IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 19, 2017, accepted July 27, 2017, date of publication August 24, 2017, date of current version October 12, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2737418

Eyes on the Target: Super-Resolution and License-Plate Recognition in Low-Quality Surveillance Videos

HILÁRIO SEIBEL, JR.,[1,2], (Member, IEEE), SIOME GOLDENSTEIN[1], (Senior Member, IEEE), AND ANDERSON ROCHA[1], (Senior Member, IEEE)

[1]Institute of Computing, University of Campinas, Campinas 13083-852, Brazil
[2]Instituto Federal do Espírito Santo, Vitória 29173-087, Brazil

Corresponding author: Hilário Seibel (hsjunior@ifes.edu.br)

This work was supported in part by the National Council for Scientific and Technological Development under Grant 304472/2015-8, in part by the Coordination for the Improvement of Higher Education Personnel under Grant DeepEyes, in part by the São Paulo Research Foundation under DéjàVu Grant 2015/19222-9, and in part by Microsoft Research.

ABSTRACT Low-quality surveillance cameras throughout the cities could provide important cues to identify a suspect, for example, in a crime scene. However, the license-plate recognition is especially difficult under poor image resolutions. In this vein, super-resolution (SR) can be an inexpensive solution, via software, to overcome this limitation. Consecutive frames in a video may contain different information that could be integrated into a single image, richer in details. In this paper, we design and develop a novel, free and open-source framework underpinned by SR and automatic license-plate recognition (ALPR) techniques to identify license-plate characters in low-quality real-world traffic videos, captured by cameras not designed specifically for the ALPR task, aiding forensic analysts in understanding an event of interest. The framework handles the necessary conditions to identify a target license plate, using a novel methodology to locate, track, align, super-resolve, and recognize its alphanumerics. The user receives as outputs the rectified and super-resolved license-plate, richer in detail, and also the sequence of license-plates characters that have been automatically recognized in the super-resolved image. Additionally, we also design and develop a novel SR method that projects the license-plates separately onto the rectified grid, and then fill in the missing pixels using inpainting techniques. We compare the different algorithms in the framework (five for tracking, three for registration, seven for reconstruction, two for post-processing, and two for the recognition step), and present discussions on the pros and cons of each choice. Our experiments show that SR can indeed increase the number of correctly recognized characters posing the framework as an important step toward providing forensic experts and practitioners with a solution for the license-plate recognition problem under difficult acquisition conditions.

INDEX TERMS Super-resolution, license-plate, recognition, tracking, video surveillance.

I. INTRODUCTION

Automatic license-plate recognition (ALPR) uses optical character recognition (OCR) on images to extract and recognize the characters of a vehicle registration plate [1], [2]. It is usually aided by cameras designed specifically for such task, since the license-plate recognition may be especially difficult under poor images resolutions (usually when the car is too far away from the camera, under adverse atmospheric conditions, or due to a low-quality acquisition camera) [3]. However, there are a number of low-quality surveillance cameras scattered throughout our cities that could help to identify a suspect, for example, in a crime scene. Fig. 1 depicts a situation in which the license-plate characters may not be easily identified even in a high-resolution video. The OCR systems and the forensic specialists may fail to recognize the alphanumerics in such setups, and super-resolution can be an inexpensive path, via software, to overcome this limitation.

Multi-frame super-resolution, usually referred to only as super-resolution (SR), is a process of constructing a high-resolution (HR) image using a set of low-resolution (LR) images of the same scene. As a matter of fact, there exist some techniques in the literature that leverage side information

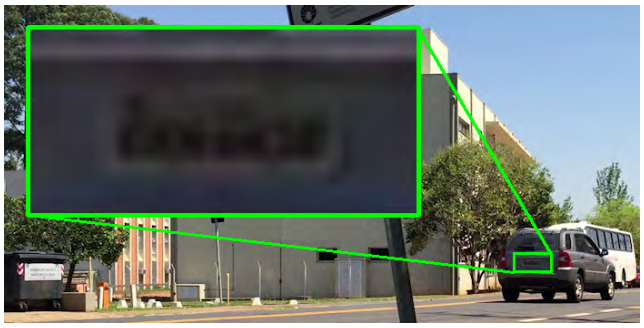


FIGURE 1. Surveillance cameras scattered throughout the cities could help to identify a suspect, for example, in a crime scene. But even in a high-resolution camera (e.g., 1920 × 1080 pixels in this frame) it might be difficult to visually recognize the license-plate characters.

from other sources — the internet, for instance, or from large collections of images — and can operate with a single image [4]–[7]. Although effective in different setups, these techniques are not the focus of this work. Fortunately, in our setup, a sequence of video frames recording a dynamic scene may contain different information about the object of interest. Generally, a moving object in the scene can be recursively seen at many positions along its moving path in the video. Due to these recurrences, many self-similar appearances between different positions can be found throughout a video sequence. Therefore, in this work, we choose to combine information from multiple video frames, instead of working with the recent Single-Image Super-Resolution (SISR) algorithms.

Consecutive frames in videos may differ not only by rigid or perspective transformations. Notwithstanding, we do not aim here at super-resolving features such as human faces, for example. Rather, we focus on enhancing the details in vehicle license plates that could help to identify a criminal suspect or activity in a crime scene. In such forensic setup, it is feasible to super-resolve only a region of interest (ROI) of a video, discarding less important parts.

The main contribution of this paper is a free and open-source end-to-end framework that super-resolves a sequence of frames containing license-plates in low-quality real-world traffic videos, captured by cameras not designed specifically for the ALPR task, aiding forensic analysts and practitioners in understanding a given event of interest. Additionally, we also design and develop a novel SR method that projects the license-plates separately onto the rectified grid, and then fill in the missing pixels using Inpating [8], [9] techniques. The framework has two main outputs:

- The first one is a rectified and super-resolved image, richer in details. The user can simply use such image for a better visualization of the alphanumerics, or even as an improved input for a SISR algorithm [4], [5].
- In addition, we apply ALPR to the output image, suggesting a sequence of license-plates characters for the user.

For a sneak peak showing the potential of the work we present herein, consider Fig. 2, in which we super-resolve

FIGURE 2. Output examples of the proposed framework. In (a), single frames with low-quality resolution. In (b), five consecutive frames are combined into a super-resolved image, richer in details.

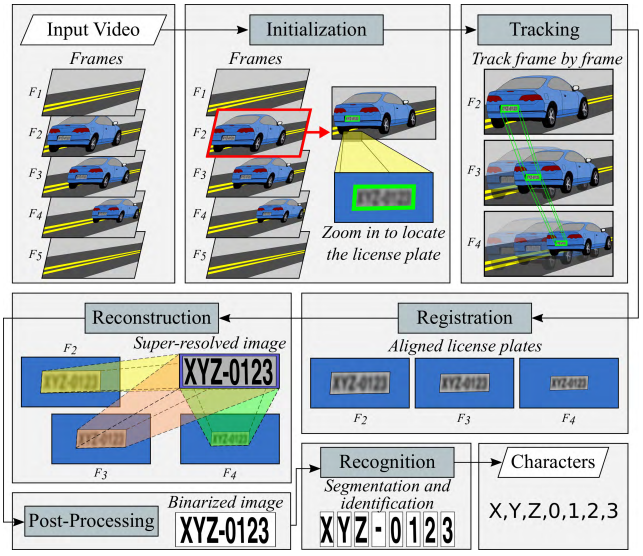


FIGURE 3. Our end-to-end framework pipeline. (1) *Initialization*: Choosing a starting frame and locating the license-plate; (2) *Tracking*: Finding the license plate over the consecutive frames; (3) *Registration*: Aligning the frames with respect to the plate positions; (4) *Reconstruction*: Combining the frames into a high-resolution grid; (5) *Post-processing*: Applying refining image processing operations to the reconstructed image, to improve the results in the final step; and (6) *Recognition* of the alphanumerics in the super-resolved license plate.

only five consecutive frames. The reconstructed image in Fig. 2a is more easily readable than the single frames in Fig 2b.

We designed the proposed framework involving six core steps to perform the license-plate recognition (see Fig. 3).

1) The forensic analyst sets up the *initial frame* wherein the suspect vehicle appears, and locate the license plate in such frame (there might be other moving cars in the scene, and the specialist identifies the target one).
2) Then, we track the license-plate region through the consecutive frames using a series of combined techniques

including optical flow with sparse and dense features as well as fast indexing for efficient purposes. In doing so, we align the frames with respect to the license plate.
3) After tracking, we refine the alignment with subpixel accuracy using optical flow once again (but more accurately, as the images are now pre-aligned).
4) We produce a super-resolved and rectified license plate within a fixed-sized grid according to country-specific specifications, relying on Geometric K-NN Super-Resolution (GSR) [10] and a novel Inpaiting-based Super-Resolution method (ISR) to combine the frames.
5) We apply post-processing operations to the image (e.g., Otsu's binarization [11]) to improve the recognition; and
6) Finally, we design an ALPR solution based on Tesseract [12] and OCRopus [13] to recognize the characters.

Our experiments compare the quality of the final results by the number of characters correctly recognized in the last step. The higher the number of correctly recognized characters in a license plate, the better the result. For validation, we consider a dataset of real-world traffic videos with moving vehicles and the correct alphanumerics in their license plates. The dataset will be publicly available upon acceptance of this paper.

II. MULTI-FRAME SUPER-RESOLUTION

A digital image stores and represents information of a real-world scene by a finite number of samples. Interpolation (or upsampling) [14] is used to increase the spatial resolution of a digital image, finding out new samples amongst those that are already known. Interpolation is an ill-posed problem since there are infinitely many HR images that may have the same LR samples. This ambiguity increases as the intended magnification becomes larger [15], [16].

Although interpolation has been extensively studied since ancient times [17], the quality of an image magnified from an aliased LR image is inherently limited. Single-image interpolation cannot recover the high-frequency components lost or degraded during the sampling process [18]. To achieve further improvements in this field, it is natural to seek multiple data sets in which additional data constraints from several observations of the same scene can be used. The information fusion of various observations for magnifying an image of the same scene is referred to as multi-frame super-resolution.

In a multi-frame super-resolution, we assume that a set of n LR images I_k ($\forall k \in [1, n]$) have been downsampled from the same HR image. While interpolation can be used to increase the resolution of an image, decimation (or downsampling) decreases the resolution. As opposed to the interpolation operation, decimation does entail loss of information. Super-resolution is then an inverse decimation problem. Each image I_k is a result of a convolution equation of the form $A_k * X = I_k$, where I_k and A_k (the filter applied to the HR image, including possible bluring, downsampling, rotation and other operators) are known vectors, and X is the vector to be determined (related to the entire HR image). The set of LR images, then, form a system of linear equations that might be used to determine the target I_{HR} completely. Nevertheless, SR is also an ill-posed problem [19] as there might be infinite HR images that satisfy the reconstruction constraint, the number of LR images is usually insufficient to solve the system, the registration of the input images is often ill-conditioned, and the blurring operators are usually unknown.

Multi-frame super-resolution has been studied since 1984, when Tsai and Huang [20] pioneered the field and introduced an algorithm in the *frequency domain* to super-resolve images using a set of similar, but globally translated images, of the same area. Those shifts between consecutive images are taken into account by the shifting property of the Fourier transformation. Most frequency domain methods have problems with real-world applications, since they accept only a global displacement between the images.

The majority of SR algorithms have since been developed in the *spatial domain* [21]. Such methods are based on interpolation over LR images. A single image interpolation does not handle the SR problem well, since it may not produce those high-frequency components that were lost during the image acquisition process. However, in multi-frame approaches, each LR observation might provide a small amount of additional information about the scene [22]. Such methods usually have a registration step, for aligning the LR images, and a reconstruction step, for producing the higher resolution image. Optionally, they can also include a deblurring step for enhancing the HR image produced in the second step.

Iterative Back Projection (IBP) algorithms [23] are among the first methods for spatial-based SR. In these cases, each HR image pixel is estimated iteratively as a sum of different projections of the same LR image area, determined by the image blurring and displacement. IBP is simple, but might not yield a unique solution due to the ill-posed nature of the SR problem. Zomet et al. [24] proposed the Robust Super Resolution, other version of the IBP algorithm using the median rather than the mean to calculate each new pixel. Papoulis [25] and Gerchberg [26], independently, demonstrated the method of iterative signal extrapolation. The classical Papoulis-Gerchberg (PG) method may not deliver good results in presence of blur and noise in the LR image. Vandewalle et al. [27] extended upon the traditional PG method to obtain SR images from multiple LR registered images.

Another group of iterative methods are based on the concept of Projection onto Convex Sets (POCS) [28]. In such methods, it is assumed that each LR image imposes an a priori knowledge on the final solution. These algorithms define an implicit cost function for solving the SR problem, do not give a unique solution and suffers from high computational costs.

In addition, the Maximum a Posteriori (MAP) methods [29] also add some a-priori knowledge about the desired HR image. As the super-resolution is often an ill-conditioned problem, the a priori term is used to prefer a specific solution

when the solutions are not unique. A critical issue of the MAP-based algorithms is the choice of the prior model for the desired solution. Such methods use regularization to solve the system of linear equations of the SR problem, and different approaches can be used to find the best possible value for the regularization parameter [21].

Another trend in the spatial domain that is currently exploited in the literature is the Direct Methods [30]. Such methods simply align and scale the LR images to an HR grid, and then choose a filter to combine the LR pixels. Different filters can be used, such as mean and median filters [31], Adaboost classifier [32], SVD-based filters [33], and adaptive normalized averaging [34].

Different from multi-frame super-resolution, image *hallucination* generates an HR image from a single LR source, with the help of a database of sample images that is used as a training set. Hallucination (also known as *example-based*, *learning-based* or *single-image* super-resolution), has become a hot research topic since it was first proposed by Freeman et al. in [35]. These approaches effectively ''hallucinate'' missing details based on similarities between the LR image and the examples in the training set [16]. Romano et al. [36] recently proposed a single-image algorithm that uses machine learning and train on pairs of images (one low quality, one high) to find filters that, when applied selectively to each pixel of the LR image, will recreate details that are of comparable quality to the original. In [5], the authors present a learning-based SR method that uses deep convolutional neural networks to learn an end-to-end mapping between low and high-resolution images. Although promising, those single-image methods do not take advantage of the multiple information from the pool of frames that our surveillance videos might comprise.

III. SUPER-RESOLUTION OF LICENSE PLATES

Caner et al. [37] seem to have pioneered the alliance of super-resolution and automatic license-plate recognition. They super-resolved a region of interest of surveillance videos recorded by multiple cameras based on POCS (see Sec. II). Although promising, the solution needed more than one camera to work. Chang et al. [3] claimed that most of the techniques until 2004 worked under very restricted conditions, such as fixed illumination, limited vehicle speed, designated routes, and stationary backgrounds. In their work, they have favored classification accuracy over efficiency whenever a choice had to be made between them. However, in their experiments, they only considered images with readable characters, in which a human could easily identify the alphanumerics on the plates.

In 2007, Suresh et al. [38] performed SR of moving vehicles in real-world traffic videos by fusing the information derived from multiple, subpixel shifted, and noisy LR observations. The image to be super-resolved was modeled as a Markov random field and was estimated from the observations by a graduated non-convex optimization procedure. However, they did not consider rotations in the license plates between the frames, and the results were only qualitatively compared.

Yuan et al. [39] presented a MAP-based algorithm to super-resolve license plates and relied upon some license-plate properties as the a priori knowledge for the regularization. For example, they claim that the the license-plate background color and the colors for the license-plate characters are usually with strong contrast. Hence, when the image is converted into a grayscale image, there remain only two kinds of intensities: the dark one and the light one, and they might be easily distinguished by thresholding. However, from our experience, this claim only holds for ideal or semi-ideal illumination conditions. Furthermore, the authors presented the reconstruction of only one license plate in their experiments, and no validation metric was used to compare the results. They only compared the running time of reconstructing two HR images.

Camargo et al. [40] investigated the SR mosaicking of aircraft surveillance videos. They used Scale-Invariant Feature Transform (SIFT) [41] to find feature points between frames and RANdom SAmple Consensus (RANSAC) [42] to estimate an homography between consecutive frames. The algorithm was tested in the infrared and visible spectra, using real and synthetic data, but uses a small number of frames from the video, and the visual results are still relatively blurred.

Kim and Ko [43] proposed a resolution enhancement method for regions of interest in surveillance videos using Bernstein interpolation [44]. They super-resolved images using stochastic data regularization in real-world surveillance videos focusing on the license plates as ROIs. Yet the reconstructed images were only visually compared to other methods, and the results were very similar to classic algorithms. Taking a different path, Yoshida et al. [45] proposed an SR using free-form deformations for low-quality surveillance videos focusing on face ROIs, including non-rigid deformations caused by changes of face poses and expressions.

Zarei et al. (2013) [46] developed a super-resolution of license-plate images by applying an iterative SR method for license-plate recognition that fused the information from a set of shifted LR images. The reconstruction problem was formulated as a system of linear equations that was solved by using the Simultaneous Algebraic Reconstruction Technique (SIRT) [47]. The input frames in the dataset were not extracted from real-world videos. They also considered only shifts between two frames (not rotations).

Employing a learning-based method, Lina and Ying [48] proposed a license-plate super-resolution algorithm based on manifold learning. Although promising, the algorithm was not validated using real-world low-resolution images by surveillance cameras as input. Instead, they used only one high-resolution image to generate a set of downscaled images, and such lower resolution images have been used as input for the reconstruction step in their experiment.

According to a recent work of Rajput et al. [49], few researchers have addressed scenarios such as reading plates

of fast-moving vehicles. They also claimed that the existing ALPR approaches assume that the text lies in a plane whose angles are normal to the sensor's optical axis, which is not the case when license plates are skewed. Their work is not related to super-resolution, but they developed a method to detect license-plate orientation on tilted plates, and rotate it to a horizontal perspective. Previously, Tang et al. [50] and Qing et al. [51] also addressed the problem of rectification of license plates to correct their inclined distortion.

IV. PROPOSED END-TO-END FRAMEWORK FOR SUPER-RESOLUTION AND AUTOMATIC LICENSE-PLATE RECOGNITION

In this work, we investigate two methods for super-resolving a sequence of LR images (see Sec. IV-D). The first one has been introduced in our previous work [10] (conference paper). It has been only validated with general static scenes gathered by mobile devices. To perform the registration step in such preliminary version, we relied upon finding representative points in two consecutive images using Scale-Invariant Feature Transform (SIFT) [41], Speeded Up Robust Features (SURF) [52], and Oriented FAST and Rotated BRIEF (ORB) [53]. The found keypoints were then matched to allow the estimation of the best possible transformation matrix between the frames.

This methodology for the registration step has worked properly for the setup we presented in [10], but it is not appropriate for fast-moving vehicles captured by static cameras as we have in the present work. Fig. 4 depicts and example of two frames extracted from a real-world traffic video. The red points in the figure are keypoints detected by SIFT, and each colored line connects a keypoint in the first image to its correspondent keypoint in the second image. Note that the pairs of keypoints found by SIFT are not appropriate to calculate a transformation between the license plates in the two consecutive frames. Most of the keypoints belong to the environment around the car and do not contribute to map one license plate onto the other. Therefore, we need another methodology to align the license plates when dealing with videos of fast-moving vehicles.

FIGURE 4. Matches between pairs of keypoints found by SIFT.

In addition to the need of dealing with fast-moving vehicles, we still have two key problems to address in this new setup:

- First, we should not expect a forensic analyst to manually crop the vehicles in each video frame, every time she needs to identify the characters in a suspect license plate.
- Moreover, even if we automatically find and crop the target car in each frame, we still may not have an appropriate homography among the images. An homography assumes that all the points belong to a planar surface. Nonetheless we cannot claim that all representative points found in a region of interest will always belong to a plane. As a matter of fact, most of the aliasing issues that we can visually identify in super-resolved images occur due to the misalignment between frames, whose keypoints do not belong to a planar surface. Even if we crop only the region of the license plate (very similar to a plane), we may not find enough keypoints to calculate the transformation between the images and this should be taken into account by any proposed approach.

Besides such issues, there are a number of image-processing methods to improve the quality of the recognition step, including *rotation / deskewing / rectification* (making the lines of the text to be perfectly aligned with respect to the borders of the image), *binarization* (converting the image to black and white) and *noise removal*. Fig. 5 depicts an example of a rectified and binarized image, to improve the results in the recognition step.

FIGURE 5. Example of a rectified and binarized license-plate image.

The framework that we propose in this article is designed to cope with the aforementioned problems regarding the super-resolution of license plates and the automatic recognition of their alphanumerics. We investigate a novel methodology to locate, track, align, super-resolve, and recognize the alphanumerics of a license plate in low-quality surveillance videos, as Fig. 3 depicts. The framework performs the super-resolution and recognition of the license-plate characters in six steps, as we discussed in Sec. I: *Initialization*, *Tracking*, *Registration*, *Reconstruction*, *Post-processing*, and *Recognition*. We describe each step in Secs. IV-A through IV-F.

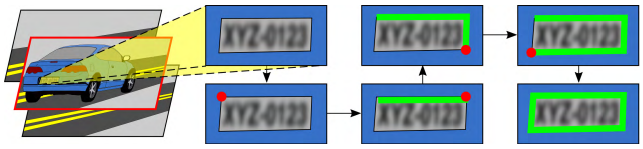


FIGURE 6. The initialization step. The user (e.g., a forensic analyst) selects the initial frame, and the system opens a window with a zoomed in version of the plate. The user selects four points and creates a bounding box around the characters to be identified. This step is done only once (for the initial frame).

A. INITIALIZATION

The framework starts with a graphical interface playing the traffic video (c.f., Fig. 6). The user can interact with the system by alternating between *playing* and *pause* mode, forwarding the video frame by frame or by selecting the initial frame wherein the license plate is fully shown.

To select the initial frame, the user clicks in the middle of the suspect license plate, and then the system opens a window with a zoomed in version of the region around the license plate upon which the user can click to select the four corners of the license plate, creating a bounding box around the characters to be identified. As the user selects the points, the interface automatically shows the lines of the bounding box. After selecting the points, if the region was not correctly created, the user can click again in the middle of the license plate and reselect the four points.

The interaction with the forensic analyst is required **only** in this first step. This is reasonable, since she needs to identify which of the moving vehicles in the video is the suspect one. The execution of all the additional steps are transparent to the user, who sees only the super-resolved image and its recognized characters at the end of the process. If a completely automatic solution is intended, a license-plate detector can be used in this first step. The system is designed in such a way that integrating a license-plate detector would be straightforward. We do not include such feature in this paper because our focus is to aid the forensic analyst with specific unresolved cases (in which other simpler solutions have been unhelpful, and the interaction with the user is desirable). The outputs of this step are: (1) the identifier of the initial frame, and (2) the points of the ROI around the plate.

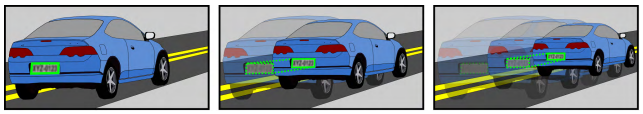


FIGURE 7. Tracking step: the objective is to find re-occurrences of the license plate over the consecutive frames.

B. TRACKING

Fig. 7 illustrates the *tracking* step. The inputs of this step are: (1) the initial frame identifier (defined in the previous step), (2) the ROI around the license plate (also calculated in the initialization step), and (3) the original video frames.

In this work, we examine five different solutions to track the license plates, as detailed in Secs. IV-B.1 through IV-B.3.

1) PYRAMIDAL LUCAS-KANADE OPTICAL FLOW (PyrLK)

The first method is based on the Lucas-Kanade optical flow [54]. Given a set of n input frames $F_1, F_2, \ldots F_n$ (from the initial frame until the end of the video), we first find good features to track in the license plate of F_1 using Shi-Tomasi corner detector [55]. Then, we use the Lucas-Kanade optical flow to track the points from F_1 to F_2, creating an homography that allows us to find the position of the license-plate in F_2. Finally, we iteratively use optical flow to track the points from F_k to F_{k+1} for each $k \in [2, n)$.

The Lucas-Kanade method can deal with small pixel displacements between consecutive frames. As we do not want to constrain our framework just to videos with slow-moving vehicles, we use a pyramidal and iterative implementation of the Lucas-Kanade optical flow (PyrLK) [56]. When we go up in the pyramid, the images are downscaled, the small motions are removed, and the large motions become small motions.

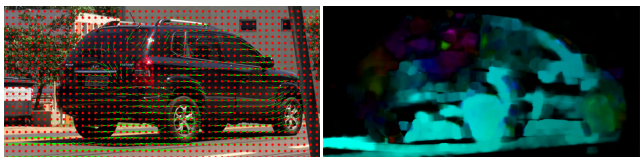
The Lucas-Kanade method assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighborhood. Fig. 8 shows a piece of an original frame and one example of the optical flow calculated by PyrLK (red dots are Shi-Tomasi points, and green lines represent the motion of the points throughout the frames).

FIGURE 8. PyrLK feature tracker to estimate the optical flow.

It is worth mentioning that the Shi-Tomasi corner detector might not find enough points to create an homography between the images in a very small and low-resolution license plate. To handle this problem, the feature points are found inside a region slightly larger than the license plate. As we discussed in Sec. IV, we need to find points inside a planar surface in order to create an homography between images. This expanded region around the license plate might still be similar to a planar surface. Even if it is not entirely planar, we do not need a precise alignment in the tracking step, but only an estimation of the license-plate position frame-by-frame. The subpixel accuracy (with the appropriate refinement) will be further obtained in the registration step.

2) PYRAMIDAL FARNEBACK'S DENSE OPTICAL FLOW (PyrDense)

In the second method, instead of computing the optical flow only for the Shi-Tomasi corner points, we use the Gunner Farneback's algorithm [57], also with pyramids, to compute the grid-based optical flow for the whole frames (PyrDense). We do not create the homography matrix here, since the algorithm calculates the motion of each pixel in the image. Therefore, we first track the known position of the license plate in F_1 to F_2, then we iteratively track the points from F_k to each F_{k+1}. Fig. 9a illustrates an example of the grid (red dots) and the flow motion (green lines). Fig. 9b uses HSV to visualize the same flow as in Fig. 9a (*hue* shows the flow direction, and *value* shows the flow magnitude).



(a) (b)

FIGURE 9. Dense optical flow as a result of the Farneback's algorithm.

In both PyrLK (Sec. IV-B.1) and PyrDense solutions, the tracked points in a frame F_k are used as previous points in the frame F_{k+1}. For a robust tracking, we also run a backward-check [58] of the optical-flow points to select only good ones, and we verify if the dimensions of the license plate in F_{k+1} are proportional to its dimensions in F_k. Therefore, the last tracked frame may not be the last video frame if the vehicle disappears before the end of the video.

3) SIFT, SURF AND ORB DETECTORS

In the third method for tracking, we use SIFT to find keypoints in the initial frame. This is possible as the user previously annotated the region describing the license plate of interest in the first step. As in the first tracking solution (PyrLK), we expand the bounding box to include a region slightly larger than the license plate in F_1, but now we match them with the SIFT points found in the entire F_2 image (see Fig. 10). The points are matched using k-NN and Fast Library for Approximate Nearest Neighbors (Flann) [59]. The best matches are used to estimate an homography matrix mapping F_1 onto F_2, and then we iteratively track the points in the consecutive frames (mapping each F_k onto F_{k+1}).

FIGURE 10. Tracking using SIFT. The white polygon is the region around the license plate in F_k. Yellow dots are the SIFT keypoints inside the small region in F_k and in the entire image F_{k+1}. Green lines are matches between the frames. The red line is an incorrect match that will not be used.

In addition to the SIFT-based matching solution for tracking, we also exploited the feature detectors SURF and ORB, instead of SIFT. Their rationale and operational conditions are the same as the ones described above for SIFT.

After finding the re-occurrences of license plates, we also align them with respect to the initial frame's license plate. The outputs of this step are: (1) the set of aligned frames in which the license plate was successfully tracked, and (2) the four points of the license plate's bounding box in each frame.

C. REGISTRATION

To track the license-plate in the previous step, we search its re-occurrences over consecutive frames. Since the framework does not limit the vehicle speeds and routes, we use pyramids and a large enough search window for the optical-flow-based solutions. Larger values increase the algorithm robustness to fast motion detection, but yield less accuracy. Therefore, since the license-plates have already been tracked and the frames have previously been aligned in the tracking step, we now refine this alignment with subpixel accuracy. From this point forward, instead of working on the video, the inputs of the *registration* step are: (1) the frames in which the license plate was successfully found and tracked, and (2) the license-plate bounding box in each frame. There are three available possibilities for the realignment in our framework:

a) Using Lucas-Kanade Optical Flow, as in the first tracking solution (see Sec. IV-B.1), without pyramids, since the frames were pre-aligned with respect to the license plate in F_1, and we need to find a subpixel motion of the license plate from F_k to F_{k+1}.
b) Using Farneback's Dense Optical Flow (Dense), also without pyramid decomposition.
c) In the last possibility, we consider that the tracking had performed a good alignment of the frames, and we do not realign them. We refer to this possibility as "None", since we do not use any further refinement method.

Fig. 11 illustrates this step. In both LK and Dense methods, we start with a small window size, and increase it until the registration is successfully performed through a given number of frames. Larger window sizes increase the algorithm robustness to image noise and give more chances for fast motion detection but, at the same time, lead to a more blurred motion field. The outputs are: (1) the set of realigned frames and (2) the license plate's bounding box in each frame.

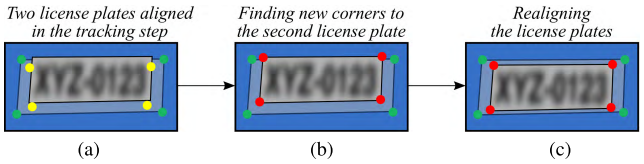


FIGURE 11. The registration step. Green dots in (a) are the corners in the initial frame, and yellow dots are the corners that were tracked in the subsequent frame. The red dots in (b) represent the new bounding box found in the registration step. In (c), we use the new points to realign the second license plate with respect to the first one.

D. RECONSTRUCTION

After aligning the input frames in the previous step, we turn our attention to the multi-frame super-resolution. Different from traditional SR algorithms, we do not create here a high-resolution image of $(s \times w, s \times h)$ pixels. Instead, we define a fixed spatial resolution to the HR grid, which is proportional to country-specific specifications for vehicle registration plates (e.g., 400×130 mm in Mercosur member countries and 12×6 inches in U.S.A. and Canada). Therefore, the super-resolved image is similar to a rectified license plate (see Fig. 5), and contains only the region inside the license-plate ROI. We fixed the size of the HR grid because in addition to generating a good visual HR image, we focus on creating HR images amenable to a better character-recognition task in the super-resolved license plate, and this rectification may improve the results in the recognition step. To combine the license-plate images, we use two super-resolution methods:

1) THE GEOMETRIC k-NEAREST NEIGHBORS MULTI-FRAME SuPER-RESOLUTION (*GSR*)

We first introduced *GSR* in [10]. In this previous work, n LR images I_k ($k \in [1, n]$) of size (w, h) pixels generated

a super-resolved image of size $(s \times w, s \times h)$ pixels for a scale factor s. The algorithm in [10] was validated using only planar and static scenes. Following the categorization discussed in Sec. II, GSR falls within the category of direct methods, which are known to be simple and fast [21].

The core idea of *GSR* is illustrated in Fig. 12. Blue squares and red dots in Fig. 12a represent pixels from two input frames F_k and F_{k+1}, aligned in the registration step. In Fig. 12b, we see projections of both F_k and F_{k+1} on a HR grid (black triangles are the pixels of the HR grid, whose values we need to calculate). In Fig. 12c, we choose each pixel in the grid to be the value of its nearest neighbor among all pixels in the LR images. Finally, the ultimate SR image is composed by pixels from both F_k and F_{k+1} in Fig. 12d.

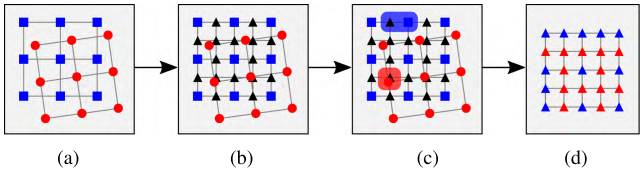


FIGURE 12. The GSR algorithm for super-resolving a sequence of frames. (a) Blue squares represent pixels of the frame F_k, and red dots are the pixels of F_{k+1} that were aligned with respect to F_k. (b) The two frames are projected onto an HR grid (black triangles are the pixels of the grid). (c) For each grid pixel, we find its geometric nearest neighbor among all pixels in the input frames. (d) The resulting HR image, comprising pixels from F_k and F_{k+1}.

As there are different possible policies to choose from and combine nearest neighbors geometrically, we exploit five variations of GSR in our framework. Such variations differ from each other in the choice of the best candidates for each pixel in the HR grid (Fig. 13). Let p be a pixel in the HR grid (for example, the black triangle in Fig. 13a) and q_k (red dots in Fig. 13a) be the nearest pixel from p in each frame F_k:

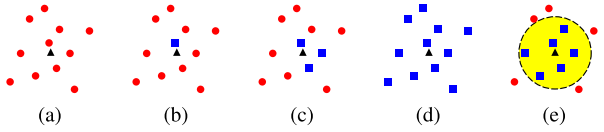


FIGURE 13. Variations of GSR. Let p be a pixel in the HR grid that we need to calculate, and q_k be the nearest pixel from p in each frame F_k (a). In GSR_1 (b), the HR pixel is the value of its nearest q_k; (c) Both GSR_2 and GSR_3 find 3-NN among all available q_k; (d) all available q_k are used to calculate the value of the HR pixel in GSR_4; and (e) all q_k inside a circular region of a given radius are combined in GSR_5.

- In the first algorithm variation (GSR_1), the value of p is exactly its closest neighbor among all possible values of q_k. In Fig. 13b, such closest neighbor is highlighted with a blue square. Thus, we use here a k-NN with $k = 1$ to find q_k in all LR images, and then another 1-NN to find the q_k that is closest to p.
- In the second variation (GSR_2), we use 3-NN to find the three best values (such as the three blue squares in Fig. 13c) among all q_k. The resulting pixel p is the average among the three closest neighbors.
- For GSR_3, in a similar way, we find the three best values of q_k, but we combine them as a weighted average. We use static weights (60% for the nearest one and 20% for each other), but we could also set them to be an inverse proportion to their distances.
- In GSR_4, p is the average among all q_k (the nearest pixel from p in each F_k). If we have 30 input frames, for example, q_k is the nearest pixel from p in each of the 30 frames and p is the average among all of them.
- Finally, for GSR_5, we set a maximum radius distance with respect to p (Fig. 13e), which is calculated as the weighted average among all q_k within this circular region (the weights are inversely proportional to the distances between p and each q_k).

It is important to mention that we do not apply rectification or any other transformation to the LR input frames, because it would cause loss of information. Instead, we use the matrices calculated in the registration step (that map each F_k onto F_{k+1}), to know exactly the xy position where each q_k is with respect to the grid. Each input image is individually processed, hence there will not be more than one LR image in memory at the same time. For an HR image of $w \times h$ pixels, the memory consumption of all five variations is $\mathcal{O}(w \times h)$. Basically, the algorithms need to store, in memory, the following information during each step: (1) the $w \times h$ pixels p in the HR grid and the distances to their nearest pixels q_k (GSR_4 and GSR_5 do not store such distances); (2) the $\frac{w}{r} \times \frac{h}{r}$ pixels of the current LR image, for a resizing factor of r; and (3) the positions of the $\frac{w}{r} \times \frac{h}{r}$ LR pixels with respect to the grid.

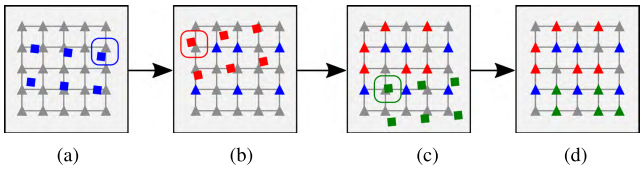


FIGURE 14. In (a), the first frame is projected onto the HR grid. The blue region highlights a pixel $F_k(x, y)$ and its closest pixel p in the grid. The intensity value of $F_k(x, y)$ contributes only to such closest grid neighbor, so p receives the intensity value of $F_k(x, y)$. In (b), the grid contains information from the first frame (the blue triangles), and another frame is projected onto the grid. Similarly, in (c) a third frame is projected onto the grid, which already contains information from the two previous LR images. Finally, the gray triangles in (d) are the missing pixels of the grid, whose values we want to calculate using Inpainting techniques.

2) THE INPAINTING-BASED SUPER-RESOLUTION (*ISR*)

In this novel method, we start projecting each frame F_k, separately, onto the HR grid. The beginning of the algorithm is illustrated in Fig. 14. In Fig. 14a, the first frame is projected onto the grid. For each pixel $F_k(x, y)$, we calculate the positions (x', y') of its pixels with respect to the grid. The intensity value of each pixel $F_k(x, y)$ contributes only to the grid pixel that is closest to (x', y'). In Fig. 14b, we project a second frame onto the grid, and the blue triangles are grid pixels that have already been calculated using the information from the first frame. Similarly, in Fig. 14c, a third frame is projected onto the grid, which already contains information from the two previous LR images. Finally, some pixels in the grid might

not be filled with the intensity values of any frame. We use Inpainting [8], [9] techniques to calculate such unknown grid pixels (the gray triangles in Fig. 14d).

Now, we must fill in the unknown pixels, onto which no LR pixel has been projected. To calculate such information, we use a technique for restoration of degraded photos called ''Image Inpainting'' [8], [9]. The basic idea is simple: it fills in part of an image using information from surrounding areas, and several algorithms have been designed for this purpose. Inpainting has already been used to fill in missing pixels of super-resolved images, as in [60]. However, the technique is commonly applied to example-based super-resolution, which uses only one LR frame as input and does not take advantage of situations when we might have multiple observations of the same scene. In [61], the authors use Inpainting to fill the missing pixels in shifted LR images (i.e., with only translations between to input images), and then they use Tikhonov [62] regularization to solve the optimization problem.

As in *GSR*, we do not apply transformations onto the frames, because it would cause loss of information. In addition, each frame can be individually processed, so there must not be more than one LR image in the memory at the same time.

We implemented two variations of the method, each one using a different Inpainting technique to fill in the missing grid pixels. The first one draws ideas from classical fluid dynamics to propagate isophotes (lines of equal gray value) continuously from the exterior into the region to be inpainted, based on the work of Bertalmío et al. [9]. Their method is directly based on the Navier-Stokes equations for fluid dynamics [63], which has the immediate advantage of well-developed theoretical and numerical results. We refer to this implementation as ISR_1.

The second variation uses the work of Telea [8], an Inpainting algorithm based on propagating an image smoothness estimator along the image gradient, similar to the work in [64]. Such algorithm is supposed to be simple to implement and faster than other inpainting methods, and to produce similar results as compared to the other methods. We refer to the second implementation as ISR_2.

E. POST-PROCESSING

As we know the bounding box of the license plate (through the user input for the initial frame), the super-resolved image has already been rectified with respect to the grid in the previous step. However, the license plate may have additional information (e.g., the country and city names) that could mislead or confuse the OCR system. In the post-processing step, we focus on the region which contains the alphanumerics that we want to recognize discarding the other areas. Then, we use Otsu's binarization [11] to facilitate the recognition process. The binarization is performed in addition to a Gaussian blur, to remove possible noisy artifacts. The ultimate output is a rectified, binarized and super-resolved image.

Otsu's binarization is designed for a bimodal image (i.e., an image whose histogram has two peaks – in our case, the color of the alphanumerics and the color of the background). The method might fail when heavy occlusion and shadows are present. Hence, in our solution, the user can also choose between two possibilities in the post-processing step: (a) using Otsu's binarization; or (b) using an adaptive thresholding to binarize the image. Adaptive thresholding may be good when the image has different lighting conditions in different areas. It calculates the threshold for small regions of the image, leading to different values for different regions of the same image (which gives us better results for images with varying illumination). The output of this step is the rectified and binarized image containing the license plate of interest.

F. RECOGNITION

We rely upon two OCR systems to identify the license-plate characters: *Tesseract* [12] and *OCRopus* [13]. Both are free software, released under the Apache 2.0 License. However, these solutions cannot be used directly and require appropriate training. In the following, we describe how we adapt them to the license-plate recognition problem we have in this paper.

Usually, for training traditional OCR systems, we first define the target language comprising the characters that might occur in the dataset (in our case, all the possible license-plate alphanumerics). In doing so, we seek to avoid possible unnecessary mistakes during recognition. For example, the letter ''I'' might be easily confused with the character ''|'' in many font types. However, as we do not expect the character ''|'' in license plates, we do not add it to the target language.

Each OCR system requires several graphical training examples comprising font styles (such as bold and italic), and frequency of specific words. As we do not have enough real-world examples for training the recognition algorithms, we resort to generating synthetic training examples seeking to mimicry real-world license plates.

Given that we use Brazilian license plates in our experiments, we trained the Tesseract and OCRopus using the fonts *Mandatory* (the standard font since 2008) and *DIM Mittleschrift* (most common until 2008). In such font, the letters 'I' and 'O' are equals to the digits '1' and '0', so we divided the license plates in two parts, one for digits and one for characters (the license plates in Brazil have always three letters followed by four digits). We created synthetic images with $17,576$ combinations of letters ($26\times26\times26$) and $9,999$ combinations of digits (the sequence 0000 is not used in Brazil) to train the OCR systems. It is worth mentioning that such training, including the separation between digits and letters, is specific for our particular setup. The OCR systems can be easily trained for license-plates from other countries.

We also added, purposefully, small rotations and noisy artifacts to the synthetic images in order to simulate the real-world license plates more accurately. For an appropriate training process, we also annotate the bounding-box coordinates around every character in each image. At the end of the training process, each OCR is custom-tailored to the problem of license-plate recognition discussed herein. It is also

possible to further extend the training data using real-world license-plates images in addition to the synthetic computer-generated cases. However, it is advised to rectify and binarize those images, as well as annotate them properly.

G. FINAL CONSIDERATIONS

The ultimate products of the framework are the super-resolved image and the sequence of recognized characters. If the framework does not find an appropriate solution, the user intervene by:

1) Restarting the video, and re-selecting the license-plate bounding box. As the choice is manually defined, it is possible that a new initialization may facilitate the following steps (especially the tracking step).
2) Alternatively, the user can easily customize the framework and change the methods to use in each step.

The user can choose from five different tracking methods (PyrLK, PyrDense, SIFT, SURF and ORB), three registration solutions (LK, Dense and None), seven SR algorithms (the variations of *GSR* and *ISR*), two binarization methods, and two different OCR methods, but there is one default method for each step (see in Sec. VI). For implementation details, the source-code will be freely available on GitHub upon acceptance of this paper.

V. EXPERIMENTAL METHODOLOGY

In this present work, we focus the evaluation on the number of correctly recognized characters by the framework in a suspect license plate. The higher the number of hits,[1] the better a given result. As we previously discussed in Sec. II, most ALPR methods either do not provide quantitative validations in their experiments (depicting only a small number of output examples to verify the accuracy of their methods) or do not validate the results with real-world traffic videos.

For validation, we gathered a dataset comprising 200 real-world traffic videos, wherein the movement of the vehicles is away from the camera (one target license plate per video). All collected streams are 1080p HD videos @30 fps. As we have a good resolution of the license plate in the beginning of each video, we manually annotated the correct characters of its target license plate and created its ground-truth file. Unlike the beginning of the video, the license-plate alphanumerics in the last frames are harder to recognize (see Fig. 15). We use those LR last frames to super-resolve and recognize the alphanumerics in our experiments, and then we compare the results to the annotated ground-truths files.

The videos were captured in different places, with different illumination conditions, different vehicle average speeds, non-stationary backgrounds, non-predictable routes, and containing trees and road signs that may cast different shadows over the license plates between consecutive frames.

We execute the framework for the 200 videos, using the five tracking methods, three registration solutions, seven variations of the reconstruction algorithms, two binarization methods, two recognition systems, and from 1 to 10 consecutive frames as input. For each video, we have $5 \times 3 \times 7 \times 2 \times 2 \times 10 = 4,200$ possible results (420 for each number of input frames, and a total of 840, 000 results considering all the 200 videos). This dataset will be freely available online upon acceptance of this paper (including all videos, ground-truth files, license-plate annotations, and the OCR training files).

The experiments also include a qualitative evaluation of the framework. We select visual examples to illustrate that the super-resolution solution can, indeed, increase the readability of license plates in real-world traffic videos.

[1]We use the term "hit" to refer to the the number of correctly recognized characters in a given video.

(a) (b)

FIGURE 15. The license-plate alphanumerics are easily readable in the first video frames (a) but not in the last ones (b). We create ground-truth files with the initial frames, and the last frames are used to be reconstructed and recognized. The same zoom-in factor was applied to the license plates in (a) and (b).

VI. EXPERIMENTS AND RESULTS

We start investigating if our decision of manually selecting the ROI around each license plate (in the initialization step) is appropriate for our framework. The chart in Fig. 16 shows higher recognition rates as we increase the number of input frames. For each video in the dataset, we calculate the highest results that the framework could achieve, from 1 to 10 input frames, using all the available methods.

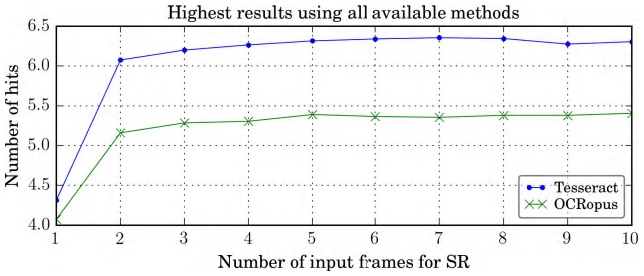


FIGURE 16. Quality of the initialization step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using all the available methods.

Each curve in Fig. 16 is the average of such highest results for each video. The blue curve shows the recognition rate using Tesseract, and the green curve uses OCRopus. Since the number of hits is higher in the reconstructed images, we may claim that our initialization step is providing an appropriate ROI for the super-resolution. Moreover, bad choices for the corners of the license plates would lead to bad tracking, registration, reconstruction, and so on. Hence, good results for the other steps might support our claim that this initialization step is appropriate for the framework. In Secs. VI-A through VI-E,

we validate the other framework's steps. For the tracking, registration, reconstruction and post-processing steps, from now on, we plot only the Tesseract recognition rates (later on we will show it was the best-performing OCR method with a specific experiment).

A. VALIDATION OF THE TRACKING METHODS

Now, we turn our attention to investigating the tracking methods. We start with some tracking issues that the framework should overcome. The first challenge is to keep tracking the license plates while the camera loses focus on the target license plate, as illustrated in Fig. 17. For these cases, normally PyrDense, SIFT, SURF and ORB fail to track the plate, unlike PyrLK. The red dots in the figure depict the bounding boxes tracked by PyrLK. In Fig. 17a, the camera re-focuses on the object after a few frames. In Fig. 17b, the license-plate focus is lost, and not acquired until the end of the video.

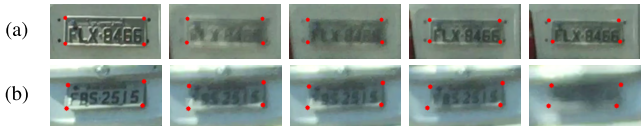


FIGURE 17. PyrLK keeps tracking even if the camera loses focus of the license plate. In (a), the camera re-focuses on the object after a few frames. In (b), the camera does not re-focus on the license plate even after some time.

Due to the pyramidal implementation of Lucas-Kanade and Farneback's dense optical flow, we did not find tracking issues with fast-moving vehicles. In contrast, different cast shadows over a license plate may mislead or confuse the alignment. Fig. 18 depicts consecutive license plates with varying illumination conditions throughout the route (the red dots show that, even in this case, the tracking could be performed). PyrLK could track the bounding box through seven frames of the example, PyrDense tracked it through all the 10 frames, and SIFT, SURF and ORB could not align any consecutive images.

FIGURE 18. Cast shadows may impact the tracking step. The red dots show that PyrLK and PyrDense could track the frames even with a different lighting condition in each consecutive frame.

FIGURE 19. Examples of an input video tracked by PyrLK (a), PyrDense (b), SIFT (c), SURF (d) and ORB (e).

Finally, we select another video to illustrate the tracking performed by each method. Fig. 19 shows the tracked points by each algorithm in the tenth (and last) frame. Note that the tracked corners by PyrLK seems more accurate than the others.

The chart in Fig. 20 depicts that PyrLK flow outperforms the other tracking methods, on average. The blue curve in Fig. 20 is the same as the blue curve in Fig. 16, and shows the best possible values if we alternate among all tracking available.

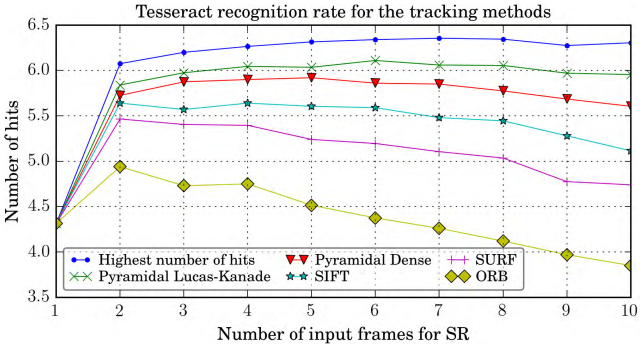


FIGURE 20. Quality of the tracking step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using each of the tracking methods.

We also investigate the number of videos in which the framework correctly identified all the seven license-plate characters (7 is the maximum number of license-plate characters in our dataset) for a given method. We refer to this accuracy number as Acc_7. To find this number, we verify if the method scored seven hits at least once (using from 1 to 10 input frames), in each video. Table 1 confirms that PyrLK is the most promising tracking method, and it is also the fastest solution. Hence, PyrLK results are highlighted in the table, and PyrLK is the default tracking method of the framework.

TABLE 1. Accuracy and runtime of the framework's tracking step. In the second row, the number of videos in which the framework correctly recognized all the seven characters (for some number of input frames among 1 and 10). Then, the runtime (in seconds) to track the license plate through a pair of frames.

Method	PyrLK	PyrDense	SIFT	SURF	ORB
Acc_7	**55.5%**	51.5%	54.0%	53.0%	47.5.0%
Time (s)	**0.02**	0.9	11.2	5.5	0.3

It is worth mentioning that more than one tracking method may recognize the same number of hits in the same video.

B. VALIDATION OF THE REGISTRATION METHODS

In this section, we investigate the best registration solution of the framework. The most important in this step is to improve the result of a bad tracking. Thus, we select an input video as example to illustrate the quality of each registration. In Fig. 21a, we see the ROI of the license plate that we manually defined in the initial frame. In Fig. 21b, we see the tracked corners in the last frame. Our objective is to refine

FIGURE 21. Examples of the registration methods. In (a), the ROI in the initial frame. In (b), the tracked corners in the last frame. In (c), (d) and (e), the registration results using LK, Dense and None.

this tracking. Then, we see the registration results using LK, Dense and None in Figs. 21c, 21d and 21e.

We see in Fig. 21 that only the Lucas-Kanade method could refine the incorrect tracking for the example. However, there are cases in which the tracked corners are good enough to align the images. The chart in Fig. 22 shows that "None" outperforms both the Lucas-Kanade optical flow and the Farneback's dense optical flow, on average.

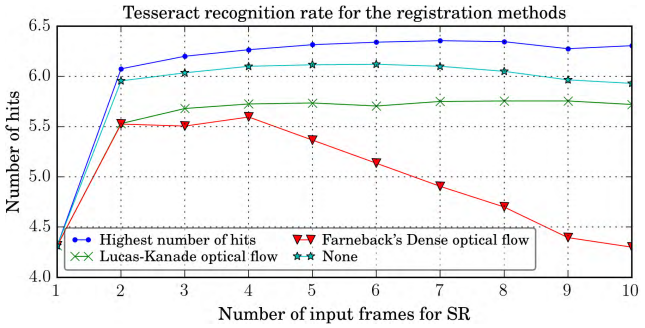


FIGURE 22. Quality of the registration step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using each of the registration solutions.

The Acc_7 values for the registration methods, in Table 2, support the claim that "None" outperforms the other registration solutions, on average (i.e., keeping the previous alignment calculated in the tracking step might be better than performing the subpixel realignment). Using None, the framework correctly recognized all the seven characters in 123 videos (61.5%). The Acc_7 for LK it was 57.0% (114 videos), and for Dense it was 51.5% (103).

However, there were videos in which LK performed better than using no method in the registration step. If we fix PyrLK as the tracking method, we see that LK registration outperformed "None" in 47 videos (and in 40 videos if we track with PyrDense).

TABLE 2. Accuracy and runtime of the framework's registration step.

Method	LK	Dense	None
Acc_7	54.0%	42.5%	**56.5%**
Time (s)	**<0.01**	0.7	-

Using no method for registration achieved better results, on average, that the Lucas-Kanade optical flow because the LK method cannot overcome a bad tracking (very common using the ORB tracker solution, for example). We expect only a subpixel adjustment in the registration step, since we do not work with pyramidal layers. Hence, even if "None" achieved better results, on average, it may still be useful to provide the forensic analysts with other methods for the subpixel adjustment. For this reason, we select LK as the default method for registration in the framework. In addition, the runtime for LK is less than 0.01 seconds, while Dense's runtime is 0.7 seconds. Note that the runtime of Lucas-Kanade and Farneback's optical flow is lower for registration than for tracking (since we do not use pyramid layers for the subpixel adjustment).

C. VALIDATION OF THE RECONSTRUCTION METHODS

For a qualitative evaluation of the reconstruction methods, we also select input videos containing issues with illumination and shadows, as in Secs. VI-A and VI-B. The first row of Fig. 23, shows a zoomed version of the initial frame, without super-resolution, for four input videos. The alphanumerics are hard to recognize in all the examples. The second row shows the results for GSR_4. The last row shows a frame with good resolution of the license plate in the beginning of each video. The GSR_4 characters are not perfectly recognized, but they are more similar to the characters in the last row (in a good resolution), than those in the first row (without super-resolution).

FIGURE 23. Quality of the reconstruction applied to frames with illumination and shadow issues. In (a), the initial frame without super-resolution. Then, the results for GSR_4 (b). In (c), a frame with a good resolution of the license plate.

The chart in Fig. 24 shows the results for the seven variations. It is very difficult to identify the best reconstruction method by the chart (their results are all very similar, on average). The blue curve depicts that the highest number of hits alternates between one method and another, so it is worthwhile to provide different methods for the user.

The chart in Fig. 24 is not adequate to select a default solution for the reconstruction step, since the results for each method are very similar. Therefore, we investigate the Acc_7 values for super-resolution in Table 3. The Acc_7 values for the reconstruction methods in Table 3 are also very similar. We choose GSR_4 as the default super-resolution algorithm, since its result is slightly higher in the table. However, each variation might be more or less advantageous for each case. For example, the Inpainting-based variations found results higher than GSR_4 in 24 videos, and the user can easily change the method when the result is not appropriate.

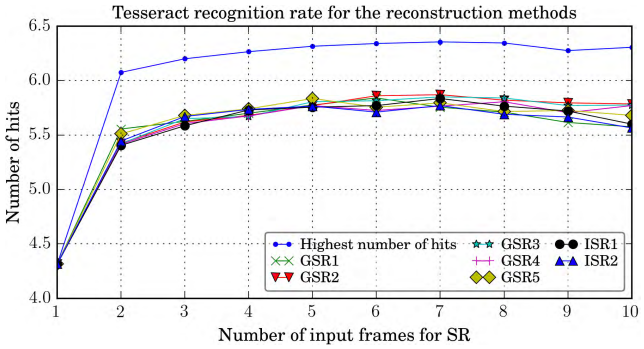


FIGURE 24. Quality of the reconstruction step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using each variation of *GSR*.

TABLE 3. Accuracy and runtime of the reconstruction step.

Method	Acc_7	*Time (s)*
GSR_1	46.5%	0.5
GSR_2	51.0%	0.5
GSR_3	49.0%	0.5
GSR_4	**51.5%**	**0.5**
GSR_5	48.5%	0.5
ISR_1	48.0%	3.0
ISR_2	47.0%	2.5

D. VALIDATION OF THE POST-PROCESSING METHODS

In Sec. VI-A, we showed examples in which the tracking methods could be performed even under the presence of cast shadows. However, such issue may impact the binarization step. In Fig. 18, we showed a license plate with varying illumination conditions throughout the route. Due to the cast shadows and to the sunlight reflected on such plate, each frame might have a different character that is not visible through the vehicle route. In Fig. 25, we show the visual difference when we super-resolve only two frames of the license-plate in Fig. 18. Tesseract could correctly identify all the characters using Adaptive thresholding (EYG-9890), but could not recognize any digits when using Otsu's binarization.

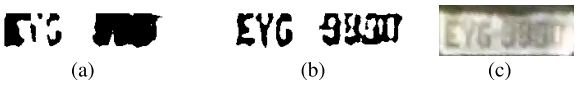


FIGURE 25. Examples of the post-processing methods. In (a), the Otsu's binarization. In (b), Adaptive thresholding. In (c), the super-resolved image.

In Sec. IV-E, we have discussed that the Otsu's binarization is designed for a bimodal image. Usually, a license-plate histogram has two peaks (one for the alphanumerics color, and other to the background color). However, frames containing issues with illumination or shadows, as in the Fig. 23, may lead to a bad Otsu's binarization. Fig. 25 depicts an example in which the Adaptive outperforms Otsu in such cases.

The chart in Fig. 26 summarizes the results for both binarization methods, and shows that the results for both

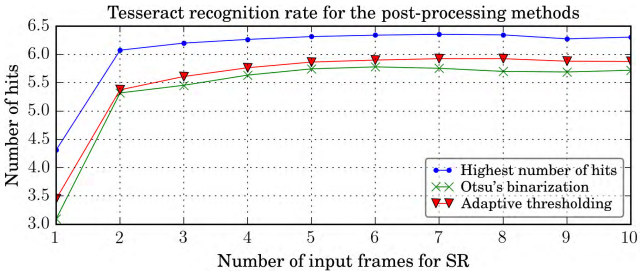


FIGURE 26. Quality of the post-processing step.

binarization methods are similar, if we use Tesseract to calculate the recognition rate.

TABLE 4. Accuracy and runtime of the post-processing step.

Method	Otsu's binarization	Adaptive thresholding
Acc_7	**53.0%**	41.0%
Time (s)	<0.01	<0.01

To define the default solution for the post-processing step, we investigate the Acc_7 values for the binarization methods in Table 4. The Acc_7 for Otsu is higher than for the Adaptive thresholding. In addition, Otsu's runtime is 0.1 seconds, faster than Adaptive (4.6s). Based on these values, we choose the Otsu's binarization as the default solution for the post-processing step. However, we strongly recommend the Adaptive thresholding for the scenes with varying illumination conditions throughout the license-plate's route.

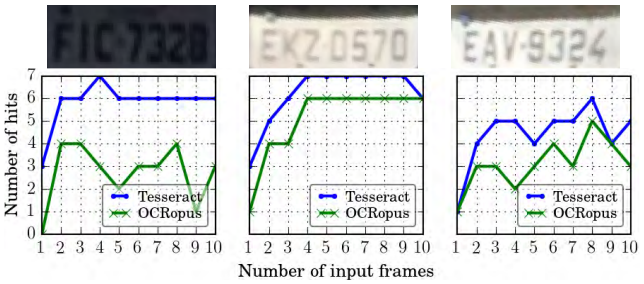


FIGURE 27. Quality of the recognition methods for three super-resolved images, using the defaults methods in each step.

E. VALIDATION OF THE RECOGNITION METHODS

Finally, we investigate the best OCR system in the last step. First, we select three super-resolved images using the default methods for the other steps (PyrLK for tracking, LK for registration, GSR_4 for reconstruction and Otsu for post-processing). In Fig. 27, we show such images and the recognition rate for each one. Using Tesseract, we obtain higher number of hits as we increase the number of input frames, for the three examples. Using OCRopus, the recognition rate seems more stable as we increase the number of frames.

As we showed in Figure 16, the Tesseract recognition rate also outperforms the OCRopus number of hits, on average, using all the available methods and all the videos in

TABLE 5. Accuracy and runtime of the recognition step.

OCR system	Tesseract	OCRopus
Acc_7	**61.0%**	32.5%
Time (s)	0.1	4.6

the dataset. The Acc_7 shows Tesseract is more effective than OCRopus, almost twice as effective in a significantly lower runtime (see Table 5).

Due to the results in Fig. 16, Fig. 27 and the Acc_7 values, we choose Tesseract as the default OCR solution.

The charts in Figs. 16, 20, 22, 24 and 26 depict that, when we increase from 1 to only 2 the number of frames used as input for the SR, we already have a significant improvement in the quality of the characters recognition. The curves grow, on average, until five frames and then become stable. Therefore, using more input frames than we used in our experiments may not compensate the consequential increase in execution time. For these reasons, five may be a good number when choosing the length of an input sequence. Notwithstanding, all the available methods in our framework run in linear time as we increase the number of input frames to super-resolve.

VII. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This work presents a free and open-source framework that relies upon super-resolution and automatic license-plate recognition to help forensic analysts to identify the license plate's alphanumerics of the vehicle of a criminal suspect, for example in a crime scene. The framework handles the necessary steps to identify the target license plate, using a methodology to locate, track, align, super-resolve, and recognize the alphanumerics in low-quality and challenging traffic videos.

The framework provides a user interface for the forensic analyst to choose the license plate of interest. The interaction with the user is required only in the initial step (this is reasonable, since she needs to identify which of the moving vehicles in the video is the suspect one). The only input of the framework is the traffic video, and the ultimate result is not only a super-resolved image, as in the traditional SR techniques, but the recognized license-plate alphanumerics.

After this initialization, the framework comprises a series of methods for tracking, registration, super-resolve, post-process and recognize the alphanumerics of a detected license plate. Each method may be more or less advantageous, depending on a number of situations (e.g, different illumination conditions through the license-plate's route, high vehicle speeds, etc). There is one default solution for each step. However, if the framework does not find an appropriate solution, the user can customize the framework and change the method for any of the framework steps.

The experiments showed that the tracking and the registration play a key role in the framework. The pyramidal implementation of the Lucas-Kanade optical flow, combined to Shi-Tomasi corner points, outperformed the pyramidal Farneback's dense optical flow and the classic feature detectors SIFT, SURF and ORB in the tracking step, on average. Due to the pyramid layers, we did not have issues with fast-moving vehicles with both the LK and Dense tracker. In addition, only PyrLK could properly keep tracking the license plates in situations when the camera loses focus on the license plate, and then re-focuses on the object after a few frames. However, PyrDense outperformed the pyramidal LK optical flow when there were cast shadows over the license plate.

The LK method also outperformed Dense optical flow in the registration step (both without pyramid layers). In addition, the subpixel adjustment improved the number of hits in several cases. The LK's runtime is significantly lower than all the other methods for tracking and registration.

For the reconstruction step, we considered the Geometric K-Nearest Neighbors Super-Resolution (GSR) and the novel Inpainting-based Super-Resolution (ISR). The experiments show that, indeed, it is possible to increase the number of recognized characters using our super-resolution method. From the charts in Sec. VI, we can notice a significant improvement in the quality of the characters recognition even if we increase the number of super-resolved frames from 1 to only 2. As a suggestion, collecting five frames may be a good amount of input frames to use in the super-resolution step. Furthermore, all the seven variations that we implemented produced similar results in the reconstruction step.

In the post-processing step, the binarization methods also had approximate results (Otsu's binarization performed better in bimodal images, whereas Adaptive thresholding helped to recognize characters in license plates when heavy occlusion or cast shadows are present). Finally, Tesseract OCR is faster and could correctly recognize almost twice the number of characters than OCRopus, on average.

For all these reasons, we suggest PyrLK for the tracking method, Lucas-Kanade optical flow without pyramid layers for the subpixel adjustment, and Tesseract to recognize the alphanumerics. All the seven reconstruction variations performed similarly, on average, in our experiments. Moreover, they also run in a small execution time. Since the GSR_4 recognized all the seven license-plate characters in more input videos than the other variations, we chose GSR_4 as the default method for the reconstructed step. However, each variation might be more or less advantageous for each case. For example, the Inpainting-based variations found results higher than GSR_4 in 12% of the videos (and the results were equal to GSR_4 in 80% of the dataset), so the user can change the method when the result is not appropriate. In addition, we strongly recommend the Adaptive thresholding for the post-processing step for the scenes with varying cast shadows or illumination conditions throughout the license-plate's route. Otherwise, we suggest using Otsu's binarization.

The collected dataset is another contribution of this work. We collected real-world traffic videos, containing one target license plate per video. Each video has an associated ground-truth file with the annotated characters of the suspect plate. The videos were captured in different places, under different

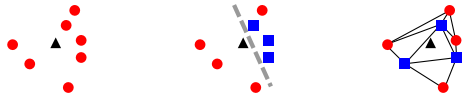


FIGURE 28. In (a), the black triangle is the target pixel p and each red dot is its nearest pixel q_k from an LR image I_k. In (b), 3-NN chooses the blue pixels to be combined, but they may not be a good choice because they do not form a region to which p belongs. In (c), a triangulation is calculated among all q_k, and then p belongs to the triangle formed by the blue squares.

illumination conditions, different vehicle average speeds, non-stationary backgrounds, non-predictable routes, and containing trees and road signs that may cast different shadows over the license plates between consecutive frames.

It is worth mentioning that the framework can be extended upon the user's need. For future work, one could think of including other methods for the tracking, registration, reconstruction, post-processing and recognition steps. For example, GSR_2 and GSR_3 can produce bad results if the three closest values of q_k (the LR nearest pixels) do not form a region which includes p (the point to be reconstructed). Fig. 28 illustrates an example of this situation: it is possible to find a straight line that passes through p such that all the three nearest points q_k are on the same side of the line. In this case, the second and third nearest q_k may not add relevant information to p.

In [65], the authors propose a Delaunay triangulation for all available points and combine the vertices of the triangle to which p belongs (as in Fig. 28c). According to [66], the triangulation can be build in $\mathcal{O}(n \log n)$, while GSR_1 is $\mathcal{O}(n)$ (for an HR image with n pixels). We did not implement this option to avoid increasing the algorithm runtime for now, but one could investigate this and other triangulations in the future.

The Geometric K-Nearest Neighbor Super-Resolution algorithm combines pixels in a neighborhood onto the pixels of the HR grid. Another possible strategy would be projecting the LR images directly onto the HR grid (using the mapping matrices calculated during the registration step). Each LR pixel would contribute to exactly one high-resolution pixel. However, there might be pixels in the HR grid onto which no LR pixel has been projected. Then, to complete these unknown pixels, one might use some noisy-removal technique or even use some in-painting method [64].

Another suggestion for future work is to automatically locate the license plate, substituting the user interaction in the first step. Hence, it might be possible to create a completely automatic and unsupervised framework, and use it for real-time applications. In addition, it might be useful to automatically detect and discard blurred frames that may appear while the camera loses focus, and improve the recognition training process including real-world license-plate images.

REFERENCES

[1] 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, Feb. 2013.
[2] 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 Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 377–391, Sep. 2008.
[3] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42–53, Mar. 2004.
[4] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016.
[5] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.
[6] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.
[7] K. Zhang, D. Tao, X. Gao, X. Li, and Z. Xiong, "Learning multiple linear mappings for efficient single image super-resolution," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 846–861, Mar. 2015.
[8] A. Telea, "An image inpainting technique based on the fast marching method," *J. Graph. Tools*, vol. 9, no. 1, pp. 23–34, 2004.
[9] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier–Stokes, fluid dynamics, and image and video inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2001, pp. 355–362.
[10] H. Seibel, Jr., S. Goldenstein, and A. Rocha, "Fast and effective geometric k-nearest neighbors multi-frame super-resolution," in *Proc. IEEE Int. Conf. Graph., Patterns Imag.*, Aug. 2015, pp. 103–110.
[11] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
[12] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. IEEE Int. Conf. Document Anal. Recognit.*, Sep. 2007, pp. 629–633.
[13] T. M. Breuel, "The OCRopus open source OCR system," *Proc. SPIE*, vol. 6815, pp. 68150F-1–68150F–15, Jan. 2008.
[14] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer-Verlag, 2010.
[15] P. Chatterjee, S. Mukherjee, S. Chaudhuri, and G. Seetharaman, "Application of Papoulis–Gerchberg method in image super-resolution and inpainting," *Comput. J.*, vol. 52, no. 1, pp. 80–89, 2009.
[16] Y.-W. Tai, S. Liu, M. Brown, and S. Lin, "Super resolution using edge prior and single image detail synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2400–2407.
[17] E. Meijering, "A chronology of interpolation: From ancient astronomy to modern signal and image processing," *Proc. IEEE*, vol. 90, no. 3, pp. 319–342, Mar. 2002.
[18] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: A technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21–36, May 2003.
[19] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.
[20] R. Y. Tsai and T. S. Huang, "Multiframe image restoration and registration," *Adv. Comput. Vis. Image Process.*, vol. 1, no. 2, pp. 317–339, 1984.
[21] K. Nasrollahi and T. B. Moeslund, "Super-resolution: A comprehensive survey," *Mach. Vis. Appl.*, vol. 25, no. 6, pp. 1423–1468, 2014.
[22] J. Tian and K.-K. Ma, "A survey on super-resolution imaging," *Signal, Image Video Process.*, vol. 5, no. 3, pp. 329–342, 2011.
[23] M. Irani and S. Peleg, "Super resolution from image sequences," in *Proc. Int. Conf. Pattern Recognit.*, vol. C-90. Jun. 1990, pp. 115–120.
[24] A. Zomet, A. Rav-Acha, and S. Peleg, "Robust super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2001, pp. 645–650.
[25] A. Papoulis, "A new algorithm in spectral analysis and band-limited extrapolation," *IEEE Trans. Circuits Syst.*, vol. 22, no. 9, pp. 735–742, Sep. 1975.
[26] R. W. Gerchberg, "Super-resolution through error energy reduction," *Opt. Acta, Int. J. Opt.*, vol. 21, no. 9, pp. 709–720, 1974.
[27] P. Vandewalle and S. E. Süsstrunk, and M. Vetterli, "Superresolution images reconstructed from aliased images," *Proc. SPIE*, vol. 5150, pp. 1398–1405, Jun. 2003.
[28] H. Stark and P. Oskoui, "High-resolution image recovery from image-plane arrays, using convex projections," *J. Opt. Soc. Amer. A*, vol. 6, no. 11, pp. 1715–1726, 1989.

[29] P. Cheeseman, B. Kanefsky, R. Kraft, J. Stutz, and R. Hanson, "Super-resolved surface reconstruction from multiple images," in *Maximum Entropy and Bayesian Methods*, G. R. Heidbreder, Ed. Dordrecht, The Netherlands: Springer, 1996, pp. 293–308.

[30] M.-C. Chiang and T. E. Boult, "Efficient image warping and super-resolution," in *Proc. IEEE Workshop Appl. Comput. Vis.*, Dec. 1996, pp. 56–61.

[31] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super resolution," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1327–1344, Oct. 2004.

[32] K. Simonyan, S. Grishin, D. Vatolin, and D. Popov, "Fast video super-resolution via classification," in *Proc. IEEE Int. Conf. Image Process.*, Apr. 2008, pp. 349–352.

[33] H. Nasir, V. Stanković, and S. Marshall, "Singular value decomposition based fusion for super-resolution image reconstruction," *Signal Process. Image Commun.*, vol. 27, no. 2, pp. 180–191, 2012.

[34] T. Pham, L. van Vliet, and K. Schutte, "Robust fusion of irregularly sampled data using adaptive normalized convolution," *EURASIP J. Appl. Signal Process.*, vol. 2006, no. 1, pp. 1–12, 2006.

[35] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.*, vol. 22, no. 2, pp. 56–65, Mar./Apr. 2002.

[36] Y. Romano, J. Isidoro, and P. Milanfar, "RAISR: Rapid and accurate image super resolution," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 110–125, Oct. 2017.

[37] G. Caner, A. M. Tekalp, and W. Heinzelman, "Super resolution recovery for multi-camera surveillance imaging," in *Proc. Int. Conf. Multimedia Expo*, Jul. 2003, p. 109.

[38] K. V. Suresh, G. M. Kumar, and A. N. Rajagopalan, "Superresolution of license plates in real traffic videos," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 321–331, Jun. 2007.

[39] J. Yuan, S. dan Du, and X. Zhu, "Fast super-resolution for license plate image reconstruction," in *Proc. Int. Conf. Pattern Recognit.*, 2008, pp. 1–4.

[40] A. Camargo, R. R. Schultz, Y. Wang, R. A. Fevig, and Q. He, "GPU-CPU implementation for super-resolution mosaicking of unmanned aircraft system (UAS) surveillance video," in *Proc. IEEE Southw. Symp. Image Anal. Interpretation*, May 2010, pp. 25–28.

[41] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.

[42] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[43] M. Kim and H. Ko, "Resolution enhancement of ROI from surveillance video using bernstein interpolation," in *Proc. IEEE Int. Conf. Adv. Video Signal-Based Surveill.*, Aug. 2011, pp. 331–336.

[44] J. Kolibal and D. Howard, "The novel stochastic Bernstein method of functional approximation," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst.*, Jun. 2006, pp. 97–100.

[45] T. Yoshida, T. Takahashi, D. Deguchi, I. Ide, and H. Murase, "Robust face super-resolution using free-form deformations for low-quality surveillance video," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2012, pp. 368–373.

[46] K. Zarei, W. V. Aarle, K. J. Batenburg, and J. Sijbers, "Super-resolution of license plate images using algebraic reconstruction technique," *J. Image Graph.*, vol. 1, pp. 94–98, Apr. 2013.

[47] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. Piscataway, NJ, USA: IEEE Press, 1988, ch. 7, pp. 275–296.

[48] W. Lina and L. Ying, "A license plate super-resolution reconstruction algorithm based on manifold learning," in *Proc. Int. Conf. Comput. Sci. Eng.*, 2014, pp. 1855–1859.

[49] H. Rajput, T. Som, and S. Kar, "Using radon transform to recognize skewed images of vehicular license plates," *IEEE Comput.*, vol. 49, no. 1, pp. 59–65, Jan. 2016.

[50] Q. Tang, Z.-Y. Wang, X.-W. Xu, Y.-H. Liang, and X.-Y. Cao, "A rapid inclined distortion rectification method for vehicle licenses plate images," in *Proc. IEEE Int. Conf. Mach. Learn. Cybern.*, vol. 5. Sep. 2008, pp. 2744–2748.

[51] Y. Qing, Z. L. Hong, Z. S. Hong, and L. Xue, "A new fast algorithm to rectify tilt image of vehicle license plates," in *Proc. IEEE Control Conf.*, May 2007, pp. 485–488.

[52] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[53] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[54] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[55] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Apr. 1994, pp. 593–600.

[56] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm," Intel Corp. Microprocess. Res. Labs, Tech. Rep., 2000.

[57] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. 13th Scandin. Conf. Image Anal.*, 2003, pp. 363–370.

[58] Y. Yuan, S. Emmanuel, Y. Fang, and W. Lin, "Visual object tracking based on backward model validation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 11, pp. 1898–1910, Nov. 2014.

[59] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2009, pp. 331–340.

[60] O. L. Meur and C. Guillemot, "Super-resolution-based inpainting," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2012, pp. 554–567.

[61] T. F. Chan, M. K. Ng, A. C. Yau, and A. M. Yip, "Superresolution image reconstruction using fast inpainting algorithms," *Appl. Comput. Harmon. Anal.*, vol. 23, no. 1, pp. 3–24, Jul. 2007.

[62] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. Washington, DC, USA: V. H. Winston & Sons, 1977.

[63] J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*. Berlin, Germany: Springer, 1999.

[64] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. Conf. Comput. Graph. Interact. Techn.*, 2000, pp. 417–424.

[65] S. Lertrattanapanich and N. K. Bose, "High resolution image formation from low resolution frames using Delaunay triangulation," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1427–1441, Dec. 2002.

[66] L. P. Chew, "Constrained delaunay triangulations," in *Proc. 3rd Annu. Symp. Comput. Geometry*, 1987, pp. 215–222.

HILÁRIO SEIBEL, JR., received the degree and the M.Sc. degree in computer science from UFES, Brazil, in 2004 and 2007, respectively. He is currently pursuing the Ph.D. degree with Unicamp, Brazil, and his research involves super-resolution techniques for forensics and general applications. He has been a permanent Professor with the Federal Institute of Espírito Santo, Brazil, since 2007. His main interests include visual computing, digital forensics, computer programming, and information retrieval.

SIOME GOLDENSTEIN (SM'16) received the degree in electronic engineering from UFRJ, Brazil, in 1995, the M.Sc. degree in computer science from PUC-Rio, Brazil, in 1997, and the Ph.D. degree in computer and information science from UPenn, USA, in 2002. He is currently an Associate Professor with the Institute of Computing, Unicamp, Brazil. His interests lie in computational forensics, computer vision, computer graphics, and machine learning. He is an Area Editor of the IEEE TIFS, CVIU, and GMOD, and has been in the program committee of multiple conferences and workshops.

ANDERSON ROCHA (SM'15) is currently an Associate Professor with the Institute of Computing, Unicamp. His main interests include digital forensics, reasoning for complex data, and machine intelligence. He is currently an associate editor of several international journals, such as the IEEE TIFS and the IEEE S&P. He is a Microsoft and Google Faculty Fellow and an Affiliate Member of the Brazilian Academy of Sciences and the Brazilian Academy of Forensics Sciences.