Some Problems On Temporally Consistent Video Editing And Object Recognition

Rida Sadek

Tesi Doctoral UPF / 2012

Supervised by
Prof. Vicent Caselles Costa
Departament de Tecnologies de la Informació i les Comunicacions

To my family

Acknowledgments

The single most influential source in shaping this thesis has been my advisor Vicent Caselles, to whom I am deeply greatful and forever indebted. He has been a great mentor who went far beyond his responsibilities as an advisor in encouraging and inspiring me to accomplish this work. Vicent is an exceptional person so as working with him. I feel priviliged to have been his student for the past five years.

Many thanks also to Coloma Ballester for all the help she provided me during these years and for reviewing this thesis.

I am also thankful to my former and current colleagues who have helped ensure an enjoyable and relaxed environment at the department: Sira Ferradans, Pablo Arias, Gabriele Facciolo, Enric Meinhardt, Constantinos Constantinopoulos, Edoardo Provenzi, Gloria Haro, Felipe Calderero,Vanel Lazcano, Alex Albore and Miquel Ramirez. Although special thanks must be noted to Pablo Arias and Gabriele Facciolo for all the interesting discussions on this research and their major help in the development of the work in this thesis. Thanks as well to Felipe Calderero who translated the abstract of this thesis into Spanish.

I also feel thankful to Veselin Stefanov Ovcharov, the owner of a pub close to the university (Tampa Café), for his outgoing personality and the unlimited supply of coffee and beer which held me through all the deadlines and stressful times over the past five years.

I would especially like to thank my friends Emil Keyder, Antonio Pedro Araújo and Tristan Whitmarsh for their sincerity, extreme transparency and for always maintaining an open heart. I am also very thankful to my friends Daniel Romero García and Anton Albajes who have played a big role in making my stay in Barcelona more enjoyable.

I would like to thank Mahmoud El Amin for being as faithful and true as a friend can be. He has always given me good and constructive advice during these years.

I thank my brothers and sister for the great encouragement they gave me whom without I wouldn't have endeavoured in this project. Dala, Karim and Mouhannad Sadek have given a lot so that I could do this PhD. Their moral and sometimes financial support is un-equated.

I would like to state that not only there is no friendship nor love like that of parents for their child, but also no support for a child can be more meaningful, constructive and efficient than the one of his/her parents. Nazha Fakhreddine and Sadek Sadek, I am, as I always have been, in awe of you.

My deepest gratitude goes to my wife Nardine, whom without her emotional support I would have quit a long time ago. She has cleverly managed to always shed a positive light on things (and being rightfully convincing at it). She has been to my side every time I needed her and I thank her for her patience, love and support.

Abstract

Video editing and object recognition are two significant fields in computer vision: the first has remarkably assisted digital production and post-production tasks of a digital video footage; the second is considered fundamental to image classification or image based search in large databases (e.g. the web). In this thesis, we address two problems, namely we present a novel formulation that tackles video editing tasks and we develop a mechanism that allows to generate more robust descriptors for objects in an image.

Concerning the first problem, this thesis proposes two variational models to perform temporally coherent video editing. These models are applied to change an object's (rigid or non-rigid) texture throughout a given video sequence. One model is based on propagating color information from a given frame (or between two given frames) along the motion trajectories of the video; while the other is based on propagating gradient domain information. The models we present in this thesis require minimal user intervention and they automatically accommodate for illumination changes in the scene.

Concerning the second problem, this thesis addresses the problem of affine invariance in object recognition. We introduce a way to generate geometric affine invariant quantities that are used in the construction of feature descriptors. We show that when these quantities are used they do indeed achieve a more robust recognition than the state of the art descriptors.

Resumen

La edición de vídeo y el reconocimiento de objetos son dos áreas fundamentales en el campo de la visión por computador: la primera es de gran utilidad en los procesos de producción y post-producción digital de vídeo; la segunda es esencial para la clasificación o búsqueda de imágenes en grandes bases de datos (por ejemplo, en la web). En esta tesis se acometen ambos problemas, en concreto, se presenta una nueva formulación que aborda las tareas de edición de vídeo y se desarrolla un mecanismo que permite generar descriptores más robustos para los objetos de la imagen.

Con respecto al primer problema, en esta tesis se proponen dos modelos variacionales para llevar a cabo la edición de vídeo de forma coherente en el tiempo. Estos modelos se aplican para cambiar la textura de un objeto (rígido o no) a lo largo de una secuencia de vídeo dada. Uno de los modelos está basado en la propagación de la información de color desde un determinado cuadro de la secuencia de vídeo (o entre dos cuadros dados) a lo largo de las trayectorias de movimiento del vídeo. El otro modelo está basado en la propagación de la información en el dominio del gradiente. Ambos modelos requieren una intervención mínima por parte del usuario y se ajustan de manera automática a los cambios de iluminación de la escena.

Con respecto al segundo problema, esta tesis aborda el problema de la invariancia afín en el reconocimiento de objetos. Se introduce un nuevo método para generar cantidades geométricas afines que se utilizan en la generación de descriptores de características. También se demuestra que el uso de dichas cantidades proporciona mayor robustez al reconocimiento que los descriptores existentes actualmente en el estado del arte.

Preface

Modern digital technology has paved the way to manipulate digital images in many different ways. Mainly, given a certain digital image as input, the set of operations on that image can be divided into three different categories: Image processing which outputs another modified image, image analysis which outputs a certain set of measurements, and image understanding which outputs some *high-level* descriptions.

A two-dimensional digital image is a color (or gray-level) photograph captured by a certain sensor (scanner, digital camera, ...). The captured digital image is formed by a set of pixels, where each pixel is a measurement of the amount of light captured by the sensor. All image manipulations will have to deal with these pixels and either modify, analyze or understand them. This manipulation happens either locally (by looking only at the pixel itself or within a certain small neighborhood around it) or globally (looking at the whole image).

On the other hand, a video sequence is nothing but a sequence of images (typically called frames) captured within an evenly spaced time interval. These frames are then shown, or presented, at a certain speed giving the feeling of a continuous movement. This adds a third dimension to the different manipulation categories. Now, the time factor (or time dimension) needs to be taken into account for all manipulation tasks. Therefore, the generalization of image processing tasks to video is not straight forward and needs to account for this time dimension. This opens the problem of finding the correspondence between consecutive frames. This is an active research area and the problem is typically addressed by estimating the motion between consecutive frames of a given video sequence.

This thesis is divided into two main parts: a first part on video editing (or processing) and a second on object recognition. In the video editing part, we address the problem of changing the texture of a given object (rigid or non-rigid) throughout a video sequence. This needs to be done with minimal effort from the user and the result obtained needs to account for illumination changes in the scene. Our approach asks the user to edit a frame in the video with the newly desired texture placed on the object. In practice, this can be very useful for post-production tasks. Consider a video of an advertisement of a certain product, where the products' packaging has been changed from the time when the advertisement has been shot. Instead of re-shooting the whole video or edit the video frame-by-frame, we ask the user to provide a couple of edited frames with the new packaging showing on the product and we automate the rest of the process. We do that by propagating, throughout the video, the newly edited information along the motion trajectories of the edited object while accommodating for global illumination changes. First, we generalize the widely used *brightness-constancy* assumption (BC) to account for global additive illumination changes.

We call this generalization the *global brightness change* assumption (GBC) and it is mainly achieved by working in the gradient domain. Then, based on these two assumptions, we propose two variational models that allow the propagation of information throughout a video sequence while accounting for illumination changes. A major challenge in these approaches is to deal with numerical problems due to the discretization of the *convective* derivative (*i.e.* the derivative along motion trajectories). These problems mainly manifest in the form of blurring artefacts. We propose a numerical scheme, the *de-blurring scheme for the convective derivative* (DSCD), that alleviates these artefacts and allows for the propagation of textures for a large number of frames while maintaining the sharpness of the original texture.

In order to tackle the above video editing problem, we assume the knowledge of the motion field of the video, typically approximated by the optical flow which computes the *apparent* motion. Recently, a lot of progress has been made in optical flow computation allowing for a better and more accurate approximation of the motion field. We benefit from this progress using and testing in the context of video editing different optical flows proposed in the literature.

In the second part of the thesis we address the construction of affine invariant image descriptors for object recognition, a problem that falls into the category of image understanding. Object recognition is a mature and a well studied topic to the point that it is being used these days in commercial and security applications (*e.g.* face recognition). The problem of identifying a given object in an image gets complicated when the *pose* of the object changes from one image to another. The set of transformations describing this change in pose is typically reduced to the set of *affine* transformations. In the second part of this thesis, we define and develop a set of quantities that are *affine-invariant*, *i.e.* their value do not change under affine transformations. These quantities are then used to describe objects locally by constructing *descriptors* following the model of SIFT. Through an extensive battery of tests on the standard benchmark, we show that the proposed quantities do exhibit more discriminative power and improve the performance of the state-of-the-art descriptors.

Contents

Part I

Video editing

This Part of the thesis presents two variational models for performing video editing tasks. The first model is based on the *brightness-constancy* assumption, where it is assumed that a particle does not change color with time. The second model is based on a generalization of the *brightness-constancy* assumption to take into account additive global brightness changes.

Throughout the presentation, we consider the application where a user desires to modify the texture of an object throughout a video, for example a sign on a wall. The purpose is to achieve the desired editing with minimal user intervention and for the resulting edited video to be both spatially and temporally consistent.

The first model we present is a 1^{st}-order variational model that balances two terms: a term guaranteeing temporal consistency while another guaranteeing spatial consistency. This model requires two runs with different parameters to achieve the desired result. The second model we present is a 2^{nd}-order variational model that overcomes some of the shortcomings of the previous model. It is based on a generalization of the *brightness-constancy* assumption that takes into account additive global brightness changes. This model requires a single run and its result is a video both spatially and temporally consistent. Our presentation will first discuss both models in the continuous setting. After that, we focus on the second model and present a detailed discussion in the discrete setting. By presenting the second model with all its details, the first model could then be directly derived.

Both models make use of the *convective derivative*, which is the derivative along the motion trajectories of the video, to propagate information from a given frame to all others. It has been pointed out in the literature [KCR05, SMTK06] that usual discretization schemes of the convective derivative operator present a considerable blur in the results after just a few frames. To alleviate this problem, we introduce a new and simple numerical scheme for the convective derivative, the *De-blurring Scheme for the Convective Derivative* (DSCD), which allows for propagation along motion trajectories for a large number of frames. We also introduce a simple occlusion detection algorithm that both models make use of in order to deal with textures of objects that get occluded/dis-occluded in the video.

1 Introduction

Digital editing of a captured video footage is becoming more and more common, mainly due to advances in computer graphics and computer vision techniques. Video editing tasks vary from basic operations such as trimming, cutting, splitting and resizing video segments to more elaborate ones such as editing objects' textures or, more generally, removing and adding objects in a video segment.

Throughout our coming discussion, we consider the following video editing problem. We are given one or two reference frames where an object's surface has been edited, and a certain editing domain. We also assume the knowledge of the motion in the editing domain. The objective is then to propagate, along the motion trajectories, the edited information in the reference frame throughout the editing domain. The resulting editing needs to be *spatially* and *temporally consistent*. This problem arises in other more complex video editing tasks, such as video inpainting and object touch-up as in [BZS+07]. In what follows we discuss the concepts of temporal and spatial consistency.

Temporal consistency refers to a smooth transition between successive frames, coherent with the motion in the sequence. Due to this constraint the editing of a video cannot be reduced to a series of independent image editing problems. The temporal inter-dependence imposed by the motion in the sequence has to be taken into account. Since we desire our propagation to be temporally consistent, we are lead to propagate the given information along the motion trajectories of the video.

Generally speaking, to model the motion of a given sequence, one can distinguish between parametric and non-parametric models.

Parametric models work under assumptions made on the geometry of the scene. For example, the background is usually assumed to be static and piecewise planar [ZXS05, JTWT06]. This model permits the computation of a closed form mapping between any pair of frames which can then be used to propagate information from one frame to another.

On the other hand, non-parametric models do not make assumptions on the geometry of the scene. These models usually estimate the motion in the sequence by the optical flow which is then used for propagation. There has been in recent years a lot of progress in optical flow computation. For example, nowadays, optical flow algorithms are able to deal with large displacements and allow for sharp discontinuities in the movement. This is the case of [SRB10, CP11, BM11, ARS11] to name a few. These methods still suffer from the "aperture" problem. In practice they typically incorporate a smoothness term which causes a filling-in effect leading to dense flow fields, even if the aperture problem is present. Since we do not wish to restrict the scene geometry, we are mostly interested in using non-parametric models for the motion in the sequence.

The problem of propagating information along the optical flow to ensure temporal consistency has been addressed in the literature. In video inpainting for example, some works inpaint first the optical flow and then propagate information along the inpainted flow to fill-in the inpainting domain. For an optical flow with subpixel accuracy, an interpolation scheme is required. As it has been observed by [KCR05], using the optical flow along with a bi-linear interpolation scheme to propagate the information presents a considerable blur in the results after just a few frames. In [KCR05], the problem is alleviated by using a higher order interpolation scheme. However, in [SMTK06], the authors noted that though higher order interpolation schemes behave better than the bi-linear one, the blur artefacts still persist. In this work we face the same problem and we address it by introducing a scheme that allows for propagation of information along the optical flow through a large number of frames maintaining the sharpness of the result.

A different but related approach is followed in [BGD+10]. They integrate the optical flow, computing a set of motion trajectories that roughly cover the editing domain. For the computation of these trajectories, the optical flow itself has to be interpolated at subpixel positions. Since the movement is generally smoother than the image, the accumulation of interpolation errors in the optical flow is not so noticeable. These trajectories are then used to propagate the known color information inside the editing domain.

Finally, let us mention the *unwrap mosaics* approach [RAKRF08], which is interesting because it avoids estimating frame-to-frame motion. Instead, the authors propose to compute a static unwrapped texture, a sequence of occlusion masks, and a sequence of transformations from the unwrapped texture to each one of the frames of the video. The editing is then performed directly on the unwrapped texture, and the changes are mapped back into the video sequence using the estimated transformations. The technique of the unwrap mosaics permits to handle a wide range of situations including zooms, geometric deformations and occlusions. The method relies however on a substantial algorithmic machinery including accurate video segmentation, keypoints tracking and non-linear optimization for computing the texture and mappings. Also, since the mosaics are fixed, the illumination changes must be managed in a post-processing step.

The above mentioned works deal with propagating color information assuming that color remains constant along trajectories. This assumption is often referred to in the literature as the *brightness-constancy assumption*. However; due to shadows, reflections and other illumination changes, the color may change along trajectories. As a consequence, the propagation of color might cause spatial inconsistencies between the editing domain and its surrounding.

Spatial consistency refers to a seamless integration of the editing with its spatial surrounding in each frame. In the image editing literature, spatial consistency is usually addressed using gradient-domain methods. These have been applied extensively for tasks such as seamless cloning and compositing [PGB03, Geo05], shadow removal [FHLD06], HDR compression [FLW02], image inpainting [AFCS11, KT07], texture synthesis [KEBK05], and matting [SJTS04] among

others. Essentially, gradient-domain image editing is based on the manipulation of the image gradients instead of its gray levels. The modified gradients are then integrated to recover the resulting image. Typically, this is achieved by solving a Poisson equation with suitable boundary conditions. This procedure prevents the appearance of seams at the boundaries of the edited region. Poisson image editing [PGB03] is one of such techniques; it formulates variationally the problem as

$$\min_u \int_{O\subset\Omega} \|\nabla u - g\|^2 dx; \quad \text{with} \quad u|_{\partial O} = u_0,$$

where $\Omega \subset \mathbb{R}^2$ is the image domain, $O \subset \Omega$ is the region to be edited, $g : O \to \mathbb{R}^2$ is the guidance vector field (e.g. gradient of the image to be composed), $u : O \to \mathbb{R}$ is the solution image whose gradient best approximates the field g, and $u_0 : \Omega \to \mathbb{R}$ is the original image which provides the boundary conditions needed for reconstructing the solution u. The solution of this problem is computed solving the Poisson equation with Dirichlet boundary conditions $u|_{\partial O} = u_0$ where ∂O denotes the boundary of O. For a more detailed introduction to gradient-domain methods, the reader is referred to [AR07]. Since we want our propagation to be spatially consistent, we are led to work in the gradient-domain.

Some authors have tackled the problem of extending gradient-domain image editing techniques to video. In [WXRA05], it has been proposed to apply Poisson editing [PGB03] directly to video by considering a video to be a three-dimensional volume and using the 3D-gradient to perform editing operations. Though this eliminates artefacts such as flickering, it does not take motion into account.

In [BZS+07], it has been noted that using the method in [WXRA05] leads to severe ghosting artefacts for videos with camera motion. For this reason, they are lead to use a 3D-gradient where the temporal derivative is in the direction of the motion. That work deals with several video editing tasks, from which the most related to our application is the "object touch-up". They proceed in two steps. First they propagate the color information using *structure from motion* techniques. The result is temporally consistent in the editing domain, but may have spatial seams. Now, to remedy this, a second step is performed. Using the spatial gradient of the propagated information, an energy functional is proposed with two terms: a term performing a Poisson image editing in each frame imposing spatial consistency, and a term filtering along motion trajectories to further ensure temporal consistency. These two terms are balanced by a positive parameter. The resulting video is spatially and temporally consistent. This work has been further elaborated into a full-framework in [BZCC10] for image and video filtering.

In [BZS+07, BZCC10], the model to impose temporal consistency is based on the brightness constancy assumption. This makes it hard for the system to handle fast illumination changes along time. Furthermore, the propagation in [BZS+07, BZCC10] needs to be divided in two steps. The first step is to obtain

a temporally consistent gradient field, which is then integrated in the second step to achieve spatial consistency.

We will first discuss different mathematical models for temporal consistency, and propose a generalization of the brightness-constancy assumption, the *global brightness change* assumption, to allow for global additive illumination changes. This is achieved by working in the gradient domain. We then discuss two energy functionals: a) one based on the brightness constancy assumption, and the other, b) based on the global brightness change assumption. These energies can be used to propagate information along motion trajectories.

The minimizers of (a) need to undergo a two step procedure similar to [BZS+07]. The difference is that the first step consisting of propagating colors is done using the convective derivative (*i.e.* the derivative along the direction of the motion). This is allowed by using a numerical scheme, the *de-blurring scheme for the convective derivative* (DSCD), which makes the propagation possible through a large number of frames without the blurring effects noted in [SMTK06, KCR05]. Then, a similar two term energy to the one in [BZCC10] is used to remove spatial seams in a temporally consistent manner.

The minimizers of (b), however, are temporally and spatially consistent, being able to handle sudden illumination changes. Although it is based on a model for temporal consistency that considers only global illumination changes, the variational formulation (b) allows some spatial variation on the illumination change.

As a use case, we consider the application where a user edits a frame by changing the texture of an object's surface and then we generate the edited information throughout the rest of a given editing domain. To handle cases where an occlusion followed by a dis-occlusion occurs, we require a frame to be edited further in time so that dis-occluded information could be recovered. For the sake of clarity, Figure 1.1 shows an example of an input and an output of one of the proposed models, specifically (b). Let us further note that the models we present do not require a precise tracking of the edited object.

Additionally, in this Part of the thesis, we discuss an interpolation method in order to produce a sequence of plausible intermediate frames between two input images. This can be applied to slow camera motion for smooth playback of lower frame rate video or to smooth view interpolation and animation of still images. The main feature of the method is the handling of occlusions using a time coherent video segmentation into spatio-temporal regions.

Let us now present the organization of the "Video Editing" Part of this thesis. First we present the context of our work in Chapter 2. We start by motivating the energies we will propose by discussing some mathematical models for temporal consistency in Section 2.1 and then in Section 2.2 we give an overview of the framework considered throughout this Part. In Chapter 3 we discuss, in the continuous and discrete settings, two models for spatio-temporal coherent video editing. Then, in Chapter 4 we introduce the *de-blurring scheme for the convective derivative* (DSCD) and in Chapter 5 we present a simple occlusion detection method. We then present some experimental results in Chapter 6. Finally, Chapter 7, presents an interpolation method that produces a sequence of

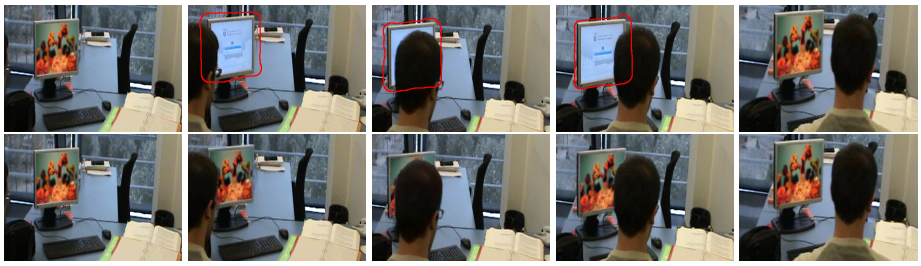


Figure 1.1: An example of an input and output of our model. The sequence has 20 frames in total. In the first row the user edits the first and last frames of the sequence. These are shown by the left most and right most images of the first row. In the remaining frames, we would like to propagate the edited information inside the area marked in red (the editing domain). In the second row, we show the output obtained using the proposed model.

plausible frames between two input images.

2 Context

In this Chapter we first discuss different models for temporal consistency, and then we introduce some useful definitions and descriptions of our framework.

2.1 Models for temporal consistency

In this Section we discuss some mathematical models for the temporal consistency of a video. This helps motivating the two energies we propose in Sections 3.1.1 and 3.1.2.

We consider a spatio-temporal domain $\Omega^T = \Omega \times [0, T]$, where $\Omega \subset \mathbb{R}^2$ is a rectangular domain, and $T > 0$. Let $u : \Omega^T \to \mathbb{R}$ be a given video in the continuous domain and let $v : \Omega^T \to \mathbb{R}^2$ be the motion field. The value of the motion field $v(x, t)$, where (x, t) is a point in Ω^T, represents the velocity of the projection of a particle in the 3D scene onto the image plane [Hor86]. The trajectory of the particle can be obtained by solving the following ordinary differential equation (ODE):

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = v\left(x(t), t\right), \tag{2.1}$$

where $t \in [0, T]$. For simplicity we assume in this Section that the functions we consider can be differentiated as many times as needed.

In what follows, we review the brightness-constancy assumption which is widely used in the literature to compute the optical flow. Then we discuss a generalization of this model to account for spatial global illumination changes. Henceforth we refer to this as the *Global Brightness Change* assumption (GBC). Finally we discuss the differences between this generalization and the gradient-constancy assumption, also used in the context of optical flow computation.

2.1.1 Brightness-constancy assumption

For a *Lambertian* object under uniform illumination, the brightness of an object's particle does not change in time. This implies that $u(x, t)$ is constant along trajectories, leading to the following brightness-constancy equation:

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}u(x(t), t) = \nabla_x u(x(t), t) \cdot v(x(t), t) + \frac{\partial u}{\partial t}(x(t), t), \tag{2.2}$$

where $\frac{\mathrm{d}}{\mathrm{d}t}u$ is the total derivative of u (*i.e.* the derivative along trajectories) and $\nabla_x u$ refers to the partial derivative of u with respect to x. Let us define the *convective derivative* as

$$\partial_v u(x, t) := \nabla_x u(x, t) \cdot v(x, t) + \frac{\partial u}{\partial t}(x, t). \tag{2.3}$$

The brightness constancy assumption can now be written as $\partial_v u(x,t) = 0$. This assumption has been used extensively for the computation of optical flow [WBBP06, BSL+11], and recently for video interpolation given an optical flow [KCR05, SMTK06, BGD+10]. In Section 3.1.1 we introduce a variational model for spatio-temporal coherent video editing assuming brightness constancy.

2.1.2 Global brightness change (GBC) assumption

Under illumination changes, the brightness-constancy assumption does not hold. In this Section, we generalize this assumption to account for spatially constant, additive illumination changes. In that case, if we follow the trajectories of two particles, the difference of their colors remains constant. In Section 3.1.2 we introduce a variational model for spatio-temporal coherent video editing based on this observation.

Let us consider two particles that at time t are in positions $x_0 \in \Omega$ and $y_0 \in \Omega$. We denote their trajectories by $\varphi(x_0, s)$ and $\varphi(y_0, s)$, with $s \in [0, T]$. Then for $k > 0$,

$$u(\varphi(y_0, t+k), t+k) - u(\varphi(x_0, t+k), t+k) \;=\; u(y_0, t) - u(x_0, t)\,. \quad (2.4)$$

This is represented by Figure 2.1(c). After rearranging terms, dividing by k, and taking $k \to 0$ we obtain $\partial_v u(y_0, t) = \partial_v u(x_0, t)$. Since this holds for all $x_0, y_0 \in \Omega$, we have that

$$\partial_v u(x,t) = g(t)\,. \quad (2.5)$$

This equation generalizes the brightness-constancy model taking into consideration global changes in illumination expressed by the illumination change rate $g(t)$.

Taking the spatial gradient on both sides of Eq. (2.5) we get the differential version

$$\nabla_x \partial_v u(x,t) = 0\,. \quad (2.6)$$

A Taylor expansion of (2.6) leads to

$$u(y_0 + kv(y_0,t), t+k) - u(x_0 + kv(x_0,t), t+k) \;\approx\; u(y_0,t) - u(x_0,t)\,, \quad (2.7)$$

which is an infinitesimal version of (2.4).

Note that, although we derived equation (2.6) under the assumption of global illumination changes, it is also reasonable under local illumination changes as long as they vary smoothly in the spatial domain.

Remark. The GBC can be regarded as a particular case of the *Generalized Dynamic Image Model* (GDIM) proposed by Negahdaripour [Neg98] in the context of optical flow computation. GDIM is a more general model for temporal consistency, accounting for additive and multiplicative illumination changes, along with their spatial variations. There, the assumption is that the convective derivative fulfills $\partial_v u(x,t) = m(x,t)u(x,t) + g(x,t)$, where m and g are referred to as

the *multiplier* and *offset* fields respectively. If we assume a spatially constant offset field (*i.e.* a global additive illumination change), and a zero multiplier field, the model reduces to the GBC model. In the energy we will propose that is based on the GBC model, the restriction of a global illumination change will be somewhat relaxed allowing some spatial variation on the illumination change rate.

2.1.3 Comparison with the gradient-constancy assumption

It is interesting to compare the GBC assumption with the related assumption where

$$u(\varphi(x_0, t+k)+h, t+k) - u(\varphi(x_0, t+k), t+k) \;=\; u(x_0+h, t) - u(x_0, t)\,.$$

Note that in this model, $y_0 = x_0 + h$ is mapped to the next frame using the mapping of x_0 and not its own. In this case, the underlying differential equation is

$$\partial_v \nabla_x u(x,t) = 0\,, \tag{2.8}$$

which is referred to in the literature as the gradient-constancy assumption [UGVT88, WBBP06, PBB+06]. Clearly, the gradient needs not to remain constant along trajectories, except for the particular case in which the motion field $v(\cdot, t)$ is constant (corresponding to a translational movement). For the purpose of optical flow computation, as discussed in [UGVT88], even when the movement is not translational the gradient-constancy assumption is a very good approximation, since the change in the gradient along trajectories between two consecutive frames is minor. However, in our application where we propagate information along a large number of frames, small changes in the gradient accumulate along a trajectory becoming significant. Figure 2.1 illustrates the three discussed models and shows their differences.

2.2 Framework

Let us now give a description of our framework. First, we will describe the video domains we consider which will lead us to derive two application settings. Then, we will give a decomposition of the boundary of the editing domain which will help in the understanding of the boundary conditions.

2.2.1 Domains and application setting

We consider a continuous scalar video $u_0 : \Omega^T \to \mathbb{R}$, and an editing domain $O \subset \Omega^T$ with a Lipschitz boundary [Ada75] (to simplify, we can consider that O has a smooth boundary). We denote temporal "slices" of O by $O_t = \{x \in \Omega : (x,t) \in O\}$. Similarly, temporal slices of Ω^T are denoted by $\Omega_t : t \in [0,T]$ representing the frames of the continuous video. An illustration of these domains can be seen in Figure 2.2.

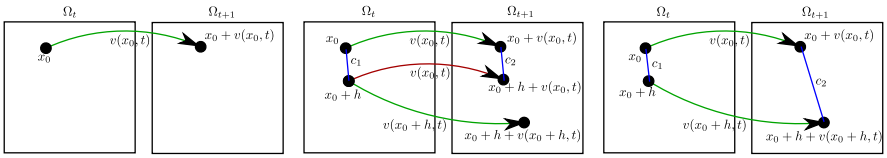


Figure 2.1: An illustration of the different models for temporal consistency. Ω_t and Ω_{t+1} refer to temporal "slices" of Ω at time t and $t+1$ respectively. (a) Shows the brightness constancy assumption. A point and its projection by the optical flow maintain the same color. (b) Illustrates the gradient constancy assumption. Here $c_1 = u(x_0+h,t) - u(x_0,t)$ and $c_2 = u(x_0+h+v(x_0,t),t+1) - u(x_0+v(x_0,t),t+1)$ and it is assumed that $c_2 - c_1 = 0$. (c) Depicts the global brightness change model. Here $c_2 = u(x_0+h+v(x_0+h,t),t+1) - u(x_0+v(x_0,t),t+1)$ and it is assumed that $c_2 - c_1 = 0$. This model differs from the gradient-constancy assumption by projecting x_0+h to the next frame using its own optical flow and not the one of x_0.

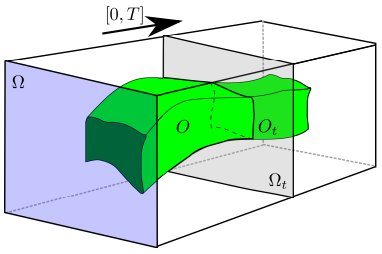


Figure 2.2: Illustration of an editing domain O inside of the video domain $\Omega \times [0,T]$. O_t and Ω_t are temporal slices at time t of O and $\Omega \times [0,T]$ respectively. To simplify the figure we do not show all complexities that a general editing domain may have (see Figure 2.3).

The models presented in Sections 3.1.1 and 3.1.2 of the next Chapter require the optical flow to be known. Therefore we only work with editing tasks that do not modify the optical flow of the original video. Such tasks consist typically in changing the texture or the appearance of an object's surface (or part of it) visible in the sequence. The editing domain O should contain the spatio-temporal volume described by the surface to be edited.

Let us now discuss the types of video editing applications we tackle using the models we will present. We consider two application settings.

In the first one, the user provides a first frame Ω_0, edited with some image editing tool (automatically or with some user intervention). Minimizing the energies that will be presented in Sections 3.1.1 and 3.1.2, with the first frame set as a Dirichlet boundary condition on u, propagates the edited information to the

rest of the sequence. We call this the *one-lid* setting where the *lid* refers to the first frame Ω_0, containing the information to be propagated. The other temporal end of the video at $t = T$ is left free. Note that one could edit the last frame Ω_T and set it as Dirichlet boundary condition, while leaving the first frame Ω_0 free. Henceforth, when we refer to the one-lid setting we always consider Ω_0 to be the lid.

For the second setting, both the first and last frames are edited and provided by the user. Both of them are set as Dirichlet boundary conditions. We refer to this as the *two-lid* setting. In this case, minimizing the energies that will be presented in Sections 3.1.1 and 3.1.2 yields an interpolation between both lids. If the editing on both lids is not consistent with the surface's motion in the sequence, the solution will present a temporal blending between the lids. The two-lid setting allows to treat cases in which the modified surface is occluded and then dis-occluded by another object in the sequence. For both the *one-lid* and the *two-lid* settings, we assume that any point of the spatio-temporal volume described by the edited surface is reachable by at least one trajectory originating from a lid (see Figure 2.3).

Let us remark that we do not require a precise tracking of the edited surface, as long as it is contained in the editing domain O. As a consequence, some parts of the editing domain may not belong to the edited surface. In these places, the original video should be restored. This is shown in Figure 2.3. The trajectory of the edited surface is shown in green. The gray regions of O are not intended to be modified.

2.2.2 A decomposition of the boundary of O

Let us denote by ∂O the boundary of O. Let ∂O_t denote the boundary of O_t, $t \in [0, T]$. Let us denote by $\nu^O = (\nu_x^O, \nu_t^O)$ the outer unit normal to ∂O (a vector in the unit sphere of $\mathbb{R}^3$) and let ν^{O_t} be the outer unit normal to ∂O_t (a vector in the unit circle of $\mathbb{R}^2$). Notice that the normal ν^O exists at any point if ∂O is smooth, and at almost any point with respect to the Hausdorff $\mathcal{H}^2$ (surface) measure on ∂O if we assume ∂O to be Lipschitz [AFP00, Ada75].

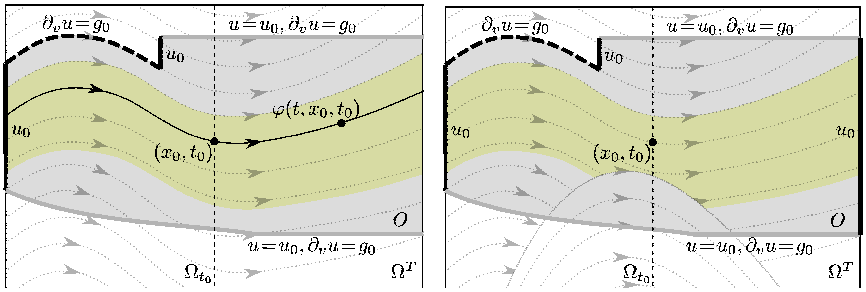
Let us introduce the following decomposition of the boundary of O. We consider the *lateral* boundary of O as the set

$$\partial O_{\text{lat}} := \{(x, t) \in \partial O : t \in (0, T)\}.$$

It corresponds to excluding the temporal ends of O, that is O_0 and O_T, from ∂O. The lateral boundary of O can be further classified into three parts. First, the *tangential* boundary, given by

$$\partial O_{\text{tang}} = \{(x, t) \in \partial O : \nu_t^O + v \cdot \nu_x^O = 0\}.$$

It corresponds to the segments of ∂O that are tangential to the motion trajectories. In the example of Figure 2.3(a) the tangential boundary is the dashed part of ∂O. Second, the *vertical* boundary ∂O_{vert} consists of the segments of the lateral



(a) *One-lid* problem. (b) *Two-lid* problem.

Figure 2.3: Domains and Dirichlet boundary conditions for a one-lid (a), and a two-lid problem (b) on a video with one spatial dimension. Trajectories of the edited object are marked in green (middle area). Note that (b) exhibits an occlusion and dis-occlusion of the edited object.

boundary parallel to Ω_0, *i.e.*

$$\partial O_{\text{vert}} := \{(x,t) \in \partial O_{\text{lat}} : |\nu^O(x,t) \cdot e_t| = 1\},$$

where $e_t = (0,0,1)$. It is formed by the vertical walls of the boundary except the initial and final slices O_0 and O_T, respectively. This is shown by the black vertical segments in Figure 2.3(a). Third, the *oblique* boundary, denoted by ∂O_{obli}, is the remaining non-tangential and non-vertical boundary (gray segments of ∂O in Figure 2.3(a)). That is

$$\partial O_{\text{obli}} := \partial O_{\text{lat}} \setminus (\partial O_{\text{tang}} \cup \partial O_{\text{vert}}).$$

Let us note that both O_{vert} and O_{obli} are non-tangential. However, they have to be distinguished because they admit different boundary conditions as we will see later.

3 Models for coherent video editing

In this Chapter we introduce two models for spatio-temporal coherent video editing. We first present the continuous setting and then the discrete setting. Sections 3.1.1 and 3.1.2 present, in the continuous setting, a 1^{st}-order model based on the brightness constancy assumption and a 2^{nd}-order model based on the global brightness change assumption, respectively. Then, Section 3.2 gives all necessary details for the discrete setting. The discussion in the discrete setting will mainly focus on the discretization of the 2^{nd}-order model. Having that, the 1^{st}-order model can be directly discretized.

3.1 The continuous setting

3.1.1 A 1^{st}-order model

In this Section we will propose an energy functional to perform video editing while enforcing the temporal consistency by assuming brightness constancy. Let us first discuss the case of forward propagation of a certain editing done on an object's surface at a given frame. Let $\Omega \subset \mathbb{R}^2$ be a rectangular open set representing the image domain and $[0, T]$ be the temporal domain.

First, let us recall the brightness constancy assumption presented in Section 2.1.1. A point moving in the real world describes a trajectory $s : [0, T] \to \Omega$ when seen in a video. The velocity field $v(s(t), t) = \frac{\mathrm{d}}{\mathrm{d}t} s(t)$ characterizes the motion of all the particles in the video. Then, given any particle and the associated trajectory s, the temporal consistency assuming brightness constancy means that along s the gray-level of $u(s(t), t)$ is constant, which can be stated as

$$\frac{\mathrm{d}}{\mathrm{d}t} u(s(t), t) = 0.$$

Applying chain's rule to this equation we obtain

$$\partial_v u(x,t) := \nabla_x u(x,t) \cdot v(x,t) + \frac{\partial}{\partial t} u(x,t) = 0, \quad \forall (x,t) \in \Omega \times [0,T]. \tag{3.1}$$

The temporal consistency can be stated in terms of the convective derivative:

$$\partial_v u(x,t) = 0, \quad \forall (x,t) \in \Omega \times [0,T].$$

Let $O \subset \Omega^T := \Omega \times [0, T]$ be the spatio-temporal domain where the editing is performed and let $O_t := \{x \in \Omega : (x, t) \in O\}$ be a temporal "slice" of O (Figure 2.2 illustrates these domains). We denote by ∂O and $\partial \Omega^T$ the boundaries of O and Ω^T respectively. We also assume that ∂O is a Lipschitz boundary and we denote by $\nu^O(x, t)$ the outer unit normal to ∂O at the point (x, t).

In order to propagate the information in time we solve

$$\min_u \int_O |\partial_v u(x,t)|^p \, dx \, dt \quad \text{with} \quad p \in \{1,2\}, \tag{3.2}$$

where we add Dirichlet boundary conditions

$$u(x,0) = u_0(x,0) \qquad x \in O_0 \tag{3.3}$$

$$u(x,t) = u_0(x,t) \qquad (x,t) \in \partial O_{lat} \setminus \partial\Omega^T, \tag{3.4}$$

with u_0 being the original video sequence.

This energy functional (3.2) propagates information in time taking into consideration the motion in the sequence. The resulting propagation is therefore temporally consistent. However, Eq. (3.2) does not adapt to changing illumination conditions and therefore does not integrate the propagated information with its spatial surrounding. The problem now becomes to correct the illumination in a video sequence where the information has been already propagated in a temporally consistent manner. That is, we assume that we already have the hole O filled-in with an image whose geometry is correct although its illumination is not consistent with its spatial surrounding. For a single frame, the problem can be solved by Poisson editing [CPT04a, PGB03, Geo05]. Let O_t be a hole in Ω at frame $t \in [0,T]$ (with a Lipschitz boundary) and let u_t be a known image in $\Omega \setminus O_t$. To copy an image f in O_t while adapting to the illumination of u_t, we describe the geometry of f by its gradient $\nabla_x f$, and we solve

$$\min_u \int_{O_t} \|\nabla_x u - \nabla_x f\|^p dx, \quad \text{with} \quad u|_{\partial O_t} = u_t|_{\partial O_t},$$

where $p \in \{1,2\}$ and $u_t|_{\partial O_t}$ denotes the trace of u_t taken from outside O_t. Then one redefines $u_t = u$ in O_t. The boundary condition ensures the good continuation with its surroundings, the geometry being inherited from f. This technique shows that the gradient performs a good job to define the object's geometry. The solution of this problem is computed by solving the Poisson equation $\Delta u = \Delta f$ in O_t (where Δ is the Laplacian operator) with Dirichlet boundary conditions $u|_{\partial O_t} = u_t|_{\partial O_t}$.

Now following the above discussions, we are led to combine the time consistency term with Poisson editing techniques and therefore to minimize the energy

$$E_P(u) = \int_O |\partial_v u(x,t)|^p + \beta |\nabla_x u(x,t) - g(x,t)|^p \, \mathrm{d}x \, \mathrm{d}t, \tag{3.5}$$

where $p \in \{1,2\}$, $\beta > 0$ and $g(x,t) = \nabla_x f(x,t)$ for any $(x,t) \in O$. Eq. (3.5) is solved with the Dirichlet boundary conditions (3.3), (3.4) and

$$u(x,t) = u_0(x,t) \qquad x \in \partial O_t, \ t \in (0,T), \ (x,t) \notin \partial\Omega^T. \tag{3.6}$$

The first term of energy (3.5) represents a regularization along the trajectories of the motion field in order to enforce the temporal consistency. The second term

in (3.5) is similar to 2D gradient-domain image editing models [CPT04a, PGB03, Geo05] and it is responsible for Poisson image editing at each frame.

It is important to discuss the parameter p in energy (3.5). The choice of $p \in \{1, 2\}$ leads to two models with different characteristics of the solution. Let us discuss these two cases.

Case $p = 2$: Here, energy (3.5) is quadratic and its solution is computed by solving the linear system

$$(\partial_v^* \partial_v \cdot + \beta \mathrm{div}_x \nabla_x \cdot)\, u = \beta \mathrm{div}_x\, g \quad \text{in } O, \tag{3.7}$$

with boundary conditions (3.3), (3.4) and (3.6) to which we add the Neumann boundary conditions

$$\partial_v u(x,t) = 0 \qquad\qquad (x,t) \in \partial O \cap \partial \Omega^T. \tag{3.8}$$

Notice that (3.8) holds in $O_0 \cup O_T$. We have denoted by div_x the spatial divergence and ∂_v^* the conjugate operator of ∂_v and is given by $\partial_v^* f = -\frac{\partial f}{\partial t} - \mathrm{div}_x(vf)$. Equation (3.7) is of Poisson-type, and can be solved using the conjugate gradient method.

The solution of (3.7) smoothly adapts to the boundary conditions of O. Moreover, any error due to inconsistencies between the boundary conditions and the potential field g is smoothly spread across the whole domain O.

Case $p = 1$: Here, energy (3.5) takes the form of a total variation minimization problem. To solve it, we perform an implicit gradient descent:

$$u^{j+1} = \arg\min_u E_1(u) + \frac{1}{2\lambda}\|u - u^j\|^2, \tag{3.9}$$

where λ is a positive number. Each iteration of the gradient descent entails the resolution of a convex problem similar to the total variation model for denoising [ROF92, Cha04]. The solution of (3.9) is computed by solving its dual problem [Cha04]. Defining the dual variables $\psi : O \to \mathbb{R}$ and $\xi : O \to \mathbb{R}^2$, we perform the fixed point iteration with time step $\tau \leq 1/8$:

$$\psi^{k+1} = \frac{\psi^k + \tau \partial_v [\partial_v^* \psi^k + \beta \mathrm{div}_x \xi^k + u^j/\lambda]}{1 + \tau |\partial_v [\partial_v^* \psi^k + \beta \mathrm{div}_x \xi^k + u^j/\lambda]|},$$
$$\xi^{k+1} = \frac{\xi^k + \tau \nabla_x [\partial_v^* \psi^k + \beta \mathrm{div}_x \xi^k + u^j/\lambda] + g/\lambda}{1 + \tau \|\nabla_x [\partial_v^* \psi^k + \beta \mathrm{div}_x \xi^k + u^j/\lambda] + g/\lambda\|},$$

and at convergence the solution is recovered as $u = u^j + \lambda(\partial_v^* \psi + \beta \mathrm{div}_x \xi)$. A complete derivation of the scheme is given in Appendix A.1.

This model allows discontinuities of u in O and at its boundary. Its solution attaches to the boundary conditions reducing the effects of the illumination changes, but, as opposed to the case $p = 2$, the transitions may not be smooth. While setting $p = 2$ favors smooth transitions, the model with $p = 1$ may produce sharp transitions which may be desirable in some circumstances. It also allows a better preservation of textures.

Discussion: The energy in (3.5) requires some further explanation. Notice that the model can be used for several tasks:

a) If we take $\beta = 0$ then we recover (3.2), the model for propagation. In practice, it is better to keep $\beta > 0$ although small in order to add some spatial regularization. In this case, we replace (3.8) by

$$\partial_v u(x,t) = 0 \quad (x,t) \in \partial O \cap \partial \Omega^T \setminus O_0,$$

and add Dirichlet boundary conditions in O_0. This deals with the *one-lid* setting. In the *two-lid* setting, we also replace Neumann by Dirichlet in O_T.

b) If we want to correct the illumination we take $\beta > 0$ large enough so that the second term is dominant. In that case, the first term acts as a time regularizer. The boundary conditions are those specified above for Eq. (3.5) with the Neumann boundary condition (3.8).

c) We may use the model advantageously in order to propagate in time and correct the illumination. For that, we first project in time using $\beta > 0$ although small as in a). Then we use that result to compute the guidance field $g(x,t)$ and solve (3.5) with $\beta > 0$ large enough as in b). This algorithm is a little bit cumbersome and better models are required to transport while correcting the illumination. A model for this will be introduced in Section 3.1.2. The model is based on the Global Brightness Change assumption (see Section 2.1.2) and the minimization of the energy

$$\widetilde{E}_P(u) = \int_O |\nabla_x \partial_v u(x,t)|^2 \, \mathrm{d}x \, \mathrm{d}t$$

under suitable boundary conditions to ensure propagation and illumination correction.

The main difficulties derived from these models are related to their numerical implementation, namely to the discretization of the convective derivative $\partial_v u(x,t)$. This issue will be addressed in Chapter 4.

3.1.2 A 2^{nd}-order model

In this Section, we present a continuous variational model for video editing imposing (2.6). We then derive the natural boundary conditions of the model.

Let us first recall our notation. Throughout this Section, we consider a continuous scalar video $u_0 : \Omega^T \to \mathbb{R}$, and an editing domain $O \subset \Omega^T$ with a Lipschitz boundary [Ada75] (to simplify, we can consider that O has a smooth boundary). We denote temporal "slices" of O by $O_t = \{x \in \Omega : (x,t) \in O\}$. Similarly, temporal slices of Ω^T are denoted by $\Omega_t : t \in [0,T]$ representing the frames of the continuous video. An illustration of these domains can be seen in Figure 2.2.

The proposed energy imposes the global brightness change model by penalizing departures from condition (2.6):

$$E(u) = \int_0^T \int_{O_t} \|\nabla_x \partial_v u(x,t)\|^2 \, \mathrm{d}x \, \mathrm{d}t \,. \tag{3.10}$$

While Eq. (2.6) implies a spatially constant illumination change, the variational model allows some spatial variation on $\partial_v u$. This is a useful feature in practical applications since it accounts for localized light sources, shadows and reflections, as long as they manifest at the boundary of the editing domain.

As we have mentioned in Section 2.2.1, the models we present require the optical flow to be known, therefore we only work with editing tasks that do not modify the optical flow of the original video. Energy (3.10) could also be applied to other video editing tasks, such as video inpainting. In that case, the optical flow needs to be inpainted first [KCR05, SMTK06].

Since this is a gradient-based energy, the choice of the boundary conditions plays an essential role in the application of the model.

3.1.2.1 Boundary conditions

Let us first recall the notation introduced in Section 2.2.2. Let us denote by ∂O the boundary of O. Let ∂O_t denote the boundary of O_t, $t \in [0,T]$. Let us denote by $\nu^O = (\nu_x^O, \nu_t^O)$ the outer unit normal to ∂O (a vector in the unit sphere of $\mathbb{R}^3$) and let ν^{O_t} be the outer unit normal to ∂O_t (a vector in the unit circle of $\mathbb{R}^2$). Notice that the normal ν^O exists at any point if ∂O is smooth, and at almost any point with respect to the Hausdorff $\mathcal{H}^2$ (surface) measure on ∂O if we assume ∂O to be Lipschitz [AFP00, Ada75].

When defining the boundary conditions we will use the decomposition of the boundary of O presented in Section 2.2.2.

Boundary conditions for the *one-lid* setting. We impose the following boundary conditions for $(x,t) \in (O_0 \times \{0\}) \cup \partial O_{\text{lat}}$ excluding the points (x,t) of the lateral boundary with $x \in \partial\Omega$ (*i.e.* no Dirichlet boundary conditions are given at the spatial boundary of the video domain Ω):

$$u(x,0) = u_0(x,0), \qquad x \in O_0, \tag{3.11}$$

$$u(x,t) = u_0(x,t), \qquad (x,t) \in \partial O_{\text{vert}}, \tag{3.12}$$

$$\partial_v u(x,t) = g_0(x,t), \qquad (x,t) \in \partial O_{\text{tang}} \setminus \partial\Omega^T, \tag{3.13}$$

$$\begin{matrix} u(x,t) = u_0(x,t) \\ \partial_v u(x,t) = g_0(x,t) \end{matrix}, \qquad (x,t) \in \partial O_{\text{obli}} \setminus \partial\Omega^T, \tag{3.14}$$

where the videos u_0 and g_0 are given.

Eqs. (3.11) and (3.12) correspond to the boundary conditions on vertical segments of the boundary. Note that they could be merged together into a single condition as

$$u(x,t) = u_0(x,0), \qquad t \in [0,T),\ |\nu^O(x,t) \cdot e_t| = 1.$$

We write them separately to highlight the Dirichlet boundary condition that fixes the first *lid* containing the editing provided by the user.

On the lateral boundary, there are boundary conditions on u and on its convective derivative $\partial_v u$. The latter specifies the rate of illumination change at ∂O. The illumination change rate in the interior of O corresponds to a smooth spatial interpolation of g_0 given at the boundary. In a typical editing application we set $g_0 = \partial_v u_0$. In this way, we impose in the editing domain the ambient illumination change of the original sequence.

The lateral boundary conditions on u apply when trajectories cross the lateral boundary, leaving or entering O. Note that if O corresponds to a precise tracking of the to be edited surface (the green domain in Figure 2.3(a)), trajectories will only cross the boundary at the temporal ends, O_0 and O_T. Therefore, all the lateral boundary will be tangential and only Eq. (3.13) will apply, specifying the illumination change rate. In this case, the only Dirichlet boundary conditions on u are the ones at the lid. The solution is then obtained by propagating (interpolating) the data at O_0 (O_0 and O_T) along trajectories, while accommodating for the illumination changes specified at the boundary.

In a more general case, we do not require O to be a precise tracking of the to be edited surface, as long as O contains the surface. In such cases, other trajectories, not belonging to the target surface, are included in O and may leave or enter the domain through non-tangential segments of the boundary (either at the temporal ends O_0, O_T or through the lateral boundary). The Dirichlet boundary conditions of Eqs. (3.12) and (3.14) apply in these cases. In the editing application we are discussing, the value of u_0 at those locations corresponds to the original video sequence.

Note that in the presented boundary conditions, none was specified on $\partial\Omega$ apart from the lids. These will be specified later when we derive the Euler-Lagrange equation in Section 3.1.2.2.

Remark. Notice that the vertical and oblique parts of the boundary admit different boundary conditions, although both of them are non-tangential. The rigorous derivation of possible boundary conditions in the different parts of the boundary will be given in Appendix A.2. But some intuition on the reasons can be gained by looking at the cases in which the minima are attained with zero energy. This happens only for some choices of the boundary conditions, referred to as *compatible boundary conditions*. In such cases, u comes as the solution of the partial differential equation (PDE)

$$\nabla \partial_v u(x,t) = 0.$$

To solve this PDE, we proceed in two steps. In the first step we integrate spatially the equation, to get $\partial_v u(x,t) = g(t)$. This integration is independent for each temporal slice. It requires the value of $g(t)$ to be specified at the spatial boundary ∂O_t of the slice O_t. Note that, modulo sets of zero surface measure, $\bigcup_{t\in[0,T]} \partial O_t = \partial O_{\text{tang}} \cup \partial O_{\text{obli}}$. This justifies why g has to be specified at $\partial O_{\text{tang}} \cup \partial O_{\text{obli}}$, but not on ∂O_{vert}. In the second step, we integrate $\partial_v u(x,t) = g(t)$ along optical

flow trajectories. To perform this integration we need the value of u whenever a trajectory crosses the editing domain, *i.e.* on the non-tangential components of ∂O. As an illustration, in Appendix A.6 we provide two analytical examples for non-zero energy solutions, showing how the solution can be computed from these boundary conditions.

Boundary conditions for the *two-lid* case. Here, in addition to the boundary conditions of the one-lid case, the frame O_T is specified as a Dirichlet boundary condition:

$$u(x,T) = u_0(x,T), \quad x \in O_T. \tag{3.15}$$

This setting is relevant to handle, for example, the case in which part of the editing domain is occluded and then dis-occluded (see example in Figure 2.3(b)).

Remark. As seen in the Appendix A.2, the boundary conditions (3.11), (3.12), (3.13) and (3.14) permit to prove that there is a unique minimizer of the energy. At an intuitive level, one can say that

1. Each trajectory needs to have at least one Dirichlet boundary condition on u.
2. At each temporal slice we need that a Dirichlet condition on $\partial_v u$ is specified on a set of positive length of the boundary of O_t.

Analogous conditions are needed in the discrete case.

As a consequence, the model cannot handle cases in which a point is dis-occluded and occluded again, since its corresponding trajectory will not reach a Dirichlet boundary condition on the boundary of O. To solve this, one could partition the problem into several smaller ones fulfilling conditions 1 and 2.

3.1.2.2 Euler-Lagrange equation

The Euler-Lagrange equation of energy (3.10) is given by the following fourth order PDE

$$\partial_v^* \mathrm{div}_x \nabla_x \partial_v u(x,t) = 0, \qquad (x,t) \in O, \tag{3.16}$$

where div_x is the spatial divergence adjoint to $-\nabla_x$ and ∂_v^* denotes the adjoint operator of the convective derivative, given by $\partial_v^* f = -\frac{\partial f}{\partial t} - \mathrm{div}_x(vf)$.

For the one-lid setting, in addition to the Dirichlet boundary condition discussed above, the following Neumann type boundary conditions apply on $\partial O \cap \partial \Omega^T$:

$$\mathrm{div}_x \nabla_x \partial_v u(x,t) = 0, \qquad t = T, \tag{3.17}$$

$$\nabla_x \partial_v u(x,t) \cdot \nu^{O_t}(x,t) = 0, \qquad (x,t) \in \partial O_{\text{tang}} \cap \partial \Omega^T, \tag{3.18}$$

$$\begin{array}{l} \nabla_x \partial_v u(x,t) \cdot \nu^{O_t}(x,t) = 0 \\ \mathrm{div}_x \nabla_x \partial_v u(x,t) = 0 \end{array}, \qquad (x,t) \in \partial O_{\text{obli}} \cap \partial \Omega^T. \tag{3.19}$$

In the two-lid setting, condition (3.17) does not apply, since O_T has a Dirichlet boundary condition on u.

We refer the reader to Appendix A.2 for a derivation of the Euler-Lagrange equation and its boundary conditions.

3.2 The discrete setting

In this Section we discretize the needed operators and apply them to derive the discrete energies as a discretization of (3.5) and (3.10). Since the dicretization of (3.10) presents a bigger challenge than the discretization of (3.5) and since the discretization of (3.5) could be directly derived from the discretization of (3.10), we will base our discussion in this Section with the purpose of discretizing (3.10). The operators needed to discretize (3.5) will appear naturally in the process. Therefore, we start by deriving a discrete version of operator $\nabla\partial_v u$.

3.2.1 Discretization of operator $\nabla\partial_v u$

For simplicity we consider now that $\Omega \subset \mathbb{R}$ (*i.e.* u is a one dimensional video), the resulting discretization can be easily extended to higher spatial dimensions (see Section 3.2.3). We consider a discrete video obtained by regularly sampling the continuous one with a spatial step h and a temporal step k. Let us approximate operator $\nabla\partial_v u$ (presented in Eq. (2.6)) at (x_0, t_0).

Using a forward difference scheme for the spatial derivative, we have

$$\frac{\partial}{\partial x}\partial_v u(x_0, t_0) \approx \frac{1}{h}\left[\partial_v u(x_0 + h, t_0) - \partial_v u(x_0, t_0)\right].$$

The convective derivatives can be approximated with a forward difference scheme as well:

$$\partial_v u(x_0 + h, t_0) \approx \frac{1}{k}[u(\varphi(x_0 + h, t_0 + k), t_0 + k) - u(x_0 + h, t_0)],$$

where in the last term we used that $\varphi(x, t_0) = x$. Similarly we have that

$$\partial_v u(x_0, t_0) \approx \frac{1}{k}[u(\varphi(x_0, t_0 + k), t_0 + k) - u(x_0, t_0)].$$

The value of $\varphi(x, t + k)$ can be approximated as follows:

$$\varphi(x, t_0 + k) \approx x + kv(x, t_0)\,.$$

For subpixel motions, $x + kv(x, t_0)$ will fall outside of the sampling grid, and $u(x + kv(x, t_0), t_0 + k)$ has to be interpolated from the available samples. We will denote by $\hat{u}(x + kv(x, t_0), t_0 + k)$ the interpolated value. When using bilinear interpolation, this results in an *upwind scheme with an adaptive stencil* for the convective derivative, as the one in [ZMQ98].

Putting everything together, we have the following operator

$$\begin{aligned}\frac{\partial}{\partial x}\partial_v u(x_0, t_0) \approx \\ \frac{1}{kh}\left[\hat{u}(x_0 + h + kv(x_0 + h, t_0), t_0 + k) - u(x_0 + h, t_0)\right] - \\ \frac{1}{kh}\left[\hat{u}(x_0 + kv(x_0, t_0), t_0 + k) - u(x_0, t_0)\right]. \quad (3.20)\end{aligned}$$

Figure 3.1 illustrates the discrete operator (3.20).

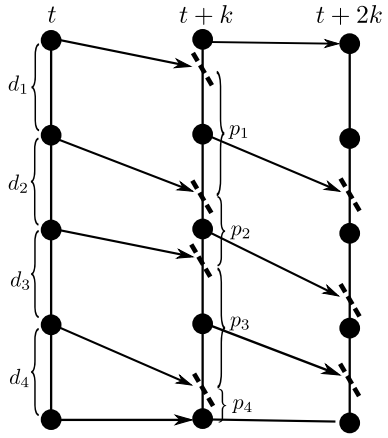


Figure 3.1: Illustration of the proposed discrete operator ($\nabla_x \partial_v$). The operator computes the difference between two points on a grid and their projection onto the next frame by the optical flow. In the figure, the operator computes $p_1 - d_1$, $p_2 - d_2$, etc... Our energy is based on this operator, imposing the squared L_2 norm of these differences to be constant along the video.

3.2.2 The discrete energies

In this Section we consider a video with two spatial dimensions and we propose our variational models in the discrete setting. Let us first define the following notation. We now consider u to be a scalar discrete video defined as a function $u : \Omega^T \to \mathbb{R}$. Here $\Omega^T = \Omega \times \{0, 1, \cdots, T\}$ is the discrete spatio-temporal domain, and $\Omega \subset \mathbb{Z}^2$ is a rectangular discrete domain (the spatial domain of each frame). Let us also denote by $O \subset \Omega^T$ the editing domain. Notice that we are using the same notations both for continuous and discrete domains. Each case will be clear from the context.

For a discrete video we use the optical flow computed on the original video u_0, as a computable approximation of the motion field. The *forward optical flow* $v^f : \Omega \times \{0, 1, \cdots, T\} \to \mathbb{R}^2$ establishes a correspondence between (x, t) and $(x + v^f(x, t), t + k)$. Similarly, we define the *backward optical flow* as the vector field $v^b : \Omega \times \{0, 1, \cdots, T\} \to \mathbb{R}^2$. At frame t, $v^b(\cdot, t)$ establishes a correspondence with the frame at time $t - k$.

The discrete form of energy (3.5) reads

$$E_{p,Occ}(u) = \sum_{(x,t)\in\bar{O}} \|Occ(x,t)\partial_v u(x,t)\|^p + \beta\|\nabla_x^+ u(x,t) - g(x,t)\|^p, \qquad (3.21)$$

where ∂_v is the convective derivative, $Occ : \Omega^T \to \{0, 1\}$ is an occlusion mask, ∇_x^+ is a forward difference spatial gradient and $p \in \{1, 2\}$. The domain $\bar{O} \subseteq \Omega^T$ is defined as the set of video pixels whose ∇_x^+ or ∂_v stencils intersects O.

Now the discrete form of energy (3.10) reads

$$E_\kappa(u) = \sum_{(x,t)\in\widetilde{O}} \|\kappa(x,t)\nabla_x \partial_v u(x,t)\|^2, \tag{3.22}$$

where ∇_x is a discrete gradient that is different from ∇_x^+ since it operates on convective derivatives and not directly on u. This difference will be clarified in Section 3.2.3, where we define the ∇_x operator along with the ∂_v operator. $\kappa : \Omega^T \to \{0,1\}^{2\times 2}$ is an *occlusion tensor* which will be defined along with *Occ* in Section 3.2.3.3. The domain $\widetilde{O} \subseteq \Omega^T$ is defined as the set of video pixels whose $\nabla_x \partial_v$ stencil intersects O. This domain, together with the operators presented next, implement the needed boundary conditions specified in Section 3.1.2.1.

Remark. We would like to stress the fact that in this Section we will base all our discussion with the purpose of providing a discretization for (3.10). The discretization for (3.5) can be directly derived from the discretization of (3.10).

3.2.3 Definition of the operators

In what follows we define the spatial gradient and the convective derivative operators as a generalization to two spatial dimensions of the discretizations presented in Section 3.2.1, considering $k = h = 1$.

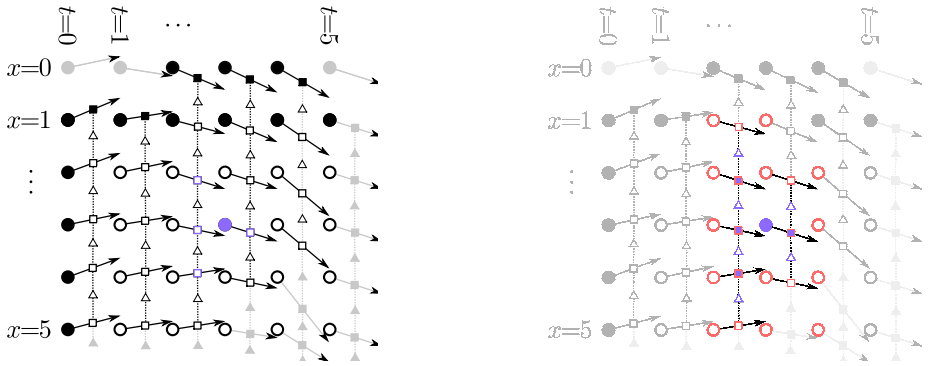
We denote by $x = (x_1, x_2)$ the spatial components of a point in Ω, and by $v^f(x,t) = (v_1^f(x,t), v_2^f(x,t))$ the components of the field $v^f(x,t)$. We denote $v^{f,I}(x,t) \in \mathbb{Z}^2$ and $v^{f,F}(x,t) \in [0,1)^2$ as the integer and fractional parts of $v^f(x,t)$, *i.e.* $v_i^f(x,t) = v_i^{f,I}(x,t) + v_i^{f,F}(x,t)$ for $i = 1,2$.

Our definition of the operators will integrate the different boundary conditions discussed in Section 3.1.2.1. We will define the operators over Ω^T, and by doing so, the Dirichlet boundary conditions on ∂O are straightforwardly implemented by assuming that $u = u_0$ in $\widetilde{O} \setminus O$. That is, if some of the values needed by the operators fall in $\widetilde{O} \setminus O$, the values of u_0 are used. The Neumann type boundary conditions on $\partial\Omega$ will be incorporated in the definitions of the operators.

Throughout this Section we will use a running example to illustrate the definitions of the operators. Figure 3.2 shows a discrete sequence with one spatial dimension: $\Omega^T = \{0,1,\ldots,5\} \times \{0,1,\ldots,5\}$. The circles represent the pixels in the video. Pixels in the editing domain O are depicted in white, whereas the black pixels correspond to $\widetilde{O} \setminus O$. The rest of the pixels of Ω that are not involved in the energy are shown as gray circles. There is a forward optical flow $v^f(x,t)$ at each pixel, represented by the vector $(v^f(x,t), 1) \in \mathbb{R}^2$.

3.2.3.1 Convective derivative operator

We discretize the convective derivative using an *upwind scheme with an adaptive stencil* [ZMQ98]. Let us introduce some useful notation before presenting the definition.



(a) Domains of the operators (b) Stencil of Euler-Lagrange Eq.

Figure 3.2: Graphical representation of the discrete operators for a 1D video sequence. The circles represent pixels of the video. Each arrow originating from a circle at (x,t) represents the forward optical flow $v^f(x,t)$. The square attached to the optical flow $v^f(x,t)$ represents the forward convective derivative $\partial_v^f u(x,t)$, computed as the difference between the values at both ends of the arrow, that is $\hat{u}(x+v^f(x,t),t+1)-u(x,t)$. The spatial gradient of the convective derivative at (x,t), $\nabla^f\partial_v^f u(x,t)$, is represented by the triangle connecting the squares that represent $\partial_v^f u(x+1,t)$ and $\partial_v^f u(x,t)$. See text for more details. Note that the domains of $\partial_v^f u$ and $\nabla^f\partial_v^f u$ are the same as the video domain, namely $\Omega\times\{0,1,\dots,T\}$. Let us stress that the squares and triangles are just a graphical representation of the operators. The fact that they are drawn at subpixel positions does not imply that the operators are defined on a subpixel grid.

For each video pixel (x,t) we define its forward interpolation neighborhood $N^f(x,t)\subset\Omega_{t+1}$ as

$$\begin{aligned}N^f(x,t)=\{&(x_1+v_1^{f,I},x_2+v_2^{f,I}),\\&(x_1+v_1^{f,I}+1,x_2+v_2^{f,I}),\\&(x_1+v_1^{f,I},x_2+v_2^{f,I}+1),\\&(x_1+v_1^{f,I}+1,x_2+v_2^{f,I}+1)\}.\end{aligned}$$

$N^f(x,t)$ contains the grid positions surrounding $(x+v^f(x,t),t+1)$. Analogously, $N^b(x,t)$ contains those surrounding $(x+v^b(x,t),t-1)$. The values at these pixels are used to bi-linearly interpolate the subpixel values at $(x\pm v^{f/b}(x,t),t\pm1)$.

We will also define the set $S_t^f=\{x\in\Omega_t : N^f(x,t)\subset\Omega_{t+1}\}$. Pixels $x\in S_t^f$ have their interpolation neighborhood inside the video domain.

We define the discrete *forward* convective derivative using the forward opti-

cal flow as follows

$$\partial_v^f u(x,t) = \begin{cases} \hat{u}\left(x + v^f(x,t), t+1\right) - u(x,t), & x \in S_t^f, \\ 0, & x \notin S_t^f, \end{cases} \tag{3.23}$$

where $\hat{u}(x + v^f(x,t), t+1)$ is the bilinear interpolation of $x + v^f(x,t)$ at $u(\,\cdot\,, t+1)$. We will refer to equation (3.23) as the v^f*-scheme*. Let us also define the v^b*-scheme* using the backward optical flow:

$$\partial_v^b u(x,t) = \begin{cases} u(x,t) - \hat{u}\left(x + v^b(x,t), t-1\right), & x \in S_t^b, \\ 0, & x \notin S_t^b, \end{cases} \tag{3.24}$$

Note that when the interpolation neighborhood is not completely inside Ω, the convective derivative is set to zero. This amounts to a Neumann boundary condition on $\partial\Omega$, and as will be discussed in Section 3.2.3.4, it causes a boundary condition of the type $\mathrm{div}_x \nabla_x \partial_v u(x,t) = 0$ on the Euler-Lagrange equation (at certain parts of $\partial\Omega$).

In order to compute the bi-linear interpolation $\hat{u}(x + v^f(x,t), t+1)$, we need to consider stencil points in $N^f(x,t)$ as follows:

$$\hat{u}(x + v^f(x,t), t+1) = [u_{11}(1 - v_1^F) + u_{12} v_1^F](1 - v_2^F) + [u_{21}(1 - v_1^F) + u_{22} v_1^F] v_2^F, \tag{3.25}$$

where

$$\begin{aligned} u_{11} &= u(x_1 + v_1^I, x_2 + v_2^I, t+1), \\ u_{12} &= u(x_1 + v_1^I + 1, x_2 + v_2^I, t+1), \\ u_{21} &= u(x_1 + v_1^I, x_2 + v_2^I + 1, t+1), \\ u_{22} &= u(x_1 + v_1^I + 1, x_2 + v_2^I + 1, t+1). \end{aligned}$$

An analogous expression holds for the backward optical flow. This results in a potentially different stencil for each x. When used for simulating the advection equation, this adaptive scheme allows to achieve stability with time steps beyond the one prescribed by the Courant-Friedrichs-Lewy condition [ZMQ98].

In the example of the Figure 3.2, each convective derivative $\partial_v^f u(x,t)$ is represented by a square node attached to the optical flow vector $(v^f(x,t), 1)$. It computes the difference between the values of u at both ends of the arrow, the value at the tip being given by the bilinear interpolation (linear in this one-dimensional example). The black squares represent known convective derivatives (their stencil is contained in $\widetilde{O} \setminus O$). These act as Dirichlet boundary condition setting the value of $\partial_v u$. Gray squares represent the convective derivatives of pixels $x \notin S_t^f$ which are set to zero in the definition.

Adjoint of the convective derivative. To define the adjoint of the convective derivative operator, let us first introduce an equivalent expression for the bilinear interpolation (3.25):

$$\hat{u}(x + v^f(x,t), t+1) = \sum_{y \in N^f(x,t)} w^f(x,y) u(y, t+1),$$

where $w^f(x,y)$ are the bilinear weights for $x + v^f(x,t)$.

The adjoint of the forward convective derivative is then given by

$$\partial_v^{f\,*} g(x,t) \;=\; \chi_{S_t^f}(x) g(x,t) - \sum_{y : x \in N^f(y,t-1)} \chi_{S_{t-1}^f}(y) w^f(x,y) g(y, t-1)\,, \quad (3.26)$$

where for any set $A \subset \Omega$, $\chi_A(x)$ is the indicator function of A, returning the value 1 if $x \in A$, and 0 otherwise.

Using the graphical representation of the operators in Figure 3.2, the convective derivative operator can be thought of as acting on a video (represented by the round nodes) and returning a function represented by the square nodes. Conversely, its adjoint operator $\partial_v^{f\,*}$ acts on functions g represented by the square nodes, and returns a function on the round nodes. Its value at a node (x,t) can be thought of as the net outgoing flow through the node. The outgoing flow is given by $g(x,t)$, whereas the incoming flow is given by the summation in (3.26), computed taking into account the values of g at the previous frame, on the pixels whose convective derivative stencil includes (x,t). This is illustrated in Figure 3.2(a): the blue squares depict the values of g needed to compute $\partial_v^{f\,*} g(3,3)$, the round node in blue.

3.2.3.2 Spatial gradient operator

Since the gradient operates on convective derivatives, we will define two gradient operators, $\nabla_x{}^f, \nabla_x{}^b : \mathbb{R}^{\Omega^T} \to \mathbb{R}^{2 \times \Omega^T}$ in correspondence with the v^f and v^b-schemes for the convective derivative (recall that $\Omega^T = \Omega \times \{0, 1, \ldots, T\}$). Both gradients are implemented using a forward difference scheme (spatially).

Let us consider first the gradient for the v^f-scheme. Some care must be taken with its definition. The spatial partial derivative in the direction $e_1 = (1,0)$ of the convective derivative $\partial_v^f u$ is given at a point (x,t) by $[\nabla_x{}^f \partial_v^f u(x,t)]_1 = \partial_v^f u(x + e_1, t) - \partial_v^f u(x,t)$. Thus, two convective derivatives are needed, $\partial_v^f u(x,t)$ and $\partial_v^f u(x + e_1, t)$. Therefore we will require that both x and $x + e_1$ lie in S_t^f. This motivates the definition of the following sets

$$\widetilde{S}_{e_1,t}^f = \{x \in S_t^f : x + e_1 \in S_t^f\}, \quad (3.27)$$

$$\widetilde{S}_{e_2,t}^f = \{x \in S_t^f : x + e_2 \in S_t^f\}, \quad (3.28)$$

where $e_2 = (0,1)$.

For an arbitrary video $q : \Omega \times \{0, 1, \ldots, T\} \to \mathbb{R}$, the ith component of its spatial gradient $\nabla_x{}^f q$ is then defined at a point $(x, t) \in \Omega \times \{0, 1, \ldots, T\}$ as

$$[\nabla_x{}^f q(x,t)]_i = \begin{cases} q(x + e_i, t) - q(x,t), & x \in \widetilde{S}^f_{e_i,t}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.29}$$

for $i = 1, 2$. This definition of the gradient implements a boundary condition of Neumann type at the spatial boundary of S^f_t.

The adjoint operator is given by the negative backward spatial divergence. For a vector-valued video $g \in \mathbb{R}^{2 \times \Omega^T}$, we will define the backward spatial divergence by

$$\begin{aligned} \operatorname{div}^f_x g(x,t) = {} & \chi_{\widetilde{S}^f_{e_1,t}}(x) g(x,t)_1 \\ & - \chi_{\widetilde{S}^f_{e_1,t}}(x - e_1) g(x - e_1, t)_1 \\ & + \chi_{\widetilde{S}^f_{e_2,t}}(x) g(x,t)_2 \\ & - \chi_{\widetilde{S}^f_{e_2,t}}(x - e_2) g(x - e_2, t)_2. \end{aligned}$$

Let us recall that for any set $A \subset \Omega$, $\chi_A(x)$ is the indicator function of A, returning the value 1 if $x \in A$, and 0 otherwise.

Analogously, we define a spatial gradient operator for the v^b-scheme, ∇^b_x, with its corresponding definition domains $\widetilde{S}^b_{e_1,t}$ and $\widetilde{S}^b_{e_2,t}$. ∇^b_x is implemented with a forward difference scheme as in Eq. (3.29). The difference between ∇^f_x and ∇^b_x lies only in their definition domains, $\widetilde{S}^f_{e_i,t}$ and $\widetilde{S}^b_{e_i,t}$.

In the example in Figure 3.2, the spatial gradient reduces to a forward spatial derivative. It acts on convective derivatives (or functions defined on the square nodes attached to the optical flow vectors) and returns a function defined on the triangular nodes on the spatial edges between convective derivatives. Each triangle in Figure (3.2) represents a value of $\nabla^f_x \partial^f_v u$, and therefore a term in the energy. The gray triangles correspond to the spatial gradients that are set to zero in the definition of the operator. They cannot be computed since they require a convective derivative not in S^f_t. The white triangles depict the set $\widetilde{S}^f_{x,t}$. Conversely the spatial divergence acts on the triangular nodes, and returns a function defined on the squares.

3.2.3.3 Treatment of occluded pixels

Most optical flow algorithms assign for all pixels an optical flow vector, even if they are occluded in the adjacent frame. These "false correspondences" can be detrimental to the performance of the method, and have to be removed from the energy.

Occlusions are intrinsic to the problem of the optical flow computation. In fact, some optical flow algorithms detect occlusions as part of the estimation of

the movement [ADPS07, ARS11]. Such algorithms output an occlusion mask together with the optical flow. Many optical flow algorithms however, do not provide occlusion masks. To be able to work with such optical flows, we describe in Section 5 a simple method to detect occlusions. Any other occlusion detection method could be used instead.

In the following we assume that occluded pixels have been detected, either as part of the optical flow algorithm or by a post-processing detection step. We denote by $K_t^f, K_t^b \subset \Omega_t$ the sets of forward and backward occluded pixels, where a forward occluded pixel is a pixel visible in frame t but not visible in frame $t+1$ and a backward occluded pixel is a pixel visible in t and not visible in frame $t-1$.

Let us consider the forward displacement from frame t to frame $t+1$. At a pixel $x \in K_t^f$, the forward optical flow establishes a correspondence between x, visible at t, and $x + v^f(x,t)$ which is not visible at $t+1$. The terms in the energy using the convective derivative associated to this correspondence needs to be removed from our energy.

Case of the 1^{st}-order model In the case of the 1^{st}-order model, removing these terms amount to setting to zero any convective derivative involving the occluded pixel (x,t). For that, we introduce the operator $Occ^f : \Omega \times \{0,1,\dots,T\} \to \{0,1\}$ as

$$Occ^f(x,t) = \begin{cases} 0, & x \in K_t^f, \\ 1 & \text{otherwise} \end{cases}$$

An analogous defintion is given to Occ^b.

Case of the 2^{nd}-order model In the case of the 2^{nd}-order model, the situation is more delicate. In order to remove the terms from the energy using the convective derivative associated to a wrong correspondence, we need to set to zero any spatial partial derivative involving $\partial_v^f u(x,t)$. To that aim, we introduce a *forward occlusion tensor* $\kappa^f : \Omega \times \{0,1,\dots,T\} \to \{0,1\}^{2\times 2}$ as the following diagonal matrix

$$\kappa^f(x,t) = \begin{bmatrix} \kappa_{e_1}^f(x,t) & 0 \\ 0 & \kappa_{e_2}^f(x,t) \end{bmatrix}.$$

When applying $\kappa^f(x,t)$ to $\nabla_x^f \partial_v^f u(x,t)$, $\kappa_{e_1}^f : \Omega \times \{0,1,\dots,T\} \to \{0,1\}$ acts on the spatial partial derivatives in the $e_1 = (1,0)$ direction, whereas $\kappa_{e_2}^f : \Omega \times \{0,1,\dots,T\} \to \{0,1\}$ applies to the partial derivative in the direction $e_2 = (0,1)$.

Given K_t^f we define $\kappa_{e_1}^f$ as

$$\kappa_{e_1}^f(x,t) = \begin{cases} 0 & \text{if } x \in K_t^f \text{ or } x + e_1 \in K_t^f, \\ 1 & \text{otherwise.} \end{cases} \tag{3.30}$$

Figure 3.3: Effect of the occlusion tensor. The image on the left shows an output without using the occlusion tensor. On the right, the output at the same frame with occlusion handling.

A similar definition is given to $\kappa^f_{e_2}$ and the same applies for the *backward occlusion tensor* κ^b.

Figure 3.3 gives and example of a result obtained with and without the occlusion tensor.

Remark. The removal of the convective derivatives associated to occluded pixels could be equally achieved by modifying the sets S^f_t and S^b_t, without introducing the occlusion tensors $\kappa^{f/b}$. Recall that S^f_t refers to the domain of the convective derivative. In Section 3.2.3 a pixel (x,t) is excluded from S^f_t whenever its forward mapping falls out of the frame domain. This situation can indeed be regarded as an occlusion: $(x + v^f(x,t), t+1)$ is not visible. Thus, the same treatment could be given for the pixels in $K^{f/b}_t$. In doing so, the treatment of occluded pixels becomes implicit in the definition of the operators $\partial_v^{f/b}$ and $\nabla_x{}^{f/b}$. For the sake of clarity, we use the occlusion tensors making the occlusion handling explicit.

3.2.3.4 Minimizing energy (3.22)

Energy (3.22) is quadratic, thus the Euler-Lagrange equation is given by the following linear system:

$$\partial_v^* \mathrm{div}_x(\kappa(x,t)\nabla_x \partial_v u(x,t)) = 0, \qquad (x,t) \in O, \tag{3.31}$$

with the boundary conditions specified above (and writing $u = u_0$ at $\widetilde{O} \setminus O$). We solve this equation using the conjugate gradient method.

Let us briefly discuss the stencil of the Euler-Lagrange equation and its boundary conditions. For simplicity, we base this discussion on the one dimensional sequence of Figure 3.2, and we assume that there are no occlusions, that

is $\kappa(x,t) = 1$ for all $(x,t) \in \Omega$. Figure 3.2(b) shows the stencil of the Euler-Lagrange equation at $(x = 3, t = 3)$, the pixel shown in blue. Determining $\partial_v^* \mathrm{div}_x \nabla_x \partial_v u(x,t)$, requires the values of $\mathrm{div}_x \nabla_x \partial_v u$ at the four blue squares. For computing these spatial divergences, $\nabla_x \partial_v u$ is needed at the triangular nodes highlighted in blue. Now, these require the convective derivative to be computed at the red square nodes, resulting the stencil shown by the red circles (in addition to the original (x,t)).

If at any of these steps, one of the required quantities falls out of Ω, it is assumed to be zero. Depending on the optical flow, this amounts to setting to zero either (a) $\mathrm{div}_x \nabla_x \partial_v u$ and $\nabla_x \partial_v u$, or (b) only $\nabla_x \partial_v u$, or (c) only $\mathrm{div}_x \nabla_x \partial_v u$. This comes as a consequence of the definition of the adjoint operators ∂_v^* and div_x.

Remark. Energy (3.22) imposes a spatially smooth brightness change. This is appropriate for blurry shadows as those cast by a diffuse light, but it is not suitable for sequences with sharp shadows, such as those cast by a hard light illumination. In these situations, the following energy can be used instead:

$$E(u) = \sum_{(x,t)\in\widetilde{O}} \|\kappa(x,t) \nabla_x \partial_v u(x,t)\|. \tag{3.32}$$

Minimizing this energy takes the form of a total variation minimization problem. It can be solved by an implicit gradient descent given by

$$u^{j+1} = \arg\min_u E(u) + \frac{1}{2\tau}\|u - u^j\|^2, \tag{3.33}$$

where $\tau > 0$. Each iteration of the gradient descent entails the resolution of a convex problem similar to the total variation model for denoising [ROF92, Cha04]. The solution of (3.33) is computed by solving its dual problem using the algorithm in [Cha04] for instance.

4 A de-blurring scheme for the convective derivative (DSCD)

In Section 3.2, we presented two different schemes for the discretization of the convective derivative, namely the v^f- and v^b-schemes. In the present section we comment on the behaviour of these schemes. This discussion will lead us to the derivation of a hybrid scheme that exploits the intrinsic properties of the v^f- and v^b-schemes. The resulting hybrid scheme allows to handle a much larger number of frames.

4.1 A motivating example

For the sake of this discussion, we consider minimizing (3.22) with $\kappa(x,t) = I$ for all $(x,t) \in \widetilde{O}$ (*i.e.* no occlusions):

$$E(u) = \sum_{(x,t)\in\widetilde{O}} \|\nabla_x \partial_v u(x,t)\|^2.$$

In a general case, the minimum energy may be non-zero due to incompatible boundary conditions. Let us assume that the boundary conditions are compatible and let u be a minimizer of E with zero energy, *i.e.* $E(u) = 0$. In this case, we have that

$$\begin{aligned} \partial_v u(x,t) &= g(t), && \forall (x,t) \in \widetilde{O}\,, \\ u(x,t) &= u_0(x,t), && \forall (x,t) \in \widetilde{O} \setminus O\,, \end{aligned} \tag{4.1}$$

where $g(t)$ is the constant illumination change rate at frame t obtained from the boundary conditions. Additionally, we will assume that there is no illumination change and therefore $g(t) = 0$ for all t.

Let us now consider a concrete example of a one-lid problem where we fix the first frame $t = 0$ and set it as a Dirichlet boundary condition. In this case Eq. (4.1) can be solved by propagating forward the information at the lid sequentially from one frame to the next. In this context the v^b-scheme is an explicit scheme, whereas the v^f-scheme is implicit. To see this, let us consider that the problem is one-dimensional with constant optical flow $v^f(x,t) = v_0 \in (0,1)$, and correspondingly $v^b(x,t) = -v_0$.

Let us discuss the effect of using the v^b-scheme for solving (4.1). In this case, the following recursive relation between two adjacent frames holds for $0 \le t < T, x \in O$:

$$\begin{aligned} u(x,t+1) - \hat{u}(x - v_0, t) = \\ u(x,t) - [v_0 u(x-1,t) + (1-v_0)u(x,t)] = 0. \end{aligned} \tag{4.2}$$

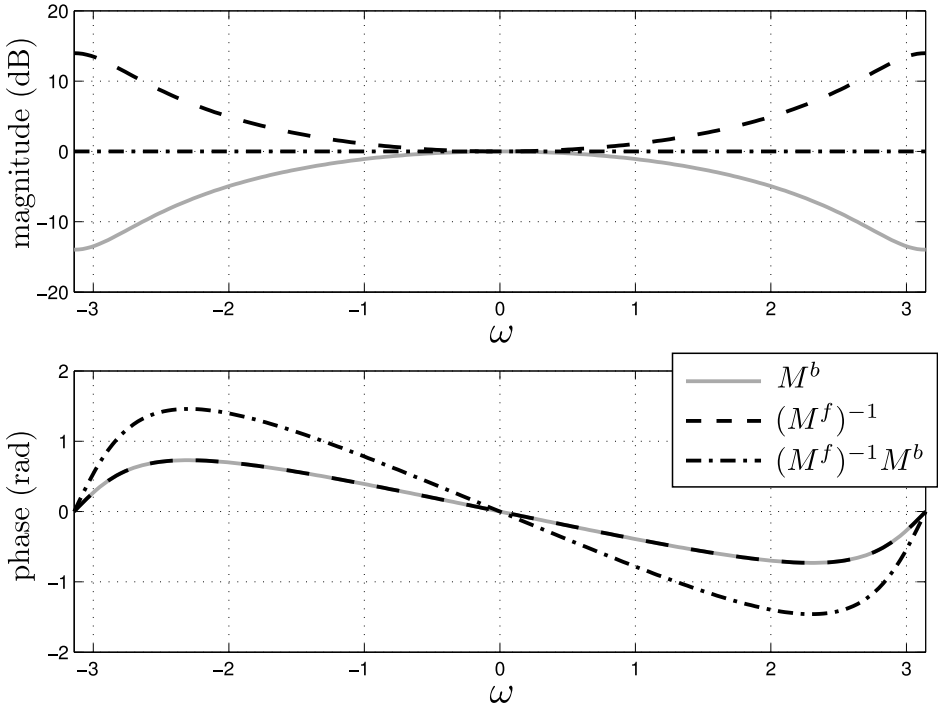


Figure 4.1: Analysis of a zero-energy solution for a one-lid problem, with a purely translational motion. Frequency responses of the M^b and $(M^f)^{-1}$ filters, and their composition $(M^f)^{-1}M^b$.

Thus, the values of $u(\cdot, t+1)$ are explicitly determined by applying the averaging operator with coefficients $[1-v_0, v_0]$ to frame $u(\cdot, t)$. Denoting this operator as M^b, we can describe this relation as: $u(\cdot, t+1) = M^b u(\cdot, t)$. It is interesting to look at the frequency response of the averaging filter M^b, given by

$$M^b(\omega) = (1-v_0) + v_0 e^{i\omega}.$$

This frequency response is shown in Figure 4.1. It has an approximately linear phase for low frequencies with a slope of v_0. Thus, the filter shifts v_0 pixels the low frequency components of the signal. Note also that there is a significant attenuation of medium and high frequencies. By recursing equation (4.2) we can express the solution at frame t in terms of the first frame (the lid) as $u(\cdot, t) = (M^b)^t u(\cdot, 0)$. Therefore, the solution while being shifted according to the constant optical flow, becomes increasingly blurry with t.

A similar argument for the implicit v^f-scheme reveals that for $0 \leq t < T$,

$x \in O$ we have

$$0 = \hat{u}(x, t+1) - u(x,t) = [(1-v_0)u(x,t+1) + v_0 u(x+1, t+1)] - u(x,t) \tag{4.3}$$

which can be written as $M^f u(\cdot, t+1) = u(\cdot, t)$, where M^f applies the averaging operator $[v_0, 1-v_0]$ to $u(\cdot, t+1)$. $u(\cdot, t+1)$ is given by the pseudo-inverse of M^f applied to $u(\cdot, t)$. The frequency response of the inverse operator is given by

$$(M^f)^{-1}(w) = \frac{1}{(1-v_0) + v_0 e^{i\omega}} = \frac{1}{\overline{M^b(w)}},$$

where $\overline{M^b(w)}$ denotes the convex conjugate of $M^b(w)$. Figure 4.1 shows the modulus and phase of $(M^f)^{-1}$. The phase is the same as for M^b, corresponding to a shift of v_0 of the low frequency components. However, the modulus is inverted: high frequencies are amplified. Therefore, the repeated application of the pseudo-inverse acts as an inverse smoothing, which enhances the high frequencies in the solution (sharpening) introducing numerical artifacts which accumulate along time.

It is interesting to note that the effects of the v^b and v^f-schemes for discretizing the convective derivative are opposite. The v^b-scheme introduces blurring in the solution, while the v^f-scheme sharpens the solution but also introduces oscillations. This suggests that they can be combined into a hybrid scheme with the hope that their negative effects cancel out.

4.1.1 The DSCD: a mixed scheme

We propose a mixed scheme which we call the *de-blurring scheme for the convective derivative* (DSCD for short). The idea of the DSCD is to attain this objective by alternating between the v^b and the v^f-schemes. Shortly, if from $t = 0$ to $t = 1$ we apply the v^f-scheme, then from frame $t = 1$ to $t = 2$ we apply v^b-scheme and so on.

There are two ways to implement the DSCD, depending on whether it starts at frame $t = 0$ with the v^b or the v^f-scheme. This determines the way in which the data given at the lid is related to subsequent frames: Either with an explicit averaging filter in the case of starting with the v^b-scheme, or with an implicit sharpening filter when starting with the v^f-scheme. The one that starts with the v^f-scheme does not use the optical flows at odd frames: It will use a v^f step at $t = 0$ with the forward optical flow from $t = 0$ to $t = 1$; then it will use a v^b step at frame $t = 2$ with the backward optical flow from $t = 2$ to $t = 1$, and so on. We call this scheme the *even assignation* of the DSCD (or even DSCD). Alternatively, the *odd assignation* of the DSCD (or odd DSCD) starts with the v^b-scheme and only uses the forward and backward optical flows at odd frames.

Let us discuss the resulting schemes when applied to the 1D example with the constant translational optical flow given above. Suppose t is even, and consider the odd DSCD. Then, between frames t and $t+1$ we use the v^b-scheme,

and the v^f-scheme between $t+1$ and $t+2$. Following the previous discussion, we have that $u(\cdot, t+2) = (M^f)^{-1}M^b u(\cdot, t)$. The frequency response of the compound filter $(M^f)^{-1}M^b$ has a flat magnitude: the blurring effect of M^b and sharpening effect of $(M^f)^{-1}$ cancel out. The phase is approximately linear for low and medium frequencies, but now the slope is $2v_0$ corresponding to the shift between two frames. The same holds for the even DSCD, since two linear filters commute. Thus, if we apply both DSCDs to the one-lid problem at hand, the result obtained should coincide at even frames, but will differ at odd frames. For the odd DSCD, odd frames are computed by a blurring M^b filter of the previous even frame, whereas the even DSCD applies a sharpening $(M^f)^{-1}$ filter.

This behaviour can be appreciated in Figure 4.2. The Figure compares the results of the v^b and v^f schemes together with both DSCDs in a one-lid problem with a purely translational optical flow. The experiment was artificially generated by translating an image with a constant horizontal displacement of $v_0 = [-0.426, 0]$ px/frame. An editing domain is also generated by translating a binary mask with the same optical flow. To test the different schemes we consider a one-lid problem on the given editing domain. In this way we can qualitatively and quantitatively evaluate the ability of the schemes to propagate the first frame. A good propagation should recreate the original sequence inside the editing domain.

As expected, the v^b-scheme incrementally blurs the result in the horizontal direction. The results are not shown for the v^f-scheme, since it rapidly amplifies high horizontal frequencies, destroying the signal within just a couple of frames. The results for both DSCD schemes perform a better propagation throughout the whole sequence (41 frames). To better appreciate the differences between the even and the odd DSCD, we show the result for three pairs of frames formed by an even frame and the subsequent odd frame. Both DSCDs yield very similar results at even frames, while they differ at odd frames. In particular the even DSCD shows high frequency artifacts due to the application of the sharpening filter $(M^f)^{-1}$.

The plots in Figure 4.3(a) show the root mean square error (RMSE) between the sequences obtained by each of the four propagation schemes and the original sequence. The RMSE rapidly grows with the iterations for both the v^f and v^b schemes. The RMSE curves for both DSCD schemes grow considerably slower. The even DSCD shows a high RMSE at odd frames due to high frequency artifacts. Notice that both DSCDs present a very similar RMSE at even frames. The lowest RMSE is attained by the odd DSCD, in accordance to the results shown in Figure 4.2.

4.2 Scope and limitations of the previous analysis

The preceding discussion holds only for zero-energy solutions with a purely translational flow. However, it provides some insights on the behaviour of the DSCD on more general cases.

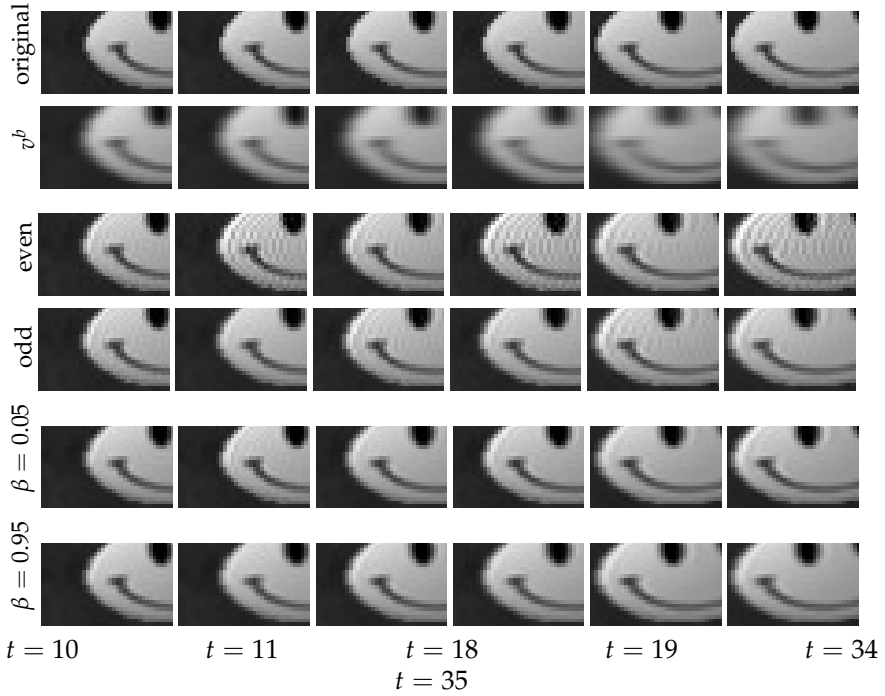
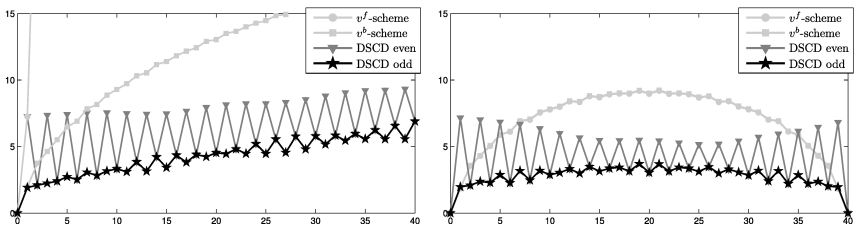


Figure 4.2: Results obtained for a synthetic one-lid problem with a constant translation of $v_0 = [-0.425, 0]$ px/frame, shown at frames $t = 10, 11, 18, 19, 34, 35$ (columns from left to right). First four rows from top to bottom: original sequence (ground truth), v^b-scheme, even DSCD, odd DSCD. The v^f-scheme is not shown since it destroys the signal after just a couple of frames. The last two rows show results obtained with a combination of the even and odd DSCD explained in Section 4.3.1. Fifth row: $\beta = 0.05$ in Eq. (4.6), *i.e.* a combination of 95% of even DSCD with 5% of odd DSCD in the energy. Sixth row: same as fifth row but with $\beta = 0.95$. These correspond to the way we propose for the DSCD to be used in an energy. Note that the high frequency artifacts are greatly diminished, specially with $\beta = 0.95$.



(a) One-lid case. (b) Two-lids case.

Figure 4.3: Evolution of the root mean square error w.r.t. the ground truth corresponding to the synthetic problem with a constant translation of $v_0 = [-0.425, 0]$ px/frame. The sequence has 40 frames. Some frames of the one-lid results are shown in Figure 4.2. The RMSE curves corresponding to the results shown in the last two rows of Figure 4.2, have been omitted to avoid cluttering in the graphs. Let us note that they behave similarly to the odd DSCD.

One-lid problems. In a real one-lid problem, the boundary conditions may not be compatible, and the Euler-Lagrange equation cannot be reduced to solving the PDE $\nabla_x \partial_v u(x,t) = 0$. Furthermore, the cancellation of the blurring and sharpening is exact only when the flow is translational. Still, a similar behavior of the schemes can be observed for one-lid problems with an approximately fronto-parallel movement, even if they do not correspond to a zero energy solution with a pure translational flow. Both DSCDs are able to propagate the information for a much larger number of frames, with much less noticeable artifacts.

This is illustrated in Figure 4.4, for an example with a real sequence. The editing surface is mainly translated, but also suffers mild tilts and zooms. The optical flow was computed using the algorithm of [BM11]. The results shown were obtained with the energy in Eq. (3.22) for a one-lid problem using the v^b and v^f-schemes, as well as for the DSCD. The results of the DSCD were obtained using an energy which combines the even and odd DSCDs (Eq. (4.6), setting $\beta = 0.02$, which correspond to a 98% of even DSCD and a 2% of odd) and that will be explained in Section 4.3.1. Let us just say for the moment that the variational combination of even and odd DSCD schemes greatly attenuates the formation of high frequency artifacts in the solution. The solution obtained with the v^b-scheme progressively blurs the data on the lid (second row in Figure 4.4). On the other hand, the v^f-scheme gradually amplifies high spatial frequencies (third row in Figure 4.4). The DSCD allows to propagate the logo throughout the sequence without appreciable artifacts (fourth row in Figure 4.4). The last two rows in Figure 4.4 will be explained below.

Two-lid problems. The situation for a two-lid setting is different, specially for the v^f and v^b schemes. For these schemes, the boundary conditions at both lids are not likely to be compatible. Consider for instance the v^b-scheme. To have compatible lids, the second lid should not only be a transformed version of the first lid according to the motion, but also would have to account for the blurring caused by the v^b-scheme. Of course in a practical editing application, both lids are non-blurred, thus the minimum is not a zero energy solution and the analysis of the preceding section does not apply. Intuitively speaking, there are two opposing effects competing: the first lid should be blurred as time increases, whereas the second lid at $t = T$ should be sharpened backwards in time (and vice versa for the v^f-scheme). We have observed empirically that the averaging effect dominates, with results presenting considerable blur at intermediate frames away from the lids. This has been observed in the literature as well [KCR05, SMTK06].

Figure 4.3(b) depicts RMSE curves for a two-lid version of the synthetic translational example. Note that in this case the v^f-scheme and v^b-scheme are symmetrical, with a high RMSE at intermediate frames. Close to lids, the RMSE is lower than that of the even DSCD. As in the one-lid case, the odd DSCD shows the lowest RMSE. This is a consequence of the fact that the sequence has an odd number of frames. Thus, the even DSCD is connected to both lids through

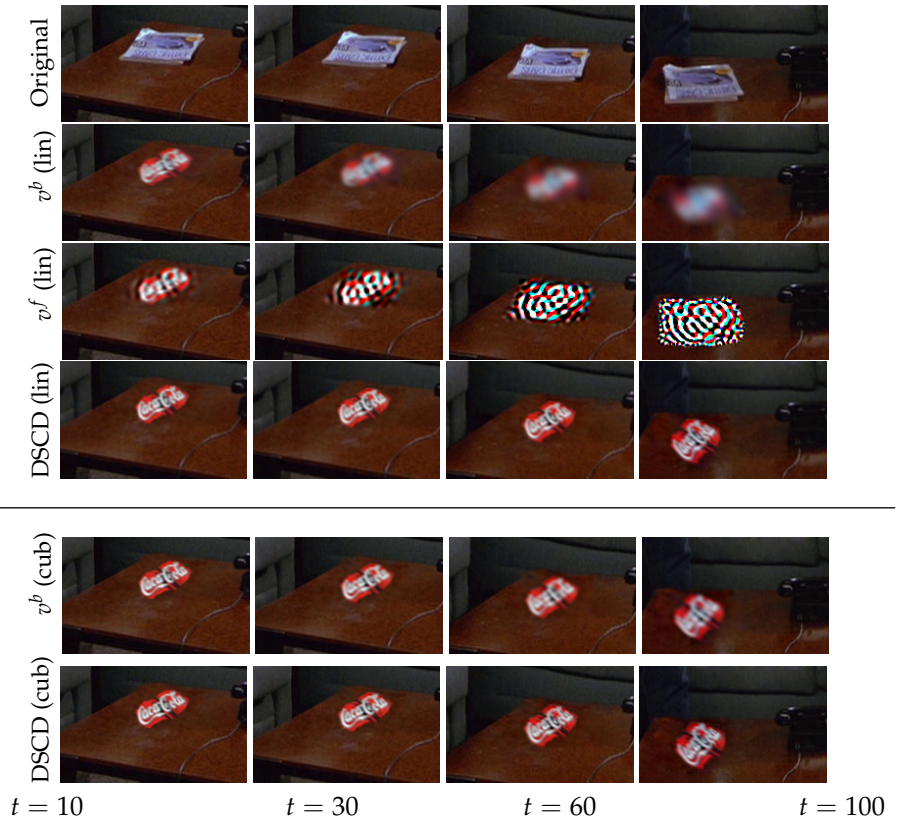


Figure 4.4: Results obtained for a real one-lid problem. The lid is placed at the first frame $t = 0$. The first row shows the original sequence before the editing. The following three rows show the results obtained using bi-linear interpolation to implement the convective derivative schemes. From top to bottom show: Results obtained with v^b-scheme (explicit), v^f-scheme (implicit), and DSCD hybrid scheme. The last two rows were computed using bi-cubic interpolation. From top to bottom: v^b-scheme (explicit) and DSCD hybrid scheme. Both DSCD results were obtained with the variational combination of the even and odd DSCD of Eq. (4.6) with $\beta = 0.02$ (thus, mainly even DSCD), explained in Section 4.3.1.

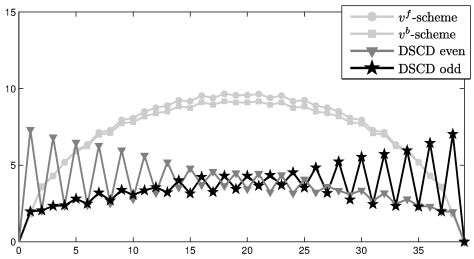


Figure 4.5: Evolution of the root mean square error, in the two lid-setting, w.r.t. the ground truth corresponding to the *smiley* synthetic problem with a constant translation and with an even number of frames (40 frames).

an implicit v^f step, whereas the odd DSCD is connected to them with explicit v^b steps. Let us now consider an even number of frames by removing the last frame in the sequence, *i.e.* now the first lid is at frame $t = 0$ and the second lid is at frame $t = 39$ instead of $t = 40$. Figure 4.5 shows the resulting RMSE for the *smiley* experiment.

The behaviour of the v^b and v^f is roughly the same, but there is a change in the behaviour of the even and odd DSCDs: for the sequence with an odd number of frames, the even and the odd DSCDs coincide at even frames, and at odd frames, the even DSCD has a higher RMSE, associated with high frequency artifacts introduced by the sharpening step. When the total number or frames is even (Figure 4.5), for the first 10 frames in the sequence, the behaviour of even and odd DSCDs resembles the one in Figure 4.3(b): Both DSCDs coincide at even frames, and the even DSCD yields high RMSE at odd frames. However, the situation is inverted by the end of the sequence towards the second lid: the DSCDs coincide on odd frames, and now it is the odd DSCD with high RSME at even frames.

The reason for this become clear when we write the DSCD energies in terms of the M^b and M^f interpolation filters:

$$E_{\kappa}^{\text{odd}}(u) = \|M^b u_0(\cdot,0) - u(\cdot,1)\|^2 + \|u(\cdot,1) - M^f u(\cdot,2)\|^2 + \cdots + \begin{cases} \|M^b u(\cdot,T-1) - u_0(\cdot,T)\|^2 & \text{if } T+1 \text{ is odd,} \\ \|u(\cdot,T-1) - M^f u_0(\cdot,T)\|^2 & \text{if } T+1 \text{ is even.} \end{cases}$$

$$E_{\kappa}^{\text{even}}(u) = \|u_0(\cdot,0) - M^f u(\cdot,1)\|^2 + \|M^b u(\cdot,1) - u(\cdot,2)\|^2 + \cdots + \begin{cases} \|u(\cdot,T-1) - M^f u_0(\cdot,T)\|^2 & \text{if } T+1 \text{ is odd,} \\ \|M^b u(\cdot,T-1) - u_0(\cdot,T)\|^2 & \text{if } T+1 \text{ is even.} \end{cases}$$

The first term determines the nature of the connection with the first lid. As previously discussed, the odd DSCD enforces an explicit (averaging) link between the lid and $u(\cdot,1)$. For the even DSCD, on the other hand, the link is

implicit, responsible for the high RMSE erros at odd frames when in close to the first lid.

The nature of the connection to the last lid depends on the parity of the total number of frames $T+1$. For an odd number of frames the connection to the last lid is of the same type as the connection to the first lid. The odd DSCD is linked to the second lid via a v^f step, which when seen in the backwards direction of propagation, is an explicit (and thus averaging) step. The even DSCD, is linked to the last lid through a term enforcing an implicit sharpening relation between $u(\cdot, T-1)$ and the lid, when seen in the backwards direction. This is the reason for the symmetric behaviour of both DSCD curves in Figure 4.3(b).

However, when the $T+1$ is even, the situation is reversed: the odd DSCD establishes an implicit link to the second lid, whereas the even DSCD is linked explicitly. This explains the exchange in the behaviour of even and odd DSCDs in Figure 4.5.

Zooms. Let us discuss the case of sequences with significant zooms on the edited surface. Consider for example a sequence consisting of a close-up on the edited object: the resolution of the edited surface increases with each frame. In a one-lid setting, if the lid is placed on the first frame, the scheme will propagate a low resolution version of it. Indeed, in this case the problem becomes one of super-resolution and the model will do as good as the interpolation scheme used. One can solve it by performing the editing on the last high resolution frame and setting it as the lid. In Section 6.4 we discuss further this issue in a more complex example.

Higher order interpolation. The different schemes for the convective derivative can be implemented using higher orders of interpolation to estimate u at subpixel positions in Eqs. (3.23) and (3.24) (and similarly for both DSCD schemes). As an example, we have computed the outputs for the v^b scheme and the DSCD using bi-cubic interpolation. The results are shown in Figure 4.4 for the one-lid setting. As before, the results of the DSCD correspond to energy (4.6), which combines the even and odd DSCD (we set $\beta = 0.02$, thus a predominantly even DSCD). The higher order interpolation reduces the rate at which the v^b-scheme blurs the frames, but eventually the blurring becomes apparent (as also noted in [KCR05, SMTK06]). The result obtained is considerably better than the one for the v^b-scheme with bi-linear interpolation, but it is still blurrier than the results obtained with the bi-linear DSCD schemes. As with the bi-linear interpolation, the bi-cubic v^f-scheme completely destroys the signal and its result is omitted. The bi-cubic DSCD behaves very similar to the bi-linear one. The reason for this is that the motion is mostly fronto-parallel. Thus, by alternating between the v^b and v^f schemes, the DSCD approximately compensates for the blurring caused by the low-order bi-linear interpolation. For sequences with significant zooms in which the lid is placed at a low resolution frame (as discussed previously), a higher order interpolation scheme yields better results. An example is given in Chapter 12, Figure 6.13. This improvement comes at the

expense of a greater computation cost (*e.g.* the bi-cubic interpolation uses a 16 point stencil whereas the bi-linear interpolation uses only four). For most of our experiments we found good results using the bi-linear DSCD, without the need to consider higher order interpolators.

4.3 An operator implementing the DSCD

Let us recall that there are two versions of the DSCD schemes. One that starts at $t = 0$ with the v^f-scheme and another one starting at $t = 1$ with the v^b-scheme. We refer to them as *even-assignation* and *odd-assignation* respectively. These schemes for the convective derivative can be implemented by defining corresponding operators.

Let us consider first the odd-assignation and define the following hybrid operator for the convective derivative:

$$h_v^{\text{odd}} u(x,t) = \begin{cases} \partial_v^f u(x,t), & t \text{ odd}, \\ \partial_v^b u(x,t+1), & t \text{ even}, \end{cases}$$

$$= \begin{cases} \hat{u}(x + v^f(x,t), t+1) - u(x,t), & t \text{ odd and} \\ & x \in S_t^f, \\ u(x,t+1) - \hat{u}(x + v^b(x,t+1), t), & t \text{ even and} \\ & x \in S_{t+1}^b, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

This operator computes the convective derivatives corresponding to the forward and backward optical flows of the odd frames. Note that the backward derivatives are shifted: $\partial_v^b u(x,t+1)$ is assigned to location (x,t). For this reason we do not consider h_v^{odd} as a discretization of the convective derivative, and use the notation h_v instead of ∂_v. The h here stands for *hybrid*.

As in Section 3.2.3.2, we define a corresponding spatial gradient ${\nabla_x}^{\text{odd}}$ which takes into account the domain where h_v^{odd} can be computed. For the definition of the gradient, we define the sets $\widetilde{S}_{e_i,t}^{\text{odd}}$ for each frame t. Recall that these sets contain the locations x where both convective derivatives needed to compute the partial derivative in the direction e_i are computable. Due to the definition of h_v^{odd} it can be seen that if t is odd, $\widetilde{S}_{e_i,t}^{\text{odd}} = \widetilde{S}_t^f$, and if t even, $\widetilde{S}_{e_i,t}^{\text{odd}} = \widetilde{S}_{t+1}^b$. The diagram in Figure 4.6 show how the convective derivatives and their gradients are taken.

Analogously, we define the corresponding h_v^{even} implementing the even-assignation DSCD (using the forward and backward optical flows at even frames), and its associated spatial gradient ${\nabla_x}^{\text{even}}$.

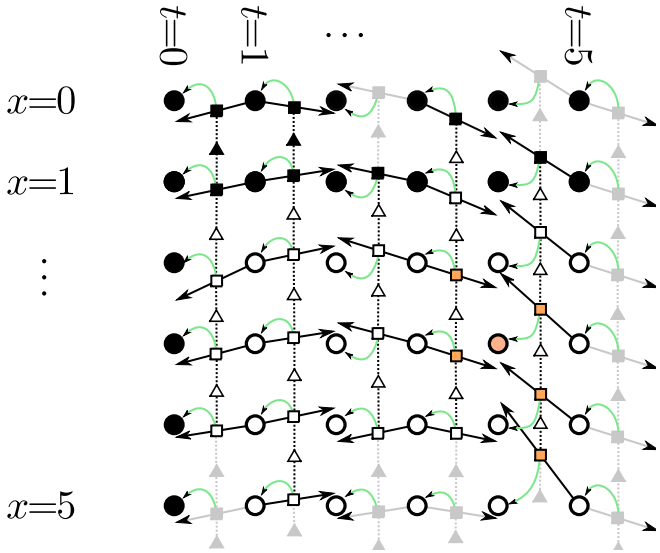


Figure 4.6: The hybrid scheme h_v^{odd} for the convective derivative using the forward and backward optical flow of odd frames, for the same one dimensional image sequence shown in Figure 3.2. The same graphical conventions as in Figure 3.2(a) are used.

4.3.1 Using the DSCD in the 2^{nd}-order model

Based on the operator implementing the odd DSCD we define the following energy

$$E_\kappa^{\text{odd}}(u) = \sum_{(x,t)\in\tilde{O}} \|\kappa^{\text{odd}}(x,t)\nabla_x{}^{\text{odd}} h_v^{\text{odd}} u(x,t)\|^2. \tag{4.5}$$

Note that we consider an occlusion tensor $\kappa^{\text{odd}} : \Omega \times \{0,1,\ldots,T\} \to \{0,1\}^{2\times 2}$ for the forward and backward optical flows at odd frames, defined in an analogous way to the forward and backward occlusion tensors in Section 3.2.3.3.

Similarly we define an energy E_κ^{even}, corresponding to the hybrid operator h_v^{even} implementing the even-assignation DSCD (using the forward and backward optical flows at even frames).

The proposed energy for using the DSCD considers both the odd and even assignations, and reads

$$E_\beta(u) = \beta\, E_\kappa^{\text{odd}}(u) + (1-\beta)\, E_\kappa^{\text{even}}(u), \tag{4.6}$$

where $\beta \in [0,1]$ is a weighting coefficient. A value of $\beta = 1$ yields the odd-assignation of the DSCD, whereas $\beta = 0$ corresponds to the even-assignation.

Both versions of the DSCD exhibit a comparable behaviour. In general, a v^f step permits to recover the frequencies smoothed by the previous v^b step.

However, as shown in Figure 4.2, it may also introduce other high frequencies, which on the long term will build up as high frequency artifacts (particularly for the even DSCD). These can be attenuated by adding to the used version of the DSCD a small component of the other one, which corresponds to values of $\beta \in (0,1)$, either close to 0 or 1. This can be understood in the context of the previous example: we add a slight amount of averaging to the implicit steps, and sharpening to the explicit ones. In practice we found that for the odd-assignation setting a value of $\beta \approx 0.95$ turns out to alleviate the DSCD from high frequency artifacts and without introducing much blurring (correspondingly, $\beta \approx 0.05$ for the even-assignation). This can be appreciated in the last two rows of Figure 4.2.

The use of both DSCD schemes has another advantage. Let us consider, for example, that $\beta = 0$ (*i.e.* only the even-assignation DSCD is used). Let us assume as well that $\kappa^{\text{even}}(x,t) = I$ for all $(x,t) \in \widetilde{O}$. Pixels on odd frames are included in the energy only if they form part of the interpolation stencil of a pixel in an adjacent even frame. Depending on the optical flow, it may occur that some pixels in O only appear in the energy with a small or even zero weight. This may cause the system to be ill-conditioned. On the other hand, with $\beta \in (0,1)$ and assuming that $\kappa^{\text{odd}}(x,t) = I$, every pixel in $\widetilde{O}$ is at least "connected" to the rest by its own forward and backward optical flows, which suggests a better conditioned system.

In the case of occlusions, the occlusion tensors remove terms from the energy by setting them to zero (see Section 3.2.3.3). This might cause the system to become ill-conditioned, even with $\beta \in (0,1)$. For this reason we add a spatial regularization term. The resulting energy reads

$$E_{\beta,\lambda}(u) = \frac{1}{2}E_\beta(u) + \frac{\lambda}{p}\sum_{(x,t)\in\widetilde{O}} \|\nabla_x^+ u(x,t)\|^p, \tag{4.7}$$

where $\lambda \geqslant 0$ and $p = 1$ or 2 (in our experiments we have taken $p = 2$). If there are no occlusions, no elements in the energy are removed and λ can be set to zero. Otherwise, λ is set to a small value so that the smoothing effect of the spatial regularization is only noticeable on "weakly connected" pixels. With the addition of this spatial regularization term, if $p = 2$ then the solution is unique as long as there exist at least one Dirichlet boundary condition on u in each frame (assuming that $O_t \neq \Omega$ for all $t \in \{0, 1, \cdots, T\}$). The problem of existence and uniqueness of solutions (both in the continuous and discrete settings, and also $p \in \{1,2\}$) is considered in Appendices A.3, A.4 and A.5 under some assumptions on the optical flow that amount to say that trajectories of points are well defined. A more thorough analysis on the conditioning of the resulting system of equations is an important issue and further study is required.

Other regularization terms could also be used. In particular, in our experiments, we sometimes found better results with the addition of a temporal regularization term given by $\gamma \sum_{(x,t)\in\widetilde{O}} \|\partial_v u(x,t)\|^p$, as in [BZCC10, BZS+07] and also as we have already proposed in Section 3.1.1. This term, together with the spatial regularizer, forms a weighted 3D gradient $(\nabla_x^+, \left(\frac{\gamma}{\lambda}\right)^{1/p}\partial_v)$ with the temporal component in the direction of the optical flow. Note that this term enforces

brightness constancy. Therefore, if used in a sequence with significant illumination changes, it should be given a very small weight.

4.4 Implementation details

The minimization of the energy is given by the solution of a linear system which we solve using a conjugate gradient solver. In our implementation we used sparse matrices to store the discrete operators described previously. For example, ∂_v^f can be stored as an $N \times N$ sparse matrix, where N is the number of pixels in $\widetilde{O}$. For each row in the matrix, five values have to be stored: the four bilinear interpolation weights and -1 at the diagonal (see Eq. (3.23)). Storing the operators as sparse matrices, greatly simplifies the computation of the adjoint operators, which can be computed by a simple matrix transpose operation (*e.g.* the matrix associated to $\partial_v^{f^*}$ is the transpose of ∂_v^f).

Let us now give a pseudo-code that implements the 2^{nd}-order model. We will explain how to build the sparse linear system for minimizing $E_{\boldsymbol{\kappa}}^{\text{odd}}(u)$. First let us rewrite the energy in an equivalent form which allows a simpler implementation, and then we present a pseudo-code for building and solving the system.

Simplification of the energy. We start by observing that minimizing

$$E_{\boldsymbol{\kappa}}^{\text{odd}}(u) = \sum_{\widetilde{O}} \|\boldsymbol{\kappa}^{\text{odd}} \boldsymbol{\nabla}^{\text{odd}} h_{\boldsymbol{v}}^{\text{odd}} u\|^2,$$

with respect to u is equivalent to solving the constrained problem (defined over $\Omega \times \{0, 1, \cdots, T\}$):

$$\min_u \sum_{\Omega \times \{0,1,\cdots,T\}} \|\boldsymbol{\kappa}^{\text{odd}} \boldsymbol{\nabla}^{\text{odd}} h_{\boldsymbol{v}}^{\text{odd}} u\|^2 \qquad \text{s.t.} \quad u|_{O^c} = u_0,$$

where O^c denotes the complement of O. Since h_v^{odd} only keeps the forward and backward convective derivatives at odd frames, we can split the energy to expose the forward and backward terms

$$\begin{aligned}\sum_{\Omega \times \{0,1,\cdots,T\}} \|\boldsymbol{\kappa}^{\text{odd}} \boldsymbol{\nabla}^{\text{odd}} h_{\boldsymbol{v}}^{\text{odd}} u\|^2 \quad = & \sum_{\Omega \times \{0,1,\cdots,T\}} \|\boldsymbol{\kappa}^f \boldsymbol{\nabla}^f \left(O_{odd}\, \partial_{\boldsymbol{v}}^f u\right)\|^2 + \\ & \sum_{\Omega \times \{0,1,\cdots,T\}} \|\boldsymbol{\kappa}^b \boldsymbol{\nabla}^b \left(O_{odd}\, \partial_{\boldsymbol{v}}^b u\right)\|^2\end{aligned}$$

where $O_{odd}(\boldsymbol{x}, t)$ is 1 if t is odd, and 0 otherwise. Let us introduce the finite difference gradient $\bar{\nabla} q$ defined over the whole domain $\Omega \times \{0, 1, \cdots, T\}$ and with appropriate Neumann boundary conditions. Recalling the definitions of $\nabla^{f,b}$ and the sets $S^{f,b}$ and $\widetilde{S}_{e_i}^{f,b}$ we observe that we can re-write the gradients in simpler terms

$$\nabla^{f,b} q := \begin{bmatrix} \widetilde{S}_{e_1}^{f,b} & 0 \\ 0 & \widetilde{S}_{e_2}^{f,b} \end{bmatrix} \bar{\nabla} q, \tag{4.8}$$

where the S-sets are being used as indicator functions (i.e. $S(\boldsymbol{x},t)$ is 1 if $(\boldsymbol{x},t) \in S$, and 0 otherwise). Incorporating these definitions in the energy we get:

$$\sum_{\Omega\times\{0,1,\cdots,T\}} \left\| \boldsymbol{\kappa}^f \begin{bmatrix} \widetilde{S}^f_{e_1} & 0 \\ 0 & \widetilde{S}^f_{e_2} \end{bmatrix} \bar{\nabla}\left(O_{odd}\,\partial^f_{\boldsymbol{v}} u\right) \right\|^2 + \sum_{\Omega\times\{0,1,\cdots,T\}} \left\| \boldsymbol{\kappa}^b \begin{bmatrix} \widetilde{S}^b_{e_1} & 0 \\ 0 & \widetilde{S}^b_{e_2} \end{bmatrix} \bar{\nabla}\left(O_{odd}\,\partial^b_{\boldsymbol{v}} u\right) \right\|^2.$$

We further simplify the energy by collapsing the tensors $\bar{\boldsymbol{\kappa}}^f := \boldsymbol{\kappa}^f \begin{bmatrix} \widetilde{S}^f_{e_1} & 0 \\ 0 & \widetilde{S}^f_{e_2} \end{bmatrix}$

$$E^{\text{odd}}_{\boldsymbol{\kappa}}(u) = \sum_{\Omega\times\{0,1,\cdots,T\}} \|\bar{\boldsymbol{\kappa}}^f\,\bar{\nabla}\left(O_{odd}\,\partial^f_{\boldsymbol{v}} u\right)\|^2 + \sum_{\Omega\times\{0,1,\cdots,T\}} \|\bar{\boldsymbol{\kappa}}^b\,\bar{\nabla}\left(O_{odd}\,\partial^b_{\boldsymbol{v}} u\right)\|^2. \tag{4.9}$$

Implementation and minimization. The following pseudo-code explains how the linear system for solving (4.9) is constructed. We will use the following conventions: the monochrome input video u_0, the masks and the flow fields v^f, v^b treated as lexicographically ordered 1d vectors. However, for simplicity we use (x,t) as indices of the entries of the 1d vectors, and the notation $[A(u)](x,t)$ to refer to rows of matrices (index of (x,t)-th row of A in this case).

1. Let m be the number of pixels in $\Omega \times \{0,1,\cdots,T\}$.
2. Generate the masks of the editing domain O, its complement O^c and the pixels in the even/odd frames O_{even} and O_{odd} respectively. Also compute the masks $S^f, \widetilde{S}^f_{e_i}, S^b, \widetilde{S}^b_{e_i}$ and the occlusion tensors $\boldsymbol{\kappa}^f, \boldsymbol{\kappa}^b$. And compute the collapsed tensors $\bar{\boldsymbol{\kappa}}^{f,b} := \begin{bmatrix} \tilde{S}^{f,b}_{e_1}\kappa^{f,b}_{e_1} & 0 \\ 0 & \tilde{S}^{f,b}_{e_2}\kappa^{f,b}_{e_2} \end{bmatrix}$ (as justified above).
3. Construct the following sparse matrices (with lexicographically ordered entries):
   - $K^{f,b}$: $2m \times 2m$ binary diagonal matrices acting on gradients and implementing $\bar{\boldsymbol{\kappa}}^{f,b}$.
   - $S^{f,b}, O, O^c, O_{odd}, O_{even}$: $m \times m$ binary diagonal matrices acting on images and implementing the homogeneous masks.
   - I^f_v: $m \times m$ matrix implementing the forward warping by v^f of its input $[I^f_v(u)](x,t) = \hat{u}(x + v^f(x), t+1)$, where $\hat{u}$ denotes the bi-linear or bi-cubic interpolation of u. Similarly I^b_v for the backward warping.
   - I_0: $m \times m$ identity matrix.
   - $J^f_v := I^f_v - I_0$. $m \times m$ matrix implementing $\partial\cdot$, the forward convective derivative. Similarly $J^b_v := I_0 - I^b_v$.
   - G: $2m \times m$ matrix implementing $\bar{\nabla}\cdot$, the spatial gradient $G := \left[\begin{smallmatrix} G_{e_1} \\ G_{e_2} \end{smallmatrix}\right]$, with $[G_{e_i}(u)](x,t) = u(x+e_i,t) - u(x,t)$.

4. Build the operator. For implementing (4.9) we write a $4m \times m$ matrix

$$A := \begin{bmatrix} K^f G\, O_{odd}\, J_v^f \\ K^b G\, O_{odd}\, J_v^b \end{bmatrix},$$

the energy will be $E_{\boldsymbol{\kappa}}^{odd}(u) = u^T A^T A u$. And for implementing $E_{\boldsymbol{\beta},\lambda}(u)$ we write a $10m \times m$ matrix

$$A := \begin{bmatrix} \beta K^f G\, O_{odd}\, J_v^f \\ \beta K^b G\, O_{odd}\, J_v^b \\ (1-\beta) K^f G\, O_{even}\, J_v^f \\ (1-\beta) K^b G\, O_{even}\, J_v^b \\ \lambda G \end{bmatrix}.$$

5. Resolution of: $u^* = \arg\min_u \|Au\|^2$ s.t. $u|_{O^c} = u_0$.

   - Since the variables $u|_{O^c}$ are fixed we can split the variable $u = Ou + O^c u_0$ and rewrite the problem as $\|AOu + AO^c u_0\|^2$ with homogeneous constraints $u|_{O^c} = 0$. Its solution is then obtained by solving the linear system $Su = b$ where $E := OA^T AO$ and $b = -OA^T AO^c u_0$.
   - The final video is recovered as $u^* = S^{-1}b + u_0$.

   Note that because of the restriction matrices O and $K^{f,b}$ the final system to solve is much smaller than $m \times m$.

5 An occlusion detection method

In this Chapter we discuss a two-step simple method to detect occlusions given a dense optical flow. In the first step we aim to identify regions which are potentially occluded. In the second step we categorize the previously selected regions into occluded and still visible. Figure 5.1 shows an example of the two-step occlusion detection method.

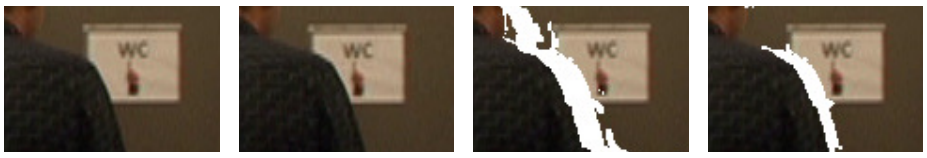
The problem of occlusion detection is intrinsic to the optical flow problem. In fact, some optical flow algorithms estimate occlusions as part of the estimation of the movement [ADPS07, ARS11]. Such algorithms output an occlusion mask together with the optical flow.

Many optical flow algorithms however, do not provide occlusion masks. To be able to work with such optical flows, we describe a simple method to detect occlusions. Any other occlusion detection method could be used instead.

5.1 First step, the detection of potentially occluded regions

Let us consider two adjacent frames $t-1$ and t. In an ideal case the forward optical flow from $t-1$ to t should map points in $t-1$ that exist in t, to their corresponding location in t. Occluded points at $t-1$, do not have a correspondence at t, and therefore an ideal optical flow should not be defined at these locations. The same applies for the backward optical flow.

In practice, most optical flow algorithms compute a dense correspondence from one frame to another. Thus we have a forward dense mapping from $t-1$ to t given by the forward optical flow as $\boldsymbol{\varphi}^f_{t-1} : \Omega_{t-1} \to \Omega_t$, as $\boldsymbol{\varphi}^f_{t-1}(\boldsymbol{x}) = (\boldsymbol{x} + \boldsymbol{v}^f(\boldsymbol{x}, t-1), t)$. Similarly, we have a backward dense mapping at t given by the



(a) Ω_{t-1} (b) Ω_t (c) 1st step applied on Ω_{t-1} (d) 2nd step applied on Ω_{t-1}

Figure 5.1: The proposed two-step occlusion handling. For clarity, we show only the processing related to the forward optical flow from Ω_{t-1} to Ω_t. Figures (a) and (b) show two consecutive frames. (c) shows the selection result of the first step described in Section 5.1. (d) shows the selection result after the second step described in Section 5.2.

backward optical flow, $\boldsymbol{\varphi}_t^b : \Omega_t \to \Omega_{t-1}$.

We will focus in the following on the forward mapping. An analogous discussion holds for the backward mapping.

For a given $\boldsymbol{x} \in \Omega_t$, we consider a neighborhood $Q_{\boldsymbol{x}} \subset \Omega_t$ given by $Q_{\boldsymbol{x}} = \boldsymbol{x} + (-1, 1)^2$. We denote by $[\boldsymbol{\varphi}_{t-1}^f]^{-1}(Q_{\boldsymbol{x}}) = \{\boldsymbol{z} \in \Omega_{t-1} : \boldsymbol{\varphi}_{t-1}^f(\boldsymbol{z}) \in Q_{\boldsymbol{x}}\}$ the pre-image of Q_x under the forward mapping φ_{t-1}^f. Note that this set is also defined even when $\boldsymbol{\varphi}_{t-1}^f$ is not invertible. We define the *forward area* as

$$\mathcal{A}_{t-1}^f(Q_{\boldsymbol{x}}) := \int_{[\boldsymbol{\varphi}_{t-1}^f]^{-1}(Q_{\boldsymbol{x}})} \mathrm{d}\boldsymbol{z}.$$

To compute this integral, we extend the discrete mapping $\boldsymbol{\varphi}_{t-1}^f$ to a continuous spatial domain using a bi-linear interpolation.

If $Q_{\boldsymbol{x}}$ is not being occluded nor dis-occluded from $t-1$ to t, then $[\boldsymbol{\varphi}_{t-1}^f]^{-1}$ is a well defined function on $Q_{\boldsymbol{x}}$. That is, every point in $Q_{\boldsymbol{x}}$ has a unique pre-image by the forward mapping $\boldsymbol{\varphi}_{t-1}^f$. Furthermore, for a locally translational flow $\boldsymbol{\varphi}_{t-1}^f$, we can expect $\mathcal{A}_{t-1}^f(Q_{\boldsymbol{x}})$ to be close to the area of $Q_{\boldsymbol{x}}$, given by $\mathcal{A}(Q_{\boldsymbol{x}}) = \int_{Q_{\boldsymbol{x}}} \mathrm{d}\boldsymbol{z}$. In the case of $Q_{\boldsymbol{x}} = \boldsymbol{x} + (-1, 1)^2$ we have $\mathcal{A}(Q_{\boldsymbol{x}}) = 4$.

Based on the comparison between $\mathcal{A}_{t-1}^f(Q_{\boldsymbol{x}})$ and $\mathcal{A}(Q_{\boldsymbol{x}})$ we will define potentially occluded regions to be further examined. We add a small margin of $\varepsilon = 0.5$ to the comparison, to avoid marking all pixels as potentially occluded. We consider two cases:

1. $\mathcal{A}_{t-1}^f(Q_{\boldsymbol{x}}) > \mathcal{A}(Q_{\boldsymbol{x}}) + \varepsilon$: There is an excess of points mapped into $Q_{\boldsymbol{x}}$ by $\boldsymbol{\varphi}_{t-1}^f$. This situation may arise when there is an occlusion from frame $t-1$ to t, causing several points in Ω_{t-1} to be mapped forward into the same location at frame t. Some of them are being occluded, whereas others remain visible. Therefore, we mark all pixels in $[\boldsymbol{\varphi}_{t-1}^f]^{-1}(Q_{\boldsymbol{x}})$ as candidates for occluded regions, for further examination. Note that there may be other reasons for which $\mathcal{A}_{t-1}^f(Q_{\boldsymbol{x}}) > \mathcal{A}(Q_{\boldsymbol{x}}) + \varepsilon$, for example in the case of a zoom out. In this first step, we treat all of these cases equally.

2. $\mathcal{A}_{t-1}^f(Q_{\boldsymbol{x}}) < \mathcal{A}(Q_{\boldsymbol{x}}) - \varepsilon$: Few points are mapped by $\boldsymbol{\varphi}_{t-1}^f$ into $Q_{\boldsymbol{x}}$. This may occur if $Q_{\boldsymbol{x}}$ lies in an region that has been dis-occluded from $t-1$ to t. If this is the case, some points in $Q_{\boldsymbol{x}}$ at frame t, do not have a correspondence in $t-1$, and we mark points in $Q_{\boldsymbol{x}}$ as candidates for dis-occluded regions (or occluded when looking from t to $t-1$). Their backward optical flow will be examined in the second step. Note that the forward flows arriving at the dis-occluded region might be as well wrong (no point in $t-1$ should have a correspondence in a dis-occluded region at t). Thus we also mark them as candidates for further examination.

The same process is then applied considering the *backward area* mapped into frame $t-1$ from t by the backward optical flow.

Figure 5.1(c) shows an example output of the above discussed first-step occlusion detection.

Remark. A more general method for detecting occlusion candidates can be derived based on the consistency between the forward and backward optical flows [ADPS07]. In addition to the forward area $\mathcal{A}^f_{t-1}(Q_{\boldsymbol{x}})$, one could also define the *backward area* as

$$\mathcal{A}^b_t(Q_{\boldsymbol{x}}) := \int_{\boldsymbol{\varphi}^b_t(Q_{\boldsymbol{x}})} \mathrm{d}\boldsymbol{z}.$$

For an ideal optical flow, the forward and backward areas should be equal.

In an ideal case, the forward and backward flows are symmetrical (see for instance [ADPS07]). Thus, the area of the pre-image of $Q_{\boldsymbol{x}}$ under the forward flow ($\mathcal{A}^f_{t-1}(Q_{\boldsymbol{x}})$) and the area of the image of $Q_{\boldsymbol{x}}$ under the backward flow ($\mathcal{A}^b_t(Q_{\boldsymbol{x}})$) should be equal. Therefore, differences between these areas could be considered as evidence of occlusions or dis-occlusions. Cases 1 and 2 in the above discussion apply, using $\mathcal{A}^b_t(Q_{\boldsymbol{x}})$ instead of $\mathcal{A}(Q_{\boldsymbol{x}})$. In practice, for the sequences we used, both approaches behave similarly.

5.2 Second step, the categorization into occluded and still visible regions

As a result of the first step, we have for each frame t two sets $C^f_t, C^b_t \subset \Omega_t$ of potentially occluded pixels. If a pixel $\boldsymbol{x}$ belongs to C^f_t, then its corresponding position at Ω_{t+1}, given by the forward optical flow $\boldsymbol{\varphi}^f_t(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{v}^f(\boldsymbol{x}, t)$, might be occluded. Similarly, for a pixel $\boldsymbol{x} \in C^b_t$, its corresponding backward position $\boldsymbol{\varphi}^b_t(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{v}^b(\boldsymbol{x}, t)$ in Ω_{t-1} might be occluded. In the first step, all the analysis has been done based on the properties of the mapping between two frames, without considering the visibility of a pixel in the next frame. In this second step we try to determine which pixels in C^f_t and C^b_t are occluded and which are still visible. In what follows we describe this procedure for C^f_t. The same applies to C^b_t.

We base our occlusion detection on the error between a patch from frame t centered at $\boldsymbol{x} \in C^f_t$ and its corresponding patch at frame $t+1$, centered at $\boldsymbol{x} + \boldsymbol{v}^f(\boldsymbol{x}, t)$. We will denote by $p_u(\boldsymbol{x}, t)$ the patch at $(\boldsymbol{x}, t)$. We define the patch error as the squared L_2 distance between corresponding patches:

$$\begin{aligned} e^f(\boldsymbol{x}) &= \|p_u(\boldsymbol{x}, t) - p_u(\boldsymbol{x} + \boldsymbol{v}^f(\boldsymbol{x}, t), t+1)\|^2 \\ &= \sum_{h \in \Omega_p} (u(\boldsymbol{x} + h, t) - u(\boldsymbol{x} + \boldsymbol{v}^f(\boldsymbol{x}, t) + h, t+1))^2, \end{aligned}$$

where $\Omega_p \subset \mathbb{Z}^2$ denotes the patch domain (a square neighborhood of $\mathbf{0} \in \mathbb{Z}^2$ in our case).

To detect occluded regions we threshold e^f. However, setting a constant threshold that is not input sensitive might lead to undesired results. For that, we set a dynamic threshold adapted to each image pair. Therefore, we pose the occlusion detection as the following statistical test:

$\mathcal{H}_0$**:** the point at $(\boldsymbol{x}, t)$ is not occluded at $t+1$

$\mathcal{H}_1$**:** the point at $(\boldsymbol{x}, t)$ is occluded at $t+1$.

As the statistic for the test we consider the patch error e^f, (we assume patch errors at different pixels to be independent). We estimate a statistical model for $\mathcal{H}_0$ by constructing a histogram of patch errors using the unlabeled points $S_t^f \setminus C_t^f$.

Based on this histogram, we compute a threshold $\tau_t^f > 0$ by fixing the false alarm rate $\alpha \in [0,1]$, *i.e.* τ_t^f such that $P(e^f > \tau_t^f | \mathcal{H}_0) < \alpha$. In this way we compute an adaptive threshold for each frame transition. Patch errors above the threshold are considered as occluded. Let us mention that this test is designed not to be tolerant to false negatives (*i.e.* occluded pixels that have passed the test and been categorized as not occluded). Because of that, it is bound to allow for some false positives (*i.e.* pixels that are not occluded being categorized as occluded). In practice this over-estimation does not affect the quality of the result.

After applying this second step, for each frame in the sequence, the occlusion detection scheme yields two sets $K_t^f, K_t^b \subset \Omega_t$ of forward and backward occluded pixels.

Figure 5.1(d) shows an example output of the above discussed second-step occlusion detection.

6 Experimental results

In this Chapter we present some experimental results showing the behaviour of the presented models in practice. Throughout this Chapter, the conversation will be mainly based on the 2^{nd}-order model presented above. When showing the experiments, some results obtained with both models will be shown and compared in order to provide a more qualitative assessment on the behaviour of both models. Many aspects are shared by both models: from assuming the knowledge of the motion field (approximated by the optical flow), the discretization of the operators and the use of the DSCD to the different application settings presented in Section 2.2.1. The 2^{nd}-order model have tackled some drawbacks and limitations of the 1^{st}-order one, however; it still retains some of them. For that, in this Chapter we will also discuss the limitations of the 2^{nd}-order model and propose ways to address them. As an application, we will consider replacing the texture of an object's surface throughout a video sequence. We will distinguish the two application settings discussed in Section 2.2.1: the *one-lid* and *two-lid* settings. These differ only in the choice of the boundary conditions.

6.1 Experimental setup

Let us first describe some elements of the experimental setup.

Processing color videos. In our experiments, a color video $\mathbf{u} : \Omega \times \{0, 1, \dots, T\} \to \mathbb{R}^3$ is treated channel by channel, each as an independent scalar video. For the 2^{nd}-order model, this amounts to the minimization of the following energy

$$E^{\text{color}}_{\beta,\lambda}(\mathbf{u}) = \sum_{i=1}^{3} E_{\beta,\lambda}(u_i), \tag{6.1}$$

where $E_{\beta,\lambda}$ is defined for a scalar video in Eq. (4.7) and u_i, $i = 1, 2, 3$ are the color channels. Analogously, one can define a color energy for the 1^{st}-order model. Note that, although the processing of each channel is done independently, the same optical flow (and thus the same operators) is used for all channels. We use the *RGB* color space, but any other color space could be used as well.

Optical flow. In all the sequences used, we impose no restriction on the movement of the camera nor the movement of the objects in the scene. For the computation of the optical flow, except when otherwise specified, we use the algorithm described in [SRB10] and we use the code provided by the authors through the webpage [SRB]. We also use the default parameters provided with the code.

Parameters of the model. The results are obtained by minimizing the energy in Eq. (6.1). Except when otherwise stated, we use $\beta = 0.05$, $p = 2$, $\lambda = 0$ when no occlusions occur and $\lambda = 0.02$ otherwise. The minimization is done with the conjugate gradient algorithm. Let us mention that the combination of odd and even DSCDs with $\beta = 0.05$ removes most high frequency artifacts caused by the sharpening steps. However, for some frames in certain sequences, we have noticed that some high frequency artifacts remain mildly apparent. For that, we apply a linear filter to the output sequence removing very high frequencies.

Editing domain. In a practical editing application, it is important to automate the computation of the editing domain. In the present context, this amounts to an approximate tracking of the portion of the surface that the user wants to edit. In the experiments shown in this section we used different approaches for computing the editing domain, to highlight the flexibility of the model on this issue. For the experiments shown in Figures 6.2, 6.10, 6.13 the editing domain was determined by tracking the edited surface. In Figures 6.1, 6.7 and 6.8, the edited surface has been tracked as well, then manually distorted to simulate big tracking errors. Finally, for the rest of the experiments we used a big rectangular domain.

For tracking the edited surface any tracking algorithm can be used. Since we are given an optical flow, the problem of tracking a certain object amounts to propagating a binary mask of the object, specified at a lid (or the lids), along optical flow trajectories. In our experiments we performed this propagation using the proposed model. The output corresponds to a mask tracking the edited surface along the video, which gives the editing domain. Note that for the particular case of a one-lid propagation of a binary mask, the minimum of the proposed energy can be computed efficiently by a frame by frame propagation. The reason is that this problem has compatible boundary conditions, thus the minimum is attained with zero energy. In a two-lid setting (for instance in the presence of an occlusion) the solution can be approximated by two frame by frame propagations, the first propagating the first lid forward, and the second, the last lid backward. These can be combined by a point-wise maximum.

In what follows we first present four experiments for the one-lid setting, then we present three more experiments for the two-lid setting. We then comment briefly on the behaviour of different optical flow algorithms in the context of this work by experimenting with several of them proposed in the literature. Finally we discuss some practical limitations of the 2^{nd}-order model and we offer ways to overcome some of them. For every experiment below we show the original video sequence, the input to our model where the editing domain has been painted in red indicating the absence of data, and the output video after minimizing (6.1). Due to the difficulty of actually showing all frames processed, we only show a few snapshots from the processed sequence. For the full video sequences used and the results obtained, we refer the reader to the following webpage [SFAC].

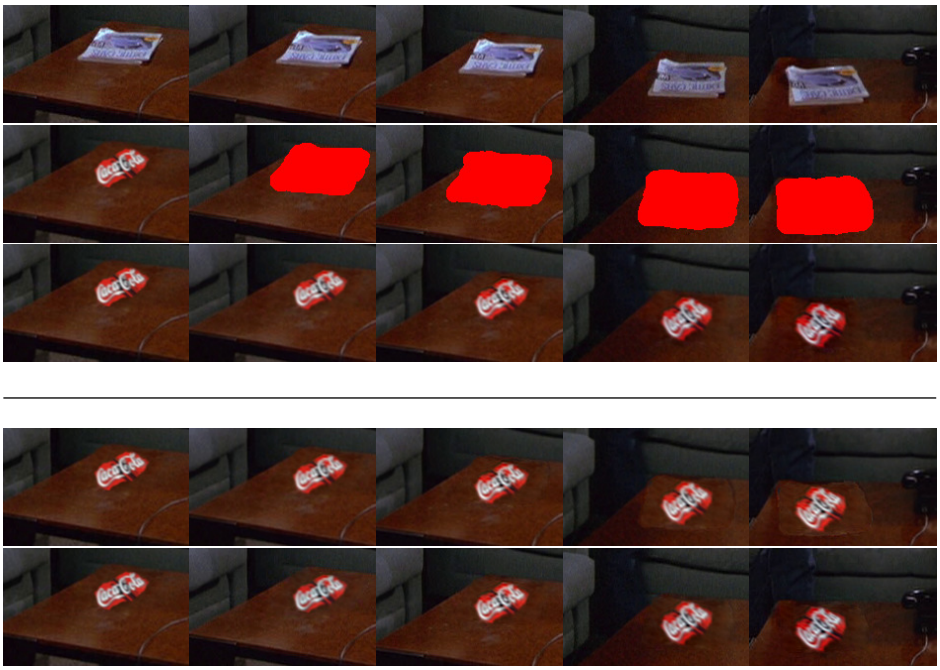


Figure 6.1: The first row shows the original sequence. A person is moving casting a shadow on the table while a light reflection is also cast on the table. The second row shows the input sequence where the editing domain has been marked in red. The third row shows the output of the 2^{nd}-order model. Rows four and five show respectively the output of the first and second step of the 1^{st}-order model with $p = 2$. From left to right, the frames shown correspond to $t = 0, 25, 50, 75$ and 100 respectively.

6.2 One-lid setting

In this setting, the editing is performed on the first frame. The edited first frame is then set as a Dirichlet boundary condition and, by minimizing energy (6.1), we obtain the output video with the editing propagated along the remaining frames. We present four experiments.

In the first one, the video contains a newspaper placed on a table. After several frames, a moving light starts illuminating the newspaper and a shadow is cast by a moving person. We replace the newspaper in the first frame of the sequence by a logo and we minimize (6.1). The result shown in Figure 6.1 demonstrates how our model handles this complex illumination change. The total length of the sequence is 106 frames with around $5 \cdot 10^5$ variables inside the editing domain. Figure 6.1 also shows the result obtained with the 1^{st}-order model for $p = 2$. The result of the 1^{st}-order model for $p = 1$ is similar.

The second experiment involves a video of a newspaper placed on a table

Figure 6.2: The first row shows the original sequence. A newspaper is filmed while the camera moves and the light in the room is being dimmed. The second row shows the input sequence after editing the first frame and where the editing domain has been marked in red. The third row shows the output of our method. Note how the resulting video accommodates this fast and sudden illumination change along time. For comparison we show results obtained using the two-step procedure of the 1^{st}-order model. In the fourth row the output of the first step: it is temporally consistent, but not spatially consistent. The fifth row shows the final result after the second step. The spatial discontinuity around the editing domain has been removed, but note how the new GBC model integrates the illumination change better. From left to right, the frames shown correspond to $t = 0, 13, 18, 24$ and 29 respectively.

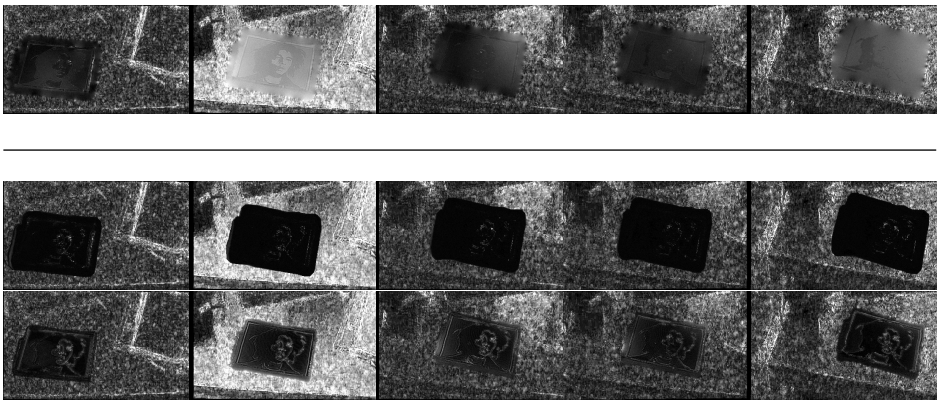


Figure 6.3: Magnitude of the illumination change rate (measured as the norm of the convective derivative) for the results shown in Figure 6.2. The first row shows the norm of the convective derivative for the GBC model (third row in 6.2). Last two rows show the norm of the convective derivative for the two-step procedure of the 1^{st}-order model. The second row corresponds to the output of the first step (fourth row in 6.2), and the third row to the result after the second step (fifth row in 6.2). From left to right, the frames shown correspond to $t = 1, 11, 17, 18$ and 29 respectively (note that the shown frames differ from those shown in Figure 6.2).

and the light in the room gets dimmed until its off. This causes a considerably large and fast global illumination change. In particular, notice that due to the change in the illumination in the room, there is a change in the dominant color: the colors shift towards blue as the light gets dimmed. We place a poster on top of the newspaper and minimize energy (6.1). Figure 6.2 shows the result. It can be seen that this large and sudden illumination change is handled by the GBC model. The total number of frames of the sequence processed in this experiment is 30 with approximately $9 \cdot 10^5$ variables inside the editing domain.

For comparison purposes, we show the result obtained with the 1^{st}-order model which uses a temporal consistency model based on the brightness constancy assumption. As we have already discussed, using that model needs a two step process (similar to [BZS+07]). The first step propagates the information from the lid using the brightness constancy model and the second step solves a Poisson editing problem for each frame which takes care of the spatial consistency of the editing. For the first step, we show the sequence obtained in the fourth row of Figure 6.2. The inability of the brightness constancy model to adapt to the illumination change in the scene causes spatial discontinuities around the editing domain. To remove them the second step is performed. The gradient of the first step's result is used as guiding vector field for the Poisson problems. This second step is aimed to adapt each edited frame to its spatial context by means of the boundary conditions of the Poisson equation, integrat-

ing the gradients of the propagated first lid. Thus, we can interpret this two-step procedure as an implementation of the gradient-constancy assumption. Solving a Poisson problem for each frame independently generates a flickering artifact in the resulting sequence. To avoid it, the brightness constancy model is used as a temporal regularizer with a low weight. The resulting sequence of the second step is shown in the fifth row of Figure 6.2.

This two-step procedure achieves a spatially and temporally consistent editing. However, notice that the result of the GBC model integrates much better the illumination change into the editing (in particular notice that the colors in the editing domain shift gradually towards blue, in accordance with the change in the dominant color of the scene). The main reason for this, is the brightness-constancy-based temporal regularization term added to the second step. Even if it has a low weight, this regularizer causes a slow reaction to fast illumination changes. This can be better appreciated in Figure 6.3, where we show the norm of the convective derivative for the results of both the 1^{st}- and 2^{nd}-order models. The norm of the convective derivative measures the illumination change rate. High values (shown in white in Figure 6.3) denote that an illumination change is happening. The result of the GBC model smoothly interpolates the illumination change rate at the boundary of the editing domain, resulting in a better integration of the editing with the surrounding. The result after the first step of the 1^{st}-order model has an almost zero illumination change rate, as expected from the brightness constancy model. This is corrected to some extent after the second step, but still the result has a limited capability to adapt for high illumination change rates.

In the third experiment we show a piece of cloth which exhibits a “wave" like movement. We edit the first frame and we minimize (6.1). Figure 6.4 shows the result. The total number of frames processed in this sequence is 20 with approximately $10 \cdot 10^5$ variables inside the editing domain. Note how the deformation of the inserted image follows the deformation of the cloth. For this sequence the optical flow was computed using a multi-scale Horn-Schunck optical flow algorithm [MLS12].

In the last experiment we show for the one-lid setting, we consider a sequence taken from [LFAW08] which is available through the webpage [LFAW]. A video of a box and of a cylindrical can is shot. Throughout the sequence, the box and the can are occluding the background before finally interacting when the can occludes the box. We edit the first frame by modifying the textures on both objects and minimize (6.1). Figure 6.5 shows the result. Note that the editing domain includes large portions that do not belong to the edited surfaces. These places should keep the texture they had in the original video. The result shows that the model has been able to reconstruct the original textures seamlessly. This demonstrates that a precise tracking of the edited surfaces is not required. The total number of frames in the sequence is 13 with $9 \cdot 10^5$ variables inside the editing domain. The results shown are snapshots taken every two or three frames.

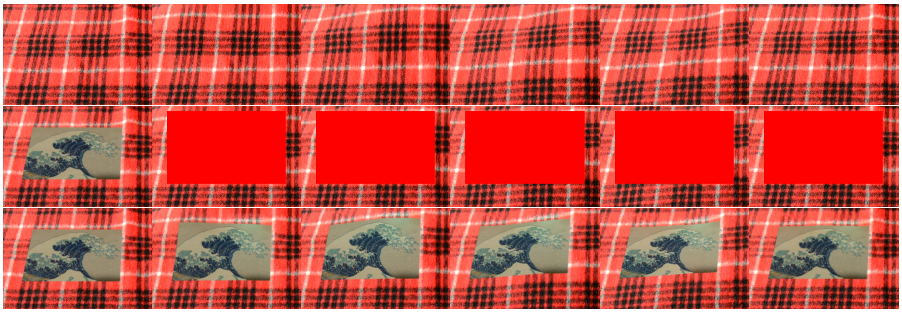


Figure 6.4: The first row shows the original sequence of a cloth exhibiting a "wave" like motion. The second row shows the input sequence after editing the first frame and where the editing domain has been marked in red. The third row shows the output after minimizing (6.1). Note how the editing done accommodates to the movement of the cloth in the resulting video. From left to right, the frames shown correspond to $t = 0, 7, 8, 9, 10$ and 20 respectively.

Figure 6.5: The first row shows the original sequence. A box and a cylindrical can are being filmed while the camera moves. The second row shows the input sequence after editing the first frame and where the editing domain has been marked in red. The third row shows the output after minimizing (6.1). From left to right, the frames shown correspond to $t = 0, 2, 5, 9$ and 12 respectively.

6.3 Two-lid setting

In this setting, the editing is performed on the first and last frames. These are set as Dirichlet boundary conditions and, by minimizing energy (6.1), we obtain the output video with the editing propagated into the remaining frames. Basically, the result is a smooth interpolation between the two edited frames along the trajectories of the optical flow.

The first experiment is shown in Figure 6.6, and uses the same sequence of Figure 6.5. We edit the first frame as in Figure 6.5, by introducing the yellow text on the blue box, and the formula on the can. For the last frame, we only change the color of the formula on the can, from yellow to red. No text is added on the box. In the resulting sequence, the formula and the text move coherently with the can and the box respectively. While moving, the formula changes its color smoothly from yellow to red, and the text in the box gradually vanishes. This demonstrates the interpolation between both lids along the optical flow trajectories. Note that the second “i" in the text “rishi" does not vanish as expected. The reason is that this part of the box is occluded in the last frame. In that specific location, the solution behaves as in a one-lid setting, propagating the information from the first frame only.

The inconsistent editing of the lids in this experiment gives a good insight on the working of the model. It puts in evidence the differences in the behaviour of trajectories reaching both lids and those reaching only one lid. For trajectories that reach both lids, the result is a smooth interpolation between the information present at those lids. This can be seen clearly from the smooth transition of the formula’s color from yellow to red and the smooth vanishing of the “rish” text. On the other hand, for trajectories that only reach a single lid (for instance due to an occlusion) the problem becomes a one-lid problem and the information will be transported from that lid only. This is what actually happens with the second “i" in the text “rishi": these trajectories do not reach the second lid, and therefore the “i" is being transported from the first lid. As a consequence, if the purpose of the editing is to perform a blending between two lids edited in a non-consistent manner, it is imperative that every trajectory in the editing domain reaches both lids.

When the application is to edit an object’s surface with a non-changing texture, the editing in both lids has to be consistent. This way, there will be no appreciable differences between one-lid trajectories propagating data from one of the lids, and two-lid trajectories blending data from both lids. In this context, consistent editing means consistency with the motion in the scene, and consistency with the overall change in illumination from the first lid to the last. The motion consistency implies that the editing in the second lid corresponds to the warping of the editing in the first lid according to its motion. The consistency with the illumination change implies that the editing in the second lid suffers approximately the same (additive) illumination change as its surrounding with respect to the first lid. In practice, inconsistencies in the editing of both lids are tolerable.

The second experiment, shown in Figure 6.7 depicts a computer screen which

Figure 6.6: The first row shows the editing done in the first and last frames and the domain has been marked in red. The second row shows the output after minimizing (6.1). Note how the result is a smooth interpolation between the two lids. Let us mention that the original sequence of the experiment is the first row of Figure 6.5. From left to right, the frames shown correspond to $t = 0, 2, 5, 9$ and 12 respectively.

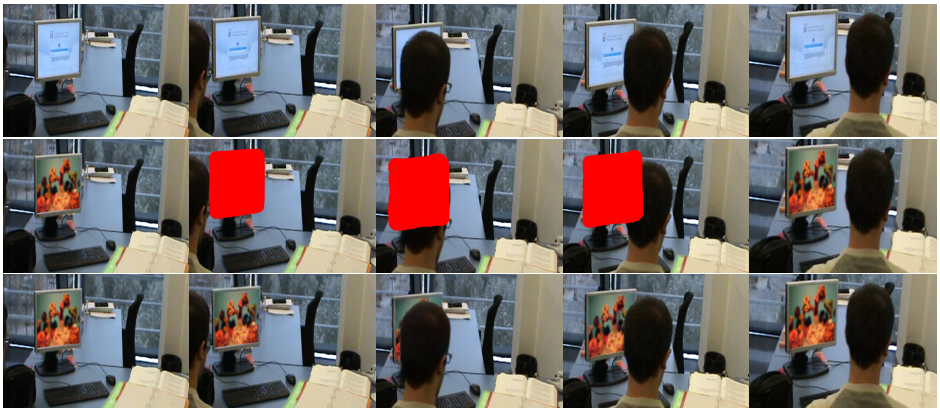


Figure 6.7: The first row shows the original sequence. A screen is filmed when a person sitting on a chair moves in front of it. The second row shows the input sequence after editing the first and last frames and where the editing domain has been marked in red. The third row shows the output. From left to right, the frames shown correspond to $t = 0, 4, 12, 16$ and 19 respectively.

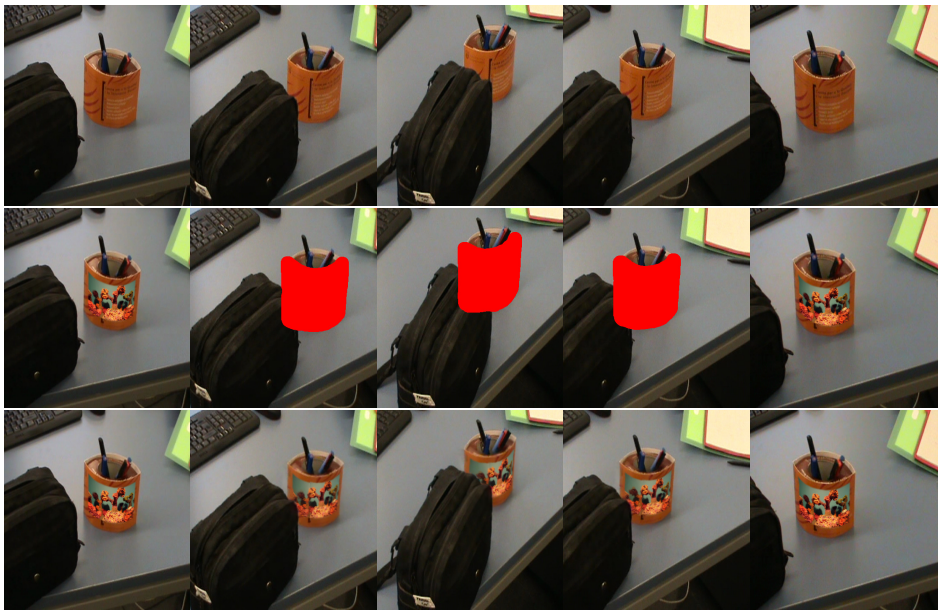


Figure 6.8: The first row shows the original sequence. A pen-holder is filmed while it gets occluded by black bag. The second row shows the input sequence after editing the first and last frames and where the editing domain has been marked in red. The third row shows the output after minimizing (6.1). From left to right, the frames shown coorespond to $t = 0, 16, 26, 36$ and 42 respectively.

gets almost completely occluded, and then dis-occluded, by a person moving in front of it. We replace the image on the screen by an image of corals, both in the first and the last frames. The sequence consists of 20 frames with more than $4 \cdot 10^5$ variables in the editing domain.

Finally, in the third experiment, we edit a pen-holder lying on a table. The pen-holder gets partially occluded then dis-occluded by a black bag. We edit the curved surface of the pen-holder in both the first and last frames and place the image of the corals on it. The total number of frames of this sequence is 43 with around 10^6 variables in the editing domain. Figure 6.8 shows the result obtained after minimizing (6.1). For this experiment we found better results with the addition of a temporal regularization term as mentioned at the end of Section 4.3 with a weight $\gamma = 0.02$.

6.4 Discussion on the limitations of the proposed method

In this Section we would like to discuss, with some practical details, the limitations of the proposed method and propose ways to overcome some of them.

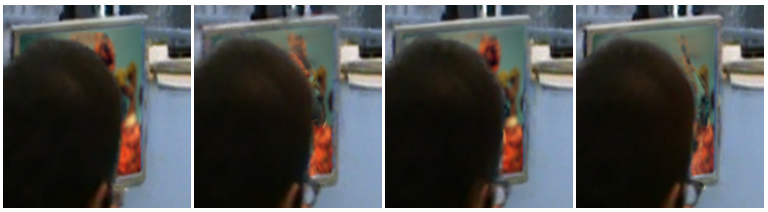


Figure 6.9: Experiment showing the behaviour of the model using four different optical flow algorithms. We show only one frame for each result. The complete result sequences can be found in [SFAC]. From left to right, the images correspond to using the optical flow algorithm presented in [SRB10], [CP11], [ARS11] and [BM11] respectively.

Optical flow related. It is well known that optical flow algorithms compute *apparent motion*, not the true motion of the scene. This implies that in some circumstances, as in the presence of moving shadows, the optical flow may not give a correct estimation of the true motion. If the apparent motion and the true motion coincide then using the optical flow gives very good results. If not, relying on apparent motion will likely give rise to visual artifacts. Confronting these situations is a current trend of research in the optical flow community, for example by incorporating gradient-based terms for handling illumination changes, or segmenting the scene to improve the occlusion handling. These improvements will in turn expand the applicability and performance of the presented models. To highlight the dependence on the optical flow, we show two experiments.

In the first one we test four optical flow algorithms from the literature, applying them on the two-lid editing problem shown in Figure 6.7. We used the following optical flow algorithms: the large displacement optical flow algorithm of Brox and Malik [BM11], the TV-L1 optical flow of Chambolle and Pock [CP11], the optical flow with sparse occlusion detection of Ayvaci *et al*. [ARS11] and finally the layer-based algorithm of Sun *et al*. [SRB10]. For all optical flows, we use the code provided by the authors with the default parameters. For the optical flow algorithms presented in [BM11], [CP11] and [SRB10], since no occlusion masks are given by the algorithms, we use the occlusion detection method described in Appendix 5. The optical flow of [ARS11] provides occlusion masks and we use them (after a dilation of two pixels) to handle occlusions. Figure 6.9 shows one frame from the result of the 2^{nd}-order model using the above discussed optical flow algorithms. We have chosen a frame where the differences between the different algorithms is apparent. Let us recall that the complete set of results can be found in [SFAC]. For all tested optical flow algorithms, the model behaves similarly and is able to handle the occlusion. The differences in the results are due to the differences in the motion perceived by each optical flow method. For example some optical flows show a "dragging" effect: parts of the screen that are about to be occluded by the head seem to be dragged by the head instead of being occluded. The reason is that the optical flow algorithm has assigned the movement of the head to these parts of the screen. It might

Figure 6.10: The first row shows the original sequence. A person wearing a shirt moves by bending down and straightening back up. The second row shows the input sequence after editing the first frame and where the editing domain has been marked in red. The third row shows the output after minimizing (6.1). From left to right, the frames shown correspond to $t = 0, 9, 19, 26, 33, 38, 41$ and 44 respectively.

also be interesting to test the proposed model with other optical flow methods which incorporate the temporal consistency in the computation of the optical flow [SS07, SSB12, VBVZ11]. It has been reported that considering a larger number of frames improves on the accuracy of the optical flow.

In the second experiment, in Figure 6.10 we show a different kind of artifacts. In this experiment, we edit a part of a shirt being worn by a person by adding to it the “UPF" logo. The person bends over and then straightens back up. While straightening up, the shirt suffers severe deformations. At that point, inaccuracies in the optical flow result in unnatural distortions of the transported texture (second half of the sequence). However, let us note that the model correctly handles the local illumination change caused by the shadow cast by the shirt on itself while bending. The result was obtained by minimizing (6.1). The total number of frames of this sequence is 35 with around $5 \cdot 10^5$ variables in the editing domain.

Multiple occlusions. So far we have covered the following basic situations:

a) A single occlusion or dis-occlusion occurs. This can be treated as a one-lid setting, for an occlusion the lid is placed at $t = 0$, and for a dis-occlusion the lid is at $t = T$.

b) An occlusion followed by a dis-occlusion occurs. This can be handled by a two-lid setting, where the lids are at $t = 0$ and $t = T$.

Cases with multiple occlusions and dis-occlusions of the edited surface can be handled by temporally splitting the editing domain into temporal segments that fall into the above a) and b) basic cases. The splitting of the editing domain amounts to adding lids at the splitting frames. To illustrate this we consider a sequence taken from [LFAW08], available online through the webpage [LFAW]. Figure 6.11 shows an experiment with multiple occlusions of the edited surface.

A hand is moving back and forth, repeatedly dis-occluding and occluding a disk-shaped object in the background. The Figure shows first the result obtained by setting a lid at frame $t = 8$, after the first dis-occlusion. This splits the sequence into two one-lid problems: one from $t = 0$ to $t = 8$, and another from $t = 8$ to $t = 44$ (the last frame). In the second segment there is still one occlusion followed by a dis-occlusion and yet another occlusion towards the end. No trajectories from the lid reach the region that gets dis-occluded, and we can see what sort of artifacts one expects to see in regions where no information from the lid is arriving. As shown in the Figure, this can be corrected by further splitting the sequence and adding an intermediate lid before the last occlusion starts. Let us also mention that the occlusions in this experiment have been detected using the procedure described in Appendix 5.

In summary, the rule is to ensure that all trajectories inside the editing domain reach at least one Dirichlet boundary condition.

Big zooms and tilts. Consider the case of a one-lid setting where the editing has been performed on the first frame and a camera is doing a big zoom-in on the edited object. The resolution of the object increases considerably with time. The method will propagate the low resolution information given at the lid and will not recover a higher resolution version of the propagated information. The resulting propagation will be blurred. However, this could be solved by editing the last high resolution frame and propagate that editing backwards. Figure 6.12 shows an example.

Consider now the case where a big zoom-out is followed by a big zoom-in. This problem can be solved in a two-lid setting where both lids are at high resolution. Figure 6.13 shows an example where a cloth is being laid down on a table and then taken back to its original position. This simulates the just discussed example with an added tilt transformation as well. We show two results in a one-lid setting with different interpolation schemes (bi-linear and bi-cubic) and one result in a two-lid setting. Notice how in the second half of the sequence, when zooming-in, the one-lid result presents considerable blur with the bi-linear interpolation, particularly at the last frames where the resolution increases significantly. Using the bi-cubic interpolation, the result is much sharper but it still suffers from blur artifacts. Adding a second lid at the last frame solves this issue. Let us also note that a considerable illumination change also occurs in the sequence and the method deals with it seamlessly. The result was obtained by minimizing (6.1). The total number of frames of this sequence is 53 with around $5 \cdot 10^5$ variables in the editing domain.

Let us note that this limitation comes as a consequence of propagating a tex-

Figure 6.11: The first row shows the original sequence. A hand is moving back and forth, dis-occluding and occluding a disk-shaped object repeatedly in the background. From left to right, the frames shown correspond to $t = 0, 8, 20, 26, 30, 33, 37$ and 44 respectively. The second row shows the input sequence after editing frame 8 by removing the disk-shaped object and setting it as a lid. It also shows the remaining frames and the editing domain marked in red. The third row shows the output after removing the object in frame 8 and propagating the modification back to frame 0, and forward until frame 44 by minimizing (6.1) in a one-lid setting. Note that the first dis-occlusion (from 0 to 8) is correctly handled as well as the first occlusion (from 8 to 20). But, as a new dis-occlusion starts around frame 26, no information is reaching this area from the lid (frame 8) and it appears as if the hand spills its color into this area. This double occlusion could be handled by editing the frame 33 and setting it as a second lid, as shown in fourth row. In this experiment, the occlusions have been detected using the procedure described in Section 5.

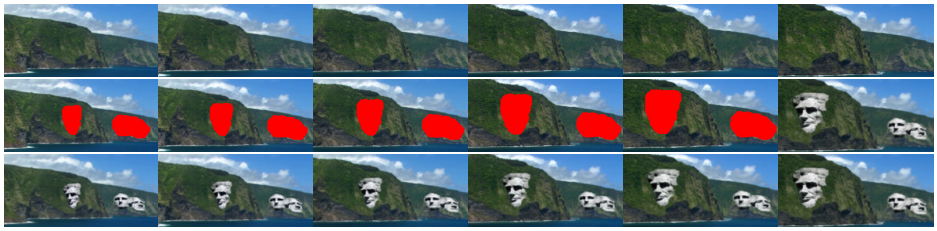


Figure 6.12: An experiment simulating the case of a zoom in. The first row shows the original sequence. The second row shows the input sequence after editing the last frame and where the editing domain has been marked in red. The third row shows the output after minimizing Eq. (6.1) in a one-lid setting. Notice that we have edited the lid high-resolution to obtain a non-blurry result. From left to right, the frames shown correspond to $t = 1, 13, 23, 33, 40$ and 46 respectively.

ture using correspondences between adjacent frames. Other methods such as [BGD+10, ST08, RAKRF08] establish a transform between an input texture and all other frames in the sequence. The computation of these transforms is not trivial. In the case of [RAKRF08], the method relies on an accurate video segmentation, keypoint tracking and non-convex optimization in order to compute the mappings. On the other hand, and more related to this work, [BGD+10, ST08] integrate the optical flow to compute a set of trajectories covering the editing domain. This requires dealing with complexities inherent to the explicit management and computation of trajectories. In any case, these methods need post-processing steps in order to deal with illumination changes and filling-in holes that are not covered by the mapping (for instance due to occlusions). In our approach trajectories are dealt with implicitly, illumination correction is intrinsic to the model, and the filling-in of small holes caused by occlusions is taken care by the regularization term in the energy. This is attained by a single convex minimization process.

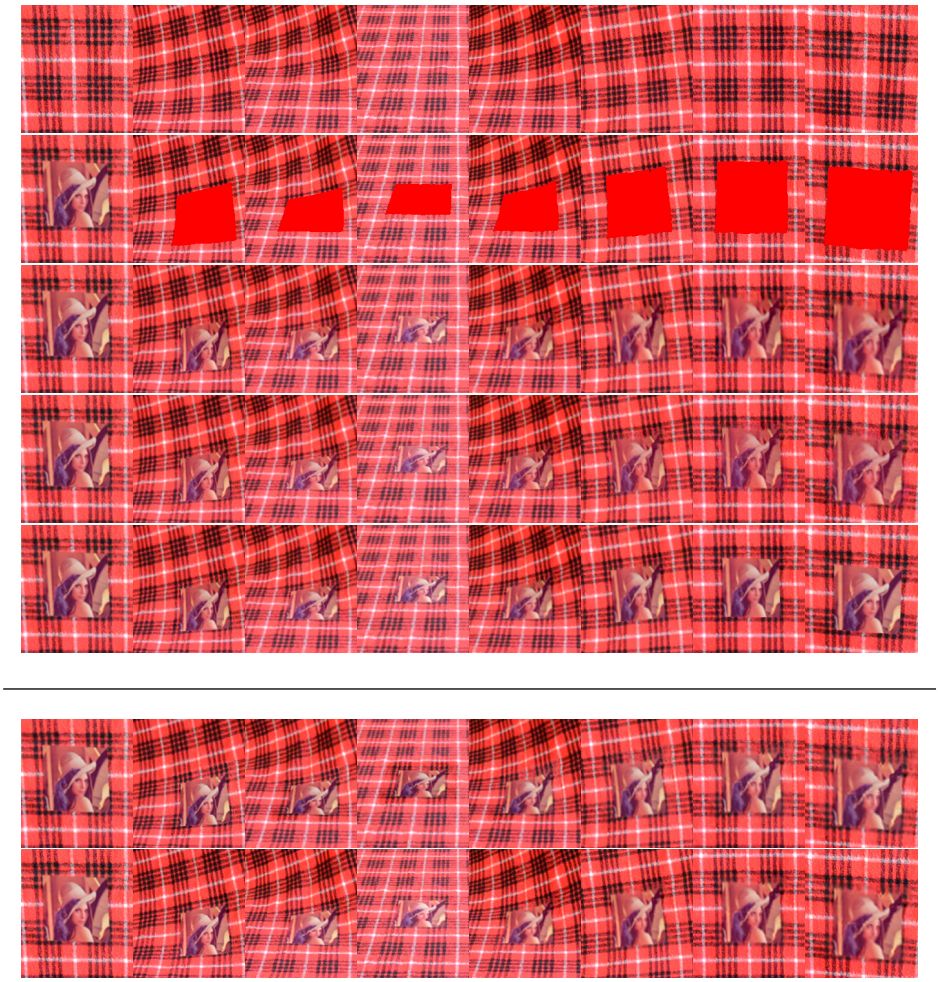


Figure 6.13: An experiment simulating the case of a zoom out with tilt followed by a zoom in and tilting back to the original position. The first row shows the original sequence. The second row shows the input sequence after editing the first frame and where the editing domain has been marked in red. The third row shows the output after minimizing Eq. (6.1) in a one-lid setting. The artifacts are clear when coming back from low-resolution to high resolution. The fourth row shows the output when using the bi-cubic interpolation scheme in a one-lid setting as well. It can be noticed that the artifacts diminished but they are still visible. The last row shows the output in a two-lid setting where the artifacts have been dealt with. From left to right, the frames shown correspond to $t = 0, 16, 19, 26, 32, 39, 44$ and 53 respectively.

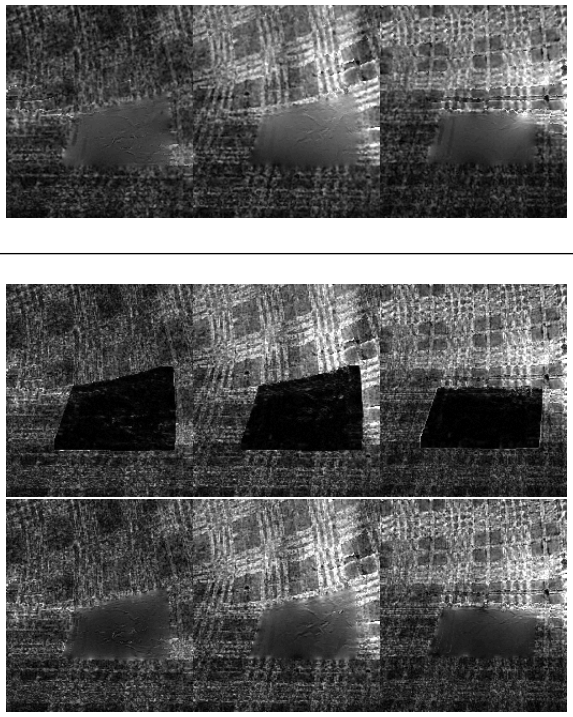


Figure 6.14: Magnitude of the illumination change rate (measured as the norm of the convective derivative) for the results shown in Figure 6.13. The first row shows the norm of the convective derivative for the GBC model (third row in 6.13). The second and third rows show the norm of the convective derivative for the two-step procedure of the 1^{st}-order model. The second row corresponds to the output of the first step (sixth row in 6.13), and the third row to the result after the second step (seventh row in 6.13). From left to right, the frames shown correspond to $t = 22, 23$ and 24 respectively (note that the shown frames differ from those shown in Figure 6.13). We would like to note that the intensities of the images have been saturated (in the same manner) in order to show better the contrasts.

7 A note on frame interpolation

In this Chapter we propose an interpolation method that produces a sequence of plausible intermediate frames between two input images. The main feature of the proposed method is the handling of occlusions using a time coherent video segmentation into spatio-temporal regions. Occlusions and dis-occlusions are defined as points in a frame where a region ends or starts, respectively. Out of these points, forward and backward motion fields are used to interpolate the intermediate frames. After motion-based interpolation, there may still be some holes which are filled using a hole filling algorithm.

7.1 Overview

Our purpose in this Chapter is to propose an interpolation method to produce a sequence of plausible intermediate images between two input images. We consider the applications to slow camera motion for smooth playback of lower frame rate video, smooth view interpolation and animation of still images.

Given two frames extracted from a certain video sequence, the problem is to generate as many intermediate frames as needed so that a slow motion effect occurs when played back at a normal frame rate.

Recent progress in optical flow estimation [BBPW04, BM11, SRB10] provides optical flows of sufficient quality for intermediate frame interpolation One of the main difficulties that has to be tackled in frame interpolation is the occlusion effects. Points visible at time t that get occluded at time $t+1$ should not have a corresponding point at frame $t+1$. Similarly, points that appear at time $t+1$ should have no correspondent at time t. Frame interpolation algorithms have to detect such occlusions in order to correctly decide how to interpolate.

Most current optical flow estimation methods do not directly compute occlusions and assign an optical flow to each pixel of each frame. If we blindly use the optical flow, artifacts are generated. They are specially visible at moving occlusion boundaries. This difficulty is usually handled by analyzing the forward and backward motion fields in order to decide from which image we interpolate. In this chapter we propose to address the occlusion effects by using a spatio-temporal segmentation of the video sequence. This segmentation allows us to interpret the sequence as a set of spatio-temporal regions whose temporal boundaries give us information about their “birth" and “death". As a result, we are able to extract a set of candidate occluding and dis-occluding points. Forward and backward motion fields are then used to interpolate the intermediate frame taking into account the latter points. After motion-based interpolation, there may still be some holes which are filled using a suitable hole filling algorithm [CPT04b, AFCS11].

7.1.1 Previous work

There are many works in the literature devoted to intermediate frame interpolation. Many of them are based on establishing correspondences between consecutive pairs of images. They are usually computed by means of block based motion estimation or dense optical flow. The former has been applied to frame-rate conversion for digital television. The image is divided into a set of non-overlapping blocks and forward and/or backward motion compensation is performed to create the interpolated frame [CRT90]. Occlusion effects are handled by analyzing the forward and backward motion fields in order to decide from which frame to interpolate. In [DN04] the authors conclude that the averaging technique is appropriate if forward and backward prediction errors are equal. In other cases, they conclude with theoretical results that the filter taps associated to the motion compensation have to be adapted to the reliability of the motion vectors.

Frame interpolation also has been tackled by means of dense optical flow. Most of current optical flow approaches compute the forward motion field based on the work of Horn and Schunk (see [BSL+11]). Other authors compute a symmetric optical flow by means of a penalty term in their model [ADPS07, IK08] which permits to define occlusion regions.

In the context of frame interpolation, [BSL+11] propose a simple method for frame interpolation and occlusion handling based on forward *splatting* which only uses the forward flow. In [HSB09] the authors enhance the latter approach by means of forward and backward splatting. Occlusion is detected by assessing the flow consistency of the forward and backward flows at the intermediate frame. Holes are treated by extending, at the intermediate frame, the motion vectors of neighboring pixels using a Markov Random Field (MRF). [LLM10] presents a long-range correspondence estimation technique that includes SIFT, edge and symmetry data terms. Occlusion also is detected by assessing the coherence between the two flows. The interpolated image is created by using a graph-cut based approach that decides at each pixel from which image the color information is taken.

In [MHM+09] the authors present a method based on graph cuts which is based on the idea that a given pixel traces out a path in the source images. Their method can be viewed as an inverse optical flow algorithm: they compute the location of a given intermediate pixel in the input images. Occlusion effects are again tackled by assessing the consistency between the forward and backward flows. The advantage of their method is that no holes are created and thus they need no treatment for them.

7.1.2 Summary of the proposed method

We briefly summarize the main Steps of our algorithm. Let $I(x, y, t)$ be a video sequence. For simplicity we consider that the video is sampled at times $t = 0, 1, 2, \ldots$. The image domain Ω is a rectangular grid in $\mathbb{Z}^2$.

Step 1 : We compute the forward (u, v) and backward (u^b, v^b) optical flows. For that we use the algorithm [BM11] (see Section 7.2.1). As an alternative, we

can also use [SRB10]. Both estimate a forward flow between images t and $t+1$ and have no occlusion treatment. The binary executables for both algorithms are publicly available.

Step 2 : Using the forward flow we compute a time consistent segmentation of the video sequence (see Section 7.2.2).

Step 3 : Potential occlusions are then derived from Step 2. We then interpolate the intermediate frames using the idea of forward and backward splatting described in [HSB09] (see Section 7.2.3). The result may contain holes made of points which could not be interpolated.

Step 4 : The holes are filled-in using an inpainting strategy (see Section 7.2.4).

7.2 Algorithm Description

7.2.1 Optical Flow Computation

For the computation of the optical flow we can use any optical flow producing good results. Our experiments have been done with the optical flow algorithm proposed in [BM11] which uses HOG (Histogram of Oriented Gradients) descriptors in order to be able to follow fast motions. Estimated motion vectors are required to follow descriptor matchings. As an alternative, we can also use [SRB10].

7.2.2 Time Coherent Video Segmentation

As stated previously, the objective is to create a time consistent segmentation of the video sequence. We consider the video sequence as a volume of 3D data. Our segmentation is based on the simplified Mumford-Shah model. Given the video sequence $I(x,y,t)$, the simplified Mumford-Shah model approximates I by a piecewise constant function $\tilde{I}(x,y,t) = \sum_{O\in\mathcal{P}} m_O \chi_O(x,y,t)$ that minimizes the energy

$$\sum_{t=0}^{N} \sum_{x\in\Omega} (I(x,y,t) - \tilde{I}(x,y,t))^2 + \lambda \sum_{O} \text{Area}(\partial O), \tag{7.1}$$

where $\lambda > 0$, $\mathcal{P}$ is a partition of $\Omega \times \{0,\dots,N\}$ into connected regions O, and *Area* corresponds to the area of the interface that separates two spatio-temporal regions. Note that the area for each interface is counted twice in Eq. (7.1), but this amounts only to a replacement of the value of the parameter λ by $\lambda/2$. As explained in [KLM94], the parameter λ controls the number of regions of the segmentation. For a given partition $\mathcal{P}$, the constant m_O is the average of $I(x,y,t)$ in the region O.

We replace the classical notion of connectivity by a time compensated one. For that we construct a graph whose nodes are the pixels of the video, i.e. $\{(x,y,t) : (x,y) \in \Omega, t \in \{0,\dots,N\}\}$. There are two types of edges in the graph: spatial and temporal ones. Spatial edges connect a pixel (x,y,t)

to its 8-neighborhood in frame t. Temporal edges are defined using the precomputed forward optical flow. If the flow vector for pixel (x, y, t) is (u, v), then we add to the graph an edge joining pixel (x, y, t) to pixel $(x', y', t') = (x + [u], y + [v], t + 1)$, where the square brackets denote the nearest integer. This graph gives us the 3D neighborhood of each point (x, y, t). This permits an easy adaptation of the algorithm in [KLM94]

For a given λ and following [KLM94], the energy is optimized with a region merging strategy that computes a 2-normal segmentation. A 2-normal segmentation is defined by the property that merging any pair of its regions increases the energy of the segmentation. Notice that 2-normal segmentations are typically not local minima of the functional; however, they are fast to compute and useful enough for our purposes. The region merging strategy consists in iteratively coarsening a given pre-segmentation, which is stored as a region-adjacency graph. Each edge of this graph is marked by the energy gain that would be obtained by merging the corresponding pair of regions into one. Then, at each step of the algorithm the optimal merge – the one that leads to the best improvement of the energy – is performed, thus reducing the region adjacency graph by one region and one or more edges. The energy gain is recomputed for the neighbouring regions and the algorithm continues to merge as long as they produce some improvement of the energy functional. Note that the parameter λ of the energy functional controls the number of regions of the resulting segmentation. When finding 2-normal segmentations by region merging, this parameter can be automatically set by specifying directly the desired number of regions.

In practice, we do not know which value of λ will produce a good segmentation. For that reason we proceed as follows: we create a set of partitions that are obtained by successively increasing λ (e.g. dyadically). Each partition is computed by taking as input the previously obtained partition and merging regions as described in the previous paragraph. The algorithm starts with a low value of λ using the time-connected graph described above, and stops when the trivial partition is obtained. The history of all the mergings is stored in a (binary) tree, whose nodes represent each region of the segmentation at some iteration. The leafs of this tree are the pixels of the input video. The internal nodes of this tree are regions appearing at some iteration, and the root of the tree is the whole video. While the tree is being built, each node is marked with the value of λ at which the corresponding region has been created.

Once this tree is built, it can be cut at any desired value of λ in real-time, to produce segmentations at different scales. We call *tubes* the spatio-temporal regions of the resulting partition (see Figure 7.1). The tubes encode a temporally coherent segmentation of the objects in the video, which can be used for several purposes (e.g. tracking). We use them here in order to determine potential occlusions by analyzing their temporal boundaries. Any connected tube O has a starting and an ending time, denoted by T_O^s and T_O^e, respectively. The section of O at time t is denoted by $O(t) = \{(x, y) \in \Omega : (x, y, t) \in O\}$. Thus O starts (resp. ends) with the spatial region $O(T_O^s)$ (resp. $O(T_O^e)$).

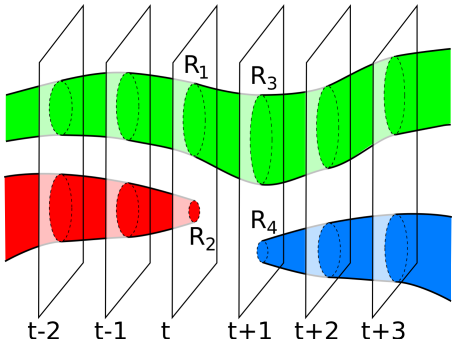


Figure 7.1: This figure illustrates the concept of tubes as we describe it here. We see a tube that ends at t, a tube that starts at $t+1$ and a tube that continues through frame t.

7.2.3 Intermediate Frame Interpolation

Given two video frames I_0 and I_1 at times $t = 0$ and $t = 1$ respectively, our purpose is to interpolate an intermediate frame at time $t = \delta \in (0,1)$. For that issue the forward and backward motion fields are first estimated. In order to deal with the occlusion effects, we use the information of the temporal boundaries obtained with our time coherent segmentation algorithm. Intuitively, a given pixel (x,y,t) at $t = 0$ is forward projected if it is not in a dying spatio-temporal region. Similarly, a given pixel (x,y,t) at $t = 1$ is backward projected if it is not in a spatio-temporal region that has been born there. Let us now go into the details of the algorithm.

The frame interpolation algorithm starts by marking all pixels to be interpolated as holes. Then two stages are performed: in the first stage a forward projection is done; in the second a backward projection is performed to (partially) fill in the holes that the first stage may have left.

For the first stage we use the forward optical flow (u,v) from $t = 0$ to $t = 1$. Let $\mathcal{F}(t = 0)$ be the set of pixels of frame $t = 0$ which do not belong to tubes that end at time $t = 0$. This information is contained in the data structure that we developed for the spatio-temporal segmentation.

Let us describe the forward projection step. It is based on the idea of splatting described in [HSB09].

For a given pixel $(x,y,t = 0)$, let $p_\delta = (x,y) + \delta(u,v)$ be the forward projected point of the pixel. Note that p_δ may be a non-integer point. Let $p_\delta^{00} = \lfloor p_\delta \rfloor$ be the pixel whose coordinates are given by the integer parts of the coordinates of p_δ, and let $p_\delta^{ab} = p_\delta^{00} + (a,b)$ for $(a,b) = (0,1),(1,0),(1,1)$, the four points bounding the square containing p_δ. We assign the flow (u,v) to each pixel p_δ^{ab} and compute its forward and backward projections: let them be $p_0^{ab} = p_\delta^{ab} - \delta(u,v)$ and $p_1^{ab} = p_\delta^{ab} + (1-\delta)(u,v)$. The values $I_0(p_0^{ab})$ and $I_1(p_1^{ab})$ are computed using bilinear interpolation.

The pixel $(x, y, t = 0)$ is forward interpolated if pixel (x, y, t) belongs to $\mathcal{F}(t = 0)$ or if

$$|I_0(p_0^{ab}) - I_1(p_1^{ab})| \leq \tau, \tag{7.2}$$

where $\tau > 0$ is a pre-specified tolerance value (in our experiments, we take the value $\tau = 10$). If the previous conditions do not hold, then we do not forward interpolate.

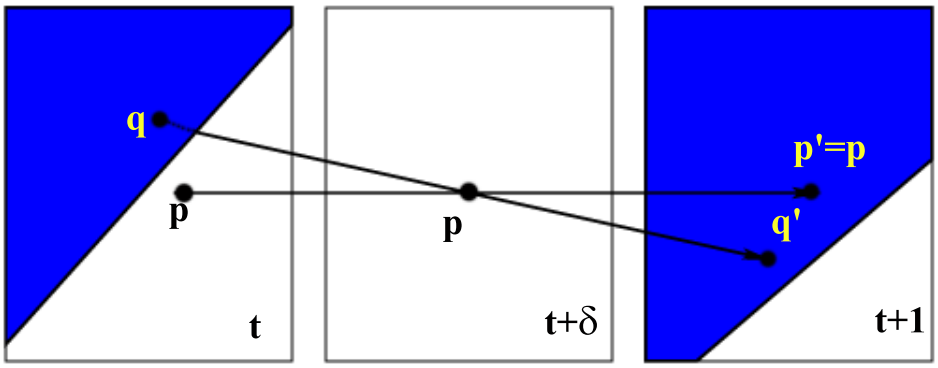


Figure 7.2: A case of inconsistent interpolation as explained in the text.

Equation (7.2) allows to deal with pixels which are part of objects that are visible in both frames I_0, I_1 (and thus belong to a continuing tube) but are occluded in frame I_1, inheriting the optical flow of the visible object as an effect of the regularization terms (see the pixel p in Figure 7.2). In that case, the correspondents of both points (p, t) and (q, t) in Figure 7.2 go through $(p, t + \delta)$ and we expect that the candidate interpolation from $I(p, t)$ and $I(p', t + 1)$ is discarded because of inconsistent gray level values, while the interpolation between $I(q, t)$ and $I(q', t + 1)$ is retained because it satisfies (7.2).

In case the pixel $(x, y, t = 0)$ can be forward interpolated, the pixels at p_δ^{ab} are computed as

$$\tilde{I}_\delta(p_\delta^{ab}) = (1 - \delta) I_0(p_0^{ab}) + \delta I_1(P_1^{ab}) \tag{7.3}$$

and marked as interpolated.

The previous process is repeated for all pixels. We scan all pixels at $t = 0$ from left to right and top to bottom. For each pixel we proceed as described above. In case multiple interpolated values are assigned to a intermediate pixel, we assign to it the average of all values $\tilde{I}_\delta(p_\delta^{ab})$ for $p_\delta^{ab} = (\bar{x}, \bar{y})$.

Once the first stage has been performed, the algorithm now uses the backward flow to deal with the holes the first step may have left. Note that all pixels labeled as interpolated (in the first stage) are not altered. The algorithm is basically the same as the one described before but in this case a backward projection is done, replacing $\mathcal{F}(t = 0)$ by $\mathcal{F}(t = 1)$. Here $\mathcal{F}(t = 1)$ is defined as the set of tubes that do not begin at time $t = 1$. This can be easily checked on the graph (which has been constructed using the forward flow).

7.2.4 Filling the Holes

Holes appear inevitably due to occlusions, dis-occlusions, or during the interpolation process due to the expansive or contractive character of the flow. Some occlusions may generate ending tubes at time t (which should not be images of the backward flow). Thus, we do not project those pixels forward for interpolation. Dis-occlusions are not images (and some of them may be starting tubes) of the forward flow and may also generate holes. To fill-in the remaining holes we use an inpainting strategy. Each pixel in a hole of I_δ is filled in by an exemplar based interpolation as in [AFCS11] (see also [CPT04b]). To fill a pixel in frame t we search for patches in the previous and next frames. To reduce the searching area we search in regions of frame t (resp. $t+1$) determined by the optical flows of pixels bounding the hole. The search of best patches can be accelerated using the Patch-Match algorithm [BSFG09].

7.3 Experimental results

Let us display some experiments. Figure 7.3 shows the interpolation between two different frames of the sequence MiniCooper, available at http://vision.middlebury.edu/flow/data/. The person is closing the door at the back of the car. The maximum displacement is 17.28. From left to right and top to bottom, the first and last images belong to the original sequence. The four intermediate frames have been interpolated. In Figure 7.4 we show the holes generated by the interpolation process. The left image shows the holes after forward interpolation, the right image shows the remaining ones after backward interpolation. Those are the ones that are filled-in by inpainting. In Middlebury an intermediate frame is given for comparison. It corresponds to the fourth image in Figure 7.3. The RMSE is in this case 4.2440.

Figure 7.5 shows the interpolation between two different frames of the sequence Foreman. The person is opening the mouth and some regions are disoccluded. The maximum displacement is 9.8331. From left to right and top to bottom, the first and last images belong to the original sequence. The four intermediate frames have been interpolated. In Figure 7.6 we show the holes generated by the interpolation process. The left image shows the holes after forward interpolation, the right image shows the remaining ones after backward interpolation which we filled-in by inpainting.

Figure 7.7 shows the interpolation of four frames from a sequence where a placard is being disoccluded. Figure 7.8 shows the interpolation of an intermediate frame in the sequence Urban taken from Middlebury. The camera moves right and there are occlusions between buildings. The experiments displayed here can be found in http://www.dtic.upf.edu/∼cballester/demos/scm .

In our experiments the computation time per frame is around: 5 sec for optical flow computation, 0.55 sec for the intermediate frame interpolation and 1 min for the inpainting algorithm.

Figure 7.3: Intermediate frame interpolation. The first and last frames are original images. The intermediate ones are interpolated.

Figure 7.4: Left: The holes appearing in the first interpolated frame of Figure 7.3 after forward interpolation. Right: The remaining holes after backward interpolation.

Figure 7.5: Intermediate frame interpolation. The first and last frames are original images. The intermediate ones are interpolated.

Figure 7.6: Left: The holes appearing in the last interpolated frame of Figure 7.3 after forward interpolation. Right: The remaining holes after backward interpolation.

Figure 7.7: Intermediate frame interpolation. The first and last frames are original images. The intermediate ones are interpolated.

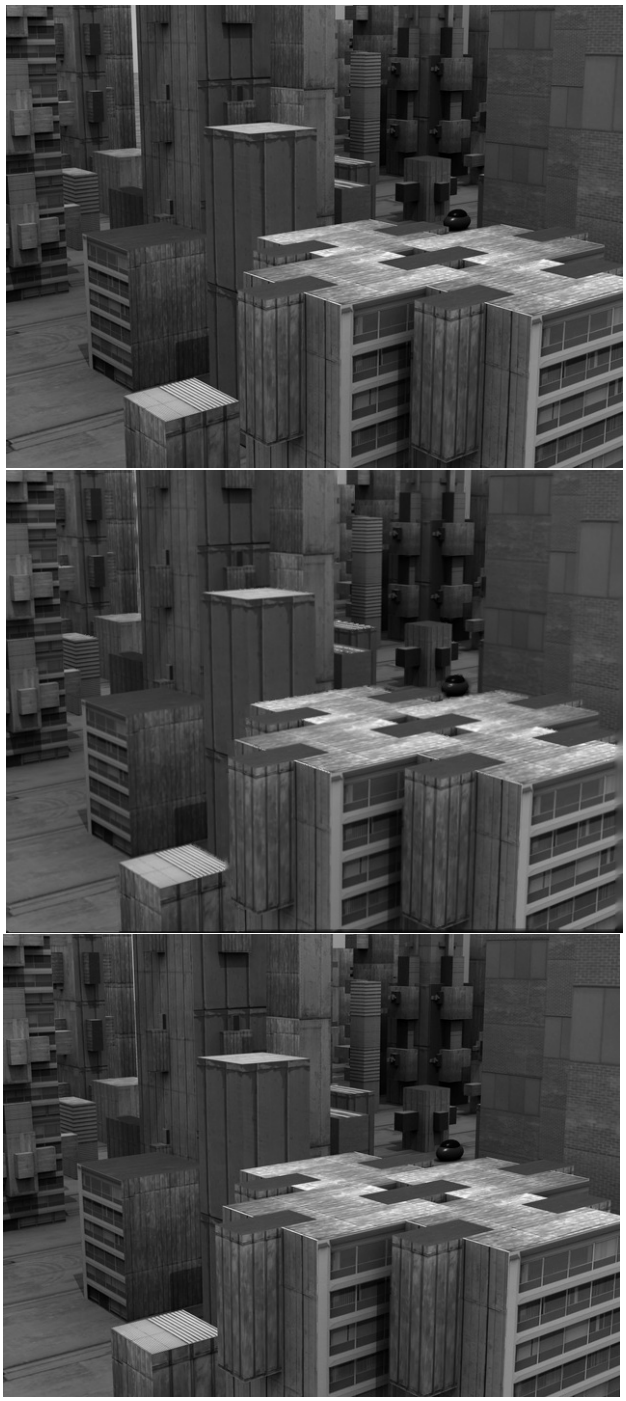


Figure 7.8: Intermediate frame interpolation. The first and third frames are original images. The middle one is interpolated.

Part II

Object recognition

Object recognition in computer vision can be briefly defined as the task of finding a given object in an image or in a video sequence. Usually, the object itself is defined by an image and may vary in scale and pose with respect to the image where the object is searched for. In recent years, there has been a huge literature on object recognition motivated by its numerous applications, in particular, the applications to object or image recognition in large databases like the web itself [TVG99, GTT01, BL02, Low04, SZ03, PCI+07], or to image classification [BZM07, BZM08].

To fix some naming conventions, let us call a “query" image, the image of the object we are given to search for and let us call a “database" image, an image where the object is searched for in order to determine whether an object in the query image is present or not. The state-of-the-art algorithms usually proceed in two main steps applied to all images (query and database): a *detection* step where a set of characteristic image locations, usually called keypoints or features, is detected; and a *description* step where a region around each detected feature is selected and a feature descriptor is computed on that region. Two features are then said to correspond if their descriptors are similar to each other. In this way, we can establish a set of correspondences between an object in one image and the same object in another image. If there are “enough" correspondences we report the presence of the object in both images.

The main difficulty resides in the fact that the database images, while they might contain the searched for object, are taken from an arbitrary distance and view point. Therefore, an algorithm that achieves the recognition process successfully needs to be invariant to these changes. In this chapter we first introduce the set of image distortions where invariances are needed and mention several ways of dealing with them. Most approaches tend to normalize these distortions; however, due to the image formation process and the non-commutation of the optical blur with a subset of image distortions, normalization techniques are not able to achieve invariance. We will discuss this issue thoroughly in Section 9. We then review some of the main state-of-the-art algorithms in the literature that address this problem. In Section 11 we present an algorithm that generates some basic geometric affine invariant quantities using a classical result on algebraic invariants. These quantities can then be used to construct some distinctive feature descriptors. In Section 12.2 we select a few of the generated quantities and test them in the context of object recognition. The evaluation is done by comparing the performance of descriptors using these quantities with the SIFT descriptor [Low04], considered to be one of the most efficient [MS05].

8 Introduction

The starting point of most approaches tackling the object recognition problem is the computation of features (or keypoints) and their feature descriptors. In order to deal with real applications, computed feature descriptors have to be invariant with respect to a set of image distortions. Let us briefly comment on the main ones.

First, let us mention the invariance with respect to blur. Image blur is the result of many factors: out of focus camera, aperture diffraction, motion blur, pixel sampling, interpolation, etc. Inverting the blur is an ill-posed problem. A typical way to obtain features invariant to blur is to use a multi-scale representation of the images. When the blur kernel has a small support, then at some scale the blur effect will disappear.

Second, ignoring the optical blur of the camera and assuming that the two images to be compared are taken from different positions and/or orientations, we need feature descriptors that are invariant with respect to projective transformations. Since this is a too general class, if the objects we are searching for can be locally approximated by a plane, then we can restrict the required invariance to the set of affine transformations. This approach is commonly used in practice and is the object of interest in this chapter. There are many approaches that satisfy affine invariance with different degrees of approximation: from Euclidean invariance to approximately affine invariance. If we introduce the camera blur into the picture, then we need descriptors that are invariant with respect to affine transforms determined by the change of position of the camera. The trouble comes from the fact that affine transforms do not commute, in general, with the blur kernel [MY11]. Assuming, as we will do in this chapter, that the blur kernel is radial, only rotations and translations commute with it. These parameters can be normalized by geometric affine invariant descriptors computed on the image. The remaining ones cannot. They are the scale parameter corresponding to the camera zoom, and the camera axis longitude and latitude. Thus, following [MY09, MY11], one is lead to distinguish between “geometric" affine invariance (corresponding to an ideal pinhole camera, with no blur) and “camera" affine invariance that models the camera blur as well. We will discuss this issue more thoroughly in Section 9. In order to get camera affine invariance, some authors have proposed to simulate the remaining set of parameters by suitably sampling the affine orbits of both images. This is the case of Ferns [LF06, OCLF09] or the variant of SIFT called ASIFT [MY09]. This may be very important in practice since in many cases there may be big scale and/or tilt changes between the two images that are compared.

Now, there is the invariance with respect to contrast changes which may be due to illumination variations or to different gamma corrections, etc. Ignoring camera blur, exact invariance to contrast changes is assured by using

morphological operations to construct the features and their descriptors, e.g. the level lines of the image [LMMM03, CLM+08] or the gradient directions which describe the directions of the normals to the level lines [BL02, Low04]. Another possibility, also used in the literature, is the binarization of the gray level comparison between couples of pixels in a neighborhood of a keypoint [LF06, OCLF09]. In practice, morphological operations are often weighted with a local contrast measure, such as the norm of the gradient. When camera blur comes into play, it mixes level lines and changes their geometry. Since the two images may be taken from different viewpoints, this cannot be addressed by only simulating the blur with the scale space, and one needs to simulate also the camera axis longitude and latitude, as proposed in [MY11].

Noise, yet another distortion, is an inevitable artifact of image acquisition. It is due mainly to physical effects which are not taken into account, and to the sampling and quantization of pixels. Noise invariance is probably impossible to achieve. The best we can aim for is noise robustness. This could be achieved by defining suitable descriptors and a suitable metric for comparing these descriptors instead of comparing them by equality. This requires a threshold on the distance of two descriptors, or some other criterion to accept or reject matched features.

Finally, let us mention occlusions. They correspond to a basic operation in image formation, and they are maybe the most important distortion. However, it is very difficult to model occlusions directly, so they are generally ignored as a source of invariants in the context of detection or recognition. The possibility to detect partially occluded objects is based on the use of local features that describe small parts of objects. Thus, we can say that occlusion invariance is achieved by using only local features (points, curves, patches, ...).

Henceforth, we will use the term *descriptor* as a set of N numerical measures from a local feature (or keypoint). Thus, a descriptor is a vector in $\mathbf{R}^N$. Using this language, we can summarize a general scheme for many methods of object recognition. The input of these methods is often a pair of images u and v, and the output is a correspondence (or list of correspondences) between parts of each image. The generic method consists of the following steps:

> **Keypoint or feature extraction:** extract the features on both images. Thus, we form two sets of features F_u and F_v associated to each one of the input images.
>
> **Computation of descriptors:** compute the descriptors of each feature. A descriptor is computed on an image patch centered at an extracted feature.
>
> **Matching:** for each feature $p_i \in F_u$ find a feature $q_j \in F_v$, such that their descriptors are as close as possible in the descriptor metric. Thus, we form a set M of pairs of matched features (p_i, q_j).

By analyzing this set of correspondences we may extract a geometric transformation between the two images. This transformation helps to group several correspondences in a geometrically consistent way. This is usually done by a step of

Clustering: search in M for clusters of pairs of features that can be matched by the same affinity (or some pre-defined kind of geometric transformation).

Finally, the last step is

Verification: verify that each cluster actually corresponds to an instance of u that appears in v under the previously found geometric transform.

Clearly, this scheme is a simplification but it gives a summarized description of the structure of many methods. In this chapter, we are mostly interested in the generation of geometric affine invariant quantities, and their use in building-up descriptors that behave more robustly than SIFT to affine transformations.

There are many descriptors that have been proposed to address the above set of invariances. Let us briefly mention here two of them: SIFT and Ferns. They will be reviewed in more detail in Section 10. Other variants will be also briefly commented. The Scale Invariant Feature Transform (SIFT) [Low99, Low04] is a descriptor defined in terms of weighted histograms of gradient directions around a given keypoint. The use of gradient directions provides robustness with respect to illumination changes. Usually keypoints are maxima of some measure (e.g. corners, edge points, blobs, ...) often computed at different scales. The histograms of orientations (in the neighborhood of a keypoint) used in the SIFT descriptor are computed on the Gaussian scale space of the image. This normalizes the scale parameter and addresses the invariance with respect to blur. The use of keypoints achieves invariance to translations. SIFT, also reorients its histogram of weighted gradient orientations to the dominant orientation of the patch where the descriptor is being computed. This makes SIFT invariant to rotations of the image plane around the camera axis. Therefore, four out of the six parameters of an affine transform are normalized (see Section 9.2.1). The other two, the longitude and latitude of the camera axis, cannot be normalized since their associated transformation does not commute with the Gaussian kernel (assuming that the camera blur can be well approximated by a Gaussian). The computation of orientation histograms provides partial invariance with respect to the angle between the object's plane and the optical axis (the latitude). In his work [Low04], D. Lowe studied the robustness of SIFT with respect to latitude changes and provided some practical bounds on their maximum variation, around 50 degrees, for SIFT to work.

It is important to note that the (geometric or camera) affine invariance is related not only with the descriptor but also to the domain where it is computed. We will call this the domain problem. Notice that, due to the camera blur, the generation of domains that are related by an affine transformation when the two images are taken by a camera from two different viewpoints is not an obvious question. Thus, we can also distinguish between geometric and camera domain problems. Mikolajczyk and Schmidt [MS04a] addressed the computation of an affine covariant domain and, by that, enforced the camera affine invariance of SIFT. They proposed an affine normalization of the keypoint neighborhood based on an iterative computation and normalization of the second

moment matrix [GL96, LG97, Lin98]. In this process, the selection of the affine domain alternates with a simulation of the corresponding blur. Thus, they attempt to compute a camera affine invariant domain; however no proof of this is provided. After this normalization, SIFT is computed, although other descriptors could be used. Let us refer to this version of SIFT as SIFT+NN (where NN refers to normalized neighborhood).

Using a different approach to impose the camera affine invariance and solve the domain problem, Yu and Morel [MY09] proposed to generate an orbit of affine deformations of one (or both) images and apply SIFT to the simulated images, obtaining results that outperform SIFT. Because of the invariance properties of SIFT, the orbit was reduced to a simulation of the longitude and latitude (tilt) parameters. Indeed, in [MY09] the authors propose a precise and careful sampling of the orbit that takes into account the SIFT performance in scale and angular (latitude) changes. In this way, they guarantee that the query image will have a positive matching with one of the orbit images, precisely the nearest one. Moreover, they proved mathematically that the resulting method is camera affine invariant, up to an arbitrary precision. On the other hand, by providing the image orbits, a set of domains is generated and one of them will be similar to the domain in the query image. The feature descriptors computed on these domains are relieved from the responsibility of being highly robust to big affine transformations; and, indeed become camera affine invariant in this setting (modulo the sampling of the orbit). This method is known as ASIFT [MY09].

Ferns [LLF05, OCLF09, LF06] also addresses the problem of illumination and camera affine invariant recognition. Although we leave the detailed description of it for Section 10.2, let us point out that, as in ASIFT, the camera affine invariance of the Ferns descriptor is obtained by the construction of an orbit of affine deformations of the model image [LLF05, OCLF09, LF06].

Finally, a different special affine invariant region detector based on the computation of sufficiently contrasted level lines, the Maximally Stable Extremal Regions (MSER), was introduced in [MCUP04]. Although the domain is normalized with respect to all six parameters of the affine transform, this normalization is not perfect, since level lines change when the amount of blur changes. Thus, MSER are geometric affine invariant but not camera affine invariant. In practice they are camera affine robust for moderate affine camera motions, but only if the scale change is not big. The MSER method provides a complementary point of view since it directly addresses the domain problem and computes geometric affine invariant domains that are based on contrasted level sets (on which arbitrary affine invariant descriptors can be later computed). We retain from it the observation that, discarding boundary effects, the level sets of the image are the natural geometric affine invariant domains on which descriptors can be computed. As pointed out above, MSER select the most contrasted ones in a precise sense that will be reviewed in Section 10.3.

In the previous paragraphs we have discussed one of the key elements for the construction of camera affine invariant descriptors around keypoints: the need to have a camera affine covariant domain, that is, the computed neighborhoods around corresponding keypoints of two images related by an affinity should

match under the affine map. The domain problem is perhaps the most difficult one and we have reviewed several proposals to solve it. Both the computation of an intrinsic affine normalized neighborhood [MS04a] and the generation of an orbit [MY09] aim to solve the domain problem, and the compensation of the partially missing invariance of SIFT to out of plane rotations. Our work in this chapter does not address this problem, but the computation of geometric affine invariant quantities assuming that the domain problem has been solved. When combined with an orbit, these quantities give also camera affine invariant descriptors. Then the following question arises:

Q. Do we need geometric affine invariant quantities if we have normalized the domain or simulated its affine distortions ?

From the above discussion, due to camera blur, we need camera affine invariant quantities. The descriptors we propose in this Chapter are only geometric affine invariant, although they can be converted to camera affine invariant if we use an image orbit, as in ASIFT, or approximately camera affine invariant if we use an intrinsic affine normalized neighborhood as in [MS04a]. In relation to ASIFT [MY09], there is still room for improvement because the images generated constitute a sampling of the affine orbit of the given images and the performance of ASIFT depends on the number of simulated images. In relation to SIFT+NN [MS04a], there is no theoretical guarantee that SIFT+NN provides a normalized camera affine invariant neighborhood, but it seems to work in practice, at least as a good approximation. In any case, the comparison of the descriptors we propose here with SIFT in both contexts SIFT+NN and ASIFT shows that the proposed quantities permit to improve the results. Let us also mention that the performance of any quantity depends on its discriminative power. The proposed quantities are different from SIFT (although they use the same organization) and exhibit more discriminative power, improving over its performance.

8.1 Contribution

We propose a set of geometric affine invariant quantities to be used in the construction of feature descriptors. They will effectively improve upon the robustness of SIFT to affine transformations. The basic quantities were introduced in [Bal95, BCG96] in the context of affine invariant image segmentation. In the context of the present Section we will omit sometimes the word "geometric" when talking about affine invariants. In [Bal95, BCG96] the authors proposed an affine covariant quantity associated to a given finite length curve Γ, namely the quantity

$$\int_0^1 \int_0^1 |c'(s) \wedge c'(t)| \, ds \, dt,$$

where $c : [0,1] \to \mathbf{R}^2$ is a parameterization of Γ and $c'(s) \wedge c'(t) := \det(c'(s), c'(t))$. By integrating on the level lines of the image, this quantity can be translated to images u defined on the plane as

$$\int_{\mathbf{R}^2} \int_{\mathbf{R}^2} |\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})| \, d\mathbf{x} d\mathbf{y}, \tag{8.1}$$

where $\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y}) := \det(\nabla u(\mathbf{x}), \nabla u(\mathbf{y}))$ is just the area determined by the two vectors $\nabla u(\mathbf{x})$ and $\nabla u(\mathbf{y})$. Note that, if $\mathcal{T}\mathbf{x} = A\mathbf{x} + \mathbf{x_0}$ where A is a 2×2 matrix, $\mathbf{x}_0$ is a given point in $\mathbf{R}^2$, and $u_{\mathcal{T}}(\mathbf{x}) = u(\mathcal{T}\mathbf{x})$, we have $\nabla u_{\mathcal{T}}(\mathbf{x}) \wedge \nabla u_{\mathcal{T}}(\mathbf{y}) = \det(A)\, \nabla u(\mathcal{T}\mathbf{x}) \wedge \nabla u(\mathcal{T}\mathbf{y})$ and, as it can be easily checked, the quantity (8.1) is affine covariant (that is affine invariant modulo a scale factor which is a power of the determinant of the affinity). This quantity can be used as a basis for constructing geometric affine invariant descriptors on keypoints.

If we had at our disposal an affine covariant neighborhood $V_{\mathbf{x}}(u)$ of a keypoint $\mathbf{x}$ (that is such that $\mathcal{T}V_{\mathbf{x}}(u_{\mathcal{T}}) = V_{\mathcal{T}\mathbf{x}}(u)$) we could define the affine invariant quantity associated to $\mathbf{x}$

$$\int_{V_{\mathbf{x}}(u)} |\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|\, d\mathbf{y}.$$

The quantity $\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})$ is one of the basic covariant quantities that we will introduce in Section 11. This and other examples are based on a classical result on algebraic invariants of the unimodular group [Wey97]. Thus, we discuss in Section 11 a generic method to construct other geometric affine covariant quantities and obtain affine invariants using combinations of them. Using these geometric affine invariant quantities, we implement in Section 12.1 a set of descriptors with an algorithmic structure similar to SIFT (see Section 12.1.2). Since we will be referring to them later on, let us call them $\mathcal{AD}$ descriptors. At this point, note that we can compute them in any given neighborhood.

Although we use geometric affine invariant quantities, even if there is no camera blur, the geometric affine invariance is only guaranteed if we have an affine covariant domain. As we said above, we do not address the domain problem in our work and we assume it is solved. We can either use the affine normalized neighborhood of Mikolajczyk and Schmidt [MS04a], or we may use an orbit of affine deformations as in [LLF05, OCLF09, MY09], or simply take advantage of the level sets of the image as it is proposed by MSER [MCUP04], although MSER do not guarantee camera affine invariance. Thus, we compared the proposed $\mathcal{AD}$ descriptors both to SIFT+NN and to ASIFT. When comparing to SIFT+NN we used the same normalized neighborhood. Our experiments show that the results obtained using $\mathcal{AD}$ improve the results obtained by SIFT+NN. When comparing to ASIFT we used the same orbit and the same square neighborhood. We did experiments with three orbit sizes and in all three cases we are able to obtain results that improve those obtained by ASIFT (specially when we use a reduced orbit). In both experiments, we see that the camera affine invariance is reinforced. Indeed, notice that, by the arguments in [MY09], geometric affine invariants computed using the Gaussian scale space become camera affine invariant in the sense of [MY09] after simulating an orbit as in ASIFT.

Inspired by MSER (see Section 10.3), we also explored a further variant based on the observation that level sets are geometric affine covariant domains. Let us first describe it in the context of SIFT. Assume that $\mathbf{x}$ is a keypoint of an image u and $\mathcal{N}_{\mathbf{x}}$ is a given neighborhood of $\mathbf{x}$ (as in common SIFT). As a first step we quantize the image u in $\mathcal{N}_{\mathbf{x}}$. To avoid the effect of illumination changes we equalize the image u in $\mathcal{N}_{\mathbf{x}}$ (that is we compute $H_{\mathbf{x}}(u)$, where

$H_{\mathbf{x}}(\cdot)$ is the distribution function of u in $\mathcal{N}_{\mathbf{x}}$), and then define the (bi)level sets $\mathcal{N}_{\mathbf{x}}^{u,j} := \{\mathbf{y} \in \mathcal{N}_{\mathbf{x}} : j\Delta \leq H_{\mathbf{x}}(u)(\mathbf{y}) < (j+1)\Delta\}$ where Δ is a quantization step and $j = 0, 1, ..., \frac{256}{\Delta} - 1$. In the case of SIFT, the new descriptor can be described as a modification of the SIFT descriptor that takes into account the level sets $\mathcal{N}_{\mathbf{x}}^{u,j}$ (in practice we use $\Delta = 64$ and $j = 0, 1, 2, 3$). It is formed by the concatenation of vectors v_j, $j = 0, 1, ..., \frac{256}{\Delta} - 1$. While in the case of common SIFT each pixel $y \in \mathcal{N}_{\mathbf{x}}$ contributes to a coordinate of a vector $v \in \mathbf{R}^{128}$, now each pixel $y \in \mathcal{N}_{\mathbf{x}}^{u,j}$ contributes to the corresponding coordinate of $v_j \in \mathbf{R}^{128}$. When $\Delta = 256$ and $j = 0$, we recover the standard SIFT. By weighting the histogram of orientations in SIFT with the geometric affine invariant quantities generated by the method described in Section 11, we have a set of descriptors that reinforce affine invariance.

Thus, for each quantity we can use the similar algorithmic structure of SIFT in $\mathcal{N}_{\mathbf{x}}$ getting the descriptors that we called $\mathcal{AD}$, or we may organize them as we described in the previous paragraph, taking into account the the level sets $\mathcal{N}_{\mathbf{x}}^{u,j}$, $j = 0, 1, ..., \frac{256}{\Delta} - 1$. Let us refer to them as the quantized level set version of $\mathcal{AD}$, or simply as $\mathcal{AD}$+QLS.

Although we use geometric affine invariant quantities, if we use a square neighborhood $\mathcal{N}_{\mathbf{x}}$ to compute the sets $\mathcal{N}_{\mathbf{x}}^{u,j}$ we break the geometric affine covariance of the domain. But we have experimentally observed that we gain distinctness with this proposal when compared to common SIFT. We have also considered the descriptors in $\mathcal{AD}$+QLS computed on the normalized affine invariant neighborhood of [MS04a] and we obtain better results than with SIFT+NN and also better results than using $\mathcal{AD}$. The comparison of $\mathcal{AD}$+QLS with ASIFT (using a square neighborhood and the same orbit for both) does not show an improvement over using $\mathcal{AD}$. The reason for this may be that the information brought by the orbit is sufficient to cancel the benefits gained by the QLS strategy. Indeed, the results obtained with $\mathcal{AD}$ and $\mathcal{AD}$+QLS are similar except in the case of a reduced orbit where the QLS version improves over $\mathcal{AD}$. The experiments will be shown in Section 12. Summarizing our comparisons, we observed that the proposed descriptors are more robust to affine deformations than SIFT.

9 Image formation and camera models

Since we are dealing with digital images, it is imperative to understand the image formation process. In this Chapter we discuss the image formation process and introduce different camera models. This discussion will serve to show the need to distinguish between geometric and camera affine invariance due to the non-commutation of the camera blur with the affine map.

9.1 Projective camera model

The oldest camera, “camera obscura", is based on what is called the pinhole camera model. This camera model is easily implemented by taking a box and making a very small hole (the pinhole) on one side of it. Now placing the pinhole in front of a light source, an inverted image of the scene (whatever is in front of the pinhole) appears at the side of the box opposite the pinhole. The image is formed by light rays emerging from the scene and entering the box through the pinhole. If the pinhole is reduced to a single point, exactly one light ray would pass by a point in the scene, the pinhole and the image plane (the projection inside the box). Of course it is impossible to reduce the size of the hole to a single point. In reality, the hole (though very small) has a finite size and each point in the image plane will be formed by the projection of a cone of light rays gathering at this point. Thus, each point in the image plane is illuminated by a cone of light rays with a finite solid angle. The larger the hole is, the wider the cone will be and in consequence the brighter the image. However, enlarging the hole will result in blurry images and shrinking the hole results in sharper but less bright images. Still very small holes might result in a diffraction effect. This model, is usually called the pinhole perspective projection model and it often provides an acceptable approximation of the imaging process [FP02].

In current days, digital cameras are equipped with lenses. The main reasons behind using a lens are:

a) To gather light since otherwise a single ray of light through the pinhole will reach exactly one point in the image under the theoretical assumption of a pinhole with the size of a single point

b) To keep the image sharp while gathering light from a large area.

These operations of light gathering is what is called the “optical blur" of the camera. Figure 9.1 shows an illustration of a projective camera model where an image of a plane is being captured by a digital camera.

As seen in Figure 9.1, the acquired image u can be written as

$$u = S_1 G_\alpha \mathcal{P} u_0, \tag{9.1}$$

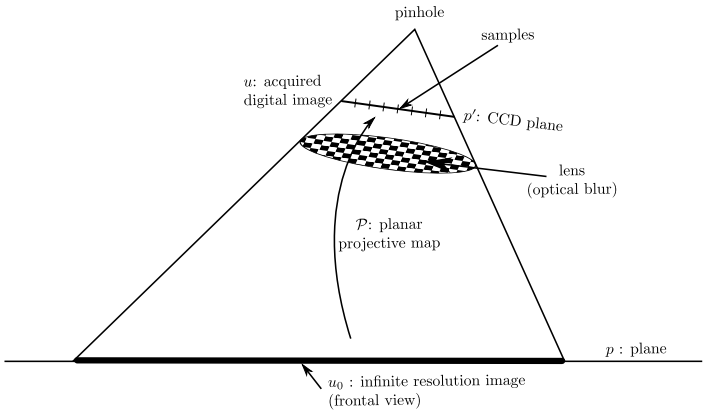


Figure 9.1: The projective camera model.

where u_0 is an infinite resolution image of a surface, $\mathcal{P}$ is a planar projective map, G_α is a Gaussian convolution kernel modeling the optical blur; and is assumed to be broad enough ensuring that no aliasing in introduced by 1-sampling of the sampling operator S_1. In general, image distortions arising from viewpoint changes can be locally approximated by affine planar transforms, assuming that objects can be locally approximated by planes. A perspective effect can be modeled by a combination of several different affine transforms applied to different image regions. Thus a reasonable and simpler model can be obtained by introducing $\mathcal{T}(x) = \mathcal{A}(x) + x_0$ with $\mathcal{A}$ being a linear map with positive determinant and $x_0 \in \mathbf{R}^2$ and writing Eq. 9.1 as

$$u = S_1 G_\alpha \mathcal{T} u_0.$$

9.2 The affine simplification

Let us first recall the decomposition of an affine matrix in terms of geometric parameters related to the observation of a plane in the scene by an affine camera [HZ03].

9.2.1 Geometric description of an affine map

Image distortions arising from viewpoint changes can be locally approximated by affine planar transforms, assuming that the objects we are searching for can be locally approximated by planes. Thus we restrict ourselves to study the invariance of descriptors with respect to planar affine transformations.

Let us describe an affine transformation in terms of intrinsic parameters that have a geometric significance. Assume that images u and v are defined in $\mathbf{R}^2$ and are related by an affine map, so that

$$v(\mathbf{x}) = u(\mathcal{T}\mathbf{x}),$$

where $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$, and

$$\mathcal{T}\mathbf{x} := A\mathbf{x} + \mathbf{x}_0 := \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}.$$

Since we will compare two keypoints, we may assume that we have already corrected the relative translation and we may assume that $\mathbf{x}_0 = 0$. We also assume that the affine map is orientation preserving, so that A has a positive determinant. Then A has a unique decomposition

$$A = H_\lambda R_1(\psi) T_t R_2(\phi) = \lambda \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{9.2}$$

where $\lambda > 0$, $R_1(\psi), R_2(\phi)$ are rotations of angles $\psi, \phi \in [0, \pi]$, respectively, and T_t is a tilt, namely a diagonal matrix whose eigenvalues are t and 1 with $t \geq 1$. Figure 9.2 shows the interpretation of this affine decomposition in terms of the relative position of the (affine) camera and the object's plane where: ϕ and $\theta = \arccos 1/t$ are the viewpoint angles, θ represents the angle between the optical axis and the normal to the object's plane, ϕ represents the relative rotation between the optical axis and a fixed axis on the plane, and ψ parameterizes a rotation of the camera plane. The angles θ and ϕ are called latitude and longitude, respectively. Finally, we can change the focal length or the distance of the camera to the scene and this is reflected in the zoom parameter λ.

9.2.2 Affine camera model

Let u_0 be an infinite resolution frontal view image of a flat object. Following [MY09], we model digital images acquired by a camera by the relation

$$u = S_1 G_\alpha \mathcal{T} u_0, \tag{9.3}$$

where $\mathcal{T}(\mathbf{x}) = A\mathbf{x} + \mathbf{x}_0$, A is a linear map with positive determinant, $\mathbf{x}_0 \in \mathbf{R}^2$, G_α is a Gaussian convolution with standard deviation $\alpha > 0$ modeling the optical blur and ensuring that there is no aliasing by 1-sampling, and finally S_1 is the sampling operator on a regular grid with 1-spacing. Notice that, following the usual simplifying assumption, the camera blur is modeled by a Gaussian kernel. From now on we do not consider the sampling operation S_1 and we assume that images are well sampled..

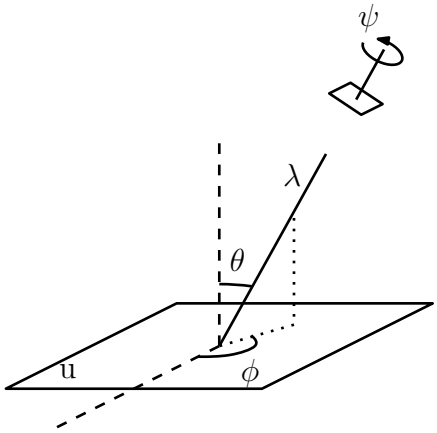


Figure 9.2: Geometric interpretation of the decomposition formula (9.2). This figure illustrates the four main parameters in the affine image deformation. The angle θ, the latitude, is the angle between the optical axis and the normal to the image plane. The plane containing the normal and the optical axis makes an angle ϕ with a fixed axis in the object's plane. This angle is called longitude. The camera can also rotate around its optical axis (rotation parameter ψ). Finally, the camera can move forward or backward, or change its focal length. This is the zoom parameter λ.

For later use, we say that the image u is a frontal snapshot if we can write it as (9.3) where $\mathcal{T}$ is a similarity, *i.e.* the tilt parameter t is equal to 1.

The presence of the camera blur obliges to distinguish between geometric affine invariant descriptors and camera affine invariant ones. Let us assume that the images are captured using an *ideal* camera, *i.e.* where the optical blur G_α is non existent and therefore $\alpha = 0$. In that case, a camera change of position leads to the images $u_0(x)$ and $\mathcal{T}u_0(x) = u_0(\mathcal{A}x + x_0)$ and we define a descriptor $\mathcal{D}$ to be a "geometric" affine invariant descriptor if

$$\mathcal{D}(u_0, \mathcal{A}x + x_0) = \mathcal{D}(\mathcal{T}u_0, x) \qquad \forall\, \mathcal{T}, x \tag{9.4}$$

But when taking $\alpha > 0$, a change of viewpoints leads to two images being $G_\alpha u_0(x)$ and $G_\beta \mathcal{T} u_0(x)$ with $\alpha, \beta > 0$. Then if $\mathcal{D}$ is a "geometric" affine invariant descriptor, the quantities $\mathcal{D}(G_\alpha u_0, \mathcal{A}x + x_0)$ and $\mathcal{D}(G_\beta \mathcal{T} u_0, x)$ are not necessarily equal, depending on the affine maps $\mathcal{T}$.

Let us illustrate this by looking at an example where the affine map is a pure rotation and another where it is a pure zoom. We will also make a simplifying assumption that the camera blur is a Gaussian with a radial kernel. Let $\mathcal{T}x = \mathcal{R}x$ where $\mathcal{R}$ is a rotation. Using the same notion we can write $\mathcal{R}u_0(x) = u_0(\mathcal{R}x)$. In this case, $\mathcal{R}$ commutes with G_α and we have that $G_\alpha \mathcal{R} u_0(x) = \mathcal{R} G_\alpha u_0(x)$.

Then if $\mathcal{D}$ is a geometric affine invariant descriptor we have

$$\mathcal{D}(G_\alpha \mathcal{R} u_0, x) = \mathcal{D}(\mathcal{R} G_\alpha u_0, x) = \mathcal{D}(G_\alpha u_0, \mathcal{R} x).$$

On the other hand, let H_λ be an operation such that $H_\lambda u_0(x) = u_0(\lambda x)$. If $\lambda > 1$, this represents a contraction of the object. That is, the effect produced when the camera moves away from the object. It can be shown that

$$G_\alpha * (H_\lambda u_0) = H_\lambda G_{\lambda^2 \alpha} * u_0.$$

Meaning that an image taken by a camera with optical blur G_α after a zoom out by a factor λ, is equivalent to take the image by a camera with optical blur $G_{\lambda^2 \alpha}$ and then scale the object by a factor λ. We can see here that the zoom has changed the blur factor. That is if we take $\mathcal{T} x = \mathcal{H}_\lambda x, \lambda > 0$, then if $\mathcal{D}$ is a geometric affine invariant descriptor we have

$$\mathcal{D}(G_\alpha H_\lambda u_0, x) = \mathcal{D}(H_\lambda G_{\lambda^2 \alpha} u_0, x) = \mathcal{D}(G_{\lambda^2 \alpha} u_0, H_\lambda x) \neq \mathcal{D}(G_\alpha u_0, H_\lambda x),$$

With a similar reasoning, one can get to the conclusion that Euclidean invariant descriptors permit to normalize the rotation and translation parameters. Scale, longitude and latitude cannot be normalized and have to be simulated [MY09]. In the literature, SIFT, in addition to normalizing rotations and translations, simulates the scale and, thus, is a similarity invariant descriptor. The proposal of ASIFT, for example, is to normalize the remaining two parameters by simulating an orbit of transformations proving that in this way one gets a camera affine invariant descriptor. In the next Chapter we review these methods along with others.

10 Literature review

The purpose of this Section is to review SIFT and some other relevant descriptors which have been introduced in the literature and are connected with our discussion here.

10.1 Review of SIFT and some relatives

10.1.1 Common SIFT and its invariance properties

Let us briefly summarize SIFT descriptors [Low04] using the generic scheme for object recognition mentioned in Chapter 8. Its input is a pair of images. To fix ideas, let us say that the first image u is a query image, while the second image v is a target image. If the query image contains a certain object, the output of the method gives the corresponding object in image v, in case it is present, and the corresponding affine map between the parts of the images u and v containing the object. The limitations of SIFT concerning its affine invariance are described in [Low04] and further analyzed in [MY09]. Since it is based on the comparison of descriptors computed on keypoints, the translation invariance is guaranteed. The invariance with respect to scale changes is obtained by simulating them. Since $G_\delta H_\lambda u_0 = H_\lambda G_{\lambda\delta} u_0$ for any $\delta, \lambda > 0$, this can be achieved using the Gaussian scale space. Indeed, to compute the "SIFT keypoints" of an image u, first one computes a Gaussian scale space pyramid of u (with scales sampled exponentially), and then tries to capture the intrinsic scale of the keypoint by finding the scale space local maxima of the Laplacian of the Gaussian. Selecting keypoints as local maxima in the scale space makes that a point x may appear several times, at several different scales. Thus, the method is (modulo discretizations) invariant with respect to scale. Finally, for each keypoint found, a more stable positioning with sub-pixel accuracy is computed by fitting a 3D quadratic function to the keypoint. This determines the interpolated location of the maximum.

Now, given a keypoint $\mathbf{x}$, its SIFT descriptor is based on the computation of histograms of gradient orientations in a neighborhood of $\mathbf{x}$. To compute the gradient directions, one computes first the dominant orientation of the gradient in $\mathbf{x}$ (estimated from the gradients around this point) and takes it as the new x-axis. The gradient orientations are then referred to this axis. In this way, the method is invariant to rotations in the image plane. Then, the SIFT descriptor is computed in a neighborhood of $\mathbf{x}$, typically of size 16×16 in the standard SIFT implementation. This neighborhood is divided in 4×4 blocks of size 4×4. On each of them, a weighted histogram of directions quantized in 8 angular bins is computed. The weights of this histogram depend both on the modulus of the gradient at the pixel and on the distance to the central keypoint. At the

end we have a descriptor which is a vector v with 128 coordinates. Each coordinate of v contains the weight contribution of all pixels of a given block with a given angular orientation. Finally, in practice, to avoid quantization effects, each orientation occurrence is distributed in several neighboring bins. We note that the use of scale space provides some invariance to camera blur; and, since SIFT uses gradient directions, it is robust to illumination changes. Furthermore, the descriptor is first normalized, then large values are thresholded and finally normalized again which makes SIFT robust to saturation effects.

As it is proved in [MY11] (Theorem 1), for two frontal snapshots of the same full resolution flat image u_0, SIFT descriptors are identical if both camera blurs coincide and, otherwise, they become similar as soon as the scale of the simulated Gaussian grows. In conclusion, as we have mentioned, the translation, rotation and scale parameters can be fixed, although one has to pay attention to the sampling issues [MY09]. Now, taking (9.2) into account, there are still two parameters to be fixed in order to get camera affine invariance; namely, the angle between the normal to the object's plane and the optical axis of the camera (the latitude), and the relative rotation between the optical axis and a fixed axis on the object's plane (the longitude), see Figure 9.2. Let us mention the two main methods that have been proposed to address this problem: either by computing an intrinsic neighborhood of the keypoint which aims to achieve camera affine invariance [MS04a], or by simulating the remaining affine deformations of the query image, related to the longitude and latitude angles, as proposed in ASIFT [MY09] (see also [PH03] for a first step towards this method). A full discussion of this problem and the relevance of respecting Shannon sampling theorem when simulating the affine deformations can be found in [MY09].

A related variation of SIFT proposed in [LSP04], which improves somewhat its rotation invariance but does not tackle invariance under arbitrary affinities, consists in using circular neighborhoods instead of square ones. Anyhow, this is a minor modification, since the square neighborhoods of common SIFT are rotated towards a principal direction, and the points inside the neighborhood are weighted using a circular distribution around the center of the square.

10.1.2 ASIFT: simulating the tilt and longitude parameters

Although the method in [MS04a] produces good results, better results are obtained by simulating all possible affine distortions of the image and applying SIFT to each of them [MY09]. By our previous discussion (see [MY09]) it suffices to simulate the affine deformations determined by the two parameters ϕ and $t = |\frac{1}{\cos\theta}|$. Given an image u, one first simulates the rotations with respect to the longitude parameter. An important point for digital images, as discussed in [MY09], is that the tilt involves a subsampling of factor t in the x-direction (after a rotation by ϕ), and therefore its simulation requires the previous application of an anti-aliasing filter, namely the convolution by a gaussian with standard deviation $c\sqrt{t^2-1}$ (for good anti-aliasing, $c \sim 0.6$) [MY09]. Thus, a digital tilt in the direction x (resp. y), is simulated by the map $T_t^x G_{c\sqrt{t^2-1}}$ (resp. $T_t^y G_{c\sqrt{t^2-1}}$).

Since $T_t^x G_{c\sqrt{t^2-1}} G_c = G_c T_t^x$ ([MY09], Theorem 1), by simulating digital tilts we are able to commute tilts with Gaussian blur, and this is the main ingredient to prove that ASIFT is camera affine invariant ([MY09], Theorem 2). On the other hand, it is not geometric affine invariant. The conclusion from [MY09] is that, by simulating tilts, longitudes and scales, one can convert a descriptor that is invariant to similarity transforms into a camera affine invariant one.

SIFT on affine neighborhoods We have mentioned the invariance of SIFT with respect to the parameters λ and ψ in (9.2). The lack of invariance with respect to the image deformations induced by the angles θ and ϕ has to be compensated. In [MS04a], the authors proposed to compute an affine invariant interest point detector, called Harris affine, which detects a keypoint and provides it with an affine region given by a 2×2 matrix $\mathbf{M}$ representing this region. The Harris interest point detector is first applied at several scales [HS88], followed by an iterative selection of the scale and the location (as an extremum over scale of the Laplacian of the Gaussian) [MS04a]. This provides a set of points that are robust to scale changes. Then, an affine normalization of the point neighborhood is computed as a fixed point by alternating the selection of the affine domain with a simulation of the corresponding blur. One can interpret that as an approximate way to obtain a camera affine invariant domain. The method and the algorithm for finding such keypoints and their associated normalized neighborhoods is discussed in depth in [MS04a]. The output is a set of keypoints where for each of them we have its position, an affine covariant neighborhood, and its dominant orientation which is obtained as in SIFT but using the normalized neighborhood. The neighborhood is expressed by the so called shape adaptation matrix which is a symmetric positive-definite $\mathbf{2} \times \mathbf{2}$ matrix $\mathbf{M}$ whose eigenvectors express the axis directions of the corresponding elliptical shape $\{\mathbf{x} \in \mathbf{R}^2 : \langle \mathbf{M}\mathbf{x}, \mathbf{x} \rangle \leq 1\}$. Having $\mathbf{M}$, one can normalize the image restricted to the affine neighborhood of the keypoint by mapping it to an image restricted to the unit circle. With this, one compensates the affine transform caused by the camera change of position. Applying SIFT to the normalized image on the unit circle, one obtains in principle a camera affine invariant descriptor. Although the authors do not demonstrate rigorously this camera affine invariance, it seems to work in practice.

Other descriptors related to SIFT Inspired by SIFT, many other related descriptors have been proposed with the purpose of improving or speeding up SIFT. In particular, let us mention SURF detector and descriptor (Speeded-Up Robust Features) [BTVG06], PCA SIFT [KS04], and FAST [RD06]. Also, other descriptors specially tailored for operating on a dense set of keypoints have been proposed like HOG (Histograms of Oriented Gradients) [DT05], GLOH (Gradient Location and Orientation Histograms) [MS05], LESH (Local Energy Based Shape Histograms) [SH08]and, in some sense, spin images [JH99]. Let us note that it is also possible, and useful, to compute SIFT descriptors on dense keypoints [LYT+].

10.2 Review of Ferns method

A different method which also addresses the problems of illumination and affine invariant recognition was proposed in [LLF05, LF06, OCLF09]. Given a query and a target image, the camera affine invariance of the method is given, as in ASIFT, by the construction of an orbit of affine and blur deformations of the target image. Then a set of keypoints (using a cornerness measure) are computed on this orbit. Only those keypoints that are stable under the deformations are used as effective keypoints. Each of them determines a class which is characterized by the comparisons (encoded in a binary vector) of the gray levels of a selected random set of pairs of points in its neighborhood. These comparisons determine the posterior probability that a given point belongs to a class and may be used for classification using a naive Bayes approach. Thus, assuming that the priors of each class are uniform we are lead to the computation of the likelihood of each class for each set of answers to the questions. Let us mention the Ferns method in [OCLF09] (note that the randomized tree method [LLF05] is a variant of the Ferns method with a different organization). Since one cannot assume that each question is independent of the others, the proposal in [OCLF09] is to assume that the set of questions is organized in subsets, called Ferns, which are independent. Thus, a Fern is determined by a random selection of a subset of pairs of pixels where the values of the gray levels are compared. Several Ferns are used to characterize a keypoint. Each comparison produces a binary answer. This guarantees the morphological invariance of the method which amounts to its invariance with respect to illumination changes [Ser82]. The probability of each Fern, given the class, is learned offline from the keypoints computed in the orbit. In order to compute it, a tree structure is used. The product of the likelihoods of Ferns gives the likelihood of the set of answers that characterizes each class. This method does not require the storage of the orbit of affine deformations since, for each Fern, the probability computations can be organized as a tree that can be learned offline.

10.3 Review of MSER method

A completely different approach to affine invariance is that of MSER [MCUP04]. The main idea behind MSER, as opposed to SIFT, is to build-up affine covariant domains on which arbitrary affine invariant descriptors (or arbitrary descriptors, if one performs first an affine normalization [MCUP04]) will be computed, giving thus a solution to the domain problem. Two key observations in [MCUP04] lead to the choice of Maximally Stable Extremal Regions (MSER) as robust elements to establish image correspondences under severe viewpoint changes for automatic reconstruction of 3D scenes. In the wide-baseline stereo problem, local image deformations cannot be realistically approximated by Euclidean motions and a full affine model is required (as an approximation to the projective transformation between the images). On the other hand, the elements should be robust against illumination changes modeled here as monotonic trans-

formation of image intensities. Thus, MSER are defined as most contrasted connected components of upper and lower level sets of the image [MCUP04]. Let us point out that MSER are only geometric, and not camera, affine invariant. If we take the camera blur into account, MSER only achieves invariance with respect to translation and rotation. The other three parameters (zoom, camera axis longitude and latitude) cannot be perfectly normalized since they do not commute with the image blur and have to be simulated [MY09].

Let us review the definition of MSER. We recall it here using the language of the trees of connected components of upper and lower level sets [Ser82, SS95, CM02, CM10] (an analogous notion for the tree of shapes [MG00] can also be defined, see [MG00, CM10]). Let Ω be the image domain, usually a closed rectangle in $\mathbf{R}^2$. Let $u : \Omega \to \mathbf{R}$ be a given image (modeled as an upper semicontinuous function). Let us first introduce some basic notation that will be also of later use in Chapter 11. Given $\lambda \in \mathbf{R}$, we denote $\{u \geq \lambda\} := \{\mathbf{x} \in \Omega : u(\mathbf{x}) \geq \lambda\}$ which is the upper level set of u determined by the height λ. We denote by $\mathcal{CC}(\{u \geq \lambda\})$ the family of connected components of $\{u \geq \lambda\}$. In order to define the Maximally Stable Extremal Regions we fix a threshold value $\delta > 0$. If u takes values in a discrete set of integers we may take $\delta = 1$. This is the more relevant case in applications since we deal with digital (sampled and quantized) images. For each connected component of an upper level set of u $X_\lambda \in \mathcal{CC}(\{u \geq \lambda\})$ we consider $X_{\lambda-\delta} \in \mathcal{CC}(\{u \geq \lambda - \delta\})$, $X_{\lambda+\delta} \in \mathcal{CC}(\{u \geq \lambda + \delta\})$ such that $X_{\lambda+\delta} \subseteq X_\lambda \subseteq X_{\lambda-\delta}$. We define the function

$$F_\delta^u(\lambda) := \frac{\text{Area}(X_{\lambda-\delta}) - \text{Area}(X_{\lambda+\delta})}{\text{Area}(X_\lambda)}. \tag{10.1}$$

We say that X_λ is a Maximally Stable Extremal (upper) Region if X_λ achieves a local minimum of $F_\delta(\lambda)$. The function F_δ is well defined on the maximal branches of the tree of connected components of upper level sets of u, where no bifurcation takes place [CM10]. When X_λ contains a bifurcation, the connected component $X_{\lambda+\delta} \in \mathcal{CC}(\{u \geq \lambda + \delta\})$ is not uniquely defined. We have dismissed those elements.

In a similar way we can define a Maximally Stable Extremal (lower) Region using this time the connected components of the lower level sets of u. We shall refer to both of them as MSER.

As in Section 10.1, we dismiss the problems caused by boundary effects and we assume that the image u is defined in $\mathbf{R}^2$. If $\mathcal{T}$ is an affine transformation and we define $u_\mathcal{T}(\mathbf{x}) = u(\mathcal{T}\mathbf{x})$, then the trees of shapes of u and $u_\mathcal{T}$ have the same structure. Since the function $F_\delta^u(\lambda) = F_\delta^{u_\mathcal{T}}(\lambda)$, we have that the MSER of u and $u_\mathcal{T}$ are related by $\mathcal{T}$ in a covariant way. Thus, they are invariant under affine transformations. Its invariance under contrast changes comes from the fact that we are using connected components of level sets.

As we mentioned previously, the affine invariance of MSER is only geometric, and camera invariance does not hold. This makes that MSER may fail under large scale changes, large tilts, or when well contrasted shapes are not present [MY09]. In these cases, the image shape boundaries tend to mix. Following

[MY09], camera affine invariance can be obtained after simulating the scale, longitude and latitude parameters.

Finally, let us say that a related descriptor based on level lines was proposed in [LMMM03, CLM+08]. As MSER, it is geometric affine invariant and photometric invariant.

Descriptors for MSER Following the general scheme mentioned in Section 8 MSER are geometric affine covariant domains. On them, many different descriptors can be computed, for example SIFT. The descriptors originally proposed in [MCUP04] are based on constructing one or several *distinguished regions* (DR) around each MSER (e.g., an ellipse, the MSER itself, or several enlarged/reduced copies of the MSER or its convex hull). Then, rotationally invariant complex moments of the image inside the DR are computed after applying an affine transformation that normalizes the DR (e.g. so that the covariance matrix of the transformed DR becomes diagonal) [MCUP04]. Alternatively, one can build the descriptor using standard geometric affine invariant moments of the color values inside the DR [MMVG99, MCUP04]. An analogous approach using the shapes of the image [MG00, CM10] can be found in [Mon99].

11 Affine-invariant descriptor generator

Our purpose in this section is the generation of geometric affine invariant quantities. The quantities we propose here will be used in Section 12.1 to generate new descriptors which are geometric affine invariant. When used with an affine normalized neighborhood as in [MS04a] the descriptors become approximately camera affine invariant, although no formal proof of this exists. Camera affine invariance, up to an arbitrary level of precision, can be obtained by simulating scales, longitude and latitude parameters (which are the three parameters that do not commute with radial camera blurs) as in ASIFT [MY09]. This camera affine invariance can be obtained if we start from any Euclidean invariant descriptor, like SIFT for example. However, the experiments displayed in Section 12.2 comparing our descriptors and SIFT, both in the context of SIFT-NN and ASIFT using their normalization strategies, show that the proposed ones are more robust to affine perturbations.

We have distinguished above between geometric and camera affine invariance. Henceforth, to simplify our expressions, when we say affine invariant or covariant we mean geometric affine invariant or covariant, respectively. When we refer to camera affine invariance (resp. covariance) we will say it explicitly.

Let us denote by $GL(2,\mathbf{R})^+$ the set of all 2×2 matrices with positive determinant. To avoid boundary effects we assume in this Section that images are defined on $\mathbf{R}^2$. If $A \in GL(2,\mathbf{R})^+$ and $u : \mathbf{R}^2 \to \mathbf{R}$ is a given image, we denote $u_A(\mathbf{x}) = u(A\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^2$.

Let $\mathcal{L}$ be a class of images (e.g. continuous, of bounded variation,...). We say that $\mathcal{L}$ is $GL(2,\mathbf{R})^+$ invariant if $u_A \in \mathcal{L}$ for any $u \in \mathcal{L}$ and any $A \in GL(2,\mathbf{R})^+$. Notice that we are identifying the word image with function from $\mathbf{R}^2$, or a domain of $\mathbf{R}^2$, to $\mathbf{R}$ (gray level image) ignoring for the moment the presence of the blur kernel.

Definition 1 (Affine invariant and covariant descriptors)**.** *Let $\mathcal{L}$ be a $GL(2,\mathbf{R})^+$ invariant class of images and $\mathcal{F}$ be a class of allowed subsets of $\mathbf{R}^2$. Let $H(u,R)$ be a quantity which can be computed for any image $u \in \mathcal{L}$ and any subset $R \in \mathcal{F}$. Let $k \in \mathbf{R}$. We say that the quantity H is affine k-covariant if $H(u_A, A^{-1}(R)) = (\det(A))^k H(u,R)$ for any image u, any subset $R \in \mathcal{F}$ and any $A \in GL(2,\mathbf{R})^+$. When $k = 0$ we say that H is affine invariant.*

In other words, affine k-covariant quantities are quantities that are invariant to affine transformations of the image, up to a scale factor that is a power of the affine matrix determinant.

There are many ways to build new covariant quantities from old ones. Functions of covariants (of possibly different degree) can be made covariant, provided the functions have a suitable degree of homogeneity. Arbitrary homo-

geneous functions of covariants are very general. They include, for instance, taking limits, integrals, maxima and minima of sets of covariants.

All of these constructions produce new invariant quantities defined over the same class $\mathcal{F}$. A more interesting way to produce new invariants is by extending the class $\mathcal{F}$ on which an invariant is defined. The following lemma extends any invariant defined on upper level sets to an invariant defined on level curves.

Let $\mathcal{F}_u$ denote the family of connected components of all upper level sets of the image u.

Lemma 1. *Let $\mathcal{L}$ be a $GL(2,\mathbf{R})^+$ invariant class of continuous images. Let $k \in \mathbf{R}$ and H be an affine k-covariant quantity defined on pairs (u, X) where $u \in \mathcal{L}$ and $X \in \mathcal{F}_u$. For each $\lambda \in \mathbf{R}$ we define $F(u, \partial X_\lambda) := \lim_{\delta \to 0+} \frac{H(u, X_{\lambda-\delta}) - H(u, X_\lambda)}{\delta}$, assuming that the limit exists, where $X_\lambda \in \mathcal{CC}(\{u \geq \lambda\})$, $X_{\lambda-\delta} \in \mathcal{CC}(\{u \geq \lambda - \delta\})$ and $X_\lambda \subseteq X_{\lambda-\delta}$. Then $F(u, \partial X_\lambda)$ is an affine k-covariant quantity defined on the boundaries of upper level sets.*

We have written the previous lemma in an informal way. We observe that if $X_\lambda \in \mathcal{CC}(\{u \geq \lambda\})$ and δ is small enough, then there is only one connected component $X_{\lambda-\delta} \in \mathcal{CC}(\{u \geq \lambda - \delta\})$ containing X_λ. Then the Lemma follows essentially by observing that $F_\delta(u, \partial X_\lambda) := \frac{H(u, X_{\lambda-\delta}) - H(u, X_\lambda)}{\delta}$, $\delta > 0$, is affine k-covariant and passing to the limit as $\delta \to 0^+$.

The introduction of this Lemma was motivated by the next example.

Example 1. Let us compute the affine invariant quantity on level lines associated to the area function defined on the upper level sets of an image u. Assume first that u is smooth and the integrals converge. As usual $\nabla u(\mathbf{x})$ denotes the gradient of u at the point $\mathbf{x}$ and $|\nabla u(\mathbf{x})|$ its modulus. Then for each $\mu \in \mathbf{R}$, if X_μ is a connected component of $\{u \geq \mu\}$ and we denote by $u|_{X_\mu}$ the restriction of u to X_μ, by the coarea formula (see [AFP00]), we have

$$H(X_\mu) := \text{Area}(X_\mu) = \int_\mu^\infty \int_{\partial\{u|_{X_\mu} \geq \eta\}} \frac{1}{|\nabla u(\mathbf{x})|}\, d\mathcal{H}^1(\mathbf{x})\, d\eta,$$

where $d\mathcal{H}^1$ denotes the one-dimensional Hausdorff measure, i.e., the arc length on $\partial\{u|_{X_\mu} \geq \eta\}$ in the above integral. Hence

$$F(u, \lambda, \delta) := \frac{1}{\delta} \int_{\lambda-\delta}^{\lambda+\delta} \int_{\partial\{u|_{X_\mu} \geq \eta\}} \frac{1}{|\nabla u(\mathbf{x})|}\, d\mathcal{H}^1(\mathbf{x})\, d\eta \approx 2 \int_{\partial X_\lambda} \frac{1}{|\nabla u(\mathbf{x})|}\, d\mathcal{H}^1(\mathbf{x}).$$

If $g^u(x) = \frac{1}{|\nabla u(\mathbf{x})|}$ and, for any set of finite perimeter E we define the weighted perimeter $P_{g^u}(E) := \int_{\partial^* E} g_u(\mathbf{x}) d\mathcal{H}^1(\mathbf{x})$, where $\partial^* E$ denotes the essential boundary of E [AFP00], then we have

$$F(u, \lambda, \delta) \to 2P_{g^u}(X_\lambda) \quad \text{as } \delta \to 0+.$$

This is the affine invariant quantity on level lines associated to the area of the level sets of u.

Remark 1. Using the previous example, we may redefine Maximally Stable upper regions of the image u as the local minimizers in the tree of upper connected components of

$$G(\lambda) := \frac{P_{g^u}(X_\lambda)}{|X_\lambda|} = \lim_{\delta \to 0+} F^u_\delta(\lambda),$$

where $F^u_\delta(\lambda)$ is defined in (10.1), $\lambda \in \mathbf{R}$. Notice that the above quotient is a perimeter/area ratio, hence we may interpret MSER as local Cheeger sets (with respect to the weighted perimeter P_{g^u}) of the image domain, when we restrict the family of sets to the connected components of upper (or lower) level sets of the image. Similarly we may redefine Maximally Stable lower regions of the image u (or the Maximally Stable Shapes, see [Mei11, CM10]). The same analysis has been given in [KZBB10] where other interesting affine invariant measures for shape selection are also derived.

Our purpose in this section is to describe some other basic rules to generate affine invariants and covariants. In the language of definition 1, these new invariants are defined over the same class of subsets, by combining vector-valued invariants using simple algebraic rules.

Our discussion will be restricted to $\mathbf{R}^2$. Vectors of $\mathbf{R}^2$ will be designed by italic letters $\mathbf{x}$, $\mathbf{y}$, sometimes with sub-indices. Covectors of $\mathbf{R}^2$, that is, elements of the dual space, will be denoted by greek letters ξ, $\hat{\xi}$, etc. By $\langle \xi, \mathbf{x} \rangle$ we denote the standard dual pairing between the vector $\mathbf{x}$ and the covector ξ. In what follows, we fix the standard canonical basis $\mathbf{e}_1 = (1, 0)$, $\mathbf{e}_2 = (0, 1)$ of $\mathbf{R}^2$ and its dual basis ξ_1, ξ_2 (so that $\xi_i(e_j) = \delta_{ij}$, $i, j = 1, 2$, where $\delta_{ij} = 1$ if $i = j$, and 0 if $i \neq j$), and the action of a covector ξ of coordinates (ξ_x, ξ_y) on a vector $\mathbf{x} = (x, y)$ will be denoted by $\langle \xi, \mathbf{x} \rangle = \xi_x x + \xi_y y$, which is the standard scalar product. Clearly, given two vectors $\mathbf{x}, \mathbf{y} \in \mathbf{R}^2$ the determinant of the matrix whose columns are $\mathbf{x}$ and $\mathbf{y}$ is affine 1-covariant. This is the basic covariant made of vectors and the others can be deduced from it. Let us denote by $\mathbf{x} \wedge \mathbf{y}$ this determinant. Notice that vectors transform cogradiently, i.e., as $\mathbf{x} \to A\mathbf{x}$ while covectors transform contragradiently, ie., as $\xi \to A^{-t}\xi$ by the group $GL(2, \mathbf{R})^+$ [Wey97]. Notice that when using coordinates, we can identify covectors as elements of $\mathbf{R}^2$, although we have to keep in mind their transformation rules.

Let $SL(2, \mathbf{R})$ be the unimodular group in $\mathbf{R}^2$, that is, the set of 2×2 matrices of determinant 1. Then we specify to the case $N = 2$ the following result proved in [Wey97], Theorem 2.6.A.

Theorem 1. *([Wey97], Theorem 2.6.A) Let $\mathbf{x}, \mathbf{y} \in \mathbf{R}^2$ be vectors and $\xi, \hat{\xi}$ be two covectors. Then $\mathbf{x} \wedge \mathbf{y}$, $\langle \xi, \mathbf{x} \rangle$, and $\xi \wedge \hat{\xi}$ are the basic invariants for the unimodular group. Thus any other algebraic invariant is a polynomial in those basic elements.*

Observe that $\mathbf{x} \wedge \mathbf{y}$ generates an affine 1-covariant for the group $GL(2, \mathbf{R}^+)$, $\langle \xi, \mathbf{x} \rangle$ is an affine invariant and $\xi \wedge \hat{\xi}$ generates an affine -1-covariant.

Definition 2. *Let $\mathcal{L}$ be a $GL(2, \mathbf{R})^+$ invariant class of images. Let $k \in \mathbf{R}$. Let $H(u, \mathbf{x}_1, \ldots, \mathbf{x}_p)$ be a quantity which can be computed for any image $u \in \mathcal{L}$ and any*

points $\mathbf{x}_1, \ldots, \mathbf{x}_p \in \mathbf{R}^2$. *We say that the quantity* H *is affine* k*-covariant density if* $H(u_A, \mathbf{x}_1, \ldots, \mathbf{x}_p) = (\det(A))^k H(u, A\mathbf{x}_1, \ldots, A\mathbf{x}_p)$ *for any image* $u \in \mathcal{L}$, *any points* $\mathbf{x}_1, \ldots, \mathbf{x}_p$ *and any* $A \in GL(2, \mathbf{R})^+$. *If* $k = 0$ *we say that* H *is an affine invariant density.*

Inspired by Theorem 1 we give some examples of affine k-covariant densities. We always assume that the image $u : \mathbf{R}^2 \to \mathbf{R}$ is smooth enough so that we can compute its gradient. This is so for instance if $u = G_t * u_0$ where $u_0 : \mathbf{R}^2 \to \mathbf{R}$ is a given image in $L^\infty(\mathbf{R}^2)$ (the space of measurable and essentially bounded functions) and G_t is the Gaussian of variance $t > 0$.

Examples.

1. The most basic invariant density is $H_{00}(u, \mathbf{x}) = u(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^2$.

2. Since $\nabla u_A(\mathbf{x}) = A^t \nabla u(A\mathbf{x})$, we have

$$\langle \mathbf{y}, \nabla u_A(\mathbf{x}) \rangle = \langle \tilde{\mathbf{y}}, \nabla u(\tilde{\mathbf{x}}) \rangle,$$

where $\tilde{\mathbf{x}} = A\mathbf{x}$ and $\tilde{\mathbf{y}} = A\mathbf{y}$, and $\langle \cdot, \cdot \rangle$ denotes the standard scalar product. Thus $H_{01}(u, \mathbf{x}, \mathbf{y}) = \langle \mathbf{y}, \nabla u(\mathbf{x}) \rangle$, $\mathbf{x}, \mathbf{y} \in \mathbf{R}^2$ is an affine invariant density.

3. Observe that

$$\nabla u_A(\mathbf{x}) \wedge \nabla u_A(\mathbf{y}) = \det A \, \nabla u(A\mathbf{x}) \wedge \nabla u(A\mathbf{y}).$$

Thus, we see that $H_{10}(u, \mathbf{x}, \mathbf{y}) = \nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})$ is an affine 1-covariant density.

4. Let

$$J := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

The matrix J corresponds to a rotation by an angle of $\frac{\pi}{2}$. Observe that $J^2 = -I$, $JJ^t = I$, and

$$JA^tJ^{-1} = \mathrm{Cof}\, A^t = \det A \cdot A^{-1} \qquad \text{and} \qquad AJA^t = \det A \, J. \tag{11.1}$$

Notice that we have

$$D^2 u_A(\mathbf{x}) = A^t D^2 u(A\mathbf{x}) A.$$

Then the quantity $H_{20}(u, \mathbf{x}, \mathbf{y}, \mathbf{z}) := \langle D^2 u(\mathbf{x})(J\nabla u(\mathbf{y})), J\nabla u(\mathbf{z}) \rangle$ is an affine 2-covariant density, $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{R}^2$. Indeed

$$\begin{aligned} \langle D^2 u_A(\mathbf{x})(J\nabla u_A(\mathbf{y})), J\nabla u_A(\mathbf{z}) \rangle &= \langle A^t D^2 u(A\mathbf{x}) A(JA^t \nabla u(A\mathbf{y})), J\nabla u(A\mathbf{z}) \rangle \\ &= \langle D^2 u(A\mathbf{x})(AJA^t \nabla u(A\mathbf{y})), AJA^t \nabla u(A\mathbf{z}) \rangle \\ &= (\det A)^2 \langle D^2 u(A\mathbf{x})(J\nabla u(A\mathbf{y})), J\nabla u(A\mathbf{z}) \rangle. \end{aligned}$$

Notice that $\langle D^2 u(\mathbf{x})(J\nabla u(\mathbf{x})), J\nabla u(\mathbf{x}) \rangle = |\nabla u(\mathbf{x})|^3 \mathrm{curv}(u)(\mathbf{x})$ where $\mathrm{curv}(u)(\mathbf{x})$ denotes the curvature of the level line of u passing by the point $\mathbf{x}$.

5. Combining the above quantities one can get other affine invariant quantities. For instance, the quantity

$$Q(u)(\mathbf{x},\mathbf{y}) := \frac{\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})}{|\langle D^2u(\mathbf{x})(J\nabla u(\mathbf{x})), J\nabla u(\mathbf{x})\rangle|^{1/2}}$$

is affine invariant.

Lemma 2. *Let $k \in \mathbf{R}$. Assume that $H(u, \mathbf{x}_1, \ldots, \mathbf{x}_p)$ is an affine k-covariant density defined for u in a $GL(2,\mathbf{R})^+$ invariant class of images $\mathcal{L}$. If we integrate H with respect to j of its coordinates, we obtain an affine $k-j$-covariant density.*

Proof. (i) Suppose that we integrate its first j coordinates. Then

$$\begin{aligned}\int_{\mathbf{R}^2}\ldots\int_{\mathbf{R}^2} H(u_A, \mathbf{x}_1, \ldots, \mathbf{x}_p)\, d\mathbf{x}_1 \ldots \mathbf{x}_j &= (\det A)^k \int_{\mathbf{R}^2}\ldots\int_{\mathbf{R}^2} H(u, A\mathbf{x}_1, \ldots, A\mathbf{x}_p)\, d\mathbf{x}_1 \ldots \mathbf{x}_j \\ &= (\det A)^{k-j} \int_{\mathbf{R}^2}\ldots\int_{\mathbf{R}^2} H(u, \mathbf{y}_1, \ldots, \mathbf{y}_p)\, d\mathbf{y}_1 \ldots \mathbf{y}_j\end{aligned}$$

□

Using the examples above combined with Lemmas 1 and 2 we get examples of affine covariant and invariant quantities.

Proposition 1. *Let $u : \mathbf{R}^2 \to \mathbf{R}$ be an image which we assume smooth enough and let $n(\mathbf{x})$ denote the unit normal to the level line of u passing by the point $\mathbf{x}$. Let $X_\lambda \in \mathcal{CC}(\{u \geq \lambda\})$, $\lambda \in \mathbf{R}$. Assume, if necessary, that λ is not a critical value, that is $\nabla u(\mathbf{y}) \neq 0$ for any $\mathbf{y} \in \partial X_\lambda$. Let $k \in \mathbf{R}$.*

(i) For any $\mathbf{x} \in \mathbf{R}^2$, the integrals $\int_{X_\lambda} |\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|^k\, d\mathbf{y}$ and

$$\int_{\partial X_\lambda} |\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|^k / |\nabla u(\mathbf{y})|\, d\mathcal{H}^1(\mathbf{y})$$

are affine $(k-1)$-covariant quantities.

(ii) The integrals $\int_{X_\lambda} |\langle \mathbf{y}, \nabla u(\mathbf{y})\rangle|^k\, d\mathbf{y}$ and $\int_{\partial X_\lambda} |\langle \mathbf{y}, \nabla u(\mathbf{y})\rangle|^k / |\nabla u(\mathbf{y})|\, d\mathcal{H}^1(\mathbf{y})$ are affine -1-covariant.

(iii) The quantities

$$\int_{X_\lambda} |\langle D^2u(\mathbf{y})(J\nabla u(\mathbf{y})), J\nabla u(\mathbf{y})\rangle|^k\, d\mathbf{y},$$

$$\int_{\partial X_\lambda} |\langle D^2u(\mathbf{y})(J\nabla u(\mathbf{y})), J\nabla u(\mathbf{y})\rangle|^k / |\nabla u(\mathbf{y})|\, d\mathcal{H}^1(\mathbf{y})$$

are affine $(2k-1)$-covariant quantities.

Remark 2. Many other covariant quantities can be defined. For instance, for any $\mathbf{x}, \mathbf{e} \in \mathbf{R}^2$, the integral

$$H(u, \mathbf{x}, \mathbf{e}) := \int_0^\infty |\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{x} + s\mathbf{e})|^k \, ds$$

is an affine k-covariant quantity. For any $\mathbf{x}, \mathbf{y} \in \mathbf{R}^2$, the integral

$$\int_0^\infty |\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y} + sJ\nabla u(\mathbf{y}))|^k \, ds$$

is an affine $(k-1)$-covariant quantity. Also the quantities

$$\int_{X_\lambda} |\mathrm{det} D^2 u(\mathbf{y})|^k \, d\mathbf{y} \quad \text{and} \quad \int_{\partial X_\lambda} |\mathrm{det} D^2 u(\mathbf{y})|^k / |\nabla u(\mathbf{y})| \, d\mathcal{H}^1(\mathbf{y})$$

are affine $(2k-1)$-covariant quantities.

The covariant quantities defined in (i) are related to the affine covariant quantities defined in [Bal95, BCG96]. The covariant (ii) coincides with the invariant for curved edges defined in [TVG99], [GTT01] (see also [MTS+05]). The covariant quantities defined in (iii) on the level lines of u contain the particular case $\int_{\partial X_\lambda} |\kappa(\mathbf{x})|^{1/3} \, d\mathcal{H}^1(\mathbf{x})$ which is the affine arclength parameter which played a fundamental role in the development of affine invariant scale space [AGLM93, ST93, ST94, OST+94]. This quantity has also been used in [BHNR92].

By combining the quantities given above we can get other ones. For instance the quantities

$$\frac{|\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|^m}{\int_{X_\lambda} |\langle D^2 u(\mathbf{y})(J\nabla u(\mathbf{y})), J\nabla u(\mathbf{y})\rangle|^k \, d\mathbf{y}}$$

and

$$\frac{|\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|^m}{\int_{\partial X_\lambda} |\langle D^2 u(\mathbf{y})(J\nabla u(\mathbf{y})), J\nabla u(\mathbf{y})\rangle|^k / |\nabla u(\mathbf{y})| \, d\mathcal{H}^1(\mathbf{y})}$$

are $m - 2k + 1$ affine covariant, $m, k \in \mathbf{R}$. Other examples could be generated.

We describe the behavior of a function $H(u)$ with respect to affine illumination changes. We say that $H(u)$ scales as s^α if $H(su + a) = s^\alpha H(u)$ for any $s > 0$, $a \in \mathbf{R}$. We say that $H(u)$ is illumination invariant with respect to affine changes if $H(u)$ scales as s^0.

The quantity $Q(u)$ is affine invariant and scales as $s^{1/2}$. We may combine the quantities described in the examples above and in Proposition 1 in order to get affine invariant quantities which scale as s^0. For that, let us consider an expression of the form

$$\frac{|\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|^m}{\left(\int_{\partial X_\lambda} |\langle D^2 u(\mathbf{y})(J\nabla u(\mathbf{y})), J\nabla u(\mathbf{y})\rangle|^k / |\nabla u(\mathbf{y})| \, d\mathcal{H}^1(\mathbf{y})\right)^q} \left(\int_{\partial X_\lambda} \frac{|\langle \mathbf{y}, \nabla u(\mathbf{y})\rangle|^\gamma}{|\nabla u(\mathbf{y})|} \, d\mathcal{H}^1(\mathbf{y})\right)^p, \tag{11.2}$$

where $m, k, \gamma, p, q \in \mathbf{R}$. If

$$m - (2k-1)q - p = 0,$$

then the above quantity is affine invariant. If

$$2m - (3k-1)q + (\gamma - 1)p = 0,$$

then the quantity scales as s^0. There are infinitely many solutions of these equations. As examples we can take $m = \frac{1}{2}$, $k = \frac{1}{3}$, $q = \frac{5}{2}$, $\gamma = \frac{1}{4}$, $p = \frac{4}{3}$. Another example is given by $m = \frac{1}{2}$, $k = \frac{1}{2}$, $q = 2$, $\gamma = 1$, $p = \frac{1}{2}$. We notice that there are no non null solutions with $p = 0$.

12 Experimental results

12.1 Selection of descriptors and their implementation

Using the principles described in Section 11, we select in this Section a set of affine invariant quantities, we use them to construct descriptors, and we describe the main details of their implementation. We also describe its corresponding quantized level set (QLS) version. Let us first recall the keypoints for which we will compute our descriptors.

12.1.1 Keypoints detection

Since our purpose is to compare our descriptors with SIFT on normalized neighborhoods [MS04a] and with ASIFT that uses an orbit of images [MY09], we have to work with two types of keypoints. In the first case we use the Harris-Affine keypoints [MS04a] as used in [MS05, MTS+05]. In the second, we use SIFT keypoints [Low04] as in ASIFT. For a short review on these keypoints we refer to Section 10.1.

In order to compute the Harris-Affine keypoints and their SIFT descriptor, we use the online binary software provided by Mikolajczyk (available at the website in [Mika]). We used the updated version of the code on 12-6-2007, under the name Detectors & Descriptors.

In order to compute ASIFT's orbit of images and their SIFT descriptor we use the published ASIFT C++ code [YM11] (available at the website in [YM]).

12.1.2 Descriptors

Let u be an image defined in the domain Ω, a closed rectangle in $\mathbf{R}^2$. Let $\mathbf{x} \in \Omega$ and $\mathcal{N}_0$ be a neighborhood of zero. Assume that $\mathcal{N}_\mathbf{x} = \mathbf{x} + \mathcal{N}_0 \subset \Omega$. As above, X_λ denotes a connected component of $\{u \geq \lambda\}$. In what follows $\mathbf{x}$ represents a keypoint and $\mathbf{y}$ represents a point in $\mathcal{N}_\mathbf{x} \cap \partial X_\lambda$, i.e. it lies on the level lines intersecting the neighborhood of $\mathbf{x}$. As usual $\mathcal{H}^1$, denotes the one-dimensional Hausdorff measure. We assume that ∂X_λ is rectifiable [AFP00].

Using the results of Section 11, we consider the following four quantities in our experiments:

$$r1 = \frac{|\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|}{|< D^2u(\mathbf{x})(J\nabla u(\mathbf{x})), J\nabla u(\mathbf{x}) >|^{\frac{1}{2}}},$$

$$r2 = \frac{|\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|}{|< D^2u(\mathbf{y})(J\nabla u(\mathbf{y})), J\nabla u(\mathbf{y}) >|^{\frac{1}{2}}},$$

$$r3 = \frac{|\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|^{\frac{1}{2}}}{(\int_{\partial X_\lambda} |< D^2u(\bar{\mathbf{y}})(J\nabla u(\bar{\mathbf{y}})), J\nabla u(\bar{\mathbf{y}}) >|^{\frac{1}{2}} / |\nabla u(\bar{\mathbf{y}})| \, d\mathcal{H}^1(\bar{\mathbf{y}}))^2} (\int_{\partial X_\lambda} |\bar{\mathbf{y}} \cdot \frac{\nabla u(\bar{\mathbf{y}})}{|\nabla u(\bar{\mathbf{y}})|}| \, d\mathcal{H}^1(\bar{\mathbf{y}}))^{\frac{1}{2}},$$

$$r4 = \frac{|\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})|^{\frac{1}{2}}}{(\int_{\partial X_\lambda} |< D^2u(\bar{\mathbf{y}})(J\nabla u(\bar{\mathbf{y}})), J\nabla u(\bar{\mathbf{y}}) >|^{\frac{1}{3}} / |\nabla u(\bar{\mathbf{y}})| \, d\mathcal{H}^1(\bar{\mathbf{y}}))^{\frac{5}{2}}} (\int_{\partial X_\lambda} |\bar{\mathbf{y}} \cdot \frac{\nabla u(\bar{\mathbf{y}})}{|\nabla u(\bar{\mathbf{y}})|}|^{\frac{1}{4}} \, d\mathcal{H}^1(\bar{\mathbf{y}}))^{\frac{4}{3}}.$$

Let us mention that in all the above quantities, the numerator contains the expression $H_{10}(u, \mathbf{x}, \mathbf{y}) = \nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})$. Thus, the sinus of the angle formed by the two vectors $\nabla u(\mathbf{x})$ and $\nabla u(\mathbf{y})$ appears in all of them. Since this angle is an essential ingredient of SIFT, in some sense we are adding a further justification to it.

The first and second derivatives used by these quantities are computed using centered differences. The level-lines of the image are computed using the tree of shapes [MG00, CM10]. In all the quantities we compute, we set a minimum value $\rho = 10^{-3}$ for any denominator quantity q. That is, if $q < \rho$ then we set the quantity $ri = 0$.

Now, to build up a descriptor with the above quantities we use the same structure as SIFT. Given a keypoint $\mathbf{x}$, we consider a neighborhood $\mathcal{N}_\mathbf{x}$ of $\mathbf{x}$ (of size 16×16 in common SIFT and of size 41×41 in SIFT+NN) and we divide it into 4×4 blocks. For each block we compute a weighted histogram of directions quantized in 8 angular bins (and referred to the dominant orientation at $\mathbf{x}$). In SIFT, the weights are given by the magnitude of the gradient. This time the weights are given by the descriptors ri, $i = 1, 2, 3, 4$. Each coordinate of $v \in \mathbf{R}^{128}$ contains the weight contributions of all pixels of a given block with a given angular orientation. Finally, in practice, to avoid quantization effects, each orientation occurrence is distributed in several neighboring bins. As we shall verify, using the same arrangement of the descriptor as in SIFT, these quantities permit to improve the results obtained with it.

As in the Introduction, we refer to these descriptors as $\mathcal{AD}$.

Let us mention that the computation times for our descriptors are essentially the same as for SIFT. Indeed, the only difference is that now for each keypoint $\mathbf{x}$ we need to compute quantities like $\nabla u(\mathbf{x}) \wedge \nabla u(\mathbf{y})$ where $\mathbf{y} \in \mathcal{N}_\mathbf{x}$. This amounts to a constant additional number of operations per keypoint, which does not increase the complexity with respect to SIFT. To give an example, let us mention the running times corresponding to the computation of our descriptors on the Harris-Affine keypoints of image 12.2(e). All experiments have been run on a computer with a CPU speed of 2.66 GHz. In our implementation, which is not optimized, SIFT takes 49 seconds, $r1$ and $r2$ take 50 seconds, and $r3$ and $r4$ take 83 seconds. Please note that the time difference between $r1$, $r2$, and SIFT is negligible. The extra time consumed by $r3$ and $r4$ is due to the computation of the

level lines of the image (this could be optimized by pre-computing and storing them).

12.1.3 The QLS version of the descriptors

Let us introduce a further variant based on the observation that level sets are affine covariant domains in the sense that, if u is a given image and $A \in GL(2,\mathbf{R})^+$, then $\{u_A \geq \lambda\} = A^{-1}\{u \geq \lambda\}$. Assume that $\mathbf{x}$ is a keypoint of an image u and $\mathcal{N}_\mathbf{x}$ is a given neighborhood (be it square or the affine normalized one). We want to organize the descriptor taking into account the level set structure of u in $\mathcal{N}_\mathbf{x}$. For that we quantize the image u in $\mathcal{N}_\mathbf{x}$. To avoid the effect of illumination changes we first equalize the image u in $\mathcal{N}_\mathbf{x}$, call it $H_\mathbf{x}(u)$, and then define the (bi)level sets $\mathcal{N}_\mathbf{x}^{u,j} := \{\mathbf{y} \in \mathcal{N}_\mathbf{x} : j\Delta \leq H_\mathbf{x}(u)(\mathbf{y}) < (j+1)\Delta\}$ where Δ is a quantization step and $j = 0, 1, ..., \frac{256}{\Delta} - 1$. In practice we take $\Delta = 64$ and we have 4 level sets corresponding to $j = 0, 1, 2, 3$. The descriptor associated to each *ri* ($i = 1, 2, 3, 4$) is formed by the concatenation of four vectors $v_j \in \mathbf{R}^{128}$. Each coordinate of v_j receives the weights of the pixels $\mathbf{y} \in \mathcal{N}_\mathbf{x}^{u,j}$. If $\Delta = 256$, then we have only a vector $v \in \mathbf{R}^{128}$ with the standard organization of SIFT.

As in the Introduction, we refer to these descriptors as $\mathcal{AD}$+QLS.

12.2 Experiments

Our purpose in this Section is to compare our descriptors *ri* with SIFT on affine normalized neighborhoods [MS04a] and with ASIFT [MY09]. This is the object of Sections 12.2.5 and 12.2.6, respectively. We also include in Section 12.2.4 an example of comparison with SIFT on standard neighborhoods. The comparisons will be done using the standard and the QLS versions both for SIFT and for *ri*.

Let us consider two images $\mathbf{I_l}$ and $\mathbf{I_r}$ defined in the domain Ω. Assume that both contain the image of a planar scene so that there is an homography $\mathcal{H}$ such that $\mathbf{I_r}(\mathbf{x}) = \mathbf{I_l}(\mathcal{H}\mathbf{x})$ for $\mathbf{x} \in \hat{\Omega} \subseteq \Omega$. Since $\mathcal{H}$ can be locally approximated by an affine map, we may assume that $\mathcal{H}$ is an affine transformation. Let us denote by $\mathbf{P_l}$ and $\mathbf{P_r}$ the set of keypoints (see Section 12.1.1) of $\mathbf{I_l}$ and $\mathbf{I_r}$, respectively. Notice that $\mathbf{P_l}$ and $\mathbf{P_r}$ may have different numbers of keypoints, so that there will be keypoints in $\mathbf{I_l}$ without a corresponding one in $\mathbf{I_r}$ and conversely.

Finally, to each keypoint we associate a descriptor which is a vector of 128 coordinates, or of 512 in the case of the QLS versions. The descriptors we consider are SIFT and the descriptors *ri*, $i = 1, 2, 3, 4$, in their $\mathcal{AD}$ or $\mathcal{AD}$+QLS versions as defined above. Then for each keypoint $p_l \in \mathbf{P_l}$ we look for a matching point $p_r \in \mathbf{P_r}$ using a certain matching strategy. Several of them have been used for performance evaluation [MS05, MTS+05].

To compare our descriptors with SIFT+NN we follow the experimental protocol proposed in [MS05, MTS+05]. For that, we first describe the matching strategies (Section 12.2.1), the notion of corresponding regions (Section 12.2.2), and the *precision/recall* curves (Section 12.2.3). The comparison with ASIFT will

be done in terms of the significance measure proposed in [MY09] (see Section 12.2.6).

12.2.1 Matching strategies

The definition of a match depends on the matching strategy. Assume that to any region A of any of the images $\mathbf{I_l}$ and $\mathbf{I_r}$ we may associate a descriptor D_A, that is, a vector in $\mathbf{R}^N$ for some fixed $N \in \mathbf{N}$. We look into three different matching strategies as proposed in [MS05]:

1. Threshold based matching (*TH*): Two regions A of $\mathbf{I_l}$ and B of $\mathbf{I_r}$ are matched if the distance between their descriptors is below a certain threshold. In this strategy, a descriptor can have several matches and several of them can be considered as correct (in the sense that the descriptors are really similar).

2. Nearest neighbor based matching (*NN*): Two regions A of $\mathbf{I_l}$ and B of $\mathbf{I_r}$ are matched if the descriptor D_B is the nearest neighbor to the descriptor D_A and if the distance $d(D_A, D_B) < threshold$. Please note here that a descriptor can have only one match.

3. Nearest neighbor distance ratio matching (*RNN*): This strategy, introduced in [Low04], is similar to *NN* except that the thresholding is applied to the distance ratio between the first and the second nearest neighbor. With the same example and notation used in *NN*, let D_C be the second nearest neighbor to D_A, then region A is matched to B if $\frac{||D_A - D_B||}{||D_A - D_C||} < threshold$ where $||\cdot||$ is the Euclidean norm. Note that in this case a descriptor can have only one match.

Note that the NN and RNN matchings are not symmetric concepts with respect to $\mathbf{I_l}$ and $\mathbf{I_r}$. They are computed taking $\mathbf{I_l}$ as a reference image.

12.2.2 Definition of corresponding regions

In this subsection and the next one we are in the context of the comparison of our descriptors with SIFT+NN. Thus we work with Harris-Affine keypoints with their associated elliptic neighborhood [MS04a]. We also use the terms elliptical region or, simply, region. Assume that we have two keypoints $p_l \in \mathbf{P_l}$ and $p_r \in \mathbf{P_r}$ whose elliptical neighborhoods S_{μ_l} and S_{μ_r} are defined by the shape adaptation matrices μ_l and μ_r, respectively. Let $S_{\mathcal{H}^T \mu_l \mathcal{H}}$ be the image by $\mathcal{H}$ of the elliptical region S_{μ_l}. The two regions S_{μ_l} and S_{μ_r} are said to correspond [MS05, MTS+05] if the *overlap error* is sufficiently small, that is, if

$$1 - \frac{|S_{\mu_r} \cap S_{\mathcal{H}^T \mu_l \mathcal{H}}|}{|S_{\mu_r} \cup S_{\mathcal{H}^T \mu_l \mathcal{H}}|} < \epsilon. \tag{12.1}$$

Given the homography and the matrices defining the regions, the error is computed numerically by counting the number of pixels in the union and in the

intersection of the regions (see [MS05, MTS+05] for details). In our experiment we choose $\epsilon = 0.5$ as suggested in [MS05, MTS+05].

12.2.3 Definition of precision and recall

Assume that we take $\mathbf{I_l}$ as reference image and the matches are computed accordingly. The evaluation criterion used is based on the number of correct matches (true positives TP) and the number of false matches (false positives FP) obtained for an image pair. Given a matching strategy, we have a correct match of two keypoints when their descriptors satisfy the matching criterion and their elliptical regions correspond according to the overlap criterion (12.1). The other matchings are the false matchings. Both correct and false matchings can be computed since we know the ground truth given by the matrix $\mathcal{H}$. Following [MS05, MTS+05] we take the overlap error threshold $\epsilon = 0.5$. As argued in that references, there are very few regions that should be matched, have an overlap error greater than 0.5 and pass the matching criterion.

The number of correspondences is counted as the number of possible correct matchings and depends on the matching strategy. Let us explain this. For any region A of $\mathbf{I_l}$, let $N(A)$ be the number of regions of $\mathbf{I_r}$ that correspond to A according to the overlap criterion (12.1). Then, if we use the TH matching strategy, we compute the number of correspondences as $\sum_A N(A)$, while if we use the NN or the RNN matching strategies, we compute it as $\sum_A \min(N(A), 1)$. In all cases the sum is extended to all regions of $\mathbf{I_l}$.

According to the three matching strategies discussed in Section 12.2.1, to match a region A of $\mathbf{I_l}$ to a region B of $\mathbf{I_r}$ a certain distance relation between their descriptors D_A and D_B has to be below a given threshold t. So, given the two images $\mathbf{I_l}$ and $\mathbf{I_r}$ and the threshold t, we compare every descriptor D_l of keypoints in $\mathbf{I_l}$ to every descriptor D_r of keypoints in $\mathbf{I_r}$ and we count the number of TP as well as the number of FP. We repeat this process for different values of t and, in this way, we can study the behavior of the descriptor for different thresholds. Performance of different descriptors is measured using *recall* versus *1-precision* graphs where *recall* and *1-precision* are defined as follows:

- $recall = \frac{\#correct-matches}{\#correspondences}$, where # is read as "the number of".
- $1 - precision = \frac{\#false-matches}{\#correct-matches+\#false-matches} = \frac{FP}{TP+FP}$. Note that the denominator does not take into consideration the *overlap error*. In other words, if the matching algorithm returned k matchings, then $TP + FP = k$ (i.e. $TP + FP$ is independent of the number of correspondences).

Both values depend on t. Please note that *recall* and $1 - precision$ are *independent* terms: *recall* is computed with respect to the number of correspondences and *1-precision* is computed with respect to the total number of matches returned by the matching algorithm. Now, given *recall* and *1-precision* measures along with the number of correspondences, we have:

$$TP = \#correspondences \times recall,$$

$$FP = \frac{\#correspondences \times recall \times (1 - precision)}{precision}.$$

Remark 3. Note that the *recall* is increasing with t, as is the number of matchings. But this is not the case for *1-precision*. Thus the *1-precision/recall* curve is not necessarily increasing, although it is often so.

A perfect descriptor would give a recall equal to 1 for any precision. A horizontal curve in the graph indicates that the recall value is attained with high precision. It also indicates that the detected structures are very similar to each other and the descriptor cannot distinguish them even when decreasing the precision.

12.2.4 An example of comparison with SIFT on standard neighborhoods

Although our main comparisons will be done with SIFT+NN [MS04a, MTS+05] and with ASIFT [MY09], which are SIFTs most performant versions, for the sake of illustration let us show an image to compare SIFT with the descriptors in $\mathcal{AD}$ and $\mathcal{AD}$+QLS. In both cases, we use a square neighborhood. We have chosen the descriptor based on $r1$, although we could have chosen any of the ri. In Figure 12.1 we show the result of a matching between images 12.2(d) and 12.2(e) (taken from [MS05], see Section 12.2.5). It can be seen that although the quantity $r1$ already contributes to improve the matching result, when combined with the quantization on the level sets we even get a more robust descriptor (having less false matchings). In this specific example the number of correct matchings increased from 45 for SIFT to 68 $\mathcal{AD}$+QLS, whereas the number of false matchings dropped from 33 (SIFT) to 9 ($\mathcal{AD}$+QLS). This behavior is common to all images of the dataset below. Quantitative comparisons will be done in subsequent Sections.

12.2.5 Comparing to SIFT with normalized neighborhoods

In this Section we compare our descriptors with SIFT on affine normalized neighborhoods around the Harris affine keypoints [MS04a, MTS+05].

Image dataset The images we use are taken from [MS05]. They can be downloaded from [Mikb]. Without loss of generality, and in order to reduce the running time of the experiments (mostly due to the generation of Figures 12.3 to 12.7 using the MATLAB code provided in the above website, whose execution time depends on the number of keypoints), we downscale the images by a factor of 2.

We chose four sets of images, three of them containing a reference image and two simulated affine distortions. The fourth set contains the reference image and two simulated illumination changes. The first set contains different views of a textured scene, again we compare the frontal view 12.2(a) with a 50° and a 70° tilt viewpoint changes, shown in 12.2(b) and 12.2(c). The second set contains different views of a structured scene, we compare the frontal

(a) (b) (c)

Figure 12.1: The result of a matching between images 12.2(d) and 12.2(e) using SIFT keypoints. The descriptors are therefore computed on a square neighborhood. The matching has been done using the RNN matching strategy with a threshold equal to 0.8. 12.1(a) shows the result of the SIFT descriptor with 45 correct matchings and 33 false ones. 12.1(b) shows the result of the $r1$ quantity used in a descriptor structure analogous to SIFT with 56 correct matchings and 22 false ones. 12.1(c) shows the results of the $r1$ quantity used in the descriptor structure based on the level sets as described in Section 12.1.3, with 68 correct matchings and 9 false ones.

view 12.2(d) with a 50° and 70° tilt viewpoint changes, shown in 12.2(e) and 12.2(f), respectively. The third set contains simulations of rotations and zooms, we compare 12.2(g) with 12.2(h) and then with 12.2(i). Finally, the fourth set contains a reference image and two simulations of illumination changes, we compare 12.2(j) with 12.2(k) and then with 12.2(l). More information about the set of images and their acquisition can be found in [MTS+05].

The number of Harris affine (HA) keypoints of each image is shown in Table 12.1. The number of *correspondences* between each image pair compared is shown in Table 12.2. We have used an *overlap error* parameter $\epsilon = 0.5$ and the three matching strategies {RNN, NN, TH}. Note that, according to its definition in Section 12.2.3, the number of correspondences is the same for the matching strategies *NN* and *RNN*, and is much higher for *TH*.

Experiments First we compare our descriptors $\mathcal{AD}$ with SIFT+NN using in both cases the affine normalized neighborhood. We display the $1 - precision/recall$ curves for the four images in Figure 12.2 for all matching strategies.

Image	Nb of HA Keypoints
12.2(a)	5027
12.2(b)	4120
12.2(c)	3930
12.2(d)	2325
12.2(e)	2675
12.2(f)	2504
12.2(g)	2822
12.2(h)	1748
12.2(i)	1481
12.2(j)	1730
12.2(k)	1334
12.2(l)	1039

Table 12.1: Table showing the number of Harris-Affine keypoints found for every image used in the experiments.

Image pair	*#correspondences*		
	TH	NN	RNN
12.2(a)→12.2(b)	46414	2821	2821
12.2(a)→12.2(c)	23631	1633	1633
12.2(d)→12.2(e)	13016	990	990
12.2(d)→12.2(f)	5520	493	493
12.2(g)→12.2(h)	10189	739	739
12.2(g)→12.2(i)	3840	288	288
12.2(j)→12.2(k)	12012	1161	1161
12.2(j)→12.2(l)	9686	929	929

Table 12.2: Table showing the *#correspondences* between every image pair used in the experiments and for all matching strategies. Note that, according to its definition in Section 12.2.3, the *#correspondences* is the same for the matching strategies *NN* and *RNN*, and is much higher for *TH*.

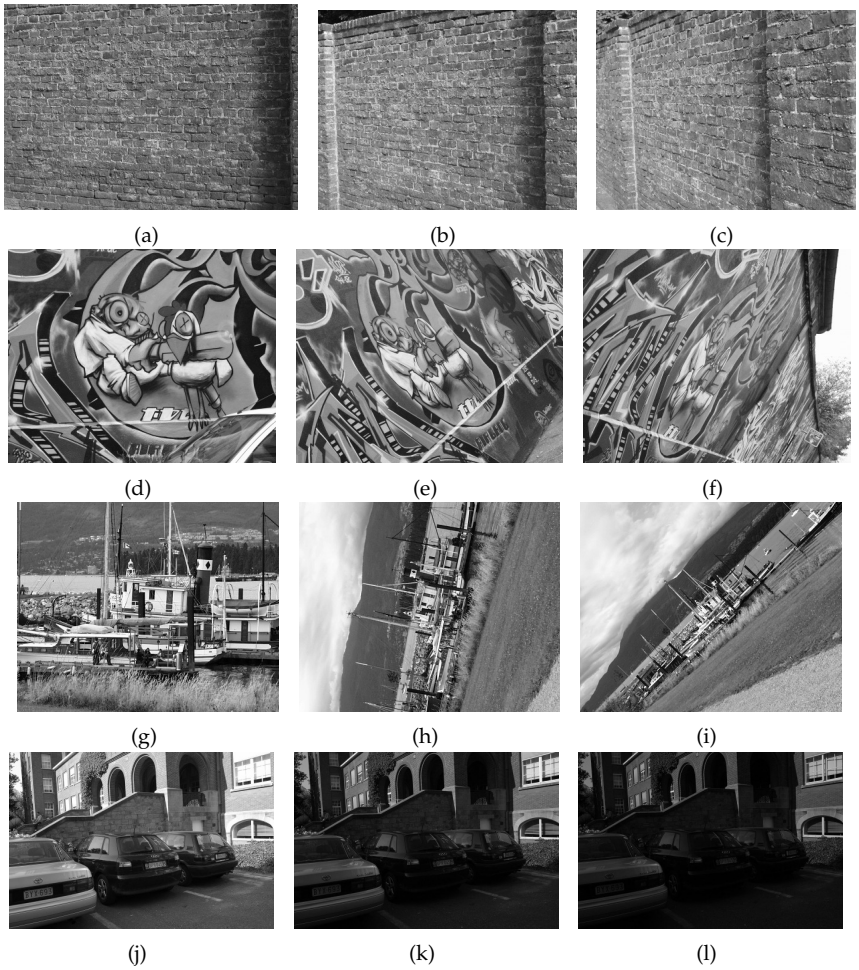


Figure 12.2: The images used to compare $\mathcal{AD}$ and $\mathcal{AD}$+QLS to SIFT+NN. A textured scene where 12.2(a) front view, 12.2(b) an approximately 50° change in viewpoint, and 12.2(c) an approximately 70° change in viewpoint. A structured scene where 12.2(d) front view, 12.2(e) an approximately 50° change in viewpoint, and 12.2(f) an approximately 70° change in viewpoint. 12.2(g), 12.2(h) and 12.2(i) increasing rotation and zoom factors. 12.2(j), 12.2(k) and 12.2(l) increasing illumination change.

In a second experiment we compare $\mathcal{AD}$, $\mathcal{AD}$+QLS and SIFT+NN (using again the normalized neighborhood). The purpose of it is to compare the performance added by the QLS strategy. Since all quantities *ri* behave similarly under the different matching strategies, we just show the results obtained with $r1$ and RNN.

In any case, for each pair of images and for each *matching strategy* we com-

Sample Experiment Input: Figures 12.2(d) and 12.2(e) Chosen matching strategy: Nearest Neighbor distance ratio matching (RNN) Other chosen values: *overlap error* = 0.5 (yielding. *#correspondences* = 990)											
Threshold $t =$		0.3704	0.4167	0.4762	0.5556	0.6667	0.8333	0.8696	0.9091	0.9524	1.0000
SIFT	TP	0	0	0	1	6	67	87	118	163	249
	TP + FP	0	0	1	3	10	119	194	332	655	2325
r1	TP	1	2	2	5	15	102	140	189	271	460
	TP + FP	1	2	2	5	15	109	162	248	474	2325
r2	TP	1	2	2	3	12	93	130	180	274	464
	TP + FP	1	2	2	3	12	106	152	237	461	2325
r3	TP	0	2	2	5	19	98	134	183	276	464
	TP + FP	0	2	2	5	19	109	155	236	480	2325
r4	TP	0	2	2	5	20	96	129	183	270	465
	TP + FP	0	2	2	5	20	108	152	243	465	2325

Table 12.3: Results of an experiment where the input images are Figure 12.2(d) and Figure 12.2(e), comparing our 4 $\mathcal{AD}$ descriptors to SIFT, using in both cases an affine normalized neighborhood.

pute the number of correspondences, the number of true positives (TP) and the total number of returned matches (TP+FP) varying the threshold t. We used the same code used in [MS05] which can be found at [Mikc]

Let us show the Figures corresponding to the the comparison of $\mathcal{AD}$ with SIFT+NN. Figure 12.3 shows the $1 - precision/recall$ curves corresponding to the matching of images 12.2(a) with 12.2(b) (left column) and of 12.2(a) with 12.2(c) (right column). In each column we show the results corresponding to the three different matching strategies $\{TH, NN, RNN\}$. Figure 12.4 shows the results corresponding to the matching of images 12.2(d) with 12.2(e) and of 12.2(d) with 12.2(f). Figure 12.5 shows the results corresponding to the matching of images 12.2(g) and 12.2(h), and images 12.2(g) and 12.2(i). Figure 12.6 shows the result of matching images 12.2(j) and 12.2(k), and images 12.2(j) and 12.2(l).

In general, our descriptors perform better than SIFT, in particular in Figure 12.4(c) there is an improvement by a factor of 2. Notice that Figure 12.4(d) (which corresponds to a large change of viewpoint) shows that when SIFT could retrieve only 5% of TP at a low precision, we retrieve up to 15% of TP. Let us note that, in most cases, the ranking of the descriptors does not change between one matching strategy and another.

In order to illustrate how the curves displayed in the above figures are generated, we show Table 12.3. To generate these curves, we need the number of TP and TP+FP for each experiment. Table 12.3 contains these numbers for one

of the experiments of Figure 12.4, namely the matching of Figure 12.2(d) to Figure 12.2(e) using the RNN *matching strategy*.

In order to show the relative performance of $\mathcal{AD}$ and $\mathcal{AD}$+QLS, we show in Figure 12.7 the 1-precision/recall curves corresponding to all images in Figure 12.2 for $\mathcal{AD}$, $\mathcal{AD}$+QLS, and SIFT+NN. Again, our descriptors are based on $r1$ and the RNN matching strategy. As it can be seen, in some cases, specially when there are big affine changes, the $\mathcal{AD}$+QLS version provides improvements over $\mathcal{AD}$.

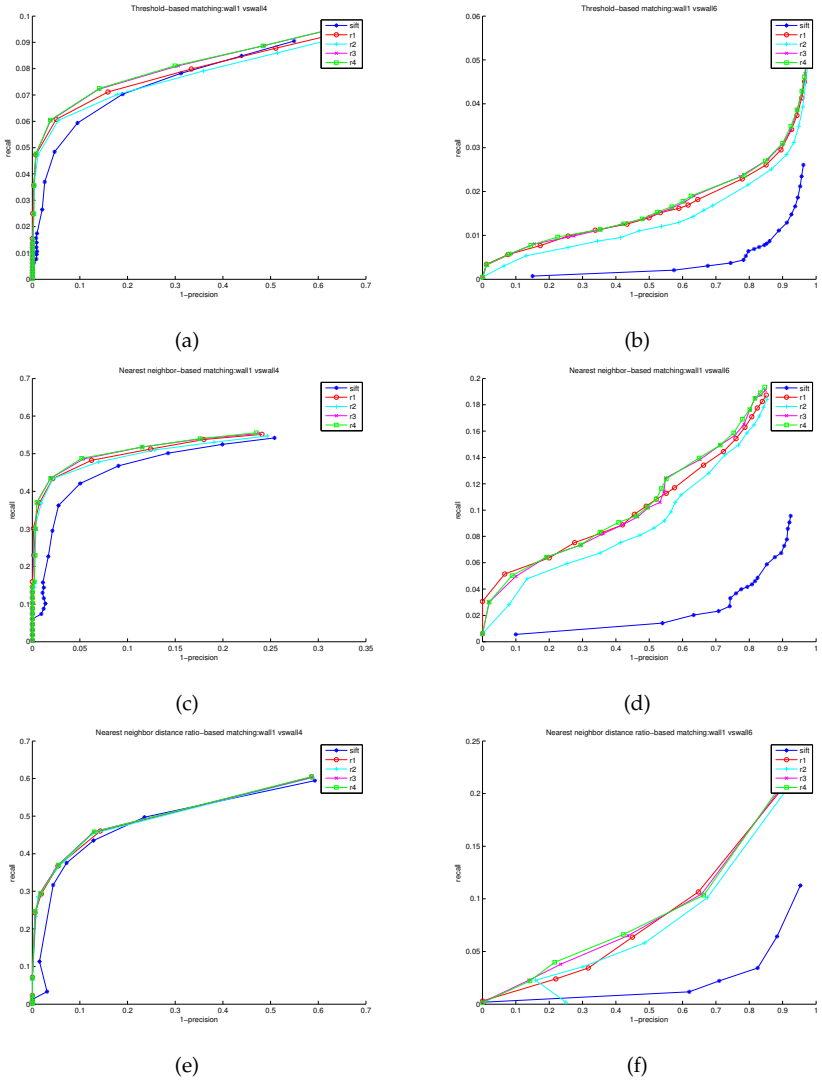
12.2.6 Comparing to ASIFT

In this experiment we compare our descriptors $\mathcal{AD}$ with SIFT in the context of ASIFT [MY09] (available at [YM]). Thus we match $\mathbf{I}_l$ with $\mathbf{I}_r$ using the orbit of images generated by ASIFT for both of them. To test the robustness of SIFT and our descriptors we compare them on different orbit sizes. We first start with the suggested orbit size (ASIFT default) made out of 7 tilts and generating 61 images. Then we reduce the orbit to 5 tilts (27 simulations) and 3 tilts (10 simulations) Throughout these experiments we use the $r1$ quantity. The other quantities behave similarly. Both SIFT and our descriptors are computed on the same set of SIFT keypoints (see Section 10.1). For both we use the RNN matching strategy. Thus, for both of them the conditions are equal, except that SIFT is used for ASIFT and we use $\mathcal{AD}$ based on $r1$.

The tests are performed on five image pairs, shown in Figure 12.8. We compare the number of matches obtained with each method and also the significance of those matches. The significance measure is computed as the logarithm of the NFA (number of false alarms), a quantity obtained during the computation of the affine map relating both images. The affine map is computed using an *a contrario* optimized version of RANSAC, as in [MS04b]. We use the same implementation as in ASIFT.

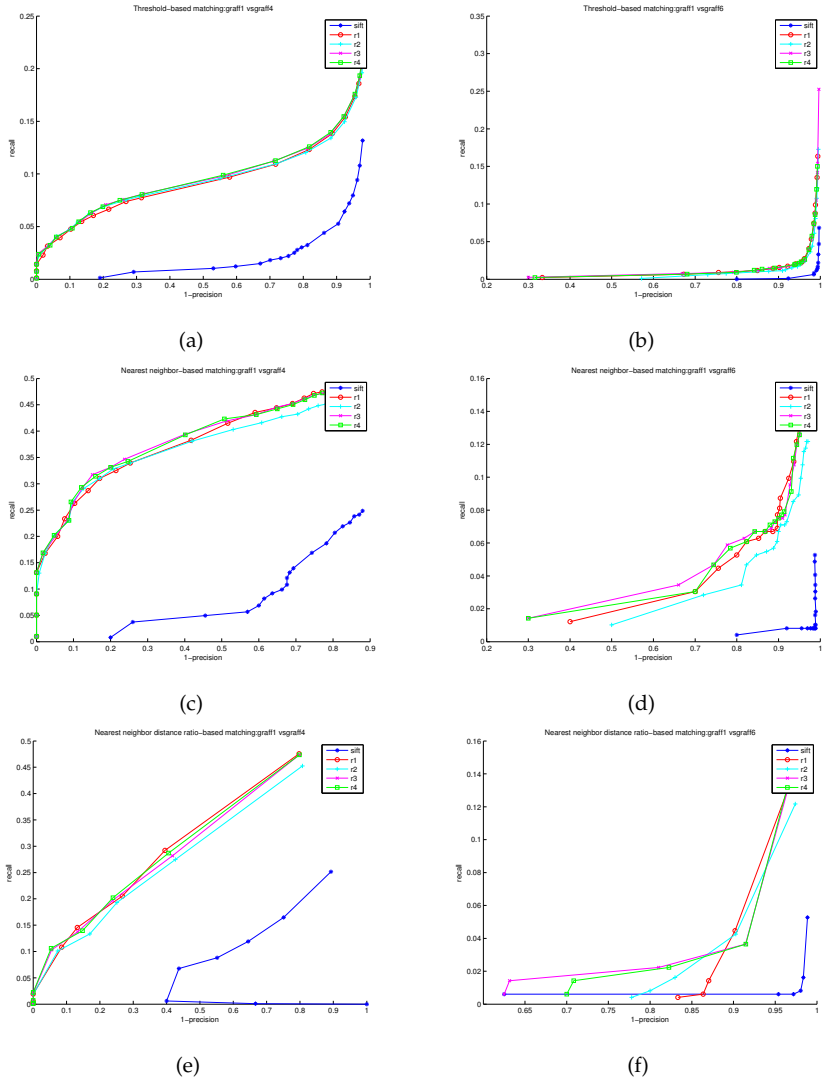
In Table 12.4 we show the results of the comparison between ASIFT and the descriptor $\mathcal{AD}$ based on $r1$ computed on a square neighborhood. We show the results corresponding to three orbit sizes. We can see the improvement due to the use of the new descriptor both in the number of matchings and in their significance measure. This improvement is clearly visible for a reduced orbit, when using $r1$ we get a matching in four of five images, but only in two of them when using SIFT.

The performance of $\mathcal{AD}$+QLS versus ASIFT is similar to the the performance of $\mathcal{AD}$ versus ASIFT, and therefore we omitted the Table. Let us only mention that when using $\mathcal{AD}$+QLS we get a matching for the five image pairs in the case of a reduced orbit with 10 simulated images. On the other hand, the reason for the similar behavior of $\mathcal{AD}$ versus $\mathcal{AD}$+QLS in the context of this comparison may be that the information brought by the orbit is sufficient to cancel the benefits gained by the QLS strategy. This is in contrast with the behavior of QLS in the context of SIFT+NN.



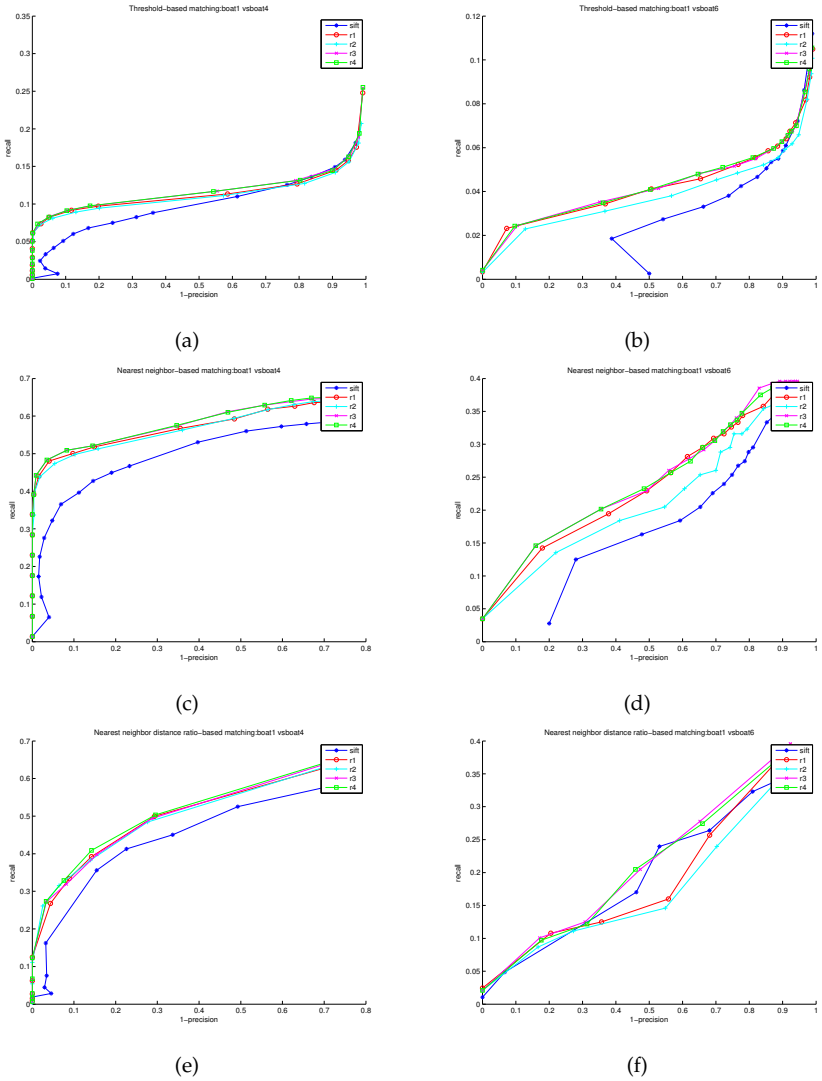
(a) (b) (c) (d) (e) (f)

Figure 12.3: Comparison between SIFT+NN and our descriptors $\mathcal{AD}$ computed on an affine normalized neighborhood. '*1-precision* vs, *recall*' graphs showing the results of matching 12.2(a) with 12.2(b) and 12.2(a) with 12.2(c), in the first and second columns, respectively. The left column shows the matching 12.2(a) with 12.2(b): 12.3(a) using the TH matching strategy, 12.3(c) using the NN matching strategy, and 12.3(e) using the RNN matching strategy. The right column shows the matching 12.2(a) with 12.2(c): 12.3(b) using the TH matching strategy, 12.3(d) using the NN matching strategy, and 12.3(f) using the RNN matching strategy



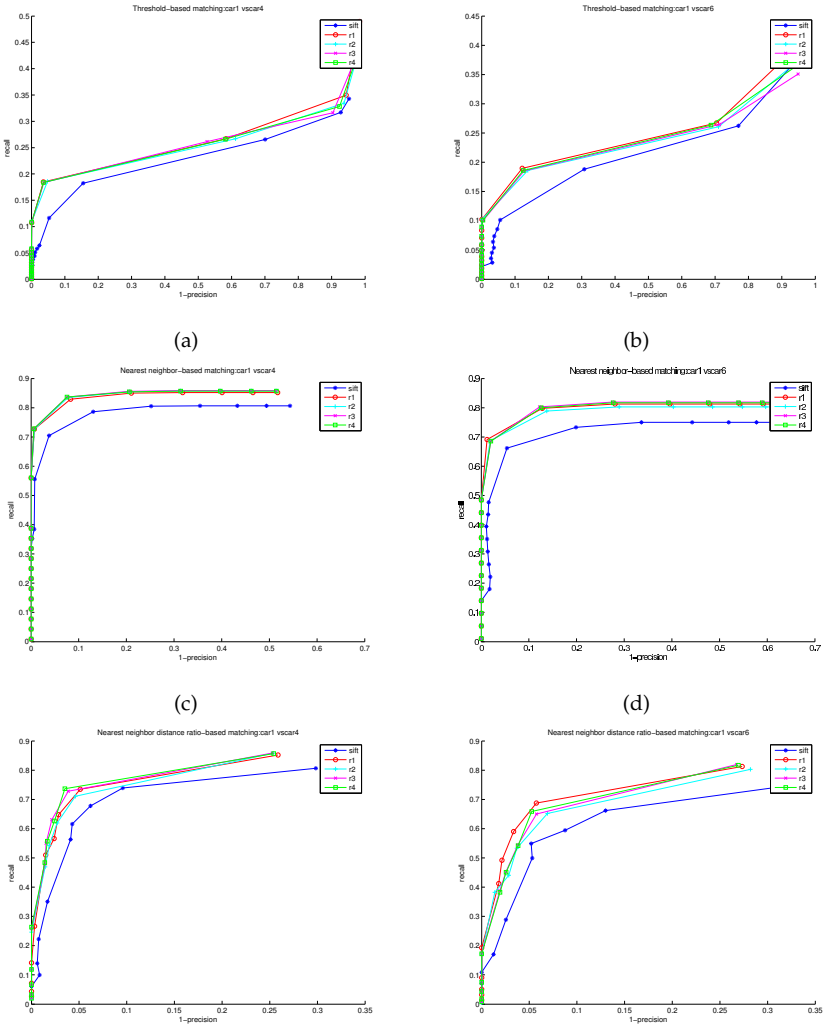
(a) (b) (c) (d) (e) (f)

Figure 12.4: Comparison between SIFT+NN and our descriptors $\mathcal{AD}$ computed on an affine normalized neighborhood. '*1-precision* vs, *recall*' graphs showing the results of matching 12.2(d) with 12.2(e) and 12.2(d) with 12.2(f), in the first and second columns, respectively. The left column shows the matching 12.2(d) with 12.2(e): 12.4(a) using the TH matching strategy, 12.4(c) using the NN matching strategy, and 12.4(e) using the RNN matching strategy. The right column shows the matching 12.2(d) with 12.2(f): 12.4(b) using the TH matching strategy, 12.4(d) using the NN matching strategy, and 12.4(f) using the RNN matching strategy



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

Figure 12.5: Comparison between SIFT+NN and our descriptors $\mathcal{AD}$ computed on an affine normalized neighborhood. '*1-precision* vs, *recall*' graphs showing the results of matching 12.2(g) with 12.2(h) and 12.2(g) with 12.2(i), in the first and second columns, respectively. The left column shows the matching 12.2(g) with 12.2(h): 12.5(a) using the TH matching strategy, 12.5(c) using the NN matching strategy, and 12.5(e) using the RNN matching strategy. The right column shows the matching 12.2(g) with 12.2(i): 12.5(b) using the TH matching strategy, 12.5(d) using the NN matching strategy, and 12.5(f) using the RNN matching strategy



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

Figure 12.6: Comparison between SIFT+NN and our descriptors $\mathcal{AD}$ computed on an affine normalized neighborhood. *'1-precision* vs, *recall'* graphs showing the results of matching 12.2(j) with 12.2(k) and 12.2(j) with 12.2(l), in the first and second columns, respectively. The left column shows the matching 12.2(j) with 12.2(k): 12.6(a) using the TH matching strategy, 12.6(c) using the NN matching strategy, and 12.6(e) using the RNN matching strategy. The right column shows the matching 12.2(j) with 12.2(l): 12.6(b) using the TH matching strategy, 12.6(d) using the NN matching strategy, and 12.6(f) using the RNN matching strategy

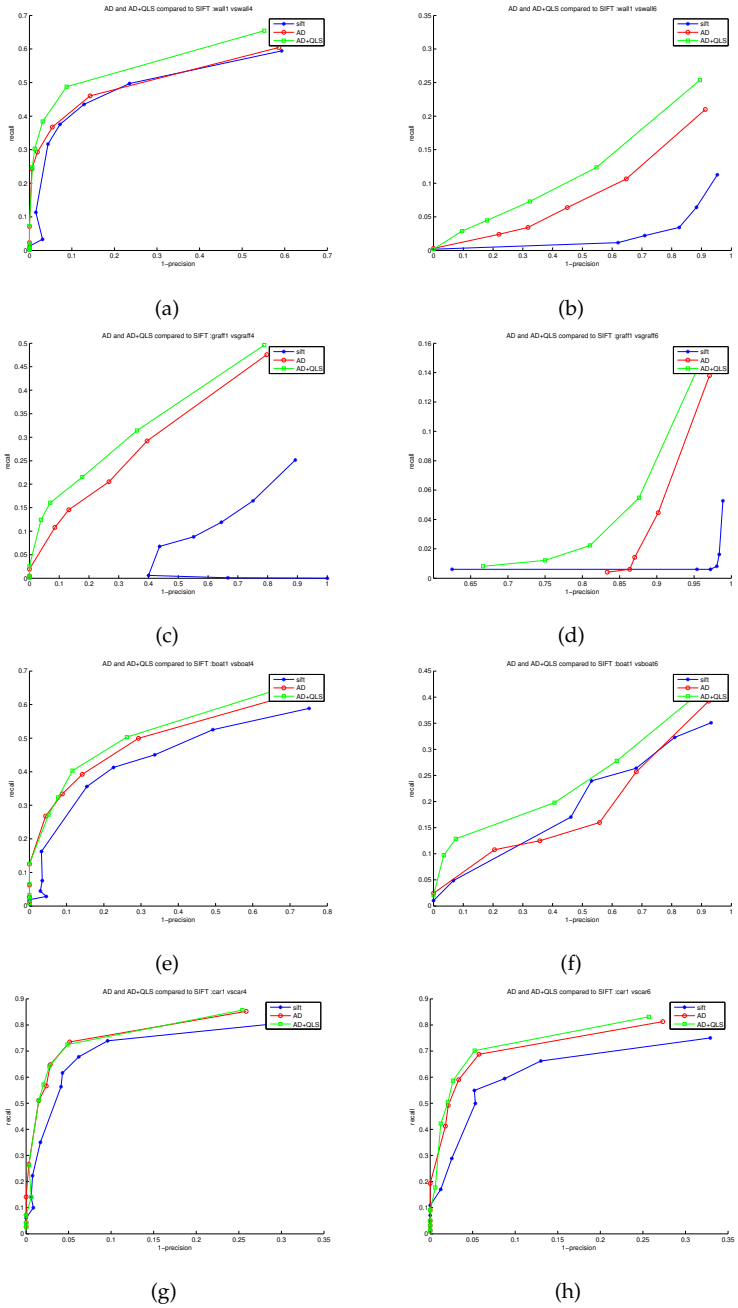
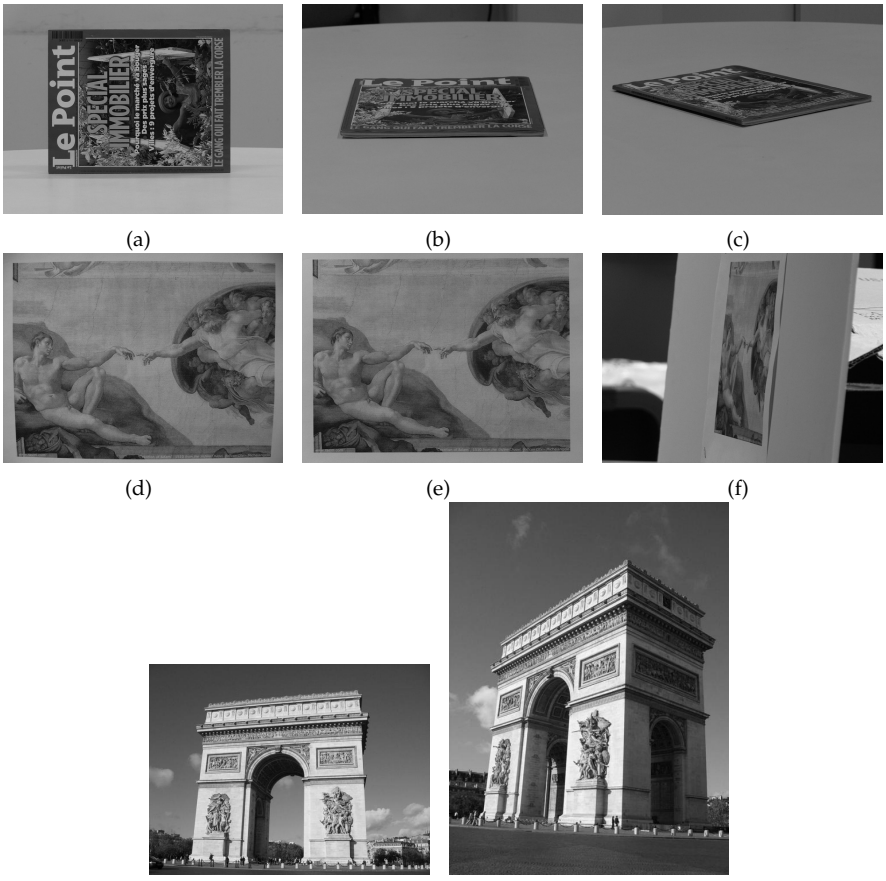


Figure 12.7: Comparison between SIFT+NN, and the descriptors $\mathcal{AD}$ and $\mathcal{AD}$+QLS based on $r1$. *'1-precision* vs, *recall'* graphs showing the results of matching using the RNN matching strategy. 12.7(a) and 12.7(b) show the result of matching 12.2(a) with 12.2(b) and 12.2(c) respectively. 12.7(c) and 12.7(d) show the result of matching 12.2(d) with 12.2(e) and 12.2(f) respectively. 12.7(e) and 12.7(f) show the result of matching 12.2(g) with 12.2(h) and 12.2(i) respectively. 12.7(g) and 12.7(h) show the result of matching 12.2(j) with 12.2(k) and 12.2(l) respectively.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Figure 12.8: Image pairs used as an input in the experiments comparing our descriptors to SIFT in the context of ASIFT. 12.8(a) frontal view of a magazine; 12.8(b) a transition tilt of 4 with a 90 degree rotation applied to the magazine; 12.8(c) a transition tilt of 4 with a 50 degree rotation applied to the magazine; 12.8(d) an image of a painting with no optical zoom; 12.8(e) image of the same painting with 10x optical zoom; 12.8(f) an 80° viewpoint change to the painting image with 10x optical zoom; finally, 12.8(g) and 12.8(h) illustrate a case of two images taken from different viewpoints of a real big 3D object.

Image Pair	Descriptor	Number Of Tilts					
		7 (61 simulations)		5 (27 simulations)		3 (10 simulations)	
		Matches	log(nfa)	Matches	log(nfa)	Matches	log(nfa)
12.8(a)→ 12.8(b)	SIFT	234	-322.685	140	-181.15	14	-4.33
	r1	309	-418.38	182	-247.03	24	-16.7
12.8(a)→ 12.8(c)	SIFT	98	-125.84	68	-87.72	No match	0
	r1	145	-192.489	101	-129.25	12	-2.45
12.8(g)→ 12.8(h)	SIFT	173	-185.27	113	-120.57	65	-79.05
	r1	318	-373.025	349	-417.36	181	-226.47
12.8(d)→ 12.8(f)	SIFT	53	-54.14	29	-26.2	No match	0
	r1	67	-71.95	63	-57.1	No match	0
12.8(e)→ 12.8(f)	SIFT	62	-65.23	42	-48.08	No match	0
	r1	97	-96.31	76	-69.49	14	-3.34

Table 12.4: Table comparing the result of matching an image pair using SIFT computed on an orbit of images and the $\mathcal{AD}$ based on r1 descriptor on the same orbit. The results obtained using $\mathcal{AD}$ + QLS are similar to the ones obtained using $\mathcal{AD}$, except that now we obtain a matching for the 5 image pairs in the case of a reduced orbit with 10 simulated images.

Conclusions

We now summarize the contributions of this thesis, and briefly discuss some extensions to the work presented here.

Contributions Let us summarize the main contributions of the thesis:

1. Introducing the Global Brightness Change (GBC) assumption for video editing as a generalization of the brightness constancy assumption. The main advantage of the GBC assumption is that it models temporal consistency while allowing for global additive illumination changes.

2. Presenting an energy functional based on the GBC assumption to propagate gradient domain information. Unlike models based on the brightness constancy assumption which, on one hand, propagate color information and, on the other, require two minimization processes, the proposed energy propagates gradient domain information and its minimizers are both spatially and temporally consistent

3. As it has been noted in the literature [KCR05, SMTK06], propagating information along motion trajectories results in a blurring artefact after just a few frames. In this thesis, we have introduced a novel numerical scheme, the de-blurring scheme for the convective derivative (DSCD), that allows to propagate information along motion trajectories for a large number of frames while maintaining the sharpness of the result.

4. We have introduced a way to generate affine invariant quantities that can be used in the construction of "geometric" affine invariant feature descriptors for object recognition. We have experimentally observed that descriptors using the proposed quantities have more discriminative power and improve the performance of state of the art ones such as SIFT.

Future work We believe that the work presented in this thesis opens the door for several interesting issues to be further investigated, among which are:

1. The detailed study of the DSCD to provide a better understanding of its behavior and limitations. This study might also lead to an improved version of the DSCD.

2. Investigating the use of operator $\nabla_x \partial_v$ as a regularizer in the computation of optical flow.

3. Investigating the use of the proposed 1^{st} and 2^{nd} order video editing models for frame interpolation between two input images with different lighting conditions.

4. Studying the practical details of the angular quantization argument presented in Appendix B.2, in particular its relevance regarding the reduction of the number of simulations in the orbit of affine transformations in algorithms like ASIFT.

5. Applying the argument presented in Appendix B.3 in applications where high accuracy in the keypoint detection is needed, for example the $3D$ reconstruction of a scene using satellite images.

Appendices

A On video editing

A.1 Derivation of the discrete energy minimization of the 1^{st}-order model with $p = 1$

The discrete formulation of (3.9) is written as follows:

$$E(u) = \min_u \sum_{(i,j)\in\Omega} |(\partial_v u)_{i,j}| + \beta\|(\nabla_x u)_{i,j} - g_{i,j}\| + \frac{1}{2\lambda}\|u - f\|^2. \tag{A.1}$$

To minimize this energy we use the algorithm described in [Cha04]. In order to do that we introduce two vector fields ξ and ψ where ξ is defined as a map $\xi : \{1,...,N\} \times \{1,...,N\} \to \mathbf{R}^2$ and ψ is defined as a map $\psi : \{1,...,N\} \to \mathbf{R}$.

Introducing the dual variables ξ and ψ we have

$$E(u) = \max_{\|\xi\|\leq 1, |\psi|\leq 1} \min_u \langle \psi, \partial_v u\rangle + \beta\langle \xi, \nabla_x u - g\rangle + \frac{1}{2\lambda}\|u - f\|^2, \tag{A.2}$$

and minimizing with respect to u we get:

$$u = f + \lambda(\partial^*{}_v\psi + \beta \mathrm{div}_x \xi). \tag{A.3}$$

After replacing (A.3) in (A.2) we obtain the dual formulation:

$$\min_{\|\xi\|\leq 1, |\psi|\leq 1} \|\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + \frac{f}{\lambda}\|^2 + \frac{2}{\lambda}\beta\langle \xi, g\rangle \tag{A.4}$$

Adding the Lagrange multipliers corresponding to the restrictions $|\psi| \leq 1$ and $\|\xi\| \leq 1$: $\sum_\Omega \alpha_1(\|\psi\|^2 - 1)$ and $\sum_\Omega \alpha_2(\|\xi\|^2 - 1)$, and differentiating with respect to ψ, and ξ we get the Euler-Lagrange equations

$$\begin{aligned} \partial_v[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda] + \alpha_1\psi = 0, \\ \nabla_x[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda] + g/\lambda + \alpha_2\xi = 0. \end{aligned}$$

The *Karush-Kuhn-Tucker* theorem yields the existence of the Lagrange multipliers α_1 and α_2 with values $\alpha_1 = |\partial_v[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda]|$ and $\alpha_2 = \|\nabla_x[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda] + g/\lambda\|$. Then the solution of (A.4) is computed with the semi-implicit scheme

$$\begin{aligned} \psi^{k+1} &= \frac{\psi^k + \tau\partial_v[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda]}{1 + \tau|\partial_v[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda]|}, \\ \xi^{k+1} &= \frac{\xi^k + \tau\nabla_x[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda] + g/\lambda}{1 + \tau\|\nabla_x[\partial^*{}_v\psi + \beta \mathrm{div}_x \xi + f/\lambda] + g/\lambda\|}, \end{aligned}$$

where τ is a time step small enough for assuring the convergence of the fixed point iteration. Finally the solution of the primal problem is recovered with (A.3).

A.2 The Euler-Lagrange equation

Throughout the rest of the paper, we assume that O is a subset of $\Omega^T = \Omega \times [0,T]$ with Lipschitz boundary. Then the unit normal is defined almost everywhere on ∂O with respect to the Hausdorff measure $\mathcal{H}^2$ (surface measure) on ∂O. Let us denote by $\boldsymbol{\nu}^O = (\boldsymbol{\nu}_{\boldsymbol{x}}^O, \nu_t^O)$ the outer unit normal to ∂O (a vector in the unit sphere of $\mathbb{R}^3$) and $\boldsymbol{\nu}^{O_t}$ the outer unit normal to ∂O_t (a vector in the unit circle of $\mathbb{R}^2$).

Let us compute the Euler-Lagrange equations associated to the energy

$$E_{\boldsymbol{\kappa},\lambda}(u) = \int_O \left(\frac{1}{2} \|\boldsymbol{\kappa}(\boldsymbol{x},t)\, \boldsymbol{\nabla} \partial_v u(\boldsymbol{x},t)\|^2 + \frac{\lambda}{p} \|\boldsymbol{\nabla} u(\boldsymbol{x},t)\|^p \right) \mathrm{d}\boldsymbol{x}\, \mathrm{d}t, \tag{A.5}$$

where $\lambda \geq 0$ and $p = 1, 2$. For that, assume that $u : O \to \mathbb{R}$ is a minimum of $E_{\boldsymbol{\kappa},\lambda}$. To compute the Euler-Lagrange equations, we consider a perturbation $\bar{u}$ such that $E_{\boldsymbol{\kappa},\lambda}(\bar{u}) < \infty$. Since u is a minimum of $E_{\boldsymbol{\kappa},\lambda}$ we have

$$\begin{aligned} \lim_{\epsilon \to 0+} \frac{E_{\boldsymbol{\kappa},\lambda}(u + \epsilon \bar{u}) - E_{\boldsymbol{\kappa},\lambda}(u)}{\epsilon} &= \int_O \boldsymbol{\kappa} \boldsymbol{\nabla} \partial_v u \cdot \boldsymbol{\kappa} \boldsymbol{\nabla} \partial_v \bar{u}\, d\boldsymbol{x} dt \\ &+ \lambda \int_O \boldsymbol{\xi} \cdot \boldsymbol{\nabla} \bar{u} \mathrm{d}\boldsymbol{x}\, \mathrm{d}t = 0, \end{aligned}$$

where, when $\lambda > 0$ and $p = 1$, $\boldsymbol{\xi} : O \to \mathbb{R}^2$ is a measurable vector field such that $\|\boldsymbol{\xi}\|_\infty \leq 1$, $\boldsymbol{\xi} \cdot \boldsymbol{\nabla} u = |\boldsymbol{\nabla} u|$, and the arguments $(\boldsymbol{x}, t)$ of the functions are omitted for simplicity. If $\lambda > 0$ and $p = 2$, then $\boldsymbol{\xi} = \boldsymbol{\nabla} u$. Integrating by parts we have

$$\begin{aligned} 0 &= \int_O \boldsymbol{\kappa} \boldsymbol{\nabla} \partial_v u \cdot \boldsymbol{\kappa} \boldsymbol{\nabla} \partial_v \bar{u}\, d\boldsymbol{x} dt + \lambda \int_O \boldsymbol{\xi} \cdot \boldsymbol{\nabla} \bar{u} \mathrm{d}\boldsymbol{x}\, \mathrm{d}t \\ &= \int_O \partial_v^* \boldsymbol{\nabla}^* (\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u) \bar{u}\, d\boldsymbol{x} dt + \lambda \int_O \boldsymbol{\nabla}^* \boldsymbol{\xi}\, \bar{u} \mathrm{d}\boldsymbol{x}\, \mathrm{d}t \\ &+ \int_{\partial O} \boldsymbol{\nabla}^* (\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u)(\nu_t^O + \boldsymbol{v} \cdot \boldsymbol{\nu}_{\boldsymbol{x}}^O) \bar{u}\, d\mathcal{H}^2 + \lambda \int_0^T \int_{\partial O_t} \boldsymbol{\xi} \cdot \boldsymbol{\nu}^{O_t} \bar{u} d\mathcal{H}^1 dt \\ &+ \int_0^T \int_{\partial O_t} \boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u \cdot \boldsymbol{\nu}^{O_t} \partial_v \bar{u}\, d\mathcal{H}^1 dt, \end{aligned}$$

where $d\mathcal{H}^2$, resp. $d\mathcal{H}^1$, denotes the surface measure in ∂O, resp. the length measure in ∂O_t. We have denoted by $\boldsymbol{\nabla}^*$ (resp. ∂_v^*) the adjoint operator, that is $\boldsymbol{\nabla}^* b = -\mathrm{div}\, b$ for any vector field $b : O \to \mathbb{R}^2$ (resp. $\partial_v^* \psi = -\frac{\partial \psi}{\partial t} - \mathrm{div}\,(v\psi)$, for any function $\psi : O \to \mathbb{R}$). By taking test functions that vanish in a neighborhood of the boundary we have $\bar{u} = 0$, $\partial_v \bar{u} = 0$ on ∂O and we deduce that

$$\partial_v^* \boldsymbol{\nabla}^* (\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u) + \lambda \boldsymbol{\nabla}^* \boldsymbol{\xi} = 0 \quad \text{in } O.$$

Introducing this in the above expressions we get

$$\begin{aligned} &\int_{\partial O} \boldsymbol{\nabla}^* (\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u)(\nu_t^O + \boldsymbol{v} \cdot \boldsymbol{\nu}_{\boldsymbol{x}}^O) \bar{u}\, d\mathcal{H}^2 \\ &+ \lambda \int_0^T \int_{\partial O_t} \boldsymbol{\xi} \cdot \boldsymbol{\nu}^{O_t} \bar{u} d\mathcal{H}^1 dt \\ &+ \int_0^T \int_{\partial O_t} \boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u \cdot \boldsymbol{\nu}^{O_t} \partial_v \bar{u}\, d\mathcal{H}^1 dt = 0 \end{aligned} \tag{A.6}$$

and this holds for any admissible perturbation $\bar{u}$ that will be clarified below.

Let us discuss the boundary conditions that can be specified for the problem. We use the definition and notations given in Section 3.1.2.1. A set of natural boundary conditions are those for which the identity (A.6) holds. Let us discuss the possible choices.

Dirichlet boundary conditions. Dirichlet boundary conditions for u can be specified on a given set $A \subset \partial O$ if $\lambda > 0$ or on a subset $A \subset \partial O \setminus \partial O_{\text{tang}}$ if $\lambda = 0$. Namely we can specify

$$u(\boldsymbol{x}, t) = u_0(\boldsymbol{x}, t) \quad (\boldsymbol{x}, t) \in A. \tag{A.7}$$

If u satisfies (A.7) and we take test functions $\bar{u}$ such that $\bar{u} = 0$ on A, then $u + \epsilon\bar{u}$ satisfies (A.7) and the first and second integrals in (A.6) vanishes on A.

Observe that, since ∂O is Lipschitz,

$$\{(\boldsymbol{x}, t) : \boldsymbol{x} \in \partial O_t, t \in (0, T)\} = \partial O_{\text{tang}} \cup \partial O_{\text{obli}},$$

where strictly speaking this equality holds modulo null sets with respect to the surface measure.

Specifying $\partial_v u$ on the boundary. We can specify $\partial_v u$ on a given subset of $\{(\boldsymbol{x}, t) : \boldsymbol{x} \in \partial O_t, t \in (0, T)\}$. Namely we can specify

$$\partial_v u(\boldsymbol{x}, t) = g_0(\boldsymbol{x}, t) \quad (\boldsymbol{x}, t) \in B \subset \partial O_{\text{tang}} \cup \partial O_{\text{obli}}. \tag{A.8}$$

If u satisfies (A.8) and we take test functions $\bar{u}$ such that $\partial_v \bar{u} = 0$ on $B \subset \partial O_{\text{tang}} \cup \partial O_{\text{obli}}$, then $u + \epsilon\bar{u}$ satisfies (A.8) and the third integral in (A.6) vanishes on B.

Specifying other boundary conditions. We can specify the boundary condition at $(\boldsymbol{x}, t) \in A' \subset \partial O$

$$\boldsymbol{\nabla}^*(\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u)\boldsymbol{\nu}^O \cdot (\boldsymbol{v}, 1) + \lambda \boldsymbol{\xi} \cdot \boldsymbol{\nu}^{O_t} = 0 \tag{A.9}$$

with the convention that $\boldsymbol{\xi} \cdot \boldsymbol{\nu}^{O_t} = 0$ if $(\boldsymbol{x}, t) \in \partial O_{\text{vert}} \cup O_0 \cup O_T$. Then the sum of the first and second integrals in (A.6) vanishes on A'.

Notice that if $\lambda = 0$, (A.9) reduces to

$$\boldsymbol{\nabla}^*(\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u)\boldsymbol{\nu}^O \cdot (\boldsymbol{v}, 1) = 0 \tag{A.10}$$

and is trivially satisfied if $(x, t) \in \partial O_{\text{tang}}$ since in that case $\boldsymbol{\nu}^O \cdot (\boldsymbol{v}, 1) = 0$. That is, this gives no boundary condition at points $(x, t) \in \partial O_{\text{tang}}$. Thus, when $\lambda = 0$ we can only impose (A.10) on subsets $A' \subset \partial O \setminus \partial O_{\text{tang}}$.

If $\lambda > 0$, we can impose (A.9) on any subset $A' \subset \partial O$, understanding that it reduces to

$$\boldsymbol{\xi} \cdot \boldsymbol{\nu}^{O_t} = 0. \tag{A.11}$$

Specifying $\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u \cdot \boldsymbol{\nu}^{O_t} = 0$ on the boundary. We can specify the boundary condition at $(\boldsymbol{x}, t) \in B' \subset \partial O_{\text{tang}} \cup \partial O_{\text{obli}}$

$$\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u \cdot \boldsymbol{\nu}^{O_t} = 0.$$

Then the second integral in (A.6) vanishes on B'.

Depending on the problem we choose a set of boundary conditions. The only requirements are that

$$A \cup A' = \partial O \quad \text{if } \lambda > 0, \text{ or } A \cup A' = \partial O \setminus \partial O_{\text{tang}} \quad \text{if } \lambda = 0,$$

and

$$B \cup B' = \partial O_{\text{tang}} \cup \partial O_{\text{obli}}.$$

This implies that the identity (A.6) holds.

Boundary conditions for the one-lid setting. In the context of the one-lid problem, we choose the set of boundary conditions

$$u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}, 0), \qquad \boldsymbol{x} \in O_0, \tag{A.12}$$

$$u(\boldsymbol{x}, t) = u_0(\boldsymbol{x}, t), \qquad (\boldsymbol{x}, t) \in \partial O_{\text{vert}}, \tag{A.13}$$

$$\partial_v u(\boldsymbol{x}, t) = g_0(\boldsymbol{x}, t) \ , \qquad (\boldsymbol{x}, t) \in \partial O_{\text{tang}} \setminus \partial \Omega^T, \tag{A.14}$$

$$\begin{aligned} &u(\boldsymbol{x}, t) = u_0(\boldsymbol{x}, t) \\ &\partial_v u(\boldsymbol{x}, t) = g_0(\boldsymbol{x}, t) \end{aligned} \ , \qquad (\boldsymbol{x}, t) \in \partial O_{\text{obli}} \setminus \partial \Omega^T, \tag{A.15}$$

to which, when $\lambda > 0$, we add

$$u(\boldsymbol{x}, t) = u_0(\boldsymbol{x}, t) \qquad (\boldsymbol{x}, t) \in \partial O_{\text{tang}} \setminus \partial \Omega^T, \tag{A.16}$$

where the videos u_0 and g_0 are given. Notice that the boundary condition (A.16) is interpreted classically if $p = 2$ and it has to be interpreted in a relaxed sense if $p = 1$.

The boundary conditions on the rest of ∂O are

$$\boldsymbol{\nabla}^*(\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u)(\boldsymbol{x}, t) = 0, \qquad \boldsymbol{x} \in O_T, \tag{A.17}$$

$$\begin{aligned} &\lambda \boldsymbol{\xi} \cdot \boldsymbol{\nu}^{O_t}(\boldsymbol{x}, t) = 0 \\ &\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u(\boldsymbol{x}, t) \cdot \boldsymbol{\nu}^{O_t}(\boldsymbol{x}, t) = 0 \end{aligned} \ , \qquad (\boldsymbol{x}, t) \in \partial O_{\text{tang}} \cap \partial \Omega^T, \tag{A.18}$$

$$\begin{aligned} &\boldsymbol{\nabla}^*(\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u)(\boldsymbol{x}, t) + \lambda \boldsymbol{\xi} \cdot \boldsymbol{\nu}^{O_t}(\boldsymbol{x}, t) = 0 \\ &\boldsymbol{\kappa}^2 \boldsymbol{\nabla} \partial_v u(\boldsymbol{x}, t) \cdot \boldsymbol{\nu}^{O_t}(\boldsymbol{x}, t) = 0 \end{aligned} \ , \qquad (\boldsymbol{x}, t) \in \partial O_{\text{obli}} \cap \partial \Omega^T. \tag{A.19}$$

Boundary conditions for the two-lid setting. They are given by (A.12),(A.13),(A.14),(A.15),(A.18),(A.19), and (A.17) is replaced by

$$u(\boldsymbol{x}, T) = u_0(\boldsymbol{x}, T) \quad \text{in } O_T. \tag{A.20}$$

Let us observe that the boundary conditions (A.12),(A.13),(A.14), (A.15), and (A.20) in the two lid-case, are specified in the set of admissible functions in which $E_{\boldsymbol{\kappa},\lambda}$ will be minimized.

Remark. Under some assumptions on the vector field **v**, we can prove existence and uniqueness of minima of $E_{\kappa,\lambda}$ in a suitable class of functions (the functional space where the energy is finite and permits to incorporate boundary conditions). In particular, this shows that the boundary conditions are sufficient to determine the solution.

A.3 The functional analytic framework and existence of minima of $E_{\kappa,\lambda}$

The study of minima of $E_{\kappa,\lambda}$ requires to define suitable functional spaces. For that we assume that $v \in L^\infty(\Omega \times (0,T); \mathbb{R}^2)$ with $\mathrm{div}_x\, v \in L^2(\Omega \times (0,T))$. We also assume that κ is a diagonal matrix with entries in $L^\infty(O)$ and $\kappa \geq \alpha I$, where I is the identity matrix and $\alpha > 0$. To fix ideas, we will consider the case $p = 1$. The case $p = 2$ is similar and more simple.

If Q is an open subset of $\mathbb{R}^N$ with Lipschitz boundary, we denote by $W^{1,2}(Q)$ the set of functions $w \in L^2(Q)$ such that $\nabla w \in L^2(Q)$. We denote by $BV(Q)$ the space of functions of bounded variation in Q. We refer to [AFP00] for the definition and properties of BV functions.

Recall that if $w \in BV(Q) \cap L^2(Q)$ and $z \in L^\infty(Q; \mathbb{R}^2)$ is such that $\mathrm{div}\, z \in L^2(Q)$, then the distribution defined by

$$\int_Q z \cdot Dw\phi := -\int_Q w \mathrm{div}\, z\, \phi\, \mathrm{d}x - \int_Q wz \cdot \nabla\phi\, \mathrm{d}x,$$

where ϕ is a smooth test function with compact support in Q, is a Radon measure in Q such that

$$\int_Q |z \cdot Dw| \leq \|z\|_\infty \int_Q |Dw|. \tag{A.21}$$

The normal trace $z \cdot \nu^Q$ of z in ∂Q is well defined and the integration by parts formula holds

$$\int_Q z \cdot Dw + \int_Q w \mathrm{div}\, z = \int_{\partial Q} z \cdot \nu^Q w\, d\mathcal{H}^{N-1}, \tag{A.22}$$

where $w \in BV(Q) \cap L^2(Q)$, $\nu^Q(x)$ denotes the outer unit formal to ∂Q at $x \in \partial Q$ and $\mathcal{H}^{N-1}$ is the $N-1$-dimensional Hausdorff measure. We refer to [Anz83] for details.

We assume that $u \in L^1_w(0,T; BV(O_t))$, that is $u : (0,T) \in BV(O_t)$ is weakly measurable, i.e. $t \in (0,T) \to u(t) \in BV(O_t)$ such that $u \in L^1(O)$ and $u \in (0,T) \to \int_{O_t} \varphi \cdot Du$ is a measurable map for any $\varphi \in C^1(O)$ with compact support in O. Then $\partial_v u = (\partial_t + v \cdot \partial_x)u$ is a distribution in O.

Notice that the energy $E_{\kappa,\lambda}$ is defined for all $u \in L^1_w(0,T; BV(O_t))$ such that $\partial_v u \in L^2(O)$ and $\nabla_x \partial_v u \in L^2(O)$. We also assume that these functions satisfy the boundary conditions (A.12),(A.13),(A.14),(A.15), and (A.20) in the two-lid case. Let us denote this set of functions by $\mathcal{A}$.

The boundary conditions in ∂O_{tang} have to be considered in a relaxed form. For that, we consider the energy

$$E^b_{\kappa,\lambda}(u) = E_{\kappa,\lambda}(u) + \lambda \int_{\partial O \setminus \partial\Omega^T} |u(x,t) - u_0(x,t)|\, d\mathcal{H}^2$$

defined on the class of admissible functions $\mathcal{A}$. With this the boundary integral is defined only in $\partial O_{\text{tang}} \setminus \partial\Omega^T$. Then the boundary condition (A.16) is written as

$$\xi \cdot \nu^{O_t} \in \text{sign}(u_0(x,t) - u(x,t)) \quad (x,t) \in \partial O_{\text{tang}} \setminus \partial\Omega^T. \tag{A.23}$$

Assume that for almost any $t \in (0,T)$ $\partial O_t \setminus \partial\Omega$ is not an $\mathcal{H}^1$ null set. We need this to use Poincaré's inequality (see [Zie89]) in the proof of next Proposition.

Proposition 2. *Let $\lambda > 0$. There exits a minimum of $E^b_{\kappa,\lambda}$ in $\mathcal{A}$.*

Proof. Let u_n be a minimizing sequence. For almost any $t \in (0,T)$, $\partial_v u_n(t) \in W^{1,2}(O_t)$ for every n. By Poincaré's inequality [Zie89] we have

$$\begin{aligned}\int_{O_t} |\partial_v u_n(t)|^2 \mathrm{d}x &\leq C_1 \int_{O_t} |\nabla_x \partial_v u_n(t)|^2 \mathrm{d}x \\ &+ C_2 \int_{\partial O_t \setminus \partial\Omega} |g_0(x,t)|^2\, d\mathcal{H}^1.\end{aligned}$$

Integrating it in $(0,T)$ and using that $E_{\kappa,\lambda}(u_n)$ is bounded we deduce that $\partial_v u_n$ is bounded in $L^2(O)$.

Now,

$$\partial_t u_n = \partial_v u_n - v \nabla_x u_n.$$

Using our assumptions on v, the fact that $u_n(t) \in BV(O_t)$ for all n, and (A.21) we have that $v\nabla_x u_n$ is a Radon measure in O and

$$\int_0^T \int_{O_t} |v \nabla_x u_n| \leq \|v\|_\infty \int_0^T \int_{O_t} |\nabla_x u_n| \leq \frac{\|v\|_\infty}{\lambda} E_{\kappa,\lambda}(u_n).$$

Then $\partial_t u_n$ are Radon measures and their total mass is uniformly bounded in n. Since also $\nabla_x u_n$ are Radon measures and their total mass is uniformly bounded in n, then u_n is uniformly bounded in $BV(O)$. We may extract a subsequence converging in $L^1(O)$ to a function $u \in L^1(O)$. Then $\partial_v u \in L^2(O)$ and

$$E^b_{\kappa,\lambda}(u) \leq \liminf_n E^b_{\kappa,\lambda}(u_n).$$

To prove that u is a minimum of $E_{\kappa,\lambda}$ we need to prove that u satisfies the boundary conditions (A.12), (A.13), (A.14), (A.15). Since $\{\nabla_x \partial_v u_n(t)\}_n$ is bounded in $L^2(O_t)$ for almost any $t \in (0,T)$, the boundary $\partial_v u(x,t) = g_0(x,t)$ are satisfied on $(\partial O_{\text{tang}} \cup \partial O_{\text{obli}}) \setminus \partial\Omega^T$.

Let us prove that u satisfies the Dirichlet boundary conditions given in (A.12), (A.13), (A.15). Let $\psi \in C^1(\overline{O})$. Since $u_n \in BV(O)$, using Green's formula (A.22) we have

$$\int_O \partial_v u_n \psi\, dxdt = \int_O u_n \partial_v^* \psi\, dxdt + \int_{\partial O} (\nu_t^O + v \cdot \nu_x^O) u_0 \psi\, d\mathcal{H}^2.$$

Since $\partial_v u_n \to \partial_v u$ weakly in $L^2(O)$ and $u_n \to u$ in $L^1(O)$, letting $n \to \infty$ we obtain

$$\int_O \partial_v u \psi \, dxdt = \int_O u \partial_v^* \psi \, dxdt + \int_{\partial O} (\nu_t^O + v \cdot \nu_x^O) u_0 \psi \, d\mathcal{H}^2. \tag{A.24}$$

This implies that $u(x,t) = u_0(x,t)$ in (A.12), (A.13), (A.15). □

Remark. Let us give a more classical point of view on the Dirichlet boundary conditions for u out of the tangential boundary. In that context, to prove that u satisfies the Dirichlet boundary conditions requires some additional assumptions on the vector field v. Let us define the incoming (resp. outgoing) boundary, that we denote by $\partial_{\text{in}} O$ (resp. $\partial_{\text{out}} O$), as the set of points $(x,t) \in \partial O$ such that $\nu_t^O + v \cdot \nu_x^O < 0$ (resp > 0). Notice that O_0 is part of the incoming boundary. Assume that $Z \subset \partial_{\text{in}} O$ is such that for any $(\bar{x}, \bar{t}) \in Z$ we have that $v \in L^1([\bar{t}, \bar{t} + \delta], W^{1,\infty}(V_{\bar{x}}))$ for some $\delta > 0$ and a neighborhood of $\bar{x}$. Then we have a unique solution of the equation $X_t(t, \bar{x}) = v(X(t, \bar{x}), t)$, $t \in [\bar{t}, \bar{t} + \delta]$, such that $X(\bar{t}, \bar{x}) = \bar{x}$. Since $c_n = \partial_v u_n \in L^2(O)$, then we may write

$$u_n(X(t, \bar{x}), t) = u_0(\bar{x}, \bar{t}) + \int_{\bar{t}}^t c_n(X(s, \bar{x}), s) \, \mathrm{d}s. \tag{A.25}$$

By passing to the limit we have that

$$u(X(t, \bar{x}), t) = u_0(\bar{x}, \bar{t}) + \int_{\bar{t}}^t c(X(s, \bar{x}), s) \, \mathrm{d}s. \tag{A.26}$$

holds a.e. $(\bar{x}, \bar{t}) \in Z$ where $c \in L^2(O)$. Thus $u(\bar{x}, \bar{t}) = u_0(\bar{x}, \bar{t})$ on Z.

The same argument can be repeated for the outgoing boundary. This time we assume that $Z \subset \partial_{\text{out}} O$ is such that for any $(\bar{x}, \bar{t}) \in Z$ we have that $v \in L^1([\bar{t} - \delta, \bar{t}], W^{1,\infty}(V_{\bar{x}}))$ for some $\delta > 0$ and a neighborhood of $\bar{x}$. We deduce that $u(\bar{x}, \bar{t}) = u_0(\bar{x}, \bar{t})$ on Z.

In particular u satisfies the Dirichlet boundary conditions in $\partial O_{\text{vert}} \cup (\partial O_{\text{obliq}} \setminus \partial \Omega^T)$ if $v \in L^1(0, T; W^{1,\infty}(\Omega))$.

In case that v does not satisfy the local Lipschitz condition on incoming and outgoing boundary points, we cannot guarantee that the Dirichlet boundary conditions for u are satisfied in the classical sense and we consider them in the relaxed form. We can impose them on the admissible class of functions out of the tangential boundary $\partial O_{\text{tang}} \setminus \partial \Omega^T$, penalizing their deviation on $\partial O_{\text{tang}} \setminus \partial \Omega^T$ in the energy. We could also impose all of them by penalization in the energy. In that case, we require that admissible functions satisfy only the boundary conditions for $\partial_v u$.

The case $\lambda = 0$. Let us assume that the vector field v satisfies

$$v \in L^1(0, T; W^{1,1}(\Omega; \mathbb{R}^2)) \cap L^1(0, T; L^\infty(\Omega; \mathbb{R}^2)), \tag{A.27}$$

$$\operatorname{div} v \in L^1(0, T; L^\infty(\Omega)). \tag{A.28}$$

Those assumptions replace the assumptions on v that we did at the beginning of this Section. By extending $v(\cdot,t)$ by parity and then by periodicity to $\mathbb{R}^2$, we may assume that v is the restriction to $\Omega\times(0,T)$ of a vector field

$$v\in L^1(0,T;W^{1,1}_{\mathrm{loc}}(\mathbb{R}^2;\mathbb{R}^2))\cap L^1(0,T;L^\infty(\mathbb{R}^2;\mathbb{R}^2)), \tag{A.29}$$

$$\operatorname{div} v\in L^1(0,T;L^\infty(\mathbb{R}^2)). \tag{A.30}$$

Those are the assumptions under which the generalized DiPerna-Lions theory of transport equations holds [DL89]. These results have been extended to vector fields

$$v\in L^1_w(0,T;SBD_{\mathrm{loc}}(\mathbb{R}^2;\mathbb{R}^2))\cap L^1(0,T;L^\infty(\mathbb{R}^2;\mathbb{R}^2))$$

satisfying (A.30) by Ambrosio-Crippa-Maniglia [ACM05]. We have denoted by $SBD_{\mathrm{loc}}(\mathbb{R}^2,\mathbb{R}^2)$ the space of vector fields $b=(b_1,b_2)$ in $L^1_{\mathrm{loc}}(\mathbb{R}^2,\mathbb{R}^2)$ such that $\frac{\partial b_1}{\partial x_2}+\frac{\partial b_2}{\partial x_1}$ is a Radon measure in $\mathbb{R}^2$ with no Cantor part. The case where

$$v\in L^1_w(0,T;BV_{\mathrm{loc}}(\mathbb{R}^2;\mathbb{R}^2))\cap L^1(0,T;L^\infty(\mathbb{R}^2;\mathbb{R}^2))$$

and satisfies (A.30) has been considered in [Amb04] (see also [AC08]). To fix ideas, assume that DiPerna-Lions assumptions hold.

Proposition 3. *Assume that (A.29) and (A.30) hold. Let $M>0$. There exits a minimum of $E_{\kappa,0}$ in $\mathcal{A}\cap\{u\in L^\infty(O):|u|\le M\}$.*

Imposing that $|u|\le M$ for some $M>0$ is not a restrictive assumption for images, since they are bounded by the maximum intensity (usually 255).

Proof. Let us give a sketch of the proof. Let u_n be a minimizing sequence of $E_{\kappa,0}$. As in the proof of Proposition 3, $\partial_v u_n$ is bounded in $L^2(O)$. Since $|u_n|\le M$, by extracting a subsequence we may assume that $u_n\to u$ weakly* in $L^\infty(O)$ and $\partial_v u_n\to\partial_v u$ weakly in $L^2(O)$. Then $\partial_v u\in L^2(O)$ and by the lower semicontinuity of the energy we have

$$E_{\kappa,0}(u)\le\liminf_n E_{\kappa,0}(u_n).$$

To prove that u is a minimum of $E_{\kappa,0}$ we need to prove that u satisfies the boundary conditions (A.12), (A.13), (A.14), (A.15). Since $\{\nabla_x\partial_v u_n(t)\}_n$ is bounded in $L^2(O_t)$ for almost any $t\in(0,T)$, the boundary condition $\partial_v u(x,t)=g_0(x,t)$ is satisfied on $(\partial O_{\mathrm{tang}}\cup\partial O_{\mathrm{obli}})\setminus\partial\Omega^T$.

Let us prove that u satisfies the Dirichlet boundary conditions given in (A.12), (A.13), (A.15). By our assumptions on v and the results in [Anz83, ACM05], $(1,v)u$ has a trace on ∂O and the integration by parts formula (A.24) holds for u_n and any $\psi\in C^1(\overline{O})$. Letting $n\to\infty$, (A.24) holds for u and any $\psi\in C^1(\overline{O})$. The boundary conditions for u are satisfied in this weak sense. □

Remark 4. Although the assumptions for v above are quite general, they may not be sufficient to cover real video cases, since optical flow may have discontinuities along curves and its divergence may be a Radon measure. In the continuous framework, one could compute the optical flow by imposing constraints that guarantee at least that div v has some integrability properties, e.g. being in L^2.

Remark 5. Let us comment again on the classical point of view to prove existence when $\lambda = 0$. In this case we do not assume that admissible functions are bounded by M. Assume that $\overline{O} \subset \Omega^T$, v satisfies (A.29) and (A.30), and $\partial_{\text{vert}}O \cup \partial_{\text{obli}}O = \emptyset$. Under that conditions, for almost any $x \in \Omega$ we have a unique solution of the equation $X_t(t,x) = v(X(t,x),t)$ such that $X(0,x) = x$. Assume for simplicity that all trajectories in O start at O_0 and end on O_T. In that case, we can get a bound on u_n in $L^2(O)$. Indeed, if $c_n = \partial_v u_n \in L^2(O)$, then we may write

$$u_n(X(t,x),t) = u_0(x,0) + \int_0^t c_n(X(s,x),s)\,\mathrm{d}s. \tag{A.31}$$

Since v satisfies (A.29) and (A.30), the Jacobian of the map $y = X(t,x)$ is bounded and bounded away from zero [DL89] and from the above identity we deduce that u_n is bounded in $L^2(O)$.

As in the case $\lambda > 0$, the boundary conditions that specify $\partial_v u$ are satisfied. Also the Dirichlet boundary conditions on O_0 and O_T are satisfied.

The consideration of existence and uniqueness results of solutions of transport equations and the corresponding ordinary differential equations in bounded domains under very mild conditions leads to more deep mathematical analysis and is not the object of the present paper. We refer to [CL05] for a uniqueness result when the boundary of the domain is transversal to the flow. General existence results in $\mathbb{R}^N$ or in bounded domains where the flow is tangential can be found in [DL89, Amb04, ACM05, Amb08, AC08].

Remark 6. Notice that we had to assume that $\kappa \geq \alpha I$, where I is the identity matrix and $\alpha > 0$, for any $(x,t) \in O$. The above techniques can also be adapted to consider the case where $\kappa(x,t) = 0$ for $(x,t) \in \Gamma \subset O$ where Γ is a closed set of zero measure.

We will discuss below the discrete approach to these problems.

A.4 On uniqueness of minima of $E_{\kappa,\lambda}$

Let us assume that the vector field satisfies assumptions (A.29) and (A.30). The proof holds both for $p = 1, 2$.

Let u_1, u_2 be two minima of $E^b_{\kappa,\lambda}$ in $\mathcal{A}$ (or of $E_{\kappa,0}$ in $\mathcal{A} \cap \{u \in L^\infty(O) : |u| \leq M\}$). If $\nabla_x \partial_v u_1 \neq \nabla_x \partial_v u_2$, since the quadratic term of the energy is strictly convex, then

$$E_{\kappa,\lambda}\left(\frac{u_1+u_2}{2}\right) < \frac{1}{2}E_{\kappa,\lambda}(u_1) + \frac{1}{2}E_{\kappa,\lambda}(u_2).$$

Since $\frac{u_1+u_2}{2} \in \mathcal{A}$, this contradicts the fact that u_1, u_2 are minima of $E_{\kappa,\lambda}$.

Let $u = u_1 - u_2$. Then $\nabla_x \partial_v u = 0$ in O and all boundary conditions (A.12),(A.13),(A.14),(A.15) (plus (A.20) in the two-lid case) hold with homogeneous right hand side. This implies that $\partial_v u = 0$ in O. By (A.24) we have that

$$\int_O u \partial_v^* \psi \, dxdt = 0 \quad \forall \psi \in C^1(\overline{O}). \tag{A.32}$$

Now, for any test function $\phi \in \mathcal{D}(\mathbb{R}^2 \times (0,T))$ (that is, infinitely differentiable with compact support in $\mathbb{R}^2 \times (0,T)$)we consider the solution of

$$\frac{\partial \Psi}{\partial t} + \operatorname{div}(v\Psi) = \phi \quad \text{in } \mathbb{R}^2 \times (0,T),$$

with initial condition $\Psi(0) = 0$ in $\mathbb{R}^2$ [DL89]. Let $\rho_\epsilon(x) = \frac{1}{\epsilon^2}\rho(\frac{x}{\epsilon})$ where $\rho \in \mathcal{D}(\mathbb{R}^2)$, $\rho \geq 0$, and $\int_{\mathbb{R}^2} \rho(x)\, dx = 1$. By the regularization result in [DL89], we have that $\Psi_\epsilon = \rho_\epsilon * \Psi$ satisfies

$$\frac{\partial \Psi_\epsilon}{\partial t} + \operatorname{div}(v\Psi_\epsilon) = \phi + r_\epsilon \quad \text{in } \mathbb{R}^2 \times (0,T),$$

where $r_\epsilon \to 0$ in $L^1(0,T; L^1_{\text{loc}}(\mathbb{R}^2))$. By replacing ψ by Ψ_ϵ in (A.32) we have

$$\int_O u(\phi + r_\epsilon)\, dxdt = 0 \quad \forall \phi \in \mathcal{D}(\mathbb{R}^2 \times (0,T)). \tag{A.33}$$

Letting $\epsilon \to 0+$ we obtain

$$\int_O u\phi \, dxdt = 0 \quad \forall \phi \in \mathcal{D}(\mathbb{R}^2 \times (0,T)). \tag{A.34}$$

This implies that $u = 0$. That is, $u_1 = u_2$.

A.5 Remarks on existence and uniqueness in the discrete case

For the discrete discussion, we will use the same notation as in the continuous domain as we did in the paper.

Let us consider the energy (A.5) in the discrete case which amounts to replace the integrals in O by sums, that is

$$E^d_{\kappa,\lambda}(u) = \sum_{(x,t)\in\widetilde{O}} \|\kappa(x,t)\nabla_x \partial_v u(x,t)\|^2 + \lambda \sum_{(x,t)\in\widetilde{O}} \|\nabla_x u(x,t)\|^p,$$

where $\lambda \geq 0$ and $p = 1, 2$. The energy is defined in vectors $u \in \mathcal{X} := \mathbb{R}^{|\widetilde{O}|}$. The boundary conditions have been described in the Section entitled "Definition of the Operators" in the paper.

Assume first that $\lambda > 0$. If u_n is a minimizing sequence for $E^d_{\kappa,\lambda}$, then we have that $\nabla_x u_n$ is bounded. From the Dirichlet boundary conditions, we deduce that u_n is bounded in $\mathcal{X}$. Then we may extract subsequence converging to $u \in \mathcal{X}$

satisfying the boundary conditions. Then u is a minimum of $E^d_{\kappa,\lambda}$. This result has been obtained for any $\kappa \geq 0$.

When $\lambda = 0$, we assume that $\kappa \geq \alpha I$, $\alpha > 0$. In that case, we first observe that $\nabla_x \partial_v u_n$ is bounded. From the specification of $\partial_v u_n$ on the boundary of each O_t (out of $\partial\Omega^T$), we deduce that $\partial_v u_n$ is bounded. Getting from this the boundedness of u_n requires specifying the discretization of ∂_v. To illustrate this (our treatment will be sketchy), assume that ∂_v is discretized as the backward derivative $\partial^b_v u$. Assume that $u_n(x,t)$ is bounded in O_t uniformly in n. Then $\hat{u}_n(x + v^b(x,t+1),t)$ is also bounded, being based on bilinear interpolation of the values of $u_n(x.t)$. Since $u_n(x,t+1) = \partial^b_v u_n(x,t+1) + \hat{u}_n(x + v^b(x,t+1),t)$ we deduce that $\{u_n(x,t+1) : x \in O_{t+1}\}$ is bounded uniformly in n. Assume now that ∂_v is discretized as the forward derivative $\partial^f_v u$ and $u_n(x,t)$ is bounded in O_t uniformly in n. Then $\hat{u}_n(x + v^f(x,t),t+1) = \partial^f_v u_n(x,t) + u_n(x,t)$ and we get that $\{\hat{u}_n(x + v^f(x,t),t+1) : x \in O_t\}$ is bounded uniformly in n. The flow has to be dense enough so that, from this and bi-linear interpolation equations , we can get that $\{u_n(x,t+1) : x \in O_{t+1}\}$ is bounded uniformly in n. Clearly, this depends on the optical flow and for that reason it is convenient to use $\lambda > 0$. The same conclusions apply to the DSCD scheme.

When $\lambda > 0$ and $p = 2$, the energy is strictly convex and uniqueness follows. When $\lambda > 0$ and $p = 1$, or $\lambda = 0$, uniqueness is a more delicate issue. As in Section A.4, uniqueness is reduced to prove that if $\nabla_x \partial_v u = 0$ in $\widetilde{O}$ and all boundary conditions (A.12),(A.13),(A.14),(A.15) (plus (A.20) in the two-lid case) hold with homogeneous right hand side, then $u = 0$. In a first step, from the specification of $\partial_v u$ on each ∂O_t we get that $\partial_v u = 0$. Getting from this that $u = 0$, we need to be able to connect by the flow v each pixel (x,t) in the interior of $\widetilde{O}$ to a boundary pixel where u is specified. To fix ideas, let us consider the case of the DSCD based on the odd-assignation. Assume that

$$h^{\text{odd}}_v u(x,t) = 0. \tag{A.35}$$

Since $u(x,0) = 0$, the interpolation of intermediate values gives $\hat{u}(x + v^b(x,1),0) = 0$. Hence

$$u(x,1) = \hat{u}(x + v^b(x,1),0) = 0 \quad \forall x \in O_1.$$

Now,

$$\hat{u}(x + v^f(x,1),2) = u(x,1) = 0 \quad \forall x \in O_1.$$

We know that $u(x,2) = 0\ \forall x \in \partial O_2$. The important point here is that, given our interpolation model, the density of points $x + v^f(x,1)$ has to be sufficient to guarantee that $\hat{u}(x + v^f(x,1),2) = 0\ \forall x \in O_1$ implies that $u(x,2) = 0\ \forall x \in O_2$. By iterating this argument, we obtain that $u = 0$. This requires an information on the optical flow v that cannot be guaranteed before hand. Here, the use of conjugate gradient method can help to stabilize the numerical solution.

A.6 Some examples of analytic solutions of the Euler-Lagrange equation

To illustrate how the prescribed boundary conditions determine the solution, we compute in this section the analytic solution of the Euler-Lagrange equation of the continuous energy $E_{\kappa,\lambda}$ with $\lambda = 0$. We consider two simple examples, one for the one-lid setting and one for the two-lid setting.

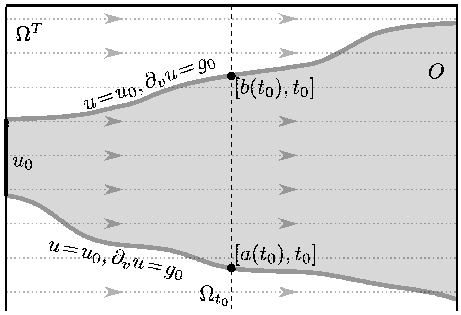


Figure A.1: Domain and boundary conditions for a one-lid problem. The optical flow in this example is zero.

We will consider a simple case, in which $v(x,t) = 0$ everywhere in Ω. In this case the convective derivative coincides with the partial derivative with respect to time: $\partial_v u = u_t$. The energy is therefore

$$E(u) = \int_O (u_{xt}(x,t))^2 \mathrm{d}x\mathrm{d}t. \tag{A.36}$$

We consider an editing domain $O = \{(x,t) : t \in [0,T], x \in [a(t), b(t)]\}$, such as the one depicted in Figure A.1. We consider $a : [0,T] \to \mathbb{R}$ and $b : [0,T] \to \mathbb{R}$ to be functions with a continuous bounded derivative, and such that $a(t) < b(t)$ for $t \in [0,T]$. We also suppose for simplicity that a is a strictly decreasing function, whereas b is strictly increasing.

A.6.1 One-lid problem

For the one-lid problem, the boundary conditions are as follows:

$$u(x,0) = u_0(x), \qquad x \in [a(0), b(0)], \tag{A.37}$$
$$u(a(t),t) = u_a(t), \qquad t \in [0,T], \tag{A.38}$$
$$u(b(t),t) = u_b(t), \qquad t \in [0,T], \tag{A.39}$$
$$u_t(a(t),t) = g_a(t), \qquad t \in [0,T], \tag{A.40}$$
$$u_t(b(t),t) = g_b(t), \qquad t \in [0,T]. \tag{A.41}$$

The minimum of the energy can be computed by solving the Euler-Lagrange equation:

$$u_{txxt}(x,t) = 0, \qquad \forall (x,t) \in O \tag{A.42}$$

with the following additional boundary condition, stemming from the computation of the first variation:

$$u_{xxt}(x,T) = 0, \qquad x \in [a(T), b(T)]. \tag{A.43}$$

Let us compute the solution of the PDE (A.42). Integrating it with respect to t, between t and T yields,

$$u_{xxt}(x,t) = u_{xxt}(x,T) = 0, \qquad \forall (x,t) \in O,$$

the last equality is due to the Neumann boundary condition at $t = T$. If we now integrate w.r.t. x, between $a(t)$ and x we obtain

$$u_{xt}(x,t) = u_{xt}(a(t),t), \qquad \forall (x,t) \in O.$$

We integrate again on variable x:

$$u_t(x,t) - u_t(a(t),t) = \int_{a(t)}^{x} u_{xt}(a(t),t)\mathrm{d}s = u_{xt}(a(t),t)(x - a(t)).$$

If we evaluate this expression on $x = b(t)$, we can express $u_{xt}(a(t),t)$ in terms of g_a and g_b:

$$u_{xt}(a(t),t) = \frac{u_t(b(t),t) - u_t(a(t),t)}{b(t) - a(t)} = \frac{g_b(t) - g_a(t)}{b(t) - a(t)}.$$

Therefore we have that

$$u_t(x,t) = g_a(t) + \frac{g_b(t) - g_a(t)}{b(t) - a(t)}(x - a(t)). \tag{A.44}$$

As a function of x, $u_t(x,t)$ is a linear function passing through $(a(t), g_a(t))$ and $(b(t), g_b(t))$, *i.e.* the rate of illumination change is a smooth interpolation of the values specified at the boundary.

To obtain the solution of the PDE we now integrate with respect to t. Let us define the function $\ell : [a(T), b(T)] \to R$ as

$$\ell(x) = \begin{cases} a^{-1}(x) & \text{if } a(T) \leqslant x \leqslant a(0), \\ 0 & \text{if } a(0) < x < b(0), \\ b^{-1}(x) & \text{if } b(0) \leqslant x \leqslant b(T). \end{cases}$$

Note that $(x, \ell(x))$ with $x \in [a(T), b(T)]$, corresponds to the "left" boundary of O, where the value of u is specified by u_a, u_0 and u_b. We now integrate (A.44) on t between $\ell(x)$ and t, yielding

$$u(x,t) - u_0(x, \ell(x)) = \int_{\ell(x)}^{t} g_a(s)\mathrm{d}s + \int_{\ell(x)}^{t} \frac{g_b(s) - g_a(s)}{b(s) - a(s)}(x - a(s))\mathrm{d}s.$$

This example demonstrates that the given boundary conditions are sufficient to compute a minimizer of the energy. It also shows how each boundary condition is used.

In this example, for fixed t, the illumination change rate $\partial_v u(x,t) = u_t(x,t)$ is the result of a linear interpolation between the illumination change rate given at the boundary, $g_a(t)$ and $g_b(t)$. Once the illumination change rate is known in the whole editing domain, the solution is computed by integrating it along the trajectories. The temporal integration starts at $\ell(x)$, the time instant where the trajectory through (x,t) reaches a point of the boundary where u_0 is specified (including the first lid where $\ell(x) = 0$). This integration starts with $u_0(x, \ell(x))$ and propagates it along the trajectory, while accommodating for the illumination changes previously computed in u_t.

A.6.2 Two-lid problem

For the two-lid problem, in addition to the boundary conditions of the one-lid problem, we add the second lid:

$$u(x,T) = u_T(x), \quad x \in [a(T), b(T)] \tag{A.45}$$

in substitution of (A.43). We use the notation of Section A.6.1.

Let us compute the solution of

$$u_{txxt}(x,t) = 0, \qquad (x,t) \in O. \tag{A.46}$$

Observe that

$$O = \{(x,t) \in \Omega \times [0,T] : t \geq \ell(x)\ x \in [a(t), b(t)]\}.$$

Integrating with respect to t, we have

$$u_{xxt}(x,t) = u_{xxt}(x, \ell(x)) \quad (x,t) \in O. \tag{A.47}$$

Let us denote $G(x) = u_{xxt}(x, \ell(x))$. Integrating again with respect to t, we get

$$u_{xx}(x,t) = u_{xx}(x, \ell(x)) + G(x)(t - \ell(x)), \quad (x,t) \in O. \tag{A.48}$$

For $t = T$ and $x \in [a(T), b(T)]$ we have

$$u_{xx}(x,T) = u_{xx}(x, \ell(x)) + G(x)(T - \ell(x)). \tag{A.49}$$

Evaluating the above expression for $x \in [a(0), b(0)]$, we obtain

$$G(x) = \frac{1}{T}(u_{xx}(x,T) - u_{xx}(x,0)) \quad x \in [a(0), b(0)]. \tag{A.50}$$

Integrating (A.48) with respect to x, we obtain

$$\begin{aligned} u_x(x,t) = u_x(a(t),t) + \int_{a(t)}^{x} u_{xx}(s, \ell(s))\, ds \\ + \int_{a(t)}^{x} G(s)(t - \ell(s))\, ds. \end{aligned} \tag{A.51}$$

Integrating (A.51) with respect to x from $a(t)$ to x, we obtain

$$u(x,t) = u(a(t),t) + u_x(a(t),t)(x-a(t)) + \int_{a(t)}^{x}\int_{a(t)}^{y} u_{xx}(s,\ell(s))\,ds\,dy + \int_{a(t)}^{x}\int_{a(t)}^{y} G(s)(t-\ell(s))\,ds\,dy \tag{A.52}$$

and integrating (A.51) from x to $b(t)$, we obtain

$$u(x,t) = u(b(t),t) + u_x(a(t),t)(x-b(t)) - \int_{x}^{b(t)}\int_{a(t)}^{y} u_{xx}(s,\ell(s))\,ds\,dy - \int_{x}^{b(t)}\int_{a(t)}^{y} G(s)(t-\ell(s))\,ds\,dy. \tag{A.53}$$

Evaluating (A.52) for $x = b(t)$ we obtain

$$u_b(t) = u_a(t) + u_x(a(t),t)(b(t)-a(t)) + \int_{a(t)}^{b(t)}\int_{a(t)}^{y} u_{xx}(s,\ell(s))\,ds\,dy + \int_{a(t)}^{b(t)}\int_{a(t)}^{y} G(s)(t-\ell(s))\,ds\,dy. \tag{A.54}$$

Using (A.49) we obtain

$$(t-T)\int_{a(t)}^{b(t)}\int_{a(t)}^{y} G(s)\,ds\,dy = A(t), \tag{A.55}$$

where

$$A(t) := u_b(t) - u_a(t) - u_x(a(t),t)(b(t)-a(t)) - \int_{a(t)}^{b(t)}\int_{a(t)}^{y} u_{xx}(s,T)\,ds\,dy. \tag{A.56}$$

Notice that by differentiating (A.38) with respect to t and using (A.40) we obtain

$$u_x(a(t),t) = \frac{1}{a'(t)}(u_a'(t) - g_a(t)). \tag{A.57}$$

Similarly, we obtain

$$u_x(b(t),t) = \frac{1}{b'(t)}(u_b'(t) - g_b(t)). \tag{A.58}$$

Now, taking derivatives with respect to t in (A.53) and evaluation the expression for $x = b(t)$ we obtain

$$(u_x(a(t),t) - u_x(b(t),t) - u_x(a(t),T) + u_x(b(t),T))b'(t) = -\int_{a(t)}^{b(t)} G(s)(T-\ell(s))\,ds - \int_{a(t)}^{b(t)} G(s)(t-\ell(s))\,ds. \tag{A.59}$$

Let $B(t)$ be the left hand side of (A.59). By differentiating twice (A.55) with respect to t, we get

$$G(b(t))b'(t) - G(a(t))a'(t) = Q'(t), \tag{A.60}$$

where

$$Q(t) = \frac{1}{(t-T)b'(t)}\left(A'(t) - \frac{A(t)}{(t-T)}\right).$$

By differentiating (A.59) with respect to t, we get

$$G(b(t))b'(t) - G(a(t))a'(t) = \frac{B'(t) + Q(t)}{t-T}. \tag{A.61}$$

From (A.60), (A.61) we get $G(a(t))$ and $G(b(t))$. Together with $G(x)$, $x \in [a(0), b(0)]$ we have $G(x)$ for all $x \in [a(T), b(T)]$.

From (A.49) we obtain $u_{xx}(x, \ell(x))$. Then we have all ingredient to evaluate $u(x,t)$ using (A.52).

This example illustrates in a simple case how the boundary conditions are used to determine the solution in the two-lid setting.

B On object recognition

B.1 Two more affine invariant quantities

In this Appendix, we would like to present two more affine invariant quantities. Let us denote by $GL(2,\mathbf{R})^+$ the set of all 2×2 matrices with positive determinant. If $A\in GL(2,\mathbf{R})^+$, we denote $u_A(x)=u(Ax)$, $x\in\mathbf{R}^2$.

As we have already discussed in the Object recognition Part of this thesis, if $A\in GL(2,\mathbf{R})^+$, then $\nabla u_A(x)=A^T\nabla u(Ax)$, where A^T denotes the transposed matrix of A. Then

$$\nabla u_A(x)\wedge\nabla u_A(y)=\det(A)\nabla u(Ax)\wedge\nabla u(Ay).$$

Let us consider that x is the center of a circle and $y\in B(x,R)$, $R>0$ being the radius of the circle. The point symmetric of y with respect to x is $S_x(y)=2x-y$. Since collinearity is affine invariant, $S_{Ax}(Ay)=AS_x(y)$ and we have

$$\frac{\nabla u_A(x)\wedge\nabla u_A(y)}{\nabla u_A(x)\wedge\nabla u_A(S_x(y))}=\frac{\det(A)\nabla u(Ax)\wedge\nabla u(Ay)}{\det(A)\nabla u(Ax)\wedge\nabla u(AS_x(y))}$$

$$=\frac{\nabla u(Ax)\wedge\nabla u(Ay)}{\nabla u(Ax)\wedge\nabla u(S_{Ax}(Ay))}.$$

Thus, the quantity

$$Q_x(y)=\frac{\nabla u(x)\wedge\nabla u(y)}{\nabla u(x)\wedge\nabla u(S_x(y))}$$

is affine invariant. Similarly, the quantity

$$\bar{Q}_x(y)=\frac{\nabla u(x)\wedge\nabla u(y)}{\nabla u(x)\wedge\nabla u((x+y)/2)}$$

is affine invariant. Note that the modulus of this quantity is also affine invariant and this form is more adapted to our algorithmic structure. Notice that

$$Q_x(S_x(y))=\frac{1}{Q_x(y)} \tag{B.1}$$

since $S_x(S_x(y))=y$. Hence it suffices to compute $Q_x(y)$ in half of the neighborhood of x, since the other half is determined by the first by (B.1). For $\bar{Q}_x(y)$ we have to compute the descriptor in the whole neighborhood.

B.2 Trying to incorporate invariances by adapting the angular quantization of the histogram of orientations

In this Appendix, we discuss how to improve invariance with respect to some classes of affine transformations by adapting the angular quantization of the histogram of orientations. Assume that we have a descriptor that is rotation and scale invariant. For instance, in SIFT the scale invariance is obtained by using the scale space in the detection of keypoints and the rotation invariance is achieved by normalizing the direction of the gradient at x with respect to the dominant orientation (aligning the dominant orientation with the first coordinate axis).

There are still two missing affine invariances to be dealt with: the tilt parameter and the relative scale of the two coordinate axis. They can be represented by the matrices

$$A_t = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \tag{B.2}$$

$$D_a := \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} \tag{B.3}$$

Let $e_1 = (1,0)^T$, $e_2 = (0,1)^T$. Notice that $A_t e_1 = e_1$, $A_t e_2 = te_1 + e_2$, and $A_t A_s = A_{t+s}$. Note that A_t represents a horizontal shear.

Notice that $D_a e_1 = ae_1$, $D_a e_2 = e2$, and $D_a D_b = D_{ab}$. The deformation induced by D_a on a circle amounts to change the relative size of the axis (change the scale of the first coordinate axis).

Two possibilities present themselves:

1. Construct an orbit including distortions of type D_a and use a descriptor invariant with respect to A_t.

2. Construct an orbit including distortions of type A_t and use a descriptor invariant with respect to D_a.

B.2.1 Angular quantization when the missing distortions are A_t

Let us discuss possibility 1. In the orbit, the missing distortions are those of A_t. Let us describe the distortion induced on gradient directions by the action of A_t on images. Let u be a given image and let $u_{A_t}(x) = u(A_t x)$, $x \in \mathbf{R}^2$. The direction of the gradient of u_{A_t} at x is

$$e(u_{A_t}, x) = \frac{\nabla u_{A_t}(x)}{|\nabla u_{A_t}(x)|} = \frac{A_t^T \nabla u(A_t x)}{|A_t^T \nabla u(A_t x)|} = \frac{A_t^T e(u, A_t x)}{|A_t^T e(u, A_t x)|}$$

where

$$e(u, A_t x) = \frac{\nabla u(A_t x)}{|\nabla u(A_t x)|} = (\cos\theta, \sin\theta),$$

for some $\theta \in [0, 2\pi)$. Thus,

$$e(u_{A_t}, x) = \frac{1}{\lambda}(\cos\,\theta, t\cos\,\theta + \sin\,\theta).$$

where $\lambda = |A_t^T e(u, A_t x)|$.

If for any unit vector $e_\theta = (\cos\,\theta, \sin\,\theta)$, $\theta \in [0, 2\pi)$, we define

$$m(e_\theta) = \frac{\sin\,\theta}{\cos\,\theta} = \tan\,\theta,$$

then

$$m(e(u_{A_t}, x)) = \frac{t\cos\,\theta + \sin\,\theta}{\cos\,\theta} = t + \tan\,\theta$$

As one can see, if we quantize the angular bins uniformly in $\tan\,\theta$, the direction of the transformed gradient will fall in a bin shifted by the quantity t. One could then use a metric such as the Earth Movers Distance to match the descriptors.

Remark. An analogous argument can be done for the case where the missing distortions are D_a.

B.3 A note concerning keypoint detection

The keypoint detection used in SIFT defines an interest point or a keypoint to be a point that achieves an extremum in the scale space of the Laplacian of the Gaussian. In [Low04], that scale space is approximated by the differences-of-Gaussians (DoG), within a difference-of-Gaussians pyramid, as originally proposed by Burt and Adelson [BEA83] and by Crowley and Stern [CS84]. A Gaussian pyramid is constructed from the input image by repeated smoothing and subsampling, and a difference-of-Gaussians pyramid is computed from the differences between the adjacent levels in the Gaussian pyramid. Then, interest points are obtained from the points at which the difference-of-Gaussians values assume an extrema with respect to both the spatial coordinates in the image domain and the scale level in the pyramid. This method for detecting interest points in SIFT can be seen as a variation of a scale-adaptive blob detection method proposed by Lindeberg [Lin94, Lin98], where blobs with associated scale levels are detected from scale-space extrema of the scale-normalized Laplacian.

Notice that we have just mentioned three definitions of keypoints. Before analyzing them, let us recall that the main role of keypoints is to establish correspondences between any two given images $u(x)$ and $v(\overline{x})$ of the same object or scene. Thus, our main purpose here is to relate keypoints of an image u and keypoints of another image v. In other words, given a keypoint of u, are we able to find a corresponding keypoint of v, and conversely?

To fix ideas, let $u(x) = G_\beta H_\lambda u_0(x)$ and $v(\overline{x}) = G_\delta H_\mu u_0(\overline{x})$ be two acquired images of the same ideal image u_0 where H_λ, H_μ are homothecies with

scale factor λ and μ respectively. G_β and G_δ are Gaussian kernels with standard deviations β and δ respectively modeling the camera blur. For any other notation issues, we will use the same notation used in the Object recognition Part of this thesis.

Let $\mathbf{u_0}(s, y) := G_s * u_0(y)$ denote the scale space of u_0. Then the scale space of u can be written as

$$\begin{aligned} G_\sigma * u(x) &= G_\sigma G_\beta H_\lambda u_0(x) \\ &= G_{\sqrt{\sigma^2+\beta^2}} H_\lambda u_0(x) \\ &= H_\lambda G_{\lambda\sqrt{\sigma^2+\beta^2}} u_0(x) \\ &= \mathbf{u_0}(\lambda\sqrt{\sigma^2+\beta^2}, \lambda x). \end{aligned}$$

Similarly, the scale space of v can be written as

$$G_{\overline{\sigma}} * v(\overline{x}) = H_\mu G_{\mu\sqrt{\overline{\sigma}^2+\delta^2}} u_0(\overline{x}) = \mathbf{u_0}(\mu\sqrt{\overline{\sigma}^2+\delta^2}, \mu\overline{x}).$$

From now on, we will use the bold face letters $\mathbf{u}(\sigma, x) := G_\sigma * u(x)$ and $\mathbf{v}(\overline{\sigma}, \overline{x}) := G_{\overline{\sigma}} * v(\overline{x})$ to denote the scale spaces of $u(x)$ and $v(\overline{x})$ respectively. Notice that we distinguished the notation for the variables in both cases. We will also denote by Δ the Laplacian operator and we will use the symbol $\approx$ to read "in the neighborhood of" or "close to". Let us distinguish between three definitions of a keypoint:

Definition B.3.1. *The Laplacian of the Gaussian keypoint (L-kpt).*
We say (s_0, y_0) is a L-kpt of u_0 if and only if

$$\Delta\mathbf{u_0}(s, y) \leq \Delta\mathbf{u_0}(s_0, y_0) \quad \forall (s, y) \approx (s_0, y_0). \tag{B.4}$$

Definition B.3.2. *The DoG keypoint (D-kpt), which is used by SIFT.*
We say (s_0, y_0) is a D-kpt of u_0 if and only if

$$\mathbf{u_0}(ks, y) - \mathbf{u_0}(s, y) \leq \mathbf{u_0}(ks_0, y_0) - \mathbf{u_0}(s_0, y_0) \quad \forall (s, y) \approx (s_0, y_0), \tag{B.5}$$

where k is a constant factor ($k = 2^{1/a}$ where a is the number of scales per octave) separating the Gaussian images of the scale space.

Definition B.3.3. *The scale-normalized Laplacian of the Gaussian keypoint (NL-kpt), which is defined by Lindeberg [Lin94, Lin98].*
We say (s_0, y_0) is a NL-kpt of u_0 if and only if

$$s^2\Delta\mathbf{u_0}(s, y) \leq s_0^2\Delta\mathbf{u_0}(s_0, y_0) \quad \forall (s, y) \approx (s_0, y_0). \tag{B.6}$$

Recall that our basic question was: given a keypoint of u, are we able to find a corresponding keypoint of v, and conversely ? Clearly the answer depends on the definition of a keypoint and is different for each of the three Definitions above. In [MY11] the answer was proved to be affirmative if we use keypoints in the sense of Definition B.3.1. Their result is stated in [MY11] under the name "**Lemma 3**" and reads:

> Let u and v be two digital images that are frontal snapshots of the same continuous flat image u_0, $u = S_1 G_\beta H_\lambda u_0$ and $v = S_1 G_\delta H_\mu u_0$, taken from different distances , with different Gaussian blurs and possible different sampling rates. Let $w(\sigma, x) := (G_\sigma u_0)(x)$ denote the scale space of u_0. Then the scale spaces of u and v are
>
> $$u(\sigma, x) = w(\lambda\sqrt{\sigma^2 + \beta^2}, \lambda x) \text{ and } v(\sigma, x) = w(\mu\sqrt{\sigma^2 + \delta^2}, \mu x).$$
>
> If (s_0, x_0) is a keypoint of w satisfying $s_0 \geq \max(\lambda\beta, \mu\delta)$, then it corresponds to a keypoint of u at the scale σ_1 such that $\lambda\sqrt{\sigma_1^2 + \beta^2} = s_0$, whose SIFT descriptor is sampled with mesh $\sqrt{\sigma_1^2 + c^2}$, where c is the tentative standard deviation of the initial image blur as described in Section 2.3 (Section reference is relative to [MY11]). In the same way (s_0, x_0) corresponds to a keypoint of v at scale σ_2 such that $s_0 = \mu\sqrt{\sigma_2^2 + \delta^2}$, whose SIFT descriptor is sampled with mesh $\sqrt{\sigma_2^2 + c^2}$.

Our purpose here is to analyze the case of Definitions B.3.2 and B.3.3. It turns out that we cannot give a positive answer unless we add to the initial image a set of blurred images. In other words, given an input image u, we do not consider only the scale space of u, but we consider a set of initial images $u_{\sigma_i}(x) = \mathbf{u}(\sigma_i, x)$, where $\sigma_i \in [0, \Sigma]$ for a given range Σ. Then we compute their corresponding scale spaces, denoted by $\mathbf{u}_{\sigma_i}(\sigma, x)$. Note that $\mathbf{u}_{\sigma_i}(\sigma, x) = \mathbf{u}(\sqrt{\sigma^2 + \sigma_i^2}, x)$. We compute the keypoints of $u_{\sigma_i}(x)$ using either Definition B.3.2 or B.3.3.

Using Definition B.3.2 We state the following: Let $(\overline{\sigma}_0, \overline{x}_0)$ be a D-kpt of $v_{\overline{\sigma}_i}(\overline{x})$. Then there is a corresponding D-kpt (σ_0, x_0) of $u_{\sigma_i}(x)$ for a suitable choice of σ_i, σ_0, and x_0.

Let us now prove it. According to (B.5), we have

$$\mathbf{v}_{\overline{\sigma}_i}(k\overline{\sigma}, \overline{x}) - \mathbf{v}_{\overline{\sigma}_i}(\overline{\sigma}, \overline{x}) \leq \mathbf{v}_{\overline{\sigma}_i}(k\overline{\sigma}_0, \overline{x}_0) - \mathbf{v}_{\overline{\sigma}_i}(\overline{\sigma}_0, \overline{x}_0) \quad \forall (\overline{\sigma}, \overline{x}) \approx (\overline{\sigma}_0, \overline{x}_0).$$

In terms of $\mathbf{v}$, this translates to the following

$$\mathbf{v}(\sqrt{k^2\overline{\sigma}^2 + \overline{\sigma}_i^2}, \overline{x}) - \mathbf{v}(\sqrt{\overline{\sigma}^2 + \overline{\sigma}_i^2}, \overline{x}) \leq \mathbf{v}(\sqrt{k^2\overline{\sigma}_0^2 + \overline{\sigma}_i^2}, \overline{x}_0) - \mathbf{v}(\sqrt{\overline{\sigma}_0^2 + \overline{\sigma}_i^2}, \overline{x}_0)$$
$$\forall (\overline{\sigma}, \overline{x}) \approx (\overline{\sigma}_0, \overline{x}_0).$$

Finally, in terms of $\mathbf{u_0}$ we have

$$\begin{aligned}&\mathbf{u_0}(\mu\sqrt{k^2\overline{\sigma}^2+\overline{\sigma}_i^2+\delta^2},\mu\overline{x})-\mathbf{u_0}(\mu\sqrt{\overline{\sigma}^2+\overline{\sigma}_i^2+\delta^2},\mu\overline{x})\leq\\&\mathbf{u_0}(\mu\sqrt{k^2\overline{\sigma}_0^2+\overline{\sigma}_i^2+\delta^2},\mu\overline{x}_0)-\mathbf{u_0}(\mu\sqrt{\overline{\sigma}_0^2+\overline{\sigma}_i^2+\delta^2},\mu\overline{x}_0)\quad\forall(\overline{\sigma},\overline{x})\approx(\overline{\sigma}_0,\overline{x}_0).\end{aligned}\tag{B.7}$$

Now, let (σ_0, x_0) be a D-kpt of u_{σ_i}. An analogous derivation can be done and we get in terms of $\mathbf{u_0}$ the following expression

$$\begin{aligned}&\mathbf{u_0}(\lambda\sqrt{k^2\sigma^2+\sigma_i^2+\beta^2},\lambda x)-\mathbf{u_0}(\lambda\sqrt{\sigma^2+\sigma_i^2+\beta^2},\lambda x)\leq\\&\mathbf{u_0}(\lambda\sqrt{k^2\sigma_0^2+\sigma_i^2+\beta^2},\lambda x_0)-\mathbf{u_0}(\lambda\sqrt{\sigma_0^2+\sigma_i^2+\beta^2},\lambda x_0)\quad\forall(\sigma,x)\approx(\sigma_0,x_0).\end{aligned}\tag{B.8}$$

Given the keypoint $(\overline{\sigma}_0, \overline{x}_0)$ for v, to find the corresponding keypoint of u we should have the following equations:

$$\lambda\sqrt{k^2\sigma_0^2+\sigma_i^2+\beta^2}=\mu\sqrt{k^2\overline{\sigma}_0^2+\overline{\sigma}_i^2+\delta^2}\tag{B.9}$$

$$\lambda\sqrt{\sigma_0^2+\sigma_i^2+\beta^2}=\mu\sqrt{\overline{\sigma}_0^2+\overline{\sigma}_i^2+\delta^2}\tag{B.10}$$

$$\lambda x_0=\mu\overline{x}_0\tag{B.11}$$

By appropriately choosing σ_i, equations (B.9), (B.10) and (B.11) permit to compute a D-kpt (σ_0, x_0) of u_{σ_i}. Indeed, from (B.11), we can directly compute $x_0 = \frac{\mu}{\lambda}\overline{x}_0$. We are left with two equations and two unknowns, specifically σ_0 and σ_i. Solving this system of equations we get the following solutions for the remaining variables

$$\sigma_0^2=\left(\frac{\mu}{\lambda}\right)^2\overline{\sigma}_0^2,\tag{B.12}$$

$$\sigma_i^2=\left(\frac{\mu}{\lambda}\right)^2\overline{\delta}^2-\beta^2.\tag{B.13}$$

Clearly, to solve the above equations, since the value of $\sigma_i \geq 0$ we need that $\left(\frac{\lambda}{\mu}\right)^2(\overline{\sigma}_i^2+\delta^2)-\beta^2\geq 0$.

λ and μ correspond to the camera distance to the object being observed in u_0 and therefore their values are unknown (and arbitrary). β and δ correspond to the optical blur of the camera which varies from one camera to another, and therefore cannot be set. However, from the previous argument, and given identities (B.12) and (B.13) we know that, given a D-kpt in v, there exist a corresponding keypoint in u for some σ_i that depends on λ, μ, β and δ which are either unknowns or arbitrary. From that, it is clear that there is a need to build a kind of orbit of scale spaces with different σ_i. The range Σ should enable us to achieve sufficient correspondences.

Using Definition B.3.3 Let $(\overline{\sigma}_1, \overline{x}_1)$ be a NL-kpt of $v_{\overline{\sigma}_i}(\overline{x})$. Then there is a corresponding NL-kpt (σ_1, x_1) of $u_{\sigma_i}(x)$ for a suitable choice of σ_i, σ_1, and x_1.

According to (B.6), we have

$$\overline{\sigma}^2 \Delta \mathbf{v}_{\overline{\sigma}_i}(\overline{\sigma}, \overline{x}) \leq \overline{\sigma}_1^2 \Delta \mathbf{v}_{\overline{\sigma}_i}(\overline{\sigma}_1, \overline{x}_1) \quad \forall (\overline{\sigma}, \overline{x}) \approx (\overline{\sigma}_1, \overline{x}_1).$$

In terms of $\mathbf{v}$, this translates to the following

$$\overline{\sigma}^2 \Delta \mathbf{v}(\sqrt{\overline{\sigma}^2 + \overline{\sigma}_i^2}, \overline{x}) \leq \overline{\sigma}_1^2 \Delta \mathbf{v}(\sqrt{\overline{\sigma}_1^2 + \overline{\sigma}_i^2}, \overline{x}_1) \quad \forall (\overline{\sigma}, \overline{x}) \approx (\overline{\sigma}_1, \overline{x}_1).$$

And, in terms of $\mathbf{u_0}$ we have

$$\overline{\sigma}^2 \Delta \mathbf{u_0}(\mu\sqrt{\overline{\sigma}^2 + \overline{\sigma}_i^2 + \delta^2}, \mu\overline{x}) \leq \overline{\sigma}_1^2 \Delta \mathbf{u_0}(\mu\sqrt{\overline{\sigma}_1^2 + \overline{\sigma}_i^2 + \delta^2}, \mu\overline{x}_1)\, \forall (\overline{\sigma}, \overline{x}) \approx (\overline{\sigma}_1, \overline{x}_1). \tag{B.14}$$

Now, let (σ_1, x_1) be a NL-kpt of u_{σ_i}. An analogous derivation can be done and we get in terms of $\mathbf{u_0}$ the following expression

$$\sigma^2 \Delta \mathbf{u_0}(\lambda\sqrt{\sigma^2 + \sigma_i^2 + \beta^2}, \lambda x) \leq \sigma_1^2 \Delta \mathbf{u_0}(\lambda\sqrt{\sigma_1^2 + \sigma_i^2 + \beta^2}, \lambda x_1)\, \forall (\sigma, x) \approx (\sigma_1, x_1). \tag{B.15}$$

Given the keypoint $(\overline{\sigma}_1, \overline{x}_1)$ for v, to find the corresponding keypoint of u we should have the following equations:

$$\lambda\sqrt{\sigma_1^2 + \sigma_i^2 + \beta^2} = \mu\sqrt{\overline{\sigma}_1^2 + \overline{\sigma}_i^2 + \delta^2} \tag{B.16}$$

$$\lambda\sqrt{\sigma^2 + \sigma_i^2 + \beta^2} = \mu\sqrt{\overline{\sigma}^2 + \overline{\sigma}_i^2 + \delta^2} \tag{B.17}$$

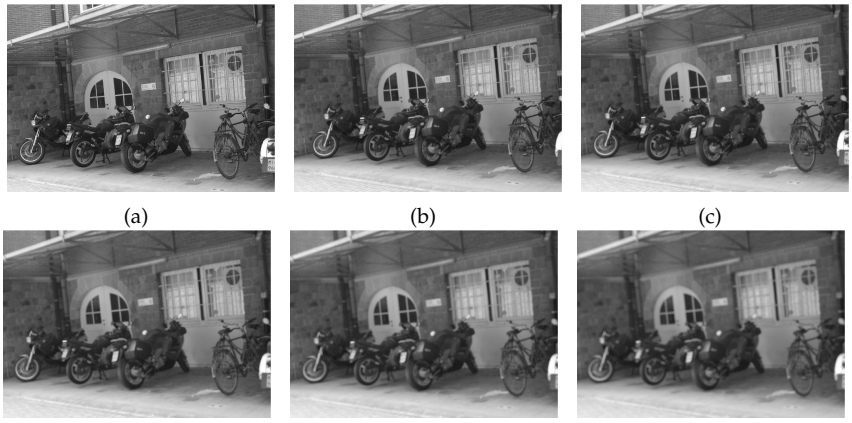
$$\lambda x_1 = \mu\overline{x}_1 \tag{B.18}$$

By appropriately choosing σ_i, equations (B.16), (B.17) and (B.18) permit to compute a NL-kpt (σ_1, x_1) of u_{σ_i}. Indeed, from (B.18) we can directly compute $x_1 = \frac{\mu}{\lambda}\overline{x}_1$. From (B.16), we are able to compute $\sigma_1^2 = \left(\frac{\mu}{\lambda}\right)^2 \left(\overline{\sigma}_1^2 + \overline{\sigma}_i^2 + \delta^2\right) - \sigma_i^2 - \beta^2$. Similarly, from (B.17), we are able to compute $\sigma^2 = \left(\frac{\mu}{\lambda}\right)^2 \left(\overline{\sigma}^2 + \overline{\sigma}_i^2 + \delta^2\right) - \sigma_i^2 - \beta^2$. Substituting σ_1^2 and σ^2 by these expressions in equation (B.15) we can extract the following equation

$$\left(\frac{\lambda}{\mu}\right)^2 (\sigma_i^2 + \beta^2) - \overline{\sigma}_i^2 + \delta^2 = 0,$$

in order to satisfy the keypoint definition. This permits us to compute $\sigma_i^2 = \left(\frac{\mu}{\lambda}\right)^2 \overline{\sigma}_i^2 + \delta^2 - \beta^2$. Clearly, to solve the above equations, since the value of $\sigma_i \geq 0$ we need that $\left(\frac{\mu}{\lambda}\right)^2 \overline{\sigma}_i^2 + \delta^2 - \beta^2 \geq 0$.

We can add the same comments as in the end of the last paragraph on D-kpt.



(a) (b) (c)

(d) (e) (f)

Figure B.1: Image dataset with increasing camera blur from left to right and top to bottom.

B.3.1 Preliminary result

Let us now test experimentally our proposal of building an orbit of scale spaces simulating different σ_i's. In our experiment, we use both Definitions B.3.2 and B.3.3 for a keypoint. Notice that simulating different σ_is can be done in an equal manner by simulating different initial guesses of the camera blur (In SIFT, the camera blur guess is 0.5). We make this remark only to fit in a better way the implementation of SIFT described in [Low04] and we use this remark to implement our experiment. We also propose to take the union of the detected keypoints for each σ_i to be the final set of keypoints. The SIFT descriptor is then computed and matching is performed. We run an experiment on a set of six images where the camera blur is being increased from one image to another. The dataset can be seen in Figure B.1.

Let the first image (Figure B.1(a)) be the *query* image where we assume that the camera blur is known and set to 0.5 as proposed by SIFT. For the other images, call them the *database* images, we simulate different camera blurs which amounts to simulating as well different σ_is. The camera blurs we simulate are $\alpha_n \in \{2^{-1}, 2^{-0.75}, 2^{-0.5}, 2^{-0.25}, 2^0, 2^{0.25}, 2^{0.5}, 2^{0.75}, 2^1\}$ with $n = 0, \cdots, 8$. Let κ_{α_n} denote the set of keypoints obtained by the SIFT keypoint detection (for any given definition of a keypoint) assuming a camera blur of α_n. For any α_n, we define the union of the keypoints $K_{\alpha_n} = \kappa_{\alpha_n} \cup \kappa_{\alpha_{n-1}} \cup \kappa_{\alpha_{n-2}} \cup \cdots \cup \kappa_{\alpha_0}$. This will be the set of keypoints that we use for the matching. Let us also mention that repeated elements of the set K_{α_n} are removed. Figure B.2 shows the result. In each graph, the x-axis shows the chosen value of α_n. For a fixed α_n, we compute the number of correct correspondences (shown in the y-axis) using the set of keypoints K_α_n made of the union of all found keypoints for $\alpha_j \leq \alpha_n$. Let us mention that the matching strategy used is the nearest neighbor strategy (see

Section 12.2.1).

Notice how the number of correctly matched keypoints increases when simulating different camera blurs. Also, notice that depending on the initial camera blur of the database image, the nb of correct correspondences saturates at a different α_n.

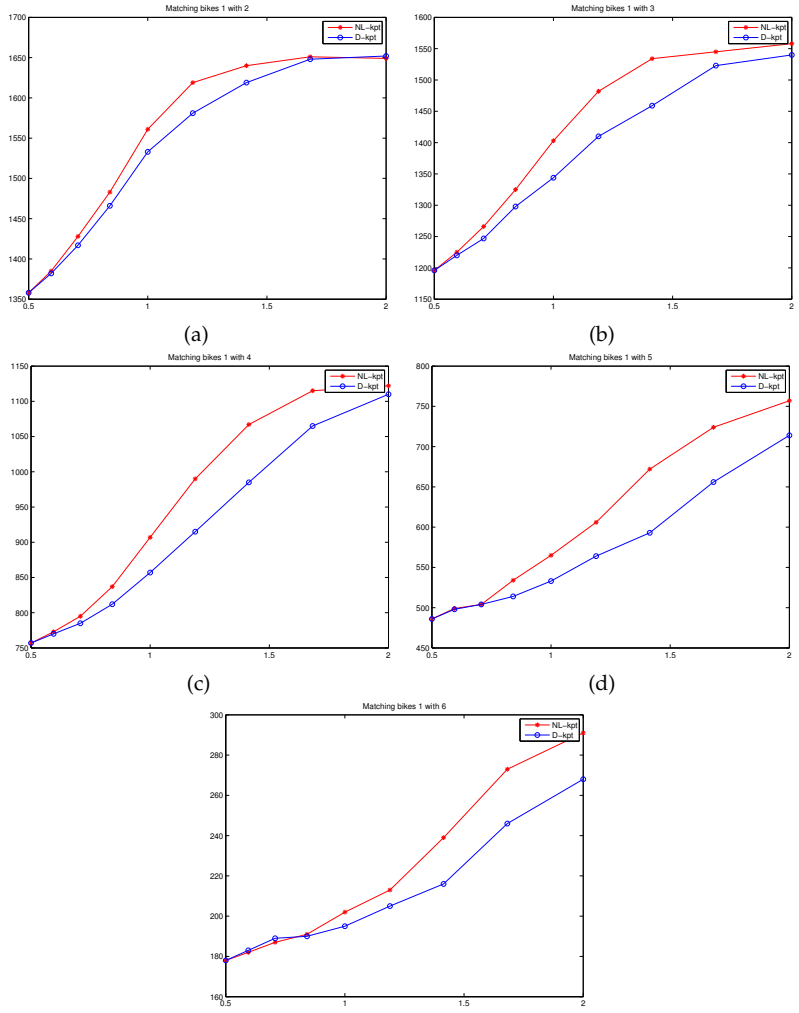


Figure B.2: Graphs showing the number of correct correspondences (y-axis) found for different values of α_n (x-axis). As a matching strategy we use the nearest neighbor (NN) matching strategy. Note that the bottom left node, when $\alpha_n = 0.5$ correspond to running the standard SIFT keypoint detection without adding the proposed simulation of different camera blurs. (a), (b), (c), (d) and (e) show the result of matching image B.1(a) with B.1(b), B.1(c), B.1(d), B.1(e) and B.1(f) respectively.

C Published work

Most of the research reported above has been published in the following peer reviewed journals:

- R. Sadek, G. Facciolo, P. Arias, and V. Caselles. "A variational model for gradient-based video editing". 2012. Submitted to IJCV for publication (second revision).
- R. Sadek, C. Constantinopulos, E. Meinhardt, C. Ballester, and V.Caselles. "On affine invariant descriptors related to SIFT". SIAM Journal on Imaging Sciences (SIIMS), 2012, Vol. 5 issue 2, 652-687.
- P. Arias, V. Caselles, G. Facciolo, V. Lazcano, and R. Sadek. "Nonlocal variational models for inpainting and interpolation". Mathematical Models and Methods in Applied Sciences, 2012, Vol 22. Issue supp02.

and peer reviewed conferences:

- R. Sadek, C. Ballester, L. Garrido, E. Meinhardt, and V. Caselles. "Frame interpolation with occlusion detection using a time coherent segmentation". In International Conference on Computer Vision Theory and Applications (VISAPP), Roma, Italy, February 2012.
- G. Facciolo, R. Sadek, Aurelie Bugeau, and Vicent Caselles. "Temporally consistent gradient domain video editing". In Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR), Vol 6819 of Lecture Notes in Computer Science, pages 59-73. Springer, 2011.

Bibliography

[AC08] L. Ambrosio and G. Crippa. Existence, uniqueness, stability and differentiability properties of the flow associated to weakly differentiable vector fields. *Transport equations and multi-D hyperbolic conservation laws*, pages 3–57, 2008.

[ACM05] L. Ambrosio, G. Crippa, and S. Maniglia. Traces and fine properties of a bd class of vector fields and applications. *Ann. Fac. Sci. Toulouse Math.(6)*, 14(4):527–561, 2005.

[Ada75] R. Adams. Sobolev spaces. 1975, 1975.

[ADPS07] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *Int. Journal of Comp. Vis.*, 75:371–385, 2007.

[AFCS11] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro. A variational framework for exemplar-based image inpainting. *International Journal of Computer Vision (IJCV)*, 93:319–347, July 2011.

[AFP00] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of bounded variation and free discontinuity problems*. Oxford University Press, USA, 2000.

[AGLM93] L. Alvarez, F. Guichard, P. Lions, and J. Morel. Axioms and fundamental equations of image processing. *Archive for Rational Mechanics and Analysis*, 123(3):199–257, 1993.

[Amb04] L. Ambrosio. Transport equation and cauchy problem for bv vector fields. *Inventiones Mathematicae*, 158(2):227–260, 2004.

[Amb08] L. Ambrosio. Transport equation and cauchy problem for non-smooth vector fields. *Calculus of variations and nonlinear partial differential equations*, pages 1–41, 2008.

[Anz83] G. Anzellotti. Pairings between measures and bounded functions and compensated compactness. *Annali di Matematica Pura ed Applicata*, 135(1):293–318, 1983.

[AR07] A. Agrawal and R. Raskar. Gradient domain manipulation techniques in vision and graphics, 2007. ICCV'07 short course.

[ARS11] A. Ayvaci, M. Raptis, and S. Soatto. Sparse occlusion detection with optical flow. *International Journal of Computer Vision (IJCV)*, (in press) 2011.

[Bal95] C. Ballester. Affine invariant segmentation by variational method. *Ph.D.*, 1995.

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, volume 3024 of *LNCS*, pages 25–36. Springer, 2004.

[BCG96] C. Ballester, V. Caselles, and M. González. Affine invariant segmentation by variational method. *SIAM Journal on Applied Mathematics*, pages 294–325, 1996.

[BEA83] P. J. Burt, Edward, and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[BGD+10] A. Bugeau, P. Gargallo, O. D'Hondt, A. Hervieu, N. Papadakis, and V. Caselles. Coherent Background Video Inpainting through Kalman Smoothing along Trajectories. In *Modeling, and Visualization Workshop*, page 8, 2010.

[BHNR92] A. Bruckstein, R. Holt, A. Netravali, and T. Richardson. Invariant signatures for planar shape recognition under partial occlusion. In *International Conference on Pattern Recognition*, pages 108–108. IEEE Computer Society Press, 1992.

[BL02] M. Brown and D. Lowe. Invariant features from interest point groups. In *British Machine Vision Conference, Cardiff, Wales*, pages 656–665. Citeseer, 2002.

[BM11] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(3):500–513, 2011.

[BSFG09] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: a randomized correspondence algorithm for structural image editing. In *Proc. of SIGGRAPH*, pages 1–11, 2009.

[BSL+11] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[BTVG06] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[BZCC10] P. Bhat, C. L. Zitnick, M. Cohen, and B. Curless. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Transactions on Graphics*, 29:1–14, April 2010.

[BZM07] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, volume 7. Citeseer, 2007.

[BZM08] A. Bosch, A. Zisserman, and X. Muñoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4):712, 2008.

[BZS+07] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, M. F. Cohen, B. Curless, and S. B. Kang. Using photographs to enhance videos of a static scene. In *Proceedings of the Eurographics Symposium on Rendering Techniques*, pages 327–338. Eurographics Association, 2007.

[Cha04] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1-2):89–97, 2004.

[CL05] F. Colombini and N. Lerner. Uniqueness of l solutions for a class of conormal bv vector fields. *Contemporary Mathematics*, 368:133, 2005.

[CLM+08] F. Cao, J. Lisani, J. Morel, P. Musé, and F. Sur. *A theory of shape identification*. Springer Verlag, 2008.

[CM02] V. Caselles and P. Monasse. Grain filters. *Journal of Mathematical Imaging and Vision*, 17(3):249–270, 2002.

[CM10] V. Caselles and P. Monasse. Geometric description of topographic maps and applications to image processing. *Lecture Notes in Mathematics*, 1984, 2010.

[CP11] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[CPT04a] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004.

[CPT04b] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. on IP*, 13:1200–1212, 2004.

[CRT90] C. Cafforio, F. Rocca, and S. Tubaro. Motion compensated image interpolation. *IEEE Transactions on Communications*, 38:215–222, 1990.

[CS84] J. Crowley and R. Stern. Fast computation of the difference of low-pass transform. *Pattern Anal. Mach. Intell.*, 6:212–222, 1984.

[DL89] R. DiPerna and P. Lions. Ordinary differential equations, transport theory and sobolev spaces. *Inventiones mathematicae*, 98(3):511–547, 1989.

[DN04] G. Dane and T. Nguyen. Motion vector processing for frame rate up conversion. In *IEEE ICASSP*, volume 3, pages 309–312, 2004.

[DT05] N. Dalai and B. Triggs. Histogram of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[FHLD06] G. D. Finlayson, S. D. Hordley, C. Lu, and M. S. Drew. On the removal of shadows from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(1):59–68, 2006.

[FLW02] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. *ACM Transactions on Graphics*, 21:249–256, July 2002.

[FP02] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 1 edition, August 2002.

[Geo05] T. Georgiev. Image reconstruction invariant to relighting. In *Eurographics*, pages 61–4, 2005.

[GL96] J. Gårding and T. Lindeberg. Direct computation of shape cues using scale-adapted spatial derivative operators. *International Journal of Computer Vision*, 17(2):163–191, 1996.

[GTT01] L. Gool, T. Tuytelaars, and A. Turina. Local Features for Image Retrieval. In *State-of-the-Art in Content-Based Image and Video Retrieval [Dagstuhl Seminar, 5-10 December 1999]*, page 41. Kluwer, BV, 2001.

[Hor86] B. K. Horn. *Robot Vision*. Electrical Engineering and Computer Science. MIT Press, 1986.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, volume 15, page 50. Manchester, UK, 1988.

[HSB09] E. Herbst, S. Seitz, and S. Baker. Occlusion reasoning for temporal interpolation using optical flow. In *Technical report UW-CSE-09-08-01*. Dept. of Comp. Sci. and Eng., University of Washington, 2009.

[HZ03] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[IK08] S. Ince and J. Konrad. Occlusion-aware optical flow estimation. *IEEE Trans. on IP*, 17:1443–1451, 2008.

[JH99] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3Dscenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[JTWT06] J. Jia, Y.-W. Tai, T.-P. Wu, and C.-K. Tang. Video repairing under variable illumination using cyclic motions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):832–9, 2006.

[KCR05] A. Kokaram, B. Collis, and S. Robinson. Automated rig removal with bayesian motion interpolation. *IEEE Journal on Vision, Image and Signal Processing*, 152:407–414, Aug 2005.

[KEBK05] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics*, 24(3):795–802, 2005.

[KLM94] G. Koepfler, C. Lopez, and J. Morel. A multiscale algorithm for image segmentation by variational method. *SIAM Journal on Numerical Analysis*, 31:282–299, 1994.

[KS04] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2004.

[KT07] N. Komodakis and G. Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Transactions on Image Processing (TIP)*, 16(11):2649–61, 2007.

[KZBB10] R. Kimmel, C. Zhang, E. Bronstein, and M. Bronstein. Are MSER features really interesting ? *available at http://www.cs.technion.ac.il/ ron/publications.html*, to appear, 2010.

[LF06] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465, 2006.

[LFAW] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss. `http://people.csail.mit.edu/celiu/motionAnnotation/`.

[LFAW08] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss. Human-assisted motion annotation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2008.

[LG97] T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure* 1. *Image and Vision Computing*, 15(6):415–434, 1997.

[Lin94] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[Lin98] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[LLF05] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, page 775. Citeseer, 2005.

[LLM10] C. Linz, C. Lipski, and M. Magnor. Multi-image interpolation based on graph-cuts and symmetric optical flow. In *15th International Workshop on Vision, Modeling and Visualization (VMV)*, pages 115–122, 2010.

[LMMM03] J. Lisani, L. Moisan, P. Monasse, and J. Morel. On the theory of planar shape. *SIAM Multiscale Modeling and Simulation*, 1(1):1–24, 2003.

[Low99] D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[Low04] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSP04] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local affine parts for object recognition. In *British machine vision conference*, volume 2, pages 959–968. Citeseer, 2004.

[LYT+] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman. SIFT flow: dense correspondence across different scenes. *Computer Vision–ECCV 2008*, pages 28–42.

[MCUP04] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[Mei11] E. Meinhardt. Morphological and statistical techniques for the analysis of 3d images. *Ph.D. Thesis*, 2011.

[MG00] P. Monasse and F. Guichard. Fast computation of a contrast-invariant image representation. *IEEE Transactions on Image Processing*, 9(5):860–872, 2000.

[MHM+09] D. Mahajan, F. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: a path-based method for plausible image interpolation. In *ACM SIGGRAPH*, volume 28, pages 1–11, 2009.

[Mika] K. Mikolajczyk. `http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries`.

[Mikb] K. Mikolajczyk. http://www.robots.ox.ac.uk/~vgg/research/affine/index.html.

[Mikc] K. Mikolajczyk. http://www.robots.ox.ac.uk/~vgg/research/affine/desc_evaluation.html#code.

[MLS12] E. Meinhardt-Llopis and J. Sánchez. Horn-schunck optical flow with a multi-scale strategy. 2012. preprint.

[MMVG99] F. Mindru, T. Moons, and L. Van Gool. Recognizing color patterns irrespective of viewpoint and illumination. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999.

[Mon99] P. Monasse. Contrast invariant registration of images. In *Acoustics, Speech, and Signal Processing, 1999. ICASSP'99. Proceedings., 1999 IEEE International Conference on*, volume 6, pages 3221–3224. IEEE, 1999.

[MS04a] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[MS04b] L. Moisan and B. Stival. A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix. *International Journal of Computer Vision*, 57(3):201–218, 2004.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[MTS+05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1):43–72, 2005.

[MY09] J. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[MY11] J. Morel and G. Yu. Is sift scale invariant? *Inverse Problems and Imaging*, 5(1):115–136, 2011.

[Neg98] S. Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(9):961–979, 1998.

[OCLF09] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1):5555, 2009.

[OST+94] P. Olver, G. Sapiro, A. Tannenbaum, et al. Differential invariant signatures and flows in computer vision: A symmetry group approach. *Geometry driven diffusion in computer vision*, 1994.

[PBB+06] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision (IJCV)*, 67(2):141–158, April 2006.

[PCI+07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–8, 2007.

[PGB03] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics*, 22:313–318, 2003.

[PH03] D. Pritchard and W. Heidrich. Cloth motion capture. In *Computer Graphics Forum*, volume 22, pages 263–271. Wiley Online Library, 2003.

[RAKRF08] A. Rav-Acha, P. Kohli, C. Rother, and A. Fitzgibbon. Unwrap mosaics: A new representation for video editing. *ACM Transactions on Graphics (SIGGRAPH 2008)*, August 2008.

[RD06] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *European Conference on Computer Vision–ECCV 2006*, pages 430–443, 2006.

[ROF92] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica*, D(60):259–268, 1992.

[Ser82] J. Serra. Image analysis and mathematical morphology. *New York*, 1982.

[SFAC] R. Sadek, G. Facciolo, P. Arias, and V. Caselles. `http://www.dtic.upf.edu/~rsadek/gbve/`.

[SH08] M. Sarfraz and O. Hellwich. An efficient front-end facial pose estimation system for face recognition. *Pattern Recognition and Image Analysis*, 18(3):434–441, 2008.

[SJTS04] J. Sun, J. Jia, C. K. Tang, and H. Y. Shum. Poisson matting. *ACM Transactions on Graphics*, 23(3):315–321, 2004.

[SMTK06] T. Shiratori, Y. Matsushita, X. Tang, and S. B. Kang. Video completion by motion field transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 411–418, 2006.

[SRB] D. Sun, S. Roth, and M. J. Black. `http://www.cs.brown.edu/~qsun/research/software.html`.

[SRB10] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2439, 2010.

[SS95] P. Salembier and J. Serra. Flat zones filtering, connected operators, and filters by reconstruction. *IEEE Transactions on image processing*, 4(8):1153–1160, 1995.

[SS07] A. Salgado and J. Sánchez. Temporal constraints in large optical flow estimation. In *Proc. of the 11th Int. Conf. on Comp. Aided Systems Theory*, EUROCAST'07, pages 709–716, Berlin, Heidelberg, 2007. Springer-Verlag.

[SSB12] D. Sun, E. Sudderth, and M. J. Black. Layered segmentation and optical flow estimation over time. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR*, 2012.

[ST93] G. Sapiro and A. Tannenbaum. Affine invariant scale-space. *International Journal of Computer Vision*, 11(1):25–44, 1993.

[ST94] G. Sapiro and A. Tannenbaum. On affine plane curve evolution. *Journal of Functional Analysis*, 119(1):79–120, 1994.

[ST08] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision (IJCV)*, 80:72–91, 2008.

[SZ03] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Ninth IEEE international conference on computer vision, 2003. Proceedings*, pages 1470–1477, 2003.

[TVG99] T. Tuytelaars and L. Van Gool. Content-based image retrieval based on local affinely invariant regions. In *Visual Information and Information Systems, Lecture Notes in Computer Science*, pages 493–500. Springer, 1999.

[UGVT88] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–87, 1988.

[VBVZ11] S. Volz, A. Bruhn, L. Valgaerts, and H. Zimmer. Modeling temporal coherence for optical flow. In *Proc. Thirteenth Int. Conf. on Computer Vision*, Barcelona, November 2011. IEEE Computer Society Press.

[WBBP06] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg. A survey on variational optic flow methods for small displacements. In O. Scherzer, editor, *Mathematical Models for Registration and Applications to Medical Imaging*, volume 10 of *Mathematics in Industry*. Springer, Berlin, 2006.

[Wey97] H. Weyl. *The classical groups. The invariants and representations*. 15 Edition, Princeton Academic Press, 1997.

[WXRA05] H. Wang, X. Xu, R. Raskar, and N. Ahuja. Videoshop: A new framework for spatio-temporal video editing in gradient domain. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[YM] G. Yu and J. Morel. `http://dx.doi.org/10.5201/ipol.2011.my-asift`.

[YM11] G. Yu and J.-M. Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *Image Processing On Line, DOI:10.5201/ipol.2011.my-asift*, 2011.

[Zie89] W. Ziemer. *Weakly differentiable functions: Sobolev spaces and functions of bounded variation*, volume 120. Springer, 1989.

[ZMQ98] M. H. Zhou, M. Mascagni, and A. Y. Qiao. Explicit Finite Difference Schemes for the Advection Equation. *Relation*, 10(1.55):7098, 1998.

[ZXS05] Y. Zhang, J. Xiao, and M. Shah. Motion Layer Based Object Removal in Videos. In *7th IEEE Workshops on Application of Computer Vision*, 2005.