A Video-Based System for Vehicle Speed Measurement in Urban Roadways

Diogo C. Luvizon, Bogdan T. Nassu and Rodrigo Minetto

***Abstract*—In this paper, we propose a non-intrusive, video-based system for vehicle speed measurement in urban roadways. Our system uses an optimized motion detector and a novel text detector to efficiently locate vehicle license plates in image regions containing motion. Distinctive features are then selected on the license plate regions, tracked across multiple frames, and rectified for perspective distortion. Vehicle speed is measured by comparing the trajectories of the tracked features to known real world measures. The proposed system was tested on a data set containing approximately five hours of videos recorded in different weather conditions by a single low-cost camera, with associated ground truth speeds obtained by an inductive loop detector. Our data set is freely available for research purposes. The measured speeds have an average error of -0.5 km/h, staying inside the [-3,+2] km/h limit determined by regulatory authorities in several countries in over 96.0% of the cases. To the authors' knowledge, there are no other video-based systems able to achieve results comparable to those produced by an inductive loop detector. We also show that our license plate detector outperforms other two published state-of-the-art text detectors, as well as a well-known license plate detector, achieving a precision of 0.93 and a recall of 0.87.**

***Index Terms*—vehicle speed measurement; license plate detection; feature tracking; vehicle motion detection.**

I. Introduction

Systems for vehicle detection and speed measurement play an important role in enforcing speed limits. They also provide relevant data for traffic control. Those systems are divided in intrusive and non-intrusive [1]. Intrusive sensors, usually based on inductive loop detectors, are widely used, but have complex installation and maintenance, accelerate asphalt deterioration, and can be damaged by wear and tear. Non-intrusive sensors, which include laser meters and Doppler radars, avoid these problems, but are usually more expensive and require frequent maintenance. As digital cameras become cheaper and able to produce images with higher quality, video-based systems can become a lower cost alternative for non-intrusive speed measurement. In fact, existing systems are often connected to video cameras [2] that record the license plates of vehicles that exceed the speed limit — thus, the infrastructure for such systems is already available in most cases.

In this paper we describe the pipeline for a non-intrusive video-based system for vehicle speed measurement in urban roadways. Our goal is measuring vehicle speeds with accuracy comparable to that obtained by a system based on inductive loop detectors. The input video is captured by a single fixed overhead camera, positioned so that the rear license plate of vehicles in three adjacent lanes are clearly visible, as shown in Fig. 1. A sample image from this setup is shown in Fig. 2. This setup allows the same images to be used for both speed measurement and license plate identification (e.g. for locating stolen vehicles, or in the case of a speed limit violation).

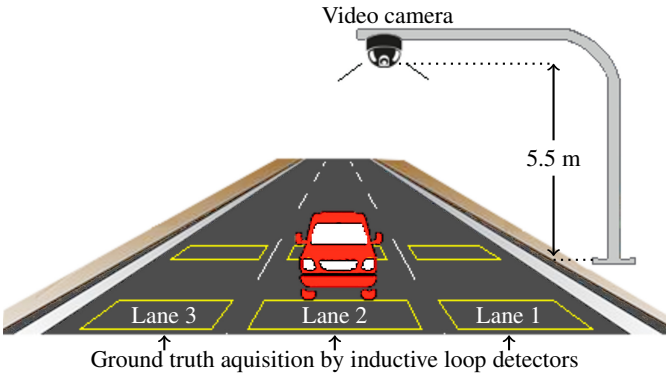


Fig. 1. System setup.

Fig. 2. Sample image captured by our system.

We make some assumptions about the scene and the problem domain: video frames are equally spaced in time; each lane lies on a plane; the vehicles move at a constant speed and with a straight trajectory from the lower to the upper part of the image; and the license plates are at approximately the same distance from the ground. These assumptions allow us to measure vehicle speeds without modeling the 3-D space, or requiring precise camera calibration or positioning.

The proposed system works by tracking sets of distinctive features extracted from image regions around each vehicle's

Diogo C. Luvizon, Bogdan T. Nassu and Rodrigo Minetto are with the Department of Informatics, Federal University of Technology - Paraná (UTFPR), Curitiba, Brazil (e-mail: diogoluvizon@gmail.com, nassu@dainf.ct.utfpr.edu.br, rminetto@dainf.ct.utfpr.edu.br)

Manuscript received X; revised X.

license plate, and is divided into five main parts, as shown in Fig. 3. Initially, an optimized motion detection algorithm identifies image regions containing moving vehicles. These regions are fed to a novel license plate detector, which returns a set of axis-aligned rectangular sub-images around the vehicles' license plates. Features are then extracted from each sub-image [3], and tracked using the Kanade-Lucas-Tomasi (KLT) algorithm [4]. To cope with large displacements, from vehicles moving at high speeds, an initial motion estimation is performed by matching features extracted by the Scale-Invariant Feature Transform [5]. Finally, vehicle speeds are measured by comparing the trajectories of the tracked features to known real world measures.

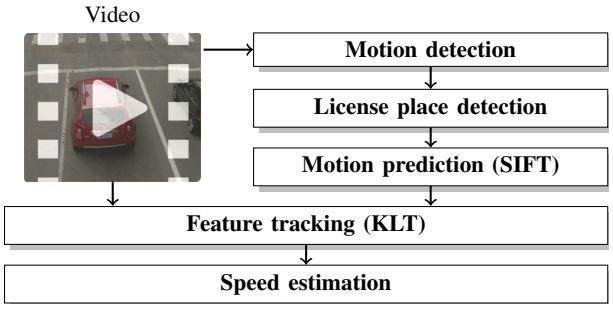


Fig. 3. Overview of the proposed system.

A proof-of-concept of our system was evaluated on approximately five hours of videos in different weather and recording conditions. The videos have an associated ground truth dataset containing vehicle speeds measured by a high precision system based on inductive loop detectors, properly calibrated and approved by the Brazilian national metrology agency (Inmetro). This data set is itself a contribution of our work, and can be freely obtained for research purposes[1]. Our system was able to measure speeds with an average error of -0.5 km/h, staying inside the [-3,+2] km/h limit determined by regulatory authorities in several countries, in over 96.0% of the cases. We also show that our license plate detector outperforms other two published state-of-the-art text detectors, as well as a well-known license plate detector.

A preliminary version of the system described here was published at the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [6]. The system described in the present paper differs from that version in several aspects, such as the inclusion of an initial motion detection step, a new algorithm for license plate detection, and refined computations for the final measured speed. The new system was evaluated on a broader and deeper manner.

The rest of this paper is divided as follows. In Section II, we discuss related work. In Sections III, IV, V and VI we present our motion detection, license plate detection, vehicle tracking and speed measurement methods, respectively. Experimental evaluation and results are reported in Section VII. Finally, in Section VIII we state the conclusions.

[1]The full dataset will be made available at the time of publication, a sample is available at www.dainf.ct.utfpr.edu.br/%7erminetto/projects/vehicle-speed.

II. Related work

A. Vehicle speed estimation and measurement

Several video-based approaches were proposed for estimating or measuring the speed of vehicles in roadways. Most methods include a background/foreground segmentation step to detect image regions containing motion. Common approaches for this task include simple frame differences [7], [8], [9], [10], as well as statistic models based on medians [11], [12], [13], gaussian distributions [14] or other measures [15]. Vehicle speeds are estimated by tracking image features or regions, including blobs [12], image patches [14], edges [7], [11], corners [8], [9], the license plate region [16], [17], [18], or a combination of such features [13]. Rectification for perspective distortion is also a step found in most methods, and may occur before or after feature tracking.

Although the cited methods have some steps in common, they also have fundamental differences, not only on the way they measure vehicle speeds, but also on the type of scenario they can be used.

Methods based on direct blob analysis [7], [8], [10], [12], [14], [15] are sensitive to conditions such as shadows, perspective, and illumination variations. Moreover, these methods produce satisfactory results only when the camera is positioned high above the roadway, with the blobs being tracked for many frames. The same issues affect methods which use other types of features, but still compute them from blobs, such as those proposed by Zhiwei *et al.* [11], which detects edges near the limits of each blob, or Palaio *et al.* [13], which extracts from each blob features such as derivatives, Laplacian and color. As discussed in Section VII, we have compared our system with a blob tracking approach based on a particle filter, similar in concept to the one proposed by Maduro *et al.* [12].

The method from Dogan *et al.* [9] avoids the problems associated with blob analysis by directly tracking distinctive features using the Lucas-Kanade optical flow algorithm [4]. However, their method assumes that all the tracked features belong to the same vehicle — thus it can handle only a single vehicle at a time. Moreover, they do not take perspective into account, and require a side view of the vehicles.

The work from Garibotto *et al.* [16] relies on how characters detected by an optical character recognition (OCR) algorithm vary in size and position. Their method demands a very robust OCR, and did not produce acceptable results even in a controlled environment — average errors for a single-camera setup ranged from 3% to 13%. A similar issue was observed in the work from Czajewski and Iwanowski [17], which is also based on license plate recognition. Note that, although our system has a license plate detection step, it does not require the characters to be precisely segmented or recognized.

B. License plate detection

Although license plate detection is not our primary concern in this work, it is a necessary step for the proposed

speed measurement system. The surveys from Anagnostopoulos *et al.* [19] and Du *et al.* [20] review state-of-the-art license plate detection algorithms up to 2013. Those algorithms rely on attributes such as edges, texture, color, shape, and geometry — extracted using techniques such as the Sobel operator [18], [21], the Canny detector [22], conditional random field [23], template-matching [24], wavelets [25], or the Scale Invariant Feature Transform (SIFT) [26]. The faced challenges include poor maintenance, occlusion, variations in position and illumination, complex backgrounds, low contrast, low image resolution, and motion blur.

As detailed in Section VII, the specialized license plate detector employed by our system was compared with three other alternatives. The Zheng *et al.* algorithm [21] uses a rectangular sliding window to identify image regions with high gradient density, which probably contain license plates. The Stroke Width Transform (SWT), by Epshtein *et al.* [22], is a text detector based on the orientations of the gradients over edge pixels, which are used to determine a local stroke width for candidate characters. SnooperText, by Minetto *et al.* [27], is a multi-scale text detector that uses morphological image segmentation and character/non-character classification based on shape descriptors. SnooperText validates candidate text regions using the T-HOG classifier [28], a specialized gradient-based descriptor tuned for single-line text regions. All these algorithms include steps of pre-processing and tests for filtering out incorrect results, based on size and geometry.

III. Motion detection

The first step in our system's pipeline is detecting moving vehicles, limiting further processing to a set of regions of interest. Ideally, each region of interest must contain the entire license plate from a single vehicle. An overview of the motion detection approach that we develop is shown in Fig. 4.

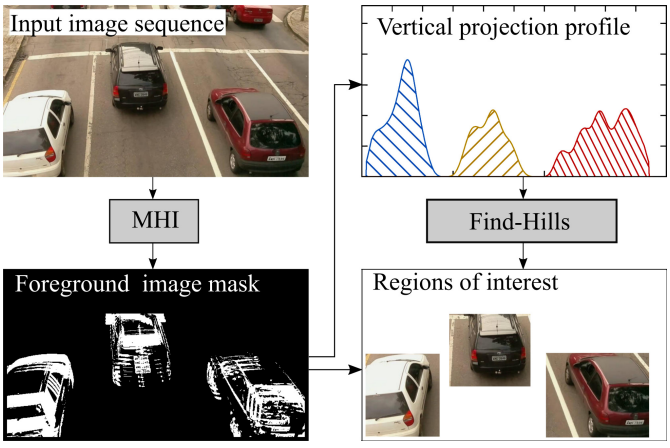


Fig. 4. Motion detection: regions of interest are delimited based on a foreground image mask and a vertical projection profile.

Motion detection begins with a rough foreground / background segmentation. We use the *Motion History Image* (MHI) concept from Bobick and Davis [29]. The MHI $\mathbb{H}$ for time t is given by

$$\mathbb{H}(x,y,t) = \begin{cases} \tau & \text{if} \quad \mathbb{D}(x,y,t) = 1, \\ \max(0,\ \mathbb{H}(x,y,t-1) - 1) & \text{otherwise.} \end{cases} \quad (1)$$

where $\mathbb{D}$ are the binary images obtained from thresholded frame differences, and the τ parameter represents the duration of the expected motion in frame units. In our tests, we used $\tau = 5$. The binary segmentation mask $\mathbb{M}$ is obtained by

$$\mathbb{M}(x,y,t) = \begin{cases} 1 & \text{if} \quad \mathbb{H}(x,y,t) > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

An example is shown in Fig. 4. Note that we do not have to identify vehicle boundaries in a precise manner, as our system does not rely on blob analysis.

To reduce the processing time when generating image $\mathbb{M}$ and in subsequent steps, the images are subsampled — i.e. the values of x and y in (1) are restricted to a regular sparse grid, with pixels outside the grid being skipped. The segmentation may become less precise, but this is acceptable as long as the vehicle license plate remains entirely within its corresponding region. In tests involving the complete system, we observed that a subsampling factor of 4 in both axes (i.e. processing 1 of each 16 pixels) reduced the processing time to 16.47% of the original time, without any loss in detection performance.

After the sub-sampled binary segmentation mask $\mathbb{M}$ is obtained, we perform a vertical *projection profile* analysis [30] to separate vehicles horizontally. We take the lower part of image $\mathbb{M}$, which shows the region closer to the camera, and count the foreground pixels in each column, generating a histogram with n bins (for n image columns). This histogram is smoothed, to reduce noise, and interpreted as an array Ψ.

Figure 4 shows an example of vertical projection profile. It can be seen that the interval containing a vehicle is delimited by an ascending and a descending slope, corresponding respectively to the left and right boundaries of the vehicle. The FIND-HILLS routine (Fig. 5) is used to determine these boundaries for each vehicle. It receives as parameters the projection profile array Ψ, and a threshold ρ (0.1, in our tests) that defines the minimum angle of inclination for a boundary. It returns a pair of lists $\{A, D\}$, such that each list element represents a hill's ascending and descending border, respectively.

In step 2 of FIND-HILLS, we call the FIND-INCLINATION routine, outlined as Fig. 6. The purpose of this routine is to determine the rising and falling phases of Ψ (see Fig. 8(a)), given as arrays R and F, respectively. The threshold $0 \leq \rho \leq 1$ is used to discard false phases: since vehicle regions are represented by high values in the projection profile, it prevents against incorrectly dividing a vehicle in two regions.

In steps 3 and 4 of FIND-HILLS, we call the PHASES routine, outlined as Fig. 7. This function scans the R and F arrays, with the order depending on whether we are looking for ascending or descending regions. In Fig. 8(b), we show the values for these functions over a sample projection profile.

By pairing the ascending and descending boundaries produced by FIND-HILLS in arrays A and D, we can determine the left and right boudaries of each region of interest, as

function FIND-HILLS($\Psi[1,\ldots,n]$, ρ)
  $\{R, F\} \leftarrow$ FIND-INCLINATION(Ψ, ρ);
  $S_a \leftarrow$ PHASES($R[1\ldots n]$, $F[1\ldots n], n$); ▷ Ascending
  $S_d \leftarrow$ PHASES($R[n\ldots 1]$, $F[n\ldots 1], n$); ▷ Descending
  $A \leftarrow \{\}$; $D \leftarrow \{\}$; ▷ set initialization
  for each $x \in \{1,\ldots,n-1\}$ **do**
    if $(S_a[x] = 0)$ **and** $(S_a[x+1] = 1)$ **then**
      $A \leftarrow A \cup \{x\}$; ▷ slope ascending
    end if
    if $(S_d[x] = 1)$ **and** $(S_d[x+1] = 0)$ **then**
      $D \leftarrow D \cup \{x\}$; ▷ slope descending
    end if
  end for
  return $\{A, D\}$;
end function

Fig. 5. Routine to find hills in Ψ.

function FIND-INCLINATION($\Psi[1,\ldots,n]$, ρ)
  for each $x \in \{1,\ldots,n\}$ **do**
    $R[x] \leftarrow F[x] \leftarrow 0$; ▷ array initialization
  end for
  for each $x \in \{1,\ldots,n-1\}$ **do**
    $\delta \leftarrow (1 - (\Psi[x]/\Psi[x+1]))$;
    if $\delta > \rho$ **then**
      $R[x] \leftarrow 1$; ▷ rising phase
    end if
    if $\delta < -\rho$ **then**
      $F[x] \leftarrow 1$; ▷ falling phase
    end if
  end for
  return $\{R, F\}$;
end function

Fig. 6. Routine to find rising and falling phases in Ψ.

function PHASES($R[\ldots]$, $F[\ldots], n$)
  for each $x \in \{1,\ldots,n\}$ **do**
    $S[x] \leftarrow 0$; ▷ array initialization
  end for
  for each $x \in \{2,\ldots,n-1\}$ **do**
    if $(R[x] = 0)$ **and** $(R[x+1] = 1)$ **then**
      $S[x] \leftarrow 1$;
    else if $(F[x] = 0)$ **and** $(F[x+1] = 1)$ **then**
      $S[x] \leftarrow 0$;
    else
      $S[x] \leftarrow S[x-1]$;
    end if
  end for
  return S;
end function

Fig. 7. Routine to compute the array of ascending slopes.

exemplified in Fig. 8(c). We assume each region of interest corresponds to a vehicle in the image. Upper and lower boundaries for each region are obtained directly from the binary segmentation mask $\mathbb{M}$. To guarantee a vehicle's license plate is inside its corresponding region of interest, we discard regions with lower boundaries close to the image bottom — these cases may correspond to a vehicle that is entering the frame, so its license plate is not visible yet.

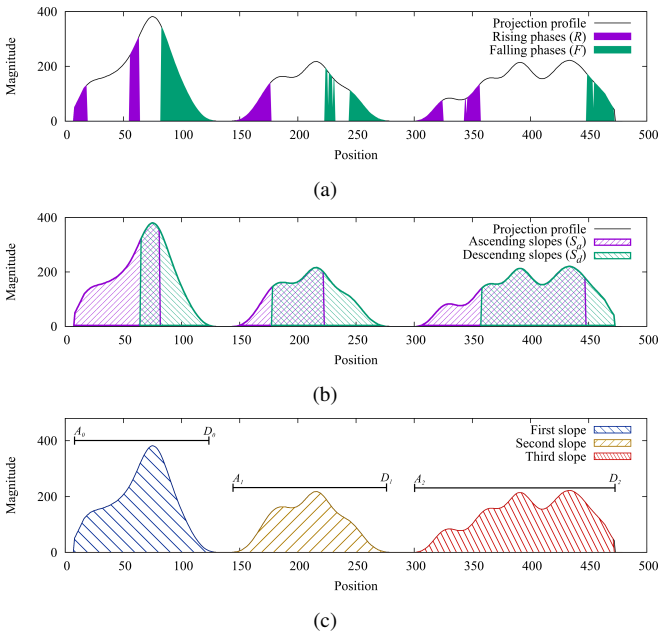


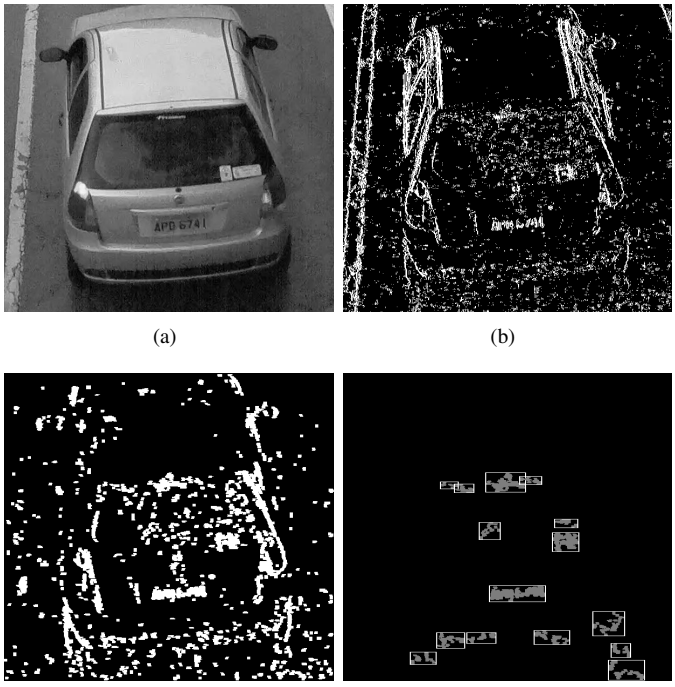
Fig. 8. Internal steps of the FIND-HILLS routine: rising and falling phases according to a given threshold ρ (a), ascending and descending slopes (b), and three slope regions delimited by the rising edge of ascending slopes and the falling edge of descending slopes (c).

IV. LICENSE PLATE DETECTION

The motion detector produces one region of interest for each moving vehicle present in the scene at a given time. The license plate detector finds, for each region of interest, an axis-aligned rectangle, which is an approximate bounding box of the vehicle's license plate region. This procedure is performed for each region of interest only until a license plate is detected — afterwards, features extracted from the license plate region are tracked across frames, as explained in Section V.

Our detector follows the *hypothesis generation and validation paradigm* [27]. Namely, in the hypothesis generation phase (outlined in Fig. 9) we use *edge extraction*, *edge filtering*, and *region grouping* modules to provide coarse candidate regions based on the edge attribute that makes up the license plate. At this phase, we aim to isolate the license plate region and prevent false negatives, even at the cost of several false positives. In the *hypothesis validation* phase, we use a *region classification* module to refine the candidates. For this classification we use the *Text HOG* (T-HOG) descriptor [28], which is based on the observation that the license plate textual information can often be characterized by the distribution of the directions of the image gradients.

For edge extraction, we follow the observation from Zheng *et al.* [21] that background areas around the license plate region often have large horizontal edges or small random noise. Thus, we extract only the vertical image gradients $\mathbb{G}_x$ by convolving the input image with a 3×3 horizontal Sobel operator. The gradients are then stored in a binary edge image



(a) (b) (c) (d)

Fig. 9. Hypothesis generation for license plate detection: (a) input image; (b) vertical edges; (c) filtered and dilated edges; (d) candidate regions.

$\mathbb{E}$ by comparing each gradient magnitude with the average magnitude μ of $\mathbb{G}_x$ multiplied by some threshold τ (τ = 2 in our tests), that is

$$\mathbb{E}(x,y) = \begin{cases} 1 & \text{if} \quad |\mathbb{G}_x(x,y)| > \mu\tau \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

An example of edge extraction is exemplified in Fig. 9 (b). As the sign of the gradients is not taken into account, this scheme is invariant to distinctive license plate color schemes.

Edge filtering is performed to remove from $\mathbb{E}$ edges that are too small (below 4 pixels) or too large (above 120 pixels). This is done using a standard connected component labeling algorithm and direct comparisons with given minimum and maximum values for each component's width and height. Neighboring vertical edges that remain after filtering are merged by performing a morphological dilation using a centered 1×7 structuring element. Figure 9 (c) shows an image example containing filtered and dilated edges.

In order to avoid super-segmenting the license plate, we group candidate regions according to the geometric criteria defined by Retornaz and Marcotegui [31]. These criteria take into account the heights h_1, h_2 and widths w_1, w_2 of two bounding boxes b_1 and b_2, as well as the coordinates (x_1, y_1) and (x_2, y_2) of their centers. Specifically, let $h = \min(h_1, h_2)$, $d_x = |x_1 - x_2| - (w_1 + w_2)/2$, and $d_y = |y_1 - y_2|$. Then b_1 and b_2 are said to be *compatible* — that is, assumed to belong to the same object — if and only if

$$\begin{aligned} |h_1 - h_2| &< t_1\, h \\ d_x &< t_2\, h \\ d_y &< t_3\, h \end{aligned} \tag{4}$$

where t_1, t_2 and t_3 are fixed parameters (respectively, 0.7, 1.1 and 0.4 in our tests).

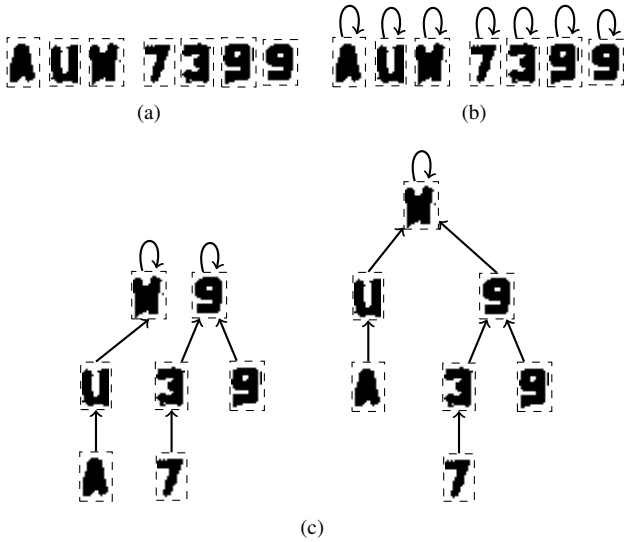
The above criteria are applied to each isolated region by using the union-find data structure, which was adapted from Cormen [32] as shown in Fig. 10. Specifically, at the beginning each region b is a disjoint set created by the MAKE-SET algorithm, as shown in Fig. 11 (a,b). The UNION routine then tries to group two *compatible* candidate regions $b1, b2$, as shown in Fig. 11 (c). These regions are then filtered using simple geometric tests that remove regions with dimensions not compatible with license plates. In our tests we filtered regions with dimensions below 32×10 pixels. Figure 9 (d) shows the grouped and filtered regions for a sample image.

```
1: function MAKE-SET (b)
2:     father[b] = b;

1: function FIND-SET (b)
2:     if father[b] = b then
3:         return b;
4:     else
5:         return FIND-SET (father[b]);

1: function UNION (b1, b2)
2:     f1 ← FIND-SET (b1);          ▷ f1 is the father of b1
3:     f2 ← FIND-SET (b2);          ▷ f2 is the father of b2
4:     if (f1 ≠ f2) then
5:         if COMPATIBLE (b1, b2) (see equation 4) then
6:             father[f2] = f1;
```

Fig. 10. MAKE-SET, FIND-SET and UNION routines, adapted from Cormen *et al.* [32].



(a) (b) (c)

Fig. 11. Region grouping: (a) region bounding boxes of a sample image; (b) MAKE-SET routine applied to all regions, the arrows indicate the node parent; (c) result of UNION$(w, 7)$. Adapted from Cormen *et al.* [32].

The last stage of our license plate detector is a region classification step, which discards regions that do not seem to contain any textual information. We use for this task the T-HOG text descriptor [28] which is a texture classifier specialized for capturing the gradient distribution characteristic

of character strokes in occidental-like scripts. We first estimate a center line for each candidate image region, by taking, at each column, the center point between the uppermost and the bottommost pixels from the filtered and dilated edge image, as shown in Fig. 12 (a,b). Fixed-size windows are then centered at regularly spaced points along the center line, as shown in Fig. 12 (c). This sub-sampling is done to reduce the computational effort, as well as to avoid very similar classifications in neighboring positions. For each window, we compute the T-HOG descriptor and use it as an input for a SVM classifier. For the SVM classifier we used a Gaussian χ^2 kernel, whose standard deviation parameter was optimizied by cross-validation on a training set of text and non-text samples as described by Minetto *et al.* [28] [2]. The classifier output is thresholded to give a binary text/non-text region classification. Figure 12 (d) shows the windows classified as text for a sample image.

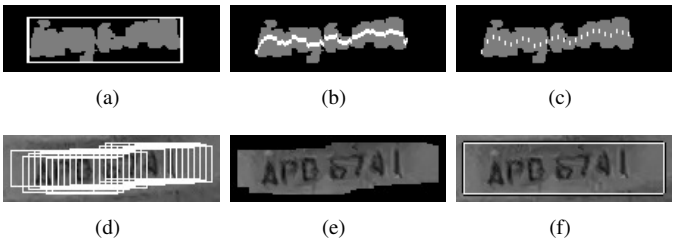


Fig. 12. T-HOG/SVM classification: (a) the candidate region; (b) the region center line in white color; (c) sampled points to guide the classification in white color (step of 4 pixels); (d,e) regions classified as text; (f) the text region bounding box.

The final license plate detection is performed by taking regions containing a large number of windows classified as text (Fig. 12 (e)). Note that vehicles displaying textual labels and advertisements may have multiple regions satisfying this condition. In those cases, we select the region closest to the image bottom, which corresponds to the license plate in most cases. The license plate is approximated by a single axis-aligned rectangle that encloses all the text windows from the selected region, as shown in Fig. 12 (f). Note that this rectangle does not have to be accurately adjusted to the license plate — our system will work as expected as long as the feature tracking module, explained in Section V, can detect enough distinctive features inside the rectangle.

A final observation about our detector is that, in our data set, the license plates from motorcycles have different dimensions than other vehicles. Moreover, they contain two text lines instead of one, and smaller letters and digits, which are more frequently merged to each other. Our detector still works for those cases, requiring only adjustments to some thresholds, as well windows spread over the entire region for the T-HOG/SVM classification, instead of following the center line.

V. Feature Selection and Tracking

[2]We used the source code and training dataset available at www.dainf.ct.utfpr.edu.br/%7erminetto/projects/thog.html.

Once a license plate region is detected, our system selects a set of distinctive features and tracks it across multiple video frames. Our aim is producing a list containing the trajectories of the tracked features. The structure of our tracking scheme is outlined in Fig. 13.

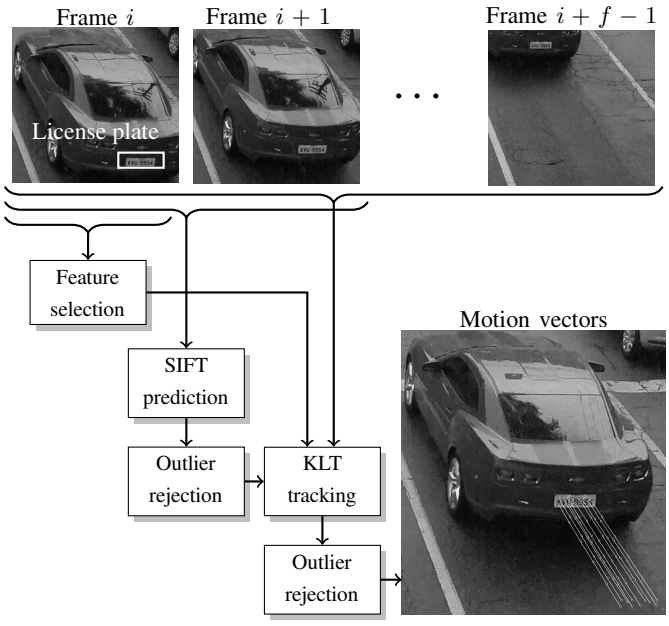


Fig. 13. Overview of the proposed feature tracking method.

Feature selection is performed only once for each vehicle, immediately after its license plate is detected. Following the approach from Shi and Tomasi [3], a "*good feature*" is a region with high intensity variation in more than one direction, such as textured regions or corners. Let $[\mathbb{I}_x\ \mathbb{I}_y]$ be the image derivatives in the x and y directions of image $\mathbb{I}$, and let

$$\mathrm{Z} = \sum_{\Omega} \begin{bmatrix} {\mathbb{I}_x}^2 & \mathbb{I}_x\mathbb{I}_y \\ \mathbb{I}_x\mathbb{I}_y & {\mathbb{I}_y}^2 \end{bmatrix} \tag{5}$$

be the 2×2 gradient matrix in a given window Ω. The region covered by the window is selected if both eigenvalues of Z are above a given threshold. In our system, we used a threshold of 1, which leads to a large number of features, but track only the 10 features with the highest eigenvalues.

The selected features are tracked with subpixel accuracy using the pyramidal Kanade-Lucas-Tomasi (KLT) [4] algorithm. Let $\mathbb{I}$ and $\mathbb{J}$ be two video frames such that $\mathbb{J}$ appears after $\mathbb{I}$ in the video sequence. The KLT algorithm takes a small window Ω around each feature extracted from $\mathbb{I}$, and looks for its corresponding window in $\mathbb{J}$. For a feature centered at position $u = (x_u, y_u)$, the corresponding window is the one that minimizes the sum of the squared errors, that is

$$E = \sum_{\Omega} \left[\mathbb{I}(u) - \mathbb{J}(u + \vec{d})\right]^2 \tag{6}$$

where $\vec{d}$ is the displacement vector that describes the motion of the feature between frames $\mathbb{I}$ and $\mathbb{J}$. To obtain $\vec{d}$, the KLT algorithm takes a current estimate $\vec{e}$ and iteratively solves for increments $\vec{\Delta d}$, namely

$$E = \sum_{\Omega} \left[\mathbb{I}(u) - \mathbb{J}(u + (\vec{e} + \vec{\Delta d})) \right]^2 \quad (7)$$

updating $\vec{e}$ at each iteration until it converges. As we are tracking features, the initial estimate for each frame may be the displacement $\vec{d}$ obtained for the previous frame.

The traditional Lucas-Kanade algorithm only works for small displacements (in the order of one pixel). To overcome this limitation, we consider the pyramidal version of KLT, described by Bouguet [33]. The algorithm builds, for each frame, a multi-scale image pyramid by using the original image at the pyramid base, and putting at each subsequent level a version of the image in the previous level with width and height reduced by half. The pyramidal KLT algorithm starts by finding the displacement vector $\vec{d}$ at the last pyramid level, using the result as the initial estimate for $\vec{d}$ in the next level, repeating the process until the pyramid base (i.e. the original image) is reached.

The number of levels ℓ in the pyramid determines the maximum allowed displacement for a feature in pixels, given by $2^{\ell+1}-1$ (*e.g.* for $\ell = 3$, the maximum displacement is of 15 pixels). Larger values of ℓ may allow for larger displacements, but they may also produce very small images at the upper pyramid levels, which may lead to confusion and large errors in the initial estimates. Another limitation of the algorithm is that for the first frame in a sequence (i.e. the moment a license plate is detected), there is no known motion. In these cases, it is common to start with $\vec{e} = (0, 0)$ in Equation 7. If there are large feature displacements, e.g. from a vehicle moving at a high speed, this initial estimate will be too far from the correct displacement, preventing the feature from being properly tracked.

To overcome the limitations described above, an initial value for $\vec{d}$ is estimated by using a different method for matching features extracted from $\mathbb{I}$ and $\mathbb{J}$. We have used the SIFT (Scale-Invariant Feature Transform) features proposed by Lowe [5]. SIFT is a popular method for detecting and describing image keypoints, being robust to illumination variations and a number of geometric distortions. Using the standard parameters proposed by Lowe [5], SIFT features are extracted from an expanded window around the license plate region from image $\mathbb{I}$, and matched to other features extracted from image $\mathbb{J}$, using the *nearest neighbor distance ratio* matching strategy described by Mikolajczyk [34]. The obtained matches can be used to compute the displacement of each SIFT feature between frames $\mathbb{I}$ and $\mathbb{J}$. The average feature displacement is then used as the initial value for $\vec{d}$ in Equation 7. Note that this process occurs only once for each detected license plate — after the motion is roughly predicted, the system relies on the KLT algorithm, which allows for faster and more accurate estimates.

The feature selection and tracking module produces, for each frame, a list of displacement vectors. Since we suppose all the features belong to the same license plate, we expect all the vectors computed for a frame will be similar. Outliers are rejected using an iterative method: for each set of displacement vectors, we compute the mean and standard deviation, and remove from the set those vectors outside the three-sigma deviation, with the process being repeated until the standard deviation is smaller than 0.5 pixel. The outlier rejection routine is used independently for the x and y directions, and applied both to the vectors computed by the KLT algorithm and the initial estimates obtained from matching SIFT features.

VI. Speed Measurement

Our system measures vehicle speeds based on the motion vectors obtained by the feature selection and tracking method. Each motion vector $\vec{d_i}$ can be associated with a measurement of a vehicle's instantaneous speed at a particular time, given in pixels per frame, in the image plane. The purpose of the speed measurement module is converting these measurements to kilometers per hour (km/h) in the real world.

An important assumption of our system is that each street lane lies on a plane. That assumption makes it possible for us to map the motion vectors $\vec{d_i}$, which are given in pixels in the image plane, to displacements $\vec{v_i}$, given in meters in the ground plane (see Fig. 14).

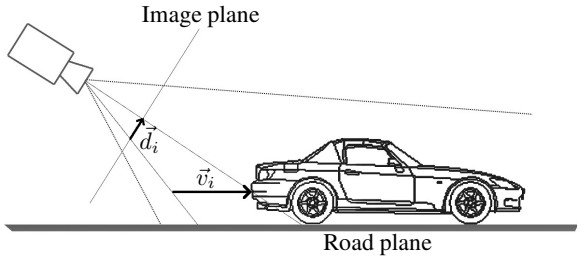


Fig. 14. Vehicle speed measurement scheme.

Assuming the pinhole camera model, this mapping can be made based on a single view of the scene, through a homography [35] — a plane-to-plane projective transformation — in a process that is sometimes referred to as *inverse perspective mapping* [36].

Given a 3×3 homography matrix H, an image point $p_i = (x_i, y_i)$ can be mapped to the point $\hat{p}_w = (x_w, y_w)$ in the world plane by

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \begin{bmatrix} zx_w \\ zy_w \\ z \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (8)$$

The homography matrix H may be obtained by associating four points in the image to known coordinates in the world plane. For our tests, we have used as references the markings left on the asphalt by the inductive loop detectors, which form a rectangle with 4.8m $\times$ 2.0m (but note that any large enough planar object could be used to this purpose). We assume that the top-left corner of the rectangle is the origin of the world coordinate system, and use a traditional technique to obtain H [37]. Different homography matrices were obtained for each road lane, i.e. instead of a single ground plane, we assume each lane lies on a different plane.

The output of the feature selection and tracking method, for each vehicle and each pair of frames, is a set of motion

vectors $\vec{d_i} = u_i(t) - u_i(t - \Delta t)$, where $u_i(t)$ is the feature position in the current video frame, $u_i(t - \Delta t)$ is the feature position in the previous video frame, Δt is the frame interval, and $i = \{1, 2, \ldots, n\}$ is a sequence of tracked features. We compute the feature displacements in the real world, denoted by $\vec{v_i}$, from Equation 8, namely

$$\vec{v_i} = Hu_i(t) - Hu_i(t - \Delta t) \tag{9}$$

The displacement in meters between frames $t - \Delta t$ and t for a motion vector $\vec{u_i}$ can be obtained by $\|\vec{v_i}\|$, the Euclidean norm of $\vec{v_i}$. Each displacement vector can be associated with a measurement of the vehicle's instantaneous speed, given by

$$s_i = \frac{\|\vec{v_i}\|}{\Delta t} \tag{10}$$

where Δt is the time, in seconds, between two frames. This time is supposed to be constant, and is the inverse of the frame rate — e.g. for a frame rate of 30 frames per second, $\Delta t = 1/30$. The instantaneous vehicle speed s is estimated by averaging the values of s_i for a set of tracked features.

The assumption that all the motion vectors for a vehicle lie on the same plane is a simplification, used so that the actual 3D position of the tracked features does not have to be discovered. As the actual license plates are always above the road level, the computed speeds will be higher than the actual speeds. To mitigate the effects of these erroneous measurements we multiply each vehicle's measured speed by a constant factor S, which we set to 0.9 in our experiments. As shown in Section VII, the use of the S factor is simple but effective, as long as the tracked features are at approximately the same distance from the ground.

The final speed for a vehicle is obtained by averaging the instantaneous speed across multiple frames, while the vehicle is located at a certain image region. In our experiments, we considered a speed measurement region close to the ground truth loop detectors, to allow a direct comparison between the measured speeds and the ground truth speeds.

VII. Experiments

A proof-of-concept system was built for evaluating the proposed approach. Besides the cameras and physical infrastructure, we used a 2.2 GHz Intel Core i7 machine with 12 GB of RAM running Linux, with the algorithms implemented in C++. In the next sections we describe our dataset, and evaluate our system's performance regarding motion detection, license plate detection, and speed measurement.

A. Dataset

Our dataset, summarized in Table I, contains 20 videos captured by a low-cost 5-megapixel CMOS image sensor, with frame resolution of 1920×1080 pixels, at 30.15 frames per second. The videos are divided in 5 sets according to weather and recording conditions. Each video has an associated *ground truth file*, in a simple XML format, containing bounding boxes for the first license plate occurrence of each vehicle, as well as each vehicle's actual speed. The ground truth for the license plates was obtained by human inspection. The ground truth speeds were obtained from a high precision speed meter based on inductive loop detector, properly calibrated and approved by the Brazilian national metrology agency (Inmetro). Note that the videos contain some vehicles with no visible license plate, and that the ground truth speed meter sometimes fails to properly assign a speed to a vehicle. The "No. valid" column in Table I indicates the number of vehicles which have both a visible license plate and an assigned speed.

TABLE I
Dataset information: time (minutes); number of videos; number of vehicles with plates and speed information. The quality options are: [H] high-quality, [N] frames affected by natural or artificial noise, [L] frames affected by severe lighting conditions, [B] motion blur, and [R] rain.

Set	Time	No. videos	No. vehicles	No. plates	No. speed	No. valid	Notes
01	34	4	1,146	1,128	1,033	1,019	[H]
02	169	11	4,829	4,713	4,345	4,241	[L]
03	26	2	960	936	876	855	[N]
04	41	2	1,045	1,034	928	917	[N,R]
05	20	1	869	800	795	734	[L,B]
Tot.	291	20	8,849	8,611	7,977	7,766	

Figure 15 shows how the vehicle speeds are distributed (the speed limit in this particular roadway is 60 km/h).

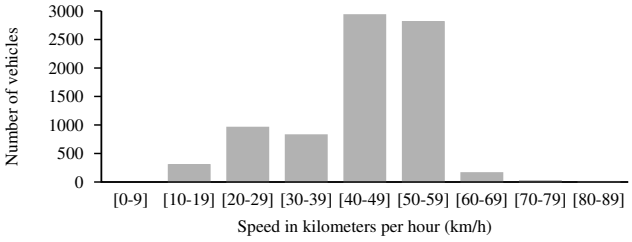


Fig. 15. Vehicles speed distribution.

In our manual ground truth annotation, we also include a flag for motorcycles and non-motorcycles (ordinary vehicles). We noted that motorcycles pose a challenge for the ground truth speed meter, which was able to measure the speed in only 43% of the cases (compared to 92% for ordinary vehicles). For that reason, motorcycles represent 4.5% of the total number of vehicles, but only 2.1% of the "No. valid" vehicles.

The whole dataset used in our experiments will be made available for research purposes, and can be itself considered one of the major contributions of our work.

B. Motion detection evaluation

To evaluate the results from the motion detector, we compare the obtained regions of interest (ROIs) with the license plates in the ground truth. Ideally, all the detected ROIs will contain a license plate, and all license plates will be fully contained within a detected ROI. Objectively, we compute precision and recall metrics as described by Wolf *et al.* [38], with the precision p being given by the proportion of ROIs

which contain at least one license plate, and the recall r being given by the proportion of license plates that were inside a ROI. Namely, we compute

$$p = \frac{\sum_{i=1}^{|D|} m(d_i, G)}{|D|} \qquad r = \frac{\sum_{i=1}^{|G|} m(g_i, D)}{|G|} \tag{11}$$

where $G = \{g_1, g_2, \ldots, g_{|G|}\}$ is the set of ground truth license plate regions, and $D = \{d_1, d_2, \ldots, d_{|D|}\}$ is the set of detected ROIs. Function m is defined by

$$m(a, S) = \max_{i=\{0,\ldots,|S|\}} m'(a, s_i) \tag{12}$$

where m' is a function that compares two rectangular regions a and b — more specifically, a ROI with a license plate region:

$$m'(a, b) = \begin{cases} 1 & \text{if } \dfrac{\text{area}(a \cap b)}{\min(\text{area}(a), \text{area}(b))} > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

with the λ threshold indicating how much of the license plate region must be contained within the ROI. We performed tests using two different values for λ: 1.0 (i.e. the entire license plate is contained within the ROI) and 0.5.

Table II shows the precision and recall, as well as the average processing time, obtained by our motion detector. The different columns show the results obtained with different amounts of subsampling — more sparse grids will reduce the processing time, but can also lead to incorrect results.

TABLE II
MOTION DETECTION PERFORMANCE: THE PRECISION p, RECALL r, AND AVERAGE TIME (IN MS, FOR EACH FRAME) FOR FIVE SUBSAMPLING CONFIGURATIONS AND TWO OVERLAPPING THRESHOLDS.

Subsampling	1×1		2×2		4×4		8×8		32×32	
	p	r	p	r	p	r	p	r	p	r
$\lambda = 1.0$	0.86	0.99	0.86	0.99	0.86	0.99	0.81	0.99	0.58	0.99
$\lambda = 0.5$	0.87	0.99	0.88	0.99	0.87	1.00	0.84	1.00	0.65	1.00
Avg. time (ms)	21.50		7.97		3.54		1.36		0.20	

C. License plate detection evaluation

To evaluate the performance of the license plate detector, we compare the detected license plates with those in the ground truth. The comparison is based on the same precision and recall metrics used for evaluating the motion detector (see Section VII-B), with some differences. First, set D refers to the set of detected license plates. Second, function m' is defined as in the PASCAL Visual Object Detection Challenge [39]:

$$m'(a, b) = \begin{cases} 1 & \text{if } \dfrac{\text{area}(a \cap b)}{\text{area}(a \cup b)} > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

For ranking purposes, we also consider the *F-measure*, which is the harmonic mean of precision and recall: $F = 2 \cdot p \cdot r/(p + r)$.

We compared our license plate detector with three text and license plate detectors described in the literature (see Section II-B): SnooperText [27], the Zheng *et al.* [21] algorithm, and the Stroke Width Transform (SWT) [22]. The parameters for these detectors were obtained by running the system on 25% of the videos from the dataset, and selecting the parameter combinations that produced the highest F-measures. The results obtained for the entire data set are shown in Table III, divided in 5 subsets according to weather and recording conditions. Our detector significantly outperformed the other approaches in these tests. The average time to process each region of interest was 58 ms for SnooperText; 918 ms for Zheng *et al.*; 402 ms for SWT; and 195 ms for our detector.

TABLE III
LICENSE PLATE DETECTION PERFORMANCE EVALUATION, BASED ON PRECISION (p), RECALL (r), AND THE F-MEASURE. THE BOLDFACE VALUES ARE THE MAXIMA OBTAINED FOR EACH CASE.

	PROPOSED			SNOOPERTEXT			ZHENG *et al.*			SWT		
Set	p	r	F	p	r	F	p	r	F	p	r	F
01	**0.96**	**0.94**	**0.95**	0.81	0.88	0.84	0.92	0.88	0.90	0.76	0.61	0.68
02	**0.92**	**0.84**	**0.88**	0.86	0.81	0.83	0.45	0.29	0.35	0.28	0.23	0.25
03	**0.94**	**0.94**	**0.94**	0.56	0.79	0.66	0.87	0.90	0.88	0.66	0.62	0.64
04	**0.94**	**0.92**	**0.93**	0.44	0.71	0.54	0.91	0.88	0.89	0.79	0.58	0.67
05	**0.88**	**0.82**	**0.85**	0.76	0.72	0.74	0.48	0.48	0.48	0.15	0.15	0.15
Tot.	**0.93**	**0.87**	**0.90**	0.73	0.80	0.76	0.65	0.52	0.58	0.44	0.37	0.40

Examples of license plates detected by the proposed method are shown in Fig. 16. Our detector worked as expected even in some situations with severe image noise or motion blur. Detection errors occurred mainly in the hypothesis generation phase, with true license plate regions being eliminated by some filtering criteria when they became connected with a background region. Samples of license plates not detected by our system are shown in Fig. 17.

D. Vehicle speed measurement evaluation

Speed measurement performance was evaluated by comparing the speeds measured by our system with the ground truth speeds obtained by the inductive loop detectors. According to the standards adopted in the USA, an acceptable measurement must be within the [-3 km/h, $+2$ km/h] error interval.

The first row in Table IV shows the results obtained by our system. Percentages are given regarding the valid vehicles — those with both a license plate and an associated speed in the ground truth — and are divided in 3 classes, depending on whether the measured speed was below, inside, or above the acceptable error interval. Figure 18 shows the distribution of the measurement errors, with 96% of the measurements being inside the acceptable limits. The maximum nominal error values for the whole dataset were -4.68 km/h and $+6.00$ km/h, with an average of -0.5 km/h a standard deviation of 1.36 km/h. We observed that the assumption that all the license plates have nearly the same distance from the ground is the main cause of speed measurement errors: when the license plates are very high above the ground

Fig. 16. Examples of license plates detected by our system, for representative samples of each set.

Fig. 17. Examples of license plates not detected by our system.

(e.g. in buses or trucks) the measured speed can be higher than the actual speed, with the opposite occurring when the license plates are unusually low. A total of 99.2% of the vehicles were successfully tracked until they reached the speed measurement region. On average, our tracking module spent 49.8 milliseconds per frame. Examples of measured speeds are shown in Fig. 19.

TABLE IV
SPEED MEASUREMENT RESULTS OBTAINED BY OUR SYSTEM AND OTHER APPROACHES: "LOWER", "IDEAL", AND "HIGHER" REPRESENT, RESPECTIVELY, SPEED ERRORS BELOW, ABOVE AND WITHIN THE ACCEPTABLE LIMITS, CONSIDERING THE USA STANDARD [−3/+2 KM/H].

	Lower	Ideal	Higher
PROPOSED SYSTEM	1.1%	96.0%	2.8%
IDEAL DETECTOR	0.9%	96.1%	3.0%
FREE FEATURE SELECTION	11.3%	73.4%	15.3%
PARTICLE FILTER	20.3%	22.1%	57.6%

In order to verify if distinctive features from a license plate region are a good choice for measuring a vehicle's speed, we performed tests using a version of our system which takes features from the whole vehicle region ("free feature selection"). The results are shown in Table IV. It can be seen

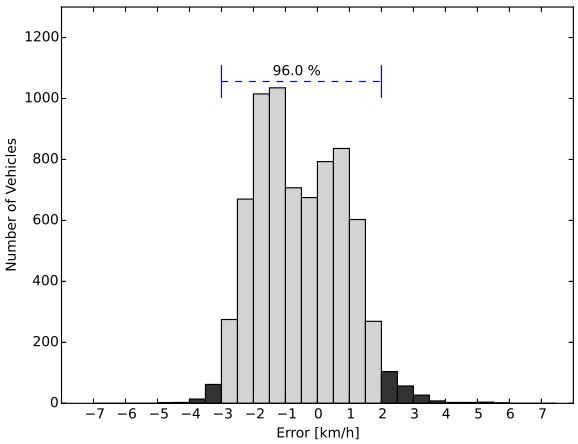


Fig. 18. Speed measurement error distribution.

that the percentage of vehicles whose measured speed is inside the allowed error interval decreased significantly. That happens because in this case the features have a larger variance in their heights from the ground — thus, motion vectors computed from the same vehicle have very different lengths. Moreover, these features are less distinctive, leading to tracking errors.

We also tested our system with an "ideal license plate detector", taking as references the manually annotated license plates from the ground truth, instead of the detected license plates. That includes some license plate regions that are hard to identify even by a human observer. As shown in the third row of Table IV, the performance in this case was not much different from the performance obtained by our complete system, indicating that the tracking can be done even with poor license plate regions, hard to be identified even by an human observer.

We also compared our system with a blob-based tracker. In this experiment, we used a particle filter algorithm [40] to track the regions of interest found by our motion detection module. We tested several parameter combinations for this approach, but we were unable to find a suitable configuration for our application. We believe that the main reason for this is that our camera is installed very close to the vehicles, in such a way that a probabilistic search cannot precisely define the position of the vehicle in all frames. Table IV shows the best results we could obtain using the blob tracking approach.

VIII. CONCLUSIONS

This paper addressed the problem of measuring vehicle speeds based on videos captured in an urban setting. We proposed a system based on the selection and tracking of distinctive features located within each vehicle's license plate region. The system was tested on almost five hours of videos with full-HD quality, with more than 8,000 vehicles in three different road lanes, with associated ground truth speeds obtained by a high precision system based on inductive loop detectors, as well as manually labeled ground truth license

Fig. 19. Examples of vehicle speeds measured by our system and by a high precision meter based on inductive loops

plate regions. Our system uses a novel license plate detection method, based on a texture classifier specialized to capture the gradient distribution characteristics of character strokes that make the license plate letters. This module achieved a precision of 0.93 and a recall of 0.87, outperforming other well-known approaches. We have also shown that extracting distinctive features from the license plate region led to better results than taking features spread over the whole vehicle, as well as an approach which uses a particle filter for blob tracking. In our experiments, the measured speeds had an average error of -0.5 km/h, staying in over 96.0% of the cases inside the +2/-3 km/h error interval determined by the regulatory authorities in several countries.

As future work, we intend to verify if estimating the distance of the license plates from the ground can improve the results. We also aim to apply an OCR on the detected license plates in order to create a traffic speed control system with integrated surveillance tools, e.g. to compute the traffic flow, to identify stolen vehicles, etc. Another topic for future work is the implementation on a compact platform that allows local processing, including optimizations such as parallel processing on GPUs, thus reducing communication bandwidth requirements.

ACKNOWLEDGMENT

This work was partially supported by the Brazilian Agency CNPq - Grant 444789/2014-6.

REFERENCES

[1] T. V. Mathew, "Intrusive and non-intrusive technologies," Indian Institute of Technology Bombay, Tech. Rep., 2014.
[2] N. Buch, S. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Trans. on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920–939, Sept 2011.
[3] J. Shi and C. Tomasi, "Good features to track," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593–600.
[4] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.
[5] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.
[6] D. Luvizon, B. Nassu, and R. Minetto, "Vehicle speed estimation by license plate detection and tracking," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6563–6567.

[7] D. Dailey, F. Cathey, and S. Pumrin, "An algorithm to estimate mean traffic speed using uncalibrated cameras," *IEEE Transactions on Intelligent Transportation Systems (ITS)*, vol. 1, no. 2, pp. 98–107, 2000.

[8] V. Madasu and M. Hanmandlu, "Estimation of vehicle speed by motion tracking on image sequences," in *IEEE Intelligent Vehicles Symposium*, 2010, pp. 185–190.

[9] S. Dogan, M. S. Temiz, and S. Kulur, "Real time speed estimation of moving vehicles from side view images from an uncalibrated video camera," *Sensors*, vol. 10, no. 5, pp. 4805–4824, 2010.

[10] C. H. Xiao and N. H. C. Yung, "A novel algorithm for estimating vehicle speed from two consecutive images," *IEEE Workshop on Applications of Computer Vision (WACV)*, p. 12, 2007.

[11] H. Zhiwei, L. Yuanyuan, and Y. Xueyi, "Models of vehicle speeds measurement with a single camera," *Int. Conf. on Computational Intelligence and Security Workshops*, pp. 283–286, 2007.

[12] C. Maduro, K. Batista, P. Peixoto, and J. Batista, "Estimation of Vehicle Velocity and Traffic Intensity Using Rectified Images," *IEEE Int. Conf. on Image Processing (ICIP)*, pp. 777–780, 2008.

[13] H. Palaio, C. Maduro, K. Batista, and J. Batista, "Ground plane velocity estimation embedding rectification on a particle filter multi-target tracking," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 825–830.

[14] L. Grammatikopoulos, G. Karras, and E. Petsa, "Automatic Estimation of Vehicle Speed from Uncalibrated Video Sequences," *Modern Technologies, Education and Professional Practice in Geodesy and Related Fields*, pp. 332–338, 2005.

[15] T. Schoepflin and D. Dailey, "Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation," *IEEE Transactions on Intelligent Transportation Systems (ITS)*, vol. 4, no. 2, pp. 90–98, June 2003.

[16] G. Garibotto, P. Castello, E. Del Ninno, P. Pedrazzi, and G. Zan, "Speedvision: speed measurement by license plate reading and tracking," in *IEEE Trans. Intell. Transp. Syst.*, 2001, pp. 585–590.

[17] W. Czajewski and M. Iwanowski, "Vision-based vehicle speed measurement method," in *Int. Conf. on Computer Vision and Graphics*, ser. 10, 2010, pp. 308–315.

[18] M. Garg and S. Goel, "Real-time license plate recognition and speed estimation from video sequences," *ITSI Transactions on Electrical and Electronics Engineering*, vol. 1, no. 5, pp. 1–4, 2013.

[19] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Transactions on Intelligent Transportation Systems (ITS)*, vol. 9, no. 3, pp. 377–391, 2008.

[20] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 311–325, 2013.

[21] D. Zheng, Y. Zhao, and J. Wang, "An efficient method of license plate location," *Pattern Recognition Letters (PRL) - Elsevier*, vol. 26, no. 15, pp. 2431–2438, 2005.

[22] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *IEEE Int. Conf on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 886–893.

[23] B. Li, B. Tian, Y. Li, and D. Wen, "Component-based license plate detection using conditional random field model," *IEEE Trans. on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1690–1699, Dec 2013.

[24] A. Ashtari, M. Nordin, and M. Fathy, "An iranian license plate recognition system based on color features," *IEEE Transactions on Intelligent Transportation Systems (ITS)*, vol. 15, no. 4, pp. 1690–1705, Aug 2014.

[25] H. Li, D. Doermann, and O. Kia, "Automatic text detection and tracking in digital video," *IEEE Transactions on Image Processing (TIP)*, vol. 9, no. 1, pp. 147–156, 2000.

[26] W. Zhou, H. Li, Y. Lu, and Q. Tian, "Principal visual word discovery for automatic license plate detection," *IEEE Transactions on Image Processing (TIP)*, vol. 21, no. 9, pp. 4269–4279, 2012.

[27] R. Minetto, N. Thome, M. Cord, N. J. Leite, and J. Stolfi, "SnooperText: A text detection system for automatic indexing of urban scenes," *Computer Vision and Image Understanding (CVIU) - Elsevier*, vol. 122, pp. 92–104, 2014.

[28] R. Minetto, N. Thome, , M. Cord, J. Stolfi, and N. J. Leite, "T-HOG: An effective gradient-based descriptor for single line text regions," *Pattern Recognition (PR), Elsevier,*, vol. 46, no. 3, pp. 1078–1090, 2013.

[29] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 23, no. 3, pp. 257–267, Mar 2001.

[30] J. Ha, R. Haralick, and I. Phillips, "Document page decomposition by the bounding-box project," in *Int. Conf. on Document Analysis and Recognition (ICDAR)*, vol. 2, 1995, pp. 1119–1122 vol.2.

[31] T. Retornaz and B. Marcotegui, "Scene text localization based on the ultimate opening," in *Int. Symposium on Mathematical Morphology (ISMM)*, vol. 1, 2007, pp. 177–188.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[33] J.-Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[34] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 27, no. 10, pp. 1615–1630, 2005.

[35] G. Wang, Z. Hu, F. Wu, and H.-T. Tsui, "Single view metrology from scene constraints," *Image and Vision Computing (IVC) - Elsevier*, vol. 23, no. 9, pp. 831–840, 2005.

[36] H. Li, M. Feng, and X. Wang, "Inverse perspective mapping based urban road markings detection," in *IEEE International Conference on Cloud Computing and Intelligent Systems (CCIS)*, vol. 03, Oct 2012.

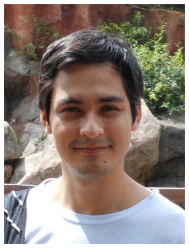
[37] D. G. R. Bradski and A. Kaehler, *Learning OpenCV, - 1st Edition*, 1st ed. O'Reilly Media, Inc., 2008.

[38] C. Wolf and J.-M. Jolion, "Object count/area graphs for the evaluation of object detection and segmentation algorithms," *Int. Journal on Document Analysis and Recognition (IJAR)*, vol. 8, no. 4, pp. 280–296, 2006.

[39] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) challenge," 2009.

[40] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *European Conference on Computer Vision (ECCV)*, 2002, pp. 661–675.

Diogo Carbonera Luvizon is a Ph.D. student at Université de Cergy-Pontoise since 2015. He received the M.Sc. degree from Federal University of Technology - Paraná (UTFPR) in 2015. He worked as a research engineer in an company of vehicle speed measurement systems from 2010 to 2014. His main research interest include vehicle detection and tracking, speed estimation and image descriptors.



Bogdan Tomoyuki Nassu is an assistant professor at Federal University of Technology - Paraná (UTFPR) - Brazil since 2012. He received the Ph.D. degree in advanced interdisciplinary studies from the University of Tokyo, Japan, in 2008. He was a researcher at the Railway Technical Research Institute, Japan, from 2008 to 2011, and a postdoctoral research fellow at the Federal University of Parana, Brazil. His main research interest is applying computer vision techniques to practical problems.

Rodrigo Minetto is an assistant professor at Federal University of Technology - Paraná (UTFPR) - Brazil since 2012. He received the Ph.D. degree in computer science from Université Pierre et Marie Curie, France and University of Campinas, Brazil, in 2012. His main research interest include text detection, object tracking, image descriptors and additive manufacturing.