www.ietdl.org

Published in IET Computer Vision
Received on 17th August 2013
Revised on 24th March 2014
Accepted on 28th March 2014
doi: 10.1049/iet-cvi.2013.0217

ISSN 1751-9632

Real-time text tracking in natural scenes

Carlos Merino-Gracia[1,2], *Majid Mirmehdi*[2]
[1]*Neurochemistry and Neuroimaging Laboratory, University of La Laguna, La Laguna, Spain*
[2]*Visual Information Laboratory, University of Bristol, Bristol, UK*
E-mail: cmerino@ull.es

Abstract: The authors present a system that automatically detects, recognises and tracks text in natural scenes in real-time. The focus of the author's method is on large text found in outdoor environments, such as shop signs, street names, billboards and so on. Built on top of their previously developed techniques for scene text detection and orientation estimation, the main contribution of this work is to present a complete end-to-end scene text reading system based on text tracking. They propose to use a set of unscented Kalman filters to maintain each text region's identity and to continuously track the homography transformation of the text into a fronto-parallel view, thereby being resilient to erratic camera motion and wide baseline changes in orientation. The system is designed for continuous, unsupervised operation in a handheld or wearable system over long periods of time. It is completely automatic and features quick failure recovery and interactive text reading. It is also highly parallelised to maximise usage of available processing power and achieve real-time operation. They demonstrate the performance of the system on sequences recorded in outdoor scenarios.

1 Introduction

Accessing textual information in the environment is crucial in our daily lives and there is a clear need for technology that can automatically extract and process such text for the benefit of those who might have difficulty accessing it, for example, for assisting the blind, for translating for tourists and for devices that need to know, such as robots. Hence, in recent years, there has been a significant increase in scene text detection and recognition works, see for example [1–4]. The main focus of such works has been on text segmentation in single images, with increasingly better results as measured against the most widely used dataset [5]. However, current state-of-the-art scene text recognition methods, such as those above, and including those that operate on video images and in real-time, for example, [3], lack scene awareness. They treat their input as a succession of unrelated images, attempting to segment and recognise the text in them without taking advantage of the fact that the same text is invariably repeated across many consecutive frames in a video sequence.

Natural scene images pose significant challenges to text understanding, such as blurred or out of focus frames, uneven lighting, complex backgrounds or perspective and lens distortion. On the other hand, a continuous stream of frames provides a temporal redundancy that can help address some of these drawbacks. For example, a blurred image can be difficult or impossible to process on its own, but as part of a sequence, a blurred frame can be ignored as there are chances that other frames in the sequence are clear. In addition, while the estimation of the three-dimensional (3D) orientation of scene text from single images is certainly possible [6, 7], the apparent changes objects undergo in the scene as the camera moves can help reduce the uncertainty in orientation estimation.

The main contribution of the work here is a text reading prototype based on text tracking. Text tracking leverages the main difference between flat-bed scanner and camera-based document analysis (i.e. spatial resolution against temporal redundancy) and allows us (i) to maintain region identity across the sequence and (ii) to smooth the estimation of the region's parameters (position and 3D orientation) to reduce jitter. Both of these outcomes play a major role in facilitating scene awareness, in reduction of false positive segmentations and increase in recognition accuracy, and in better interactive communication of text in the environment to the user, for example, by managing the frequency of communicating text seen in the scene as an audio signal to a blind user. Text regions are segmented using an adaptive threshold-based technique [8], and their 3D orientation estimated by means of an efficient geometrical algorithm [7]. Then, each text region is independently tracked by an unscented Kalman filter (UKF). Our prototype operates in real-time and it is autonomous, that is, new trackers are created when new text entities are found and their identity is kept for as long they are in view; they are removed when the entities are no longer detected, given some resilience to brief occlusions. Trackers are automatically selected for optical character recognition (OCR) – a process carried out in parallel to tracking by an off-the-shelf OCR engine. This demonstrates the role tracking plays, that is, when a text recognition result is available, the system is able to relate it to the original text entity even if the camera has moved. Likewise, available recognition results are automatically selected for synthesising into audio (using text-to-speech) and are played back to the user.

This paper is organised as follows. Section 2 outlines related works. The proposed system is described in detail in Section 3 and in particular, Section 4 explains the text tracking mechanism. We discuss the quantitative and qualitative performance results of our system in Section 5. Finally, Section 6 concludes the paper.

2 Related work

There are very few works on text tracking, let alone natural scene text tracking. Initial works in this area focused on tracking graphically overlaid text in videos (e.g. in news or sports reports), such as [9–11], and here we will not deal with such works any further. In the area of scene text tracking, we are only aware of these works [8, 12–18]. The earliest work specifically dealing with scene text tracking is that of Myers and Burns [12], where an offline system was proposed to extract scene text at arbitrary orientations. They first tracked a series of feature points across the whole sequence, and then, with the assumption that all the points belonging to a certain text area lie on the same plane, they estimated the planar transformation of the points in multiple frames simultaneously to extract a fronto-parallel representation. As it stands, this technique is unsuitable for online real-time operation.

In [8], we presented a near real-time (15 fps) scene text tracking system-based on particle filtering (PF). The text was first segmented using an adaptive threshold-based technique. Segmented components were then grouped together using a saliency filter to form text entities (i.e. words or groups of words forming small sentences). Each text entity was assigned a PF tracker which maintained a set of features and a simple state (a 2D translation and an in-plane rotation). The particles' weights were computed as the number of matched features within a search area, where the identity of individual features was established using scale invariant feature transform (SIFT) [19] descriptors. A key aspect of this method was the use of just a few high quality features for tracking – in this case, segmented characters. This required a full text segmentation stage per frame (and thus demanded a very fast text segmentation algorithm), but the advantage was that the trackers were more resilient to big changes in orientation, occlusions and illumination changes. This early work, although very useful as a proof of concept, was found to have certain drawbacks where some performance improvement was necessary. For instance, over 80% of the frame processing rate was associated with feature matching, that is, SIFT, which is a computationally expensive operator. In addition, the number of SIFT descriptors produced by each feature (i.e. characters) was rather low, and limited the ability to discriminate between measurements. SIFT is affine invariant but not perspective invariant, and no estimation of the 3D spatial orientation of the text in the scene was performed so the whole system was sensitive to wide baseline changes. This is also related to the simple state model used, namely just a 2D translation and in-plane rotation, that traded accuracy and robustness in favour of low computational complexity and hence better frame rate.

Particle filtering was also used by Tanaka and Goto [15, 16] and by Minetto *et al.* [18] for text tracking. The former works described a wearable system for the blind where text was detected using discrete cosine transform (DCT)-based features (on prior works by Goto and his co-workers [13, 14], a simple tracking system was described based on block matching between frames). Tracking was performed by generating particles on candidate text regions in new frames, and they were weighted according to a similarity function between the regions based on cumulative histograms. No perspective correction was performed, and only region identity and limited 2D motion was maintained by this method.

In Minetto *et al.* [18], candidate text regions were initially segmented using a morphological operator applied at different scales, then classified by means of a support vector machine, and grouped together based on their relative distances and sizes. For each text region, particles were propagated in subsequent frames using a first-order motion model. Particles' weights were proportional to a similarity coefficient between the histogram of oriented gradients descriptors of the respective image regions.

A different approach was used by Na and Wen [17] by tracking text directly using SIFT. A global motion between frames, modelled as a similarity transformation [20] was computed by minimising the least squares distance between the SIFT feature matches. In their work, the authors did not specify how text regions were segmented. Furthermore, the computational complexity of extracting SIFT descriptors from every frame precludes the real-time use of this technique, and the simple motion model limits the usefulness of the method when applied to outdoor hand-held camera scenarios. SIFT was also used in [21] to track and align text regions appearing in a sequence of frames. An integration of the aligned text probability maps was then used to improve OCR accuracy. Their algorithm looks for text in adjacent frames either forwards or backwards, necessarily making it an offline process and unsuitable for real-time operation. In addition, the method requires a manual selection of the initial text bounding box.

In the work by Mosleh *et al.* [22], text tracking is used to separate overlaid text from scene text in order to automatically remove the former (but not the latter) from video sequences. A CAMSHIFT algorithm is used to infer the text motion, assuming a few constraints on text movement (i.e. movement along the vertical or horizontal axis). Optical flow is then used to isolate the movement of these regions from that of the background. Other recent text detection works of note are [23–25], in which the focus is not only on segmenting the text but also recognising it. However, these approaches still lack the context and temporal redundancy awareness that tracking provides. To the authors' knowledge, no other work exists on scene text tracking which shows the scope, aim and performance of the methodology proposed here.

3 Preparing to track: text segmentation and perspective recovery

The proposed end-to-end real-time text reading system, including its tracking and other processing stages, is illustrated in Fig. 1. Text tracking is performed on high level features (i.e. perspective corrected characters). This provides several advantages over low-level tracking of feature points, such as increased resiliency against orientation changes and occlusions. However, the tracking needs to be preceded by quick and efficient techniques for text segmentation and perspective estimation, as these operations have to be performed on every frame. Here, we build on our previous work for real-time text detection, grouping and perspective rectification [7, 8, 26] and for the

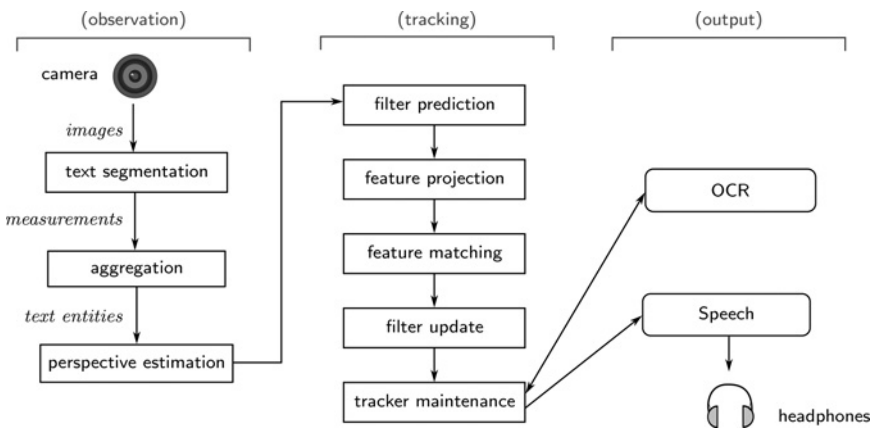


Fig. 1 *Schematic of the proposed end-to-end real-time text reading system*

sake of completeness, these fundamental stages are first briefly summarised in Sections 3.1–3.3. Then, the main focus of this work, that is, our text tracking mechanism, is explained in detail in Section 4.

3.1 Text detection

Each input image is segmented to obtain a set of candidate text regions, containing possibly none, or one, or more characters. Later, these regions will be used as measurements for the tracking filter, as well as serving as the building blocks for text line aggregation and tracker creation.

A brief outline of our segmentation algorithm, first presented in [8], is as follows. Adaptive thresholding is applied to binarise the input image and retrieve a set of connected component (CC) regions. A tree is then constructed to represent the topological relationship between these CCs. A key step of this algorithm is the hierarchical filtering of the tree nodes, based on the assumption that in natural scene images with text, structural elements (such as sign borders, posters frames etc.) can be discarded purely based on their hierarchical relationships with other text regions. In addition, the tree filtering approach we proposed in [8] allows for the segmentation of dark and light text in one pass only. The CCs are then pruned by means of a cascade of text filters that operate on characteristics, such as size and contrast against the background, and an eigenvector-based texture measure adapted from [27]. Fig. 2*a* shows an example image and Fig. 2*b* illustrates the corresponding CCs (i.e. candidate text regions) detected at this stage.

3.2 Text aggregation

For the purpose of text recognition by (off-the-shelf) OCR, we need to group the CC regions together in order to form text entities (as we previously performed in [8]) for tracking and to be able to extract common clues for perspective estimation. The grouping is performed by first determining which CC regions are closely associated by evaluating a visual saliency measure between each pair of regions, and then by searching for dominant orientations to separate

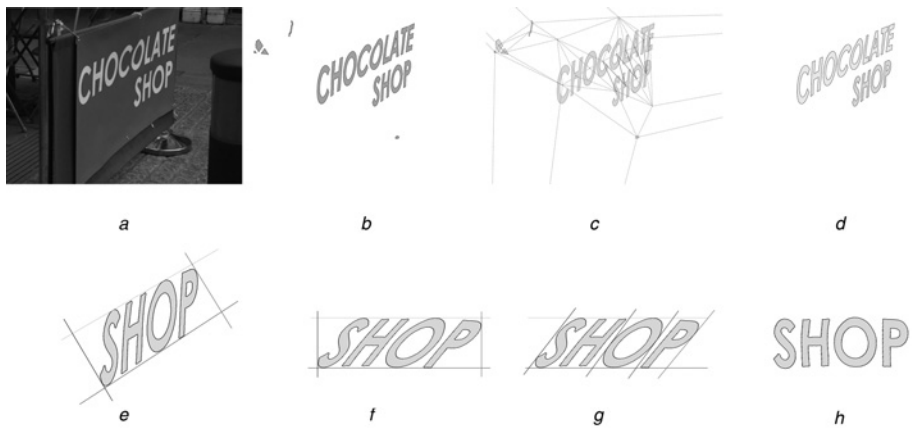


Fig. 2 *Text segmentation and perspective estimation for*

a Example of original image
First, for the segmentation and aggregation stages
b Segmented components
c Association graph (dashed edges were removed during saliency filtering and thin edges were removed during histogram filtering; the thick edges represent the segmented text lines)
d Grouped text lines
Then, for the perspective estimation step
e Top and bottom lines estimation
f Parallel image
g Shear estimation
Finally
h Full perspective rectification

independent lines of text. First, a Delaunay triangulation [28] to join the centre points of every CC is performed, with the centre points being the centre of mass of each region. The Delaunay triangulation enables us to efficiently construct a neighbour relationship graph between all the components. Fig. 2*c* shows the result of the Delaunay triangulation. For every edge of the resulting graph, which represents a pair of adjacent CCs, a saliency measure is computed [29]. Those edges with a low saliency value are removed from the graph. In Fig. 2*c*, edges removed during the saliency filtering are represented in grey.

After the saliency filtering, every remaining connected subgraph is a candidate text group, each of them possibly containing one or more lines of text. A histogram of angle distribution of the graph edges is constructed to find each group's dominant orientation, which is chosen to be the histogram bin with highest element count. The edges of the graph that do not belong to this dominant orientation are removed, after which the original subgraph may be split into smaller subgraphs separating the individual text lines. In Fig. 2*c*, filtered edges at this stage are represented in red, and the remaining connected subgraphs are represented in green. Fig. 2*d* shows the result of the text segmentation and grouping, in which each segmented text line is drawn in a different colour. Every remaining connected subgraph contains only one text line and is called a text entity. Text entities are the basis of new tracker creation (as explained later in Section 4.1).

3.3 Perspective estimation

At this stage of text recovery, we estimate a 3 × 3 homography matrix transformation of a text entity into a fronto-parallel representation [7]. Assuming a pinhole camera model, the homography is the transformation that allows the modelling of all possible orientation changes the text can undergo in an image without the need to have calibrated cameras or an explicit 3D representation. The ability to quickly estimate the text orientation makes high-level perspective-aware tracking possible, as well as the extraction of perspective covariant feature descriptors.

The orientation detection is performed in two steps: parallel rectification and shear estimation. At first we compute the farthest character points on each side of the main axis of the text line, which are used to fit a top and a bottom line (Fig. 2*e*). As both lines are assumed to be parallel in the real world, they are used to rotate and partially rectify the text entity leaving only a remaining shearing effect (Fig. 2*f*).

Then, we estimate a shear value for every character by minimising the distance between two parallel lines that rotate around antipodal vertices of the character's contour (this is a variation of the Rotating Calipers paradigm [30]). With a shear value for each character, a linear shear variation across the whole line can be obtained and used to compute the full rectifying homography of the text entity, as illustrated in Fig. 2*g*. Fig. 2*h* shows the rectified image.

4 Proposed text tracking

We now present our proposed text tracking approach. Once a text entity is identified, a tracker is created to follow the text region from frame to frame while it is in camera view. The detailed process of tracker creation and removal is explained later in Section 4.1. For now, for ease of exposition, we assume that a set of trackers already exists and properly initialised to follow a corresponding set of text entities in the scene.

A tracker is characterised by a set of tracked features $\boldsymbol{z}_i$ and a dynamic state $\boldsymbol{x}_k$, which is updated by a predictive filter. The features $\boldsymbol{z}_i$ correspond to the individual characters in the text line and are used as the anchor points to be matched against image measurements during the observation stage of the filter. They are stored in a fronto-parallel representation, in a coordinate frame referred to as 'tracker coordinates' (see Fig. 3). Each feature is defined as

$$\boldsymbol{z}_i = \begin{bmatrix} x & y & w & h \end{bmatrix}^{\mathrm{T}}, \quad i = 1, \ldots, M \tag{1}$$

where (x, y) is a feature's centre point, (w, h) are the dimensions of the feature's bounding box and M is the number of features. In addition, each feature keeps a perspective corrected image patch used during feature matching (as seen in Fig. 3).

In [8], we used PFs [31] since they model a non-linear system, such as our text tracking problem, well. However, in this work we choose the unscented Kalman filter (UKF) [32] for tracking because it provides the uncertainty of the system's state estimation via a Gaussian probability distribution, and it is also more efficient, that is, to achieve the same accuracy as the PF, it needs to use substantially fewer sampling points. The UKF is derivative free and employs a deterministic sampling approach (the unscented transform, UT) to propagate the density function across non-linear state changes. The UT captures the non-linearities better than alternatives such as the extended Kalman filter and is easier to implement.

We represent the filter state by a homography transformation $\boldsymbol{H}$ mapping the fronto-parallel view of the tracked features in tracker coordinates to image coordinates (Fig. 3). The homography can be characterised by a vector $\boldsymbol{h}$

$$\boldsymbol{h} = \begin{bmatrix} t_x & t_y & \theta & s_x & s_y & \sigma & l_x & l_y \end{bmatrix}^{\mathrm{T}} \tag{2}$$

where (t_x, t_y) defines a translation, θ is an in-plane rotation angle, (s_x, s_y) is an anisotropic scale, σ is a shearing and (l_x, l_y) is a foreshortening around both axes. Given the homography, there is a closed form unique solution for all

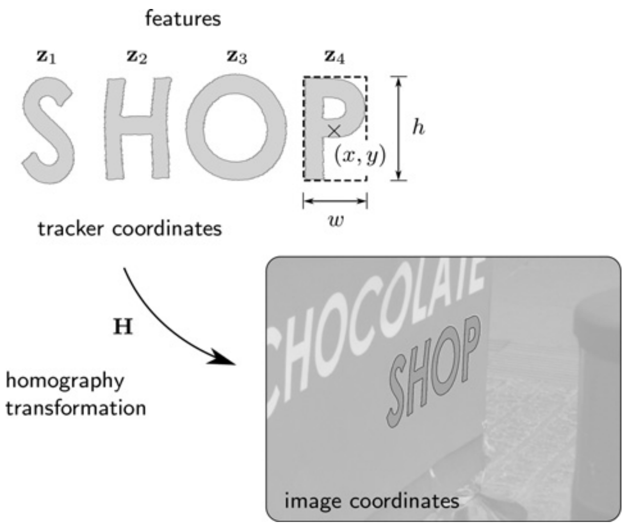


Fig. 3 *Tracker representation, which keeps a set of features* $\boldsymbol{z}_i$ *and the state of the UKF that produces the homography* $\boldsymbol{H}$

the parameters [20] (refer to the Appendix for the formulation of this solution) which we name, along with its inverse, the homography (de)composition function $\mathcal{H}$

$$\boldsymbol{H} = \mathcal{H}(\boldsymbol{h}), \quad \boldsymbol{h} = \mathcal{H}^{-1}(\boldsymbol{H}) \tag{3}$$

Although both representations are mathematically equivalent, with this decomposition the filter deals directly with the underlying parameters that define the transformation, enabling the direct estimation of the uncertainty in each parameter via the covariance matrices. The dynamic model also benefits from this representation as we can define velocity vectors that affect only the translation or rotation parameters of the transformation. For the rest of this section, when we refer to the homography $\boldsymbol{H}$, an implicit conversion will be assumed from the parameters vector $\boldsymbol{h}$ to the homography using $\mathcal{H}(\boldsymbol{h})$. The only moment in which the inverse operation $\mathcal{H}^{-1}(\boldsymbol{H})$ is needed is for tracker creation, as explained in Section 4.1.

For the prediction stage of the filter, we use a constant velocity model, where we only consider in-plane translational and angular velocities. We define the velocity vector $\boldsymbol{v}$ as

$$\boldsymbol{v} = \begin{bmatrix} v_x & v_y & \omega \end{bmatrix}^{\mathrm{T}} \tag{4}$$

and the state vector of the filter at frame k as a stacking of the homography parameters and velocity vectors

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{v} \end{bmatrix} \tag{5}$$

The new state prediction is then

$$\hat{\boldsymbol{x}}_{k+1} = \begin{bmatrix} \boldsymbol{h} + \mathring{\boldsymbol{v}}\Delta t \\ \boldsymbol{v} \end{bmatrix} \tag{6}$$

where $\mathring{\boldsymbol{v}} = \begin{bmatrix} \boldsymbol{v}^{\mathrm{T}} & 0^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is the velocity vector padded with zeros to the length of $\boldsymbol{h}$ and Δt is the elapsed time since the last frame.

The measurement function maps the tracked features to observable characteristics in the image (called measurements) using the filter state. However, as each tracker represents a line of text, the centre points of all the characters are roughly aligned. If the centre points were the only points of our measurement function, there would be a great deal of uncertainty for rotations around the horizontal axis (i.e. elevation – see Fig. 4 as an example). Since we have a good estimate of the text orientation, and we know that all the points of a text line lie on a plane, our observation model includes five points per tracked feature $\boldsymbol{z}_i$: the centre point $\hat{\boldsymbol{c}}_0$ and the four corner points $\hat{\boldsymbol{c}}_j$, $j = 1, \ldots, 4$ of the feature's bounding box (as shown in Fig. 5)

$$\hat{\boldsymbol{z}}_i = \begin{bmatrix} \hat{\boldsymbol{c}}_0^{\mathrm{T}} & \hat{\boldsymbol{c}}_1^{\mathrm{T}} & \hat{\boldsymbol{c}}_2^{\mathrm{T}} & \hat{\boldsymbol{c}}_3^{\mathrm{T}} & \hat{\boldsymbol{c}}_4^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \quad i = 1, \ldots, M \tag{7}$$

These are converted from tracker coordinates using the predicted state homography. For example, $\hat{\boldsymbol{c}}_0$ is computed as the transformation of (x, y) to image coordinates, $\hat{\boldsymbol{c}}_1$ as the transformation of $(x - w/2, y - h/2)$, and likewise for $\boldsymbol{c}_2$ to $\boldsymbol{c}_4$.

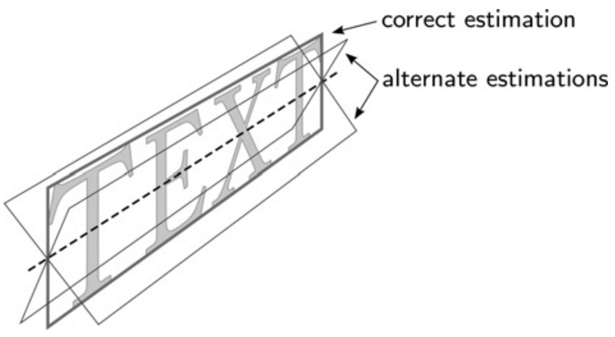


Fig. 4 *Homography estimation ambiguity if only the centre points of each character are considered*

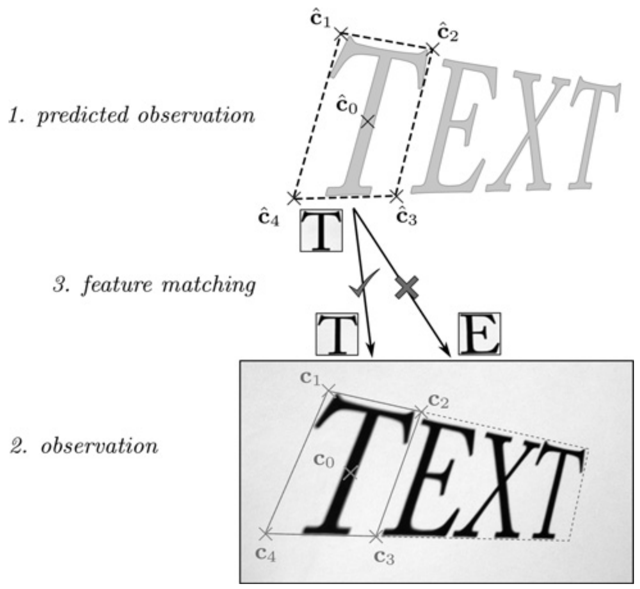


Fig. 5 *Observation model*

First a new location for the tracker features is predicted
Here the five predicted observation points $\hat{\boldsymbol{c}}_i$ for the first feature are represented in the upper part
Then, from the segmentation and perspective orientation stage, the actual measurement points $\boldsymbol{c}_i$ are obtained
These are represented in the lower part
Finally, the candidate measurements are matched using perspective corrected image patches which are also shown

The predicted observation is then a combination of all of the individual feature mappings

$$\hat{\boldsymbol{y}}_k = \begin{bmatrix} \hat{\boldsymbol{z}}_0^{\mathrm{T}} & \cdots & \hat{\boldsymbol{z}}_M^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{8}$$

Finally, as the last part of the observation model, the tracked features need to be matched against the segmented text regions or candidate measurements. To discriminate between the candidates, each feature keeps an image patch, normalised to 50×50 pixels. It is a perspective corrected image patch, extracted using the four corner points of the feature from the frame in which the tracker was created. Matching is performed using normalised cross correlation (NCC) and only on measurements within a certain search radius around the predicted feature position. The search radius is obtained from the filter's state covariance matrix, as it represents the uncertainty in the new state's prediction. After matching, each feature has one measurement candidate. As with the observation function, each measurement $\boldsymbol{m}_i$ is defined by the centre point of the region and its four corner points

$$\boldsymbol{m}_i = \begin{bmatrix} \boldsymbol{c}_0^{\mathrm{T}} & \boldsymbol{c}_1^{\mathrm{T}} & \boldsymbol{c}_2^{\mathrm{T}} & \boldsymbol{c}_3^{\mathrm{T}} & \boldsymbol{c}_4^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \quad i = 1, \ldots, M \tag{9}$$

where $\boldsymbol{c}_j, j = 0, \ldots, 4$, are the mapped centre and corner points, and are obtained from the orientation estimation stage. Hence, the observation used by the UKF is

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{m}_0^{\mathrm{T}} & \cdots & \boldsymbol{m}_M^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{10}$$

In Fig. 5, the observation model is illustrated: the predicted observation, the actual observation and the feature matching.

4.1 Tracker maintenance

Tracker maintenance refers to the set of mechanisms in which new trackers are created, new features are added to existing trackers, and bad trackers are removed. This allows the automatic continuous operation of the system.

At first we need to correlate the text entities produced in the text association stage to the current set of trackers. The text association stage groups measurements as belonging to the same text entity and the tracking stage may associate features to some of the measurements after matching.

By correlating tracked features to measurements and then to text entities, several possibilities arise: (i) a tracker has matched all the measurements belonging to a text entity – this is the perfect tracking case and no further action is needed (Fig. 6*a*); (ii) no tracker has matched any of the measurements of a given text entity – the entity is then a candidate for tracker creation (Fig. 6*b*); and (iii) a tracker has matched some of the measurements inside an entity – the remaining (unmatched) measurements are candidates for feature addition to that tracker (Fig. 6*c*). In addition, when a tracker matches most or all of its features it is considered a good track. Likewise, if a tracker did not match any of its features (or only matched a low fraction of them), it is considered a bad track or a mistrack. After a certain number of frames being a bad track, a tracker is removed. These operations are further explained in the following.

4.1.1 Tracker creation: When a text entity does not have any tracker matching any of its features, it is considered an untracked entity, thus requiring a tracker to be created for it. Tracker creation proceeds as follows: the perspective estimation stage returns a homography transformation $\boldsymbol{H}'$ of the measurements in the group to a fronto-parallel representation. All the measurements are converted into features in the new tracker by applying this homography transformation and then obtaining the centre point and dimensions of each text region in the fronto-parallel view. Then, the filter state is initialised as

$$\boldsymbol{x}_0 = \begin{bmatrix} \boldsymbol{h}_0^{\mathrm{T}} & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{11}$$

with $\boldsymbol{h}_0 = \mathcal{H}^{-1}(\boldsymbol{H}_0)$ and $\boldsymbol{H}_0 = (\boldsymbol{H}')^{-1}$ being the initial homography estimation of the transformation between the fronto-parallel representation to image coordinates.

On creation, a tracker is marked as unstable. This means that it will not be considered for feature addition, for recognition or transformation to speech, and it will not be shown as a segmented region. It is only considered stable after it is tracked continuously for a number of frames – in our case this was arbitrarily set to ten. As a text region is consistently segmented in a sequence of frames, as opposed to noisy regions, this process cleans most of the text segmentation false positives.

4.1.2 Feature addition: When a stable tracker matches some of the measurements inside a text entity, the remaining unmatched measurements are assumed to belong to the same entity. Hence, they are added to the tracker as new features. The corner points of the created feature are mapped to the tracker coordinates using the tracker's state homography, and the observation vector length is increased accordingly.

4.1.3 Tracker removal: When a tracker has been regarded a bad track for a few frames because none or too few of its features are matched, the tracker is removed. There is no long term registry of old trackers. If a tracker is removed (e.g. because it is no longer in view, or due to a long occlusion), and afterwards the text entity it was tracking is detected again, it will be added again as a new tracker. We find this to be an adequate compromise for efficient and long periods of continuous operation.

4.1.4 Occlusions: These mechanisms allow our system to deal with brief occlusions of the tracked text regions. A full occlusion will produce bad tracks for the affected trackers. If the occlusion is shorter than the number of frames needed to delete a tracker, when the text region is in view again it will be recovered. The system will also be able to recover the track even with big translations or wide baseline changes of orientation thanks to the use of high level features. Partial occlusions of text regions will be also dealt with in the same fashion, and even on tracker creation, because of the feature addition mechanism.

4.2 OCR and speech synthesis

When a tracker is considered stable, it is then a candidate for recognition. The image quadrilateral enclosing the tracked text entity is rectified to a fronto-parallel view using the state homography transformation and then sent to OCR, for

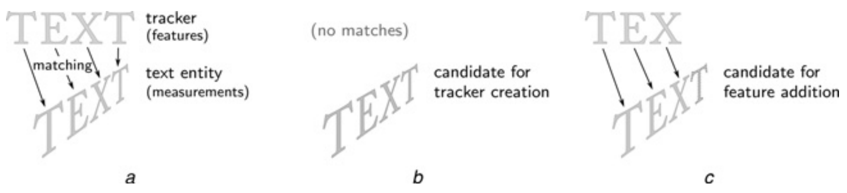

TEXT
tracker (features)
matching
TEXT
text entity (measurements)
a
(no matches)
TEXT
candidate for tracker creation
b
TEX
TEXT
candidate for feature addition
c

Fig. 6 *Tracker maintenance*

a Perfect tracking
b Tracker creation
c Feature addition

which we use the Tesseract OCR engine [http://code.google.com/p/tesseract-ocr/]. Recognition is performed in a parallel processing task, so the tracking is maintained in real-time while the recognition runs alongside (for implementation details see Section 4.3). The decision on which tracker to recognise is weighted by several factors: whether or not there is already any recognition available for this tracker, the OCR confidence value of previous recognitions, and the elapsed time since the last recognition attempt. A tracker might be sent to OCR for recognition several times, but if the returned confidence value of a new recognition is lower than a previous one, the recognition result with higher confidence is kept.

Speech synthesis is the main user interface of the system, and the main intended communication with the user. Those text regions that have a high enough OCR confidence value and have a stable text tracker are considered for being synthesised into speech. The text is sent to an speech synthesis engine (in our case we use Microsoft's speech API) so the recognised text is played back to the user. The candidate texts are queued and prioritised according to the distance to the centre of the image. Regions that stay in view of the camera for long enough might be reproduced several times, but as the region identity is maintained throughout the sequence, this delay can be adjusted for the convenience of the user, that is, by tracking the text we can avoid the system continuously repeating the same text over and over.

4.3 Implementation

One of the design objectives of our prototype is real-time operation. The system is carefully parallelised to make the best possible use of available processors. We use Intel's Threading Building Blocks (TBB) [http://threadingbuildingblocks.org/], which implements a task-based parallelisation paradigm. It features a high level C++ API for defining parallel constructions, supports nested parallelism and provides automatic scalability. The algorithm steps in Fig. 1 are implemented as separate stages of a processing pipeline. On multiprocessor machines, TBB is able to schedule different stages on different processors so several frames might be simultaneously processed at any given moment. The system maintains a global state and a transient state. The transient state is carried forward across the stages of the pipeline. At the end of a frame processing, the transient changes are atomically combined in the global system state. This is easily implemented as the library guarantees strict sequential ordering of the pipeline stages of consecutive frames (i.e. the tracking stage on frame n is guaranteed to run before the tracking stage of frame $n+1$). OCR is spawned as a task outside the main processing pipeline and thus it is scheduled concurrently with it. This allows to maintain real-time performance for the text tracking while being able to perform longer running processes at the same time and without under- or over-subscribing the available parallelism. We find this to be a superior design in terms of portability and scalability when compared with other approaches (such as e.g. PTAM [33]) in which explicit threads are defined with synchronisation mechanisms between them.

5 Results and discussion

To demonstrate and validate the proposed system, at first we present a quantitative analysis of the text tracking mechanism based on standardised metrics and annotated ground-truth data that help establish a performance baseline for comparative studies. Then, a qualitative evaluation of the prototype's operation in everyday scenarios is outlined to provide an insight into the future improvements and requirements of a text reading system. For our quantitative experiments, three challenging video sequences are used: HOSPITAL, MERCHANT and QUEEN. These contain scene text in a city environment and suffer from hand-held erratic camera motion, as well as blur and a great degree of perspective distortion. The sequences were annotated to obtain a (i) ground-truth labelling of text, (ii) 3D bounding quadrilaterals and (iii) region identity between frames [The video sequences and the ground-truth labelling are available at http://nf.ull.es/q/texttrack]. To achieve these, tracking was applied in the video sequences using a commercial match-moving software, with each video requiring extensive manual adjustment of the tracked features. After that, 3D editing software was used to locate rectangles in the 3D space around the projected positions of the text in the scene. When the rectangles are projected back as quadrilaterals into the image plane, they perfectly fit the text as seen in the image. The total number of annotated frames is 930. For the qualitative analysis, a variety of example videos, showcasing different scenarios, were experimented with as shown later.

5.1 Quantitative analysis of the tracking mechanism

Two distinct tests were performed to evaluate the two desired characteristics of a text tracking system: (i) increase in segmentation accuracy and (ii) the ability to maintain region identity. The first test is a frame-by-frame comparison of the text detection accuracy between the text segmentation stage alone against the segmented text regions while tracking by our method. The second test evaluates the tracking performance by measuring the detection accuracy along with the region identity across the sequence as a whole.

5.1.1 Frame-by-frame evaluation: For our text segmentation evaluation, we use the precision and recall measures as introduced in the ICDAR 2003 Robust Reading Competition [5], slightly adapted to operate on arbitrary quadrilaterals instead of rectangles. The degree of match between two quadrilaterals $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ is defined as

$$\text{match}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \frac{\text{area}(\boldsymbol{q}_1 \cap \boldsymbol{q}_2)}{\text{area}(\boldsymbol{q}_1 \cup \boldsymbol{q}_2)} \tag{12}$$

When comparing a quadrilateral $\boldsymbol{q}$ against a set of quadrilaterals Q, the best match is computed as

$$\text{bestmatch}(\boldsymbol{q}, Q) = \max_{\boldsymbol{q}' \in Q} \text{match}(\boldsymbol{q}, \boldsymbol{q}') \tag{13}$$

Then, the precision p, recall r and f measures for a certain frame are defined as

$$p = \frac{\sum_{\boldsymbol{q} \in E} \text{bestmatch}(\boldsymbol{q}, G)}{|E|} \tag{14}$$

$$r = \frac{\sum_{\boldsymbol{q} \in G} \text{bestmatch}(\boldsymbol{q}, E)}{|G|} \tag{15}$$

$$f = \frac{2}{1/p + 1/r} \tag{16}$$

where G is the set of quadrilaterals in the ground-truth and E is the set of quadrilaterals being evaluated.

These measures were computed for each one of the frames in the sequences, comparing the quadrilaterals produced by the text segmentation and orientation detection stages with the quadrilaterals produced by the tracking stage. In our evaluations, we are only considering text lines with four or more characters from the ground-truth, as this is the minimum length at which the perspective estimator works reliably [7]. The results are shown in Figs. 7*b*, 8*b* and 9*b*, where, for every frame in the sequences, the f measure for the segmentation and tracking outputs are plotted.

In the HOSPITAL sequence (Fig. 7), the camera approaches a road sign with strong perspective distortion being introduced gradually, and featuring a very textured tree background, which makes text segmentation challenging, as can be seen in the performance of the text segmenter. The accuracy of the tracked regions is very good throughout the sequence even when the perspective distortion is significant. The continuous change in perspective occasionally causes the trackers to lose track as the filters converge into the new orientation. This can be seen in the dips around frames 5, 20 and especially between frames 100 and 115 and at the end of the sequence. Nevertheless, the region identity is never lost, and the trackers recover the lock on the text shortly afterwards.

In the MERCHANT sequence (Fig. 8), a street sign is panned laterally, with a wide baseline change of orientation and a textured regular brick pattern in the background. Text region tracking is accurate across most of the sequence, as shown in the results. Extreme camera shake is the cause for some of the individual trackers to momentarily lose track, from frame 125 to the end. The system does not produce a box for a text region that is not tracked with enough confidence, thus the

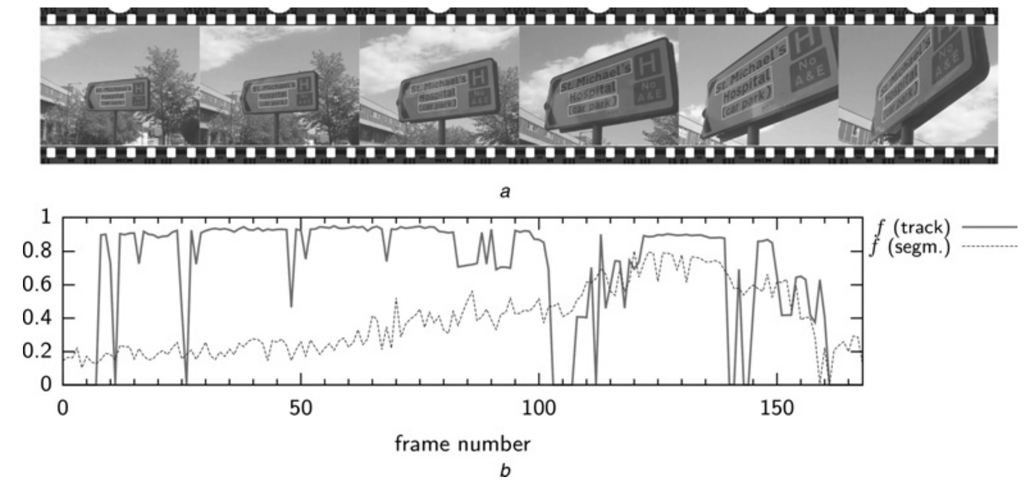


Fig. 7 *Video sequence HOSPITAL*

a Video sequence with tracked text bounding quadrilaterals
b Frame-by-frame segmentation accuracy

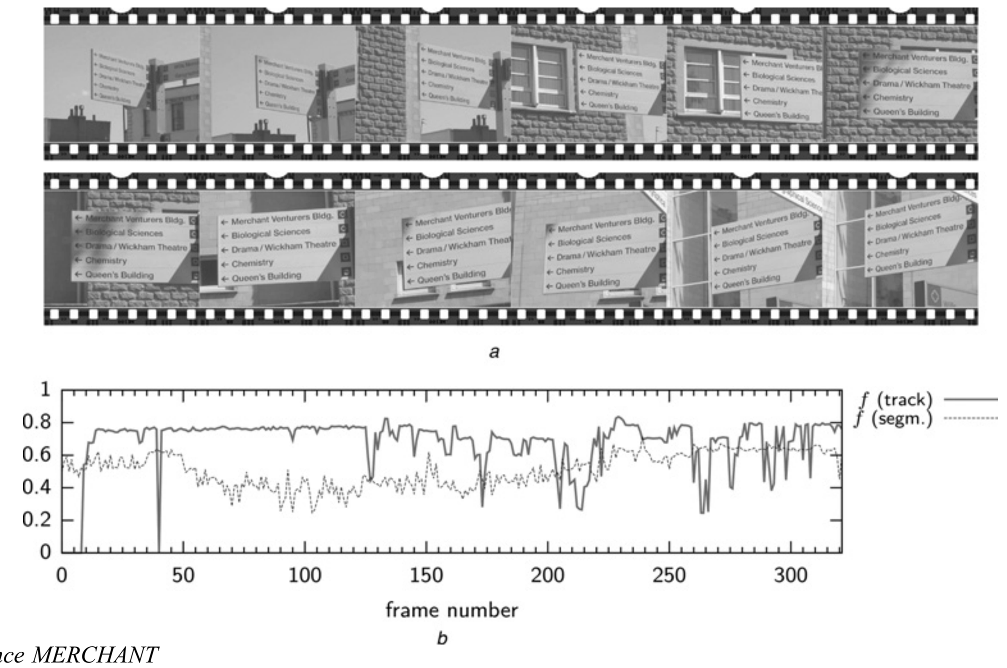
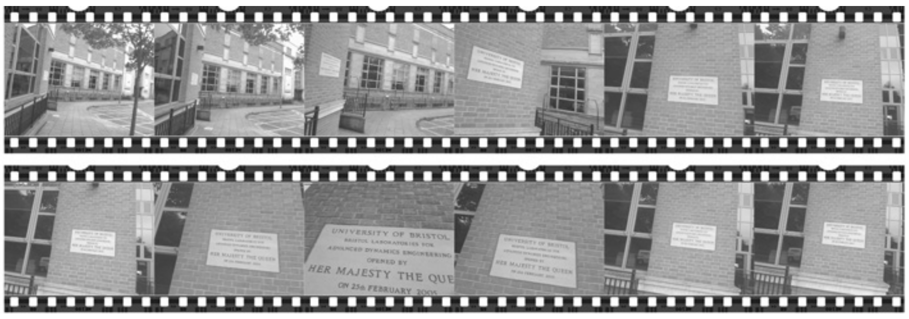
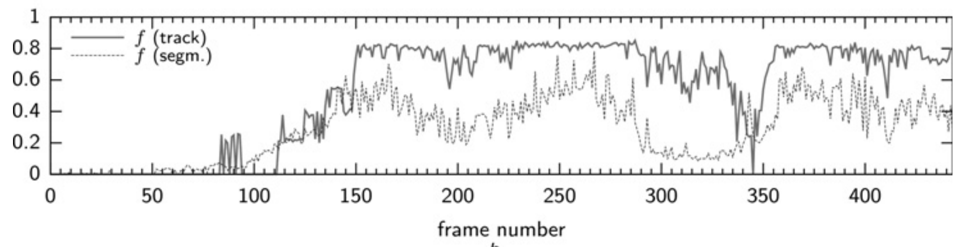


Fig. 8 *Video sequence MERCHANT*

a Video sequence with tracked text bounding quadrilaterals
b Frame-by-frame segmentation accuracy


a



b

Fig. 9 *Video sequence QUEEN*

a Video sequence with tracked text bounding quadrilaterals
b Frame-by-frame segmentation accuracy

alternating and temporary disappearances of some of the boxes across the sequence. Those are promptly recovered without losing the region identity. This is also the reason for the dip in the graph around frame 40, where all the trackers are lost for just one frame.

Finally, the QUEEN sequence (Fig. 9) features a walkabout towards a building sign which is not visible from the start of the sequence, finishing with a close approach into the text. This very challenging sequence also shows regular window and railing patterns. As can be seen in the results, during the first part of the sequence – until frame 120 – the accuracy is 0 as there is no text. When the text is completely visible, the tracking quickly locks on the text lines and this is clearly shown in the graph between frames 120 and 280. Then, during a camera movement towards the text, there are two brief camera blur events between frames 280–300 and 330–350 that cause the segmentation to produce very few regions, and thus making the tracker to momentarily lose track as there are no regions to track, especially at frame 345. The tracker promptly recovers the track after these events. The ability to quickly recover from failures demonstrates the versatility of the proposed method. To illustrate the effect that text tracking has on segmentation accuracy, Fig. 10 shows the output of the text segmentation stage, with a considerable number of false positives, compared to the output of the tracking stage, where the spurious regions have been discarded.

5.1.2 Tracking evaluation: The characteristics of our prototype system preclude making a comparative study against any of the few other published text tracking systems. For example, in the work in [12], the text regions are manually selected prior to tracking or in the work in [34] text tracking was performed on overlaid text. Furthermore, we are tracking 3D text quadrilaterals, as opposed to 2D rectangles, which is a fundamental difference with any other published work (e.g. [16, 18]). With this paper, we are also publishing our scene text tracking dataset and its associated ground-truth data – something that has not been done before – in the hope that it will be useful to other researchers and to enable future comparative studies.

However, we do present comparative results against our own previous work in [8]. To evaluate the performance of the text tracking we adopt the CLEAR object tracking metrics as suggested by Bernardin and Stiefelhagen [35] and Kasturi *et al.* [34]. The metrics are designed to evaluate the detection and tracking performance across a sequence as a whole, and thus penalise false positives and false negatives as well as region identity changes or losses. For every frame k, there is a mapping M_k between the elements in the ground-truth and detected sets

$$M_k = \{(\boldsymbol{g}, \boldsymbol{e}),\ \text{with } \boldsymbol{g} \in G_k \text{ and } \boldsymbol{e} \in E_k\} \tag{17}$$

where G_k and E_k are the sets of ground-truth and detected quadrilaterals at frame k. Mappings are unique for each element on each set. Our criteria for considering a candidate match between two quadrilaterals is that they have some overlap (e.g. area$(\boldsymbol{q} \cap \boldsymbol{q}') > 0$). To select a unique match between the candidates, we first consider the same match as in the previous frame if they still overlap; otherwise the pair with maximum overlap is chosen (refer to [35] for the full

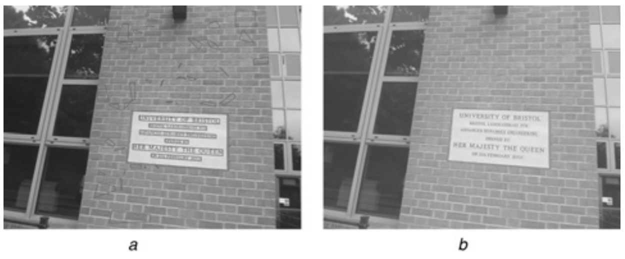

a b

Fig. 10 *Comparison of the output of the*

a Text segmentation
b Tracking stages, for one frame in the QUEEN sequence

Table 1 Whole sequence tracking evaluation

Sequence	Proposed method		PF method [8]	
	MOTP	MOTA	MOTP	MOTA
HOSPITAL	0.89	0.7	0.1	0.32
MERCHANT	0.78	0.82	0.16	0.24
QUEEN	0.8	0.65	0.1	0.13

explanation and rationale of the matching strategy). Every detected quadrilateral that is not mapped is a false positive; likewise, every unmapped ground-truth quadrilateral is a missed detection. If a quadrilateral $\boldsymbol{g} \in G_k$ is matched to different quadrilaterals $\boldsymbol{q}, \boldsymbol{q}' \in E_k$ in consecutive frames, it is considered an identity mismatch.

Two measures are defined, the multiple object tracking precision (MOTP) and the multiple object tracking accuracy (MOTA)

$$\text{MOTP} = \frac{\sum_k \sum_{(\boldsymbol{g},\boldsymbol{e}) \in M_k} \text{match}(\boldsymbol{g}, \boldsymbol{e})}{\sum_k |M_k|} \tag{18}$$

$$\text{MOTA} = 1 - \frac{\sum_k \left(\delta_k + \phi_k + \log_{10}(\rho_k)\right)}{\sum_k |G_k|} \tag{19}$$

where δ_k, ϕ_k and ρ_k are the total number of missed detections, false positives and id mismatches for frame k, respectively.

Table 1 summarises the results of our proposed method in comparison to our previous text tracking technique using PFs [8]. Our PF method had a simple 2D state model and did not

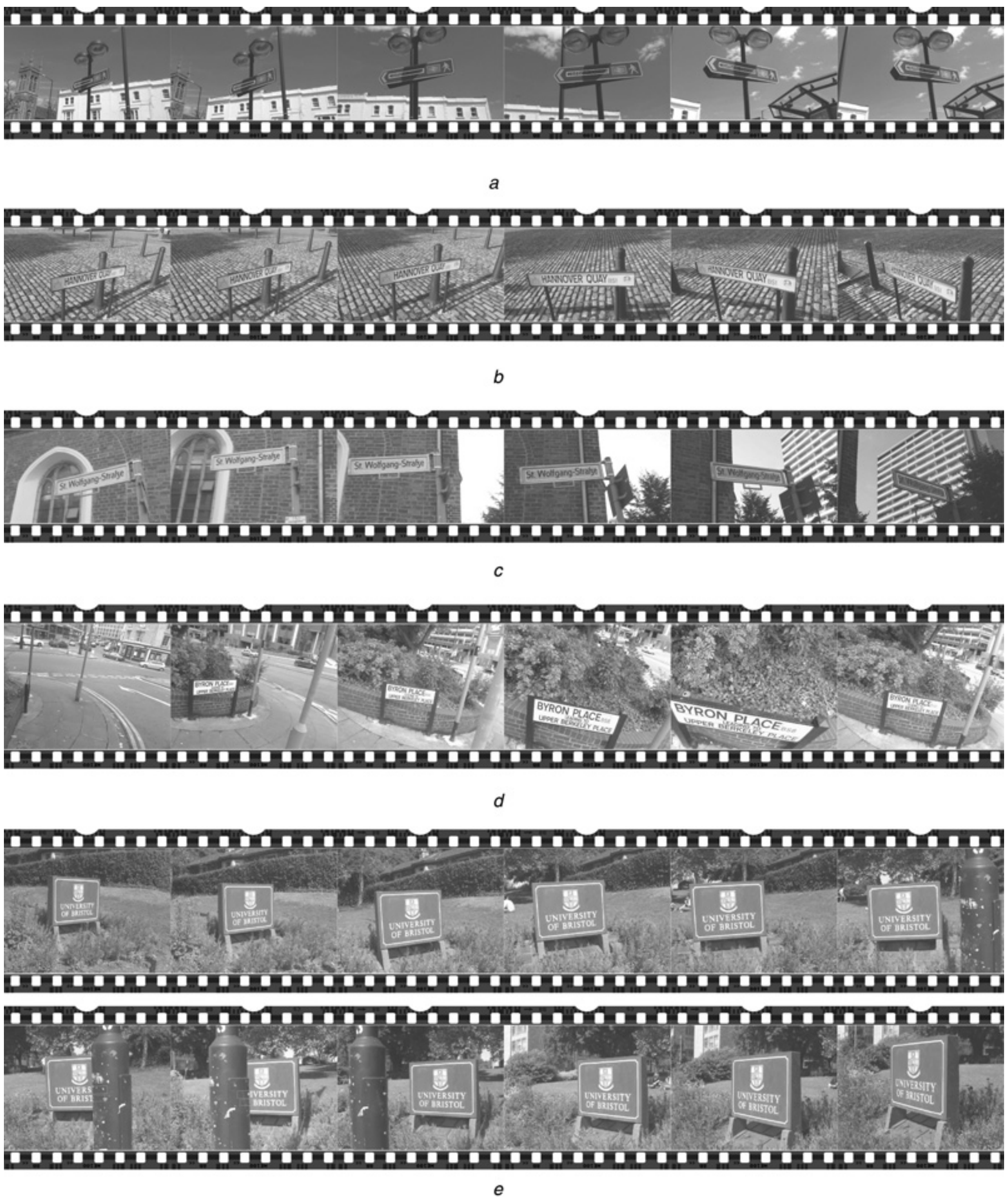


Fig. 11 *Video sequences from the qualitative evaluation experiment*

a Sequence CLIFTON
b Sequence HANNOVER
c Sequence WOLFGANG
d Sequence BYRON PLACE
e Sequence UOB

perform any perspective estimation and this is shown in the MOTP values, where the proposed method consistently achieves very high values (0.89, 0.80 and 0.78 for the HOSPITAL, QUEEN and MERCHANT sequences, respectively) thanks to the enhanced state model. It also produces high MOTA values (0.82, 0.70 and 0.59 for the MERCHANT, HOSPITAL and QUEEN sequences, respectively) because of the low number of false positives and id mismatches that the tracking produces. As previously explained, the challenging sequences used in our experiments feature very erratic camera motion, blur and perspective distortion. Our MOTA evaluation is penalised on those frames where the text is not tracked with enough confidence, in which our system does not produce any box, being counted as a missed detection.

5.2 Qualitative evaluation

We show more examples in Fig. 11 to further illustrate the operation of the proposed method. In the CLIFTON sequence (Fig. 11*a*), the text is never in a fronto-parallel, horizontal orientation with respect to the camera and undergoes a wide baseline change in orientation. This is also true for the WOLFGANG sequence (Fig. 11*c*), which also features a complex background and very blurred frames because of camera vibrations. The HANNOVER sequence (Fig. 11*b*) contains regular, very contrasted tiles in the background which produce a great number of false positives from the text segmentation stage. The BYRON PLACE sequence (Fig. 11*d*) features a change in orientation around elevation. Note, the text is not visible at the start of the sequence, but as soon as it appears, the system is able to pick up the location of the various lines of text and then track them continuously. As with previous sequences (i.e. MERCHANT and QUEEN), camera shake is responsible for the momentary disappearance of tracked regions, after which the tracker recovers without region identity loss. Finally, the UOB sequence (Fig. 11*e*) features a moving partial occlusion across the tracked text. As there are always enough visible features (i.e. characters) for each one of the text lines, the proposed method is able to keep the identity and location of all the text regions in the scene. For all the sequences, the system is able to quickly spot the text in its original orientation and maintain the region identity throughout the duration of the video.

5.3 Performance

The system operates at video rate (average 25 fps) on the video sequences used for our experiments. They were measured on a standard PC with an Intel Core 2 Quad Q6600 processor and 8 Gb of RAM. A breakdown of the times spent by the algorithm on each of the stages is presented in Table 2. The most expensive stage in terms of computation time is the tracking observation, which includes the feature matching. The text segmentation and perspective estimation are also major contributors to the time needed to process one frame. Those stages are split into a pipeline and are automatically distributed between the processor cores to achieve parallelism. The OCR task is comparatively much slower than the rest of the stages together. This demonstrates one of the benefits of text tracking: OCR runs on an independent thread and thus does not contest with the main processing pipeline to achieve the desired frame rate. When the recognition results are ready, the region identity maintained by the text tracking is used to assign the recognised text to the correct text region.

Table 2 Time spent on each stages of the algorithm

acquisition	5 ms
segmentation	22 ms
aggregation	5 ms
perspective estimation	27 ms
tracking: prediction	8 ms
tracking: observation	42 ms
OCR	250 ms

6 Conclusions

We have presented a text reading system based on text tracking. It operates autonomously and in real-time, automatically detecting and recognising new text regions and discarding the old ones. We have shown quantitative and qualitative analysis of the performance and capabilities and of our prototype. We are also releasing our scene text tracking dataset and its associated ground-truth data to enable future comparative studies.

The area of text tracking is very young and there is still a lot to be accomplished. Although we feel we have presented a novel step towards fast text segmentation and tracking, there remain a number of shortcomings and newer goals that are yet to be addressed. Our method is focused on larger text and is not suited to deal with smaller document texts. Also, as a matter of trading accuracy for speed, we do not necessarily use state-of-the-art text segmentation. There remain other avenues to explore, for example, (a) exploiting the OCR results from multiple frames to combine and obtain a more accurate global recognition (e.g. to bypass reflections and occlusions), (b) combining rectified images from several frames to help construct super-resolution and/or larger (by mosaicking) images. Moreover, we still have to study and understand how the tracking information can help build a better user interface for assistive devices. Observing the patterns of movement and context in the surroundings is crucial for deciding when and how to read text back to the user, enabling a more useful interaction experience. We plan to develop these ideas as part of our future work.

7 Acknowledgments

This work was carried out at Bristol University by Carlos Merino-Gracia, who is funded by the Spanish Ministerio de Industria y Comercio (project IPT-2012-0961-300000).

8 References

1 Epshtein, B., Ofek, E., Wexler, Y.: 'Detecting text in natural scenes with stroke width transform'. Computer Vision and Pattern Recognition, 2010, pp. 2963–2970

2 Pan, Y.-F., Hou, X., Liu, C.-L.: 'A hybrid approach to detect and localize texts in natural scene images', *Trans. Image Process.*, 2011, **20**, (3), pp. 800–813

3 Neumann, L., Matas, J.: 'Real-time scene text localization and recognition'. Computer Vision and Pattern Recognition, 2012, pp. 3538–3545

4 Chen, H., Tsai, S., Schroth, G., Chen, D., Grzeszczuk, R., Girod, B.: 'Robust text detection in natural images with edge-enhanced maximally stable external regions'. Int. Conf. on Image Processing, 2011, pp. 2609–2612

5 Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: 'ICDAR 2003 robust reading competitions'. Int. Conf. on Document Analysis and Recognition, 2003, pp. 682–687
6 Myers, G.K., Bolles, R.C., Luong, Q.-T., Herson, J.A., Aradhye, H.B.: 'Rectification and recognition of text in 3-D scenes', *Int. J. Doc. Anal. Recognit.*, 2005, **7**, pp. 147–158
7 Merino-Gracia, C., Mirmehdi, M., Sigut, J., González-Mora, J.L.: 'Fast perspective recovery of text in natural scenes', *Image Vis. Comput.*, 2013, **31**, pp. 714–724
8 Merino, C., Mirmehdi, M.: 'A framework towards realtime detection and tracking of text'. Camera Based Document Analysis and Recognition, 2007, pp. 10–17
9 Li, H., Doermann, D., Kia, O.: 'Automatic text detection and tracking in digital video', *Trans. Image Process.*, 2000, **9**, (1), pp. 147–156
10 Lienhart, R., Wernicke, A.: 'Localizing and segmenting text in images and videos', *Circuits Syst. Video Technol.*, 2002, **12**, (4), pp. 256–268
11 Gllavata, J., Ewerth, R., Freisleben, B.: 'Tracking text in MPEG videos'. Int. Conf. on Multimedia, 2004, pp. 240–243
12 Myers, G.K., Burns, B.: 'A robust method for tracking scene text in video imagery'. Camera Based Document Analysis and Recognition, 2005, vol. 1
13 Shiratori, H., Goto, H., Kobayashi, H.: 'An efficient text capture method for moving robots using DCT feature and text tracking'. Int. Conf. on Pattern Recognition, 2006, pp. 1050–1053
14 Tanaka, M., Goto, H.: 'Autonomous text capturing robot using improved DCT feature and text tracking'. Int. Conf. on Document Analysis and Recognition, 2007, 2, pp. 1178–1182
15 Tanaka, M., Goto, H.: 'Text-tracking wearable camera system for visually-impaired people'. Int. Conf. on Pattern Recognition, 2008, pp. 1–4
16 Goto, H., Tanaka, M.: 'Text-tracking wearable camera system for the blind'. Int. Conf. on Document Analysis and Recognition, 2009, pp. 141–145
17 Na, Y., Wen, D.: 'An effective video text tracking algorithm based on sift feature and geometric constraint'. Advances in Multimedia Information Processing, 2010, pp. 392–403
18 Minetto, R., Thome, N., Cord, M., Leite, N.J., Stolfi, J.: 'Snoopertrack: text detection and tracking for outdoor videos'. Int. Conf. on Image Processing, 2011, pp. 505–508
19 Lowe, D.G.: 'Distinctive image features from scale-invariant keypoints', *Int. J. Comput. Vis.*, 2004, **60**, (2), pp. 91–110
20 Hartley, R.I., Zisserman, A.: 'Multiple view geometry in computer vision' (Cambridge University Press, 2004, 2nd edn.)
21 Phan, T.Q., Shivakumara, P., Lu, T., Tan, C.L.: 'Recognition of video text through temporal integration'. Int. Conf. on Document Analysis and Recognition, 2013, pp. 589–593
22 Mosleh, A., Bouguila, N., Hamza, A.: 'Automatic inpainting scheme for video text detection and removal', *Trans. Image Process.*, 2013, **22**, (11), pp. 4460–4472
23 Wang, K., Babenko, B., Belongie, S.: 'End-to-end scene text recognition'. Int. Conf. on Computer Vision, 2011, pp. 1457–1464
24 Mishra, A., Alahari, K., Jawahar, C.: 'Top-down and bottom-up cues for scene text recognition'. Computer Vision and Pattern Recognition, 2012, pp. 2687–2694
25 González, A., Bergasa, L.M.: 'A text reading algorithm for natural images', *Image Vis. Comput.*, 2013, **31**, (3), pp. 255–274
26 Merino-Gracia, C., Lenc, K., Mirmehdi, M.: 'A head-mounted device for recognizing text in natural scenes', in Iwamura, M., Shafait, F. (Eds.): 'Camera based document analysis and recognition', 2012, (*LNCS*, **7139**), pp. 29–41
27 Targhi, A., Hayman, E., Eklundh, J., Shahshahani, M.: 'The Eigen-transform & applications'. Asian Conf. of Computer Vision, I, 2006, pp. 70–79
28 de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: 'Computational geometry: algorithms and applications' (Springer-Verlag, 2000, 2nd edn.)
29 Pilu, M.: 'Extraction of illusory linear clues in perspectively skewed documents'. Computer Vision and Pattern Recognition, 2001, pp. 363–368
30 Toussaint, G.: 'Solving geometric problems with the rotating calipers'. Mediterranean Electrotechnical Conf., 1983, pp. 10–17
31 Doucet, A., de Freitas, J., Gordon, N.: 'Sequential Monte Carlo methods in practice' (Springer-Verlag, 2001)
32 Wan, E., Van Der Merwe, R.: 'The unscented Kalman filter for nonlinear estimation'. Adaptive Systems for Signal Processing, Communications, and Control, 2000, pp. 153–158
33 Klein, G., Murray, D.: 'Parallel tracking and mapping for small AR workspaces'. ISMAR, 2007, pp. 225–234
34 Kasturi, R., Goldgof, D., Soundararajan, P., *et al.*: 'Framework for performance evaluation of face, text, and vehicle detection and tracking in video: data, metrics, and protocol', *Pattern Anal. Mach. Intell.*, 2009, **31**, (2), pp. 319–336
35 Bernardin, K., Stiefelhagen, R.: 'Evaluating multiple object tracking performance: the CLEAR MOT metrics', *J. Image Video Process.*, 2008, **2008**, pp. 1:1–1:10

9 Appendix: Homography decomposition

As noted by Hartley and Zisserman [20], a homography can be decomposed into three partial transformations $\boldsymbol{H}_{\mathrm{S}}$, $\boldsymbol{H}_{\mathrm{A}}$ and $\boldsymbol{H}_{\mathrm{P}}$, each of them representing a separate 'similarity', 'affinity' and 'projectivity', respectively

$$\boldsymbol{H} = \boldsymbol{H}_{\mathrm{S}}\boldsymbol{H}_{\mathrm{A}}\boldsymbol{H}_{\mathrm{P}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0^{\top} & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{SK} & 0 \\ 0^{\top} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{l}^{\top} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{l}^{\top} & 1 \end{bmatrix} \qquad (20)$$

where $\boldsymbol{A}$ is a non-singular matrix defined as $\boldsymbol{A} = \boldsymbol{RSK} + \boldsymbol{tl}^{\mathrm{T}}$, $\boldsymbol{R}$ is a rotation matrix, $\boldsymbol{S}$ is an anisotropic scaling matrix, $\boldsymbol{K}$ is a shear matrix, ***hbft*** is a translation vector and $\boldsymbol{l}$ is a perpsective foreshortening vector. They are further defined as

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \boldsymbol{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$
$$\boldsymbol{K} = \begin{bmatrix} 1 & \sigma \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \text{ and } \boldsymbol{l} = \begin{bmatrix} l_x \\ l_y \end{bmatrix} \qquad (21)$$

With (20) and (21) the function $\boldsymbol{H} = \mathcal{H}(\boldsymbol{h})$ is completely specified. When given the homography $\boldsymbol{H}$, defined as

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

the four direct parameters are trivially obtained as

$$t_x = h_{13}, \quad t_y = h_{23}, \quad l_x = h_{31}, \quad l_y = h_{32} \qquad (22)$$

then we define four auxiliary variables

$$g_1 = h_{11} - t_x v_x, \quad g_3 = h_{12} - t_x v_y \qquad (23)$$

$$g_2 = h_{21} - t_y v_x, \quad g_4 = h_{22} - t_y v_y \qquad (24)$$

to be able to obtain the four remaining parameters as

$$\theta = \arctan\left(\frac{g_2}{g_1}\right), \quad \sigma = \frac{g_1 g_3 + g_2 g_4}{g_1^2 + g_2^2} \qquad (25)$$

$$s_x = +\sqrt{g_1^2 + g_2^2}, \quad s_y = \frac{g_1 g_4 - g_2 g_3}{+\sqrt{g_1^2 + g_2^2}} \qquad (26)$$

that completely specify the inverse function $\boldsymbol{h} = \mathcal{H}^{-1}(\boldsymbol{H})$.