SNOOPERTRACK: TEXT DETECTION AND TRACKING FOR OUTDOOR VIDEOS

R. Minetto $^{(1,2)}$, *N. Thome* $^{(1)}$, *M. Cord* $^{(1)}$, *N.J Leite* $^{(2)}$, *J. Stolfi* $^{(2)}$

(1) Universite Pierre et Marie Curie, UPMC-Sorbonne Universities, LIP 6, 4 place Jussieu, 75005, Paris, France

(2) University of Campinas, UNICAMP, Av. Albert Einstein, 1251, Campinas, Brazil

ABSTRACT

In this work we introduced *SnooperTrack*, an algorithm for the automatic detection and tracking of text objects — such as store names, traffic signs, license plates, and advertisements — in videos of outdoor scenes. The purpose is to improve the performances of text detection process in still images by taking advantage of the temporal coherence in videos. We first propose an efficient tracking algorithm using particle filtering framework with original region descriptors. The second contribution is our strategy to merge tracked regions and new detections. We also propose an improved version of our previously published text detection algorithm in still images. Tests indicate that SnooperTrack is fast, robust, enable false positive suppression, and achieved great performances in complex videos of outdoor scenes.

Index Terms— text detection, text tracking, particle filtering.

1. INTRODUCTION

Automatic information extraction is a topic of great interest to systems as Google Street View [1] and iTowns [2] that generates a huge amount of images. Recently, a lot of advances in text detection algorithms to outdoor images have been described [1, 2, 3]. However, many data sources are digital videos, and in such case the texts trajectories along the video can be more useful to decrease the number of annotations (e.g. one text entry per trajectory instead one per frame) and to be used as input in super-resolution OCRs.

The purpose of this work is to improve the performances of still image text detection algorithms for videos. This problem has many applications such as intelligent GPS-based navigation aids, automatic indexing of video libraries, etc.

For text detection in still images, two general approaches for text detection have been proposed: *bottom-up*, consisting of a character identification step followed by grouping of adjacent characters into texts regions as Hinnerk Becker [2]; and *top-down*, consisting of a search for regions with text-like appearance, followed by segmentation of those regions into characters as Chen et al [2]. Some recent approaches [4, 5] try to combine the advantages of bottom-up and top-down approaches as our method *SnooperText* [6].

For text tracking algorithms, a survey has been proposed by Junt et al. [1]. However, most of these algorithms are designed to specific purposes in well controlled environments, e.g. detecting and tracking text information artificially inserted as subtitles in news, clips and movies. Gllavata et al. [7] and Qian et al. [8] described text tracking algorithms that use the block motion information of MPEG video to match text regions detected in successive frames. In 2010, Na and Wen [9] proposed a text tracking algorithm that uses SIFT features to identify matching text regions. Their algorithm may be considerably expensive since it allows Euclidean deformations (including rotation, translation, and scaling).

In this work, we describe an algorithm called *SnooperTrack*, for automatic detection and tracking of text objects in videos of outdoor scenes. In such videos, many different text regions such as store and street names, traffic signs, advertisements, etc, may be visible. Then, the main challenge faced in this work is the definition of reliable text trajectories to such text objects, considering that detection and tracking algorithms are prone to several failures. Common failures modes of text detection algorithms include: *false negative* (failure to detect a text object that is legible in the frame); *false positive* (mistaking a non-text region for a text one); *splitting* of a single text region into several boxes; *joining* of distinct text regions into a single box. Tracking algorithms often have *drift* and *jitter* (cumulative and random errors in the size and position of the tracked bounding boxes) and *feature lost*.

The novel approach combines detection and tracking strengths. The main contributions of this paper are: (1) an algorithm to efficiently determine the trajectories of text regions across successive video frames by a particle filter (specifically designed for text tracking purposes); (2) a method to merge the detected text regions (found by our *SnooperText* algorithm for text detection in still image) with those previously tracked; (3) a improved text validation process of *SnooperText* by using descriptors based on histogram of gradients.

Tests are reported on a dataset of outdoor videos (groundtruth XML files publicly available) to validate the *SnooperTrack* strategy.

2. STATEMENT OF THE PROBLEM

A *text object* is any string of two or more letters printed or engraved on a physical object that are easily recognizable and readable in the video. We are primarily concerned with texts written horizontally in the Roman alphabet or any of its variants, in plain print-style (rather than cursive or ornate) fonts.

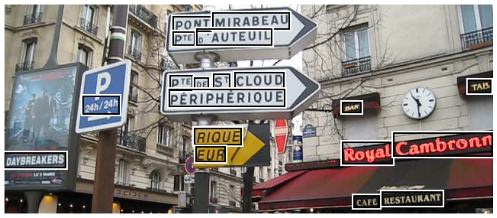

Fig. 1. The expected output of a text detection algorithm.

By *text detection* we mean identifying the approximate location and extent of *text regions*, the visible projection of text objects,

Contact: minettor@poleia.lip6.fr. This work was partially supported by FAPESP (Phd grant 07/54201-6), CAPES/COFECUB 592/08 and ANR 07-MDCO-007-03.

[1]maps.google.com/help/maps/streetview

[2]http://www.itowns.fr

in each video frame. In our algorithm, each text region is represented by a rectangular box r_i^j enclosing the j^{th} text image detected in frame i. See figure 1. By *text tracking* we mean identifying which detected text regions in successive frames are images of the same physical text object. We represent this information as a list $\pi_1, \pi_2, \ldots, \pi_m$ where each π_i is a relation between the text rectangles detected in frame $i-1$ and those detected in frame i. Specifically, a pair (r_{i-1}^k, r_i^j) is in the relation π_i iff the rectangles r_{i-1}^k and r_i^j are believed to belong to the same physical text object. These relations partition the set of all detected text regions in all frames into one or more connected subsets, that define the presumed trajectories of physical text objects along the video.

3. SNOOPERTRACK

Our algorithm for text-detection and tracking, which we call *SnooperTrack*, is outlined as Algorithm 1 below.

Algorithm 1 SNOOPERTRACK $(n, \mathbb{V}, K)$

```
1. $T_0 \leftarrow$ TEXTDETECT $(\mathbb{V}_0)$;
2. For each $i \in \{1, \ldots, n-1\}$ do
3.     $E \leftarrow \emptyset$;   $\phi \leftarrow \emptyset$;
4.     For each $r \in T_{i-1}$
5.         $v \leftarrow$ TRACK $(r, \mathbb{V}_{i-1}, \mathbb{V}_i)$;
6.         If $v \neq \emptyset$ then
7.             $E \leftarrow E \cup \{v\}$;
8.             $\phi \leftarrow \phi \cup \{(r, v)\}$;
9.
10.    If $i \in K$
11.        $D \leftarrow$ TEXTDETECT $(\mathbb{V}_i)$;
12.        $(T_i, \pi_i) \leftarrow$ MERGE $(D, E, \phi, \mathbb{V}_i)$;
13.    else
14.        $(T_i, \pi_i) \leftarrow (E, \phi)$;
15.
16. Return $T, \pi$.
```

The Algorithm 1 operates on a video $\mathbb{V}$ with n frames $\mathbb{V}_0, \mathbb{V}_1, \ldots, \mathbb{V}_{n-1}$. Since text detection algorithms are computationally expensive, we apply them (in steps 1 and 11) only to selected subset of *key frames* of the input video, specified by a given set $K \subseteq \{0, \ldots, n-1\}$ of frame indices that includes 0. The detection strategy chosen in this paper is detailed in section 3.1. In addition, for every frame $\mathbb{V}_i$ except $\mathbb{V}_0$, we use a track procedure (in step 5) (designed specifically to text) to generate tentative regions in frame $\mathbb{V}_i$ for the text objects that were determined in frame $\mathbb{V}_{i-1}$. The way that text regions are tracked is detailled in section 3.2. In step 12, the tracked regions and the regions found by the text detector are merged and pruned to obtain the regions of frame $\mathbb{V}_i$ (see section 3.3). As part of this merge, we compute the tracking relation π_i that associates regions in frame $\mathbb{V}_{i-1}$ and $\mathbb{V}_i$. The output of the algorithm are the text regions r_i^j, and the inferred tracking relations π_i with $i = 1, \ldots, n-1$.

3.1. Text Detection

The TEXTDETECT procedure used in steps 1 and 11 is based on the *SnooperText* algorithm described in 2010 by Minetto *et. al.* [6]. SnooperText includes a bottom-up *hypothesis generation* phase that uses an image segmentation step dedicated to identify candidate character regions. These regions are then grouped into *candidate text regions* using geometric criteria such as alignment and size similarity. The algorithm is applied at multiple image scales in order to deal with extreme situations encountered in urban context: huge character size variations, background clutter, *etc*. This bottom-up step is followed by a *hypothesis validation* phase that analyzes the merged characters globally to verify that the formed text candidates are indeed "text-like".

Improvement of the text validation step

To characterize a text region r in a image $\mathbb{I}$, we use a texture descriptor based on the histogram of oriented gradients [10] denoted by $HOG(e, \mathbb{I})$, that captures the distribution of local intensity gradients directions (or edge directions) in the region. Complex objects typically have different HOGs in different parts (*e.g.* humans with different gradient orientation distributions in the head, torso and leg regions). Therefore, in many applications, the candidate region is partitioned into an array of $N \times M$ *cells*, and a HOG is computed separately for each cell and normalized to unit sum. This standard HOG tiling has been directly applied in the original SnooperText algorithm [6]. However, we propose here to adopt a different tiling that is more adapted to text regions. As observed by Chen and Yuille [11], the top, middle and bottom parts of Roman text regions have distinctive distributions of edge directions. Therefore, we choose here to split the text region into those three regions and compute a separate 18-bin HOG for each one. The concatenation of those three HOGs is taken to be the descriptor of the full region.

For text validation, we trained a SVM classifier with the HOG descriptors extracted from negative and positive text samples. The SVM classifier uses a Gaussian chi2 kernel, with cross-validation to optimize its standard deviation parameter. This classifier, whose output we will denote by $C(r, \mathbb{I})$, is used in the text detection (hypothesis validation) and in the procedure TRACK (see section 3.2).

3.2. Tracking

In the proposed approach, the goal of the tracking step is two-fold.

First, it is dedicated to improve the detection accuracy by taking advantage of the temporal coherence. The TRACK routine takes a text region r previously determined in a frame $\mathbb{I}$, and estimates its position v in the subsequent frame $\mathbb{J}$. This procedure is achieved using a *particle filter (PF)* approach [12, 13] that proves to be efficient and robust to background clutter. Thus, each text box r_i^j is represented by the following state vector (x, y, w, h), where x, y are the coordinates of the top left corner in frame i, and w, h are its dimensions. We use a simple first order motion model to propagate the random regions ('particles'), that are further re-sampled by bayesian bootstrap or Importance Sampling. Let us consider two regions e in $\mathbb{I}$ and d in $\mathbb{J}$, the observation model is defined by the Battacharyya's similarity coefficient of the respective histograms of oriented gradients descriptors:

$$\mathrm{dist}(e, \mathbb{I}, d, \mathbb{J}) = \left(1 - \sum_{i=1}^{n} \sqrt{h(i)\ \hat{h}(i)}\right)^{1/2} \quad (1)$$

where $h = HOG(e, \mathbb{I})$, $\hat{h} = HOG(d, \mathbb{J})$ and n is the number of bins in the histogram.

Another requirement of the tracking procedure is the ability to reject regions that have been wrongly identified as text by the detection step. For that purpose, we propose to build and update a spatiotemporal descriptor for each tracked region. Therefore, once the most similar candidate v has been selected, its score s is set to a measure of how "text like" it is. This measure is an affine combination of the score of r (previously computed) and the output of the text classifier over v and is given by the formula

$$s_v = (1 - \rho) s_r + \rho C(v, \mathbb{J}) \quad (2)$$

where the text classification weight C is computed as described in section 3.1. The parameter ρ was set as 0.4 to take into account the confidence of the past iterations.

3.3. Merging

The Merge routine aims at combining detection and tracking outputs. It looks for an optimal partial matching between two lists D, E of text regions found by the text detector and track procedures and a partial function ϕ that maps regions E^{-1} in the previous frames to the corresponding regions in E. It outputs a subset T of $D \cup E$ which are presumed to be an optimal set of pairwise disjoint regions; and a tracking relation π that pairs the regions of E^{-1} with the regions in T. The strategy is detailed in Algorithm 2.

Algorithm 2 MERGE $(D, E, \phi, \mathbb{I})$

```
1. $T \leftarrow \emptyset$;  $\pi \leftarrow \emptyset$;
2. For each $e \in E$
3.     For each $d \in D$
4.         $P_{e,d} \leftarrow prob(e, d, \mathbb{I})$;
5.
6. $\alpha \leftarrow$ HUNGARIANALGORITHM $(p)$;
7. For each $e \in E$
8.     If there is pair $(e, d) \in \alpha$
9.         $s_e \leftarrow 1$;
10.    If $s_e > 0$
11.        $T \leftarrow T \cup \{e\}$;  $\pi \leftarrow \pi \cup \{(\phi^{-1}(e), e)\}$;
12.    For each $d \in D$
13.        If there is no pair $(e, d) \in \alpha$
14.            $s_d \leftarrow 1$;  $T \leftarrow T \cup \{d\}$;
15.
16.
17. Return $T, \pi$;
```

The decision of whether a region t in the set $D \cup E$ in frame i comes from the same text object as the region d in frame $i-1$ should take into account the geometry of the two regions and their contents. We assume that, if the regions belong to the same text object, the differences in position and size have a two dimensional normal distributions with zero mean. That is: $p_{pos}(e, d) = \frac{1}{2\pi\sigma^2} \exp(-\frac{(x_e - x_d)^2 + (y_e - y_d)^2}{2\sigma^2})$ and $p_{size}(e, d) = \frac{1}{2\pi\zeta^2} \exp(-\frac{(w_e - w_d)^2 + (h_e - h_d)^2}{2\zeta^2})$. To measure the appearance similarity we use the dissimilarity criteria described in equation 1 and we assume that the squared distance between two histograms has an exponential distribution: $p_{app}(e, d, \mathbb{I}) = \exp(-\frac{\text{dist}(e, \mathbb{I}, d, \mathbb{I})^2}{2\xi^2})$. The final probability $prob(e, d, \mathbb{I})$ (step 4) is the product of three probabilities:

$$prob(e, d, \mathbb{I}) = p_{pos}(e, d) \cdot p_{size}(e, d) \cdot p_{app}(e, d, \mathbb{I}) \quad (3)$$

The matching is performed by the Hungarian Algorithm, Step 6, which takes a matrix $P_{e,d}$, which tells the probabilities that region e is the same as region d; and returns the optimal assignment of row elements (regions of E) to column elements (regions of D) as a list of pairs α. Only the assignments with probabilities above a threshold λ ($\lambda > 0.05$) are considered (this is to avoid situations where candidates position and size could be similar but the appearance no or vice-versa, e.g. video cuts).

In step 11 the procedure selects only those tracked text regions that were matched or have a positive score (step 10). However, as the text regions e and d may differ by some pixels, a fusion between their coordinates may be performed in step 11. Moreover, if the set D has text joining, they should be broken into separate text regions according to some heuristic, e.g. as the one described by Epshtein [3]. The remaining tracked regions not matched (or not text-like) are discarded. In steps 13–14 those detected text regions (which do not match tracked regions) are added to T.

4. EXPERIMENTS

4.1. Videos, Evaluation and Settings

We evaluated SnooperTrack on 5 real videos with frame size of 640×480 pixels. Some videos have several text regions, sometimes affected by natural noise, distortion, blurring, hard illumination changes and occlusion. They are listed in table 1 and publicly available, with XML groundtruth files, in our web site [3]. To evaluate the algorithms performances we used the well-known metrics precision p, recall r and f error defined according Lucas in [2]. The final performances were obtained by averaging the precision, recall and f error over all the frames. The parameters used in this work, fixed to all videos, were: PF with 200 particles, with deviations $\sigma = 16$, $\zeta = 1$ and $\xi = 20$ (equations p_{pos}, p_{size} and p_{app}); text detection with a step of 20 frames.

4.2. Results and Discussion

In this section we compare the results of an approach based on purely text detections performed by the SnooperText called here SnooperText* (it can solve the same problem as SnooperTrack if we match it outputs with the merge routine) against SnooperTrack.

Comparing tables 1 and 2 we can see that SnooperTrack with HOG histogram is better than SnooperText* in 4 videos. The worst result of SnooperTrack, in video v3, was due to the fast camera changes which caused sudden text regions appearance and disappearance (some get occluded even before they were detected).

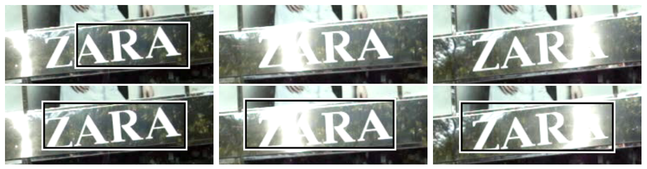

Fig. 2. Top: output of the SnooperText detector to three different frames, showing failures due to illumination changes; Bottom: output of SnooperTrack on the same frames.

The poorly detection results of SnooperText* to the video v5 was due to the hard solar reflections. See figure 2 (top). On the other hand, SnooperTrack successfully tracked the text in this extreme situation. See figure 2 (bottom). This is because our text validation step, even in the absence of newly detections, was robust enough to verify that the region being tracked was indeed text-like, and thus kept it weight positive along the execution. This reflects the good precision in table 2 (HOG histogram). Moreover, lots of detected false positives, in all videos, were successfully eliminated by our text validation step during the tracking. See figure 4. Thus, it was essential to SnooperTrack be the best in recall to all videos.

We also compared our HOG descriptor against an HSV color histogram (often used in particle filters) with 110 bins as used by Okuma [14]. As shown in table 2, HSV histogram is not so good as HOG descriptor to text tracking. The main sources of problems in

[3]Video dataset: www.liv.ic.unicamp.br/∼minetto/datasets/text/VIDEOS/

			SnooperText		
Video	#XMLs	#Text Objs.	p	r	f
v1	800	2	0.58	0.73	0.62
v2	1089	2	0.59	0.69	0.62
v3	206	18	**0.72**	**0.53**	**0.59**
v4	400	3	0.61	0.51	0.54
v5	1250	1	0.37	0.42	0.38

Table 1. SnooperText [6] detection performance.

HSV histograms to text tracking (unlikely to occur with HOG) are: tracking confusion, ease of nearby text regions have the same color histogram; location instability, shifts inside the same text region may lead to the same histogram causing tracking jitter. See figure 3.

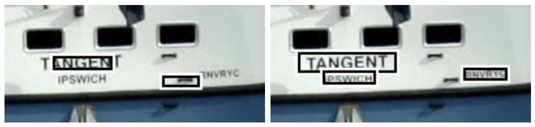

Fig. 3. SnooperTrack results in a frame of video v4 using HSV histogram (left) and with HOG descriptor (right).

SnooperTrack								
	HSV histogram				HOG histogram			
Video	p	r	f	t(s)	p	r	f	t(s)
v1	**0.62**	0.52	0.53	0.11	0.55	**0.80**	**0.63**	0.19
v2	0.55	0.65	0.59	0.38	0.57	**0.74**	**0.64**	0.45
v3	0.52	0.37	0.43	0.65	0.60	**0.53**	0.56	0.88
v4	0.41	0.31	0.34	0.08	**0.73**	**0.70**	**0.71**	0.15
v5	0.31	0.32	0.31	0.12	**0.60**	**0.70**	**0.63**	0.55

Table 2. SnooperTrack performance and time execution on a machine Intel© Quad Core 2.5GHz, 8GB of RAM, coded in JAVA© without optimizations.

Finally, the advantages of SnooperTrack can be summarized in: time execution, (e.g. text detection is measured in seconds per frame while the tracking time in milliseconds as listed in table 2); tracking consistency and false positives suppression which lead to an increased precision and recall.

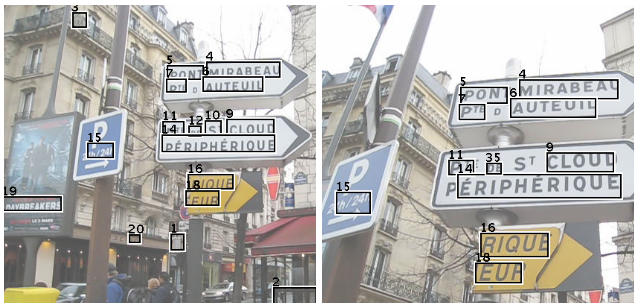

Fig. 4. SnooperTrack output using HOG. Portions of frames 1 and 73 of video v3 (after 4 text detections) The number indicates regions that are matched by the tracking relations π. Note that several false positives detected in frame 1 were eliminated (regions 1, 2, 3 and 20). However two true positives were also eliminated (regions 10 and 12) and one of them lost but detected again in a later frame (region 35).

5. CONCLUSION

We described an algorithm called *SnooperTrack*, based on text detection, text validation with histogram of oriented gradients to eliminate false positives and tracking with a particle filter designed to text tracking purposes, to recover the trajectory of text regions across successive video frames. Tests indicate that SnooperTrack is faster and more robust than an approach based on successive text detections performed by the state of the art SnooperText (a text detector to static images published in 2010).

6. REFERENCES

[1] K. Jung, K.I Kim, and Jain. A.K, "Text information extraction in images and video: A survey," *Elsevier - Pattern Recognition*, vol. 37, pp. 977–997, 2004.

[2] S.M. Lucas, "ICDAR 2005 Text locating competition results," in *ICDAR*, 2005, pp. 80–84.

[3] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," *IEEE CVPR*, pp. 886–893, 2010.

[4] Y.F. Pan, C.L. Liu, and X. Hou, "Fast scene text localization by learning-based filtering and verification," in *IEEE ICIP*, 2010.

[5] T.N. Dinh, J. Park, and G. Lee, "Text localization using image cues and text line information," in *IEEE ICIP*, 2010.

[6] R. Minetto, N. Thome, M. Cord, J. Fabrizio, and B. Marcotegui, "Snoopertext: A multiresolution system for text detection in complex visual scenes," in *IEEE ICIP*, 2010, pp. 1–4.

[7] J. Gllavata, R. Ewerth, and B. Freisleben, "Tracking text in mpeg videos," in *ACM international conference on Multimedia*, 2004, pp. 240–243.

[8] X. Qian, G. Liu, H. Wang, and R. Su, "Text detection, localization, and tracking in compressed video," *Elsevier - Image Communication*, vol. 22, pp. 752–768, 2007.

[9] Y. Na and D. Wen, "An effective video text tracking algorithm based on sift feature and geometric constraint," in *Advances in Multimedia Information Processing*, 2010, pp. 392–403.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE CVPR*, 2005, pp. 886–893.

[11] X. Chen and A.L Yuille, "Detecting and reading text in natural scenes," in *IEEE CVPR*, 2004, pp. 366–373.

[12] Michael Isard and Andrew Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[13] Nicolas Thome, Djamel Mérad, and Serge Miguet., "Learning articulated appearance models for tracking humans: a spectral graph matching approach," in *Signal Processing: Image Communication*, 2008, vol. 23, pp. 769–787.

[14] K. Okuma, A. Taleghani, N. de Freitas, J.J. Little, and D.G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *ECCV*, 2004, pp. 28–39.