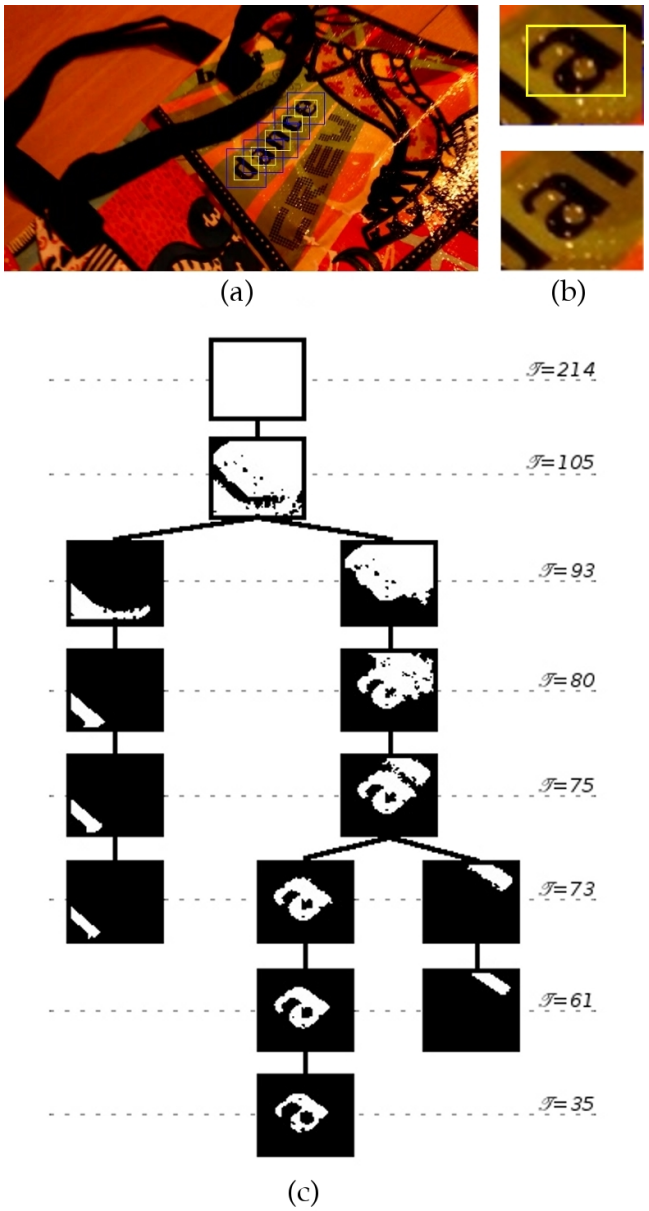

Fig. 2: Tracking of single MSER regions posed as search in the component tree of a small region of interest.

MSER-tracking has been used effectively for license plate [34] and hand [35] tracking using a weighted vector of simple features: mean gray value, region size, center of mass, width and height of the bounding box, and stability. A further extension in [36] makes use of novel shape descriptors in order to increment the robustness of the tracking by considering shape similarity.

The tracking module proposed in this paper differs from the work of Donoser and Bischof in two aspects by considering the specificities of text regions: 1) We use invariant moments as features to find correspondences, as a tradeoff between the fast computation of the simple features used in [11] and the robustness of the shape descriptors in [36]. 2) Here we consider groups of regions (text lines) instead of a single MSER as done in [34] and [35], and thus we can detect mismatches using

RANSAC when they do not fit an underlying line model.

1) MSER-tracking with incrementally computable invariant moments: The inclusion relation between regions in the component tree can be exploited to extract incrementally computable descriptors without any extra computational cost as proposed in [37] [21]. Geometric moments [38] [39] can be calculated in this way and thus result in a invariant descriptor that can be used efficiently by the MSER-tracking algorithm for matching. At each grow step of the MSER algorithm the raw moments up to order three are updated with constant complexity. Then, when required in order to find correspondences, those raw moments can derive the seven Hu's moments [38] and four Affine Invariant Moments [39] again with constant complexity.

Invariant moments have generally robust performance for rigid objects with simple contour shapes and simple transformations such as scaling, rotation, and affine transformations [40]. This is the case for text characters [41], assuming that they do not change much in shape between successive frames. We consider this compact descriptor to be tradeoff between of the simple feature based analysis in [11] and the integral shape descriptors used in [36], as they provide a richer representation than the former while being much less computationally expensive than the latter.

Moreover, in cases where invariant moments are prone to fail: e.g. for particularly weak shaped characters (e.g. letter "I"), partial occlusions, or motion blur mismatches, we can take advantage from two particularities in our scenario: first we have quite a constrained search along the component tree, and second we can exploit the group-level (text line) coherence in order to detect and reject mismatching correspondences via RANSAC.

2) Mismatch detection with RANSAC: Fitting a simple linear regression model against individually tracked MSERs using RANSAC allows to improve the whole text-line tracking because false correspondences do not affect the tracking process as they are eliminated by the RANSAC algorithm consensus set and thus not propagated to subsequent frames. This way the tracking module is able to robustly track text lines even when some of their characters are not correctly tracked. This outlier detection makes the method more robust in case of partial occlusion of the text line tracked.

Notice that RANSAC here is used as an outlier detection mechanism and not as an homography estimator as done in other tracking algorithms. Regions not correctly tracked are detected as outliers in a simple linear regression model and removed from the tracking system for the following frames. This process allows also to naturally stop the tracking process when the number of inliers for a text-line in a given frame is less than 3 regions.

C. Merging detected regions and propagated regions

Each time the detection module provides new results from a new full detection a merging mechanism, depicted with maroon diamonds in Figure 1, is needed in order to identify if the newly arrived detections are the same we are already tracking or not. This matching is done with the Hungarian algorithm by optimizing the one-to-one overlapping of the min. enclosing boxes of detected and tracked text lines.

Matched text lines are updated with the newly detected MSERs, thus regenerating the tracking process with new evidences, and may also recuperate regions that have been lost during the tracking process (i.e. detected as outliers and removed from the system). Not matched text-lines are treated as "first time viewed" objects and start their own tracking process from their initial locations.

IV. Experiments

We have evaluated our algorithm for the task of text detection and tracking in a dataset of synthetically generated video sequences. The dataset contains 10 sequences of 400 frames, with a resolution of 640×480 pixels, where still images from the ICDAR [42] and MSRRC [43] datasets are deformed iteratively with random rotation, translation, scale changes, and perspective transformations. Figure 4 shows two example sequences from the generated synthetic dataset. The main reason for the use of synthetic data in this series of experiments is that ground truth data can be created automatically, without any labelling effort.

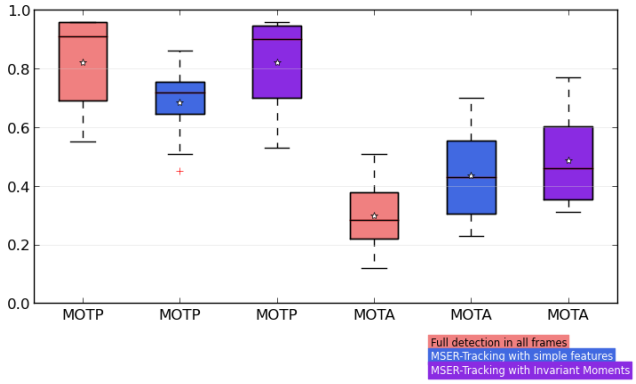

Fig. 3: Text Detection and Tracking performance comparison in the synthetic dataset.

Figure 3 shows the CLEAR-MOT metrics [44] performance comparison of the proposed method against two other approaches: Performing the full-detection on every frame, and MSER-tracking with simple features as in [11]. We can see how the proposed method outperforms the others both in tracking precision (MOTP) and accuracy (MOTA). MOTP is basically an average measure of the overlapping of correct detections and ground-truth text lines over the whole video sequence. We can see how this value is lower for the MSER-tracking with simpler features, indicating that the simple features are not enough to correctly track individual regions and thus produce less accurate bounding boxes at the text line level. The MOTA metric accounts for all errors made by the tracker: false positives, misses, and mismatches, over all the frames in a video sequence. The lower accuracy for the full-detection approach is due to missing objects, that the MSER-tracking is able to compensate by correctly propagating the detections of previous frames. In the MSER-tracking approaches the main source of MOTA errors are false positive detections provided by the detection module, which are propagated in time. On the other hand, the difference in accuracy between the two MSER-tracking methods indicate that invariant moments are more robust than the simpler feature vector.

Fig. 4: Still frame results from two of the synthetic image sequences (top) and two of the real scene image sequences (bottom) used for performance evaluation.

We further evaluated our method qualitatively in two sets of real scene image sequences: a set of 4 videos provided in [2], and a set of 10 videos obtained by the authors with a mobile device. The obtained results (see Figure 4) show that in general the system is able to detect and track the targeted text correctly dealing with rotation, translation, scaling, and perspective deformations. There are however errors of missing text components in the presence of motion blur and strong illumination changes. Nevertheless it is worth noting that in such situations the tracking module is still able to propagate some regions that would otherwise result in missed text by the detection module.

A. *Time performance*

In order to obtain time performance measurements and qualitative results for the task of text detection and tracking, we have implemented the proposed method using the Android development framework provided by the OpenCV [1] library, and tested it in a tablet computer with a 1.5GHz quad-core processor.

Table I shows average time performance measurements for each of the method modules. The average frame rate of the system is 15 fps., a value just slightly lower than the achievable from the Android camera service (without any image processing) for the concrete device used in the evaluation. Such a fast performance is achieved thanks to the negligible timestamp of the tracking module (40 ms. in average). On the other hand, the detection module running in the background needs about 1 second in average to provide localization results, this is roughly double the time to do the same processing in the main thread. Such a relatively low performance affects the system with a noticeable delay when new targets appear into the scene and to recover a missed target.

[1] http://opencv.org/platforms/android.html

TABLE I: Average time performance measurements.

	Android device		Standard PC	
	avg. time	fps.	avg. time	fps.
Text detection module (async.)	1041.01 ms.	n.a.	53.45 ms.	n.a.
Initial merging and tracking	125.57 ms.	7.96	17.11 ms.	58.44
Text tracking module	40.01 ms.	24.99	6.91 ms.	144.71

As shown in Table I the system has a variable frame rate: it is able to achieve a really high average frame rate (near 25 fps.) during the tracking process (this is most of the time), but once the asynchronous detection process finishes (every 1 second in average) the frame rate slows down to 8 fps. but just for one frame, as the new detections must be propagated and matched with the existing ones. All in all, the proposed method demonstrates real-time performance in low-resource devices with an average frame rate of 15 fps.

Time performance measurements for the tracking module would increase linearly with the number of tracked regions and the size of their search windows. In the evaluated sequences the average number of tracked regions is 21, covering around 10% of the input image size, which fits well with a realistic scene text detection scenario.

V. Conclusions and Future Work

We have presented a method for detection and tracking of scene text able to work in real-time on low-resource mobile devices. Although far from being a final solution, the proposed method goes beyond the full-detection approaches in terms of time performance optimization. The combination of text detection with a tracker, provides considerable stability, allowing the system to provide predicted estimates in cases where the detection module itself is not capable of returning a valid response. The use of MSER-tracking as an alternative, fast technique to provide simulated text detections for the

frames that are not processed by the full frame text detector proves to be an adequate solution, providing the system with enough information to continue tracking until the text detector returns updated positions.

The main limitation of the proposed method is the tracking degradation in presence of severe motion blur or strong illumination changes.

As in all tracking systems, the longer the full frame text detector takes to provide a result, the higher the uncertainty of the tracker will grow. At the same time, the response of the full frame text detector will be less reliable as more frames pass since the one being processed. Therefore, it is important that reasonably fast text detection methods are used in such a framework to ensure that tracking does not deteriorate rapidly.

As future work, further optimization of the text detection algorithm would improve the system in several ways, reducing the delay on initial detections and providing robustness to the tracking module with more frequent detection measurements.

On another hand, adding probabilistic estimation techniques (e.g. Kalman Filter) for each of the tracked regions would yield improved robustness in more challenging sequences e.g. total occlusions, severe motion blur, and strong illumination changes.

Acknowledgment

This project was supported by the Spanish projects TIN2011-24631 and 2010-CONE3-00029, the fellowship RYC-2009-05031, and the Catalan government scholarship 2013FI1126.

References

[1] I. Haritaoglu, "Scene text extraction and translation for handheld devices," in *CVPR*, 2001. 1

[2] C. Merino and M. Mirmehdi, "A framework towards realtime detection and tracking of text," in *CBDAR*, 2007. 1, 2, 5

[3] C. Merino-Gracia, K. Lenc, and M. Mirmehdi, "A head-mounted device for recognizing text in natural scenes," *CBDAR*, 2011. 1, 2

[4] H. Goto and M. Tanaka, "Text-tracking wearable camera system for the blind," in *ICDAR*, 2009. 1, 2

[5] R. Minetto, N. Thome, M. Cord, N. J. Leite, and J. Stolfi, "Text detection and tracking for outdoor videos," in *ICIP*, 2011. 1, 2

[6] V. Fragoso, S. Gauglitz, S. Zamora, J. Kleban, and M. Turk, "Translatar: A mobile augmented reality translator," in *WACV*, 2011. 1, 2

[7] M. Petter, V. Fragoso, M. Turk, and C. Baur, "Automatic text detection for mobile augmented reality translation," in *ICCV W.*, 2011. 1, 2

[8] H. Li and D. Doermann, "Text enhancement in digital video using multiple frame integration," in *ICM*, 1999. 1

[9] K. Takeda, K. Kise, and M. Iwamura, "Real-time document image retrieval on a smartphone," in *DAS*, 2012. 1, 2

[10] L. Gomez and D. Karatzas, "Multi-script text extraction from natural scenes," in *ICDAR*, 2013. 1, 2, 3

[11] M. Donoser and H. Bischof, "Efficient maximally stable extremal region (mser) tracking," in *CVPR*, 2006. 1, 3, 4

[12] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *CVPR*, 2013. 1

[13] K. Jung, K. I. Kim, and A. K. Jain, "Text information extraction in images and video: a survey," *Pattern Recognition*, 2004. 1

[14] J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: a survey," *IJDAR*, 2005. 1

[15] J. Zhang and R. Kasturi, "Extraction of text objects in video documents: Recent progress," in *DAS*, 2008. 1

[16] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng, "Text detection and character recognition in scene images with unsupervised feature learning," in *ICDAR*, 2011. 1

[17] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *ICPR*, 2012. 1

[18] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *ICCV*, 2011. 1

[19] H. Chen, S. Tsai, G. Schroth, D. Chen, R. Grzeszczuk, and B. Girod, "Robust text detection in natural images with edge-enhanced maximally stable extremal regions," in *ICIP*, 2011. 2

[20] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky, "Large-lexicon attribute-consistent text recognition in natural images," in *ECCV*, 2012. 2

[21] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *CVPR*, 2012. 2, 4

[22] H. Li, D. Doermann, and O. Kia, "Automatic text detection and tracking in digital video," *IEEE Trans. IP*, 2000. 2

[23] D. Crandall, S. Antani, and R. Kasturi, "Extraction of special effects caption text events from digital video," *IJDAR*, 2003. 2

[24] J. Gllavata, E. Qeli, and B. Freisleben, "Detecting text in videos using fuzzy clustering ensembles," in *ISM*, 2006. 2

[25] G. K. Myers and B. Burns, "A robust method for tracking scene text in video imagery," *CBDAR*, 2005. 2

[26] K. I. Kim, K. Jung, and J. H. Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," *IEEE Trans. PAMI*, 2003. 2

[27] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," in *Proc. WACV*, 1998. 2

[28] H. Goto, "Redefining the dct-based feature for scene text detection," *IJDAR*, 2008. 2

[29] R. Minetto, N. Thome, M. Cord, J. Fabrizio, and B. Marcotegui, "Snoopertext: A multiresolution system for text detection in complex visual scenes," in *ICIP*, 2010. 2

[30] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, 2004. 3

[31] F. Cao, J. Delon, A. Desolneux, P. Musé, and F. Sur, "An a contrario approach to hierarchical clustering validity assessment," INRIA, Tech. Rep., 2004. 3

[32] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*. Springer-Verlag, 2008. 3

[33] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *PAMI*, 2005. 3

[34] M. Donoser, C. Arth, and H. Bischof, "Detecting, tracking and recognizing license plates," in *ACCV*, 2007. 3

[35] M. Donoser and H. Bischof, "Real time appearance based hand tracking," in *ICPR*, 2008. 3

[36] M. Donoser, H. Riemenschneider, and H. Bischof, "Shape guided maximally stable extremal region tracking," in *ICPR*, 2010. 3, 4

[37] J. Matas and K. Zimmermann, "A new class of learnable detectors for categorisation," in *Image Analysis*, 2005. 4

[38] M.-K. Hu, "Visual pattern recognition by moment invariants," *Trans. on IRE*, 1962. 4

[39] J. Flusser and T. Suk, "Pattern recognition by affine moment invariants," *Pattern Recognition*, 1993. 4

[40] D. Zhang and G. Lu, "Evaluation of mpeg-7 shape descriptors against other shape descriptors," *Multimedia Systems*, 2003. 4

[41] J. Flusser and T. Suk, "Affine moment invariants: a new tool for character recognition," *Pattern Recognition Letters*, 1994. 4

[42] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura *et al.*, "Icdar 2013 robust reading competition," in *ICDAR*, 2013. 4

[43] D. Kumar, M. Prasad, and A. Ramakrishnan, "Multi-script robust reading competition in icdar 2013," in *MOCR Workshop*, 2013. 4

[44] B. Keni and S. Rainer, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP JIVP*, 2008. 4