Universidade Estadual de Campinas
Instituto de Computação

Hilario Seibel Junior

Super-resolution in low-quality videos for forensics, surveillance, and mobile applications

Super-resolução em vídeos de baixa qualidade para aplicações forenses, de vigilância e móveis

CAMPINAS
2017

Hilario Seibel Junior

Super-resolution in low-quality videos for forensics, surveillance, and mobile applications

Super-resolução em vídeos de baixa qualidade para aplicações forenses, de vigilância e móveis

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Siome Klein Goldenstein (Orientador)
Co-supervisor/Coorientador: Prof. Dr. Anderson de Rezende Rocha (Co-orientador)

Este exemplar corresponde à versão final da Tese defendida por Hilario Seibel Junior e orientada pelo Prof. Dr. Siome Klein Goldenstein (Orientador).

CAMPINAS
2017

Agência(s) de fomento e nº(s) de processo(s): CAPES, 1197478,146886; CNPq, 153996/3-2015

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Se41s Seibel Junior, Hilario, 1981-
Super-resolution in low-quality videos for forensics, surveillance, and mobile applications / Hilario Seibel Junior. – Campinas, SP : [s.n.], 2017.

Orientador: Siome Klein Goldenstein.
Coorientador: Anderson de Rezende Rocha.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Resolução (Óptica). 2. Videovigilância. 3. Investigação criminal. 4. Telefone celular. I. Goldenstein, Siome Klein, 1972-. II. Rocha, Anderson de Rezende, 1980-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Super-resolução em vídeos de baixa qualidade para aplicações forenses, de vigilância e móveis
Palavras-chave em inglês:
Resolution (Optics)
Video surveillance
Criminal investigation
Cell phones
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Anderson de Rezende Rocha [Coorientador]
Flávio de Barros Vidal
Daniel de Oliveira Cunha
Fernanda Alcântara Andaló
Hélio Pedrini
Data de defesa: 29-09-2017
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Hilario Seibel Junior

Super-resolution in low-quality videos for forensics, surveillance, and mobile applications

Super-resolução em vídeos de baixa qualidade para aplicações forenses, de vigilância e móveis

Banca Examinadora:

- Prof. Dr. Anderson de Rezende Rocha
  IC/Unicamp
- Prof. Dr. Flávio de Barros Vidal
  UnB
- Dr. Daniel de Oliveira Cunha
  INC/DPF
- Profa. Dra. Fernanda Alcântara Andaló
  IC/Unicamp
- Prof. Dr. Hélio Pedrini
  IC/Unicamp

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 29 de setembro de 2017

Acknowledgements

The authors thank the National Council for Scientific and Technological Development – CNPq (Grant #304472/2015-8), the Coordination for the Improvement of Higher Education Personnel – CAPES (Grant #DeepEyes), the São Paulo Research Foundation – FAPESP (DéjàVu Grant #2015/19222-9), and Microsoft Research for the financial support.

Resumo

Algoritmos de super-resolução (SR) são métodos para obter um aumento da resolução de imagens compostas por pixels. Na super-resolução por múltiplas imagens, um conjunto de imagens de baixa resolução de uma cena é combinado para construir uma imagem de resolução superior. Super-resolução é uma solução barata para superar as limitações dos sistemas de aquisição de imagens, e pode ser útil em diversos casos em que o dispositivo não pode ser melhorado ou substituído – mas em que é possível obter diversas capturas da mesma cena. Neste trabalho, é explorada a super-resolução por múltiplas imagens para imagens digitais, em cenários nos quais é possível obter diversas imagens de uma cena. São propostas cinco variações de um método que explora propriedades geométricas de múltiplas imagens de baixa resolução para combiná-las em uma imagem de resolução superior; duas variações de um método que combina técnicas de *inpainting* e super-resolução; e mais três variações de um método que utiliza filtros adaptativos e regularização para resolver um problema de mínimos quadrados.

Super-resolução por múltiplas imagens é possível quando existe movimento e informações não redundantes entre as imagens de baixa resolução. Entretanto, combiná-las em uma imagem de resolução superior pode não ser computacionalmente viável por técnicas complexas de super-resolução. A primeira aplicação dos métodos propostos é para um conjunto de imagens capturadas pelos dispositivos móveis mais recentes. Este tipo de ambiente requer algoritmos eficazes que sejam executados rapidamente e utilizando baixo consumo de memória.

A segunda aplicação é na Ciência Forense. Câmeras de vigilância espalhadas pelas cidades poderiam fornecer dicas importantes para identificar um suspeito, por exemplo, em uma cena de crime. Entretanto, o reconhecimento dos caracteres de placas veiculares é especialmente difícil quando a resolução das imagens é baixa. Por isso, este trabalho também propõe um arcabouço que realiza a super-resolução de placas veiculares em vídeos reais de vigilância, capturados por câmeras de baixa qualidade e não projetadas especificamente para esta tarefa, ajudando o especialista forense a compreender um evento de interesse. O arcabouço realiza todas as etapas necessárias para rastrear, alinhar, reconstruir e reconhecer automaticamente os caracteres de uma placa suspeita. O usuário recebe, como saída, a imagem de alta resolução reconstruída, mais rica em detalhes, e também a sequência de caracteres reconhecida automaticamente nesta imagem.

São apresentadas validações quantitativas e qualitativas dos algoritmos propostos e de suas aplicações. Os experimentos mostram, por exemplo, que é possível aumentar o número de caracteres reconhecidos corretamente, colocando o arcabouço proposto como uma ferramenta importante para fornecer aos peritos uma solução para o reconhecimento de placas veiculares sob condições adversas de aquisição. Por fim, também é sugerido o número mínimo de imagens a ser utilizada como entrada em cada aplicação.

Abstract

Super-resolution (SR) algorithms are methods for achieving high-resolution (HR) enlargements of pixel-based images. In multi-frame super resolution, a set of low-resolution (LR) images of a scene are combined to construct an image with higher resolution. Super resolution is an inexpensive solution to overcome the limitations of image acquisition hardware systems, and can be useful in several cases in which the device cannot be upgraded or replaced, but multiple frames of the same scene can be obtained. In this work, we explore SR possibilities for digital images, in scenarios wherein we have multiple frames of a same scene. We design and develop five variations of an algorithm which rely on exploring geometric properties in order to combine pixels from LR observations into an HR grid; two variations of a method that combines inpainting techniques to multi-frame super resolution; and three variations of an algorithm that uses adaptive filtering and Tikhonov regularization to solve a least-square problem.

Multi-frame super resolution is possible when there is motion and non-redundant information from LR observations. However, combining a large number of frames into a higher resolution image may not be computationally feasible by complex super-resolution techniques. The first application of the proposed methods is in consumer-grade photography with a setup in which several low-resolution images gathered by recent mobile devices can be combined to create a much higher resolution image. Such always-on low-power environment requires effective high-performance algorithms, that run fastly and with a low-memory footprint.

The second application is in Digital Forensic, with a setup in which low-quality surveillance cameras throughout the cities could provide important cues to identify a suspect vehicle, for example, in a crime scene. However, license-plate recognition is especially difficult under poor image resolutions. Hence, 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 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 details, and also the sequence of license-plates characters that have been automatically recognized in the super-resolved image.

We present quantitative and qualitative validations of the proposed algorithms and its applications. Our experiments show, for example, that SR can 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. Finally, we also suggest a minimum number of images to use as input in each application.

List of Symbols

Input and output images for the super resolution

n	Number of images to be used as input for the super-resolution.
I_k	Low-resolution images used as input for the super-resolution ($1 \leq k \leq n$).
I_{HR}	A high-resolution image that is used as target for the super-resolution.
I_G	The target image I_{HR} with a Gaussian blur to avoid the aliasing after the down-sampling.
I_{SR}	An image reconstructed by a super-resolution algorithm.
M, N	Dimensions of the image I_{HR}.
M_k, N_k	Dimensions of each image I_k.

Other inputs for the super resolution

Υ_k	Transformation matrices mapping each image I_k onto the HR grid ($1 \leq k \leq n$).
Ψ_k	Transformation matrices mapping each I_k onto I_1 ($2 \leq k \leq n$).
σ	Standard deviation of the Gaussian distribution.
d	An integer such that the radius $r = d \times \sigma$ defines the kernel size of the super-resolution.
α	The regularization parameter. Larger values increase the conditioning of the problem, reducing the variance of the estimates.

Surveillance videos

F_k	A video frame that is used as input for the super-resolution.

Contents

Chapter 1

Introduction

1.1 Motivation

The quality of a digital image is directly related to its resolution. In this context, the word "resolution" is defined not as an image size, but as a measure of the amount of detail that is visible in an image [42, 120]. The higher the resolution of a digital image, the more accurate its representation of the real scene. However, as the resolution of the image generated by a sensor increases, so does the cost of the sensor and hence it may not be an affordable solution [5, 87].

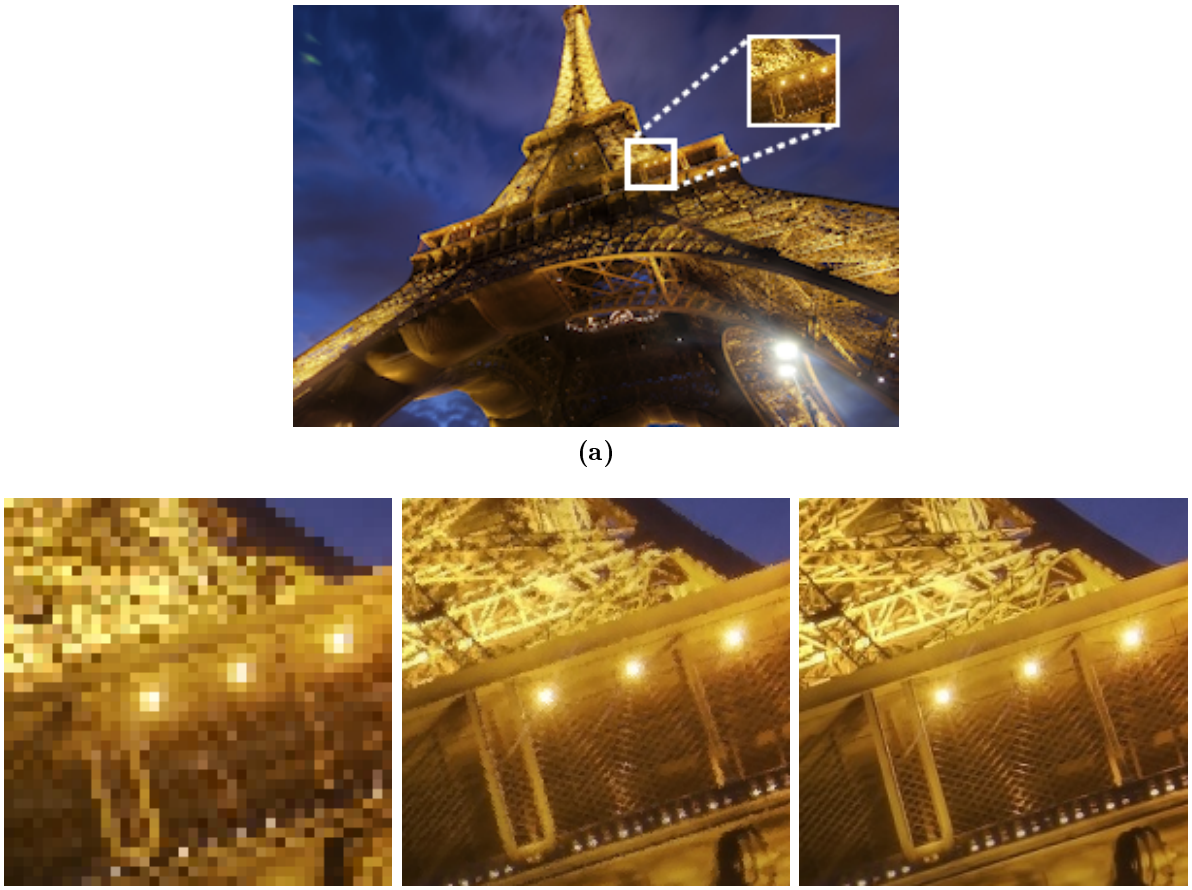
A digital image stores and represents information of a real scene by a finite number of samples. Several techniques use interpolation [29, 64, 72, 10] to increase the spatial resolution of a digital image. However, single-image interpolation cannot recover the high-frequency components lost or degraded during the sampling process [84]. Limits on the resolution of the original imaging device can be improved by using signal processing techniques to obtain a high-resolution (HR) image from observed multiple low-resolution (LR) images [20, 136, 84]. The fusion of information from various observations of the same scene is called Multi-Frame Super Resolution, or simply Super Resolution (SR).

The super-resolution image reconstruction is proved to be useful in many practical cases in which multiple frames of the same scene can be obtained [84], including medical imaging, satellite imaging, target recognition in military applications, license plate readers, and surveillance videos [38, 109, 59, 80]. The basic premise for a multi-frame super-resolution algorithm is the availability of multiple images, with small relative motion, captured from a scene. Subpixel motion provides new details for the super-resolved image that would not be found using simple interpolation. Although such requirement does not enable us to use SR in some situations, there are several cases wherein we can easily capture a sequence of images and subsequently super resolve them. For example, it may be very expensive to replace obsolete camera sensors in satellites, but a super-resolution technique could be an inexpensive solution via software to overcome this limitation.

The main purpose of this work is to explore the SR possibilities for digital images, in scenarios wherein we have multiple frames of scene. We design and develop five variations of an algorithm which rely on exploring geometric properties in order to combine pixels from LR observations into an HR grid; two variations of a method that leverages inpainting

methods [114, 7] for multi-frame super resolution; and three variations of an algorithm that uses Tikhonov regularization [116] to solve a least-square problem.

For a sneak peak showing the potential of the work we present herein, Figure 1.1 compares the result of simple interpolation to an image super resolved by one of our algorithms, using 35 low-resolution observations of the scene as input[1].



(a)

(b) **(c)** **(d)**

Figure 1.1: Super resolution *versus* interpolation: (a) piece of an Eiffel Tower's low-resolution image; (b) same piece zoomed by a factor of 5 using a simple nearest neighbor interpolation; (c) the piece super resolved by one of our SR algorithms, using 35 low-resolution images as input; and (d) equivalent piece in an HR target image.

To validate our algorithms, we first turn our attention to consumer-grade photography using recent mobile phones, that take dozens of photos per second. Those cameras can gather a set of images while somebody is holding the camera manually in approximately the same position, and we use input images with such small camera shake to create a high-resolution image. In doing so, the user might take multiple photos with a cheap camera, and then obtain an ultimate super-resolved image (as if the picture was gathered by a more powerful and expensive camera). It is worth mentioning that the SR algorithms might be fast and with a low-memory footprint, in order to be executed in this always-on low-power environment.

[1]The original picture of the Eiffel Tower was collected from *http://www.gratisography.com/* and is free of Copyright Restrictions. Anyone can copy, modify, and distribute the work, without asking permission.

We also design and develop a novel, free and open-source end-to-end framework to super resolve and recognize license-plate characters in low-quality real-world traffic videos, aiding forensic analysts in understanding an event of interest. Automatic license-plate recognition (ALPR) uses optical character recognition (OCR) on images to extract and recognize the alphanumerics of a vehicle registration plate [32, 3]. 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) [18]. 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. Figure 1.2 depicts a situation in which the license-plate characters may not be easily identified even in a high-resolution video.

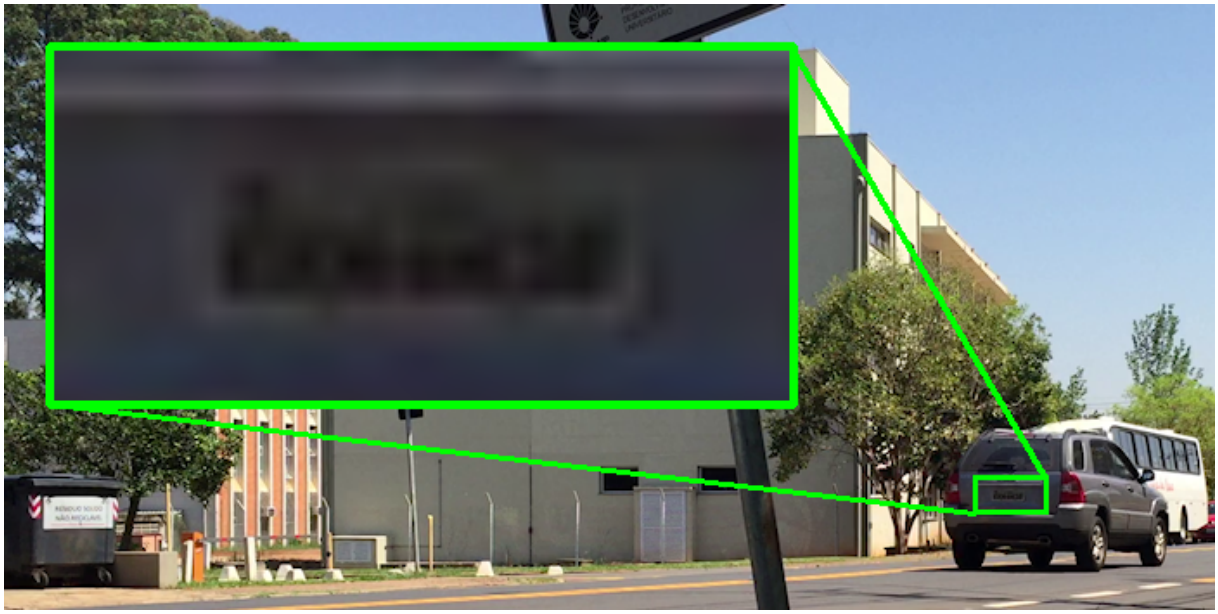


Figure 1.2: Surveillance cameras scattered throughout the cities could help to identify a suspect vehicle, for example, in a crime scene. But even in a high-resolution camera (e.g., 1920×1080 pixels in this frame) it might be difficult to visually recognize license-plate characters.

Not only OCR systems but also forensic specialists may fail to recognize the alphanumerics in such setups, and super-resolution techniques can be an inexpensive path, via software, to overcome this limitation. Therefore, we leverage super-resolution techniques to combine information from consecutive frames into a single license-plate image, richer in details. 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. We focus on enhancing the details in vehicle license plates that could help to identify a criminal suspect or activity in a crime scene, super resolving only a region of interest (ROI) of the video, and discarding less important parts. The experiment shows that it is possible to increase the number of recognized characters using the proposed super-resolution methods[2]. The framework involves six core steps to perform

[2]The framework is currently trained to recognize brazilian license plates

the license-plate recognition, showed in Figure 1.3:

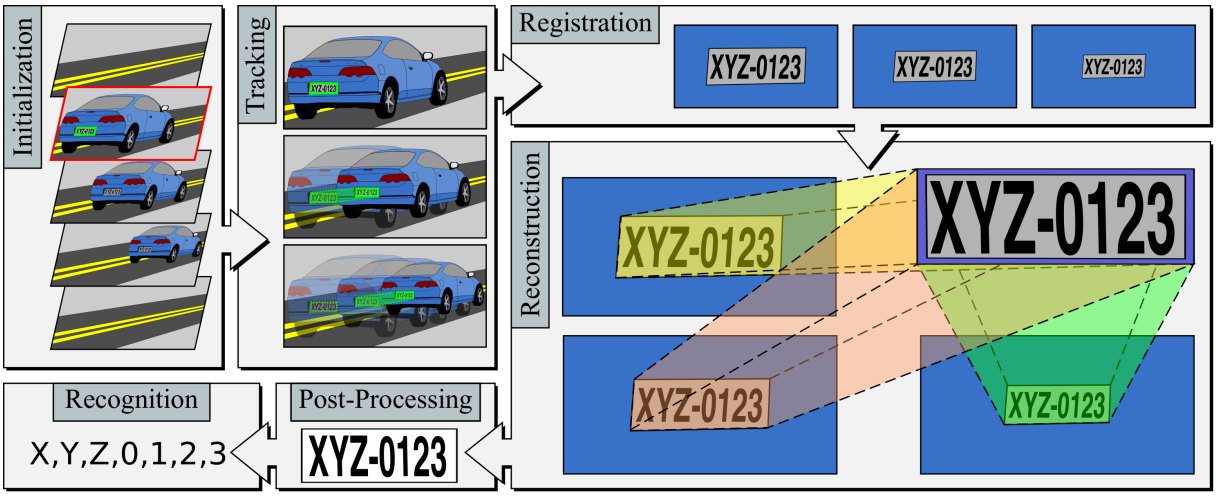


Figure 1.3: Our end-to-end framework pipeline. (1) *Initialization*: Choosing a starting frame and locating the license-plate region in this frame; (2) *Tracking*: Finding re-occurrences of the license plate over the consecutive frames, and aligning the frames with respect to the plate positions; (3) *Registration*: Refining the previous alignment with sub-pixel accuracy; (4) *Reconstruction*: Combining the sequence of consecutive frames into a high-resolution grid; (5) *Post-processing*: Applying image processing operations to the reconstructed image, to improve the results in the recognition step; and (6) *Recognition* of the alphanumerics in the super-resolved license plate.

Each proposed super-resolution method might be more or less advantageous in each setup, with different restrictions, limitations and constraints. The method based on Tikhonov regularization, for example, is the only proposed method designed to recover the high-frequency components lost during the acquisition process. On the other hand, such method is expected to run slower than the other methods, and it might not be computationally attractive for applications that require quick responses and low-memory footprint. For the application on mobile devices, which requires fast responses, the best choice is to select the proposed method that explores geometric properties to combine the LR pixels into an HR grid. On the other hand, the forensic framework reconstructs only a small area of the frames containing the license-plate image. Therefore, the runtime of the method based on Tikhonov regularization does not impact the solution as in the mobile application.

Our first results have already been published in the literature. In [96], we introduce the five variations of the algorithm which rely on exploring geometric properties to combine LR pixels into an HR grid, and we validate such method with sequences of images gathered by a mobile device. In [97], we introduce the forensic framework and the two variations of a method that leverages inpainting methodsfor multi-frame super resolution. Finally, we are currently developing an article with the three variations of the algorithm that uses Tikhonov regularization [116] to solve a least-square problem, and providing a public virtual machine with the final version of the forensic framework.

1.2 Hypothesis

The hypothesis that guides this research is that it is possible to explore geometric properties from multiple low-resolution images in order to combine them into a higher resolution image. Moreover, in doing so, it is possible to achieve good super-resolution results for photos gathered by mobile devices and for license plates in low-quality real-world surveillance videos.

1.3 Objective

Our main goal is to design and develop novel multi-frame super-resolution methods. Furthermore, for the sake of completeness, we describe the specific objectives of this research:

1. Explore different possibilities to combine pixels from different low-resolution observations of a scene into a higher resolution image, richer in details.
2. Provide a super-resolution method that recovers the high-frequency components lost during the acquisition process of the LR images.
3. Provide methods that are computationally attractive for applications that require quick responses and low-memory footprint.
4. Apply the proposed methods to super resolve license plates in surveillance videos, gathered by cameras not designed especially for the recognition task.
5. Super resolve sequences of photos taken by mobile devices, increasing the resolution provided by the device.

1.4 Contributions

The main contribution of this work is in the Visual Computing and Digital Forensics fields. First, we design, develop and mathematically describe a super-resolution algorithm that relies on Tikhonov regularization to solve a system of linear equations using an adaptive filter. Then, we also propose other simple and effective solutions to super resolve multiple low-resolution images using their geometric properties, and discuss the best number of images that should be used as input in our scenarios.

Additionally, we create 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.

Moreover, we super resolve photos from mobile phones, improving the resolution limitation of the device. For this particular scenario, we design algorithms with low-memory footprint and smaller execution time, so they can be properly applied to an always-on low-power environment.

Finally, we provide two new datasets:

1. The first one is for validation of super-resolution algorithms, containing 100 target images to be reconstructed, $10,000$ low-resolution images (100 for each target image), the $10,000$ transformation matrices mapping each LR image onto its associated target image, and the correct alignment among the input images for the super resolution. We also make available a set of scripts to extend this dataset. Therefore, anyone can introduce new target images to the dataset and, then, generate the LR images to be reconstructed, calculate their transformation matrices and also produce the correct alignment among the low-resolution images.

2. The other dataset is designed to validate super resolution of license-plates. It contains 200 real-world traffic videos, wherein the movement of the vehicles is away from the camera (one target license plate per video). The dataset also provides the ground-truth for each target license plate. The videos have been 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.

1.5 Outline

This document is organized as follows: Chapter 2 provides background concepts for the rest of the reading: the resolution increasing problem is covered, as well as its variations (interpolation, multi-frame SR, and single-image SR), and some related work. In Chapter 3, we propose novel methods to super resolve multiple frames from a sequence of images. The proposed super-resolution methods are validated in Chapter 4. Then, Chapter 5 describes the challenges to super resolve license plates in videos, and introduces a novel framework that super resolves a set of input frames from a video, and automatically recognizes the license-plate characters in the reconstructed image. The framework is validated in Section 6. In chapter 7, we discuss our contributions, results, final considerations, and future work. Finally, Appendix A shows additional charts of the results, and Appendix B has implementation details about the framework.

Chapter 2

Literature review

In this chapter, we present the state of the art for super resolution, including those works addressing SR for surveillance videos and for mobile devices. We first present the main concepts about resolution enhancement. Then, we describe the related work and main differences with respect to our work.

2.1 Resolution increase

A digital image stores and represents information of a real-world scene by a finite number of samples. Interpolation (or upsampling) [29, 101, 22, 81, 64, 72, 10] is used to increase the spatial resolution of a digital image, finding out new samples among those that are already known. Let $f(x, y)$ be an image in a lower resolution, and $g(x, y)$ be the resulting higher resolution image. Interpolation typically is implemented by convolving the image $f(x, y)$ with a $2D$ kernel $h(x, y)$, as expressed in Eq 2.1:

$$g(x, y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f(i, j)h(x - mi, y - mj), \tag{2.1}$$

where m is the upsampling factor (in practice, the spatial extent of the image is finite). Many popular image interpolation methods are defined in this way, including nearest-neighbor, bilinear and bicubic interpolation [101, 64, 72, 10]. Interpolation is an ill-posed problem since there are many HR images that may have the same LR samples. This ambiguity increases as the intended magnification becomes larger [19, 112].

Although interpolation has been extensively studied since ancient times [72], the quality of an image magnified from an aliased low-resolution image is inherently limited. Single-image interpolation cannot recover the high-frequency components lost or degraded during the sampling process [84]. To achieve further improvements in this field, it is natural to seek multiple data sets in which additional data 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.

2.2 Super resolution

Multi-frame super resolution (MFSR), usually defined only as super resolution (SR), is a process of constructing an HR image using one or more LR images of the same scene. The basic premise to increase the spatial resolution[1] in SR techniques is the availability of multiple LR images captured from the same scene. Super resolution provides an inexpensive software solution to overcome the inherent limitations of image acquisition hardware systems, and has been a very active research topic over the past decades. Super resolution from multiple frames is possible only when there is some displacement between two input images (see Figure 2.1).

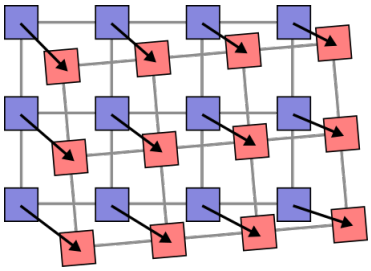


Figure 2.1: Direction of the subpixel motion between two images.

In some applications, the relative motion from frame to frame is known beforehand (or at least approximately), but the motion usually must be estimated from the data as a preprocessing step: each frame must be aligned with respect to some reference (usually, the first frame) with subpixel accuracy. This preprocessing step is called image registration [120], as shown in Figure 2.2.

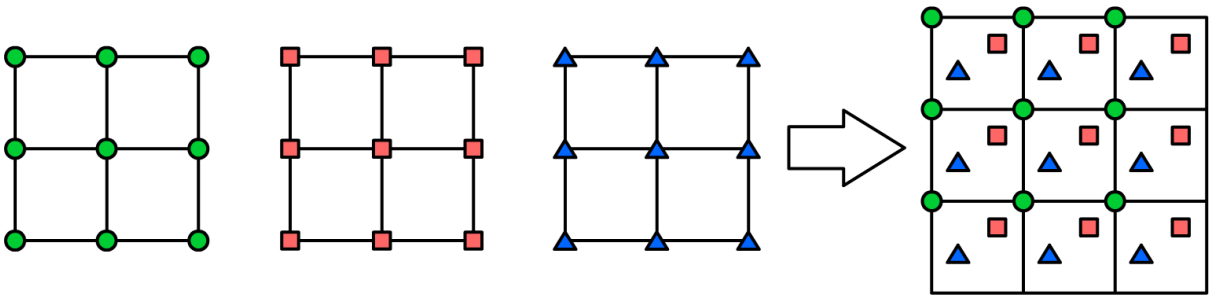


Figure 2.2: Basic idea for super resolution from multiple LR frames. Subpixel motion provides the complementary information among the LR frames, that makes the SR reconstruction possible. Original figure in [84].

If there are several LR images available with subpixel displacements, then the high frequency information of the super-resolution image can be increased. In this case, the new information contained in each LR image can be exploited to obtain an HR image. Image registration and super resolution are often treated as distinct processes, to be considered sequentially, but accurate subpixel motion estimation is a very important factor in the success of the SR image reconstruction algorithm [106, 107]. Artifacts caused by these registration errors can be visually more annoying than the blurring effect resulting from

[1] In this work, resolution is defined as a measure of the amount of detail that is visible in an image [20].

interpolation of a single image. According to [80], in order to increase the resolution by a factor of n, image registration must be accurate to within $\frac{1}{n}$ pixel. Super-resolution results from misregistered and misaligned frames are visually unsatisfactory.

The second step, after the registration, is the image reconstruction, when the aligned LR samples are combined to produce an image with more visible details. The main question in this step is how to combined the LR frames in order to form an HR image.

Multi-frame super resolution has been studied since 1984, when Tsai and Huang [118] pioneered the field and introduced an algorithm in the *frequency domain* 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, as they accept only a global displacement between images.

The majority of SR algorithms have since been developed in the *spatial domain* [77]. 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 [115]. 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 [45, 46] 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. This can be dealt with by incorporating a priori knowledge about the solution [77], and then minimizing the error using regularization, as has been done in [36]. Zomet et al. [138] proposed the Robust Super Resolution, other version of the IBP algorithm using the median rather than the mean to calculate each new pixel. Papoulis [83] and Gerchberg [41], 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. [121] 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) [109, 34]. 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 suffer from high computational costs.

In addition, the Maximum a Posteriori (MAP) methods [21] also add some a-priori knowledge about the desired HR image, and find an estimate for the solution using Bayes's rules [77]. As 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 [77].

Another trend in the spatial domain that is currently exploited in the literature comprises the Direct Methods [24, 25]. 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 [24], Adaboost classifier [104], SVD-based filters [76], and adaptive normalized averaging [86]. Direct algorithms have been shown to be faster than the IBP algorithms [77]. All super-resolution methods we propose in this work fall in this category. The first one recovers the HR pixels as an inverse problem, and the others are fusion-based (see Chap. 3).

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 (SISR), has become a hot research topic since it was first proposed by Freeman et al. in [38]. These approaches effectively "hallucinate" missing details based on similarities between the LR image and the examples in the training set [112]. Romano et al. [92] recently proposed a single-image algorithm that uses machine learning and train on pairs of images (one low quality, one high quality) 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 [30, 31], the authors present a learning-based SR method that uses deep convolutional neural networks (SRCNN) to learn an end-to-end mapping between low and high-resolution images. Such method requires large high-resolution and high-quality images for data training.

Although promising, SISR methods do not take advantage of the multiple information from the pool of frames that surveillance videos might comprise, justifying the importance of multi-frame super-resolution methods still today. In 2015, the work in [103] introduced a MFSR algorithm for shifted and rotated images that calculates the noises that arise in real-world applications, such as time-of-flight camera depth images, and uses Inpaiting to correct such noise. In [131], each HR pixel p is projected onto the LR images, and the authors select LR pixels which fall within the zone of influence of p. The HR image is recovered after minimizing a Maximum a posteriori Markov Random Field (MAP-MRF) energy function, by approximating their energy function to make it graph representable, and minimize it with a graph cut algorithm. The experimental results are good when using only rigid transformations and no real-world images. In [40], the authors propose an algorithm for multi-frame SR with two new types of regularization terms, termed as local weighted anisotropy regularization and successive regularization toward iteration process, which are characterized by the capability of suppressing noise and preserving edge information in HR image reconstruction. The multi-frame SR algorithm proposed in [126] relies on regularization, and the objective functional to be minimized consists of a fidelity term and a regularization term. The fidelity term is formed by combining L1 norm and L2 norm. The regularization term is proposed to preserve edge and flat regions.

Also in 2015, Maiseli et al. [50] developed a super-resolution framework that integrates an adaptive diffusion-based regularizer. The regularizing kernel incorporates a shape-defining parameter that can be automatically updated to ensure convexity and stability of

the corresponding energy functional. The SR approach proposed in [74] tries to preserve important image features (sharp edges and corners) while avoiding artifacts, using the framework proposed in [50]. The work in [44] reformulates the MFSR into a problem of multi-frame blind deblurring. Unfortunately, in the experiments, they show only the results for super resolving one synthetic and one real-world image. In [51], the authors propose an algorithm underpinned by a dictionary based on local self-similarity and the directional similarity. The method removes the interpolation artifacts using the patch pairs based on the image degraded model. Only two results are visually compared with other SR techniques. In [71], the authors provide understanding on the relationship between the restoration error and motion blur, and conclude that more images, even degraded by blur, could induce better SR results.

In 2016, Köhler et al. [60] proposed an Iteratively Re-Weighted optimization for Robust Super Resolution (IRW-SR). They use a weighted Gaussian observation model to consider space variant noise and weighted bilateral total variation to exploit sparsity of natural images. The algorithm is implemented as iteratively re-weighted minimization, simultaneously estimating model parameters and the super-resolved image in an iterative coarse-to-fine scheme. The method in [62] consists of a non-parametric image registration based on diffusion regularization and a nonlocal Laplace regularizer combined with a bilateral filter in the reconstruction step to remove noise and motion outliers. The diffusion registration is employed to handle the small deformation between the unregistered images, while the combination of nonlocal Laplace and BTV is used to increase the robustness of the restoration step with respect to the blurring effect and to the noise. In 2017, the work in [55] proposed to solve the registration and reconstruction problems in a unified framework, but considers only shifts between LR observations.

Finally, we address recent work for super resolution of depth images. The work in [47] proposes a new multi-frame method to enhance LR dynamic depth videos containing freely non–rigidly moving objects. The work in [119] extends the SRCNN framework [30] to enhance the depth image and obtain a high-resolution depth image using deep convolutional neural networks. Depth videos are also the focus of the work in [48]. They enhance depth videos containing non-rigidly deforming objects, relying on the assumption that 3D motion can be decoupled into lateral motions and radial displacements. This allows them to perform a simple local per-pixel tracking in which both depth measurements and deformations are dynamically optimized.

2.3 Original contributions of the proposed SR methods

In Chap. 3, we propose three novel multi-frame super-resolution methods. The first one relies on Tikhonov regularization [116] and adaptive filtering to solve a least-square problem. According to [77], regularization has already been used along with different super-resolution methods, such as iterative methods, direct methods, POCS and MAP methods. The IBP methods (see Sec. 2.2) aim to minimize the solution of the least-squares by defining an initial guess for the HR target image, and then iteratively refine the resultant image. In [137], for example, the authors use conjugate gradient to speedup

the solution of the IBP method, and combine super resolution to mosaicing. The MAP approach (see Sec. 2.2) models a priori knowledge to constrain the solution, usually using Bayesian methods. The work in [67] uses conjugate gradient in a MAP-based method. According to [133], Tikhonov regularization is the most commonly used method for the regularization of ill-condition problems. The works in [27] and [16] have also considered the use of conjugate gradient for solving the SR based on Tikhonov regularization, accelerating the methods based on the system of linear equations. In [79], the authors added preconditioners for solving the Tikhonov-regularized super-resolution problem by the conjugate gradient method, using the generalized cross-validation method for automatic calculation of regularization parameters.

Most of these works based on regularization accept only shifted LR images as input, i.e., there might be only translations among the observations. The regularization-based model that we present herein works with both **rigid** and **perspective** transformations. We also contribute with two optimizations of the method to create the system of linear equations faster than the original model, by dividing the grid into imaginary cells, and discarding less significant information. Moreover, the proposed method uses adaptive filtering to convolve the HR pixels (our LR observations can belong to different planes, and the Gaussian weights are distributed along each plane). Even the filter in one of the optimizations (that does not consider the Gaussian distribution), is space-adaptive (since the imaginary regions might contain more or less LR pixels according to each plane).

The second proposed method projects the LR pixels separately onto the HR grid, and then uses inpainting techniques [114, 7] to fill in the unknown grid pixels. Inpainting has already been used to fill in missing pixels of super-resolved images, as in [73]. 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 for which we might have multiple observations of the same scene [19]. In [17], the authors use inpainting to fill in missing pixels in the observed shifted LR images (i.e., with only translations between the input images), and then they use Tikhonov regularization to solve the optimization problem. In [103], the authors introduce a MAP-based algorithm for shifted and rotated images that calculates the noise artifacts that arise in real-world applications, such as time-of-flight camera depth images, and uses inpainting to correct such noise. Our approach, which projects the observations onto the HR grid, and then fills in the unknown pixels using inpainting techniques, is part of a work that has been recently published [97].

Finally, the third proposed method explores geometric neighborhood, to combine LR pixels into an HR grid. It is a Direct Method, and its main idea has commonly been used to describe a multi-frame super resolution. Other fusion-based methods have been previously proposed. In [24], the pixels are combined with median filter. However, the images are warped to the reference image (which could cause loss of information) and there is no quantitative validation. In [134], the authors combine the LR images using pyramids fusion [14]. According to the authors, their method is a faster alternative to methods based on solving the linear system, but there is no guarantee that a perfect reconstruction can be obtained, and some artifacts may appear in the super-resolved image. The results are only visual, and not very impressive. Our work draws a set of variations of the method that exploits geometric neighborhood around the desired pixels,

as simple as possible, to be used in the applications that require high performance and low-memory footprint. The five variations of the method have been published in [96] as an original conference paper. In a further development of our work, in [108], the authors use the method (proposed in [96]) for super resolution of optical coherence tomography.

2.4 Super resolution of license plates in surveillance videos

Automatic License-Plate Recognition (ALPR), also known as Automatic Number-Plate Recognition (ANPR), uses Optical Character Recognition (OCR) on images to extract and recognize the alphanumerics of a vehicle registration plate [32, 3]. An ALPR system usually follows some steps for identifying a license plate: plate localization (to find and isolate the plate on the picture); plate orientation (to compensate the skew of the plate and adjust its dimensions); normalization (to adjust the brightness and contrast of the image); character segmentation (to find the individual characters on the plate); and the optical character recognition. Optionally, the system may check the characters and positions against country-specific rules to produce a more reliable or confident result.

License plate recognition may be especially difficult under poor image resolution (usually when the car is too far away from the camera, under adverse atmospheric conditions, or due to a low-quality acquisition camera); blurry images (due to the motion blur) [18]; poor lighting; low contrast (due to overexposure, reflection or shadows); and different fonts (in some countries). Many ALPR systems that claim good recognition when they are trained to match license-plates from a single region, fail when trying to recognize license-plates from other regions due to variations in format, font, color, layout, and other plate features.

The license-plate capture is usually performed by specialized cameras, designed specifically for such task. However, there are a number of low-quality surveillance cameras scattered throughout our cities that could help to identify a suspect vehicle, for example, in a crime scene. In such scenarios with poor quality cameras, super resolution may be used to help recognizing the characters in the license plate. We focus on super resolving those images for which the specialists could not visually identify the characters.

Caner et al. [15] seem to have pioneered the alliance of super resolution and automatic license-plate recognition in 2003. They super resolved a region of interest of surveillance videos recorded by multiple cameras based on POCS (see Sec. 2.2). Although promising, the solution needed more than one camera to work. Chang et al. [18] 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 without the aid of any super-resolution method.

In 2007, Suresh et al. [110] performed SR of moving vehicles in real-world traffic videos by combining 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 have not considered rotations in the license plates between consecutive frames, and the results were only qualitatively compared.

Yuan et al. [128] presented, in 2008, 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 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 results. They only compared the running time of reconstructing two HR images.

Kim and Ko [58] proposed a resolution enhancement method for regions of interest in surveillance videos using Bernstein interpolation [61] in 2011. 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. [127] proposed, in 2012, an SR method 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.

Using an algebraic reconstruction method, Zarei et al. (2013) [130] 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) [53]. 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 [66] proposed, in 2014, 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.

Finally, in 2015, the work in [57] claims that metropolis worldwide invest huge sums of money in surveillance camera systems but few are closely observing the benefits, because the low resolution coupled with poor-quality optics is not enough to identify the subject of interest in crowded and far-away setups, in bad weather and other limiting factor. They introduced a multi-frame super-resolution technique that does not require explicit motion estimation, and that produces evidence that the police might reasonably accept as proof of someone's identity. Their algorithm requires a training set from a still surveillance camera.

According to a recent work of Rajput et al. (2016) [90], few researchers have addressed

scenarios such as reading plates of fast-moving vehicles. They also claimed that the existing ALPR approaches assume 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. (2008) [113] and Qing et al. (2007) [89] also addressed the problem of rectification of license plates to correct their inclined distortion.

2.5 Super resolution for mobile devices

Super resolution for mobile devices was studied by Kanumuri et al. [54] in 2007. They described an SR reconstruction of mobile videos using warped transforms and adaptive thresholding. However, for complexity and memory bandwidth reasons, they restrict themselves to operate on single frames. This way, at the end, they solve just an interpolation problem.

Shen and Xue [99] proposed in 2010 an example-based SR algorithm to enhance videos from mobile devices. They acquire the first frame of a video at full-resolution and the other frames at low-resolution, consuming less power of the mobile device. Then, the video is post-processed, using information from the first frame at full-resolution to enhance the other frames.

The objective in [26] is more similar to ours: with an algorithm to reconstruct HR images for mobile devices from a sequence of LR frames. In that work, however, frames must be acquired in a video with a high frame rate (e.g., 100 fps), so they can estimate changes in the relative motion between each LR frame during the registration step.

Amanatiadis et al. [2] proposed an SR algorithm for mobile devices based on machine learning, using only one image as input. Their method needs GPU as a co-processor to achieve good performance. The results seem visually good, but the LR images in the experiments are only downscaled versions of HR images. The authors have not considered real-world low-resolution images in their validation, however.

2.6 Final considerations

Many single-image super-resolution methods have been studied in recent years. In many cases, applying such methods frame by frame might be better than using multi-frame super resolution. However, every information added into the super-resolved image by a single-image SR method comes from the knowledge learned from the training process, using other images. If a forensic analyst super resolves, for example, the image of a suspect license plate using a learning-based method, such method might add information from other sources into the image of interest, and then the evidence might be invalidated by a court decision. On the other hand, the image super resolved by a multi-frame super-resolution method might contain **only** information that appears in the object of interest, along its moving path in the video. Additionally, in [77], the author highlights that the state of the art for super-resolution algorithms is highly dependent on the application.

This is mainly due to different constraints that are imposed on the problem under different setups. Hence, our intuition is that the multi-frame SR methods can be even more useful in a forensic setup.

Chapter 3

Proposed methods for super resolution

In this section, we propose three methods to combine multiple observations of a scene into a higher resolution image. First, in Sec. 3.1, we introduce a method that relies on Tikhonov regularization and adaptive filtering to solve a least-square problem. We refer to this method as Super Resolution using Regularized Least Squares (RLS). Then, Section 3.2 details the inpainting-based Super Resolution (ISR), a method that projects the LR pixels separately onto the HR grid, and then uses inpainting techniques to fill in the unknown grid pixels. In Sec. 3.3, we explain the Geometric k-Nearest Neighbors Multi-Frame Super-Resolution (GSR), a method which explores geometric neighborhood, to combine LR pixels into an HR grid.

In addition, we exploit two optimizations for the RLS model, two different variations for the inpainting-based methods, and five for GSR, in Secs. 3.1 through 3.3. Fig. 3.1 depicts such variations. In red, we have the group of algorithms using Regularized Least Squares; in blue, the variations of ISR; in green, five different algorithms based on GSR.

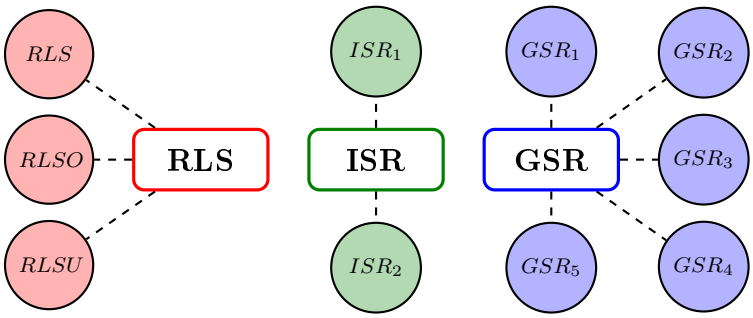


Figure 3.1: Variations of the proposed super-resolution methods.

The common pipeline for all proposed methods is shown in Fig. 2.2: the input images are aligned in a registration step, and then each method reconstructs the output super-resolved image.

3.1 Super resolution using Tikhonov regularization and adaptive Gaussian filtering (RLS)

In a multi-frame super resolution, we start by hypothesizing that a set of LR images 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. The decimated image f can be expressed [111] as a function of the original image g:

$$f(x, y) = \sum_{i,j} g(i, j)h(mx - i, my - j),$$

for each color channel separately, where $m \in \mathbb{Z}$ is the downsampling factor, and h is a low-pass filter, typically used prior to downsampling in order to prevent occurrence of aliasing. Hence, each decimated image is a result of a convolution equation [111] of the form

$$h * X = f,$$

where f and h (the filter applied to the HR image, including possible bluring, downsampling, and other operators) are known vectors, X is the vector to be determined (related to the entire HR image), and $*$ is the convolution operator. The set of LR input images then forms a system of linear equations that might be used to determine the target HR image completely.

Super resolution is then an inverse decimation problem. Since the number of LR images is usually insufficient to solve the system, we have a linear least-squares problem. There are a number of techniques to deal with such problems, and most of them have high computational cost [91]. In addition, when authors address super resolution by solving a linear least-squares problem, they usually restrict the motion between a pair of LR images to simple translations. However, even if we also include rotations between the images, we might still not attend real-world problems, because, commonly, a pair of images of a scene may differ from each other by a perspective transformation (besides additional issues with illumination, shadows, etc.).

We consider herein a setup in which we have a sequence of $n \in \mathbb{Z}$ images $I_k, \forall k \in \{1, 2, \ldots, n\}$. Each I_k is a smaller version of the target image I_{HR} that we are trying to reconstruct. In Fig. 3.2, the black dots are the pixels of a target HR image, and the red squares represent three examples of LR images of the same scene. Note that an LR pixel does not necessarily coincide with HR pixels if we consider a continuous space.

We assume that, during the images acquisition process, there exists a continuous space of information that might be discretized to represent I_k and I_{HR}. We also assume that each LR pixel was generated as a convolution of the information in its neighborhood, and that a same low-pass filter has been used to generate all I_k.

We consider we have calculated the transformation matrices Υ_k mapping each image I_k onto the HR grid that we want to reconstruct. Such matrices can be calculated, for example, in a registration step, prior to the reconstruction. Therefore, we can calcu-

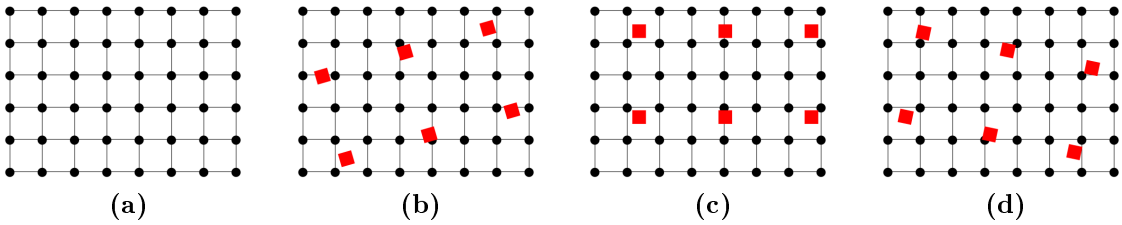


Figure 3.2: Three LR images generated from a scene, and the HR grid that we want to reconstruct. In (a), the HR target pixels. In (b), (c) and (d), the red squares are the pixels in each LR image I_k. The LR images have been aligned with respect to the grid in (a).

late exactly the alignment of each LR image with respect to the HR grid, as we see in Figs. 3.2b, 3.2c and 3.2d. If the matrices dimension is 2×3, then $\Upsilon_k \in \mathbb{R}^{2\times 3}$ and we have rigid transformations (comprising only translations and rotations). Otherwise, matrices $\Upsilon_k \in \mathbb{R}^{3\times 3}$ define a projective transformation, also known as a perspective transform or homography. The algorithm we present herein works with both rigid and perspective transformations. We can use such matrices to calculate exactly the xy position of any point in I_k with respect to the HR grid. For rigid transformations [111], we have

$$(x', y') = \Upsilon_k \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.1}$$

where $x', y' \in \mathbb{R}$ are the new positions of I_k transformed with the matrix Υ_k. For perspective transformations, we must operate on homogeneous coordinates [111]:

$$(\tilde{x}', \tilde{y}', \tilde{w}) = \Upsilon_k \begin{bmatrix} x \\ y \\ 1 \end{bmatrix},$$

where $\tilde{x}', \tilde{y}'$, and $\tilde{w} \in \mathbb{R}$ are the homogeneous coordinates that must be normalized in order to obtain an inhomogeneous point (x', y') by dividing each element by the last element $\tilde{w}$, i.e.,

$$(\tilde{x}', \tilde{y}', \tilde{w}) = \tilde{w}(x', y', 1) = \tilde{w}(x', y'). \tag{3.2}$$

Such matrices are calculated in a registration step, prior to the reconstruction, that might be different for each application of the algorithm. We describe some registration examples in Chaps. 4 (for working with a mobile application) and 5 (for automatic license-plate recognition).

As we calculate the xy position of the LR points in each I_k with respect to the HR grid, we might choose a linear filter and write each pixel $I_k(x, y)$ as a convolution among the HR pixels in the grid using such filter. We take here a Gaussian function as example for the linear filtering. The one-dimension Gaussian filter G has an impulse response given by

$$G(x;\sigma) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{x^2}{2\sigma^2}},$$

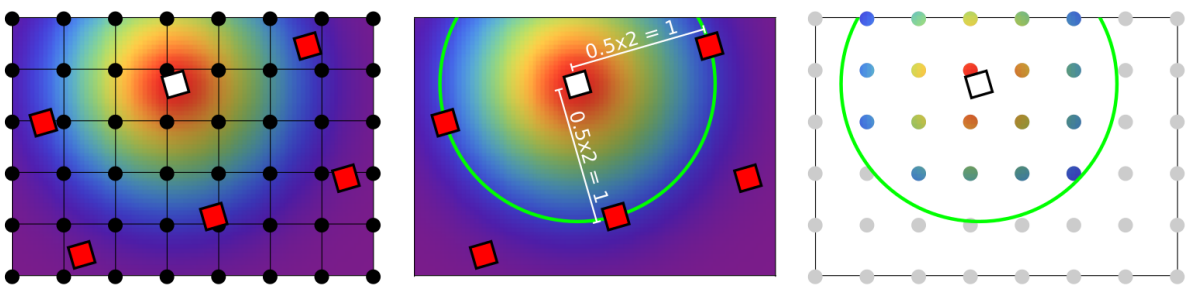
where $\sigma \in \mathbb{R}$ is the standard deviation of the Gaussian distribution [111]. In two dimensions, it is the product of two Gaussians, one per direction:

$$G(x,y;\sigma) = \frac{1}{2\pi\sigma^2}\cdot e^{-\frac{x^2+y^2}{2\sigma^2}},$$

where $x, y, \sigma \in \mathbb{R}$, x is the distance from the pixel to the center of the distribution in the horizontal axis, y is the distance in the vertical axis, and σ is the standard deviation of the Gaussian distribution [111]. The Gaussian function is for $x \in (-\infty, \infty)$. Hence, for every pixel $I_k(x,y)$ in an LR image, we can write it as:

$$I_k(x,y) = \sum_{i,j} I_{HR}(i,j)G(x'-i, y'-j, \sigma), \tag{3.3}$$

where $x', y' \in \mathbb{R}$, (x', y') are the positions of I_k mapped onto the HR grid with the matrix Υ_k, and σ is the standard deviation of the Gaussian distribution. Theoretically, we should create a linear system using Eq. 3.3 for all the pixels in all the LR images (the Gaussian function at every point on the image will be non-zero, meaning that the entire image would need to be included in the calculations for each pixel). In practice, when computing a discrete approximation of the Gaussian function, pixels at a distance of more than 3σ are small enough to be considered effectively zero. To illustrate such distance, consider Figure 3.3. The white square in Fig. 3.3a is an example of a pixel $I_k(x,y)$ in Eq. 3.3, and it is the center of a Gaussian distribution. The colors in the background represent the Gaussian values in a continuous space for $\sigma = 0.5$. In Fig. 3.3b, we define a circular region with a radius $r = 2\sigma$ around the LR pixel. Only the HR pixels inside such circular region might be convolved in this example. Finally, each HR pixel in Fig 3.3c receives the value of the Gaussian weights in its x, y position only if the pixel is within the circular region.



(a) (b) (c)

Figure 3.3: In (a), the LR pixel in the white square is the center of a Gaussian distribution. The weights of this distribution might be used to write such LR pixel as a function of the HR pixels in its neighborhood. In (b), we define a circular region with a radius $r = 2\sigma$ around the LR pixel. In (c), the grid pixel receives the values of the Gaussian weights only if the pixel is inside the circular region.

Hence, we rewrite the Gaussian filter as G':

$$G'(a,b;\sigma,r) = \begin{cases} \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{a^2+b^2}{2\sigma^2}}, & \text{if } \sqrt{a^2+b^2} \leq r \\ 0, & \text{otherwise.} \end{cases}$$

It is worth mentioning that this algorithm uses an adaptive filter to convolve the HR pixels, since we allow Υ_k to handle homographies. We see, in Fig. 3.4c, an example of an LR image I_k in a plane $\mathscr{P}_k$ that is projected onto the HR grid in a plane $\mathscr{P}$ by a perspective transformation. We want to write each LR pixel $I_k(x,y)$ as a convolution among the HR pixels using a Gaussian distribution. In Fig. 3.4d, $I_k(x,y)$ in the white square is the center of a Gaussian distribution over the plane $\mathscr{P}_k$. We see that two HR points in the plane $\mathscr{P}$ with the same distance to such white square will not necessarily have the same Gaussian weights. Additionally, we see, in Fig. 3.4e, that the Gaussian weights have a different distribution for another LR pixel $I_k(x',y')$.

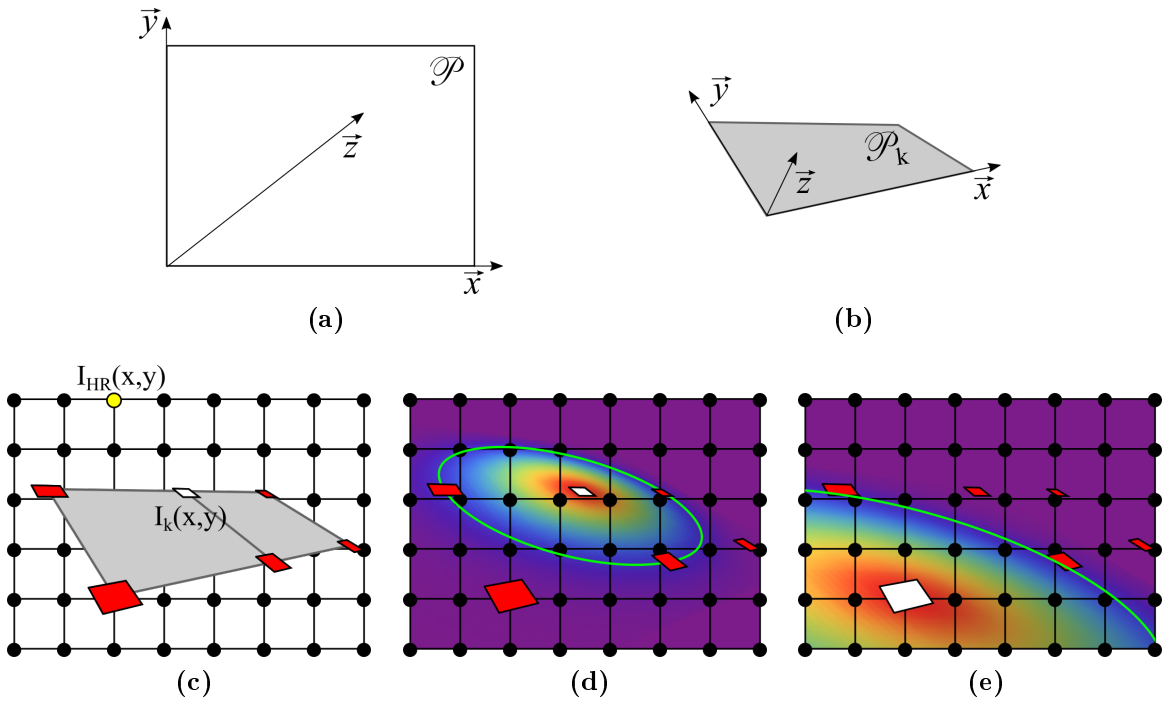


Figure 3.4: Adaptive distribution of the Gaussian filter. Two different planes $\mathscr{P}$ in (a) and $\mathscr{P}_k$ in (b). In (c), the LR image I_k belongs to the plane $\mathscr{P}_k$, and the HR grid belongs to the plane $\mathscr{P}$. In such example, the yellow dot is a pixel $I_{HR}(x,y)$ in the grid, the white square is a pixel $I_k(x,y)$ in the LR image, and Υ_k is a perspective transformation mapping I_k onto I_{HR}. In (d), $I_k(x,y)$ in the white square is the center of a Gaussian distribution over the plane $\mathscr{P}_k$. We see that two HR points with the same distance to $I_k(x_1,y_1)$ in the plane $\mathscr{P}$ will not necessarily have the same Gaussian weights. In (e), we see that the Gaussian weights have a different distribution for another LR pixel $I_k(x',y')$.

Finally, for n LR images I_k and an HR image I_{HR}, each equation of the linear system is given by

$$I_k(x,y) = \sum_{i,j} I_{HR}(i,j) \frac{G'(x'-i, y'-j; \sigma, r)}{W(x,y)}, \tag{3.4}$$

for each pixel in each LR image, where $x', y' \in \mathbb{R}$ are the positions of I_k mapped onto the HR grid with the matrix Υ_k, σ is the standard deviation of the Gaussian distribution, r is the radius of a circular region around (x, y), and $W(x, y)$ is the sum of Gaussian weights for all the points around (x, y) and inside the circular region of radius r. We calculate W to normalize the Gaussian weights that have been convolved with each LR point (x, y), so the sum of weights is one. W is given by

$$W(x, y) = \sum_{i,j} G'(x' - i, y' - j; \sigma, r), \tag{3.5}$$

For an RGB image, we must solve one linear system in Eq. 3.4 for each color channel separately. It is worth mentioning that, since we use G' instead of G, the linear system becomes sparser as the images dimensions increase.

After creating our linear system using Eq. 3.4, we might decide how to solve the system. The algorithm that we describe does not require neither a set of LR images with exactly the same dimensions, nor a fixed scale factor for all input images, since the transformation matrices Υ_k can map them onto the HR grid, no matter if they have different sizes or scales. However, suppose now without loss of generality, that we have n low-resolution images, and each one has exactly p pixels. Hence, we create p equations for each LR image using Eq. 3.4, and we have a linear system with np observations (in other words, each pixel from each LR image generates one equation of the linear system). Additionally, assume we super resolve them with a scale factor s in each axis. Therefore, I_{HR} has s^2p pixels, and we have a system with s^2p variables. If we write our system as

$$A\vec{x} = \vec{b},$$

as in [4], then $A \in \mathbb{R}^{(np)\times(s^2p)}$, $\vec{x} \in \mathbb{R}^{s^2p}$, and $\vec{b} \in \mathbb{R}^{np}$. Each unknown $x_i \in \vec{x}$ can be seen as an available degree of freedom in the system, and each equation $A_i \in A$ introduced into the system can be viewed as a constraint that restricts one degree of freedom. We might have a unique solution to the system when the number of equations and the number of free variables are equal (in our case, when $n = s^2$). For example, we need exactly $5^2 = 25$ LR images to have a linear system with a unique solution if we want a super-resolved image $5\times$ higher in each axis. If we have fewer equations than unknowns ($n < s^2$), the system is considered underdetermined. In this case, there are either infinitely many or no solutions. Otherwise, the system is overdetermined if there are more equations than unknowns (i.e., $n > s^2$). Usually, overdetermined systems have an infinite number of solutions. Since we do not want to constraint our system to having exactly $n = s^2$, we must be able to solve both underdetermined and overdetermined systems.

An ordinary least squares [63] tries to minimize the sum of squared residuals, which can be written as

$$\min_{\vec{x}} ||A\vec{x} - \vec{b}||_2^{\,2}, \tag{3.6}$$

where $|| \cdot ||_2$ is the Euclidean norm. Tikhonov regularization [116], also known as ridge regression, addresses the situation when the system of equation is ill-posed and admits infinitely many or no solutions, and seeks to determine an approximate solution by replacing the minimization problem in Equation 3.6 by a penalized least-squares problem

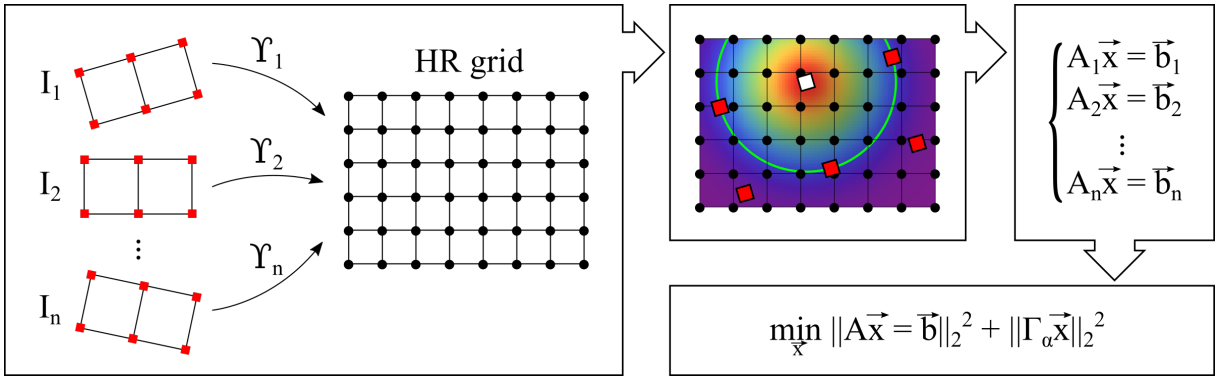


Figure 3.5: Pipeline of the proposed super-resolution algorithm using Tikhonov regularization (RLS): First, we have n LR images I_k, and the matrices Υ_k mapping each I_k onto the HR grid. Then, we use a modified Gaussian filter G' inside a circular region to write each pixel $I_k(x, y)$ as a convolution among the HR pixels that we want to reconstruct. In the third step, the coefficient matrices A_k and the vector $\vec{b}_k$ (created for each LR image) are concatenated into a system of linear equations. Finally, we use Tikhonov regularization to calculate an approximation to $A\vec{x} = \vec{b}$.

of the form

$$\min_{\vec{x}} ||A\vec{x} - \vec{b}||_2{}^2 + ||\Gamma_\alpha \vec{x}||_2{}^2, \tag{3.7}$$

for some suitable regularization matrix Γ_α [39]. The scalar $\alpha > 0$ is known as the regularization parameter, and an explicit solution is given by

$$\hat{x} = (A^\top A + \Gamma_\alpha{}^\top \Gamma_\alpha)^{-1} A^\top \vec{b}, \tag{3.8}$$

where $\hat{x} = V^\top \vec{x}$, and V is given by the singular value decomposition $A = U\Sigma V^\top$ [39]. The matrix Γ_α is commonly chosen to be a multiple of the identity matrix ($\Gamma_\alpha = \alpha I$), in order to give preference to solutions with smaller norms, and Eq. 3.8 can be solved by Conjugate Gradient (more advantageous to solve large and sparse systems) [27, 133]. We use a Tikhonov regularization separately for the system in each color channel of the image in the algorithm that we describe herein. In addition, we also normalize[1] the intensity values of the super-resolved image. The regularization finds one among the many solutions for the linear system. The normalization corrects the spatial variance of the solution, stretching the resultant intensity values to the same range as the input sequence.

The algorithm we propose herein is illustrated in Fig. 3.5, and its parameters are:

- The pool of n LR images I_k.
- The coordinates of the HR grid that we might reconstruct.

[1] To normalize the result, we calculate the minimum p_{min} and maximum p_{max} intensity values among all the LR images in the pool. Then, we apply a linear mapping to each color band of the super-resolved result (using addition and multiplication operators), and we stretch the intensity values to fit within the dynamic range $[p_{min} \cdot\cdot p_{max}]$.

- The aligning matrices Υ_k, previously calculated in the registration step, mapping each I_k onto the HR grid.
- The standard deviation σ of the modified Gaussian filter G', and an integer d such that the radius $r = d \times \sigma$ defines the circular region around each pixel $I_k(x, y)$ (i.e., the kernel size).

After the pre-alignment in the registration step, we perform the following steps:

1. First, for each LR image I_k, we create the matrix A_k and the vector $\vec{b}_k$ for such image (respectively, the coefficients and the constant terms of the system of equations). To calculate the coefficients, we use the modified Gaussian filter G' with a standard deviation σ, within a circular region of radius $r = d \times \sigma$, and adapted to the plane in which I_k belongs.
2. Then, we concatenate the n coefficient matrices A_k into a single matrix A, and the constant terms $\vec{b}_k$ into a vector $\vec{b}$.
3. To calculate an approximation for $A\vec{x} = \vec{b}$, we use Tikhonov regularization for each color band of the images.
4. Finally, we normalize each band separately, creating a super-resolved image using the same intensity range as in the pool of LR images.

We designed three variations of this algorithm. The first one implements exactly the original model, illustrated in Fig. 3.5. We refer to this first variation as *RLS* (Super Resolution using Regularized Least Squares). The others are optimizations of the model, and they are intended to run faster, but might impact the quality of the results. We detail the two optimizations in Secs. 3.1.1 and 3.1.2.

3.1.1 Optimized RLS (RLSO)

In this second variation of *RLS*, we seek to decrease the execution time for constructing the matrices A_k. In the original model, when we use Eq. 3.4 to calculate the linear system for an LR image I_k, a grid pixel $I_{HR}(x, y)$ can be used in the equations of more than one pixel $I_k(x, y)$. Fig. 3.6 illustrates such overlaps.

For each LR pixel $I_k(x, y)$, we must iterate over all grid pixels $I_{HR}(x, y)$ to calculate such Gaussian weights in the original *RLS*. For n LR images I_k, each of which containing p_k pixels, and a grid with p pixels to reconstruct, we calculate all these weights in $\mathcal{O}(n \times p_k \times p)$. Even if we perform the inner loop (over the grid pixels) in a smaller region (since we consider only the kernel weights inside a circular region o radius $r = d \times \sigma$), we still have these nested loops, because an LR pixel might contribute to more than one pixel in the grid.

Our target in this first optimization is to eliminate these two nested iterations. To achieve this, each grid pixel must not appear in more than one equation from the same LR image I_k. Hence, we divide the grid into imaginary regions of interest or cells, in a

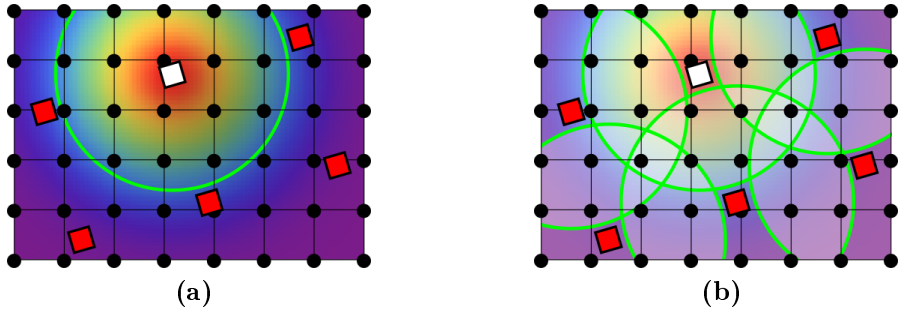


Figure 3.6: In the original *RLS* model, there might be some overlap during the calculation of the Gaussian weights. In (a), we want to calculate the weights around the LR pixel highlighted in white. In (b), each grid pixel may be inside the circular region around more than one LR pixel, and might appear in more than one equation from the same image I_k.

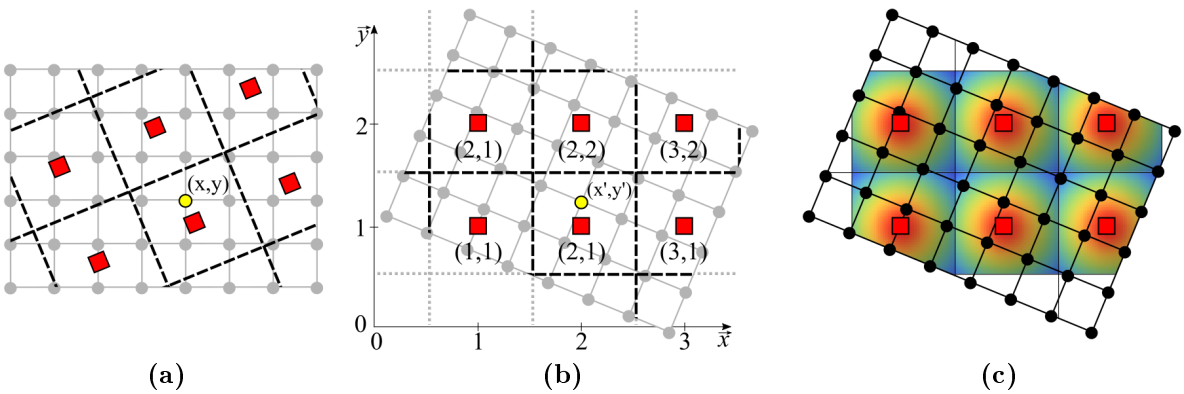


Figure 3.7: In (a), we divide the HR grid into imaginary cells, in a way that each grid pixel is related to its closest neighbor in the LR image. In (b), we project the grid onto the LR image. The grid pixel (x, y), highlighted in yellow in (a), has a position (x', y') with respect to the grid (calculated by Eqs. 3.1 and 3.2). The coordinates of the LR pixels in (b) are $(1, 1)$, $(2, 1)$, $(3, 1)$, $(1, 2)$, $(2, 2)$, and $(3, 2)$, and we may infer by this Figure that $1.5 < x' < 2.5$ and $0.5 < y' < 1.5$. We simply round the coordinates (x', y') of this point in yellow to find its closest neighbor in the LR image, i.e., the pixel $(2, 1)$. In (c), the Gaussian distribution is calculated only for the pixels inside each imaginary rectangle.

way that each grid pixel $I_{HR}(x, y)$ is related to its closest LR pixel in I_k. Fig. 3.7a depicts such divisions.

As we see in Fig. 3.7b, if we project the grid pixels onto the LR image I_k, we can easily find the closest LR pixel to each grid pixel $I_{HR}(x, y)$. We only need to iterate over the grid pixels, rounding each of them. Hence, in this same iteration, we can calculate the distance from $I_{HR}(x, y)$ to its closest LR pixel, and also the Gaussian weight for this grid pixel. The complexity for calculating all the Gaussian distributions, and consequently all the equations of the linear system, decreases from $\mathcal{O}(n \times p_k \times p)$ in *RLS* to $\mathcal{O}(n \times p)$ in this optimization, where n is the number of LR images, p_k is the number of pixels in each LR image, and p is the number of pixels in the HR grid.

The matrix A_k is sparse, and each element of A_k is defined by three values: (1) the

entry of the matrix, (2) the row index of such entry, and (3) its column index. In this implementation of *RLS*, the entries of the sparse matrix A_k are the non-zero Gaussian weights that we calculate for each grid pixel. The row index for each entry identifies its closest LR pixel, and can be easily calculated using the (x, y) coordinates of such closest LR pixel. The column numbers are simply the grid indices, between 1 and p.

We show in Chap. 4 how such optimization impact both the running time (since we eliminate the nested loops for each LR image) and the quality of results (since we calculate now the Gaussian distribution only for the neighbors inside the imaginary rectangles in Fig. 3.7, instead of the circular regions in Fig. 3.6). The main difference from this implementation to the original model is that, now, we suppose that each HR pixel has been used in the convolution of only one LR pixel (see Fig. 3.8). We refer to this algorithm as *RLSO* (*RLS* with Optimization). Unlike the *RLS* model, this optimization does not receive the kernel size as parameter.

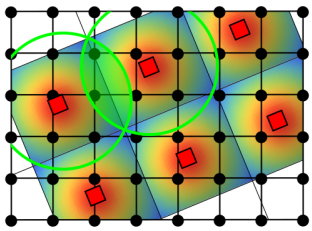


Figure 3.8: In the optimized RLS, we suppose that each HR pixel has been used in the convolution of only one LR pixel. In the original *RLS* implementation, an HR pixel can be inside the circular regions around more than one LR pixel.

3.1.2 RLS using an Uniform Distribution (RLSU)

The third implementation of *RLS* is even faster than *RLSO*. Instead of working with the Gaussian distribution, we use a simple uniform distribution along the HR pixels inside the same rectangular region as in RLSO. Therefore, we assume that each LR pixel $I_k(x, y)$ has been convolved as a simple average among all grid pixels to which $I_k(x, y)$ is the nearest neighbor in the LR image. In other words, we still use the imaginary cells from *RLSO*, but without calculating the Gaussian distribution. Since we do not calculate the exponentiation and division operations of the Gaussian distribution, we expect a lower runtime in this algorithm, that we refer to as *RLSU* (*RLS* using Uniform distribution). Unlike the RLSO method, *RLSU* does not receive the standard deviation σ as parameter.

3.2 Projections onto the HR grid and inpainting to fill in unknown pixels

The main objective of the *RLS* algorithm, described in Section 3.1, is to combine multiple low-resolution observations of a scene into a higher resolution image, richer in details.

However, even with the two optimizations proposed in Secs. 3.1.1 and 3.1.2, such algorithm might have a high computational cost to construct the coefficient matrices and to perform the regularization, as we show in our experiments (see Chapter 4). We introduce now an algorithm that is designed to be simpler and faster than the super resolution using regularized least squares. The algorithm receives three parameters as input:

- The pool of n LR images I_k.
- The coordinates of the HR grid that we might reconstruct (the grid can be based, for example, on the coordinates of the first input image and the increasing scale).
- The aligning matrices Υ_k, previously calculated in the registration step, mapping each I_k onto the HR grid.

We start projecting each LR image I_k, separately, onto the HR grid. Hence, for each pixel $I_k(x, y)$, we calculate the positions (x', y') of the LR image with respect to the grid, as in Eqs. 3.1 and 3.2. The algorithm is illustrated in Fig. 3.9. In Fig. 3.9a, the first LR image is projected onto the grid. The intensity value of each LR pixel $I_k(x, y)$ contributes only to the grid pixel that is closest to (x', y'). In Fig. 3.9b, we project a second image onto the grid, and the red dots are HR pixels that have already been calculated using the information from the first LR image. Similarly, in Fig. 3.9c, a third image 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 LR image. We use inpainting [114, 7] techniques to calculate such unknown grid pixels (the gray dots in Fig. 3.9d).

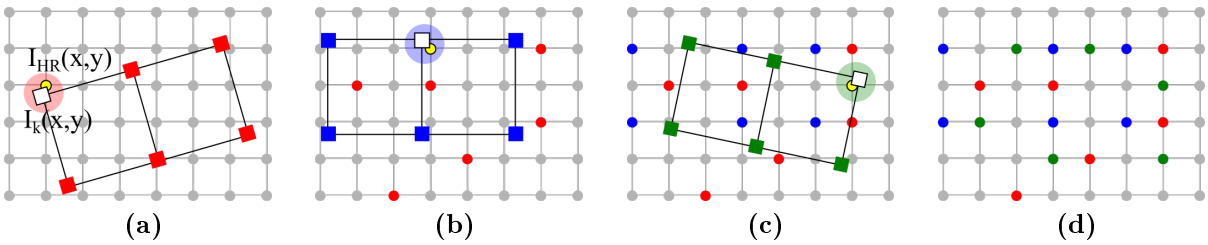
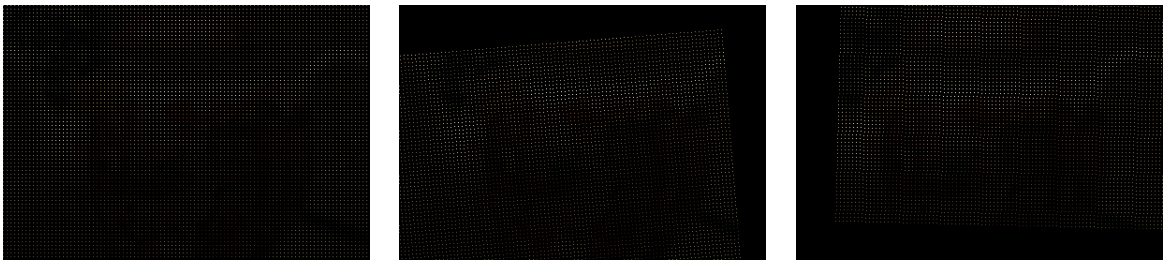


Figure 3.9: In (a), the first LR image is projected onto the HR grid. The red circle highlights an LR pixel $I_k(x, y)$ (the white square), and its closest pixel $I_{HR}(x, y)$ in the grid (the yellow dot). The intensity value of $I_k(x, y)$ contributes only to such closest grid neighbor, and then this $I_{HR}(x, y)$ receives the intensity value of $I_k(x, y)$. In (b), the grid contains information from the first LR image (the red dots), and another image is projected onto the grid. Similarly, in (c) a third image is projected onto the grid, which already contains information from the two previous LR images. Finally, the gray dots in (d) are the missing pixels of the grid, whose values we want to calculate using inpainting techniques.

To detail the first steps of this algorithm, we select 35 LR images of a scene (the first seven images are shown in Fig. 3.10) to be input for the inpainting-based Super Resolution.

Figure 3.10: Different LR observations of a scene, that we select to be input for the inpainting-based Super-Resolution method we propose.

The aligning matrices Υ_k allow us to map each I_k onto the HR grid that we might reconstruct. Fig. 3.11 illustrates three LR images projected, separately, onto an empty HR grid. The black pixels represent missing values, onto which no LR pixel has been projected.



(a) **(b)** **(c)**

Figure 3.11: The three LR images are separately mapped onto an empty HR grid. The black pixels represent missing values, onto which no LR pixel has been projected.

The algorithm iteratively projects the images I_k onto the grid, which might already contain information from the previous images I_j, $1 < j < k$. Fig. 3.12a illustrates the grid after the projection of the first five LR images from the pool. Then, in Fig. 3.12b, 20 images from the pool have already been projected. Finally, the grid in Fig. 3.12c contains information from all 35 LR images. If more than one LR pixel is projected onto the same grid pixel, we use only the closest one.

(a) **(b)** **(c)**

Figure 3.12: In (a), five LR images have been projected onto the same HR grid. In (b), 20 images from the pool have been projected. Finally, the grid in (c) contains information from 35 LR images. Black pixels represent the unknown information that we want to recover using inpainting techniques.

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" [114, 7]. The basic idea is simple: it fills in parts of an image using information from surrounding areas, and several algorithms have been designed for this

purpose [7, 114]. Commonly, an inpainting algorithm starts by selecting the image regions to be filled in. Then, the known image information is used to fill in the missing areas. The techniques should attempt to continue the isophotes (lines of equal gray value) as smoothly as possible inside the reconstructed region, in order to produce a perceptually plausible reconstruction [114].

We refer to this method as inpainting-based Super Resolution (ISR). We implemented two variations of the method, each one using a different inpainting technique. We explain such variations in Secs. 3.2.1 and 3.2.2. It is worth mentioning that, since this method does not solve the system of equations, we might not recover the high-frequency components lost during the blurring process. As we previously discussed in Sec. 3.1, a low-pass filter is typically used prior to downsampling in order to prevent occurrence of aliasing. Then, if we assume that an original high-resolution image I_{HR} has been blurred and generated the image I_G (using a Gaussian filter, for example), and the LR observations that we are super resolving (Fig. 3.10) have been downsampled from I_G, we expect to reconstruct I_G instead of I_{HR}. Unlike the inpaiting-based Super Resolution, the RLS model is designed to recover such high-frequency components lost during the blurring process, reconstructing the original I_{HR}. On the other hand, the inpainting-based Super Resolution might run faster than the RLS-based methods.

3.2.1 First inpainting-based variation (ISR1)

Both ISR variations receive as parameters: the pool of LR images I_k, the dimensions M and N of the HR grid that we want to reconstruct, and the matrices Υ_k mapping each I_k onto the grid. It is possible to implement ISR storing in memory, basically, three main information pieces:

1. The HR image that we want to reconstruct. Initially, the grid is empty.
2. The set $\mathscr{S}$ of grid positions that might be inpainted (initially, all of them).
3. The minimum distance from each grid pixel to its closest neighbor in the LR images. We suggest storing such distances because, if more than one LR pixel is projected onto the same grid pixel, only the closest one might be used.

Then, for each LR image I_k, the method can quickly find the grid pixel that is closest to each LR pixel $I_k(x, y)$ by performing only two steps:

- First, the matrices Υ_k are used to map the LR pixels onto the grid. For each $I_k(x, y)$, the position (x', y') of such pixel projected onto the grid can be calculated by Eqs. 3.1 and 3.2.
- Then, rounding the positions (x', y'), as in the RLS algorithm (see Sec. 3.1), it is possible to find the grid pixel that is closest to $I_k(x, y)$, and then update the grid and the set $\mathscr{S}$.

Each variation uses a different inpainting method to fill in the grid intensity pixels that remain in $\mathscr{S}$. It is important to mention that we advise not to apply any transformation onto the LR images, because it would cause loss of information. Instead, the matrices Υ_k can be used to know exactly the position where each $I_k(x, y)$ is with respect to the HR grid, without applying the transformation. Moreover, each input image can be individually processed, so there should not be more than one LR image in the memory at the same time.

The first inpainting technique that we use in our method is based on the work of Bertalmio et al. [7]. We refer to this implementation as ISR_1. The main idea of this approach is to think of the image intensity as a 'stream function' for a two-dimensional incompressible flow. The Laplacian of the image intensity is transported into the region to be inpainted by a vector field defined by the stream function. The algorithm is designed to continue the isophotes continuously from the exterior into the region to be inpainted, while matching gradient vectors at the boundary of the inpainting region. Their method is directly based on the Navier-Stokes equations for fluid dynamics [37], which has the immediate advantage of well-developed theoretical and numerical results.

For a sneak peak showing the potential of this algorithm, Figure 3.13 compares a simple interpolation to an image super resolved by ISR_1. Fig. 3.13a shows the first LR image in Fig. 3.10 zoomed by a factor of four, using nearest neighbor interpolation. In Fig. 3.13b, the super-resolved image generated by ISR_1, using all the 35 LR images from the pool. The dimension of the super-resolved image is four times larger than the LR images in both axis. In Fig. 3.13c, the blurred HR image that generated the LR observations (such HR image might not be available in real-world applications, but we show it here only for visual purposes).

(a) **(b)** **(c)**

Figure 3.13: In (a), an LR image is zoomed in by a factor of four using Nearest Neighbors interpolation. In (b), 35 LR images are super resolved by ISR_1. In (c), the blurred image that generated the LR observations in the pool.

Finally, Fig. 3.14 shows that ISR_1 generates a reconstructed image very similar to the original HR image if we do not apply the anti-aliasing filter. As in Fig. 3.13c, we do not expect to have such original HR image in real-world applications. We only present it here to show the potential of ISR_1. Fig. 3.14a shows the first LR image in Fig. 3.10 zoomed by a factor of four, using nearest neighbor interpolation. In Fig. 3.14b, the super-resolved image generated by ISR_1, using all the 35 LR images from the pool. Finally, in Fig. 3.14c, the original HR image that generated the LR observations, without the anti-aliasing filtering.

(a) (b) (c)

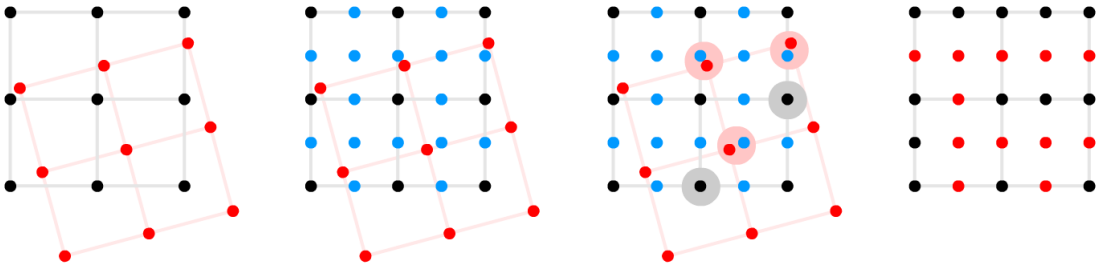
Figure 3.14: In (a), a LR image is zoomed in by a factor of four using Nearest Neighbors interpolation. In (b), 35 LR images are super resolved by ISR_1. In (c), the original image that generated the LR observations in the pool.

3.2.2 Second inpainting-based variation (ISR2)

We refer to the second implementation of the inpainting-based Super Resolution as ISR_2, in which we use the inpainting algorithm proposed by Telea in [114]. The algorithm is based on propagating an image smoothness estimator along the image gradient, similar to [8]. The image smoothness is estimated as a weighted average over a known image neighborhood of the pixel to inpaint. The missing regions are treated as level sets, and the author use the fast marching method (FMM) [98] to propagate the image information. 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.

3.3 Geometric k-Nearest Neighbors Multi-Frame Super-Resolution

Finally, we introduce another super-resolution algorithm that is designed to be even faster than ISR, and that presents low-memory footprint. To fastly reconstruct an HR image I_{HR} from a sequence of input images, we focus now on a direct fusion-based method formulation (see Sec. 2.2). The Figure 3.15 illustrates the basic idea behind our algorithm.



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

Figure 3.15: (a) Two LR images already aligned in the registration step. (b) New pixels that will appear in the HR image. (c) For each pixel in the HR image, we find its geometric nearest neighbor among all pixels in the LR input sequence. (d) The HR image, composed by pixels from both input images.

Black and red dots in Fig. 3.15a represent pixels from two LR images I_1 and I_2, already aligned in the registration step. Then, we create a grid containing the HR pixels, as Fig. 3.15b shows. In this example, blue and black dots compose the grid (blue dots are new pixels that will appear in the resulting image, and black dots are pixels in the HR image that coincide with I_1 pixels). In Fig. 3.15c, we choose the nearest neighbor for each I_{HR} pixel among all pixels in I_1 and I_2. Finally, the final SR image is composed by pixels from both I_1 and I_2 in Figure 3.15d.

There are different possible policies to choose and combine nearest neighbors geometrically. We investigate five simple variations of this algorithm, always aiming at improving the quality of the super-resolved image in a small runtime and with a low memory footprint. Such variations differ from each other in the choice of the best candidates for each pixel in the HR grid. We refer to basic algorithm as GSR (Geometric k-NN Super Resolution), and we detail the five variations in Secs. 3.3.1 through 3.3.4. All of them receive as parameters: the pool of LR images I_k, the dimensions M and N of the HR grid that we want to reconstruct, and the matrices Υ_k mapping each I_k onto the grid.

3.3.1 GSR1: 1-NN

Let p be a pixel in the HR image (the blue dot in Fig. 3.16a) and q_k be the nearest pixel to p in each LR image I_k (each black dot in Fig. 3.16a is the nearest pixel q_k to p in an LR image I_k). In the first algorithm variation, we chose each p in the HR image exactly as in Figs 3.15 and 3.16b: using its closest neighbor among all possible values of q_k. In other words, we use k-nearest neighbors with $k = 1$ to find q_k from all LR images, and then another 1-NN to find the closest q_k with respect to p.

Figure 3.16: In (a), the blue dot is a pixel p in the HR image, and each black dot is the pixel q_k in an LR image I_k that is closest to p. In (b), the value of p in GSR_1 is exactly the same value as its closest q_k (the red dot).

We refer to this variation as GSR_1. It is possible to implement GSR_1 storing in memory, basically, three main information pieces:

1. The HR image that we want to reconstruct. Initially, the grid is empty.
2. The (x, y) positions of the grid pixels.
3. The minimum distance from each grid pixel to its closest neighbor among all the LR images.

Then, for each LR image I_k, the method can quickly find the LR pixel that is closest to each grid pixel $I_{HR}(x, y)$ by performing only two steps:

- First, the matrices Υ_k are used to map the grid pixels onto the LR image (unlike ISR, in which we map the LR pixels onto the grid). Hence, for each pixel $I_{HR}(x, y)$ in the grid, the position (x', y') of such pixel projected onto I_k is calculated by Eqs. 3.1 and 3.2 (but using the inverse transformation).

- Then, rounding the positions (x', y'), the difference between x' and its rounded value is the distance in the horizontal axis from I_{HR} to its closest neighbor in I_k, and the difference between y' and its rounded value is the distance in the vertical axis.

As in the Inpaiting-based method, we advise not to apply any transformation to the LR images, because it would cause loss of information. Instead, the matrices Υ_k might be used to know exactly the (x', y') position where each grid pixel $I_{HR}(x, y)$ is with respect to the LR image I_k. Moreover, each input image can be individually processed, so there must not be more than one LR image in the memory at the same time. In doing so, the memory consumption of this method is $\mathcal{O}(M \times N)$ for an HR image of $M \times N$ pixels.

3.3.2 GSR2 and GSR3: k-NN and Weighted k-NN

In GSR_1, we use only the closest neighbor among the LR images to calculate each HR pixel. In the other variations of GSR, we also consider other pixels in the nearby. In the second variation (GSR_2), we first use 1-NN to find all q_k from the input sequence, exactly as in GSR_1. Then, we use a 3-NN to get the three best values of q_k. The resulting pixel p is an average among such three closest neighbors, as we see in Fig. 3.17.

Figure 3.17: In (a), the blue dot is a pixel p in the HR image, and each black dot is the pixel q_k in an LR image I_k that is closest to p. In (b), the resulting pixel p is an average among its three closest neighbors.

Similarly, GSR_3 finds the three best values of q_k, but combines them as a weighted average. We use static weights (60% for the nearest one and 20% for each other), but could also set them to be an inverse proportion to their distances. In both GSR_2 and GSR_3, we store three arrays for the HR pixels of the three closest neighbors, and three arrays for the three minimum distances. Then, we udpate such arrays for each LR image in the pool using only vectorized operations, as in GSR_1.

Both algorithms GSR_2 and GSR_3 can produce bad results if the three closest values of q_k do not form a region which includes p. Figure 3.18 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 [94], the authors purpose a Delaunay triangulation for all available points and combine the vertices of the triangle to

which p belongs. According to [23], the triangulation can be build in $\mathcal{O}(n \log n)$, while the simplest solution using a k-NN search is $\mathcal{O}(n)$. We did not implement this option to avoid increasing the algorithm runtime for now, but we shall investigate it further as future work (see Chapter 7).

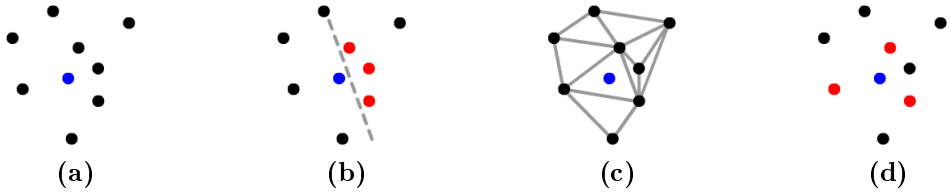


Figure 3.18: In (a), the blue dot is the target pixel p and each black dot is its nearest pixel q_k from an LR image I_k. In (b), 3-NN chooses the red pixels to be combined, but they may not be a good choice because they do not form a region to which p belongs. In (c), a triangulation is calculated among all black pixels. In (d), p belongs to the triangle formed by the red pixels.

3.3.3 GSR4: Weighted average within a neighborhood

In GSR_4, the target HR pixel p is the average among all q_k (the nearest pixel from p in each LR image I_k), as illustrated in Fig. 3.19.

Figure 3.19: In (a), the blue dot is a pixel p in the HR image, and each black dot is the pixel q_k in an LR image I_k that is closest to p. In (b), the resulting pixel p is the average among all q_k (the nearest pixel from p in each LR image I_k).

If we have 30 input images, for example, q_k is the nearest pixel from p in each of the 30 LR images and p is the average among all of them. The GSR_4 implementation is similar to GSR_1, but even simpler and faster. Now, we do not store the minimum distances, only the arrays with the reconstructed HR image and the position of its pixels. For each LR image, we accumulate in an HR pixel the intensity value of its closest LR pixel in I_k. After all input images, we divide the HR pixels by the number of LR images. We only ignore an LR pixel if q_k is far from p by more than two pixels (for example, if a region of the LR image is located outside the borders of the grid).

3.3.4 GSR5: Weighted average within a circular region

Finally, GSR_5 sets a maximum radius distance with respect to p. Each target HR pixel is calculated as the weighted average among all q_k within this circular region. The weights are inversely proportional to their distances.

Figure 3.20: In (a), the blue dot is a pixel p in the HR image, and each black dot is the pixel q_k in an LR image I_k that is closest to p. In (b), the resulting pixel p is an average among its neighbors inside a circular region around p.

In this fifth variation of GSR, we store in memory an array for the the HR image that we want to reconstruct, and an array for the positions of the HR pixels. Then, for each LR image I_k, we map the grid positions onto the LR image, round the positions to fastly calculate the distance from each HR pixel to its closest pixel in I_k, and then create the weights (inversely proportional to the distances). We use such weights to accumulate the intensity pixels for all the LR images, and in the end we divide them by the sum of weights.

It is worth mentioning that there should not be more than one LR image in memory at the same time in all five implementations of GSR, and the memory consumption can be $\mathcal{O}(M \times N)$ for an HR image of $M \times N$ pixels.

Chapter 4

Validation of the super-resolution methods

For a quantitative validation of a super-resolution algorithm, it is a standard practice in the literature to choose an HR target image I_{HR} and generate a series of LR versions of it. Then, those LR images might be super resolved onto an image I_{SR}, and I_{HR} can be used as a ground-truth for I_{SR}. In Figure 4.1, an HR image generates a pool of LR images (I_1, I_2, I_3 and I_4), that might be used as input for the super resolution.

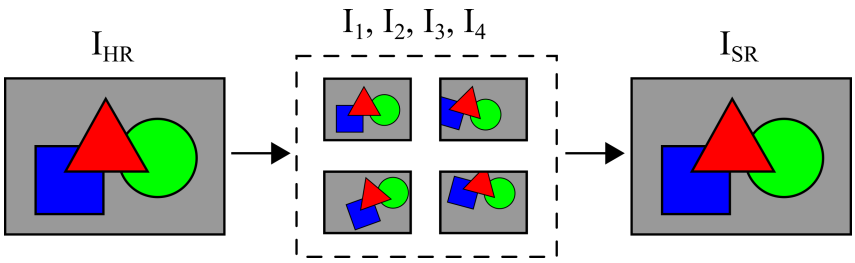


Figure 4.1: For a quantitative validation of a SR method, an HR image (I_{HR}) generates a pool of LR images (I_1, I_2, I_3 and I_4). Then, such LR images can be combined and super resolved onto a higher resolution image I_{SR}.

The most similar, according to a given similarity measure, the super-resolved image is to the original one, the most accurate the super resolution. This is a fictitious situation, but allows us to quantitatively compare algorithms. For a qualitative evaluation, as there is no target image in most applications, the results from one algorithm often are visually compared to images generated by other techniques. As most works in the literature, we use both qualitative and quantitative validation forms to evaluate our algorithms. We divide the validation into three parts:

This chapter is divided into two parts: first, in Section 4.1, we explain the adopted experimental methodology to validate the SR methods described in Chapter 3, and we show quantitative results for the proposed methods; then, in Section 4.2, we perform a qualitative evaluation of the algorithms, applying them to an always-on low-power environment. We implemented all experiments using Python 2.7 and the OpenCV 3.1 library [12, 49]. To measure the runtimes and memory footprints, we use a single machine with Intel Core i7 processor, 2.0GHz ×4, 8Gb RAM, and 64-bits Ubuntu 16.04.

4.1 Experiment #1: Quantitative validation of the SR methods

In this first experiment, we perform a quantitative evaluation of all proposed methods for multi-frame super resolution (RLS, $RLSO$, $RLSU$, ISR_1, ISR_2, GSR_1, GSR_2, GSR_3, GSR_4, and GSR_5, described in Chapter 3). All of them need an initial step, before the reconstruction, to align the input images. The general pipeline for our methods is illustrated in Figure 4.2:

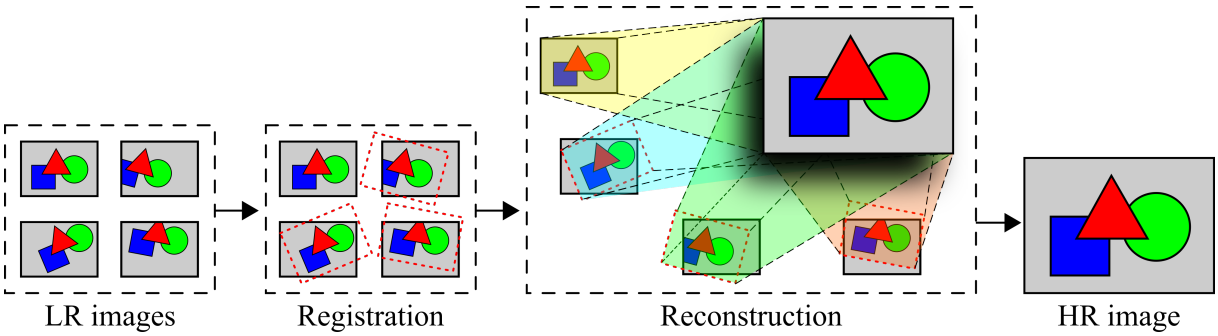


Figure 4.2: Common pipeline for our super-resolution algorithms. The registration step calculates the transformation matrices mapping the first LR image onto the others. Then, in the reconstruction step, our algorithms combine the LR images into an HR grid. The ultimate result is the super-resolved HR image.

The dataset for this first experiment is presented in Section 4.1.1. We describe the whole process to generate the pool of LR images, and to calculate the transformation matrices among them. Then, in Section 4.1.2, we explain the registration methods that we use in this first experiment. The validation metrics used are detailed in Section 4.1.3. Finally, in Section 4.1.4, we show the obtained results.

4.1.1 Dataset #1

To create a pool of LR images for the quantitative evaluation, we first collected 100 digital images to be target for our super-resolution techniques. We separate such images into two subsets:

1. The subset $\mathscr{D}_1$ contains 70 images[1] with dimensions of 512×384 pixels. The target image in Fig. 3.14c, for example, belongs to this subset.

2. The subset $\mathscr{D}_2$ contains 30 images[2] with higher dimensions, up to $6,000 \times 4,000$ pixels (e.g., the target image in Fig. 1.1).

[1]The images in $\mathscr{D}_1$ have been collected from *UCID*, an uncompressed color image dataset [95].

[2]The images in $\mathscr{D}_2$ have been collected from *http://www.gratisography.com/*. The images are free of Copyright Restrictions, under "Creative Commons Zero" license, and dedicated to public domain. Anyone can copy, modify, and distribute them, without asking permission. See *http://creativecommons.org/publicdomain/zero/1.0/* for more details.

In order to generate n low-resolution versions of each image I_{HR} in $\mathscr{D}_1$ and $\mathscr{D}_2$, we created n random transformations to be applied to I_{HR}. The transformation from the target image to each LR image is composed by a rotation matrix R around the center of the image, a scaling matrix S, and a translation matrix T. Before the transformation, we apply a Gaussian blur on I_{HR} to avoid the occurrence of aliasing after the downscale, generating an image I_G. Then, an LR image I_{LR} can be described with respect to I_G:

$$I_{LR} = \begin{pmatrix} cos\theta & sin\theta & (1-cos\theta)\cdot x - sin\theta\cdot y \\ -sin\theta & cos\theta & sin\theta\cdot x + (1-cos\theta)\cdot y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} I_G$$

$$I_{LR} = (RTS)I_G$$

Additionally, we are able to calculate what should be the "correct alignment" among the LR generated images. Let R_1, T_1 and S_1, be the transformation matrices mapping an HR image I_G onto the first LR image I_1. We can write I_G with respect to I_1, as:

$$\begin{aligned} I_1 &= (R_1T_1S_1)I_G, \\ I_G &= (R_1T_1S_1)^{-1}I_1. \end{aligned} \tag{4.1}$$

Now let R_k, T_k and S_k be the matrix mapping the HR image I_G onto an arbitrary LR image I_k ($k \in [2, n]$):

$$I_k = (R_kT_kS_k)I_G, \tag{4.2}$$

$$I_k = (R_kT_kS_k)(R_1T_1S_1)^{-1}I_1. \qquad \text{(from Equations 4.1 and 4.2)}$$

Finally, let Ψ_k be the matrix $(R_kT_kS_k)(R_1T_1S_1)^{-1}$:

$$I_k = \Psi_k I_1. \tag{4.3}$$

In Figure 4.3, we see the matrices mapping the HR image I_G onto all n LR images, and the matrices Ψ_k mapping I_1 onto the other LR images I_k.

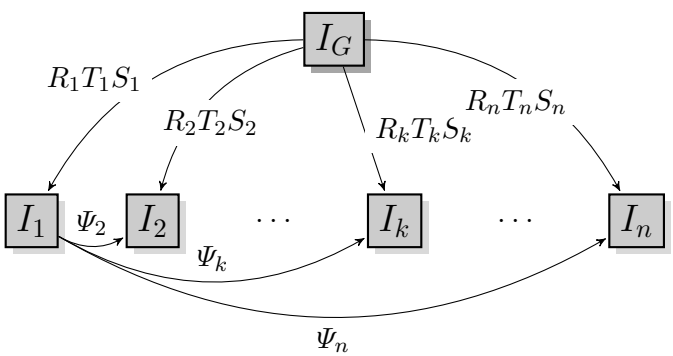


Figure 4.3: Correct alignments can be found by calculating the matrices Ψ_k, mapping I_1 onto the other LR images I_k.

From Equation 4.3, we have transformation matrices Ψ_k to describe any LR image

I_k ($k \in [2, n]$) with respect to I_1. The matrices Ψ_k are exactly the transformations that should be found in the registration step of the super-resolution algorithms. The more similar a registration matrix is with respect to Ψ_k, the more accurate will be the super-resolved image. We further refer to Ψ_k as the "correct alignment" among the LR generated images.

Each LR image of the pool was then cropped, so there will be no borders or empty spaces due to the previous transformations (see Figure 4.4 for details).

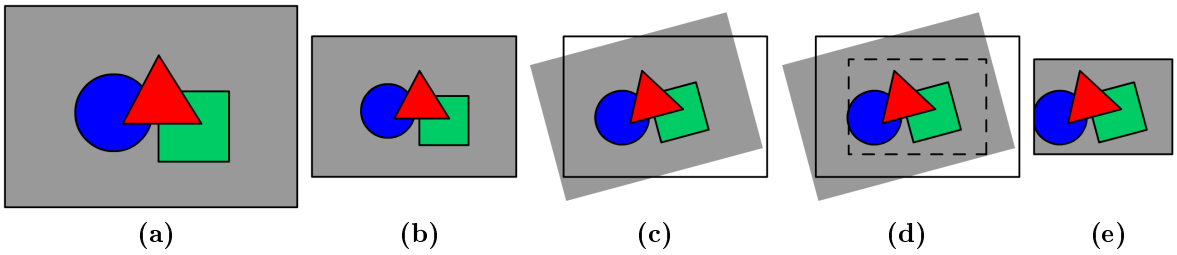


Figure 4.4: The HR image in (a) is followed by a decreasing scale in (b), and may contain empty spaces after the rotation and translation in (c). In (d), we see the area to be cropped, and in (e) the final LR image.

The cropping area is the same for all LR images (we consider the maximum possible values for rotations and translations from the input file). In Figure 4.5, we see examples of LR versions of an HR image from the subset $\mathscr{D}_2$.

Figure 4.5: Two LR versions of an HR image from the subset $\mathscr{D}_2$.

Every HR target image generated 100 random LR images (with size 25% of the original image). Therefore, we have a dataset with 100 target images, $100 \times 100 = 10,000$ transformations $R_k T_k S_k$, $99 \times 100 = 9,900$ matrices Ψ_k, and $100 \times 100 = 10,000$ LR images to be reconstructed. It is worth mentioning that two LR images in this first dataset do not differ from each other by a perspective transformation, since we apply only rotation, scaling, translation and the anti-aliasing filter to the original image. We evaluate if the proposed SR methods can handle perspective transformations in the Experiments #2 (Sec. 4.2.1) and #3 (Chap. 6).

4.1.2 Registration step

The "correct alignment", described in Sec. 4.1.1, may not be available in real-world applications, but it is an important aid for evaluating algorithms. When we have such setup,

every inaccuracy in the super-resolved image comes from the reconstruction step, and is not due to an incorrect registration. In this first experiment, since we generate the pool of LR images from a set of collected HR original images, we have all the matrices Ψ_k with the correct mapping from any LR image I_k ($k \in [2, n]$) to I_1. Hence, we use such correct alignment to evaluate the super-resolution algorithms proposed in Chap. 3.

Additionally, we also perform a registration step, prior to the reconstruction, trying to approximate this experiment to real-world situations. Given a set of n LR images $I_1, I_2, \ldots I_n$, we find representative keypoints and descriptors in all images and then match the keypoints in I_1 to keypoints of I_k for each $k \in [2, n]$. The best matches between I_1 and each I_k are used to estimate a transformation Φ_k. Thus, we write I_k with respect to I_1, as follows:

$$\forall k \in [2, n] : I_k = \Phi_k I_1 \tag{4.4}$$

The aim is to find a matrix Φ_k as similar as possible to the correct matrix Ψ_k. We do not design a new registration technique herein. Instead, we evaluate three well-known and established feature detectors to find the mentioned keypoints and descriptors: Scale-invariant feature transform (SIFT) [68, 69], Speeded Up Robust Features (SURF) [6] and Oriented FAST and Rotated BRIEF (ORB, purportedly a fast and efficient alternative to SIFT) [93]. All of them are provided by the OpenCV library [12, 49]. The points are matched using k-NN and Flann [75, 43]. To illustrate the mapping between two images, we use SIFT to find representative keypoints in Figure 4.6. Each line in such figure represents one match between two input images. The green polygon shows the position of the first image with respect to the second one.

Figure 4.6: Matches between two LR images using SIFT.

4.1.3 Validation metrics

There are three standard metrics to evaluate the similarity between two images: Mean Squared Error (MSE), Peak signal-to-noise ratio (PSNR) and Structural Similarity Index (SSIM) [124]. We validated the experiment results using these three metrics, but we show only the SSIM results in this chapter (see Appendix A for results with PSNR and MSE). The structural similarity is designed to improve on traditional methods like PSNR and MSE, and its results are more similar to human perception [123]. It attempts to measure the change in luminance (modelled as pixel intensity), contrast (variance between the reference and distorted image), and structure (cross-correlation between the two images) in an image:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{4.5}$$

where μ_x and μ_y are, respectively, the local sample means of x and y, σ_x and σ_y are, respectively, the local sample standard deviations of x and y, and σ_{xy} is the sample cross-correlation of x and y after removing their means. The items C_1 and C_2 are small positive constants that stabilize each term, so that near-zero sample means, variances, or correlations do not lead to numerical instability. The SSIM index is computed locally within a sliding window that moves pixel-by-pixel across the image, resulting in an SSIM map. The SSIM score of the entire image is then computed by simply averaging the SSIM values across the generated map. The maximum value $SSIM = 1$ happens if and only if the two images are equal. The higher the SSIM value, the more similar one image to the other.

4.1.4 Results

The results of this first experiment are divided in seven rounds:

- *Round #1*: Comparison among different values for the parameters of the *RLS* algorithm: σ (the standard deviation of the Gaussian distribution); d (an integer such that the radius $r = d \times \sigma$ defines the kernel size); and α (the regularization parameter). **Objective:** To find the best parameters for Regularized Least Squares.

- *Round #2*: Comparison among the three implementations of the *RLS* algorithm, using the best combination of parameters found in the previous round. **Objective:** To find the most efficient implementation of *RLS* with the best SSIM values.

- *Round #3*: Comparison among the two variations of the inpainting-based algorithms. **Objective:** To find the most efficient implementation of *ISR* with the best SSIM values.

- *Round #4*: Comparison among the five implementations of the Geometric k-Nearest Neighbors Multi-Frame Super-Resolution. **Objective:** To find the most efficient implementation of *GSR* with the best SSIM values.

- *Round #5*: Comparison among the best *RLS* implementations, the best inpainting-based algorithm, and the best variation of *GSR*. **Objective:** To find the most efficient reconstruction algorithm with the best SSIM values.

- *Round #6*: Comparison among our best SR algorithms and other reconstruction techniques available in the literature. **Objective:** To put the designed solutions in perspective with respect to existing counterparts in terms of efficiency and effectiveness.

- *Round #7*: Comparison among the feature detectors SIFT, SURF, and ORB, and the correct alignment. **Objective:** Point out out the most appropriate registration algorithm for our purposes.

The reconstructed images in this experiment are 4× higher than the LR images in all rounds, so the dimensions of a super-resolved image I_{SR} is always the same as the target image I_{HR}. The reconstruction algorithms are always performed using from 2 to 50 random LR images. The results in the charts are showed as a function of the number of images in the input sequence.

Round #1 - Best parameters for RLS

In order to super resolve a series of LR input observations using the original *RLS* model, described in Chap. 3, we need to specify the parameters σ (the standard deviation of the Gaussian distribution), d (an integer such that the radius $r = d \times \sigma$ defines the kernel size) and α (the regularization parameter). In this first round, we test *RLS* using different values for this three parameters.

In Sec. 4.1.1, we discussed that the original images I_{HR} from the dataset have been blurred before the downsampling, generating the images I_G. We applied a Gaussian filter with $\sigma = 0.5$ to create such images in this round. Hence, we expect that $\sigma = 0.5$ might be the best value for the standard deviation in the equations of the *RLS* linear system. We also execute the method using $\sigma \in \{0.3, 0.7\}$ to evaluate how the results change as we increase or decrease such parameter. In addition, we discussed in Chap. 3 that, when computing a discrete approximation of the Gaussian function, pixels at a distance of more than 3σ are small enough to be considered effectively zero. Hence, we expect $d = 3$ as a good choice for the kernel parameter. Similarly to σ, we evaluate how $d \in \{1, 5\}$ impacts the results. Finally, in our implementation of the original *RLS* model, we use the Ridge regression model from the Scikit-Learn library [85] to solve the Tikhonov regularization. The default value for the regularization parameter is $\alpha = 1.0$, and larger values specify stronger regularization[3]. We execute the method using the default value for α, and also with $\alpha \in \{0.1, 0.5\}$ (reducing the conditioning of the problem).

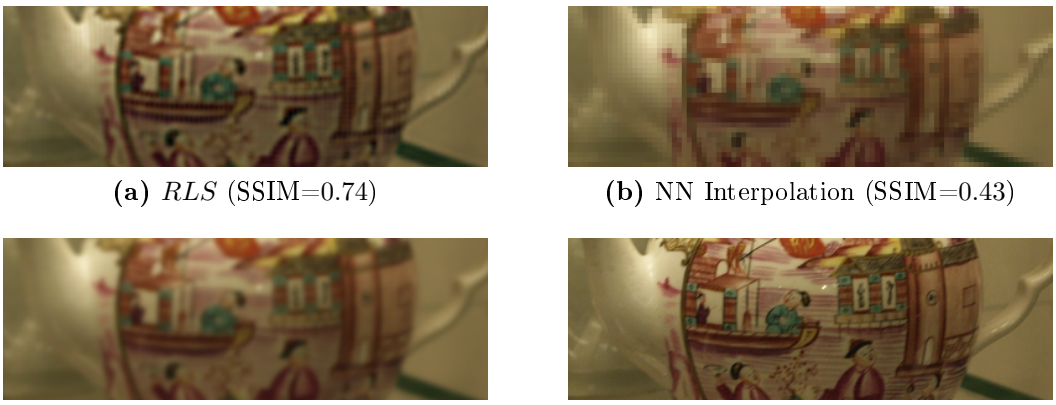
We start by finding the best values for σ and d (the parameters of the anti-aliasing filter). First, we use $\alpha = 0.1$ and different values for σ and d, for all images in $\mathscr{D}_1$. The best Gaussian parameters for such α are $\sigma = 0.5$ and $d = 3$, on average, but the results for $d = 5$ are very similar to $d = 3$ (the chart including all combinations of σ and d is in

[3]Regularization improves the conditioning of the problem and reduces the variance of the estimates.

Appendix A). We then use the pairwise Wilcoxon test [125], with Bonferroni correction [33] and 95% of confidence, to calculate the statistical difference among all the averages in the chart. The Wilcoxon test verifies how likely is to the results have occurred by chance, and the Bonferroni correction adjusts the results of such test to decrease the number of false-positives when we have multiple comparisons (e.g., the multiple algorithms in Fig. A.7). The Wilcoxon test indicates there is statistical difference between the two best combinations for σ and d ($\sigma = 0.5$ and $d \in \{3, 5\}$) for any number of input images $n > 1$. Such difference might not have occurred by chance, but because of the variation of the parameters. In addition, for all $n > 1$, we produce best similarities with $(\sigma = 0.5, d = 3)$ for the most target images in the pool (e.g., using 50 LR images as input, 87% of the SSIM results are higher with this combination of values for σ and d).

Finally, *RLS* with $\sigma = 0.5$, $d = 3$, and $\alpha = 0.1$ generated super-resolved images more similar to the target image I_{HR} than the blurred image I_G (that generated the input images for the super resolution) for all $n \geq 30$, with statistical difference between them. Therefore, we can claim that the multi-frame super resolution using Tikhonov regularization and the adaptive Gaussian filter could, indeed, recover high-frequency components lost during the blurring process.

For a better visualization of the results, in Fig. 4.7, we reconstruct one of the images from the subset $\mathscr{D}_1$ using $\sigma = 0.5$, $d = 3$, $\alpha = 0.1$, and 50 LR images as input. Then, Fig. 4.7b shows an LR image from this subset zoomed in by a factor of four. Fig. 4.7c shows the blurred image I_G that generated the input images for the super resolution, and Fig. 4.7d shows the target image I_{HR} that generated I_G. The super-resolved image in Fig. 4.7a not only presents the highest SSIM, but it is also visually more similar to I_{HR} than I_G and than the interpolated image.



(a) *RLS* (SSIM=0.74) **(b)** NN Interpolation (SSIM=0.43)

(c) I_G (SSIM=0.64) **(d)** Target HR image I_{HR}

Figure 4.7: In (a), 50 LR images from the subset $\mathscr{D}_1$ are super resolved by *RLS* using $\sigma = 0.5$, $d = 3$, $\alpha = 0.1$. In (b), an LR image is zoomed in by a factor of four using Nearest Neighbors interpolation. In (c), the blurred image I_G that generated the input sequence. In (d), the target HR image I_{HR}.

Then we find the quality of RLS for $\alpha = 0.5$. When we use $n \geq 40$ images as input, the best SSIM values are found with $(\sigma = 0.5, d = 3)$, on average. However, such highest similarity seems very similar for $(\sigma = 0.5, d = 5)$ and $(\sigma = 0.7, d = 1)$ (the chart including

different values of σ and d for $\alpha = 0.5$ is in Appendix A). We then verify the algorithm that produces the highest similarities for the most target images. Using up to $n = 30$ LR images as input, the best SSIMs are found with $(\sigma = 0.7, d = 1)$. Then, the similarity becomes higher using $(\sigma = 0.5, d = 3)$. The Wilcoxon test indicates a statistical difference between $(\sigma = 0.5, d = 3)$ and $(\sigma = 0.7, d = 1)$ only for $n \geq 50$.

Finally, we verify the best parameters of the anti-aliasing filter using $\alpha = 1.0$. Unlike $\alpha = 0.1$ and $\alpha = 0.5$, the results for $\alpha = 1.0$ are better using $(\sigma = 0.7, d = 1)$ than using $\sigma = 0.5$, on average. The combination of parameters $(\sigma = 0.7, d = 1)$ also produces the highest similarities in the most target images for all $n > 2$. Moreover, *RLS* with such values is statistically different from all the other combinations, for all $n > 2$.

Fig. 4.8 summarizes the best values for σ and d using each $\alpha \in \{0.1, 0.5, 1.0\}$ (see the individual charts in Appendix A). We can see that the best combination of parameters for *RLS* in this experiment is $(\sigma = 0.5, d = 3, \alpha = 0.1)$. According to the Wilcoxon test, such combination has statistical difference with respect to all others in the chart, for all $1 \leq n \leq 50$. In addition, for all target images in the dataset, and for any number n of input LR images, the best SSIMs are found with $(\sigma = 0.5, d = 3, \alpha = 0.1)$. This combination is also the one whose similarities outperform the blurred image I_G, for $n \geq 30$.

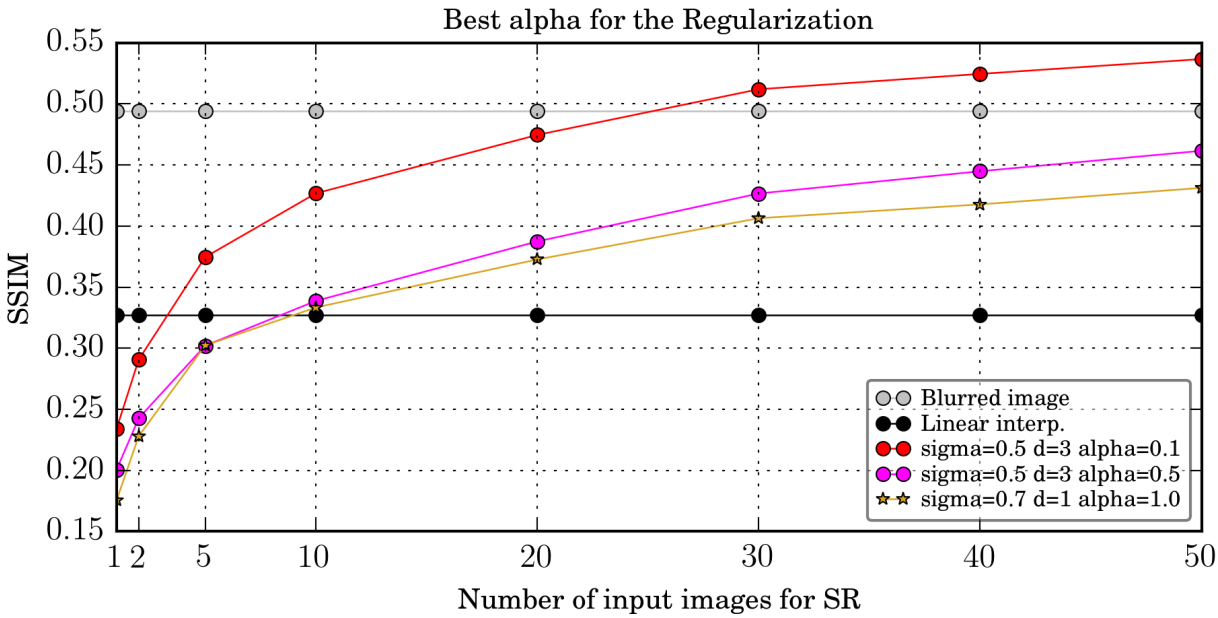


Figure 4.8: Best SSIM values for *RLS* using $\alpha \in \{0.1, 0.5, 1.0\}$.

As we previously discussed, pixels at a distance of more than 3σ are small enough to be considered effectively zero in a discrete approximation of the Gaussian filter, so we expected $d = 3$ as a good choice for the kernel size. Besides, the aforementioned results show that the similarity values do not increase for $d > 3$.

Moreover, it is worth mentioning that we expected $\sigma = 0.5$ as the best value for the standard deviation, given that the original HR image I_{HR} has been blurred with $\sigma = 0.5$. Hence, the chart in Fig. 4.9 shows *RLS* results when the original image is blurred with $\sigma = 0.3$, instead of $\sigma = 0.5$ as in the previous charts. Although the best SSIM in Fig. 4.9 are found with $\sigma = 0.3$, we still have good results when we reconstruct the images using

$\sigma = 0.5$ (additionally, such results are statistically different than using $\sigma = 0.3$ for $n < 30$ LR input images).

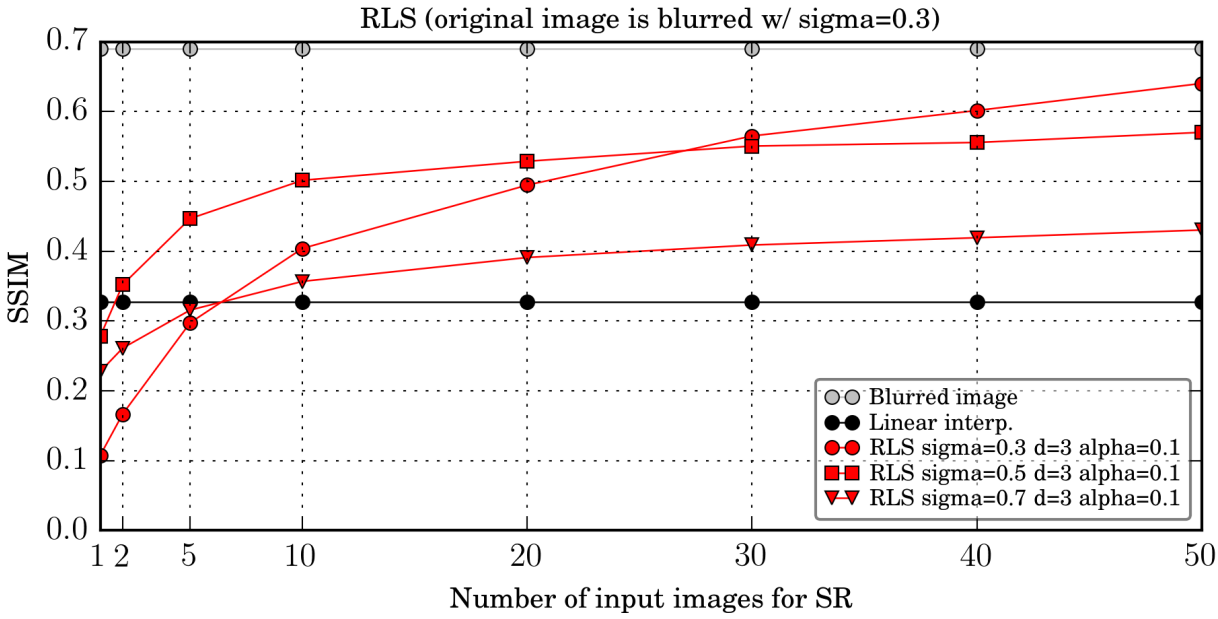


Figure 4.9: SSIM values for *RLS* when the original image is blurred with $\sigma = 0.3$.

Finally, the chart in Fig. 4.10 shows the *RLS* results when the original image is blurred with $\sigma = 0.7$. Unlike the chart in Fig. 4.9, the best SSIM values in Fig. 4.10 are found with $\sigma = 0.5$. Therefore, we suggest using 0.5 as the parameter σ in the *RLS* algorithm even when we do not know which σ has been used during the formation of the LR images.

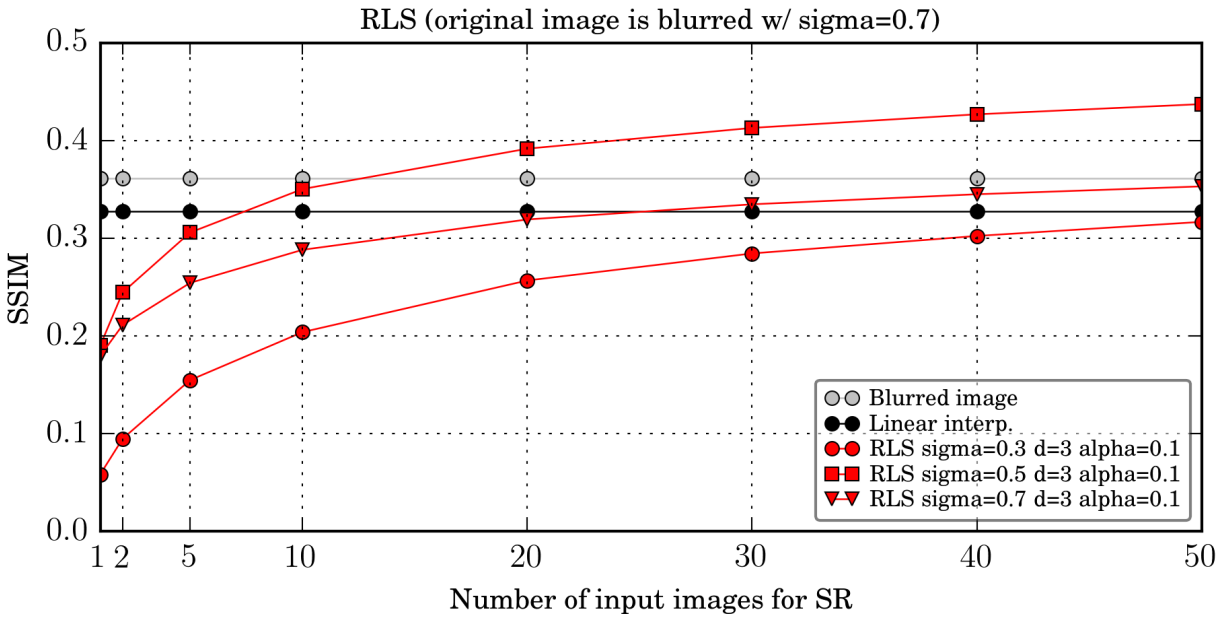


Figure 4.10: SSIM values for *RLS* when the original image is blurred with $\sigma = 0.7$.

We now turn our attention to verifying how these three parameters impact the runtime of the original *RLS* model. First, we notice in Fig. 4.11, that the first implementation of *RLS* runs in linear time as we increase the number of input frames to super resolve.

Moreover, although the regularization parameter $\alpha = 0.1$ produces images with the best SSIMs, its execution time is slightly slower than for $\alpha \in \{0.5, 1.0\}$.

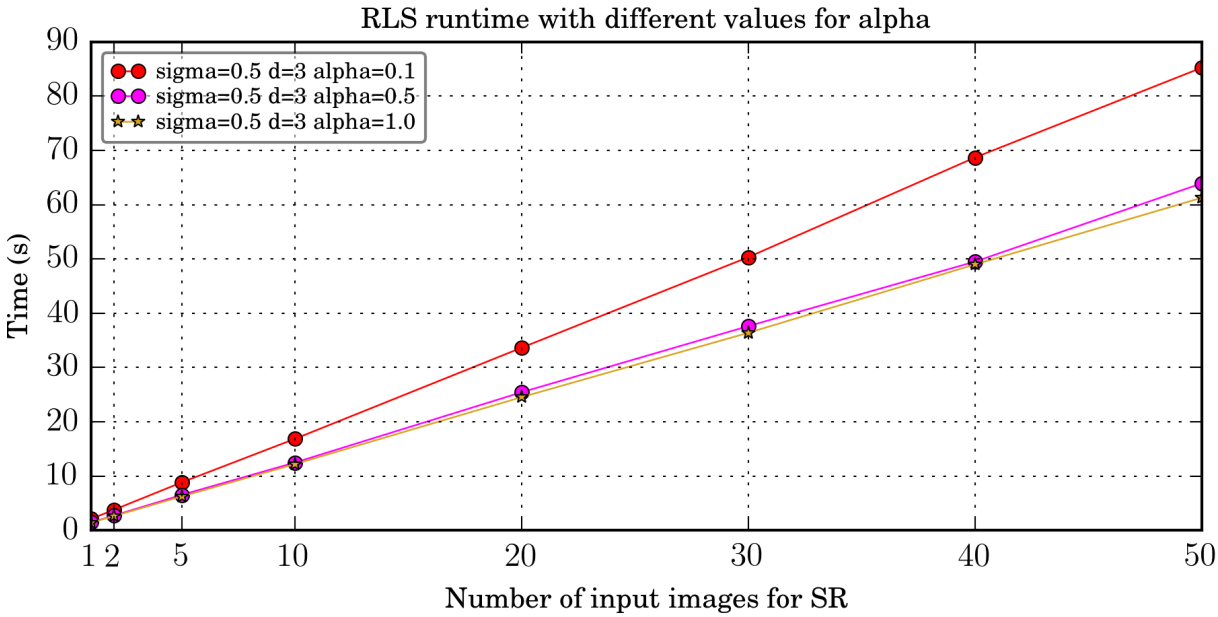


Figure 4.11: *RLS* runtime using different values for α.

Unlike the regularization parameter, the best value for the Gaussian standard deviation is not the slowest one. Fig. 4.12 shows that the method runs faster with $\sigma = 0.5$ than using $\sigma = 0.7$. Indeed, a smaller σ inserts less constraints to the linear system (since the kernel size is given by $d \times \sigma$), and, as a consequence, the system is more quickly solved.

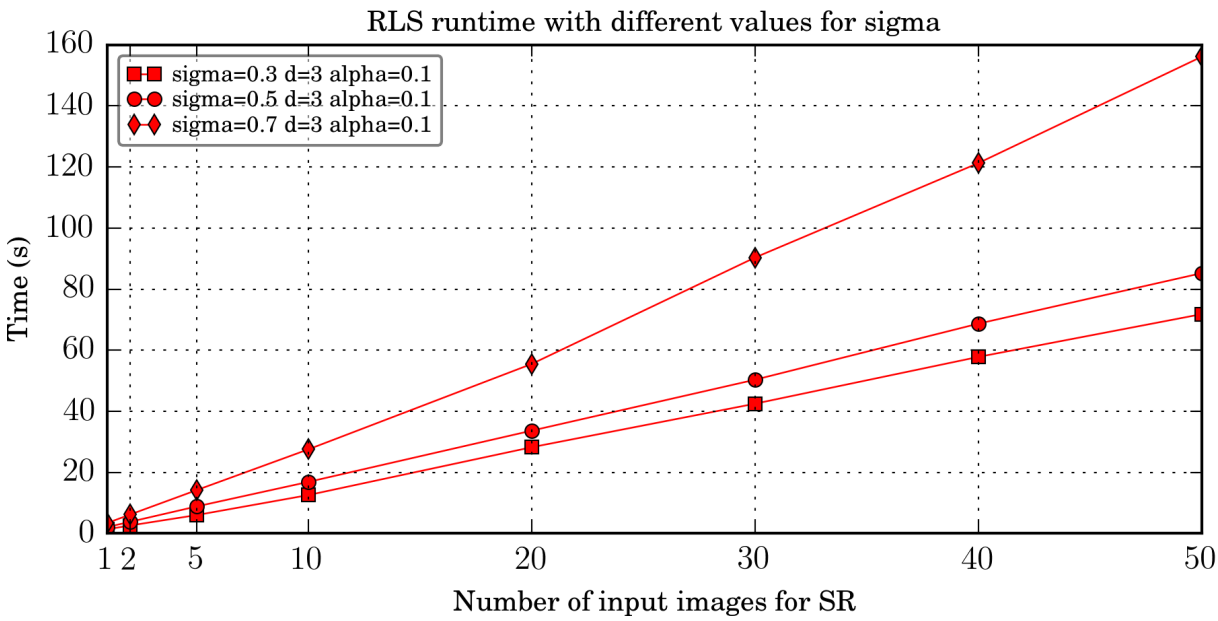


Figure 4.12: *RLS* runtime using different values for σ.

Similarly, the higher the value of d, the slower the runtime to solve the system (see Fig. 4.13). Hence, we suggest using $d = 3$ instead of $d = 5$, since the system becomes slower as we increase d but the quality of the solution does no increase.

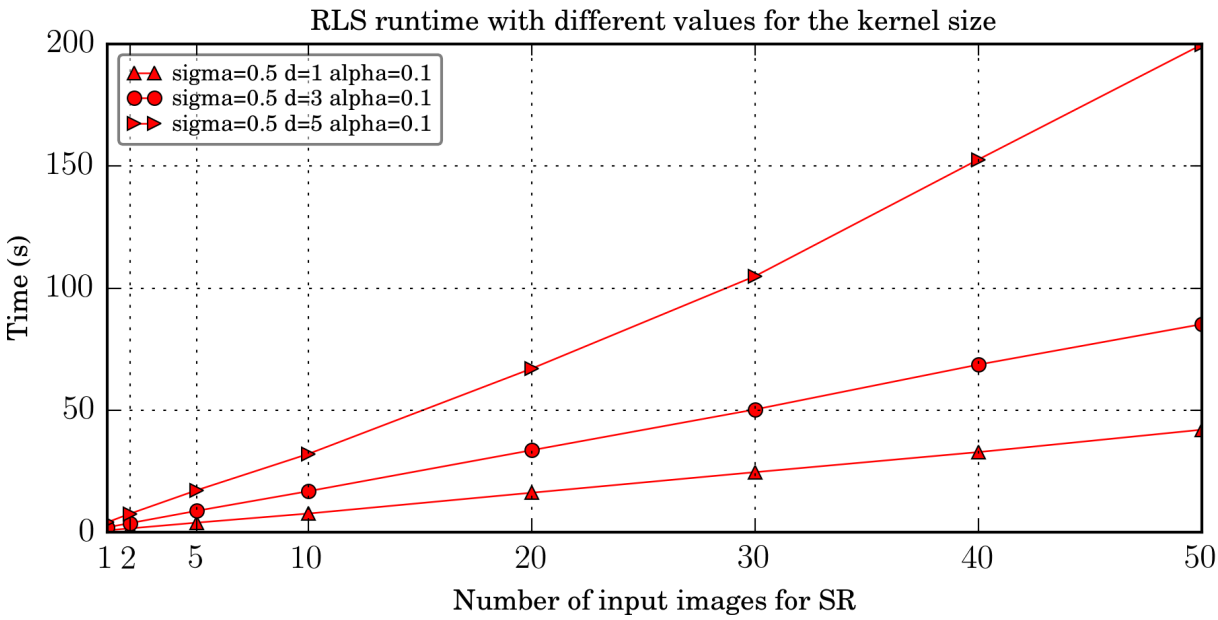


Figure 4.13: *RLS* runtime using different values for the kernel size.

As we previously discussed, in *RLSO*, we do not specify the parameter d. Instead, we divide the grid into imaginary cells, and the LR pixels inside each area contributes only to its closest grid pixel (see Chap. 3 for more details). Hence, we tune the second implementation selecting only the best values for σ and α. The best results for this optimization, on average, are found with $\sigma = 0.7$ and $\alpha = 0.1$ (the complete chart is in Appendix A).

In addition, for all $1 \leq n \leq 50$, and for all target images in the pool, the highest SSIM are also found using *RLSO* with $\sigma = 0.7$ and $\alpha = 0.1$. The results for such combination is statistically different from the results using all the other values. The chart in Fig. A.10 shows that the best results for this optimization are, on average, only slightly better than the blurred image. The Wilcoxon test indicates that the similarities produced by *RLSO* using such combination of parameters are statistically similar to I_G for $30 \leq n \leq 40$. Then, they become statistically different for $n \geq 50$, when *RLSO* outperforms the blurred image for the the most images of the dataset (as in the chart in Fig. A.10).

The regularization parameter impacts the runtime of *RLSO* as in the original model. The chart in Fig. 4.14 shows that the optimized *RLS* also runs in linear time as we increase the number of input frames, and that the execution time for $\alpha = 0.1$ is slower than for $\alpha \in \{0.5, 1.0\}$. However, Fig. 4.15 shows that σ does not affect the *RLSO* runtime as it impacts *RLS*.

In the original model, the kernel size is defined by $r = \sigma \times d$. Hence, we add more information into the linear system when we increase the standard deviation of the Gaussian distribution. In *RLSO*, we have a fixed-size kernel, so the execution time does not have a significant variation as we increase σ.

In the *RLSU* implementation, we must consider only the parameter α. The best value for such regularization parameter using the uniform distribution, on average, is $\alpha = 0.1$ (for the complete chart, see Appendix A). *RLSU* with $\alpha = 0.1$ is also statistically different than using $\alpha \in \{0.5, 1.0\}$. In addition, the images super-resolved by *RLSU* are

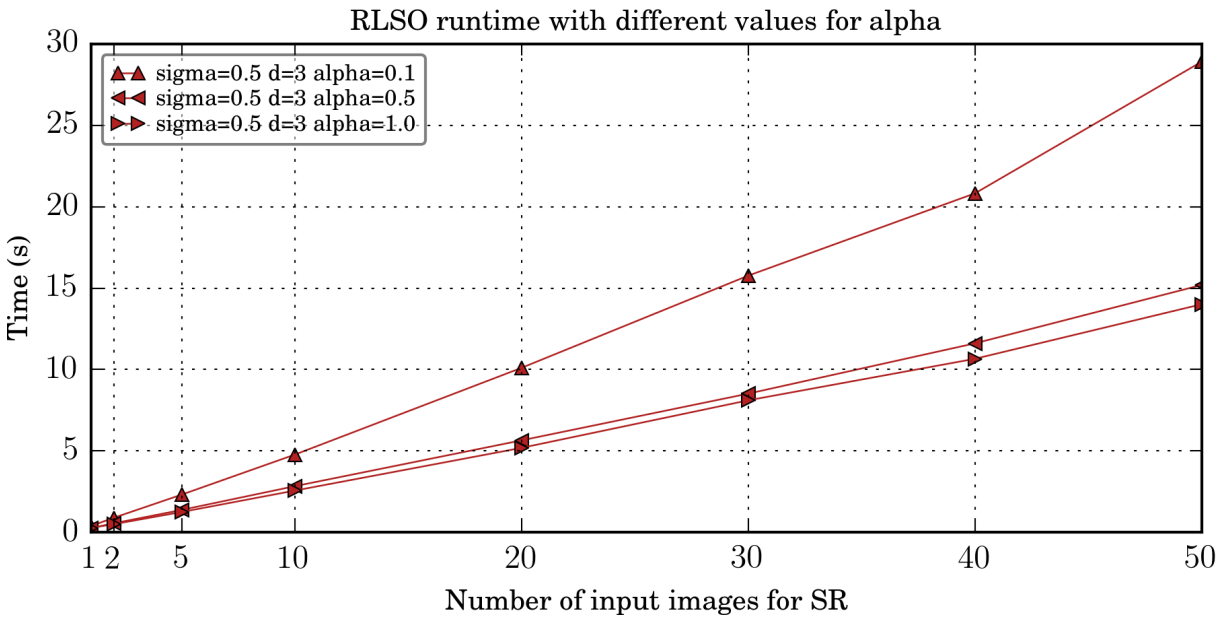


Figure 4.14: *RLSO* runtime using different values for α.

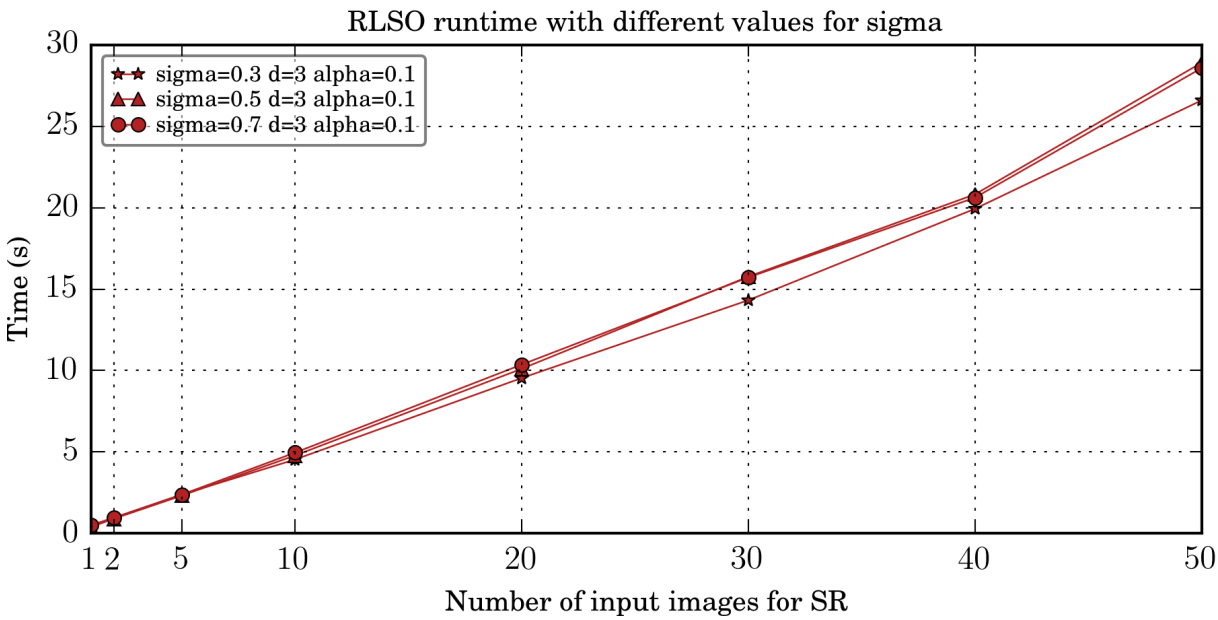


Figure 4.15: *RLSO* runtime using different values for σ.

more similar to the target I_{HR} than the blurred image I_G for $n \geq 30$, with statistical difference between them.

Finally, we see in Fig. 4.16 that the regularization parameter impacts the execution time of *RLSU* exactly as in the other two implementations.

In the light of the foregoing results of this first round, we conclude that the regularization parameter $\alpha = 0.1$ produces super-resolved images with the highest SSIMs in all the three implementations of *RLS*. In addition, we suggest using $\sigma = 0.5$ in both *RLS* and *RLSO*, and $d = 3$ in the original *RLS* implementation. In Round #2, we check which of the variations of *RLS* has the best reconstruction effectiveness.

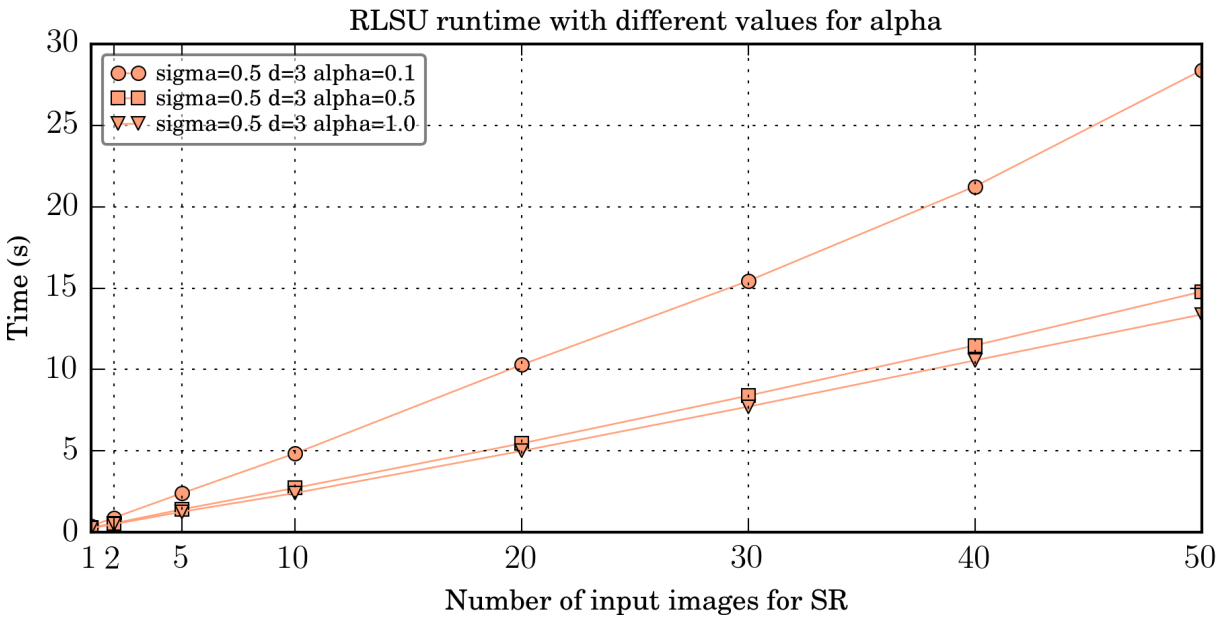


Figure 4.16: *RLSU* runtime using different values for α.

Round #2 - Best RLS variation

In this round, we summarize the quality of the three variations of *RLS*, on average, using the best combination of parameters that we found in the previous round. Each curve of the chart in Fig. 4.17 is the similarity for all the target images from the subset $\mathscr{D}_1$, on average. We have three setups to choose the best *RLS* method:

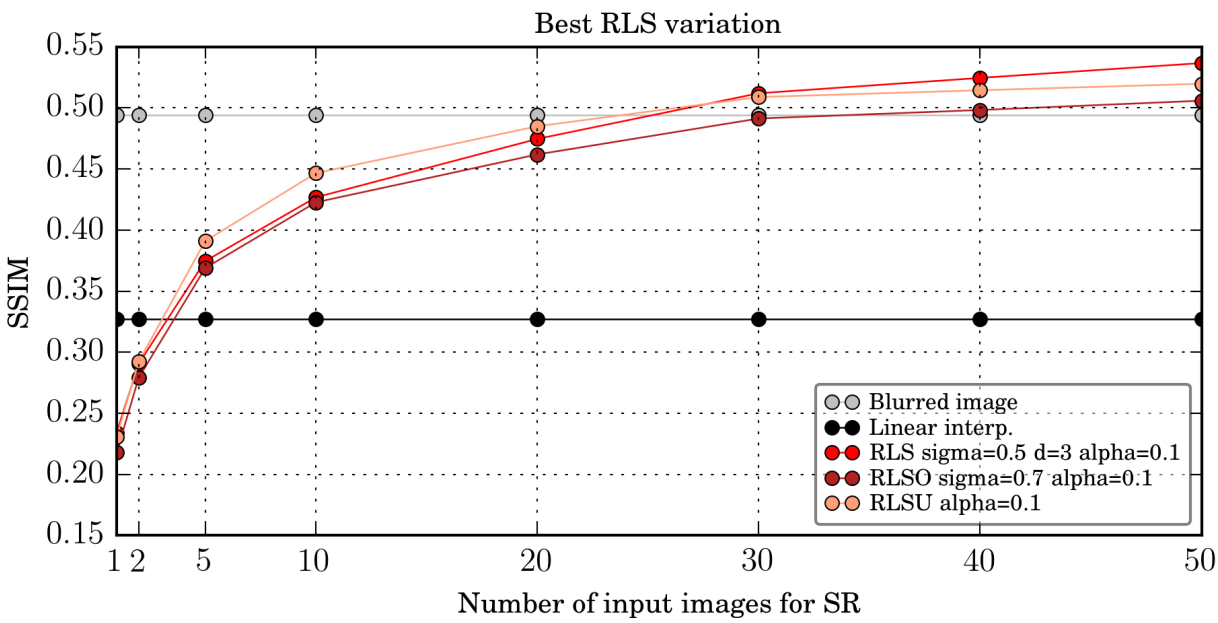


Figure 4.17: SSIM values for the three implementations of *RLS*.

- For $n < 30$, *RLSU* outperforms *RLSO* and the original *RLS* model, on average (as we see in Fig. 4.17). *RLSU* also found the best similarity values for the most target images in the pool, and it is statistically different from the other implementations (according to the Wilcoxon test).

- The *RLS* and *RLSU* results are similar for $n = 30$. Indeed, the Wilcoxon test identifies there is no statistical difference between such methods for this number of input images.

- Finally, if we super resolve at least 30 images, *RLS* not only becomes the best implementation on average (see Fig. 4.17), but also produces SSIMs higher and statistically different than *RLSO*, *RLSU*, and than the blurred image I_G.

The original *RLS* model generates images with higher SSIMs for $n \geq 30$, but it is almost three times slower than the two optimizations. The chart in Fig. 4.18 shows the execution time for all *RLS* variations. Hence, *RLSU* might be a good solution, since its results are only slightly worse than *RLS*, and its execution time is significantly slower. It is worth pointing out that *RLSU* does not use the Gaussian filtering to create the linear system, but its results are better than *RLSO* (even using input images that have been previously blurred with the Gaussian filter). Hence, the algorithm might be useful even when we do not know much about the filtering in the downsampling process.

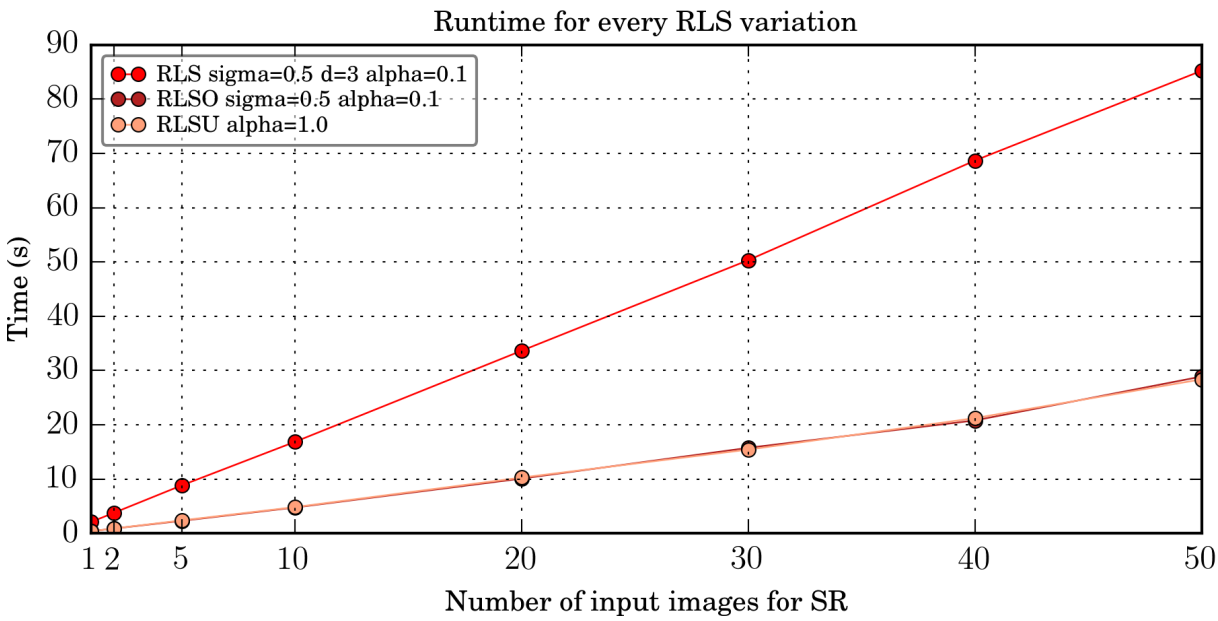


Figure 4.18: Runtime for each implementation of *RLS*.

Round #3 - Best ISR variation

Now, we turn our attention to the inpainting-based super-resolution method itself. As we discussed in Chap. 3, *ISR* is not designed to recover the high-frequency components lost during the blurring process. Therefore, we do not expect that the inpainting-based algorithms to produce SSIMs higher than I_G. Fig. 4.19 shows that the quality of the first variation is higher, on average. When we verify which *ISR* implementation produces higher SSIMs for the most target images in the pool, ISR_1 is also the best variation in 100% of the target images. Additionally, its results are statistically different from ISR_2 for all $1 \leq n \leq 50$.

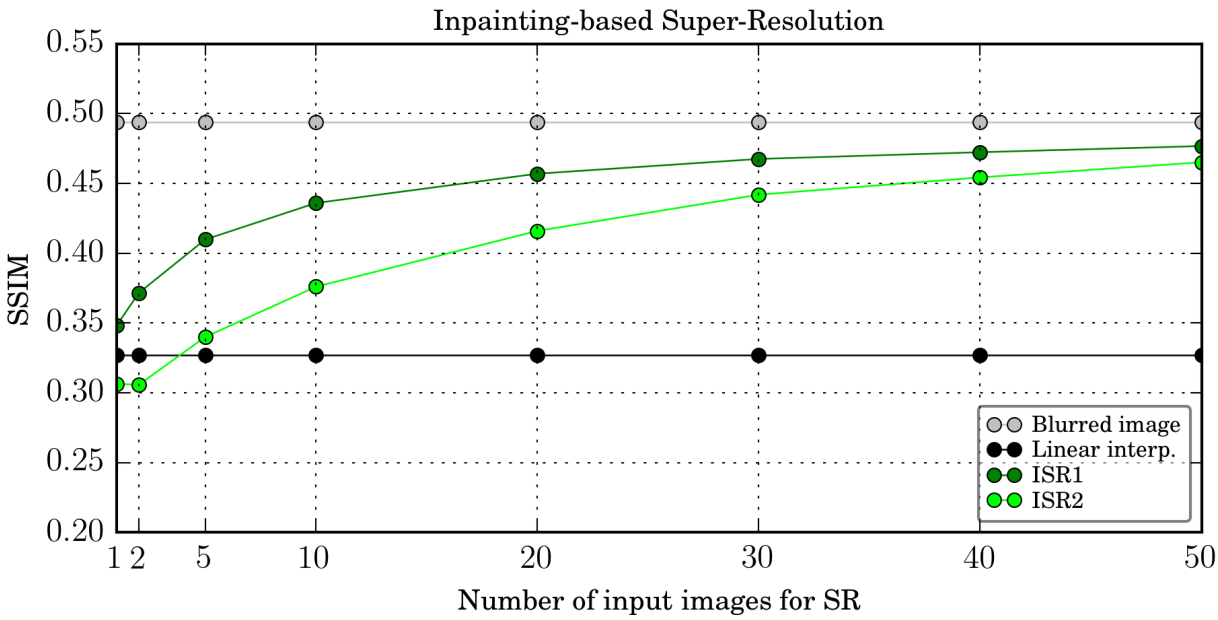


Figure 4.19: SSIM values for the two inpainting-based methods for super resolution.

We see in Fig. 4.20 that the inpainting-based methods do not run in linear time as we increase the number of input frames to super resolve. If we have 50 LR images to reconstruct, for example, there may be less unknown pixels to fill in than if we have only 20 input observations. The less missing pixels in the grid, the faster the super resolution may finish. Moreover, ISR_1 not only produces super-resolved images with higher quality than ISR_2, but is also faster.

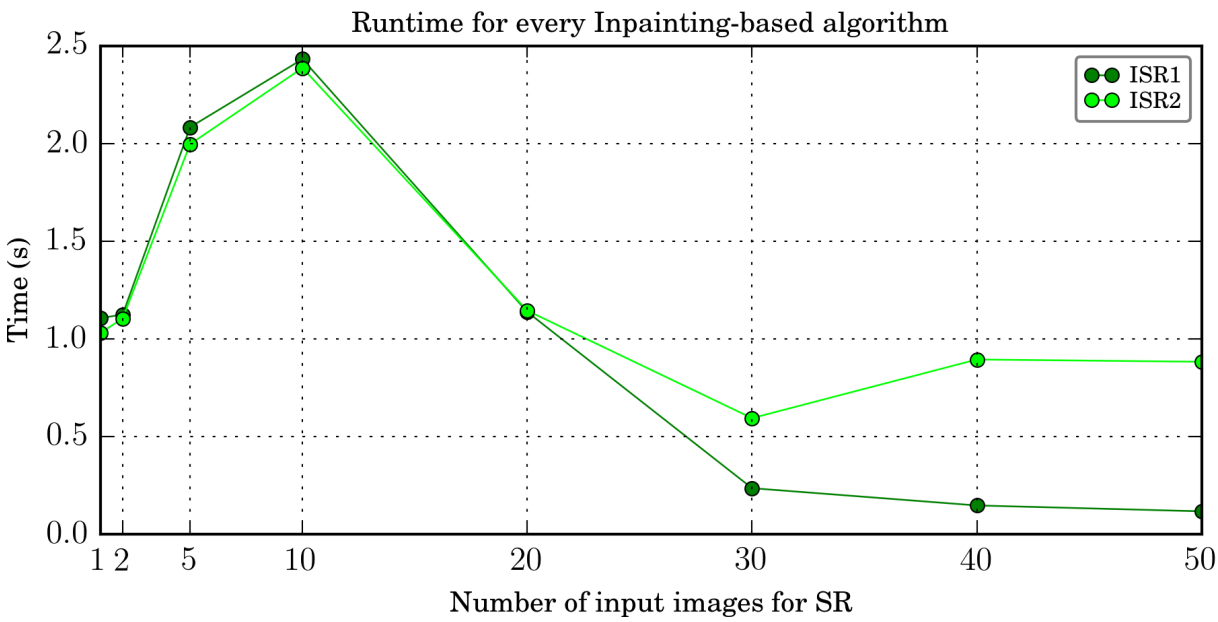


Figure 4.20: Runtime for each implementation of ISR.

Round #4 - Best GSR variation

Fig. 4.21 shows the quality of the images super resolved by the five variations of the Geometric k-Nearest Neighbors Multi-Frame Super-Resolution.

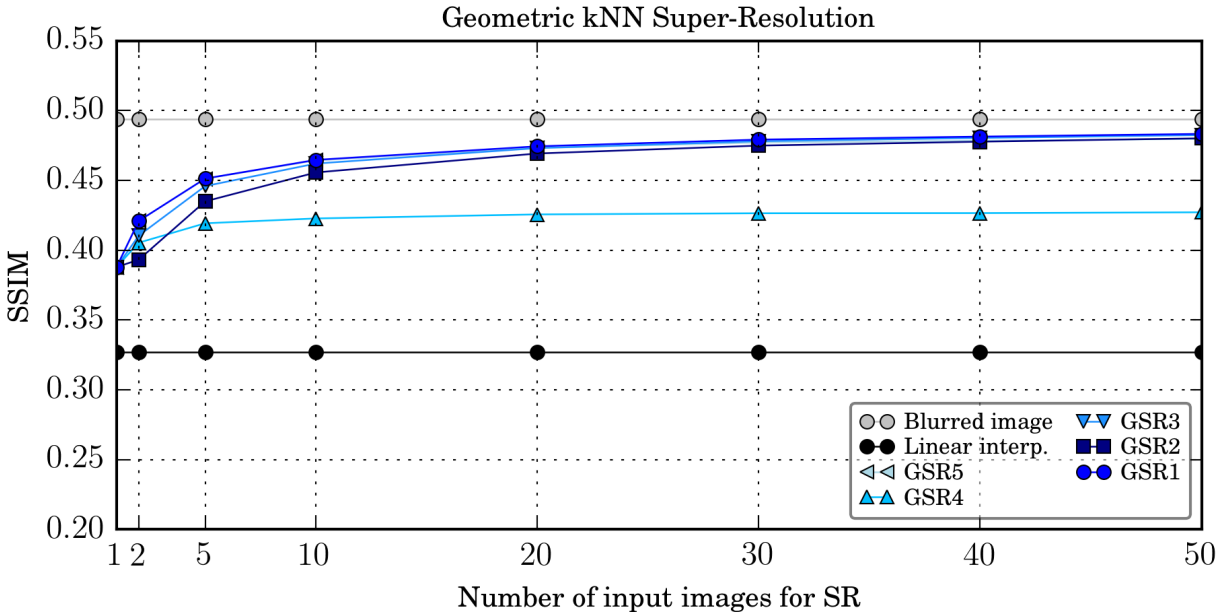


Figure 4.21: SSIM values for the five variations of GSR.

The GSR results are all very similar, on average (except for GSR_4, with the worst results). The similarities found by GSR_1 and GSR_3 in the chart are only slightly higher than GSR_2 and GSR_5 in the chart. However, the Wilcoxon test indicates statistical differences from GSR_1 to all the other variations, and this first implementation found higher SSIMs in the most target images for all $1 \leq n \leq 50$.

The first variation is also the fastest one, as we see in Fig. 4.22. In addition, similarly to RLS, the execution time for all the five variations of GSR grows linearly when we increase the number of input images. The chart in Fig. 4.22 also shows that the all GSR variations can super resolve up to 50 LR images from the subset $\mathscr{D}_1$ in at most one second, on average.

Round #5 - Best reconstruction method

We then plot the execution time for RLS, $RLSO$, $RLSU$, ISR_1 (the best inpainting-based variations) and GSR_1 (the best implementation of GSR) altogether in Fig. 4.23. For this setup, GSR_1 is two orders of magnitude faster than RLS, on average.

By contrast, RLS outperforms both ISR and GSR when we compare the quality of the solution in Fig. 4.24 (see Appendix A for the comparison using PSNR and MSE).

We see in Fig. 4.24 that only the algorithms based on the Tikhonov regularization produce images with higher similarity values than the blurred image I_G (i.e., only the three RLS variations can recover high-frequency components of the target image that have been lost during the blurring process). Comparing all the methods, we have the following setups for the best super-resolution method:

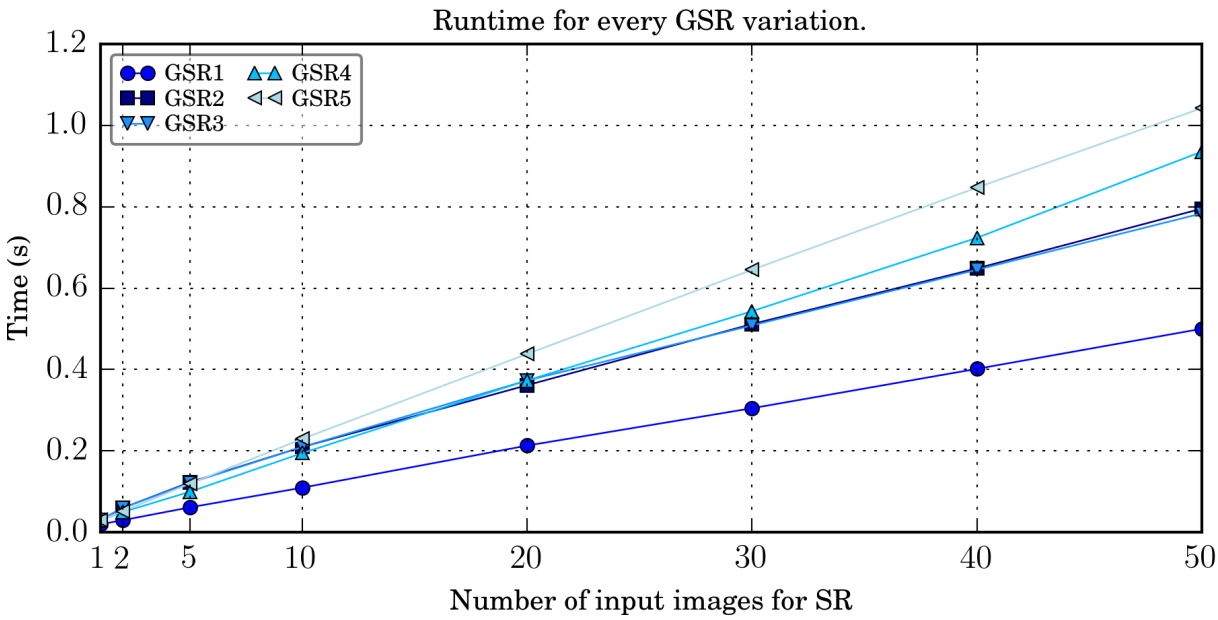


Figure 4.22: Runtime for each implementation of GSR.

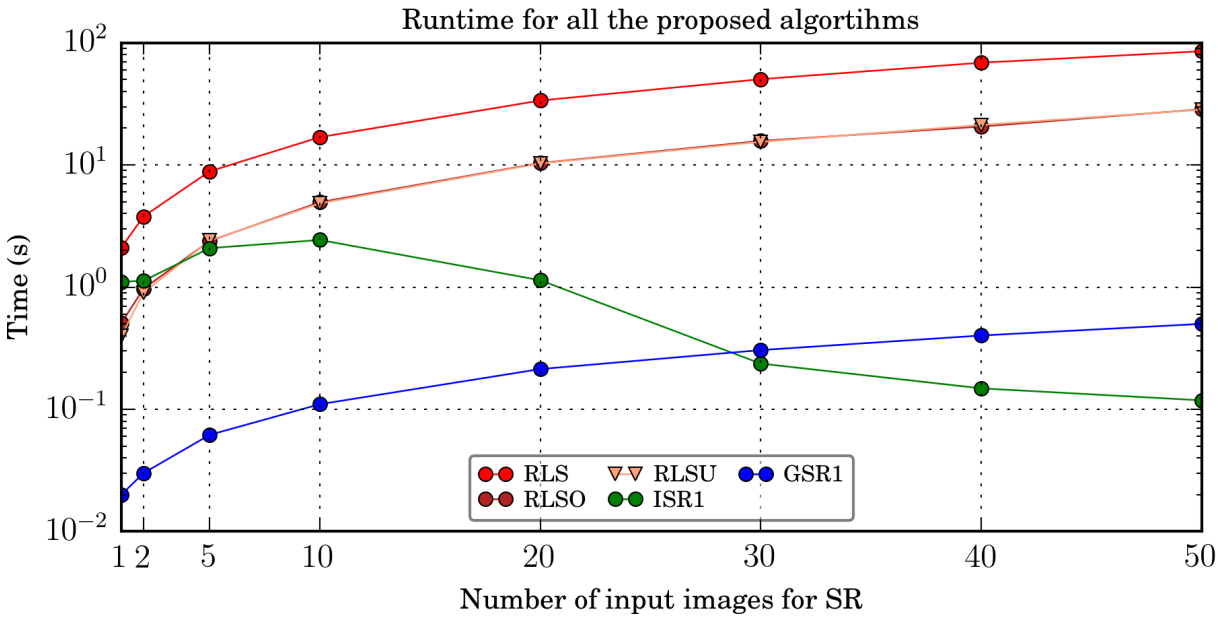


Figure 4.23: Runtime for RLS, ISR and GSR.

- For $n < 20$, GSR_1 produces super-resolved images more similar to the original target image I_{HR} than all the other methods, with statistical difference. Therefore, GSR might be the best choice for super-resolving just a few images.
- For $n = 20$, $RLSU$ finds the highest similarities.
- The best results for $n = 30$ are found with RLS and $RLSU$ (there is no statistical difference between them).
- Finally, if we super resolve more than 30 images, RLS produces the highest SSIM values (even if we compare it to I_G), with statistical different. In addition, $RLSU$ is still very competitive.

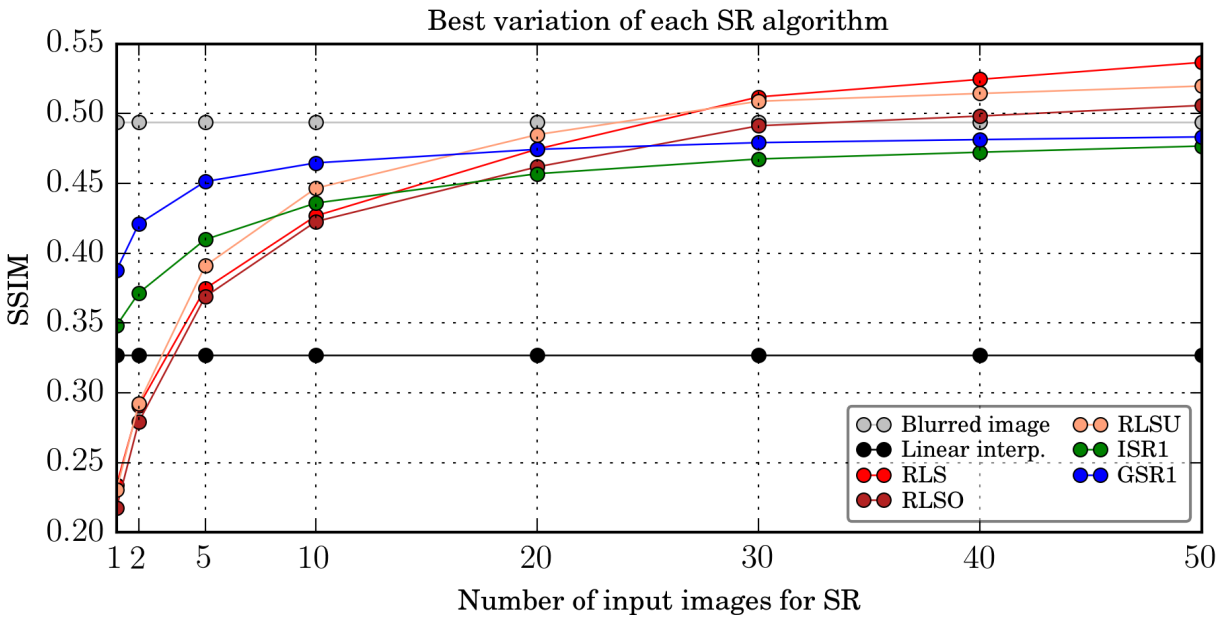


Figure 4.24: Comparison among RLS, ISR and GSR.

The high efficiency of GSR lets us execute the algorithm also using images from the subset $\mathscr{D}_2$ as input. It is the only proposed method that is adequate for high-resolution input images. We see in Fig. 4.25 that all the variations run quickly and linearly even when we super resolve such input images with higher dimensions. On average, we can combine 50 observations with dimensions $6,000 \times 4,000$ pixels into an image I_{SR} of size $24,000 \times 16,000$ pixels in less than one minute (only 20 seconds if we use GSR_1).

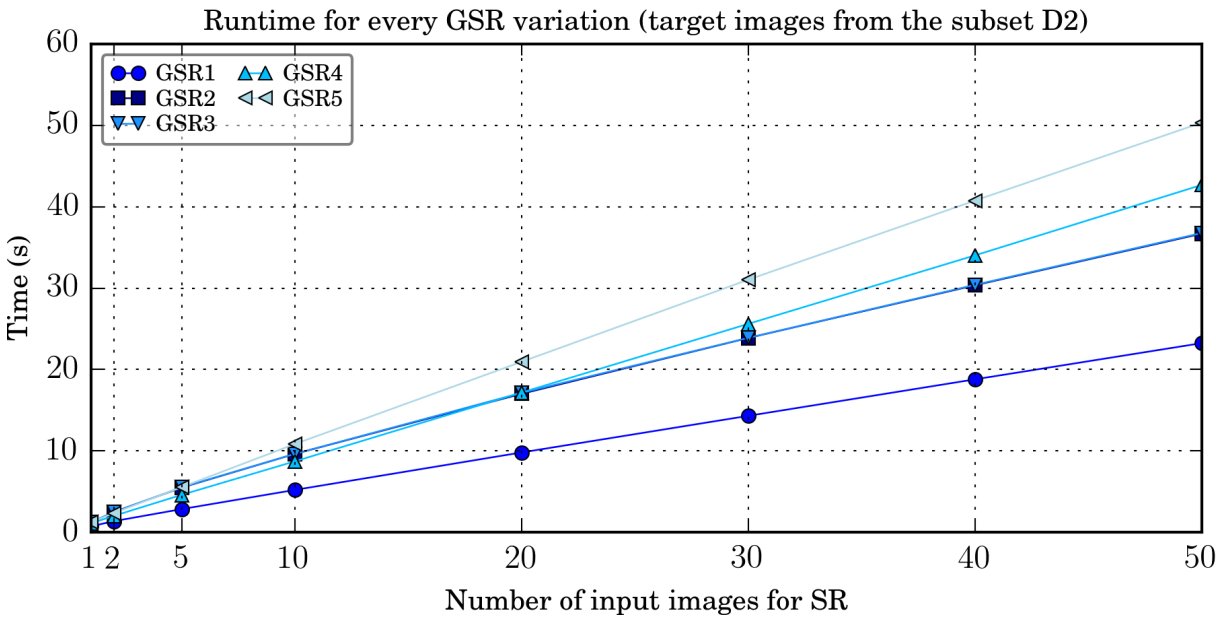


Figure 4.25: Runtime for each implementation of GSR using only images from the subset $\mathscr{D}_2$ from the Dataset #1.

Fig. 4.26 compares the runtime of GSR in logarithmic scale to ISR_1. Note that, due to the higher dimensions of the images from the subset $\mathscr{D}_2$, we could calculate the super resolution only for at least 30 input observations with the inpainting-based method in

such setup (the more images to super resolve, the faster is the ISR execution time, as we previously discussed in this section, since we have less missing pixels to inpaint). Only GSR, among all methods proposed in Chap. 3, has shown to be adequate for input images with such high dimensions.

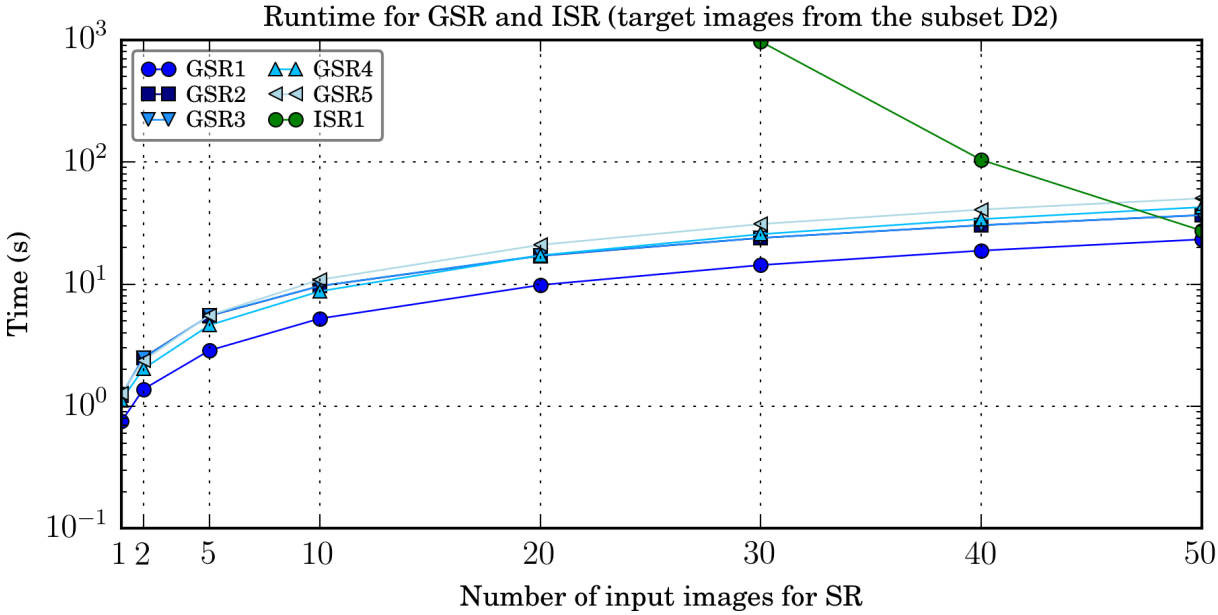


Figure 4.26: Runtime for ISR_1 and each implementation of GSR, using only images from the subset $\mathscr{D}_2$ from the Dataset #1.

Moreover, it is important to point out that RLS assumes that the low-resolution observations undergo a blurring process, prior to the downsampling, to avoid the aliasing. The goal of such algorithm is to reconstruct the original target image I_{HR}, while ISR and GSR aim to reconstruct exactly the image that generated the LR images in the pool (I_G, in all the previous setups in this experiment). Nevertheless, if we do not know details about the formation of the images (or if we know that the original image has not been blurred), GSR might be the best solution.

In Fig. 4.27, we reconstruct the same LR images than in the previous chart, but we measure the similarity between each super-resolved image I_{SR} to the blurred image I_G. In other words, we forego the anti-aliasing filter, and calculate the quality of the reconstruction with respect to the image to which we applied the downscaling and the affine transformations. We can see in Fig. 4.27 the SSIMs produced by GSR_1 are, on average, almost 1.0 (the maximum value for this similarity metric). Such method is also statistically different from the others. Additionally, we note in this chart that, now, the original RLS model has the worst results among the proposed methods. It happens because the image RLS is aiming to reconstruct is not the one we are comparing to. For all those reasons, we emphasize that the best solution to super resolve a set of images is highly dependent on the application. This is mainly due to the different constraints imposed on the problem under different setups.

Hence, we perform a last set of experiments in this round:

- First, we select an HR image I_{HR} from the subset $\mathscr{D}_2$ (the same image as in Fig. 4.6) to be the target for the super resolution.

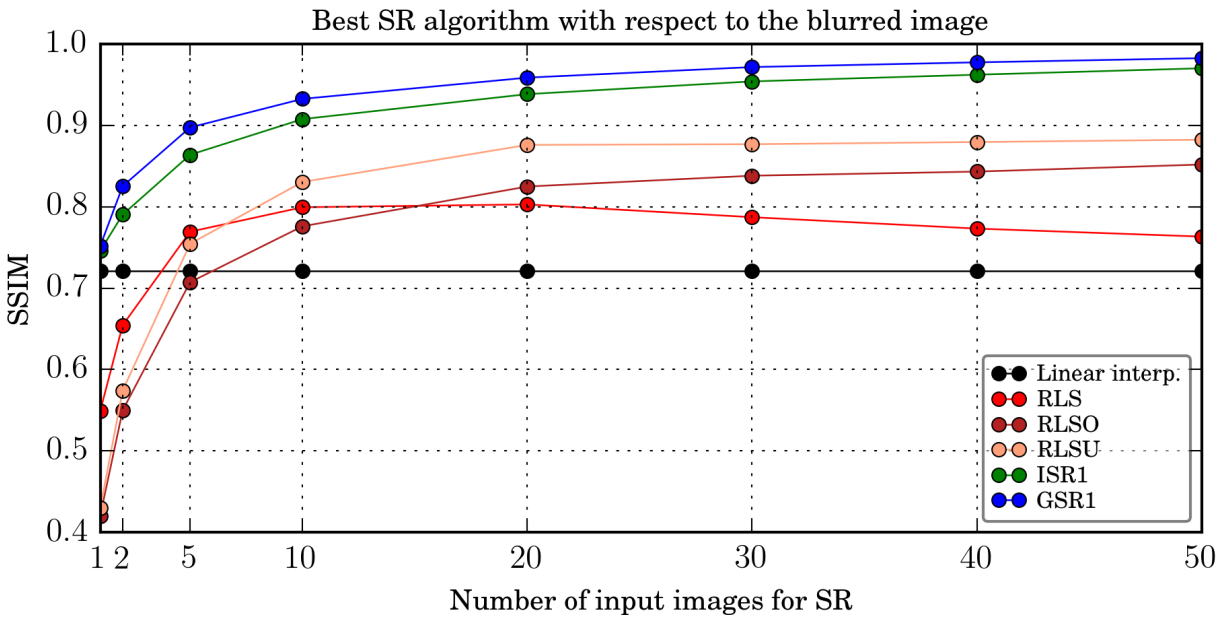


Figure 4.27: Comparison among RLS, ISR and GSR, with respect to the blurred image that generated the pool of LR images.

- Then we use the affine transformations (rotations, translations and scale), to create the pool of LR images, without applying the Gaussian blur in the target image. In this case, there is no image I_G between I_{HR} and the LR observations.
- We select only the fastest methods (GSR and ISR) to super resolve the LR images, since the images in $\mathscr{D}_2$ have higher dimensions.

We can visualize the results in Fig. 4.28 for super resolving $n = 50$ LR input images with GSR_1 and GSR_3 (the two best results among the GSR variations), GSR_4 (the worst GSR variations), and also ISR_1. For a better visualization of the reconstructed details, Figure 4.28 shows only a piece of the HR images. The green rectangle in Fig. 4.28g shows such piece in one of the LR images. We can see that even our worst result is visually more accurate than the simple interpolation. In addition, our best results are very similar to the target image.

Fig. 4.29 shows the memory footprint of the proposed methods. All variations of GSR require approximately the same memory, so we plot only GSR_1 in the chart. Likewise, we plot only the first inpainting-based method, as the memory footprint of ISR_2 is similar to ISR_1. RLSO also requires approximately the same memory as RLSU, then we plot only RLSU in the chart. We notice in the chart that the memory footprint of GSR and ISR do not change as we increase the number of input images. In addition, the memory footprint of the RLS optimizations is two orders of magnitude lower than the original model, but, even so, such algorithms are not competitive with respect to GSR and ISR in terms of memory footprint.

Round #6 - Comparison to other methods in the literature

Finally, we compare our algorithms to other reconstruction methods available in the literature. From all the multi-frame super-resolution methods proposed between 2015

(a) GSR_1, SSIM=0.97

(b) GSR_3, SSIM=0.97

(c) GSR_4, SSIM=0.80

(d) ISR_1, SSIM=0.96

(e) NN Interp., SSIM=0.46

(f) Target image

(g) LR image

Figure 4.28: 50 LR images from the subset $\mathscr{D}_2$ are super resolved by GSR_1 (a), GSR_3 (b), GSR_4 (c), and ISR_1 (d). In (e), an LR image zoomed in using nearest-neighbor interpolation. In (f), the target image. In (g), one of the LR images. The green rectangle highlights the piece of the image that we show from (a) to (f).

and 2017 [103, 131, 40, 126, 50, 74, 44, 51, 71, 60, 62, 55] that we cited in Chap. 2, only Maiseli et al. [50], Köhler et al. [60] and Kato et al. [55] provided us with their source codes for our experiment. However, the work in [55] considers only shifts between LR observations. Therefore, we do not use such algorithm in this experiment, since the images in our dataset differ from each other by both translations and rotations.

We include four methods of the literature in our comparison (see comparison with

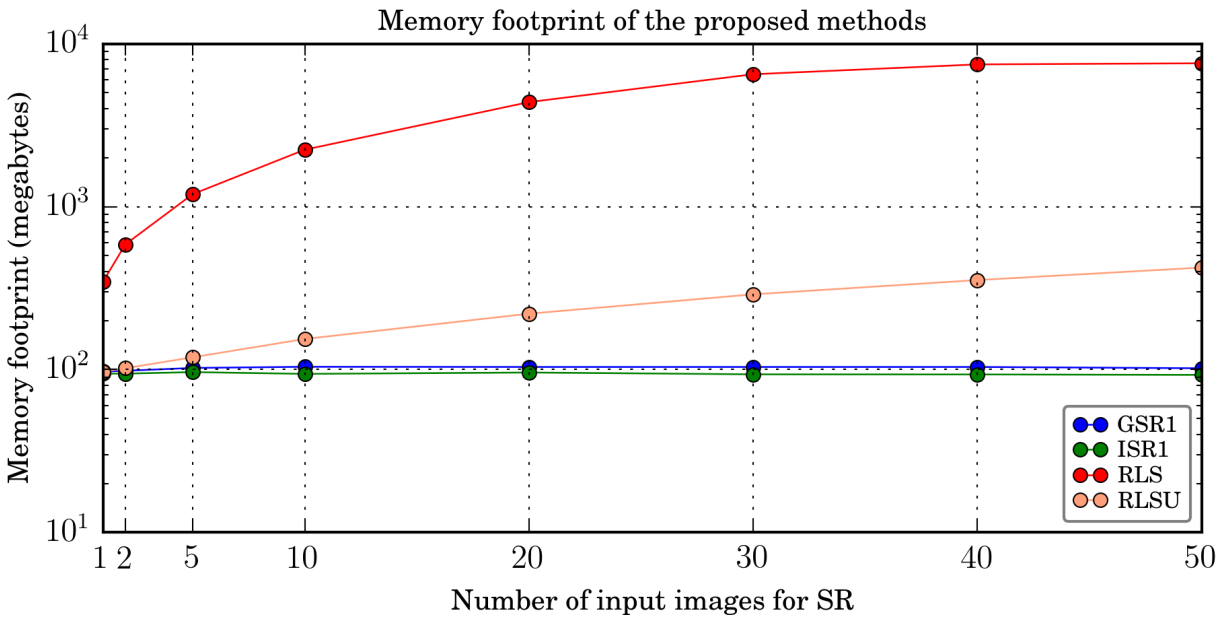


Figure 4.29: Memory footprint of the proposed methods.

other traditional SR methods in Appendix A):

- The recent work of Köhler et al., an iteratively re-weighted optimization for robust super resolution [60]. We refer to this method as "Kohler 2016".
- The SR method proposed by Maiseli et al. [50], that uses an adaptive diffusion-based regularizer. We refer to such method as "Maiseli 2015".
- The work of Pham et al. [86] (implemented by Vandewalle et al. in [122]), a Direct Method that uses a structure-adaptive normalized convolution. Such method was shown to outperform traditional methods based on IBP [45, 46], PG [83, 41], Robust Super Resolution [138], and POCS [109, 34, 136] (see Section 2.2 for more details about these methods). We refer to this method as "Pham 2006".
- A traditional MAP-based algorithm implemented by Köhler et al. [60], that uses Gaussian filtering to write the system of linear equations (such as our *RLS* algorithm) and solve the system using Scaled conjugate gradient optimization [9]. We refer to this method as "MAP-based".

Only "Pham 2006" produces RGB images. The other three methods work only for grayscale images. In this setup, we use up to 50 input images as input to reconstruct an image 4× higher in both axes. The chart, in Fig. 4.30, shows that these four implementations produce SSIM values lower than *RLS*, ISR_1, and GSR_1. The work of Maiseli et al. [50] does not work correctly for more than 10 input images, with the provided implementation.

Fig. 4.31 shows the method of Köhler et al. [60], which produces the best SSIMs among the literature methods in Fig. 4.30, is also the slowest one (four orders of magnitude slower than GSR_1) in this experimental setup. The running time of the other methods are more

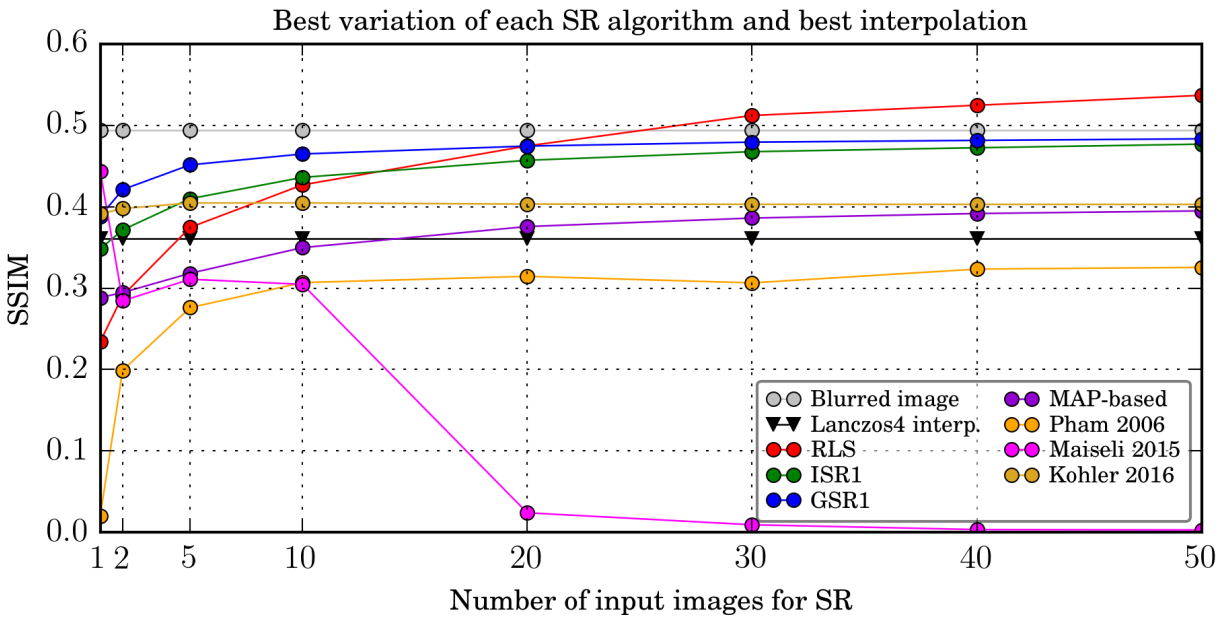


Figure 4.30: Quality of different reconstruction algorithms as a function of the number of LR input images.

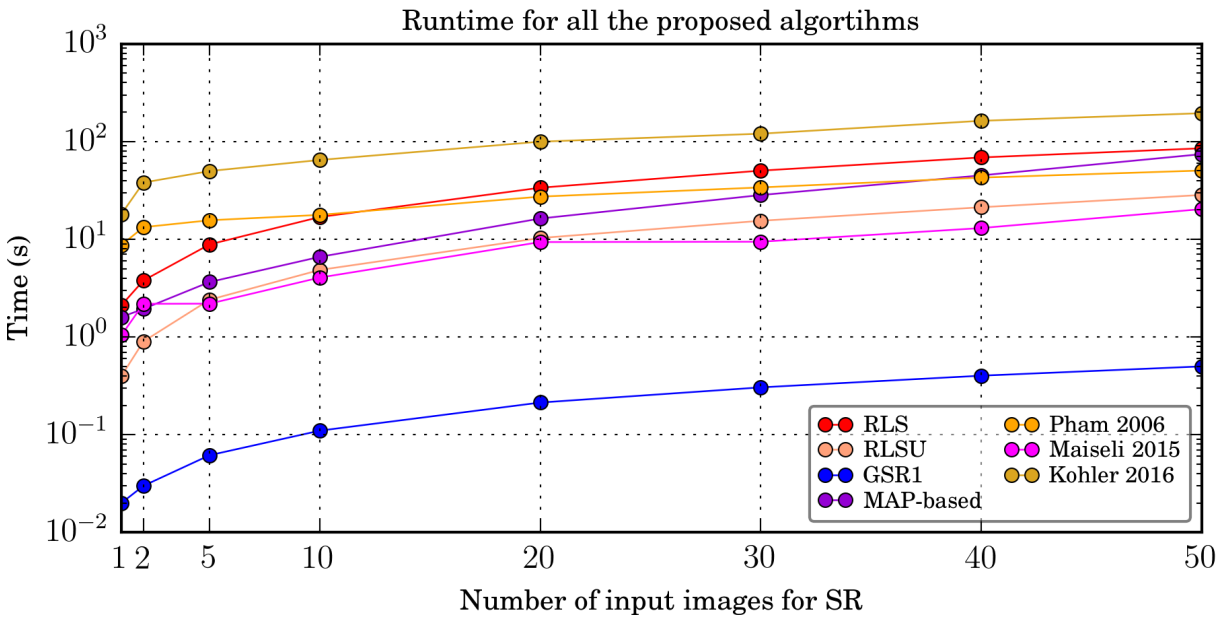


Figure 4.31: Quality of different reconstruction algorithms as a function of the number of LR input images.

similar to our RLS variations. As expected, GSR_1 is the fastest method of the chart in 4.31.

In Fig. 4.32, for the sake of completeness, we compare our algorithms to single-frame interpolations: (1) Nearest neighbor interpolation; (2) Bilinear interpolation; (3) Bicubic interpolation (4×4 pixel neighborhood); and (4) Lanczos interpolation (8×8 pixel neighborhood).

Finally, in Fig. 4.33, we visually compare our results (using 30 input images) to "Kohler 2016", to "Pham 2006" (both using also 30 LR images) and to a linear interpolation.

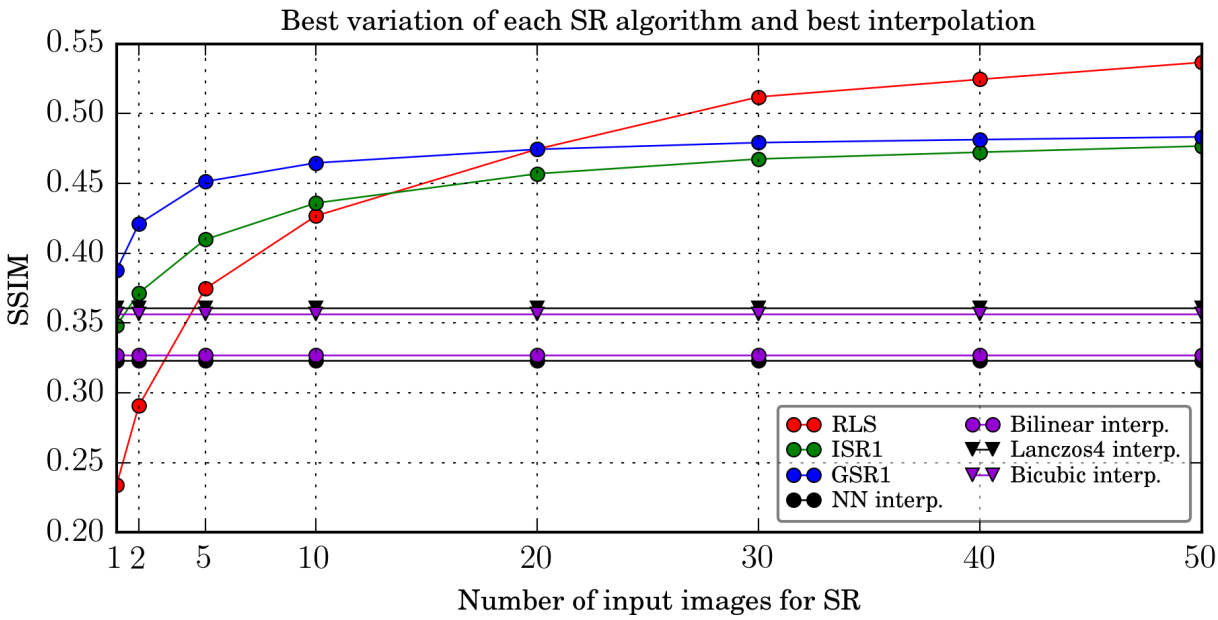


Figure 4.32: Our super-resolution methods versus single-frame interpolations.

(a) Target image **(b)** *RLS*, SSIM=0.71 **(c)** GSR_1, SSIM=0.62

(d) Linear Interp., SSIM=0.45 **(e)** Pham 2016, SSIM=0.42 **(f)** Kohler 2016, SSIM=0.51

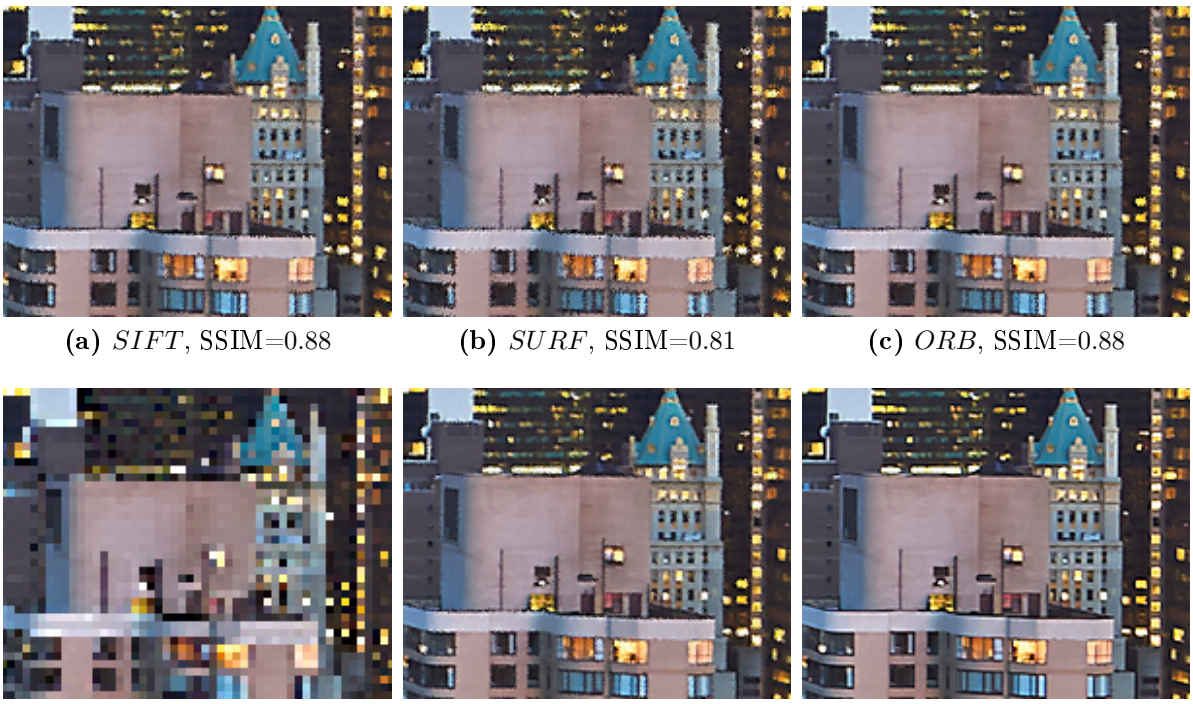
Figure 4.33: Visual comparison among our results (using 30 input images) to "Kohler 2016", to "Pham 2006" (both using also 30 LR images) and to a linear interpolation.

Round #7 - Registration

We now bring to bear the discussion about the most appropriate registration method to be used along with the super-resolution methods previously presented and validated. For a fair comparison among the three feature detectors we present in this work (SIFT, SURB, and ORB), we choose only one algorithm for the reconstruction step in each chart of this last round. Hence, different SSIM values in the results are exclusively due to the registration step. Additionally, in all charts, we also plot the quality using the correct

alignment, to visualize the best results that the registration methods could achieve. As we previously discussed, it is worth pointing out that such correct alignment might be only available in lab conditions, for comparison purposes.

First, Figure 4.34 shows a visual comparison among SIFT, SURF, ORB and the correct registration, using 50 LR images and GSR_1. The target image here is from the subset $\mathscr{D}_2$, and I_{HR} has not undergone the blurring process (as in Fig. 4.28). We also show the similarity between each reconstructed image I_{SR} to the original target image I_{HR}.



(a) *SIFT*, SSIM=0.88 **(b)** *SURF*, SSIM=0.81 **(c)** *ORB*, SSIM=0.88

(d) NN Interp., SSIM=0.27 **(e)** Correct, SSIM=0.93 **(f)** Reference image

Figure 4.34: 50 LR images from the subset $\mathscr{D}_2$ are super resolved by GSR_1 using SIFT (a), SURF (b), ORB (c) and the correct alignment (d). In (e), an LR image zoomed in using nearest-neighbor interpolation. In (f), the target HR image.

Fig. 4.35 shows the quality of the super resolution, on average, for all target images from the subset $\mathscr{D}_2$ (the target images with the highest dimensions in the Dataset #1), using GSR_1 in the reconstruction step. The similarity values found by SIFT, in the chart, are very similar to the correct registration, which are, theoretically, the best possible results of a registration method.

The Wilcoxon test indicates that, indeed, SIFT produces the highest SSIMs for the most target images in the pool (100% of the images for $n > 10$ LR images), and its results are statistically different from the others for all $n > 1$. The same happens when we compare the similarity of the super-resolved image I_{SR} to the blurred image I_G, or when we use the other reconstruction methods. In addition, only SIFT could properly align the images from $\mathscr{D}_1$ (with smaller dimensions). However, Fig. 4.36 shows that SIFT runtime is, on average, significantly higher than the other feature detectors for each pair of images.

Putting the efficiency and effectiveness of the feature detectors into perspective, the

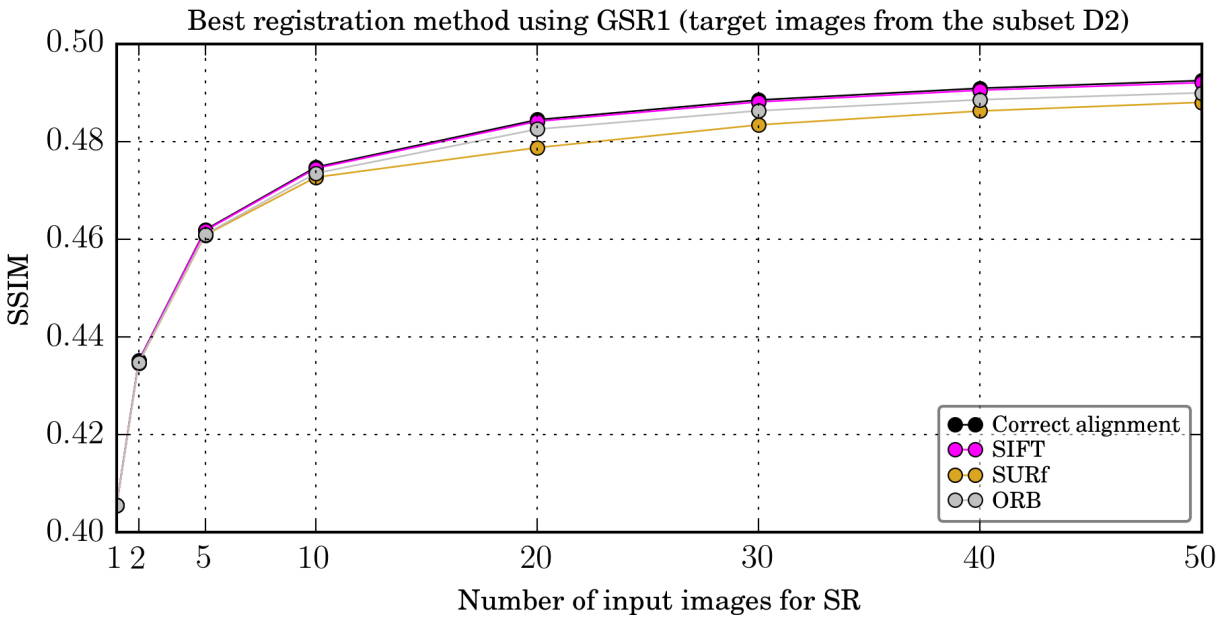


Figure 4.35: Best registration method using GSR in the reconstruction step and images in the subset $\mathscr{D}_2$ from the Dataset #1.

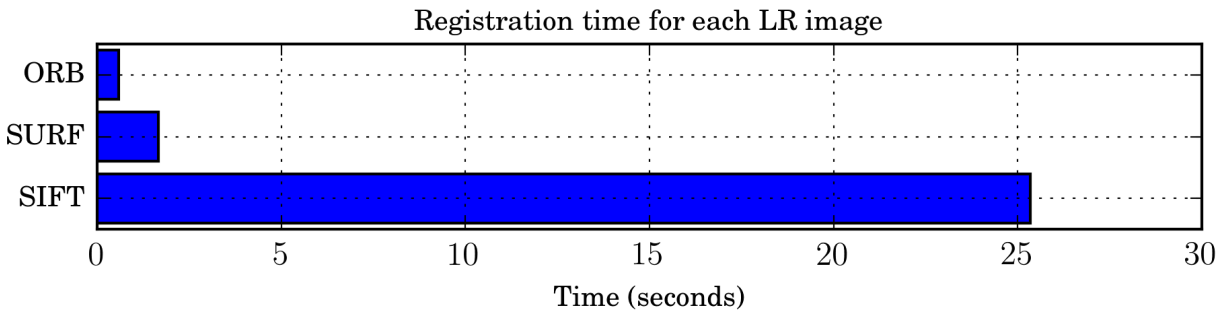


Figure 4.36: Runtime for the registration methods, using only images in the subset $\mathscr{D}_2$ from the Dataset #1.

results show that ORB might be an appropriate solution for our problem. Such SSIM values are very competitive (if compared to SIFT, which produces the highest values), and it is the fastest solution.

4.2 Experiment #2: Application on mobile devices

Finally, we apply the proposed super-resolution algorithms to an always-on low-power environment. In this application, we super resolve a sequence of photos taken by recent mobile phones, that can capture many images per second while somebody is holding the camera manually in approximately the same position. The small camera motion introduces new information between consecutive images, and we use it to reconstruct an HR image. In doing so, we allow the user to take multiple images with a cheap camera, and combine them to a higher resolution image (as if the picture was taken with a more expensive device).

We assume, in this application, that the objects in the scene remain static while cap-

turing the images. In addition, we super resolve only plain objects, because the proposed methods are not designed to handle 3D reconstructions. Moreover, the registration step in this application must find perspective transformations between two input images, as the user handling the camera may not move her hand with strict rotations and translations. Such alignment must be calculated with subpixel accuracy.

The solution for this application requires fast responses. A person who takes a dozen photos in one second in a mobile device may not be willing to wait for minutes until a high-resolution image is generated. The *Timeshift Burst* application available for *Android 4.2+*, for example, can capture up to 30 images in a second, while *iPhone 6* and *7* can gather up to 10 pictures during this time. In addition, the application must have low memory footprint, because the person may simultaneously receive and make voice calls, send messages, take, receive and send photos and videos, listen to music and/or play games while waiting for the reconstructed image. Therefore, we suggest using the GSR method for this application, due to its fast running time (see Sec. 4.1.4) and low-memory footprint (Secs. 3.3 and 4.1.4).

4.2.1 Dataset #2

The dataset for this application contains a series of real-world HR images, taken by an iPhone 6 in burst mode. We captured at least 25 HR observations of each scene, and we refer to each observation as H_k. Such HR images H_k are combined by the super-resolution algorithms onto an even higher image. We refer to such final image as a Super-High-Resolution image I_{SHR} (see Fig. 4.37).

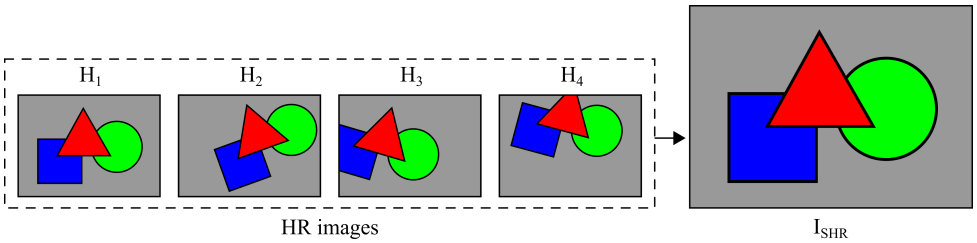


Figure 4.37: The HR images H_k are combined onto an even higher image. We refer to this final image as a Super-High-Resolution image I_{SHR}.

Additionally, we create an LR version for the HR images in the pool. Each H_k is downsampled, generating an LR image L_k. This pool of LR images is depicted in Fig. 4.38. In doing so, we have a pool of LR images that differ from each other by perspective transformations, and not only rotations and translations as in the *Experiment #1*.

If we super resolve the LR images L_k in Fig. 4.38, any HR image H_k might be used as a target for the reconstruction, and then we can visually validate if our algorithms can handle perspective transformations. Therefore, the Dataset #2 contains two pools of images that might be input for the super resolution: (1) a set of LR images L_k that might be combined onto a super-resolved image I_{SR} (Figure 4.38); and (2) a set of HR images H_k that might be super resolved onto a Super-High-Resolution image I_{SHR} (Figure 4.37).

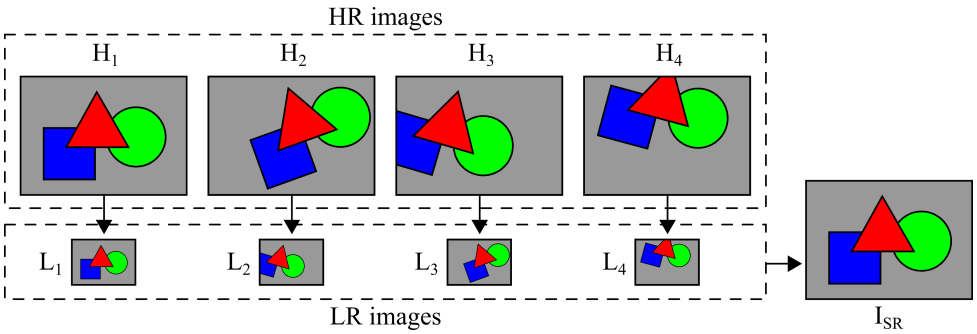


Figure 4.38: Each LR image L_k is a downsampled version of the HR image H_k. The LR images differ from each other not only by rotation and translation, but by a perspective transformation. Every HR image H_k might be a target to validate the reconstruction.

4.2.2 Experimental methodology and results

In a first round, we use 25 images H_k with the maximum resolution available in the device ($2,448 \times 3,264$ pixels) to generate 25 LR versions of these images (one per image). The size of each L_k is 25% of the size of H_k in both axis. Unlike *Experiment #1* (Sec. 4.1), now each LR image comes from a different HR image. Hence, any H_k can be a target for the reconstruction. We then super resolve the LR images and create a 4× higher image I_{SR}. Figure 4.39 shows small pieces of the results.

The five GSR variations take up to 50 seconds to super resolve 25 LR images (GSR_1, the fastest one, spends only 25 seconds). In the registration step, ORB aligns two images in 1 second, SURF in approximately 6 seconds, and $SIFT$ spends 26 seconds for each pair of input images. We see in Fig. 4.39 that all the five variations work correctly with the sequences of real photos. Each image in the first row of Fig. 4.39 is a piece of one of the target images in this dataset (i.e., an HR image that has been downsampled). Then, in the second row, we zoom in such LR image zoomed by a factor of four, using NN interpolation. Finally, the super-resolved image reconstructed by the different variations of GSR, and different registration methods.

Finally, in a second round, we super resolve the HR original images from the pool, generating a 2× higher image in both axis ($4,896 \times 6,528$ pixels). Figure 4.40 depicts small pieces of a result, using 10 HR images as input. The input sequence has been aligned with ORB. Here, we do not have a quantitative comparison among the algorithms, because in this setup we do not have an HR original image and the correct registration among the input sequence.

We can see in Fig. 4.40 that, using only 10 images as input, the scientific name of the fish is more readable in the super-resolved image than using a simple interpolation. Additionally, we strongly suggest the ORB detector when super resolving larger images, because it is our fastest registration solution.

4.3 Final considerations

In this chapter, we presented qualitative and quantitative validation of the proposed super-resolution algorithms. The charts in Sec. 4.1.4 show that the best method depends

(a) Original image **(b)** Original image **(c)** Original image **(d)** Original image **(e)** Original image

(f) NN interpol. **(g)** NN interpol. **(h)** NN interpol. **(i)** NN interpol. **(j)** NN interpol.

(k) GSR_1, ORB **(l)** GSR_2, SURF **(m)** GSR_3, SIFT **(n)** GSR_4, SIFT **(o)** GSR_5, SIFT

Figure 4.39: LR versions of 25 real images taken by a mobile phone generate a 4× higher image. In the first row, the original target images H_k for different scenes. In the second row, each L_k is zoomed in by a factor of four. In the last row, the super-resolved image I_{SR} using different variations of GSR, and different registration methods. Unlike *Experiment #1*, the LR images differ from each other by perspective transformations in this setup, and not only by rotations and translations.

(a) LR image **(b)** NN interpol. **(c)** ORB + GSR_4

Figure 4.40: 35 HR images taken by a mobile phone generate an SHR image 5× higher.

on the application. The *RLS* method, for example, reconstructs images with the highest SSIM values, but it is two orders of magnitude slower than GSR. Due to its fast runtime,

we could also apply GSR to an always-on low-power environment, and the visual results are promising.

Based on such results, we now discuss which might be an ideal number of input images to be used by a multi-frame super-resolution reconstruction with quality and efficiency constraints. Figure 4.41 illustrates an example of pixels in an LR image with respect to the grid of a 3× higher image (each red pixel may contribute to a region of 3×3 pixels in the HR grid). It means that each region needs 3^2 information pieces from the input images to generate the HR one. Therefore, we may intuitively suggest that we need at least m^2 LR frames for a super resolved image by m.

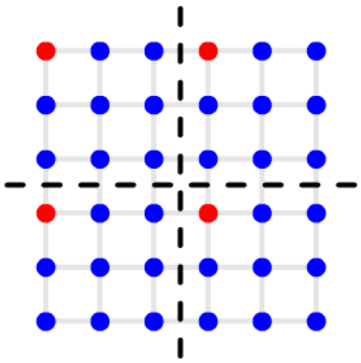


Figure 4.41: LR pixels (red dots) distributed throughout an HR grid.

In *Experiment #1* (Sec. 4.1), we quantitatively evaluated three variations of RLS, two implementations of the inpainting based-methods, and five variations of GSR, using up to 50 LR images in order to generate a 4× higher resolution image. The charts show that for more than 4^2 LR images, we have a very small improvement in the quality of generated HR images. Their curves grow quickly until 10 to 20 LR images, and then they become stable as we increase the number of images. Therefore, increasing the number of images to be analysed after m^2 may not compensate the consequential increase in execution time. For those reasons, m^2 may be a good value when choosing the number of input images: any number close to that may generate a good solution with a small response time. But if a large dataset is available, with more than m^2 input images, the reconstruction can still be calculated in an acceptable time by any variation of GSR.

Chapter 5

Eyes on the Target: Framework for SR and ALPR

In this chapter, we propose a free and open-source end-to-end framework that combines ALPR and SR techniques to recognize license-plate characters in low-quality real-world traffic videos, for forensic purposes. The framework super resolves a set of input frames from a video, and automatically recognizes the license-plate characters in the reconstructed image. Our main objective is to aid forensic analysts and practitioners in understanding a given event of interest.

5.1 The proposed framework

Automatic license-plate recognition (ALPR) uses optical character recognition (OCR) on images to extract and recognize the alphanumerics of a vehicle registration plate [32, 3]. 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) [18].

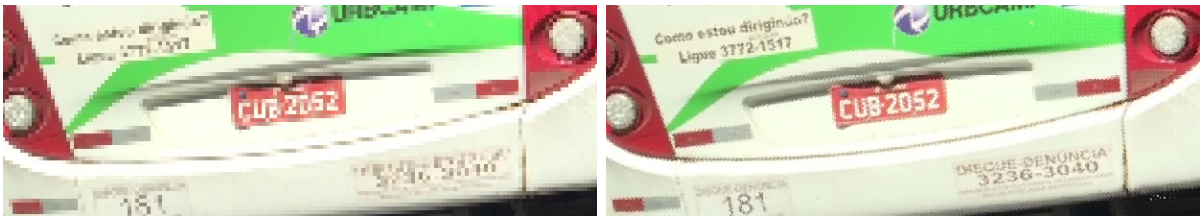
However, there are a number of low-quality surveillance cameras scattered throughout our cities that could help to identify a suspect, for example, in a crime scene. A sequence of video frames recording a dynamic scene may contain different information about the object of interest. Generally, a moving 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.

In this chapter, we design and develop a framework to super resolve and recognize license-plate characters in low-quality real-world traffic videos, aiding forensic analysts in understanding an event of interest. The "*Eyes on the Target*" 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. 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 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 single-image super-resolution algorithm [102, 31].
- 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 Figure 5.1. First, for Figure 5.1a, we have a single frame of a moving bus in a low-resolution video, zoomed by a factor of four. Then, in Figure 5.1b, we use SR to combine eight consecutive LR frames of the same scene. The license-plate alphanumerics and even other characters over the outer surface of the bus are visually easier to identify in Figure 5.1b than in Figure 5.1a.



(a) **(b)**

Figure 5.1: Super resolution *vs.* interpolation: (a) Piece of a low-resolution video frame (384×216 pixels) of a moving bus in a real-world traffic video, zoomed by a factor of four using a simple nearest-neighbor interpolation; and (b) Super-resolution version of the same piece, using eight consecutive input frames of a video. Not only the license-plate alphanumerics, but also other characters over the outer surface of the bus, are visually easier to identify in (a) than in (b).

To perform the super resolution in the Figure 5.1, we followed the same pipeline as in the the *Experiment #1* (see Chapter 4). A sequence of eight consecutive video frames was downscaled, aligned and super resolved. This methodology for the registration step has worked properly for the setup we presented in Chapter 4, but it is not appropriate for fast-moving vehicles captured by static cameras. Fig. 5.2 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 5.2: 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. 5.3 depicts an example of a rectified and binarized image, to improve the results in the recognition step.

Figure 5.3: Example of a rectified and binarized license-plate image.

The "*Eyes on the Target*" framework 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. 5.4 depicts. The framework performs the super resolution and recognition of the license-plate characters in six steps:

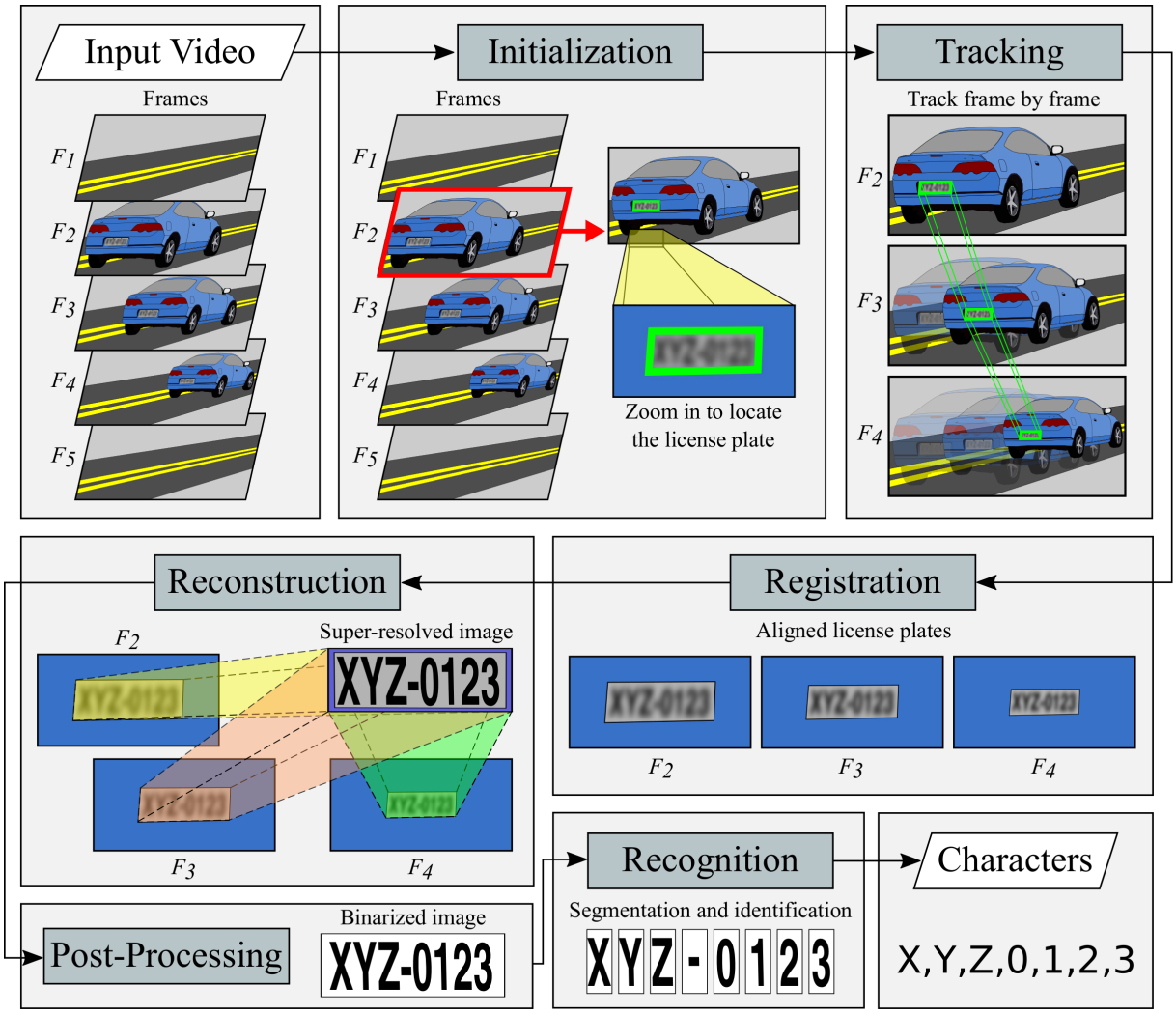


Figure 5.4: Our end-to-end framework pipeline. (1) *Initialization*: Choosing a starting frame and locating the license-plate region in such frame; (2) *Tracking*: Finding re-occurrences of the license plate over the consecutive frames, and aligning the frames with respect to the plate positions; (3) *Registration*: Refining the previous alignment with subpixel accuracy; (4) *Reconstruction*: Combining the sequence of consecutive frames into a high-resolution grid; (5) *Post-processing*: Applying refining image processing operations to the reconstructed image, to improve the results in the recognition step; and (6) *Recognition* of the alphanumerics in the super-resolved license plate.

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 use all variations of the reconstruction algorithms, proposed in Chap. 3, to combine the frames, producing a super-resolved and rectified license plate within a fixed-sized grid according to country-specific specifications.

5. We apply post-processing operations to the image (e.g., Otsu's binarization [82]) to improve the results of the final step; and

6. Finally, we design an ALPR solution based on Tesseract [105] and OCRopus [13] to recognize the license-plate 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 is publicly available for researchers upon request.

We now turn our attention to describing each step of the proposed approach in Secs. 5.2 through 5.7. For further implementation details, see Appendix B.

5.2 Initialization

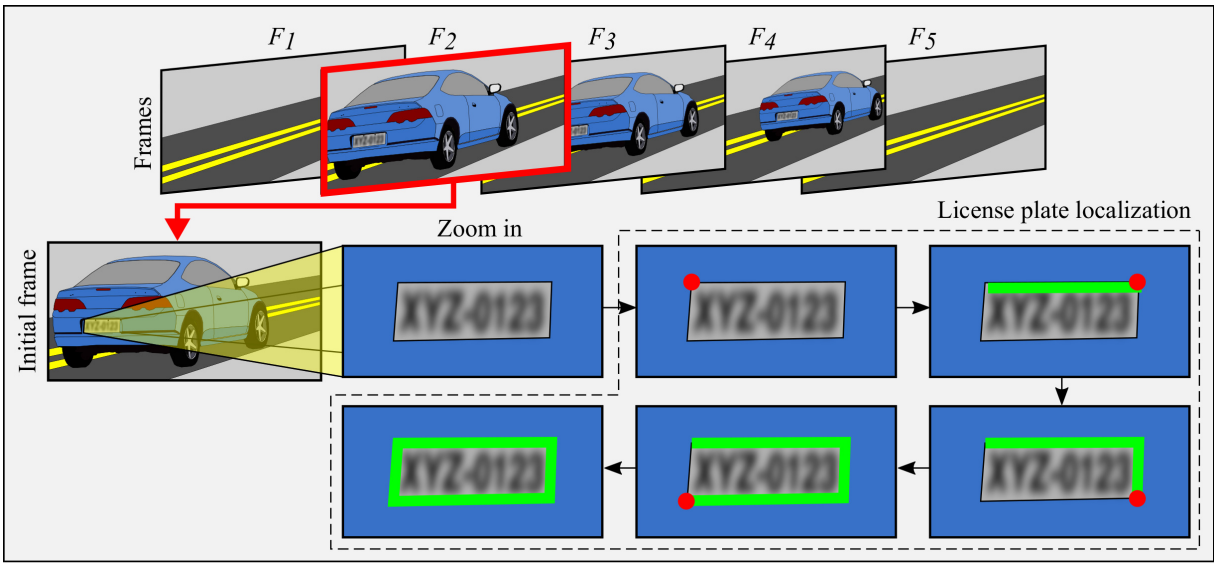


Figure 5.5: The initialization step. The user (e.g., a forensic analyst) selects the initial frame, and the system opens a window with a zoomed in version of the plate. The user selects four points and creates a bounding box around the characters to be identified. This step is done only once (for the initial frame).

The framework starts with a graphical interface playing the traffic video (c.f., Fig. 5.5). The user (e.g., a forensic analyst) can interact with the system by alternating between *pause* and *playing* 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 (see Fig. 5.6). 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.

Figure 5.6: User choosing four points and creating a bounding box around the license plate of interest.

Figure 5.7: Selected ROI around the characters to be identified.

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 work 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.

5.3 Tracking

Fig. 5.8 illustrates the *tracking* step of the framework. The inputs of this step are: (1) the initial frame identifier (defined in the previous step), (2) the region of interest around the license plate (also calculated in the initialization step), and (3) the original video frames.

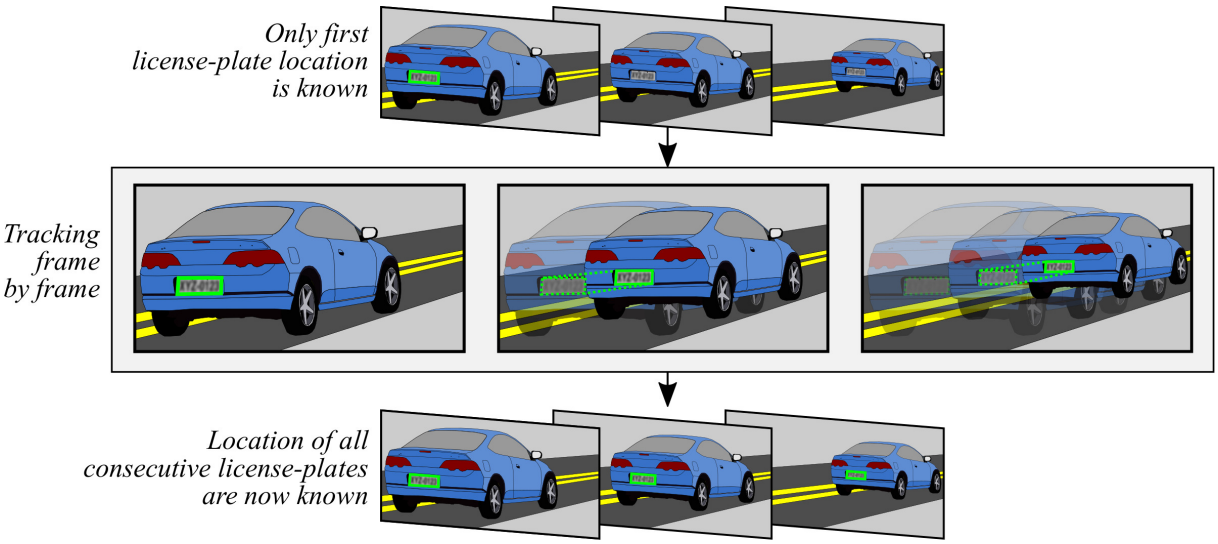


Figure 5.8: Tracking step: the objective is to find re-occurrences of the license plate over the consecutive frames.

In this work, we examine five different solutions to track the license plates, as detailed in Secs. 5.3.1 through 5.3.3.

5.3.1 Pyramidal Lucas-Kanade optical flow (PyrLK)

The first method is based on the Lucas-Kanade optical flow [70]. 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 [100]. 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) [11]. 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. 5.9 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).

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

Figure 5.9: PyrLK feature tracker to estimate the optical flow.

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. 5.1, 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.

5.3.2 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 [35], 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. 5.10a illustrates an example of the grid (red dots) and the flow motion (green lines). Fig. 5.10b uses HSV to visualize the same flow as in Fig. 5.10a (*hue* shows the flow direction, and *value* shows the flow magnitude).

(a) **(b)**

Figure 5.10: Example of dense optical flow as a result of the Farneback's algorithm.

In both PyrLK (Sec. 5.3.1) 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 [129] 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.

5.3.3 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. 5.11). The points are matched using k-NN and Fast Library for Approximate Nearest Neighbors (Flann) [75, 43]. 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 5.11: 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.

The white polygon in Fig. 5.11 indicates the region around the license plate[1] in a frame F_k. The yellow dots are the keypoints found by SIFT (inside the ROI in F_k, and inside the entire image F_{k+1}). The green lines show the matches between the frames, and the red line depicts an incorrect match that will not be used to calculate the homography.

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 the tracking 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.

[1]Note that the region around the license plate in Fig. 5.11 illustrates the same region in which we found Shi-Tomasi corner points in the first tracking solution (PyrLK).

5.4 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. The images were previously aligned in the tracking step, and now we need to refine this alignment with subpixel accuracy. There are three available possibilities for the realignment in our framework:

a) Using Lukas-Kanade Optical Flow, as in the first tracking solution (Sec. 5.3.1), 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 do not perform the registration, considering that the tracking step had already performed a good alignment of the frames (we refer to this possibility as "None", since we do not use any further refinement method).

Fig. 5.12 illustrates the registration process.

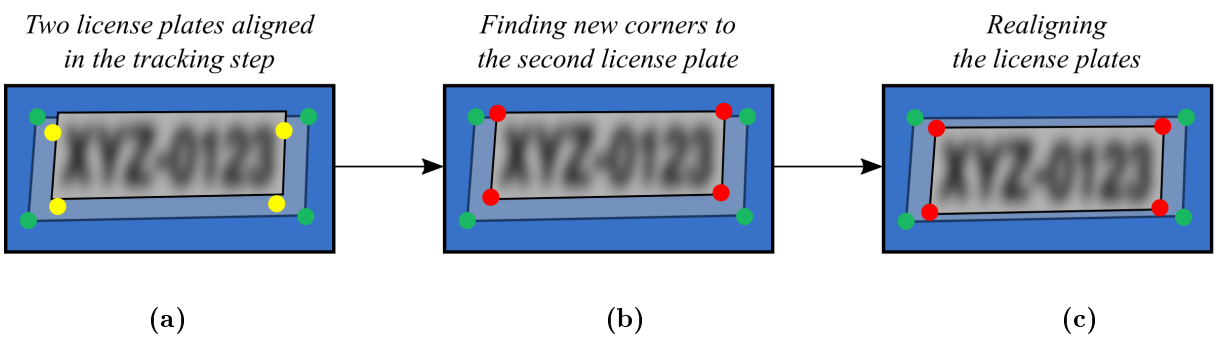


Figure 5.12: In the tracking step, the initial bounding box is tracked without subpixel accuracy, but gives us a pre-alignment of the frames. Green dots in (a) are the corners in the initial frame, and yellow dots are the corners that were tracked in the subsequent frame. The registration step finds new corners for the consecutive frame, without pyramids and within a small window size. 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.

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 of this step are: (1) the set of realigned frames and (2) the license plate's bounding box in each frame.

5.5 Reconstruction

For the super resolution of the license plates, we use all the algorithms proposed in Chapter 3. In the previous application (super resolution on an always-on low-power environment), a sequence of n low-resolution images of size (w, h) pixels generates a super-resolved image of size $(s \times w, s \times h)$ pixels for a scale factor s. Now, instead of creating a high-resolution image of $(s \times w, s \times h)$ pixels, 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). The super-resolved image is similar to a rectified license plate (see Fig. 5.3), 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.

It is important to mention that 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 HR grid. Additionally, only the ROI is super resolved, and not the entire image. Therefore, we might be able to run all the reconstruction algorithms proposed in Chap. 3, and not only the fastest ones.

5.6 Post-processing

As we know the bounding box of the license plate (through the user input for the initial frame), the super-resolved image could already be 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. Hence, in the post-processing step, we first focus on the region which contains the alphanumerics that we want to recognize discarding the other areas. Then, we use Otsu's binarization [82] in the image 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 (see Fig. 5.13).

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 [82]; or (b) using an adaptive thresholding [42] to binarize the image. Adaptive thresholding may be good when the image has different lighting

Figure 5.13: Post-processing step: removing borders and binarizing the rectified and super-resolved image.

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.

5.7 Recognition

Finally, with the super-resolved license plates of interest, we rely upon two OCR systems to identify the license-plate characters: *Tesseract* and *OCRopus*. 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 the framework.

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[2]. 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 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 the Mandatory font, the letters 'I' and 'O' are easily confused with the digits '1' and '0', so we divided the license plates in two parts, one for digits and one for characters as the license plates in Brazil have always three letters followed by four digits. Therefore, for training the methods, we created synthetic images with $17,576$ combinations of letters ($26 \times 26 \times 26$) and $9,999$ combinations of digits (the sequence 0000 is not used in Brazil). It is worth mentioning that such training, including the separation between digits and letters, is specific for our particular setup.

We also added, purposefully, small rotations and noisy artifacts to the synthetic images in order to simulate the real-world license plates more accurately. Fig. 5.14 depicts examples of some training images used in our method considering the Mandatory font.

For an appropriate training process, for each training image, we also annotate the

[2]Each OCR might define specific rules for the font size and for the number of words per line in an image

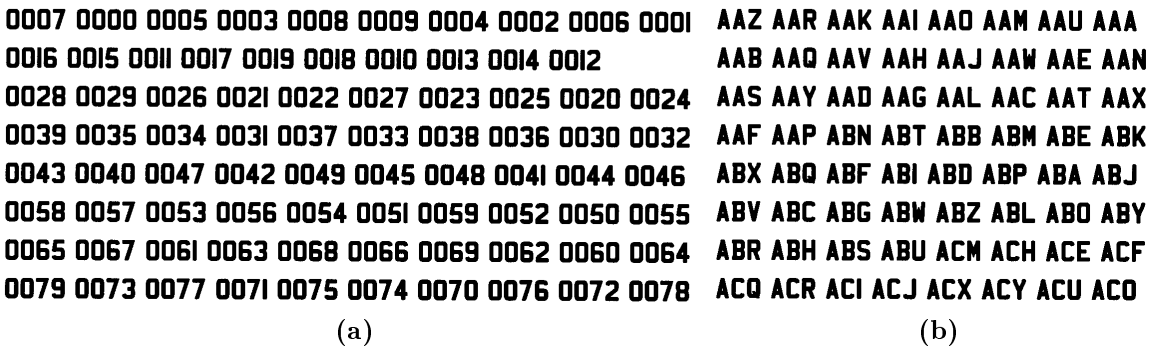


Figure 5.14: Some training images used with Tesseract considering the Mandatory font. (a) Examples of digits. (b) Examples of letters.

bounding-box coordinates around every character in the image. In Tesseract, for example, each training example encodes the characters being recognized and the bounding-box coordinates around the character. Fig. 5.15 shows the first lines of the box file for Fig. 5.14b, comprising the bounding boxes for the two first words in Fig. 5.14b (AAZ and AAR).

```
A 112 4654 139 4694 0
A 143 4654 170 4694 0
Z 174 4653 201 4694 0
  201 4653 222 4694 0
A 222 4653 249 4693 0
A 253 4653 280 4693 0
R 285 4653 311 4693 0
```

Figure 5.15: The first lines of a box file for training Tesseract. Each line contains: the character being recognized, the four bounding-box coordinates around the character, and the page wherein the word occurs (the image file might not fit in only one page).

At the end of the training process, each OCR is custom-tailored to the problem of license-plate recognition. It is also possible to further extend the training data using real-world 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.

5.8 Final considerations

The framework is part of a work that has been recently accepted [97]. The ultimate products of the framework are the super-resolved image and the sequence of recognized characters. However, different inputs and outputs are available to the user while the framework calculates the final solution, summarized in Fig. 5.16.

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

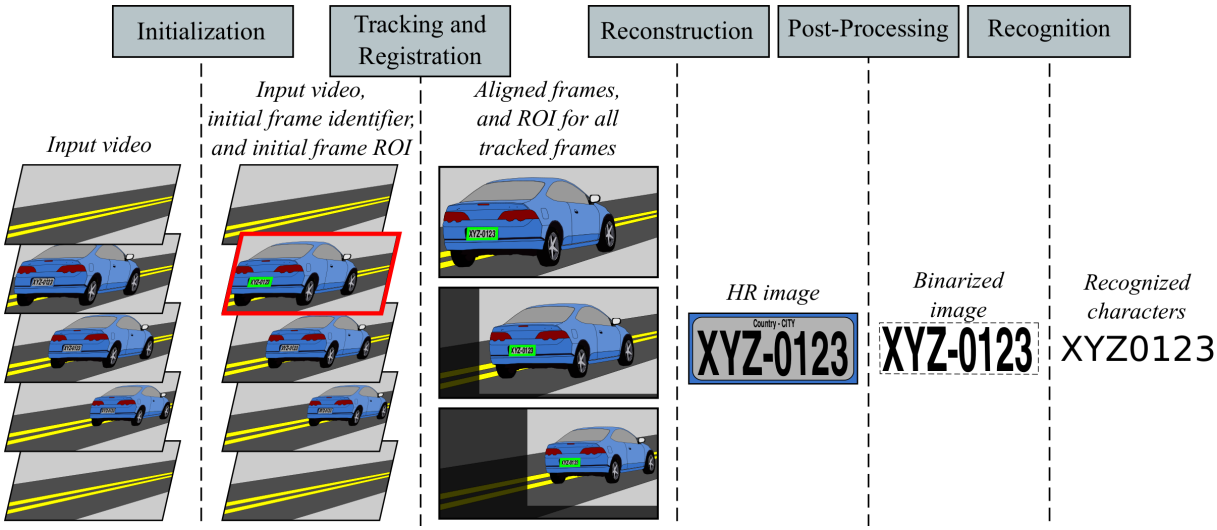


Figure 5.16: Inputs and outputs of each step of the proposed framework.

steps (especially the tracking step).

2. Alternatively, the user can easily customize the framework and change the tracking method, registration method, reconstruction algorithm and/or OCR systems.

As previously mentioned, the user can choose from five different tracking methods (PyrLK, PyrDense, SIFT, SURF and ORB), three registration solutions (LK, Dense and None), 10 variations of the reconstructed algorithms (RLS, $RLSO$, $RLSU$, ISR_1, ISR_2, GSR_1, GSR_2, GSR_3, GSR_4 and GSR_5), two binarization methods, and two different OCR methods. In addition, there is one default method for each step (as illustrated in Fig. 5.17), based on the experimental results, as we shall explain in Chapter. 6.

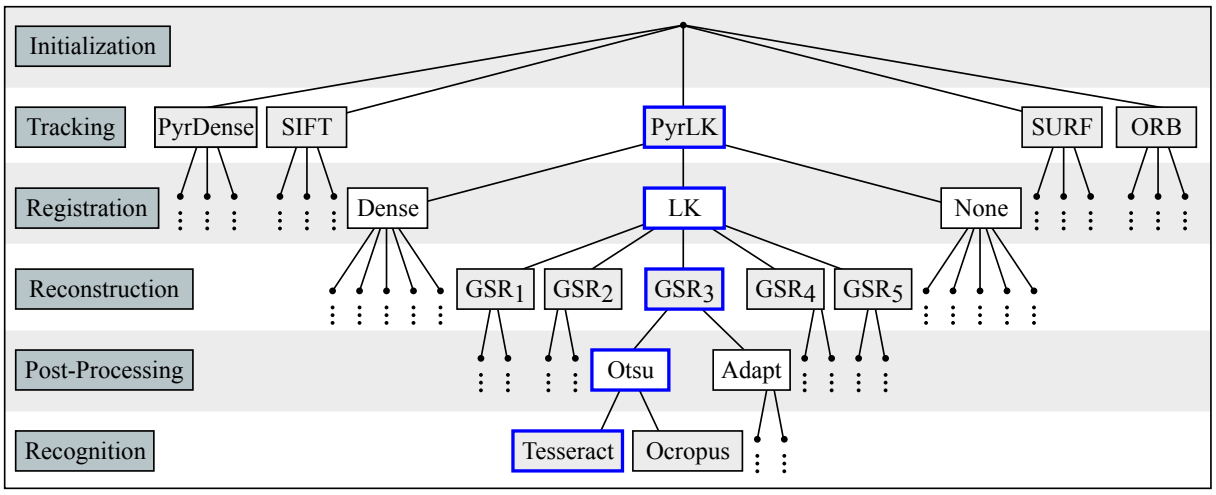


Figure 5.17: Each leaf of the tree can produce a different result: Five different methods in the tracking step, three in the registration, five in the reconstruction, two in the post-processing, and two in the recognition step. The methods highlighted in blue are the default options, based on the experimental results.

Chapter 6

Validation of the forensic framework

Finally, in this chapter, we show experiments (namely Experiment #3 of the validation plan outlines in Chap. 4) for the "*Eyes on the Target*" framework, described in Chapter 5. In our first experiment (see Chap. 4), we validated the reconstruction algorithms using target HR images and the correct alignments among the LR images, and we put the designed solution in perspective with respect to traditional SR methods in terms of efficiency and effectiveness. Differently from this standard practice, we do not have such controlled environment herein.

In this present chapter, 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 Chap. 2, 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.

We execute the framework for all the 200 videos in our dataset (see Sec. 6.1), using all the the five tracking methods, three registration solutions, 10 variations of the reconstruction algorithm, two binarization methods, two recognition systems, and from 1 to 10 consecutive frames as input. For each video, we have $5 \times 3 \times 10 \times 2 \times 2 \times 10 = 6,000$ possible results (600 for each number of input frames, and a total of $1,200,000$ results considering all the 200 videos). The dataset is publicly available for researchers upon request (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.

6.1 Dataset

For validation, we collected a dataset comprising 200 real-world traffic videos, in which 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 (video codec H.264, without additional

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

compression). 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. 6.1). 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.

(a) **(b)**

Figure 6.1: The license-plate alphanumerics are easily readable in the first video frames (a) but not in the last ones (b). The first frames are used to create the ground-truth files, 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).

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.

6.2 Results

We divide the *Experiment #3* into six parts, since we have six steps in the framework:

1. First, in Sec. 6.2.1, we investigate if our initialization step is appropriate for the framework.

2. Then, in Sec. 6.2.2, we validate the tracking methods.

3. In Sec. 6.2.3, we choose the best solution for the registration step.

4. In Sec. 6.2.4, the proposed super-resolution methods are now validated with respect to this environment.

5. In Sec. 6.2.5, we show advantages and disadvantages for each post-processing methods.

6. Finally, we validate the two recognition methods in Sec. 6.2.6.

6.2.1 Validation of the initialization step

We start investigating if our decision of manually selecting the ROI around each license plate (in the initialization step) is appropriate for our framework. First, in Fig. 6.2, we super resolve only five consecutive frames of four different videos. The reconstructed images in Fig. 6.2b are more easily readable than the single frames in Fig 6.2a, so the bounding box selected by the user might be leading to an appropriate reconstruction.

Figure 6.2: 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.

The chart in Fig. 6.3 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.

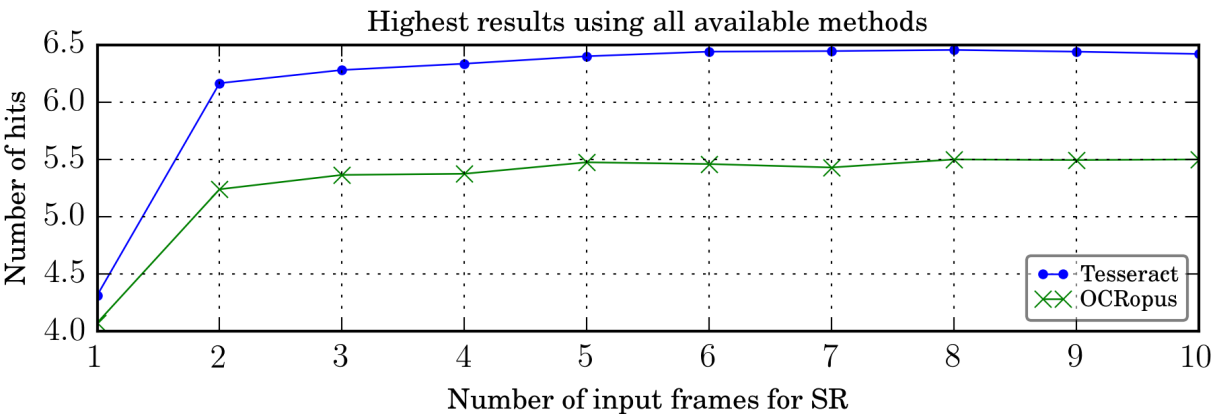


Figure 6.3: Quality of the initialization step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using all the available methods.

Each curve in Fig. 6.3 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. 6.2.2 through 6.2.6, we validate the other framework's steps, including qualitative and quantitative results for each step. 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).

6.2.2 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. 6.4. 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. 6.4a, the camera re-focuses on the object after a few frames. In Fig. 6.4b, the license-plate focus is lost, and not acquired until the end of the video.

Figure 6.4: PyrLK keeps tracking even if the camera momentarily loses focus of the license plate. In (a), the camera re-focuses on the object after a few frames. In (b), the camera is not able to 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. 6.5 depicts consecutive license plates with varying illumination conditions throughout the route.

Figure 6.5: 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.

The red dots in Fig. 6.5 show that, even in this case, the tracking could be performed. PyrLK could successfully track the bounding box through seven frames of the example, and the PyrDense method tracked the license plate through all the 10 input frames. SIFT, SURF and ORB could not align any consecutive images.

Finally, we select another video to illustrate the tracking performed by each method. Fig. 6.6 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. 6.7 depicts that the pyramidal implementation of the Lucas-Kanade optical flow outperforms the other tracking methods, on average. The blue curve in

Figure 6.6: Examples of an input video tracked by PyrLK (a), PyrDense (b), SIFT (c), SURF (d) and ORB (e).

Fig. 6.7 for the Tesseract hits is the same as the blue curve in Fig. 6.3, and shows the best possible values if we alternate among all available methods for the tracking.

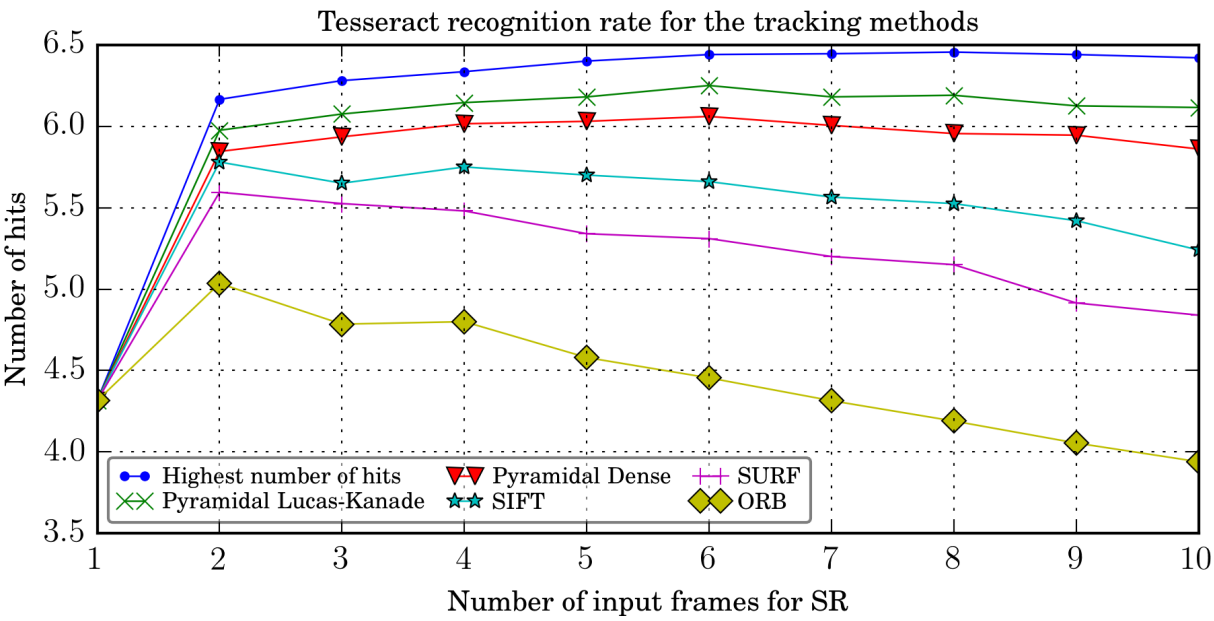


Figure 6.7: Quality of the tracking step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using each of the tracking methods and Tesseract as the recognition rate.

We also investigate the number of videos in which the framework correctly identified all the seven license-plate alphanumerics (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 (see Appendix A for the number of videos in which a method got the highest accuracies). Table 6.1 confirms that PyrLK is the most promising tracking method, and it is also the fastest solution. PyrLK results are highlighted in the table, and it is the default tracking method of the framework. It is worth mentioning that more than one tracking method may recognize the same number of hits in a video.

Tracking method	Acc_7	*Time (s)*
PyrLK	**117 (58.5%)**	**0.02**
PyrDense	108 (54.0%)	0.9
SIFT	110 (55.0%)	11.2
SURF	109 (54.5%)	5.5
ORB	99 (49.5%)	0.3

Table 6.1: 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.

6.2.3 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 three input videos as examples to illustrate the quality of each registration. In the first column of the Fig. 6.8, we see the ROI of the license plate that we manually defined in the initial frame. Then, in the second column, we see the tracked corners in the last frame. Our objective, then, is to refine this tracking. In the following columns, we have the registration results using LK, Dense and None.

We see in Fig. 6.8 that only the Lucas-Kanade method could refine the incorrect tracking for the three videos. However, there are cases in which the tracked corners are good enough to align the images. The chart in Fig. 6.9 shows that "None" outperforms both the Lucas-Kanade optical flow and the Farneback's dense optical flow, on average.

The Acc_7 values for the registration methods, in Table 6.2, 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 122 videos (61.0%). The Acc_7 for LK was 56.5% (113 videos), and for Dense it was 46.5% (93). 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 42 videos (and in 40 videos if we track with PyrDense).

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

Figure 6.8: Examples of the registration methods in three input videos. In (a), the ROI of the license plate that we manually defined 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.

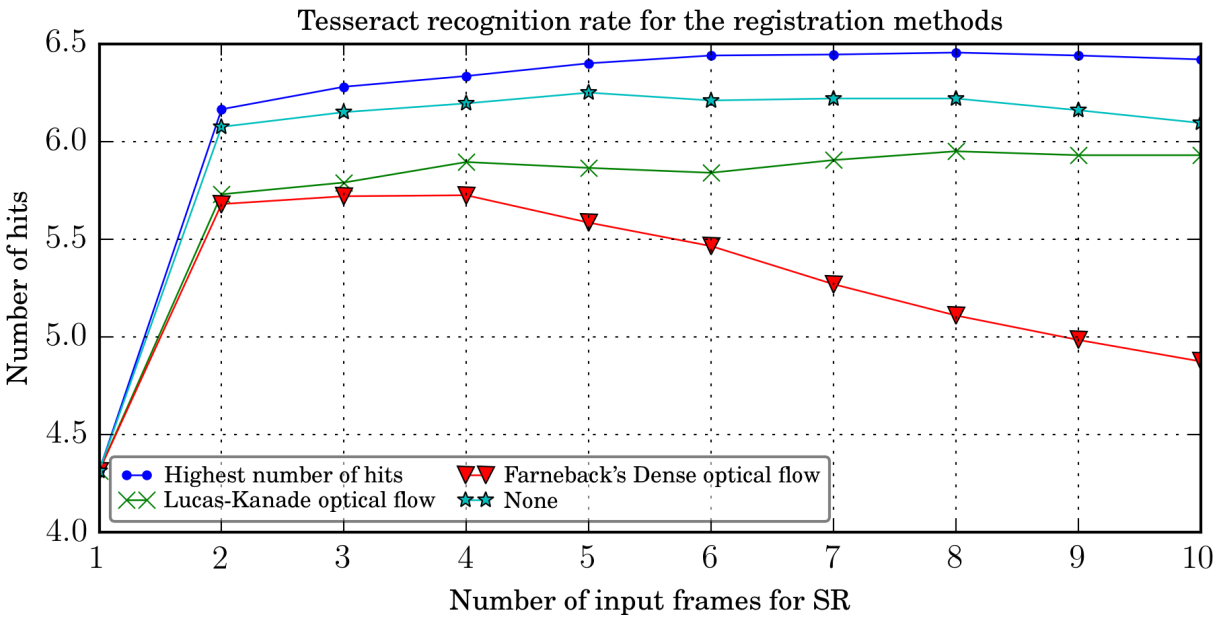


Figure 6.9: Quality of the registration step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using each of the registration solutions.

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

Registration method	Acc_7	*Time (s)*
LK	113 (56.5%)	**<0.01**
Dense	93 (46.5%)	0.7
None	**122 (61.0%)**	-

Table 6.2: Accuracy and runtime of the framework's registration step.

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).

6.2.4 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. 6.2.2 and 6.2.3. The first row of Fig. 6.10, shows a zoomed version of the initial frame, without super resolution, for four input videos.

Figure 6.10: 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 alphanumerics in Fig. 6.10 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).

The chart in Fig. 6.11 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. 6.11 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 6.3.

The Acc_7 values for the reconstruction methods in Table 6.3 are also very similar. We highlighted the results for GSR_4 (the variation of GSR with the highest Acc_7 values) and

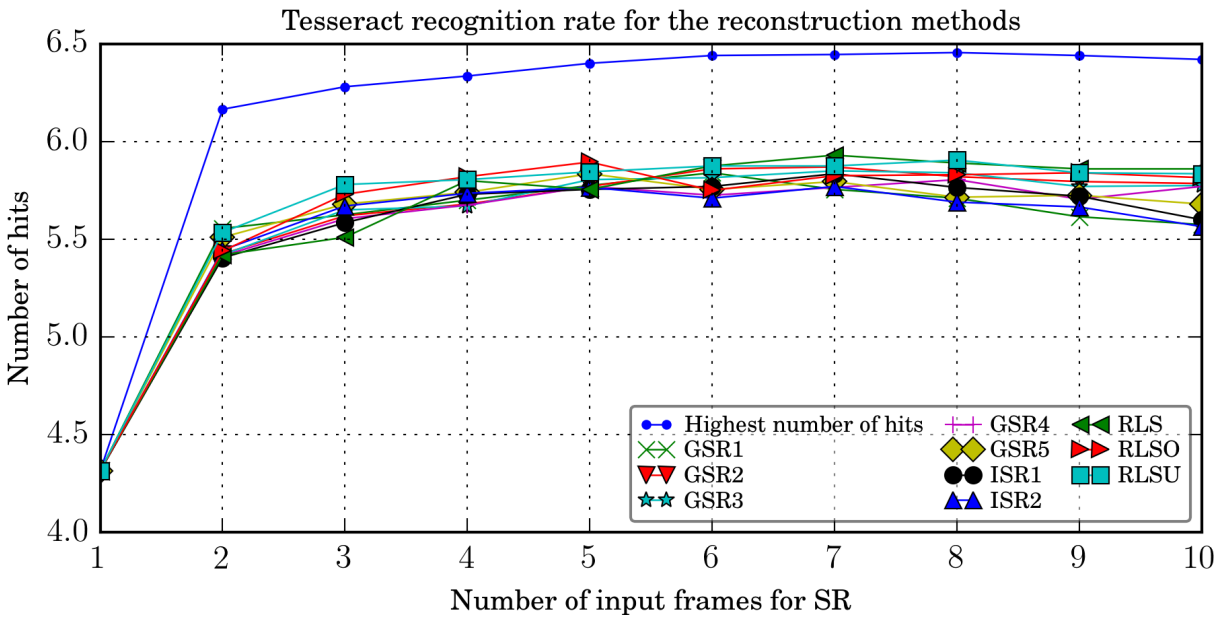


Figure 6.11: Quality of the reconstruction step as a function of the number of input frames. The highest the number of hits, the better is the result. Each curve is the average of the highest number of hits for all videos in the dataset, using each variation of *GSR*.

Method	Acc_7	*Time (s)*	*Method*	Acc_7	*Time (s)*
GSR_1	93 (46.5%)	0.5	ISR_1	96 (48.0%)	3.0
GSR_2	102 (51.0%)	0.6	ISR_2	94 (47.0%)	2.6
GSR_3	98 (49.0%)	0.6	*RLS*	100 (50.0%)	7.5
GSR_4	**103 (51.5%)**	**0.5**	*RLSO*	104 (52.0%)	2.0
GSR_5	97 (48.5%)	0.6	*RLSU*	**105 (52.5%)**	2.0

Table 6.3: Accuracy of the reconstruction step, and the runtime (on average) to reconstruct 10 frames.

RLSU (highest Acc_7 for *RLS* implementations). Given that both are very similar, we choose GSR_4 as the default super-resolution algorithm, due to its runtime is lower 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.

Fig. 6.12 shows the same results as in Fig. 6.12, but only for GSR_4, ISR_1 and *RLS* (the best variation of each method, according to the Acc_7 in Table 6.3). It is worth pointing out that GSR_4, the algorithm that we choose as the default solution for the framework, produced the **worst** similarity values in *Experiment #1*. Such result supports our claim that the best solution to super resolve a set of images is highly dependent on the application.

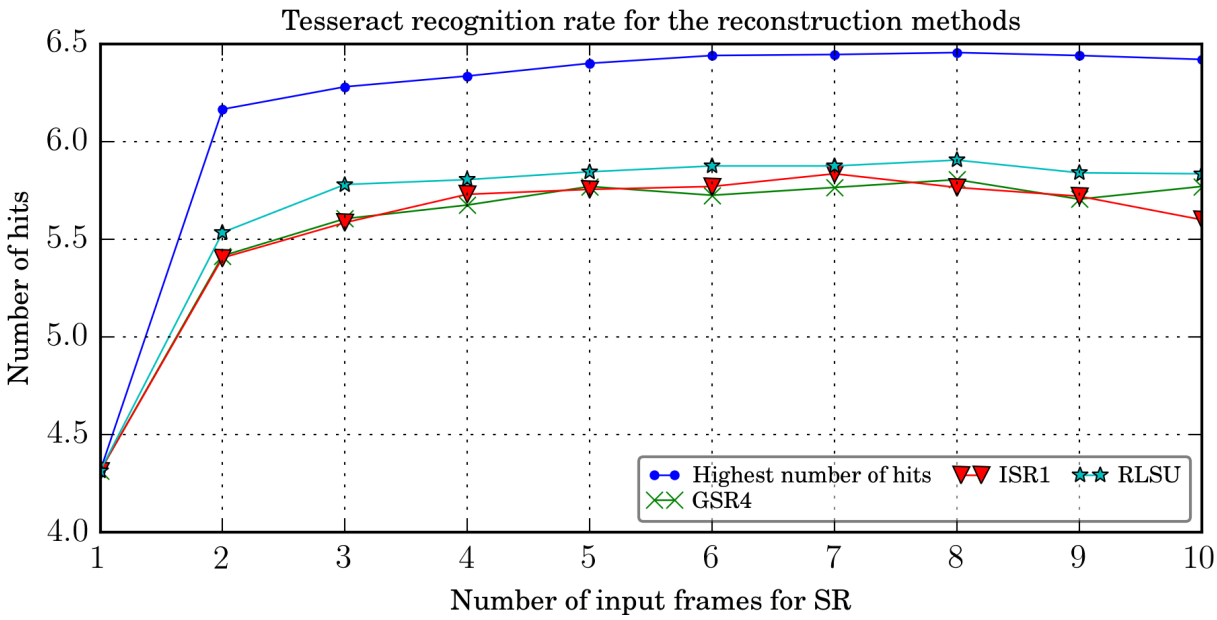
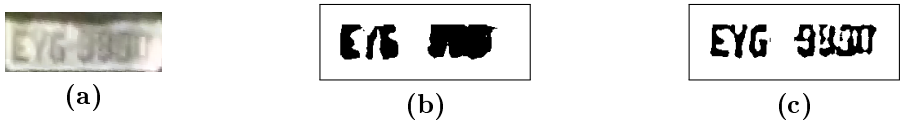


Figure 6.12: Quality of the reconstruction for the best variation of each proposed method.

6.2.5 Validation of the post-processing methods

In Fig. 6.5, 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. 6.14, we show the visual difference when we super resolve only two frames of the license-plate in Fig. 6.5. Tesseract could correctly identify all the characters using Adaptive thresholding (EYG-9890), but could not recognize any digits when using Otsu's binarization.



(a) (b) (c)

Figure 6.13: Examples of the post-processing methods using only two consecutive frames. In (a), the super-resolved image. In (b), the OTSU's binarization. In (c), the Adaptive thresholding.

In Sec. 5.6, 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. 6.10, may lead to a bad Otsu's binarization. Fig. 6.14 depicts other examples in which the Adaptive thresholding outperforms the Otsu's binarization in such cases. Fig. 6.15 summarizes the results for both binarization methods.

Fig. 6.15 shows that the results for both binarization methods are similar, if we use Tesseract to calculate the recognition rate. To define the default solution for the post-processing step, we investigate the Acc_7 values for the binarization methods in Table 6.4.

Table 6.4 shows that The Acc_7 for Otsu is higher than for the Adaptive thresholding.

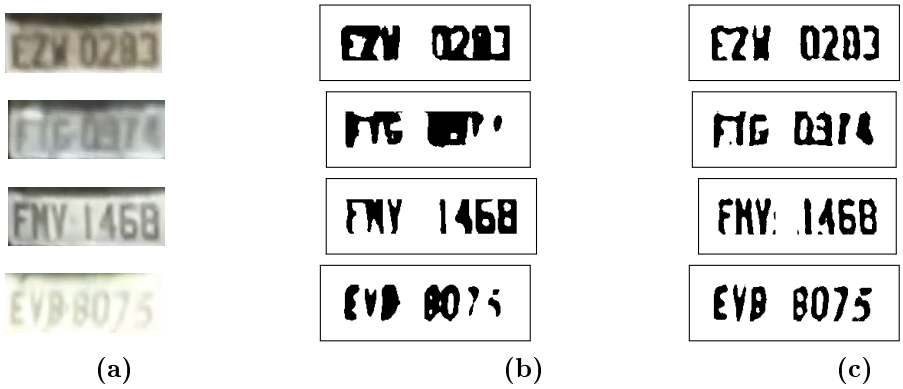


Figure 6.14: Examples of the post-processing methods in four input videos. In (a), the super-resolved image. In (b), the OTSU's binarization. In (c), the Adaptive thresholding.

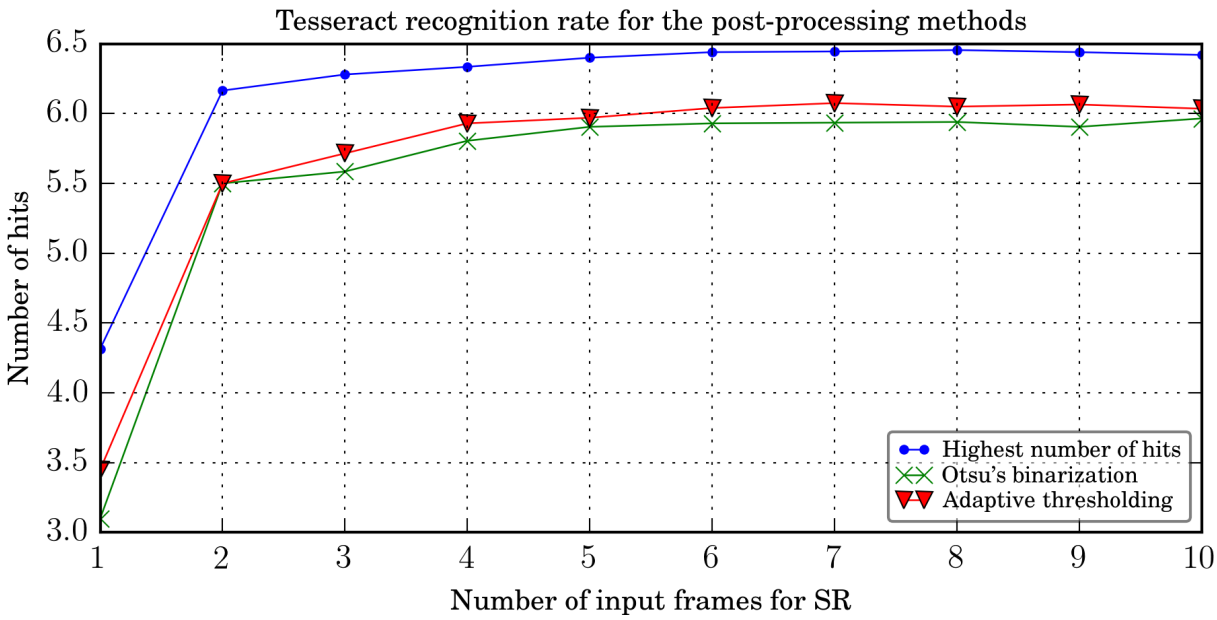


Figure 6.15: Quality of the post-processing step as a function of the number of input frames.

Binarization method	Acc_7	*Time (s)*
Otsu's binarization	**112 (56.0%)**	<0.01
Adaptive thresholding	92 (46.0%)	<0.01

Table 6.4: Accuracy and runtime of the post-processing step.

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.

6.2.6 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. 6.16, 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.

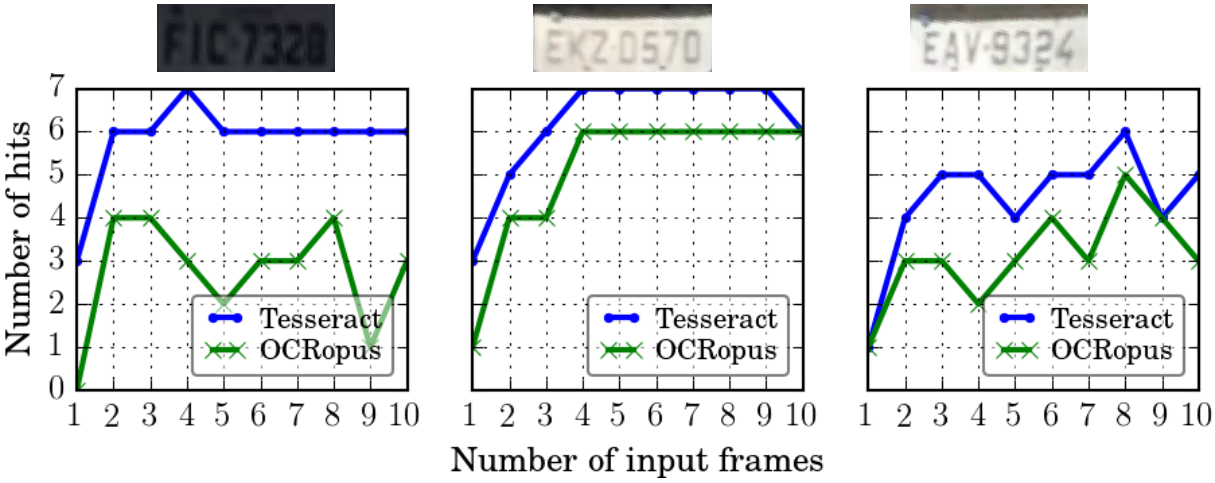


Figure 6.16: Quality of the recognition methods for three super-resolved images, using the defaults methods in each step.

As we showed in Figure 6.3, the Tesseract recognition rate also outperforms the OCRopus number of hits, on average, using all the available methods and all the videos in the dataset. The Acc_7 shows Tesseract is more effective than OCRopus, almost twice as effective in a significantly lower runtime (see Table 6.5).

OCR system	Acc_7	*Time (s)*
Tesseract	**131 (65.5%)**	**0.1**
OCRopus	72 (36.0%)	4.6

Table 6.5: Accuracy and runtime of the recognition step.

We see, in Table 6.5, that Tesseract has almost the double of accuracy than OCRopus in a significantly lower runtime. Due to the results in Fig. 6.3, Fig. 6.16 and in Table 6.5, we choose Tesseract as the default OCR solution in the framework.

6.3 Final considerations

The charts in Figs. 6.3, 6.7, 6.9, 6.11 and 6.15 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, including the super-resolution methods (see Fig. 6.17).

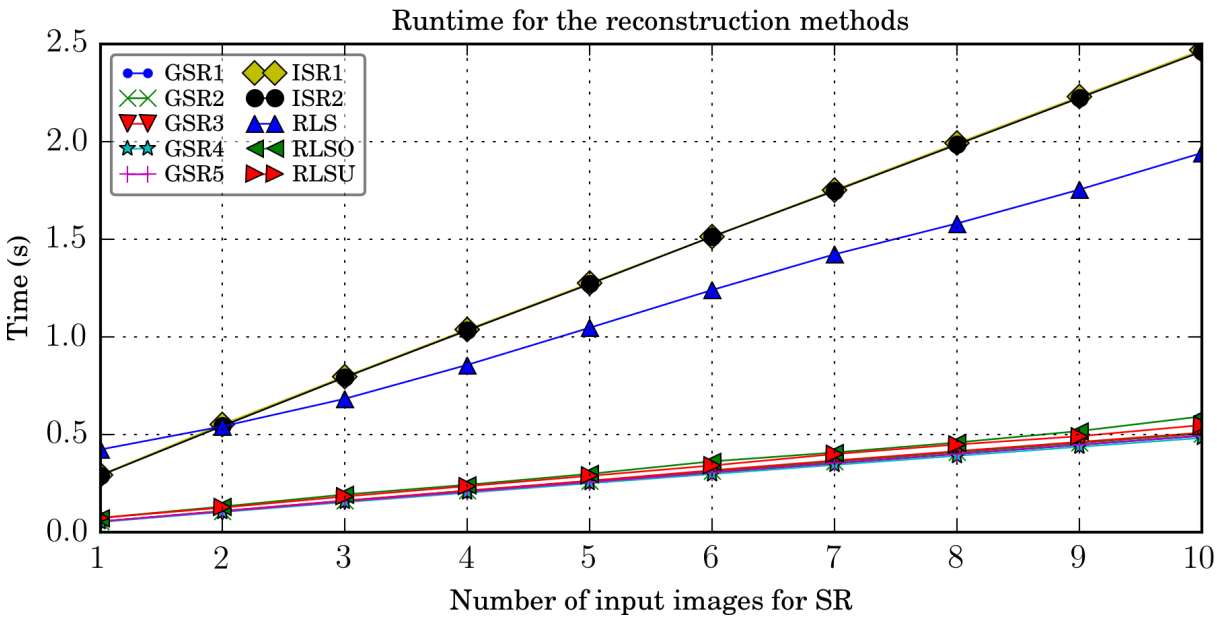


Figure 6.17: Runtime for super resolving the input frames using each reconstruction method.

Finally, all the datasets and associated transformation matrices are publicly available for researchers upon request, for reproducibility purposes.

Chapter 7

Conclusions

The main purpose of this work was to explore SR possibilities for digital images, in scenarios wherein we have multiple frames of a scene. With this goal in mind, we designed and developed three methods for super resolution of multiple observations, including different variations/optimizations of each method.

The first method is the Super Resolution using Regularized Least Squares (RLS), which relies on Tikhonov regularization and adaptive filtering to solve a sparse least-square problem. Its first optimization ($RLSO$) decreases the execution time for constructing the matrices of the linear system, by dividing the grid into imaginary areas, in a way that each LR pixel contributes only to equations of its closest grid pixel. The complexity for calculating all linear system equations decreases from $\mathcal{O}(n \times p_k \times p)$ in RLS to $\mathcal{O}(n \times p)$ in this optimization, where n is the number of LR images, p_k is the number of pixels in each LR image, and p is the number of pixels in the HR grid. The other optimization ($RLSU$) uses a simple uniform distribution, instead of the Gaussian one, in the same imaginary region as in $RLSO$.

The second method is the inpainting-based Super Resolution (ISR), that projects the LR pixels separately onto the HR grid, and then uses inpainting techniques to calculate the unknown grid pixels. Image inpainting is a technique for restoration of degraded photos that fills in part of an image using information from surrounding areas, and several algorithms have been designed for this purpose. We implemented two variations of the method (ISR_1 and ISR_2), each one using a different inpainting technique.

The third method is the Geometric k-Nearest Neighbors Multi-Frame Super-Resolution (GSR), a direct method exploring the concept of geometric neighborhood, to combine LR pixels onto an HR grid. There are five variations of such method ($GSR_{1...5}$). For each grid pixel p we want to calculate, the method uses k-NN to find the pixel q_k in each LR image I_k that is closest to p. Each variation uses a different policy to combine such nearest pixels q_k into a single information.

Only the first method (RLS) and its variations was designed to recover the high-frequency components lost during the blurring process, in which the observations have been acquired by the camera. On the other hand, RLS is expected to run slower than the other methods, and it might not be computationally attractive for applications that require quick responses and low-memory footprint. In addition, in order to execute any of the proposed methods, we need to perform a registration step, prior to the reconstruction,

that might be different for each application.

We validated the aforementioned methods in three different setups:

1. *Experiment #1*: First, we created a controlled environment, in which HR images generated LR observations. Then, such low-resolution images were super resolved, and the original HR image could be used as a target to validate the reconstruction.

2. *Experiment #2*: Then, we turned our attention to recent mobile phones, that take dozens of photos per second. Those cameras can gather a set of images while somebody is holding the camera manually in approximately the same position. It turns out we can leverage such small camera shake to reconstruct an HR image. In this setup, the SR algorithms should be fast and with a low-memory footprint, in order to be executed in this always-on low-power environment.

3. *Experiment #3*: Finally, we designed and developed a novel, free and open-source end-to-end framework to super resolve and recognize license-plate characters in low-quality real-world traffic videos, captured by cameras not designed specifically for the recognition task, aiding forensic analysts in understanding an event of interest.

The experiments showed that, indeed, it is possible to explore geometric properties from multiple low-resolution images in order to combine them into a higher resolution image, and to achieve good super-resolution results for photos gathered by mobile devices and for license plates in low-quality real-world surveillance videos.

7.1 The best of the proposed methods

In the *Experiment #1*, since we generated the pool of LR images, we could calculate the "correct alignment" among the input observations. Such information may not be available in the real-world applications, but it is an important aid for evaluating our algorithms, because every inaccuracy in the super-resolved image comes only from the reconstruction step (and not due to an incorrect registration). We use this first setup to point out the best method among the proposed ones under ideal conditions, but we emphasize that each method might be more or less advantageous in each setup, with different restrictions, limitations and constraints.

The two optimizations of the Super Resolution using Regularized Least Squares introduce less information and constraints into the linear system than the original model. Hence, *RLS* produced the best similarity values among such variations, in this first experiment. However, *RLSU* might be a good solution, since its results are only slightly worst than *RLS*, and its execution time is significantly slower. It is interesting to mention that *RLSU* does not use the Gaussian filtering to create the linear system, but its results are better than *RLSO* (even using input images that have been previously blurred with the Gaussian filter). Hence, the algorithm might be useful even when we do not know much about the filtering in the downsampling process.

The first inpainting-based variation, based on the work of Bertalmio et al. [7], produced SSIM values higher than ISR_2. Moreover, the more images we use as input, the faster the

super resolution may finish. ISR_1 not only produces super-resolved images with higher quality than ISR_2, but is also faster.

For the GSR variations, the Wilcoxon test indicates that GSR_1 is the most advantageous. In addition, such first implementation is also the fastest among the variations of all the proposed methods. Comparing the methods altogether, we have RLS and $RLSU$ producing the highest similarities, and recovering high-frequency components of the target image that have been lost during the blurring process. However, GSR is two orders of magnitude faster than RLS, on average. We suggest GSR, among all the methods proposed in Chap. 3, for super resolving input images with high dimensions.

As another contribution, the dataset of this experiment contains 100 groups of images to validate SR methods. In each group, we have 100 LR observations of a scene, the correct alignments among them, and the associated target HR image. We also make available a set of scripts to extend this dataset. Therefore, anyone can introduce new target images to the dataset and then, subsequently, generate the LR images to be reconstructed, calculate their transformation matrices and also produce the correct alignment among the low-resolution images.

7.2 Results for mobile devices

In the second experiment, we super resolved sequences of photos taken by an iPhone in burst mode. We assumed, in this application, that the objects in the scene remain static while capturing the images. In addition, we super resolved only plain objects, because the proposed methods are not designed to handle 3-d reconstructions.

This application **requires** fast responses. A person who takes a dozen photos in one second in a mobile device might not wait for minutes until the HR image is generated. In addition, the application must have low memory footprint, because the person might simultaneously receive and make voice calls, send messages, take, receive and send photos and videos, listen to music and/or play games while waiting for the reconstructed image. For such reasons, we use only GSR in this setup (due to its fast running time and low-memory footprint).

The five GSR variations took up to 50 seconds to super resolve 25 LR images into an HR image with $2,448 \times 3,264$ pixels (GSR_1, the fastest one, spends only 25 seconds). The visual result of the super-resolved images seems to have more details than the simple interpolation, even using a few images as input.

7.3 Results for license plates

The last experiment introduces the framework "*Eyes on the Target*", that handles the necessary conditions to identify a target license plate, using a novel methodology to locate, track, align, super resolve, and recognize its alphanumerics. We focus on enhancing the details in vehicle license plates that could help to identify a criminal suspect or activity in a crime scene, super resolving only a region of interest (ROI) of the video, and discarding less important parts.

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 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 did not include such feature in this work 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). If the user does not want to use the recognized alphanumerics, she can simply use the reconstructed image for a better visualization of the alphanumerics, or even as an improved input for a single-image super-resolution algorithm.

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 experiment shows that it is possible to increase the number of recognized characters using our super-resolution methods. Since GSR_4 recognized all seven license-plate characters in more input videos than the other variations, we chose GSR_4 as the default method for the reconstructed step (GSR_4 results are more blurred than the other methods, and the recognition task seems to work better using images with less noise). 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.

It is also interesting to point out that GSR_4, the algorithm that we choose as the default solution for the framework, produced the **worst** similarity values in *Experiment #1*. Such result supports our claim that the best solution to super resolve a set of images is highly dependent on the application.

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 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.

7.4 The registration problem

Registration plays an important role in a multi-frame SR. In all proposed methods, we need to know the alignment among the input observations. In the first experiment, since we generated the LR images, we could calculate and use the correct alignment among them. We do not expect, however, that such alignment is available in real-world applications. Therefore, we also perform a registration step, prior to the reconstruction, trying to approximate this experiment to real-world situations. Given a set of LR images, we found keypoints and descriptors (using SIFT, SURF and ORB) in all images and then matched them using k-NN and Flann.

For a fair comparison among the three feature detectors, we have chosen only one algorithm for the reconstruction step. Hence, different SSIM values in the results were exclusively due to the alignments produced by each detector. Additionally, we compared them to the correct alignment, to visualize the best results that the registration methods could achieve. Such experiment showed that classic feature detectors could, indeed, perform a good alignment. SIFT outperformed ORB and SURF, but ORB is the best solution for images with high dimensions, due to its faster runtime. We used this same methodology for the registration step in the experiment with mobile devices, and the results were similar.

In the third experiment, for super resolving the license plates, the pairs of keypoints found by the feature detectors in the two consecutive images were not good enough to calculate a transformation between such frames. Most of the keypoints found by the methods belong to the environment around the car, and they might not contribute to map one license plate onto the other. Therefore, we used a two-steps methodology to align the license plates when dealing with videos of fast-moving vehicles:

- First, we use the four corners around the license plate (selected by the user) to track the object of interest between the consecutive frames. Such tracking gives a pre-alignment between the frames, and the framework has five methods to find the re-occurrences of the license plates throughout the video:

    1. In the first method, we iteratively use the Lucas-Kanade optical flow to track the points, creating an homography between the frames. The keypoints in the reference frame are found with Shi-Tomasi corner detector, and only inside the license-plate region. We chose a pyramidal implementation of such optical flow, since we do not want to constrain our framework just to videos with slow-moving vehicles (when we go up in the pyramid, the images are downscaled, the small motions are removed, and the large motions become small motions).
    2. The second method uses a pyramidal implementation of dense optical flow. We do not create the homography matrix here, since the algorithm calculates the motion of each pixel in the image.
    3. In the third method for tracking, we use SIFT to find keypoints in the initial frame. This is possible because the user previously annotated the region describing the license plate of interest in the first step. We find SIFT keypoints

only inside the license-plate region (as in the first tracking solution with the Shi-Tomasi detector), and then we match them with the SIFT points found in the entire consecutive frame. The points are matched using k-NN and Flann, and the best matches estimate an homography matrix mapping the images.

4. The other two methods are similar to SIFT. However, they detect the key-points using, respectively, the feature detectors SURF and ORB. Regardless the tracking method, the tracked points in a frame F_k are also used as previous points in the frame F_{k+1}, and we then align the tracked objects with respect to the initial frame's license plate.

- After tracking, we refine the alignment with subpixel accuracy using optical flow once again (but more accurately, as the images are now pre-aligned). We use Lucas-Kanade optical flow and dense optical flow without pyramids, and inside a smaller search window (larger values increase the algorithm robustness to fast motion detection, but yield less accuracy). Optionally, as a third solution, the user can choose not to perform the refinement, and keep the pre-alignment found in the tracking step.

The experiment showed that 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. In addition, due to the pyramid layers, we did not have issues with fast-moving vehicles with both the LK and Dense tracker, and we 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. Dense optical flow, however, outperformed the other methods when there were cast shadows over the license plate.

The Lucas-Kanade method also outperformed Dense optical flow in the alignment step, when both are used without pyramid layers. The subpixel adjustment improved the number of hits in several cases. The Lucas-Kanade's runtime is also significantly lower than all the other methods for tracking and registration, so it was chosen as the default solution for both steps.

7.5 Number of images used as input

In Chap. 4, having the results from the *Experiment #1*, we have discussed that m^2 may be a good number of input images when we want to create a super-resolved image $m\times$ higher in both axis, and any number close to that may generate a good solution using all the proposed methods. However, with the experiment for license-plates in Chap. 6, we noticed 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 (it seems that the framework can correct problems with atmospheric/illumination conditions even using only two input frames). We also suggested that collecting five frames may be a good amount of input frames in such setup. We found at least two reasons not to use the number m^2 in the forensic framework:

1. In such setup, we do not have a fixed factor m to enhance the dimensions of the plate. Instead, we combine the frames onto a rectified license plate within a fixed-sized grid according to country-specific specifications. Hence, in this particular application, the dimensions of the ultimate super-resolved image does not depend the dimensions of the license-plate in the input frames.

2. Additionally, in this scenario, the videos usually contain fast-moving vehicles. Then, the license-plate image after a few number of frames might become very smaller than in the initial frame, and then it might not add relevant information to the reconstruction.

7.6 Future work

Due to the small dimensions of the license-plate images in the surveillance videos, the runtime of the proposed method impacted more the mobile application than the forensic setup. For future work, we start by highlighting that GSR (the fastest method, which has been chosen to super resolve the images in an always-on low-power environment) can become even faster: for now, all the variations have been implemented in Python (we may achieve faster runtime if using C) and they still have no parallelism. For such application, we can add parallelism to the method, taking advantage of multiple cores in recent mobile phones. It is also possible to study other nearest neighbors approaches in order to combine the LR pixels in GSR (using triangulation, for example, as we discussed in Chap. 3), compare the algorithms to a larger number of recent methods in the literature (putting all of them in this context of limited memory and battery), and adapt the method to run on a mobile phone (for now, we simulated the results only in Desktop computers and notebooks). Another interesting idea is to investigate if we can use the accelerometer and the gyroscope from such devices to calculate (or help to calculate) the motion between two consecutive images. It could aid the registration step.

For the forensic framework, a possible future direction 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. Other possibility is to automatically point out the number of input frames that returns the best result, sorting the results by the confidence rate that the OCR systems returns for each sequence. Moreover, we have already acquired videos of real-world unresolved cases from the Brazilian Federal Police. We did not add such videos in our experiments for ethical reasons, but we might relate possible good results in future work.

We are also planning to investigate other strategies to calculate the alignment between the input observations. According to [106], the accuracy of most registration algorithms is not enough for super resolution, which lead to annoying artifacts. They propose a strategy that measures the reliability of the estimated shifts and only choose the reliable frames. Such reliable shits then help to refine the other shifts, finding a refined HR

image. Similarly, the work in [107] claims that high accuracy image registration is critical for the success of multi-frame super resolution. However, the high-frequency of LR data is unreliable due to the aliasing effect of sub-sampling, which will deteriorate the accuracy of registration. They proposed then to resolve the aliasing by converting the LR images to high-resolution (HR) domain and then perform registration on the restored HR spectrum.

Finally, another future work is to develop a framework for super resolution of hyperspectral imaging [117, 88], adding a contribution to the Microscopy field. A hyperspectral image provides ample spectral information to identify and distinguish spectrally unique materials, for more accurate and detailed information extraction. Such images are acquired by an instrument that can simultaneously record spectral and spatial information of a sample, facilitating the visualization of chemical distribution or chemical compositions. For example, in medical image analysis, full information about the scene radiance can be used to detect anomalies that are not distinguishable from RGB images alone. Also, faithful color reproduction is critical for museums to record their artwork [56]. Moreover, the use of hyperspectral imagery has been shown to enhance the performance of a number of computer vision tasks, including segmentation, tracking, and recognition [78]. However, spatial resolution of current hyperspectral imaging systems is severely limited compared to RGB cameras (therefore, a good application to super resolution).

To validate the framework, we might use a sequence of hyperspectral images generated by a Near Infrared Spectroscopy (NIRS) [65] as input for our SR algorithm. Despite the low spatial resolution of the NIRS, we can manually move the objects along its X and Y axis, creating images with strictly translational displacements (perfect inputs for our proposed methods). Therefore, the registration step for this application must find only rigid transformations between two input images. Moreover, hyperspectral images contain lots of informations (in addition to the RGB channels from the traditional images) that can facilitate the alignment between the input sequence, improving the SR results. For example, we can use many of their bands to detect and match keypoints if the amount of keypoints in one single band is not enough for a good alignment.

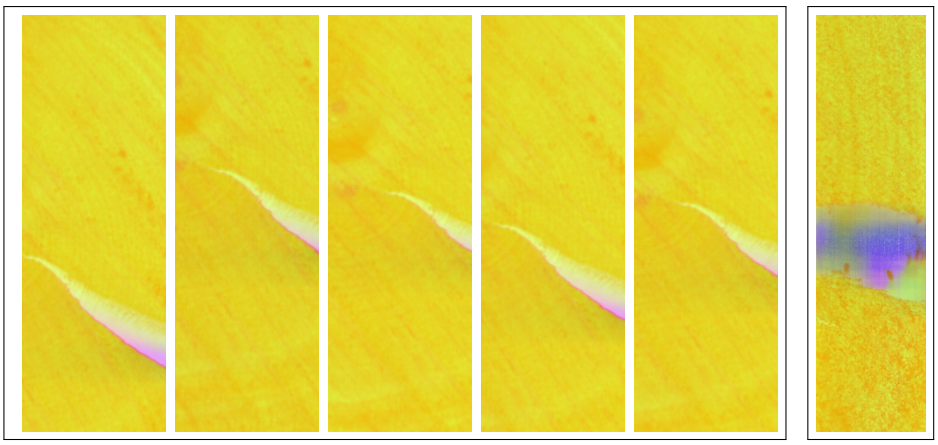
A few works have been proposed to super resolve hyperspectral images. In [56], authors generate an HR hyperspectral image using an LR hyperspectral camera and a high-resolution RGB camera. [135] presents a single-image SR for hyperspectral images with sparsity based regularization. [132] proposes a multi-frame super resolution that uses multiple components of the hyperspectral image to improve the accuracy in registration step. To reconstruct the image, it introduces a MAP-based SR in which PCA is employed to reduce computational load and remove noise. Finally, [1] models the hyperspectral image acquisition process and presents a method for applying SR hyperspectral images using such model. The method fuses information from multiple observations and spectral bands to improve spatial resolution and reconstruct the spectrum of the observed scene as a combination of a small number of spectral basis functions.

The most direct solution for hyperspectral image super resolution is to super resolve every separate spectral band individually. However, this approach has a huge computational load due to the high dimensionality of the hyperspectral images (both in motion estimation and in image reconstruction). Furthermore, high correlation exists across the spectral bands, so considering these bands separately will not fully exploit the correlation

across them [132]. Usually, a Principal Component Analysis (PCA) is used to reduce the dimensionality of the hyperspectral image [1, 28, 52], retaining as much information as possible. The first few principal components contain the most information of the data, and such components could be seen as the three bands of an RGB image. Any of the proposed methods could then be used to create a super-resolved image. However, after super resolving the three principal components, we could use the same geometric correspondences found by the reconstruction step to transpose the SR to the other bands.

To evaluate this framework, the Institute of Chemistry at Unicamp provided a Near Infrared Spectroscopy, from which we may capture a set of input images. Such device can capture 256 different spectral bands, and the material can undergo strictly translational displacements along the device. Moreover, such NIRS allows us to capture a set of hyperspectral images in a lower resolution and then to capture the same material with a pixel resolution 5× higher than the others. Then, for the quantitative validation, we might super resolve the LR observations, and compare the result to such image with higher dimensions. For example, we can calculate the similarity individually for each band, and the final similarity may be a combination of such similarities. For the qualitative validation, we might have a more interesting scenario: we are planning to add some small impurity to the material, in a way that such impurity can be visualized only in the HR hyperspectral image. So, a good SR technique should be able to detect such detail after enhancing the set of input images.

In Figure 7.1a we see five examples of low-resolution images of a piece of wood captured by this NIRS after a PCA transform. A piece of the HR target image is shown in Figure 7.1b.



(a) **(b)**

Figure 7.1: LR and HR hyperspectral images captured by a NIR device. All images have been compressed with PCA, to be possible to visualize them in screen.

Bibliography

[1] Toygar Akgun, Student Member, Yucel Altunbasak, Senior Member, and Russell M. Mersereau. Super-resolution reconstruction of hyperspectral images. *IEEE Trans. on Image Proc*, pages 1860–1875, 2005.

[2] Angelos Amanatiadis, Loukas Bampis, and Antonios Gasteratos. Accelerating image super-resolution regression by a hybrid implementation in mobile devices. In *IEEE International Conference on Consumer Electronics (ICCE)*, pages 335–336, 2014.

[3] Christos-Nikolaos E. Anagnostopoulos, Ioannis E. Anagnostopoulos, Ioannis D. Psoroulas, Vassili Loumos, and Eleftherios Kayafas. License plate recognition from still images and video sequences: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 9(3):377–391, 2008.

[4] Panos J. Antsaklis and Anthony N. Michel. *Linear Systems. [electronic resource].* Boston, MA : Birkhäuser Boston, 2006., 2006.

[5] Harrison Barrett and Kyle Myers. *Foundations of image science.* Wiley-Interscience, 2004.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[7] Marcelo Bertalmío, Andrea L. Bertozzi, and Guillermo Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 355–362, 2001.

[8] Marcelo Bertalmío, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 417–424, 2000.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., 2006.

[10] Thierry Blu, Phillippe Thevenaz, and Michael Unser. Linear interpolation revitalized. *IEEE Transactions on Image Processing*, 13(5):710–719, 2004.

[11] Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm. Technical report, Intel Corporation Microprocessor Research Labs, 2000.

[12] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[13] Thomas M. Breuel. The ocropus open source ocr system. In *The International Society for Optical Engineering (SPIE)*, volume 6815, 2008.

[14] Peter J. Burt and Raymond J. Kolczynski. Enhanced image capture through fusion. In *1993 (4th) International Conference on Computer Vision*, pages 173–182, May 1993.

[15] Gulcin Caner, A. Murat Tekalp, and Wendi Heinzelman. Super resolution recovery for multi-camera surveillance imaging. In *International Conference on Multimedia and Expo, 2003 (ICME)*, pages I–109, 2003.

[16] Raymond H. Chan, Tony F. Chan, Michael K. Ng, Wun-Cheung Tang, and Chiu-Kwong T. Wong. Preconditioned iterative methods for high-resolution image reconstruction with multisensors. volume 3461, pages 348–357, 1998.

[17] Tony F. Chan, Michael K. Ng, Andy C. Yau, and Andy M. Yip. Superresolution image reconstruction using fast inpainting algorithms. *Applied and Computational Harmonic Analysis*, 23(1):3–24, 2007.

[18] Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan Chen. Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems*, 5(1):42–53, 2004.

[19] Priyam Chatterjee, Sujata Mukherjee, Subhasis Chaudhuri, and Guna Seetharaman. Application of papoulis–gerchberg method in image super-resolution and inpainting. In *The computer journal*, pages 80–89, 2009.

[20] Subhasis Chaudhuri. *Super-resolution imaging*. Springer Science & Business Media, 2001.

[21] Peter Cheeseman, Bob Kanefsky, Richard Kraft, John Stutz, and Hanson Robin. Super-resolved surface reconstruction from multiple images. In Glenn R. Heidbreder, editor, *Maximum Entropy and Bayesian Methods: Santa Barbara, California, U.S.A., 1993*, pages 293–308. Springer Netherlands, Dordrecht, 1996.

[22] T. Chen and Rui J.P. de Figueiredo. Image decimation and interpolation techniques based on frequency domain analysis. *IEEE Transactions on Communications*, 32(4):479–484, 1984.

[23] L. P. Chew. Constrained delaunay triangulations. In *Third Annual Symposium on Computational Geometry*, pages 215–222, 1987.

[24] Ming-Chao Chiang and Terrance E. Boult. Efficient image warping and super-resolution. In *IEEE Workshop on Applications of Computer Vision*, pages 56–, 1996.

[25] Ming-Chao Chiang and Terrance E. Boult. Local blur estimation and super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 821–, 1997.

[26] Chung-Hua Chu. Super-resolution image reconstruction for mobile devices. *Multimedia systems*, 19(4):315–337, 2013.

[27] T. J. Connolly and R. G. Lane. Gradient methods for superresolution. In *Proceedings of International Conference on Image Processing*, volume 1, pages 917–920 vol.1, 1997.

[28] J.A. Costa, G. Fontoura, F.L. Gorgônio, and A.M. Martins. Compressão auto adaptativa de imagens multiespectrais. *XIV Simpósio Brasileiro de Senriamento Remoto*, pages 7181–7188, April 2009.

[29] Paulo Diniz, Sergio Netto, and Eduardo Da Silva. *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, 2002.

[30] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, volume 8692, pages 184–199. Springer, 2014.

[31] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[32] Shan Du, Mohammad Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325, 2013.

[33] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

[34] M. Elad and A. Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE Transactions on Image Processing*, 6(12):1646–1658, 1997.

[35] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, pages 363–370, 2003.

[36] Sina Farsiu, M. Dirk Robinson, Michael Elad, and Peyman Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004.

[37] Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics*. Springer, 1999.

[38] William Freeman, Thouis Jones, and Egon Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[39] Martin Fuhry and Lothar Reichel. A new tikhonov regularization method. *Numer. Algorithms*, 59(3):433–445, 2012.

[40] Ming Gao and Shiyin Qin. High performance super-resolution reconstruction of multi-frame degraded images with local weighted anisotropy and successive regularization. *Optik - International Journal for Light and Electron Optics*, 126(23):4219–4227, 2015.

[41] Ralph Gerchberg. Super-resolution through error energy reduction. *Journal of Modern Optics*, 21:709–720, 1974.

[42] Rafael Gonzalez and Richard Woods. *Digital Image Processing*. Pearson/Prentice Hall, 2008.

[43] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1312–1317, 2011.

[44] Qizi Huangpeng, Xiangrong Zeng, Jun Fan, Jing Feng, and Quan Sun. Improved multi-frame super resolution via multiframe blind deblurring using the alternating direction method of multipliers. In *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 281–285, 2015.

[45] Michal Irani and Shmuel Peleg. Super resolution from image sequences. *Intl. Conference on Pattern Recognition*, C(90):115–120, 1990.

[46] Michal Irani and Shmuel Peleg. Improving resolution by image registration. *CVGIP: Graphical models and image processing*, 53(3):231–239, 1991.

[47] K. Al Ismaeil, D. Aouada, T. Solignac, B. Mirbach, and B. Ottersten. Real-time non-rigid multi-frame depth video super-resolution. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 8–16, 2015.

[48] K. Al Ismaeil, D. Aouada, T. Solignac, B. Mirbach, and B. Ottersten. Real-time enhancement of dynamic depth videos with non-rigid deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.

[49] Itseez. *The OpenCV Reference Manual*, 3.1 edition, December 2015.

[50] Baraka Jacob Maiseli, Nassor Ally, and Huijun Gao. A noise-suppressing and edge-preserving multiframe super-resolution image reconstruction method. *Image Communication*, 34(C):1–13, 2015.

[51] Seokhwa Jeong, Inhye Yoon, Jaehwan Jeon, and Joonki Paik. Multi-frame example-based super-resolution using locally directional self-similarity. In *2015 IEEE International Conference on Consumer Electronics (ICCE)*, pages 631–632, 2015.

[52] M.A. Joshi. *Digital Image Processing: An Algorithm Approach.* PHI Learning, 2006.

[53] Avinash C. Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*, chapter 7, pages 275–296. IEEE press, 1988.

[54] Sandeep Kanumuri, Onur Guleryuz, and M. Reha Civanlar. Fast super-resolution reconstructions of mobile video using warped transforms and adaptive thresholding. In *Optical Engineering+ Applications*, pages 66960T–66960T, 2007.

[55] Toshiyuki Kato, Hideitsu Hino, and Noboru Murata. Double sparsity for multi-frame super resolution. *Neurocomput.*, 240(C):115–126, May 2017.

[56] Rei Kawakami, John Wright, Yu-Wing Tai, Yasuyuki Matsushita, Moshe Ben-Ezra, and Katsushi Ikeuchi. High-resolution hyperspectral imaging via matrix factorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2329–2336, 2011.

[57] A. Khan, M. A. Khan, F. Obaid, S. Jadoon, M. A. Khan, and M. Sikandar. A novel multi-frame super resolution algorithm for surveillance camera image reconstruction. In *2015 First International Conference on Anti-Cybercrime (ICACC)*, pages 1–6, 2015.

[58] Minjae Kim and Hanseok Ko. Resolution enhancement of roi from surveillance video using bernstein interpolation. In *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 331–336, 2011.

[59] Seung Kim, Nirmal Bose, and Hector Valenzuela. Recursive reconstruction of high resolution image from noisy undersampled multiframes. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(6):1013–1027, 1990.

[60] T. Köhler, X. Huang, F. Schebesch, A. Aichert, A. Maier, and J. Hornegger. Robust multiframe super-resolution employing iteratively re-weighted minimization. *IEEE Transactions on Computational Imaging*, 2(1):42–58, 2016.

[61] Joseph Kolibal and Daniel Howard. The novel stochastic bernstein method of functional approximation. In *NASA/ESA Conference on Adaptive Hardware and Systems*, pages 97–100, 2006.

[62] Amine Laghrib, Abdelghani Ghazdali, Abdelilah Hakim, and Said Raghay. A multi-frame super-resolution using diffusion registration and a nonlocal variational image restoration. *Computers & Mathematics with Applications*, 72(9):2535–2548, 2016.

[63] Charles L. Lawson and Richard J. Hanson. *Solving least squares problem.* Philadelphia, PA: SIAM, 1995., 1995.

[64] Thomas M. Lehmann, Claudia Gönner, and Klaus Spitzer. Survey: Interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18:1049–1075, 1999.

[65] Neil Lewis, Eunah Lee, and Linda Kidder. Combining imaging and spectroscopy: Solving problems with near-infrared chemical imaging. *Microscopy Today*, 12(6):8–12, 2004.

[66] Wei Lina and Liu Ying. A license plate super-resolution reconstruction algorithm based on manifold learning. In *International Conference on Computational Science and Engineering (CSE)*, pages 1855–1859, 2014.

[67] Xueting Liu, Daojin Song, Chuandai Dong, and Hongkui Li. Map-based image super-resolution reconstruction. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 2(1):102 – 105, 2008.

[68] David G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157 vol.2, 1999.

[69] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[70] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[71] Ziyang Ma, Renjie Liao, Xin Tao, L. Xu, J. Jia, and Enhua Wu. Handling motion blur in multi-frame super-resolution. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5224–5232, 2015.

[72] Erik Meijering. A chronology of interpolation: from ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, 90(3):319–342, 2002.

[73] Olivier Le Meur and Christine Guillemot. Super-resolution-based inpainting. In *European Conference on Computer Vision (ECCV)*, pages 554–567, 2012.

[74] Idriss El Mourabit, Mohammed El Rhabi, Abdelilah Hakim, Amine Laghrib, and Eric Moreau. A new denoising model for multi-frame super-resolution image reconstruction. *Signal Processing*, 132(C):51–65, 2017.

[75] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 331–340, 2009.

[76] Haidawati Nasir, Vladimir Stanković, and Stephen Marshall. Singular value decomposition based fusion for super-resolution image reconstruction. *Signal Processing: Image Commununication*, 27(2):180–191, 2012.

[77] Kamal Nasrollahi and Thomas B. Moeslund. Super-resolution: a comprehensive survey. *Machine Vision and Applications*, 25:1423–1468, 2014.

[78] Hien Van Nguyen, A. Banerjee, and R. Chellappa. Tracking via object reflectance using a hyperspectral video camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 44–51, June 2010.

[79] Nhat Nguyen, P. Milanfar, and G. Golub. A computationally efficient superresolution image reconstruction algorithm. *IEEE Transactions on Image Processing*, 10(4):573–583, 2001.

[80] Nhat Xuan Nguyen. *Numerical algorithms for image superresolution*. PhD thesis, Stanford University, 2000.

[81] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time Signal Processing*. Prentice-Hall, Inc., 2 edition, 1999.

[82] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[83] Athanasios Papoulis. A new algorithm in spectral analysis and band-limited extrapolation. *IEEE Transactions on Circuits and Systems Ii: Analog and Digital Signal Processing*, 22:735–742, 1975.

[84] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE Signal Processing Magazine*, 20(3):21–36, 2003.

[85] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2011.

[86] Tuan Pham, Lucas van Vliet, and Klamer Schutte. Robust fusion of irregularly sampled data using adaptive normalized convolution. *EURASIP Journal on applied signal processing*, 2006(1):1–12, 2006.

[87] Lyndsey Pickup, David Capel, Stephen Roberts, and Andrew Zisserman. Bayesian image super-resolution, continued. In *Advances in Neural Information Processing Systems*, pages 1089–1096, 2006.

[88] J.A. Fernández Pierna, P. Vermeulen, O. Amand, A. Tossens, P. Dardenne, and V. Baeten. {NIR} hyperspectral imaging spectroscopy and chemometrics for the detection of undesirable substances in food and feed. *Chemometrics and Intelligent Laboratory Systems*, 117(0):233 – 239, 2012. Special Issue Section: Selected Papers from the 1st African-European Conference on Chemometrics, Rabat, Morocco, September 2010 Special Issue Section: Preprocessing methods Special Issue Section: Spectroscopic imaging.

[89] Ye Qing, Zhu Liang Hong, Zhu Su Hong, and Li Xue. A new fast algorithm to rectify tilt image of vehicle license plates. In *IEEE Control Conference*, pages 485–488, 2007.

[90] Hitesh Rajput, Tanmoy Som, and Soumitra Kar. Using radon transform to recognize skewed images of vehicular license plates. *IEEE Computer*, 49(1):59–65, 2016.

[91] C. Radhakrishna Rao, Helge Toutenburg, and Shalabh Christian Heumann. *Linear Models and Generalizations: Least Squares and Alternatives*. Springer-Verlag Berlin Heidelberg, 3rd edition, 2008.

[92] Yaniv Romano, John Isidoro, and Peyman Milanfar. Raisr: Rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, 3(1):110–125, 2017.

[93] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[94] S. S. Lertrattanapanich and N.K. Bose. High resolution image formation from low resolution frames using delaunay triangulation. *Image Processing, IEEE Transactions on*, 11(12):1427–1441, 2002.

[95] Gerald Schaefer and Michal Stich. Ucid: an uncompressed color image database. In *Storage and Retrieval Methods and Applications for Multimedia*, volume 5307, pages 472–480, 2003.

[96] Hilario Seibel Jr, Siome Goldenstein, and Anderson Rocha. Fast and effective geometric k-nearest neighbors multi-frame super-resolution. In *IEEE International Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 103–110, 2015.

[97] Hilario Seibel Jr, Siome Goldenstein, and Anderson Rocha. Eyes on the target: Super-resolution and license-plate recognition in low-quality surveillance videos. *IEEE Access*, 5:20020–20035, 2017.

[98] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 1591–1595, 1995.

[99] Minmin Shen and Ping Xue. Low-power video acquisition with super-resolution reconstruction for mobile devices. *IEEE Transactions on Consumer Electronics*, 56(4):2520–2528, 2010.

[100] Jianbo Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[101] Jiazheng Shi and S.E. Reichenbach. Image interpolation by two-dimensional parametric cubic convolution. *IEEE Transactions on Image Processing*, 15(7):1857–1870, 2006.

[102] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[103] I. Silva, B. Mederos, L. Ortega-Maynez, and S. Cabrera. Multi-frame super-resolution for mixed gaussian and impulse noise based on blind inpainting. In *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, pages 208–213, 2015.

[104] Karen Simonyan, Sergey Grishin, Dmitriy Vatolin, and Dmitriy Popov. Fast video super-resolution via classification. In *IEEE International Conference on Image Processing*, pages 349–352, 2008.

[105] Ray Smith and Google Inc. An overview of the tesseract ocr engine. In *IEEE International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633, 2007.

[106] Q. Song, R. Xiong, X. Fan, S. Ma, and W. Gao. Registration-reliability based strategy to enhance multi-frame super-resolution algorithms. In *2015 Visual Communications and Image Processing (VCIP)*, pages 1–4, 2015.

[107] Q. Song, R. Xiong, X. Zhang, S. Ma, and W. Gao. Registration of under-sampled images via higher resolution spectrum restoration. In *2015 Visual Communications and Image Processing (VCIP)*, pages 1–4, 2015.

[108] A. Stankiewicz, T. Marciniak, A. Dąbrowski, M. Stopa, E. Marciniak, and A. Michalski. Matching 3d oct retina images into super-resolution dataset. In *2016 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 130–137, 2016.

[109] Henry Stark and Peyma Oskoui. High-resolution image recovery from image-plane arrays, using convex projections. *Journal of the Optical Society of America*, 6(11):1715–1726, 1989.

[110] Kaggere V. Suresh, G. Mahesh Kumar, and A. N. Rajagopalan. Superresolution of license plates in real traffic videos. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):321–331, 2007.

[111] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., 1st edition, 2010.

[112] Yu-Wing Tai, Shuaicheng Liu, M.S. Brown, and S. Lin. Super resolution using edge prior and single image detail synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2400–2407, 2010.

[113] Qing Tang, Zhi-Yan Wang, Xiao wei Xu, Ying hong Liang, and Xiao ye Cao. A rapid inclined distortion rectification method for vehicle licenses plate images. In

IEEE International Conference on Machine Learning and Cybernetics, volume 5, pages 2744–2748, 2008.

[114] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9(1):23–34, 2004.

[115] Jing Tian and Kai-Kuang Ma. A survey on super-resolution imaging. *Signal, Image and Video Processing*, 5(3):329–342, 2011.

[116] Andrey N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston & Sons, 1977.

[117] Chieu D. Tran. Infrared Multispectral Imaging: Principles and Instrumentation. *Applied Spectroscopy Reviews*, 38:133–153, 2003.

[118] Roger Tsai and Thomas Huang. Multiframe image restoration and registration. *Advances in Computer Vision and Image Processing*, 1(2):317–339, 1984.

[119] C. W. Tseng, H. R. Su, S. H. Lai, and J. Liu. Depth image super-resolution via multi-frame registration and deep learning. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–8, 2016.

[120] Patrick Vandewalle, Luciano Sbaiz, and Martin Vetterli. Registration for superresolution: Theory, algorithms, and applications in image and mobile video enhancement. In *Super-Resolution Imaging*. Taylor& Francis / CRC Press, 2011.

[121] Patrick Vandewalle, Sabine Süsstrunk, and MartinVetterli. Superresolution images reconstructed from aliased images. In *SPIE/IS&T Visual Communications and Image Processing Conference*, volume 5150, pages 1398–1405, 2003.

[122] Patrick Vandewalle, Sabine Süsstrunk, and Martin Vetterli. A frequency domain approach to registration of aliased images with application to super-resolution. *EURASIP Journal on Applied Signal Processing*, 2006:1–14, 2006.

[123] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

[124] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[125] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[126] Xin Yang, Tianshu Liu, and Dake Zhou. A multi-frame adaptive super-resolution method using double channel and regional pixel information. *Optik - International Journal for Light and Electron Optics*, 126(24):5850–5858, 2015.

[127] Tomonari Yoshida, Tomokazu Takahashi, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase. Robust face super-resolution using free-form deformations for low-quality surveillance video. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 368–373, 2012.

[128] Jie Yuan, Si dan Du, and Xiang Zhu. Fast super-resolution for license plate image reconstruction. In *International Conference on Pattern Recognition (ICPR)*, pages 1–4, 2008.

[129] Yuan Yuan, Sabu Emmanuel, Yuming Fang, and Weisi Lin. Visual object tracking based on backward model validation. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(11):1898–1910, 2014.

[130] Karim Zarei, Wim Van Aarle, Kees Joost Batenburg, and Jan Sijbers. Super-resolution of license plate images using algebraic reconstruction technique. *Journal of Image and Graphics*, 1:94 – 98, 2013.

[131] D. Zhang, P. M. Jodoin, C. Li, Y. Wu, and G. Cai. Novel graph cuts method for multi-frame super-resolution. *IEEE Signal Processing Letters*, 22(12):2279–2283, 2015.

[132] Hongyan Zhang, Liangpei Zhang, and Huanfeng Shen. A super-resolution reconstruction algorithm for hyperspectral images. *Signal Processing*, 92(9):2082–2096, 2012.

[133] Xin Zhang, Edmund Y. Lam, Ed X. Wu, and Kenneth K. Y. Wong. Application of tikhonov regularization to super-resolution reconstruction of brain MRI images. In *Intl. Conf. on Medical Imaging and Informatics*, pages 51–56, 2007.

[134] W. Zhao, H. Sawhney, M. Hansen, and S. Samarasekera. Super-fusion: a super-resolution method based on fusion. In *Object recognition supported by user interaction for service robots*, volume 2, pages 269–272 vol.2, 2002.

[135] Yongqiang Zhao, Jinxiang Yang, Qingyong Zhang, Lin Song, Yongmei Cheng, and Quan Pan. Hyperspectral imagery super-resolution by sparse representation and spectral regularization. *EURASIP Journal on applied signal processing*, 2011:87, 2011.

[136] Marcelo Victor Wüst Zibetti. *Super-resolução simultânea para sequência de imagens*. PhD thesis, Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia Elétrica., 2007.

[137] Assaf Zomet and Shmuel Peleg. Efficient super-resolution and applications to mosaics. In *ICPR*, 2000.

[138] Assaf Zomet, Alex Rav-Acha, and Shmuel Peleg. Robust super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 645–650, 2001.

Appendix A

Additional charts and results

In this appendix, we show additional results and charts from our experiments that might contribute to a better understanding about the results. In Secs. A.1 through A.5 we focus on the Experiment #1. First, in Secs. A.1, we compare the proposed SR methods in terms of PSNR and MSE. In Sec. A.2, we compare GSR to other traditional methods in the literature. Then, in Secs. A.3 through A.5, we present charts about the first round of such experiment, in which we choose the best parameters for the Tikhonov-based method. Finally, in Sec. A.6, we focus on the Experiment #3, introducing another accuracy measure to evaluate the steps in the forensic framework.

A.1 Best reconstruction method

Now, we verify which is the best proposed method in terms of PSNR and MSE. In Fig. A.1, we see that GSR outperforms all the methods in terms of PSNR (the higher the PSNR value, the more similar one image to the other).

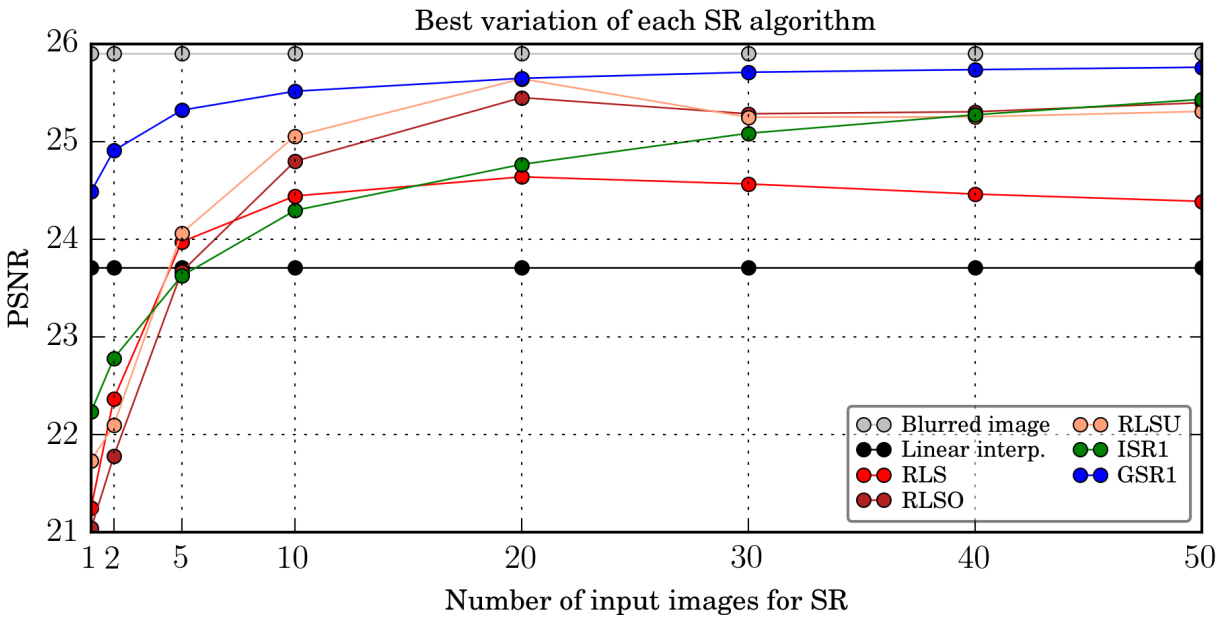


Figure A.1: Comparison among *RLS*, *ISR* and *GSR* in terms of PSNR.

In Fig. A.2, GSR also outperforms the other methods in terms of MSE (the lowest the MSE value, the more similar one image to the other.

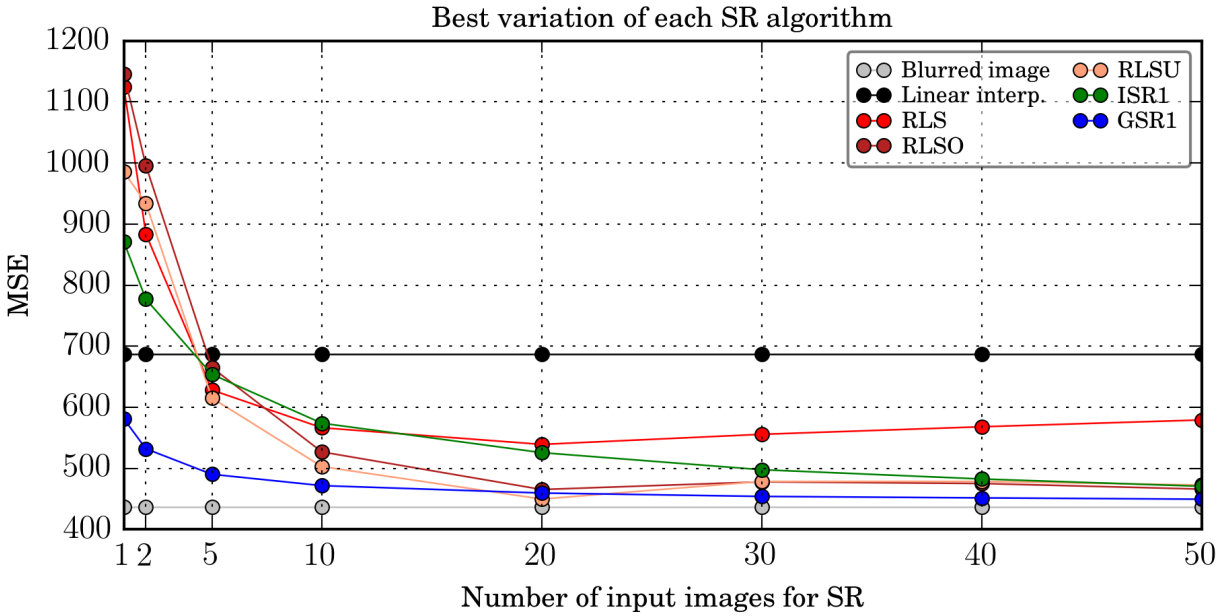


Figure A.2: Comparison among RLS, ISR and GSR in terms of MSE.

In Fig. A.3, we measure the similarity between each super-resolved image I_{SR} to the blurred image I_G, in terms of PSNR. We can see in Fig. A.3 that GSR_1 has, on average, the best PSNR values.

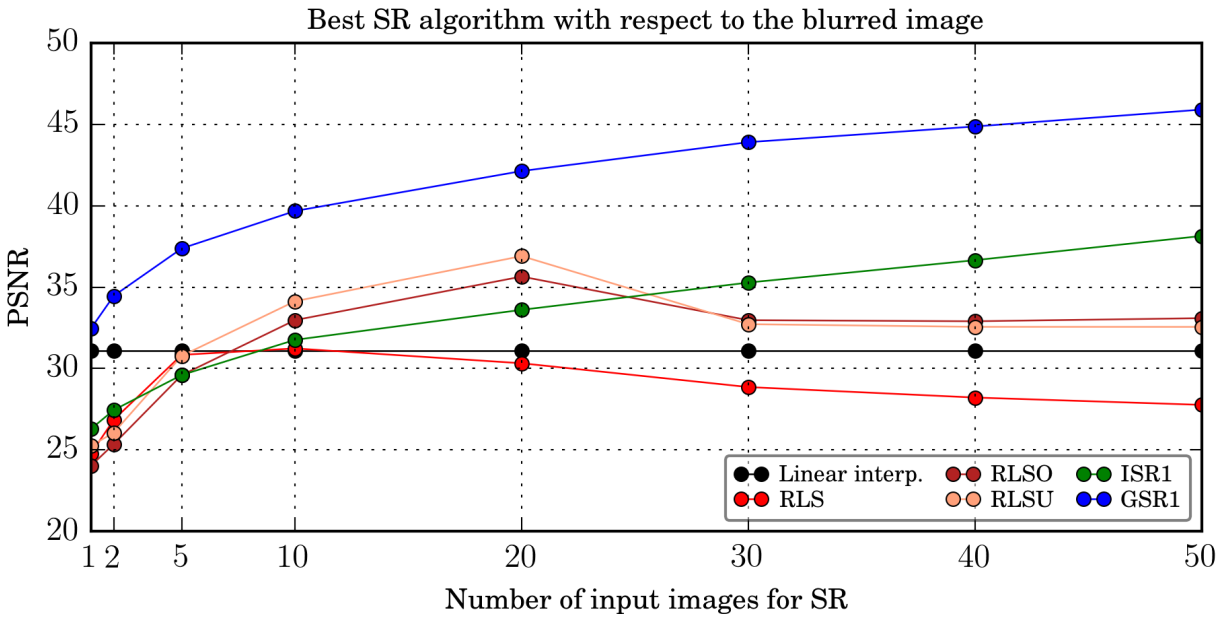


Figure A.3: Comparison among RLS, ISR and GSR, with respect to the blurred image that generated the pool of LR images.

Finally, in Fig. A.4, we measure the similarity between each super-resolved image I_{SR} to the blurred image I_G, in terms of MSE. We can see in Fig. A.4 the MSEs produced by GSR_1 are, on average, almost 0 (the minimum value for this similarity metric).

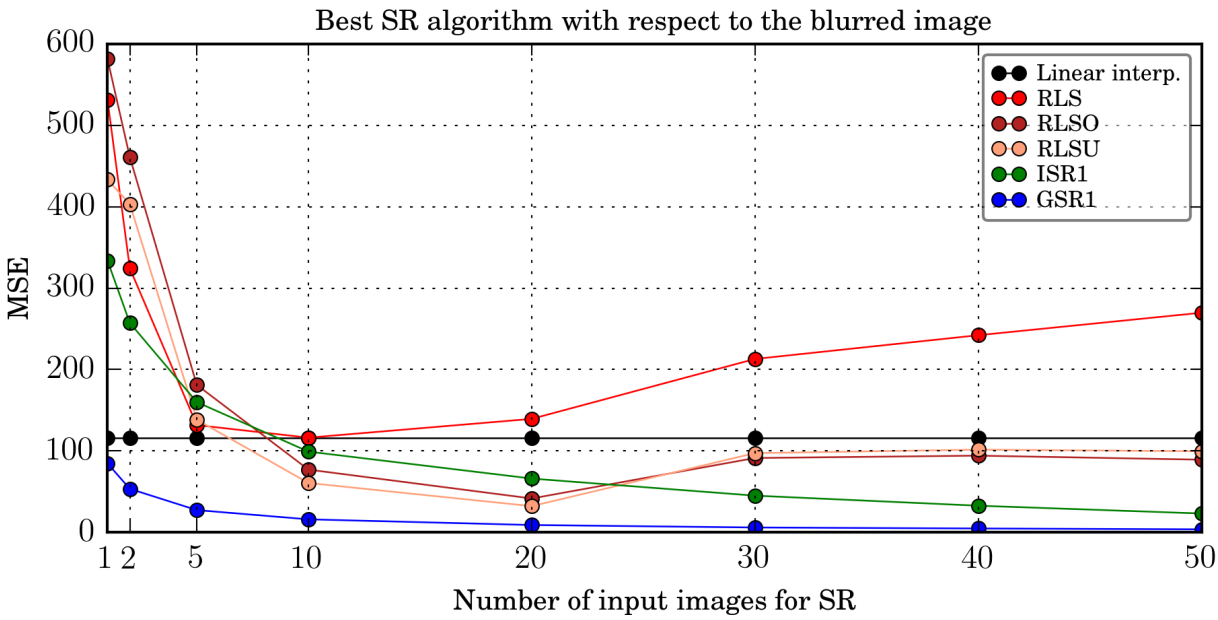


Figure A.4: Comparison among RLS, ISR and GSR, with respect to the blurred image that generated the pool of LR images.

A.2 Other methods in the literature

Now, we compare GSR_1 to other reconstruction algorithms available in the literature, using the implementations in MATLAB® from [122]: An algorithm inspired in the works of Papoulis [83] and Gerchberg [41] (PG); Iterated Back Projection (IBP) [45, 46]; Robust Super-Resolution (RS) [138]; Projection Onto Convex Sets ($POCS$) [109, 34, 136]; and Structure-Adaptive Normalized Convolution (NC) [86] (see Section 2.2 for more details about these methods). Here, we use only up to 35 input images and six target images from the dataset, due to the long runtime spent by classic algorithms.

For a fair comparison, we executed all algorithms using the correct alignment in the registration step. All classic algorithms, except NC, failed to super-resolve more than two input images. The chart, in Figure A.5, summarizes the comparison among GSR_1, PG, IBP, RS, $POCS$, and NC.

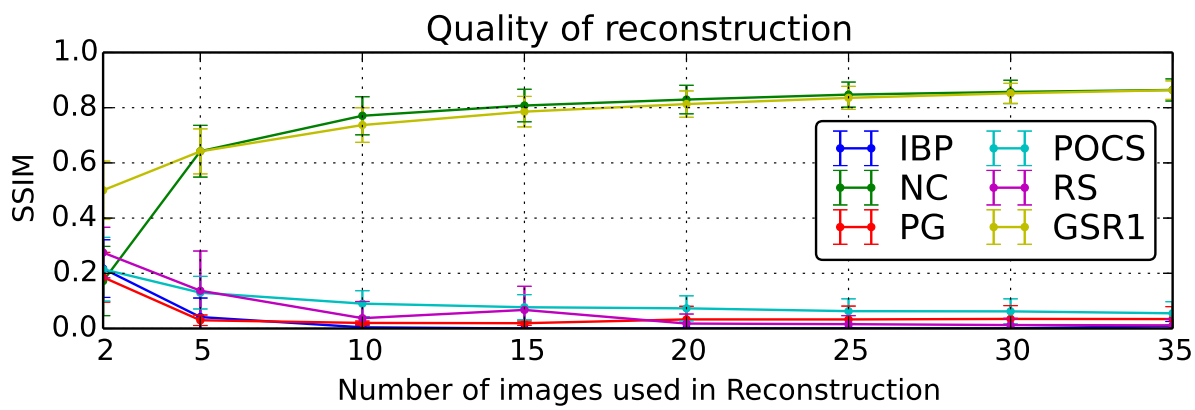


Figure A.5: Quality of different reconstruction algorithms as a function of the number of LR input images.

Figure A.6 shows a comparison between the runtime of the best investigated reconstruction algorithm (GSR_1) and the best classic one in the experiments (NC). Although GSR_1 and NC present similar SSIM values as we increase the number of input images (Figure A.5), we can conclude, from Figure A.6, that GSR_1 is more advantageous due to its low execution time (two orders of magnitude faster).

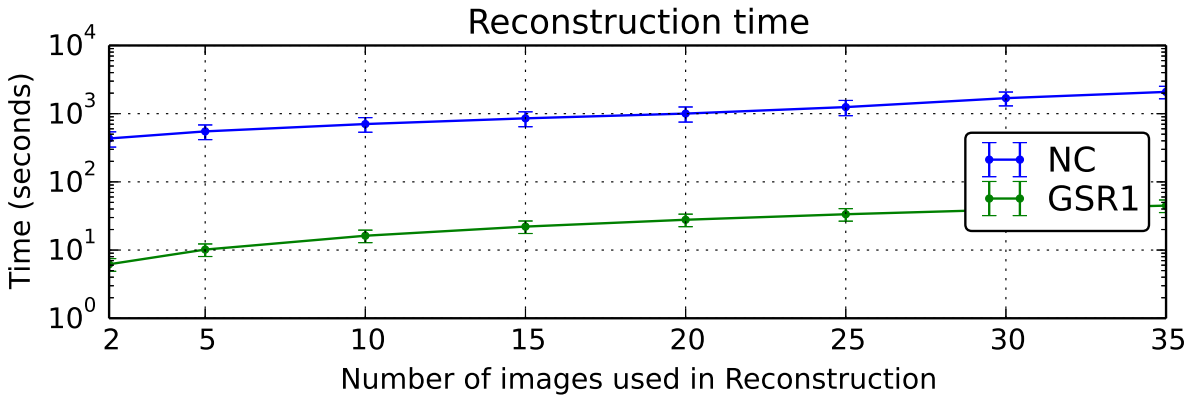


Figure A.6: Reconstruction runtime (logarithmic scale for viewing purposes).

A.3 RLS parameters

The chart in Fig. A.7 shows the quality of RLS using $\alpha = 0.1$ and different values for σ and d, for all images in $\mathscr{D}_1$. Each curve in Fig. A.7 is the average of the SSIM for all the 70 target images in the pool, using from one to 50 images as input for the super resolution. The best Gaussian parameters in the chart are $\sigma = 0.5$ and $d = 3$, but the results for $d = 5$ are very similar to $d = 3$.

The chart in Fig. A.8 shows the quality for $\alpha = 0.5$, on average. When we use $n \geq 40$ images as input, the best SSIM values are found with $(\sigma = 0.5, d = 3)$. However, such highest similarity seems very similar in the chart for $(\sigma = 0.5, d = 5)$ and $(\sigma = 0.7, d = 1)$.

Finally, Fig. A.9 shows the quality of the super resolution using $\alpha = 1.0$.

Unlike $\alpha = 0.1$ and $\alpha = 0.5$, Fig. A.9 shows that SSIM values for $\alpha = 1.0$ are better using $(\sigma = 0.7, d = 1)$ than using $\sigma = 0.5$, on average.

A.4 RLSO parameters

The chart in Fig. A.10 shows that the best results for RLSO, on average, are found with $\sigma = 0.7$ and $\alpha = 0.1$.

A.5 RLSU parameters

Fig. A.11 shows that $\alpha = 0.1$ is the best value for the regularization parameter using the uniform distribution.

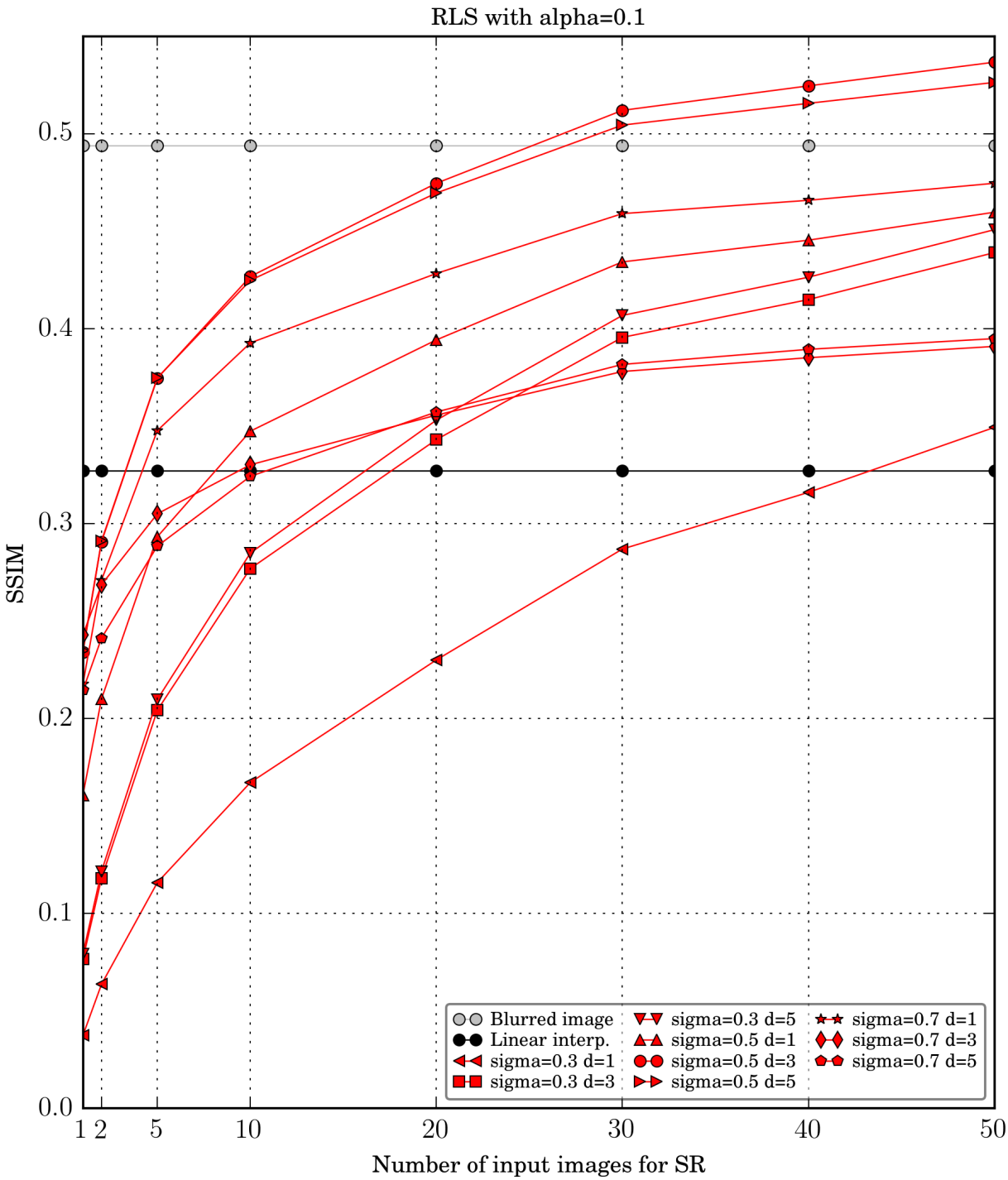


Figure A.7: SSIM values for RLS using $\alpha = 0.1$ and different values for σ and the kernel size. Each curve is the average for all the images from the pool in the subset $\mathscr{D}_1$.

A.6 Framework accuracy

In Chap. 6, we investigated the number of videos in which the framework correctly identified all the seven license-plate alphanumerics (7 is the maximum number of license-plate characters in our dataset) for a given method. We referred to this accuracy number as Acc_7. To find this number, we verified if the method scored seven hits at least once (using from 1 to 10 input frames), in each video. Now, we also calculate the number of videos in

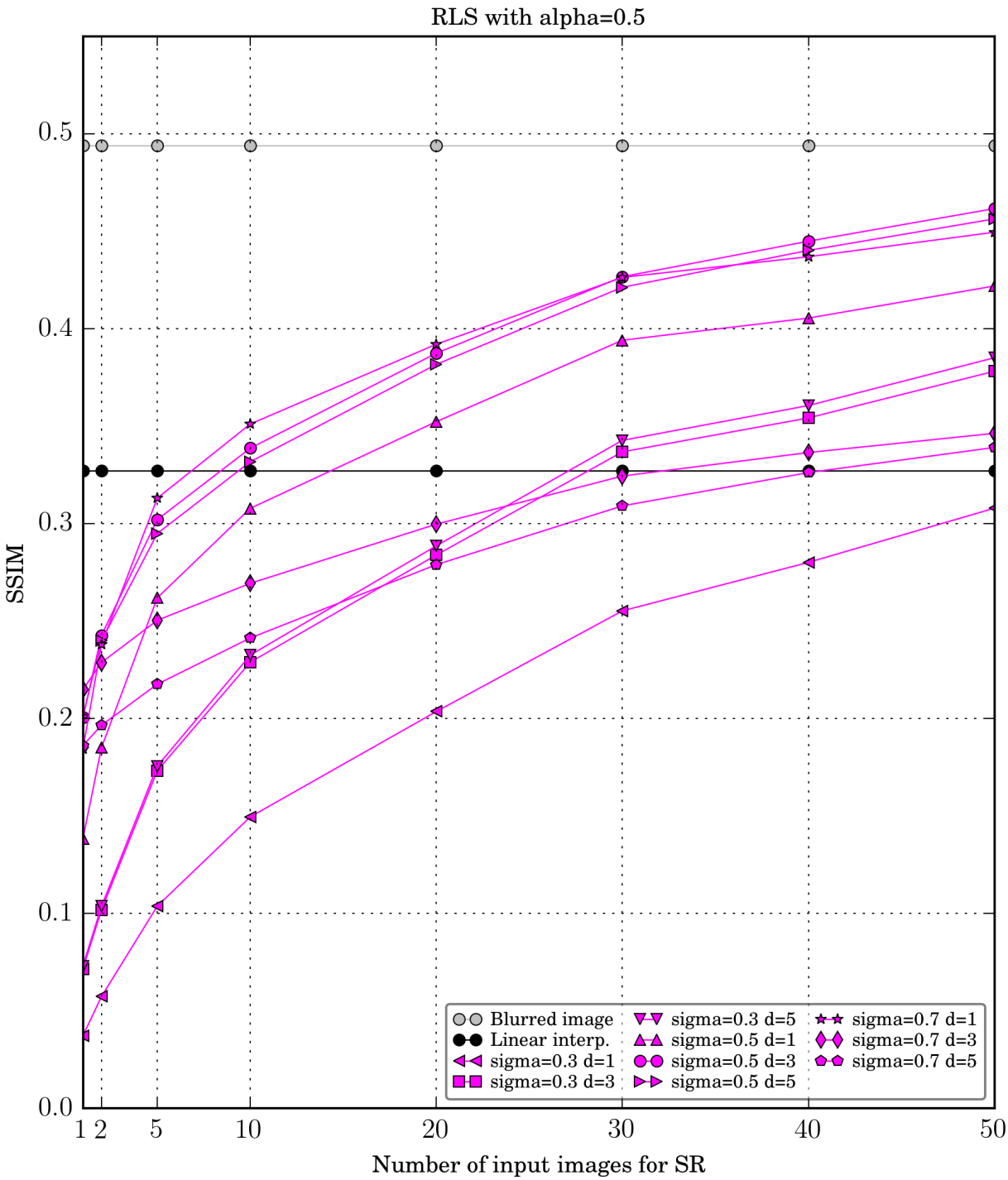


Figure A.8: SSIM values for *RLS* using $\alpha = 0.5$ and different values for σ and the kernel size. Each curve is the average for all the images from the pool in the subset $\mathscr{D}_1$.

which a given method x got the highest accuracies (i.e., the number of videos in which no other method y got more hits than the method x). We refer to such number as Acc_{BEST}.

Table A.1 confirms that PyrLK is also the most promising tracking method using such accuracy number.

The Acc_{BEST} values for the registration methods, in Table A.2, support the claim that "None" outperforms the other registration solutions, on average.

In Table A.3, we investigate the Acc_{BEST} values for super resolution.

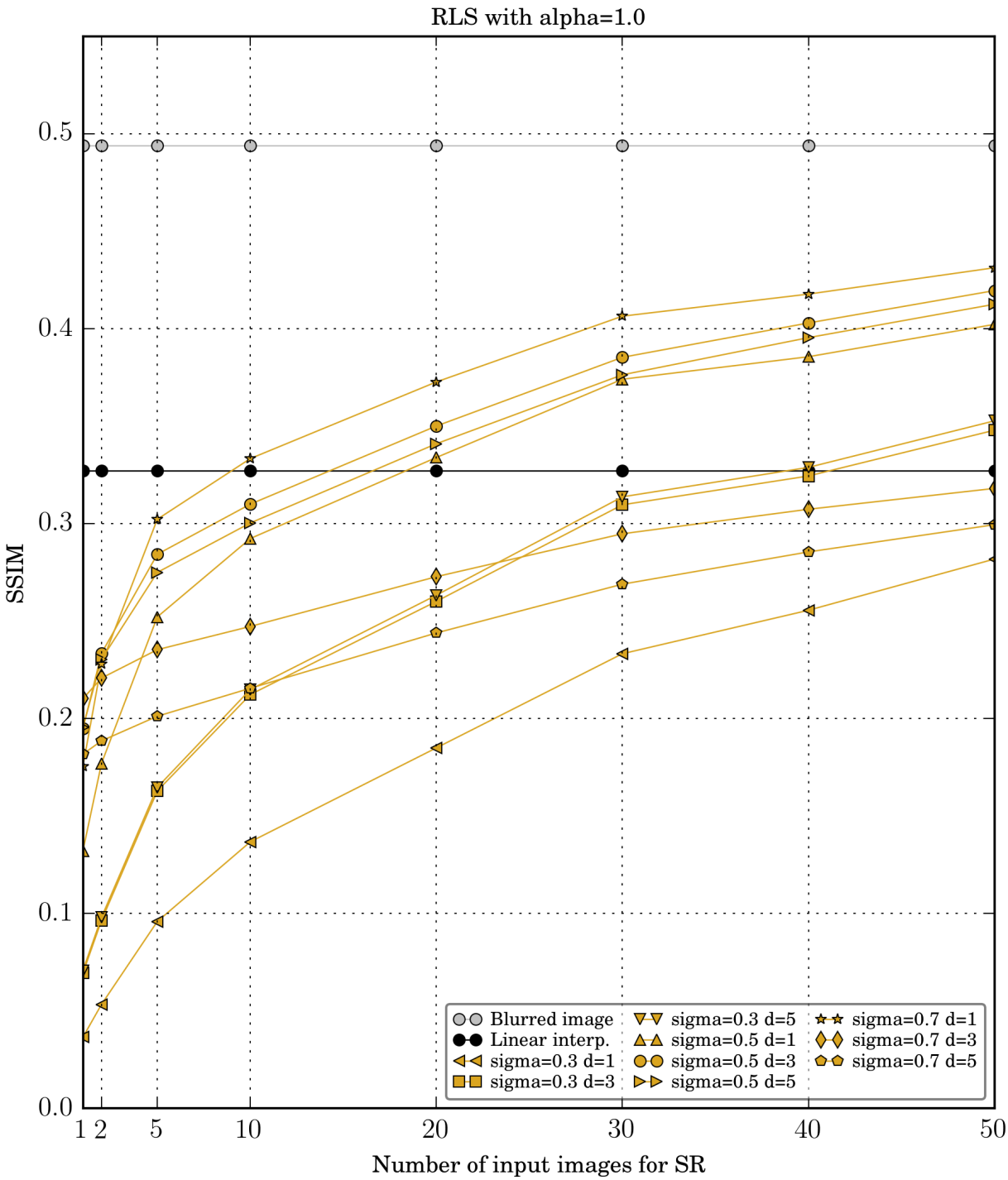


Figure A.9: SSIM values for RLS using $\alpha = 1.0$ and different values for σ and the kernel size.

We investigate now the accuracy for the binarization methods in Table A.4.

Finally, the Acc_{BEST} also shows Tesseract is more effective than OCRopus (see Table A.5).

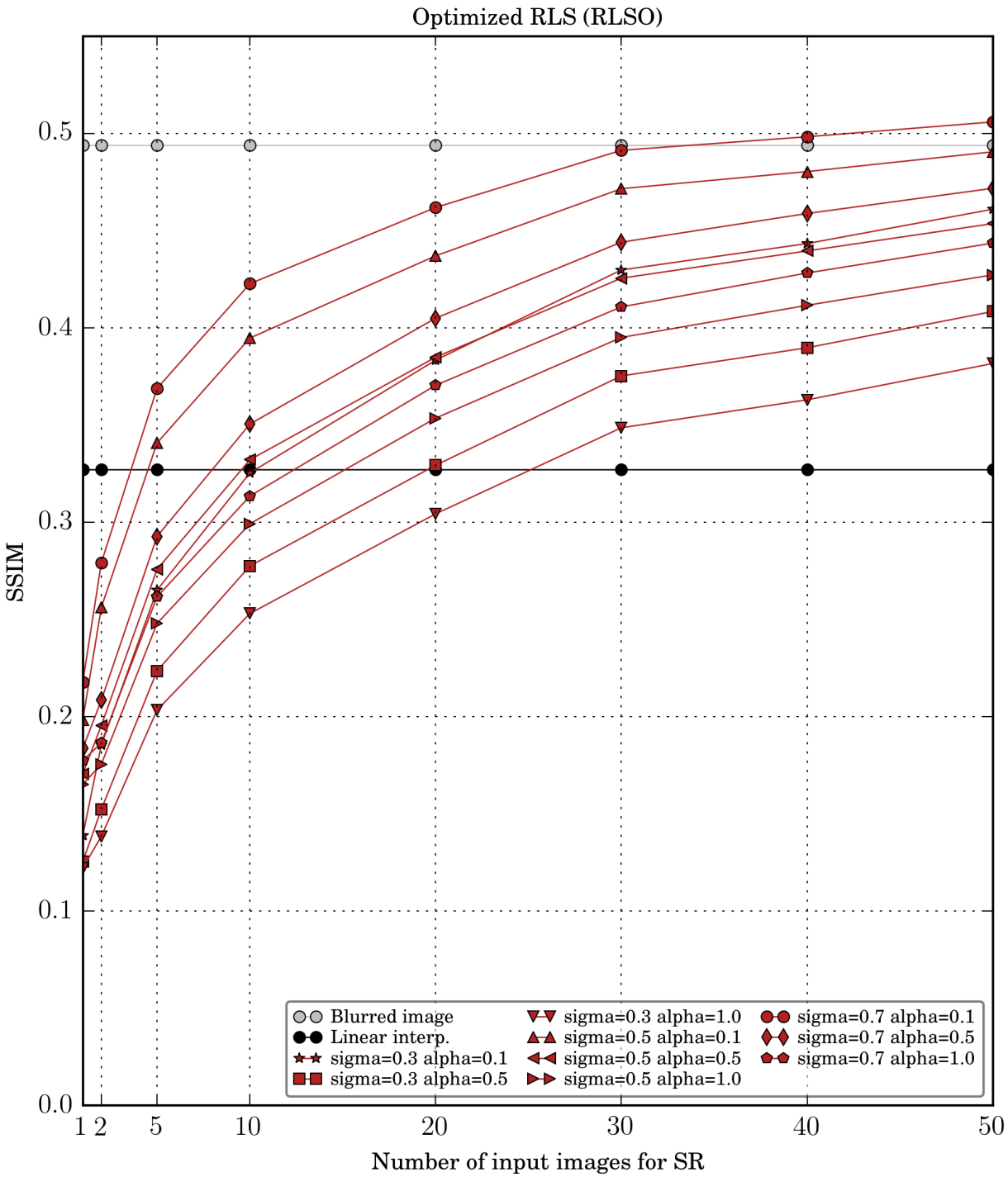


Figure A.10: SSIM values for *RLSO* using different values for α and σ.

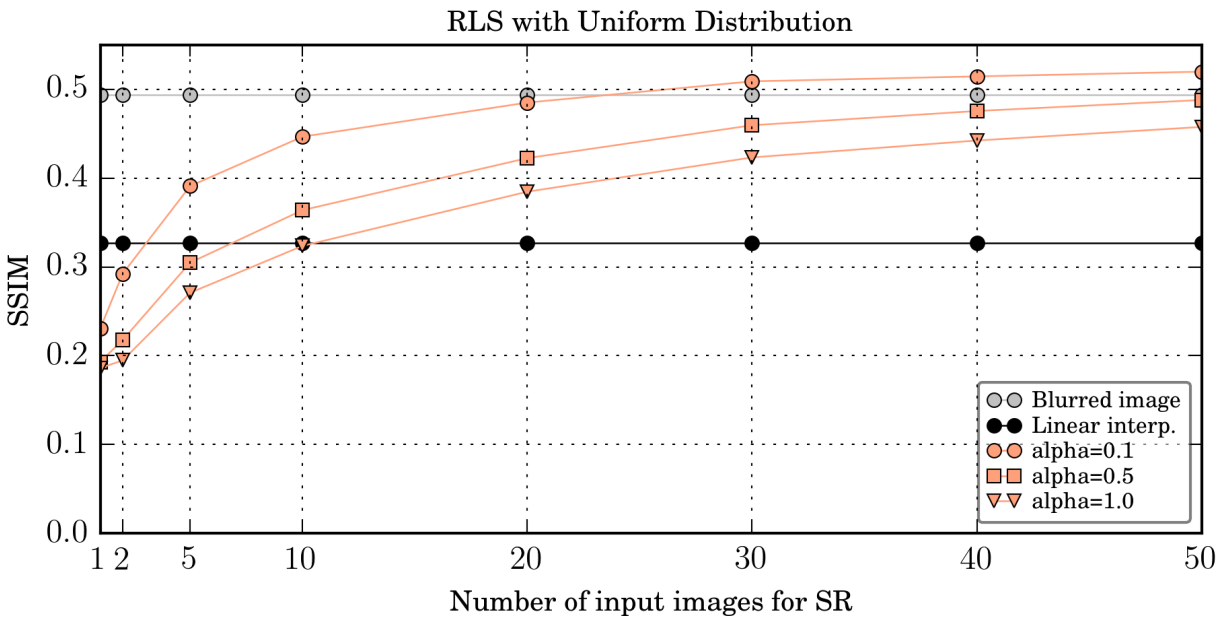


Figure A.11: SSIM values for *RLSU* using different values for α.

Tracking method	Acc_7	Acc_{BEST}
KLT	**174 (87.0%)**	**117 (58.5%)**
DENSE	167 (83.5%)	108 (54.0%)
SIFT	164 (82.0%)	110 (55.0%)
SURF	156 (78.0%)	109 (54.5%)
ORB	149 (74.5%)	99 (49.5%)

Table A.1: Accuracy of the framework's tracking step.

Registration method	Acc_7	Acc_{BEST}
LK	163 (81.5%)	113 (56.5%)
Dense	142 (71.0%)	93 (46.5%)
None	**183 (91.5%)**	**122 (61.0%)**

Table A.2: Accuracy of the framework's registration step.

Method	Acc_7	Acc_{BEST}	*Method*	Acc_7	Acc_{BEST}
GSR_1	134 (67.0%)	93 (46.5%)	ISR_1	131 (65.5%)	96 (48.0%)
GSR_2	140 (70.0%)	102 (51.0%)	ISR_2	134 (67.0%)	94 (47.0%)
GSR_3	142 (71.0%)	98 (49.0%)	*RLS*	**148 (74.0%)**	100 (50.0%)
GSR_4	**144 (72.0%)**	**103 (51.5%)**	*RLSO*	**146 (73.0%)**	**104 (52.0%)**
GSR_5	140 (70.0%)	97 (48.5%)	*RLSU*	**147 (73.5%)**	**105 (52.5%)**

Table A.3: Accuracy of the reconstruction step.

Binarization method	Acc_7	Acc_{BEST}
Otsu's binarization	**163 (81.5%)**	**112 (56.0%)**
Adaptive thresholding	150 (75.0%)	92 (46.0%)

Table A.4: Accuracy and runtime of the post-processing step.

OCR system	Acc_7	Acc_{BEST}
Tesseract	**177 (88.5%)**	**131 (65.5%)**
OCRopus	98 (49.0%)	72 (36.0%)

Table A.5: Accuracy and runtime of the recognition step.

Appendix B

Implementation details of the framework

In this appendix, we describe some implementation details for those who want to replicate the results of our framework. All the source codes are developed in Python 2.7, and most of the methods use Numpy 1.11 and OpenCV 3.1 [12]. The source-code is freely available on GitHub.

B.1 Tracking and registration steps

The PyrLK uses at most five pyramid levels[1]. The search terminates after a maximum of 10 iterations or when the search window moves by less than $\epsilon = 0.03$. The search window size is 11 (in both axis). However, if the framework cannot track at least 10 consecutive frames, the search window size is iteratively increased by 10, and the entire tracking step is performed again. The framework keeps increasing the window search size until it tracks the license plate through 10 consecutive frames, and stops trying after 10 failed attemps. To define if the tracking was successfully performed between two frames, we verify if the size of each bounding box side increases/decreases at most 20%. We find at most $1,000$ good features to track, using 0.01 as the quality level, 4 as the minimum possible Euclidean distance between the corners, and 19 as the size of an average block for computing a derivative covariation matrix over each pixel neighborhood.

PyrDense uses five pyramid levels, window search size from 11 to 101 (as in PyrLK), and a maximum of three iterations. The size of the pixel neighborhood used to find polynomial expansion in each pixel is 5, and the standard deviation of the Gaussian that is used to smooth derivatives used as a basis for the polynomial expansion is 1.2. In the registration step, LK and Dense differ from the tracking step only in the number of pyramid levels (1, instead of 5).

For SIFT, we do not limit the number of features to retain, and use 8 layers in each octave, 0.01 for the contrast threshold to filter out weak features in low-contrast regions, 10 for the threshold used to filter out edge-like features, and Gaussian $\sigma = 1.3$. In SURF,

[1]See OpenCV documentation [49] for more details about the parameters that we describe in this Section.

we use 10 as the threshold for the Hessian keypoint detector, 4 pyramid octaves, and 8 layers in each octave. Finally, we limit in $100,000$ the number of features to retain in ORB, the pyramid decimation ratio is 1.2, the number of pyramid levels is 8, the border size where the features are not detected (edge threshold) is 31, and the patch size used by the oriented BRIEF descriptor is 31. The keypoints found by SIFT, SURF and ORB are matched using the Flann's default parameters.

B.2 Reconstruction step

Both GSR_3 and GSR_5 rely upon a weighted average to calculate each pixel in the HR grid. GSR_3 combines always the three nearest neighbors. The weight of the nearest one is 60%, and the weights of others two nearest neighbors are 20%. If we have only two neighbors (e.g, if there are only two input frames to super-resolve), their weights are 70% and 30%. For GSR_5, we combine the neighbors inside a circular region of radius $r = 0.05$ pixels around the desired point p in the grid. The closer an LR pixel q_k is to p, the greater must be its weight. Hence, we define $w_k = r - d(q_k, p)$ as the weight between q_k and p, where r is the radius distance and $d(x, y)$ is the Euclidean distance between two points. For all the five variations of GSR, the HR grid size is (200×75) pixels (proportional to the Brazilian license-plate size).

B.3 Post-processing step

The Otsu's binarization is preceded by a Gaussian blur with kernel size $(3, 3)$, and the Adaptive thresholding is preceded by a Gaussian blur with kernel size $(7, 7)$. The size of a pixel neighborhood in Adaptive thresholding is 17, and the constant subtracted from the mean is 2.

B.4 Recognition step

Finally, we trained the OCR systems using the fonts *Mandatory* and *DIM Mittleschrift*. We created images with $17,576$ combinations of letters $(26 \times 26 \times 26)$ and $9,999$ combinations of digits. The training process is described in Sec. 5.7 and followed the documentations of Tesseract and OCRopus.