ISSN 1688-2784

Universidad de la República
Facultad de Ingeniería

# One-shot three-dimensional scene analysis

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Matías Di Martino

en cumplimiento parcial de los requerimientos

para la obtención del título de
Doctor en Ingeniería Eléctrica.

Directores de Tesis

Dr. José A. Ferrari ..... Universidad de la República
Prof. Alicia Fernádez ..... Universidad de la República

Tribunal

Dr. Alvaro Pardo ..... Universidad Católica del Uruguay
Dr. Gregory Randall ..... Universidad de la República
Dr. Steffen Härtel ..... Universidad de Chile
Dr. Nicolas Perez de la Blanca ..... Universidad de Granada
Dr. José A. Ferrari ..... Universidad de la República
Prof. Alicia Fernádez ..... Universidad de la República

Director Académico

Prof. Alicia Fernández ..... Universidad de la República

Montevideo
26 de mayo de 2015

*One-shot three-dimensional scene analysis*, Matías Di Martino.

ISSN 1688-2784

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).

Contiene un total de 196 páginas.

Compilada el 26 de mayo de 2015.

http://iie.fing.edu.uy/

Make everything as simple as possible, but not simpler.

Albert Einstein (1879-1955)

# Acknowledgments

In a first place, I would like to especially thank my advisors José Ferrari and Alicia Fernández for his constant guidance and support during the last five years. In particular, for their enthusiasm and constructive criticism, and for the broad and inspiring outlook they gave me for my research. Their advice in both my research and my personal career have been invaluable.

To the committee members, Prof. Alvaro Pardo, Prof. Gregory Randall, Prof. Steffen Härtel and Prof. Nicolas Perez de la Blanca, for their time and dedication.

To all my friends and colleagues in the Physics (IFFI) and Electrical Engineering (IIE) departments for making my experience at the University an extremely enjoyable stage of my life. Especially, I want to thank those members of the so called "La Sala 8" with whom I grew as a professional and as a person: Rodrigo Alonso, Pablo Iturralde, Guzman Hernández, Diego Oroño, Santiago Ibañez, Agustin Laguarda, Gaston Marin, German Fierro, Marcelo Forets, Santiago Paternain among many others friends.

To my friends and colleagues in Cachan for their friendship and support during my stay in Paris: Lara Raad, Martin Rais, Nicola Pierazzo, Aina Frau and Hernan Ponse De Leon. And to Andrés Almansa for receiving and guiding me during my stage in Telecom Paristech.

To all my IFFI fellows for giving me the possibility to work in a comfortable and pleasant environment. To all the members of the Applied Optics and the Image Processing groups for very fruitful and interesting discussions and for supporting my research; to Gaston Ayubi with whom I worked closely in the past three years.

I would also like to acknowledge financial support from ANII and CSIC, with-

out with my research would have been impossible.

Finally I want to especially thank my beloved family and Natalia for their constant support and unconditional love.

*To Natalia and my family.*

# Abstract

This thesis studies the problem of extracting three-dimensional information of a scene by means of projecting structured light over it. Instead of measuring the depth of the scene as most techniques do, we propose a novel and efficient approach to estimate depth's gradient map. This information is extremely useful if we want to recognize or automatically understand what we are looking at. For example we will show how gradient information obtained with the proposed technique can be used for face recognition.

We also study the problem of how to integrate a retrieved gradient field; these techniques combined with the proposed projection approach allows us to estimate the height map of a scene. The main difficulty of dealing with empirically obtained gradient fields is that they are not-integrable (i.e. they are not irrotational fields); hence we must use robust techniques that overcome this issue. In addition, to study the two-dimensional integration problem, more general approaches are also considered; we will show that these approaches can integrate gradient fields of dynamic scenes exploiting the data's temporal correlations to regularize the space of solutions.

# Resumen

En esta tesis estudiaremos el problema de extracción de información tridimensional de una escena mediante la proyección de luz estructurada. En lugar de medir directamente la profundidad de la superficie, proponemos una técnica que nos permite de manera eficiente estimar el campo de gradientes de la profundidad. Esta información es extremadamente útil en distintas áreas de la visión por ordenador cuyo objetivo es detectar objetos o entender de manera automática una escena. A modo de ejemplo, mostraremos como la información obtenida mediante la técnica propuesta puede ser utilizada para el reconocimiento facial.

También estudiaremos como es posible integrar el campo de gradientes calculado mediante la técnica propuesta. La mayor dificultad que debemos afrontar al integrar un campo de gradientes obtenido experimentalmente, radica en que debido al ruido, el campo que poseemos no es irrotacional, y por lo tanto no es integrable. Es necesario encontrar y definir métodos de integración robustos que puedan obtener una solución a este problema. En este trabajo en particular, extenderemos algunas de las técnicas de integración para problemas en los cuales los campos de gradientes poseen una dependencia temporal, que puede ser explotada para imponer distintos tipos de regularización.

# Preface

This thesis embraces all the efforts I put during the last four years as a PhD student at Facultad de Ingeniería, Universidad de la República. I have been working under the supervision of Prof. José Ferrari and Prof. Alicia Fernández. If you find any error or have any comment, I will be pleased to receive it at *matiasdm@fing.edu.uy*. In addition, most of the code and experiments here presented plus some supplementary materials will be publicly available at author's personal web page (www.fing.edu.uy/˜matiasdm).

To keep the presentation of the different topics clearer, and to make the reading of this thesis more fluent, we decide to comment all important implementation concerns within the code in Appendix D.

In this document it is assumed that the reader has a strong background in mathematics and signal processing techniques. Some books that I found very interesting and that may complement and extend many of the topics here presented are: "Two Dimensional Phase Unwrapping: Theory, Algorithms and Software" by Ghiglia and Pritt, "Multiple View Geometry in Computer Vision" by Hartley and Zisserman, "Optical shop testing" by Malacara, and "Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations" by Aubert and Kornprobst.

Matias Di Martino

Montevideo

December 2014

# Table of contents

**Acknowledgments** III

**Abstract** VII

**Preface** XI

**1. Introduction** 1

1.1. Thesis goal . . . . . 4

1.2. Outline . . . . . 4

**2. 3D scene reconstruction** 7

2.1. Overview of standard 3D-shape profiling methods . . . . . 7

2.1.1. Stereo vision principle . . . . . 11

2.1.2. Structured light methods . . . . . 12

2.2. Gradient approach . . . . . 21

2.2.1. Geometrical interpretation and method overview . . . . . 24

2.2.2. Evaluation and comparison with other methods. . . . . 30

2.3. Conclusions . . . . . 44

**3. Surface reconstruction from gradient fields** 47

3.1. Introduction . . . . . 47

3.2. Integration approaches . . . . . 49

3.2.1. Least Squares method . . . . . 49

3.2.2. Fankot-Chellappa method . . . . . 50

3.2.3. $L_1$-min method . . . . . 51
3.2.4. M-Estimators method . . . . . 51
3.2.5. Anisotropic Diffusion method . . . . . 52
3.2.6. Methods evaluation . . . . . 53
3.3. Gradient field of dynamic scenes: exploiting temporal consistency . 56
3.3.1. Constraining scene variation . . . . . 57
3.3.2. Constraining objects movement . . . . . 58
3.3.3. Evaluation . . . . . 61
3.4. Conclusions . . . . . 73

**4. Differential-3D applied to face recognition 75**
4.1. Introduction . . . . . 75
4.2. Description of the proposed framework . . . . . 77
4.2.1. Acquisition and preprocessing . . . . . 78
4.2.2. Nose and eyes localization . . . . . 79
4.2.3. Feature extraction . . . . . 87
4.2.4. Training and classifying . . . . . 91
4.3. Experiments and evaluation . . . . . 92
4.3.1. Evaluation over the Texas 3D Face Recognition database . 92
4.3.2. Evaluation of the entire system . . . . . 94
4.4. Conclusions . . . . . 99

**5. Conclusions 101**
5.1. Future lines of research . . . . . 103

**A. LS approach Euler-Lagrange equation: 105**

**B. Generalized LS and AD approaches Euler-Lagrange equation: 107**

**C. Integration constraining objects' movement E-L equations: 109**

**D. Code and implementation details 111**
D.1. Some 3D scanning techniques . . . . . 111

D.1.1. D3D . . . . . 111
D.1.2. PS . . . . . 119
D.1.3. Takeda's method . . . . . 120
D.1.4. BCS . . . . . 122
D.2. Integration techniques . . . . . 123
D.2.1. LS integration approach . . . . . 123
D.2.2. FC integration approach . . . . . 125
D.2.3. L1 integration approach . . . . . 126
D.2.4. M integration approach . . . . . 128
D.2.5. AD integration approach . . . . . 132
D.2.6. Integration approach constraining scene temporal variation 135
D.2.7. Integration approach constraining objects velocity . . . . . 139
D.3. Nose and eyes detection . . . . . 145

**References** **153**

**Tables index** **166**

**Figures index** **169**

# Chapter 1

# Introduction

Scene analysis from images acquired with digital sensors is the central goal of computer vision. In the past decades, computers and electronic devices have experimented an exponential growth in vision, perceiving and understanding abilities. Robots that perform some housework, smart-phones that can identify who is the person in front of them, social networks that are able to understand and predict some aspects of users personality, and medical devices that can help professionals diagnosis as like never before; are just some of the incredible advances pattern recognition, image processing, applied physics and computer vision communities have achieved.

In this thesis, we will discuss some theoretical and practical advances of interesting topics that are included in many of the previous mentioned fields of knowledge. For example, for three-dimensional scene reconstruction, different approaches can be found depending on the scientific community we are in contact with. For instance, phase shifting techniques are usually developed for scientists that work in optics, while stereo vision is commonly associated to image processing researchers and SLAM (simultaneous localization and mapping) developed by specialists in robotics. In this work we aim to give a broad treatment of different topics, trying to identify those common aspects and highlight the main differences between the approaches followed by different scientific communities. For example, in Chapter 2 we will review different selected methods for three-dimensional scene

retrieval, and propose a new "differential method". In particular we will focus in that chapter on answering the following questions: How can we estimate depth information from two views of the same scene? Which are the main strengths and drawbacks of these methods? What can be done if instead of two cameras we have a camera-projector pair? Which are the differences and similarities between these approaches? Which patterns may be used in structured light projection techniques and which are their differences and properties? After addressing these questions we will introduce a novel three-dimensional scanning technique which allows us to compute in a very fast, simple and efficient manner the depth field's partial derivatives (so we will name this technique D3D, acronym from Differential 3D). We will describe both an analytical and a geometrical derivation of the proposed method which will allow us to address questions as: Which are the main sources of error the proposed technique has? How can we overcome them? If the proposed technique provide us the depth field's gradient map: how can we integrate it to obtain the depth map of the target surface? This last question motivates the work presented in Chapter 3, in which we will explore and study different integration approaches.

Once a gradient field is retrieved, a new set of problems and questions must be treated, for example: Can all gradient fields be integrated? How can we handle noise and outliers in a robust and practical way? Which are the strengths and drawbacks of different integration methods? In Chapter 3 we will address these questions, presenting a representative set of integration approaches and reviewing their performance and properties. We will first introduce the simplest method which follows a Least-Squares approach and minimize the $L_2$-norm error between the empirical gradient field and the one of the final solution. We will show that this solution may be very efficient but it can not handle data with a large level of noise or outliers. In order to consider more robust integration methods we will show that noise and outliers can be handled by either considering an $L_1$ penalty functional, or by introducing weights to guide the $L_2$-norm minimization approach. We will compare these different strategies and study their performance and computational

cost.

After describing how a two-dimensional gradient field can be integrated, we will focus in those problems in which a spatial gradient field along the time can be estimated. In particular we will discuss: How can we exploit the temporal correlations between frames? Instead of integrating *frame by frame* using standard 2D approaches, can we generalize those methods to integrate the entire sequence in a more robust way? These questions are addressed in the final sections of Chapter 3 where different strategies are presented depending on the kind of data we are working with. The first method proposed assumes that data varies continuously (in some temporal order), while the second one exploits Optical Flow ideas to derive a regularization term.

Of course, one may think: what else can be deducted from a gradient map? Are there application in which raw gradient information could be useful? One example of potential application is presented in Chapter 4, where a robust face recognition approach is described. We compute a feature descriptor for faces that contains 3D low level geometric information extracted from the depth's gradient map; to handle expression variations and occlusions (generated e.g. by wearing a scarf or a bonnet) we extract information just from the eyes-forehead and nose regions. Fiducial point are automatically detected following a curvature approach. Once features are extracted, Support Vector Machine (SVM) hyper-planes are obtained for each subject on the database (one vs all approach). One of the main contributions of the proposed technique, is that we bypass the explicit 3D reconstruction of the faces, i.e. the integration step is not required, which is possible thanks to the information provided by the D3D approach. Obviously, face recognition is not the only problem in which differential information would be exploited, other possible applications and potential future work are presented in Chapter 5 along with the main conclusions of this work.

## 1.1. Thesis goal

This thesis explore the problem of retrieving a 3D map combining the theoretical and practical tools developed in the context of Applied Optics and Image Processing fields. The main propositions of this work can be summarized in the following hypothesis: (H1) A depth gradient map of the scene can be obtained by measuring the local bending of horizontal and vertical fringes projected over the test surface. (H2) Given a time dependent depth gradient map, the temporal consistency of the scene can be exploited to regularize the space of solutions to the problem. (H3) Face recognition can be performed in a robust way by extracting feature vectors from faces' depth gradient map, for instance bypassing the explicit three-dimensional reconstruction of faces' surface. In this work we will demonstrate (H1), (H2) and (H3) in the second, third and fourth chapters respectively.

## 1.2. Outline

The rest of this thesis is organized as follows. In Chapter 2 we first review (Sec. 2.1) some popular three-dimensional scanning techniques, and we propose a new method (Sec. 2.2). Then different aspects of the proposed technique are analyzed and compared with other methods.

Chapter 3 firstly describes and discusses the most common *2D* integration approaches, strengths and drawbacks of different methods are highlighted. Then Sec. 3.3 presents some novel alternatives for the integration of gradient fields that vary through time, where the time dependence of the data can be exploited to impose data integrability in a more robust way.

In Chapter 4, a robust framework for face recognition that takes advantage of the novel method introduced in Chapter 2 is presented. In particular we will give a practical solution to the problem of automatically detecting some facial key points as the nose-tip and eyes. The proposed technique generates a feature vector which contains textural and geometrical information extracted from the area around the nose-tip and the regions of the eyes, which have been demonstrated to be the

regions less susceptible to changes in facial expressions. Validation experiments are presented using a publicly available database and newly acquired data.

Finally, Chapter 5 presents the main conclusions and enumerates the contributions made during the course of this PhD.

# Chapter 2

# 3D scene reconstruction

Scene understanding is one of the most challenging and popular problems in the field of computer vision. Robots and computers need to extract three-dimensional information of scenes in order to understand the environment and be able to perform some task that may be trivially performed by humans. Significant efforts are being made, and large amounts of money are being invested to increase machine intelligence and abilities, e.g. cell phones are starting to understand facial expressions, and in the last five years commercial robots have become available that can perform some housework (e.g. iRobot-Roomba). Augmented reality is also another good example of how technology and machine intelligence are merging to our daily lives. In this chapter well established methods for three-dimensional scene reconstruction are reviewed. Section 2.1 introduces a set of popular 3D-shape profiling methods and in Sec. 2.2 a novel technique is described in detail; in addition, performance analyses are presented at the end of the section.

## 2.1. Overview of standard 3D-shape profiling methods

A wide variety of techniques have been proposed for the sake of three-dimensional shape recovery. In a first rough classification those methods can be divided into *Contact* and *Non-contact* methods. Contact based methods generally provide high accuracy results, nevertheless as they involve mechanical movements of a probe de-

vice from one point to another they share some drawbacks; (i) data acquisition can be very time consuming, (ii) they are limited to static and rigid scenes, (iii) they are sensitive to vibrations and temperature variations, and finally (iv) they need to contact the objects. One example of such methods is the Coordinates Measuring Machine (CMMs) [19], it measures the surface of an object using a contact probe; it has a highly sensitive pressure-sensing device that is activated by any contact with an object. The linear distances from three axes to the position of the probe are ascertained, thus giving the $x$, $y$ and $z$ coordinates of the surface.

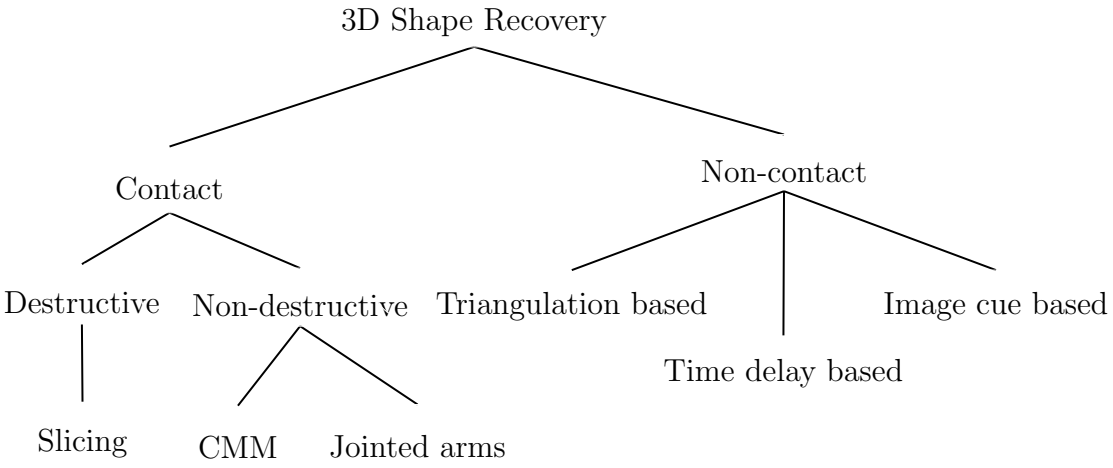


Figure 2.1: Classification of 3D shape recovery methods according to their main characteristics.

Non-contact methods, have become popular in the last decades and they are nowadays the mainly used methods for 3D shape reconstruction. Many commercial devices are being used [64, 82], for example in automobile industry the ATOS scanner of Gesellschaft fur Optische Messtechnik [76]. In this chapter we will focus on non-contact methods and particularly in a subclass of these methods where geometrical information of multiples views is exploited.

### Non contact methods

There are different kinds of non-contact methods, we will particularly focus on optical methods. They can be divided into three categories: *Time delay based*, *Image cue based* and *Triangulation based*. The first class of methods operates using some kind of transreceiver device [69] that emits a signal which is received after being reflected by the test surface; then, the delay or any distortion in the reflected signal is measured and related to the depth of the scene [70]. This is the operating principle of most radar systems, laser and active acoustic devices. The second class of methods, uses some cues on images which under certain hypothesis allow to extract three-dimensional information, examples of such methods are Depth from focus and defocus (DFF/DFD) methods [31, 62, 72]. The idea behind these methods is based on image formation principle, which states that when a lens is used only one particularly plane is brought into focus, while the elements of the scene outside this plane will suffer from blurring. This blur can be estimated and used as cue to extract depth information. These methods usually require several images to be taken while varying the focus settings . Finally, the third class of methods uses -at least- two elements in order to obtain geometrical information of the scene by means of triangulation [76, 82, 100, 101]. Some methods use two or more cameras (*multi-view stereo* [84]), while other methods use one camera and control illumination sources e.g. Structured light [14], Photometric stereo [9], Phase Shifting [7, 8, 100] and Lasers Scans [82]. Methods that control or set illumination conditions are called *active* in contrast with *passive* methods in which just the relative position between camera(s) and objects are considered. Passive methods find matching correspondences between images and depth information is extracted from either the disparity or motion vectors. The main drawbacks of these methods are: the computational cost of matching and feature extraction, and that the depth information can be accurately extracted only for those regions of the scene rich in texture. On the other hand, active approaches can handle textureless surfaces and are -in general- computationally efficient, however these techniques usually require more complex hardware, and they are not suitable in some outdoor scenarios.

Despite the differences that exist between Active and Passive approaches, they are based on the same geometrical principles that we will describe below.

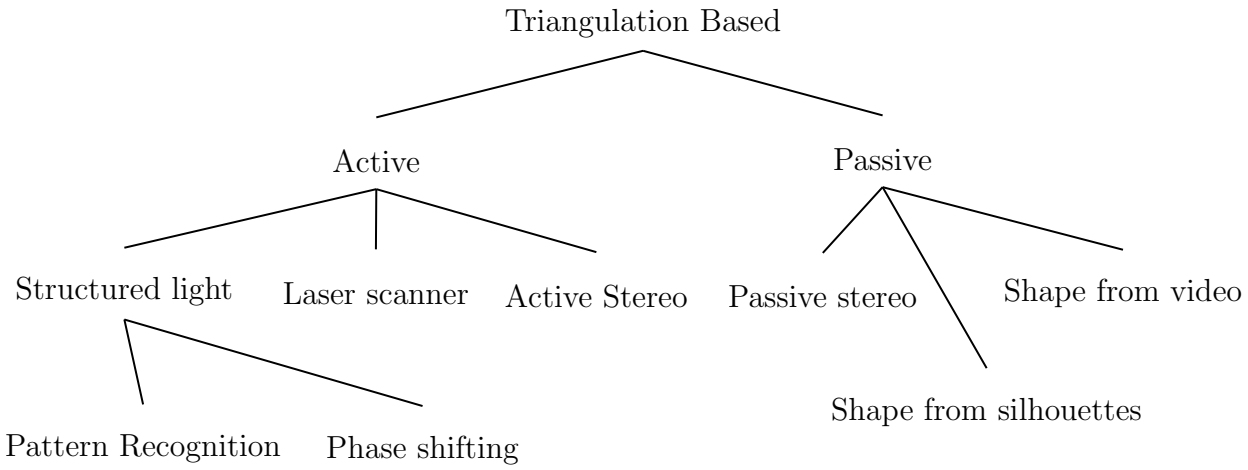


Figure 2.2: Classification of the subset of triangulation based methods.

| | Precision | Inform. | Hardware Requir. | Comp. complexity | Real time app. | Necessity of texture |
|---|---|---|---|---|---|---|
| Contact Methods [19] | High | Sparse | High | Low | No | No |
| Time delay [69, 70] | High | Sparse | Medium | Low | Yes | No |
| Laser Scanner [64, 82] | High | Dense | High | Low | No | No |
| Patterns proj. [14, 76, 82] | Medium | Dense | High | Medium | No | No |
| Phase shifting [7, 8, 100] | Medium | Dense | High | Medium/Low | Yes | No |
| Stereo [47, 84] | Medium | Sparse | Low | High | No | Yes |

Table 2.1: General features of the most popular three-dimensional scanning techniques. The objective of this table is to summarize some of the most common strength and weakness of each family of methods. Of course as it is a general description, some particular implementations in each class may not verify all the characteristics here summarized.

### 2.1.1. Stereo vision principle

When a scene is being captured by two cameras and assuming that just a translation exists between them, the apparent image shift (disparity) gives information about relative depth of the scene, as illustrated in Fig. 2.3. The disparity $d(x, y)$ between both images (i.e., the shift of the images on the detector arrays of the cameras) and the depth $z(x, y)$ of the test surface will be related by,

$$z = \frac{Bf}{d} \tag{2.1}$$

which is obtained from the similarity of the triangles highlighted on Fig. 2.3; where $f$ is the focal length and $B$ is the distance between the cameras' centers.

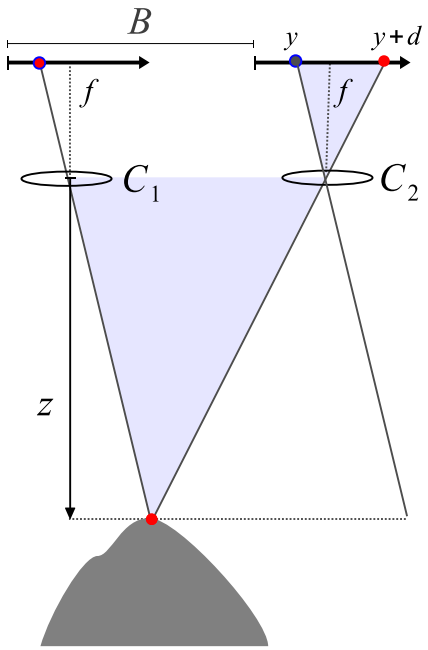


Figure 2.3: Principle of stereoscopic vision. The figure shows two cameras $C_{1,2}$ with lenses of focal distance $f$ and translated a distance $B$.

Figure 2.4 shows an example of images obtained from a stereo pair of cameras. Given a point of the scene if we compute its apparent shift (disparity), we can obtain the relative depth of it with respect to the cameras. For example, assume that the corner of the lamp is in the position $(200, 120)[mm]$ in the *left* (illustrated with the red box) and in the position $(200, 100)[mm]$ (green box) in the right

sensor, then if $B = 30cm$ and $f = 100mm$ we obtain that the point was at a relative depth of $1{,}5m$.

Figure 2.4: Example of a stereo pair of images.

Finding correspondences is the main bottleneck of all passive stereo approaches. In first place because features need to be extracted and matched, which is in general computationally expensive. And in second place, because correspondences cannot be found accurately in case textureless scenes.

### 2.1.2. Structured light methods

Structured light methods, share some similarities with stereo techniques, in these methods instead of two cameras, a pair camera-projector is used. The idea is that we know which is the pattern being projected, so we can find each of the projected pixels in the acquired image; again by measuring the shift (disparity) between the projected and acquired pattern, depth information can be extracted.

Although one can measure directly the shift between the projected and the acquired (deformed) pattern, the calculation of the shift is complex as it requires the precise knowledge of projector's parameters. Instead, the standard approach is to project the pattern over a reference plane and measure the deformation of the pattern (with respect to that obtained for the reference plane). The geometry is similar to that presented for the standard stereo approaches as illustrated in Fig. 2.5.

Assuming $H \gg max\{h, f\}$, the disparity $D(x, y)$ between the pattern project-

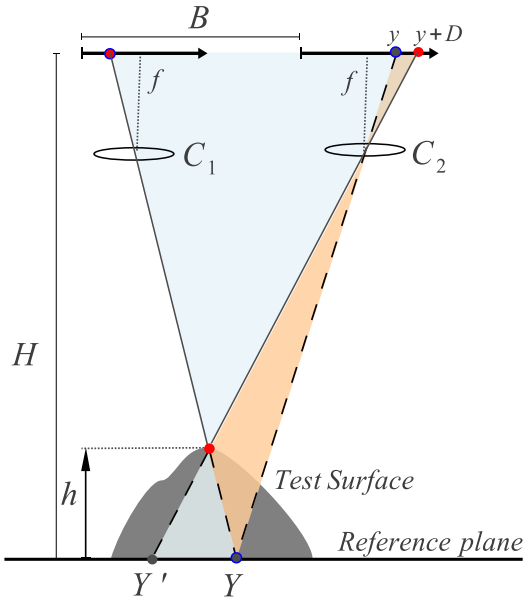


Figure 2.5: Shift between the projection over the reference plane and the projection over the test surface.

ed over the reference plane and the one projected over the test surface, will be related to the depth $h(x, y)$ by,

$$h = \frac{\overline{Y'Y}}{B}(H - f - h) \approx \frac{\overline{Y'Y}}{B/H}\ , \qquad \overline{Y'Y} = \frac{D}{f}(H - f) \approx D\frac{H}{f}. \tag{2.2}$$

The relative errors of the two approximations in Eq.2.2 are $(f + h)/(H - f - h)$ and $f/(H - f)$, respectively.

Hence,

$$h \approx D\frac{H^2}{B\,f} \tag{2.3}$$

where $f$ is the focal length, $H$ is the distance to the reference plane and $B$ is the distance between camera's and projector's center.

Numerous techniques for surface imaging by structured light are currently available. We can classify existing methods into two categories: Single-Shot and Multi-Shot as illustrated in Fig. 2.6 [40]. The first class include those methods that requires the projection of one unique pattern (and hence process only one acquired image) [38, 105]. The second class, uses multiples frames and process multiple acquired images. When the target object is static and the applications do

not impose restriction over the acquisition time, multiple-shots approaches may be used. However, if the target is a moving scene, single shot methods must be applied. These methods has also an important advantage, they do not require any synchronization between the projector and the camera/s which leads to a simpler and cheaper implementation.

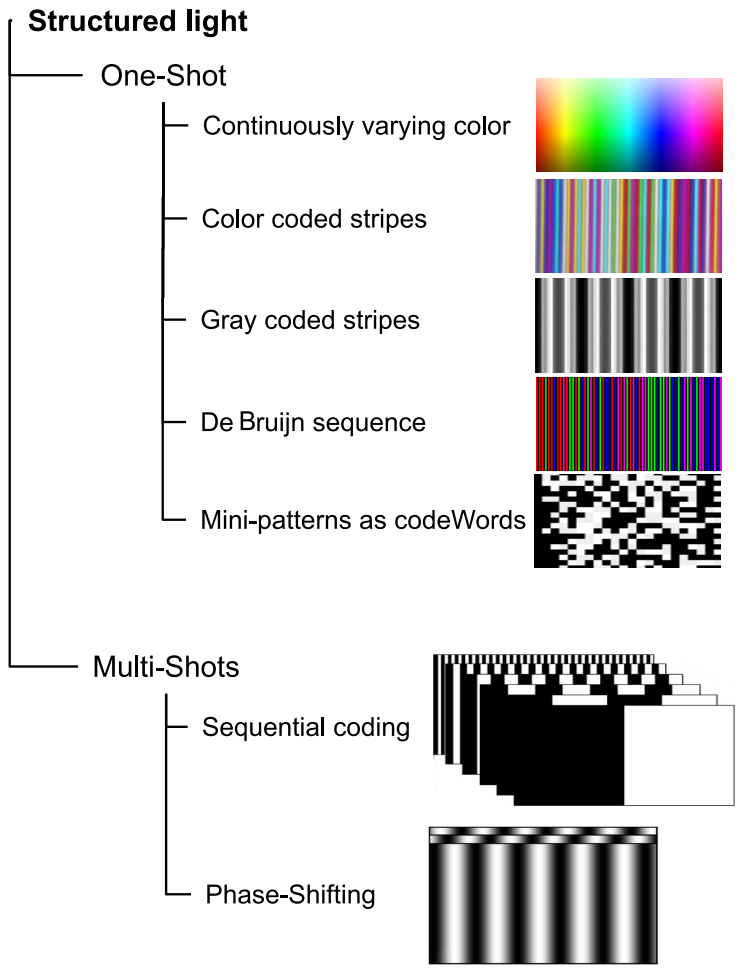


Figure 2.6: Classification of most popular structured light frameworks according to the pattern/s they use.

### Stripe projection and Grid Patterns.

Examples of one-shot 3D scanning approaches are stripe and pattern projection methods. As explained previously, the idea is to measure the shift between the

projected pattern over a reference plane with respect to the projected pattern over the target surface. Imagine we project vertical colored stripes as illustrated in Fig. 2.7; if the camera is translated in the horizontal direction and we project the pattern over a reference plane (as illustrated in Fig. 2.5) we would acquire the image illustrated in Fig. 2.7(a). If we now project the same pattern over a surface with the depth map illustrated in Fig. 2.7(b) we would acquire a deformed pattern shown in Fig. 2.7(c). If we are capable to identify and measure the translation between corresponding points, we can calculate by means of Eq. (2.3) the relative height of the surface.

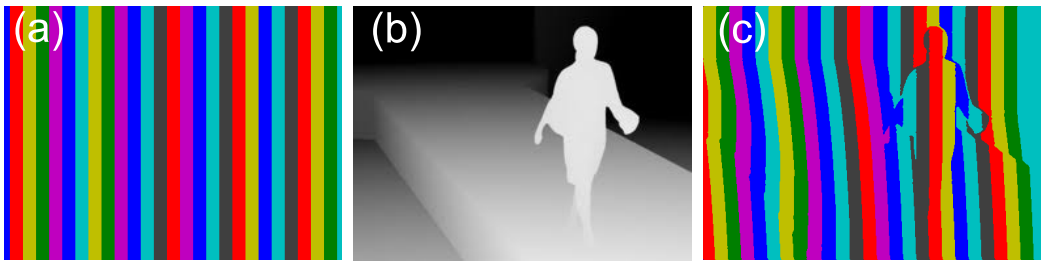


Figure 2.7: Stripe projection example, figure (a) illustrates the projected pattern, (b) illustrates the scene height map (proportional to the gray level), and finally, figure (c) illustrates the image we would obtain when the pattern (a) is projected over the scene (b).

Several patterns allow us to follow the projected points along the deformation caused by the surface. One example is the Rainbow 3D Camera [39, 49] in which a continuous spatially varying wavelength illumination is used, thus providing a one-to-one correspondence on each surface point depending on the wavelength $\lambda$ obtained. Similar approaches uses a finite number of colored stripes instead a continuous rainbow like pattern. In both cases, the resolution is limited by the precision of the camera to separate and distinguish different colors. To increase the robustness and accuracy, many approaches uses stripe indexing methods, in which the order in which the stripes are observed is taken into account. For example, it is possible to arrange the intensity levels of stripes such that any group of stripes (a sliding window of $N$ stripes) represents a unique pattern, as illustrated in Fig. 2.8(a). Other popular approaches [14] uses a De Bruijn sequence [65] to code pseudo random values with specific properties between them. A "k-ary De

Bruijn" sequence of order $n$ is a circular sequence containing each substring of length $k$ exactly once, Figure 2.9 shows an example of De Brujin sequence for $k = 2$ and $n = 2$.

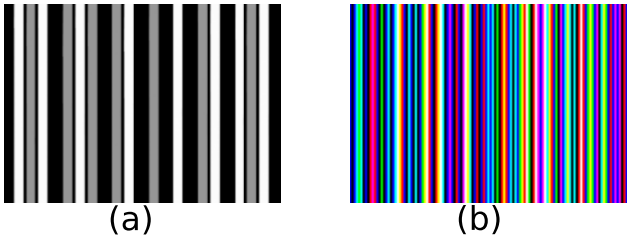


Figure 2.8: Examples of stripes indexing techniques. (a) Gray code (different columns are codded with different gray levels). (b) De Bruijn sequence (a specif sequence of colors is used to code each column).

Another popular strategy consists in projecting 2D grid patterns -as illustrated in Fig. 2.10- which uniquely label every sub-window of the projected pattern. For these methods an appropriate "window property" will determine patterns' robustness against partial occlusions and shadows [43].

In addition to one-shot strategies there are also multi-shot approaches, which allow robust sequential stripe coding. By projecting $N$ different binary patterns one can define $2^N$ different stripes as illustrated in Fig. 2.11. After analyzing the $N$ values for a given pixel, it is possible to uniquely identify to which of the $2^N$ coded stripes it belongs an so, the correspondence problem is trivially solved. Sequential methods use either binary [56, 83, 93] or gray-level [55] patterns. It is

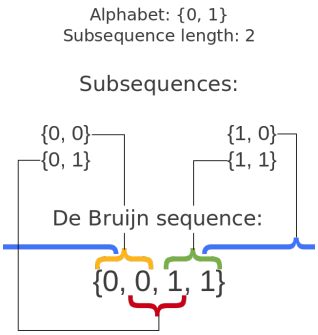


Figure 2.9: Example of De Brujin sequence for $k = 2$ and $n = 2$. Extracted from: [85]

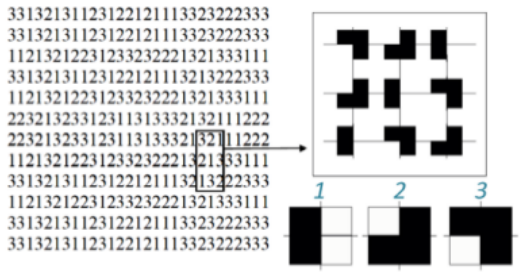


Figure 2.10: Example of 2D grid indexing pattern [43].

clear that the use of more gray levels reduce the number of projected patterns (maintaining the number of words in the alphabet), but they are less accurate when the test surface presents variable reflectivity. Binary approaches are very useful and simple methods, since in contrast to color or gray coded approaches, no complicated calibration steps are involved; the price to pay is that more patterns must be projected to obtain the same amount of coded information, which may not be possible if the target is a moving object or we have acquisition time constraints.

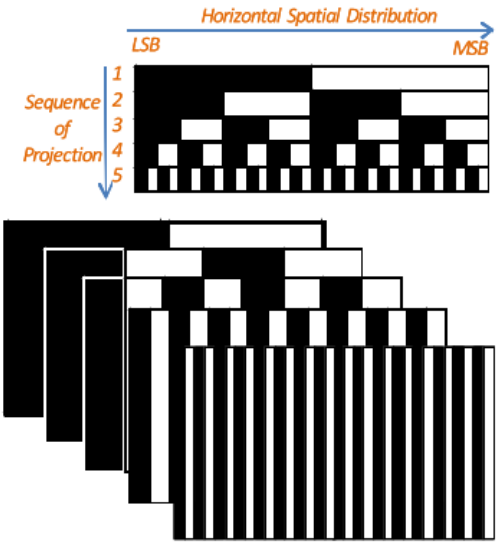


Figure 2.11: Example of a sequential binary-coded pattern; each image represents one bit of the binary codification of the pattern. Figure extracted from [40].

Phase Shifting.

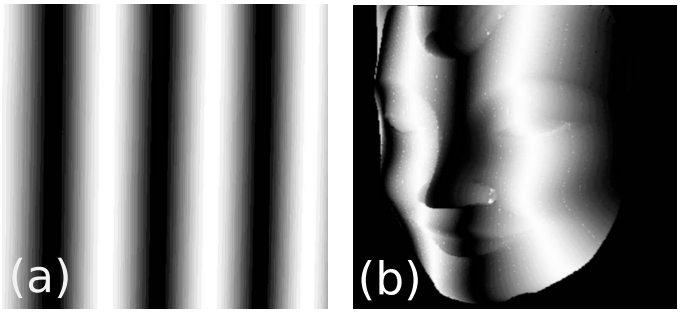


Figure 2.12: (a) Illustration of a sinusoidal pattern projected over a reference plane, and (b) the same pattern projected over a test surface.

A particular class of multi-shot methods are the so-called Phase Shifting (PS) approaches. They relate the height of a test surface with the phase of sinusoids projected over it. Imagine that a sinusoidal pattern (Fig. 2.12(a)) is projected over a test surface as illustrated in Figure 2.12(b). The acquired image will be of the form [67]:

$$I(x,y) = A(x,y)\left[1 + B(x,y)\cos\left(2\pi f_0 x + \phi(x,y)\right)\right], \tag{2.4}$$

where $A(x,y)$ and $B(x,y)$ are unknown functions that depend on surface's reflectance and ambient illumination, $f_0$ is the spatial frequency of the projected pattern and $\phi(x,y)$ is the phase shift proportional to surface height. If additional sinusoidal patterns are projected (of the same frequency but different phase) we can eliminate the unknown functions A and B, and obtain the unknown phase $\phi$. One of the simplest techniques is the three steps PS algorithm [67] in which three sinusoidal pattens with relative phase shift of $2\pi/3$ are projected. The three acquired images (one for each projected pattern) will be:

$$\begin{cases} I_1(x,y) = A(x,y)\left[1 + B(x,y)\cos\left(2\pi f_0 x + \phi(x,y) - 2\pi/3\right)\right] \\ I_2(x,y) = A(x,y)\left[1 + B(x,y)\cos\left(2\pi f_0 x + \phi(x,y)\right)\right] \\ I_3(x,y) = A(x,y)\left[1 + B(x,y)\cos\left(2\pi f_0 x + \phi(x,y) + 2\pi/3\right)\right] \end{cases} \tag{2.5}$$

from which we can retrieve the unknown phase as:

$$\phi(x,y) = \tan^{-1}\left\{\sqrt{3}\frac{I_1 - I_3}{2I_2 - I_1 - I_3}\right\} - 2\pi f_0 x. \tag{2.6}$$

Phase shifting approaches are very powerful frameworks for three-dimensional scanning as they can obtain precise and dense reconstructions with very low computational complexity. Nevertheless PS methods share some drawback which may be accounted in order to obtain accurate results. Firstly, as the phase is recovered after a $\tan^{-1}$ operation, the obtained phase is actually a *wrapped* version (to the $(-\pi, \pi]$ interval) of the true phase; this problem can be solved by the well known *unwrapping techniques* [41] (we will briefly describe this problem later). Secondly, the projected pattern must be sinusoidal or large errors are introduced; this is a very important limitation as projector devices has a nonlinear response in the light intensity (which is often called the "gamma problem"). To get a strictly sinusoidal pattern the nonlinearity must be overcome. A wide variety of techniques have been proposed with this purpose, e.g. the projection of pre-distorted patterns [54], the creation of a lookup table for error compensation [102], and other techniques [50,53]. However, it has been found [60] that the response in intensity of projectors actually changes over time, and thus needs frequent recalibration, this partially hinders the application of calibration and compensation schemes. There are also frameworks that use binary patterns to elude the nonlinearity problem. Some of them, to achieve a sinusoidal pattern, require blurring to a certain degree by means of either defocusing the projector [7, 60] or digitally smoothing the acquired patterns [81] which has the disadvantage of lowering the image contrast. Other methods use multiple binary frames and code the sinusoidal patterns [8].

### Two-dimensional phase unwrapping.

Two dimensional phase unwrapping is necessary in interferometric applications and in all methods where the phase of a complex or sinusoidal signal must be recovered. Unwrapping techniques have received lot of attention because they are the core technology that enables radar interferometry, a very popular technique used for precise measurement of surface topography. Also it is crucial in applications such as acoustic imaging, adaptive optics, magnetic resonance imaging (MRI) and phase-shifting 3D scanning techniques.

When we want to retrieve the phase $\varphi$ of a given complex or sinusoidal signal, we can only obtain through mathematical operations values that lie in the interval $(-\pi, \pi]$. These principal values are also called the *wrapped* phase $\psi$ because the actual phase (whatever it happens to be) is wrapped to the interval $(-\pi, \pi]$. Mathematically the wrapping is a non linear operation that can be defined as

$$\psi(u) = \varphi(u) + 2\pi k(u)$$

where $u$ is the variable (depending on the application it can be the time or a spatial coordinate), $k$ and integer function that forces $-\pi < \psi \leq \pi$. Thus $\psi$ is a nonlinear function of $\varphi$.

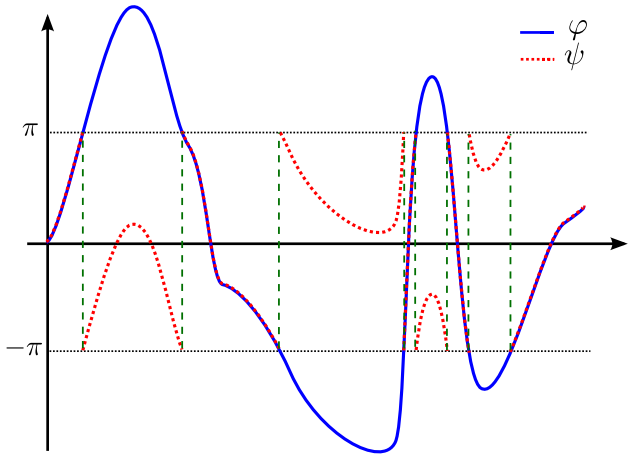


Figure 2.13: Example of a wrapped phase; the plot illustrate the original phase $\varphi$ (solid blue line) and the wrapped phase $\psi$ (dotted red line)

If we define $\mathcal{W}$ as a wrapping operator (e.g. $\mathcal{W}[\varphi] = \tan^{-1}(\tan(\varphi))$), one can relate wrapped and real phase gradient by:

$$\nabla\psi = \nabla\varphi + \nabla(2\pi k), \tag{2.7}$$

assuming that the real phase is smooth, i.e. it satisfy $\nabla\varphi \in (-\pi, \pi]$, we can apply the wrapping operator both sides of Eq. 2.7,

$$\mathcal{W}[\nabla\psi] = \mathcal{W}[\nabla\varphi + \overbrace{\nabla(2\pi k)}^{\propto 2\pi}] = \nabla\varphi. \tag{2.8}$$

Hence we can obtain the unwrapped phase, by integrating the wrapped gradient of the wrapped phase, i.e.:

$$\varphi(\mathbf{r}) = \int_{\mathbf{C}} \mathcal{W}[\nabla\psi] \cdot \mathbf{dr} + \varphi(\mathbf{r_0}). \tag{2.9}$$

Equation 2.9 shows that the unwrapping problem can be *reduced* to a integration problem. The bottleneck of unwrapping approaches, is that due to noise, aliasing, or singularities the phase defined by Eq. 2.9 may depend on the integration path which greatly complicate the problem solution and uniqueness. The theoretical requirements for the unwrapped phase to be independent of the integration path is that the gradient field defined by $\mathcal{W}[\nabla\psi]$ be irrotational on every point of the domain, which implies that the integration of the field on any closed curve is zero. Those points of the domain in which the irrotational conditions does not hold are call *residual* points.

Unwrapping techniques can be classify as [41]: (i) Path Following Methods, which are techniques that try to integrate along paths that elude those residual points ( [41, Chapter 4]). (ii) Minimum-Norm Methods, which solves the non-integrability problem by projecting the data term into a suitable subspace (by minimizing an adequate norm). This problems include Weighted and Unweighted Least-Squares approaches and $L^p$ Norm methods ( [41, Chapter 5]). In [41, Chapter 6] a complete review and comparison of the most important unwrapping methods is presented; in that reference also state of the art implementations of popular algorithms are provided, some of them are used along this thesis.

## 2.2. Gradient approach

In this section we are presenting a new method for 3D surface retrieval that can be identified as a single shot, structured light approach. In contrast with most single shot techniques [40] (e.g. Color Coded Stripes, Gray Scale Coded Stripes, De Brujin Sequence, Pseudo Random Binary Dots, Mini-Patterns and Color Coded Grids), the proposed method does not require finding correspondences or matching steps.

We are presenting a differential three-dimensional (3-D) shape profiling method that is based on the combination of orthogonal fringes projection. It allows us to compute depth gradient maps in a fast and efficient manner. The procedure for 3D-shape retrieval we are proposing consists in measuring the partial derivatives of the disparity $D$ (Fig. 2.5), and then, to integrate them with the purpose of retrieving $h(x, y)$ using Eq. (2.3).

Let us assume that we are projecting a rectangular fringe pattern of period $p$ in the $x$ and $y$ directions over a test surface, where $I(x, y)$ is the pattern acquired by the camera (see Fig.2.15(a)), and $\theta$ the angle of the direction in which the projector was translated with respect to the camera (as illustrated in Fig. 2.14). Different strategies can be followed to combine vertical and horizontal fringes in a unique pattern. One can for example, code vertical fringes in the blue channel of the projector and horizontal fringes in the red one. Of course this would limit us to work with a color pair camera-projector. Other option is to combine both fringes in a unique -gray- pattern by summing or multiplying them [35]. In that case, if $D(x, y)$ has a narrow spectral band i.e. $D_{x,y} \ll 1$ (where $D_x$ and $D_y$ denote the partial derivative with respect to $x$ and $y$ respectively) one can recover vertical and horizontal components by filtering in the Fourier domain (as illustrated in Fig. 2.15).

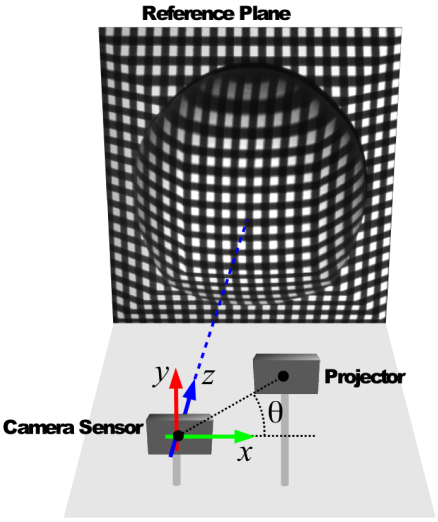


Figure 2.14: Illustration of the acquisition setup.

Therefore, it does not matter if the projected fringes are binary, sinusoidal or are codded in different colors; without loss of generality, by filtering the input image $I(x, y)$, one can obtain

$$I_h(x, y) = I_0(x, y) \cos\left((2\pi/p)(y - D(x, y) \sin(\theta))\right) \tag{2.10}$$

and

$$I_v(x,y) = I_0(x,y)\cos\left((2\pi/p)(x - D(x,y)\cos(\theta))\right) \tag{2.11}$$

where $I_0(x,y)$ is a function of the reflectance of the test surface. As usual, we are assuming[1] that $I_0(x,y)$ is a low-frequency function in comparison with the frequency of the spatial carrier, i.e. $\frac{\partial I_0}{\partial i} \ll 2\pi/p$, where $i = x, y$.

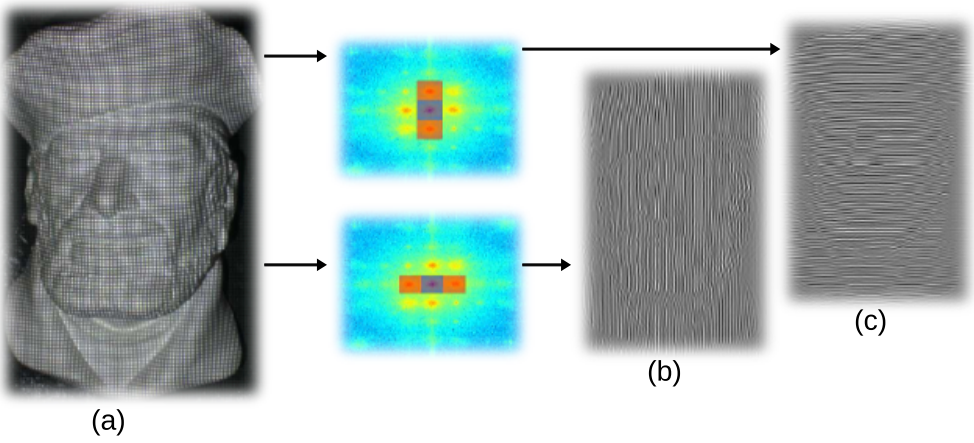


Figure 2.15: (a) Acquired image $I(x,y)$. The colored figures show the 2D-spatial spectrum of the image. By filtering the horizontal and vertical Fourier components (red in the figures), one obtains the images $I_v(x,y)$ and $I_h(x,y)$ shown in (b) and (c), respectively.

Then, by taking partial derivatives with respect to the $x$ and $y$ coordinates, from Eq. (2.10) and Eq. (2.11) one obtains

$$I_{hi}(x,y) \approx -(2\pi/p)I_0(x,y)\,\sin\left[(2\pi/p)\,(y - D(x,y)\sin(\theta))\right](y_i - D_i(x,y)\sin(\theta)) \tag{2.12}$$

$$I_{vi}(x,y) \approx -(2\pi/p)I_0(x,y)\,\sin\left[(2\pi/p)\,(x - D(x,y)\cos(\theta))\right](x_i - D_i(x,y)\cos(\theta)) \tag{2.13}$$

where $x_i = 1$ and $y_i = 0$ for $i = x$, and $x_i = 0$, $y_i = 1$ for $i = y$.

Hence, it is easy to demonstrate that

$$\frac{-D_x\sin(\theta)}{1 - D_y\sin(\theta)} = \frac{I_{hx}(x,y)}{I_{hy}(x,y)} \tag{2.14}$$

[1] If this hypothesis does not hold, we can remove it if we acquire an additional picture of the scene just with the ambient illumination and use it for normalization.

and

$$\frac{-D_y \cos(\theta)}{1 - D_x \cos(\theta)} = \frac{I_{vy}(x,y)}{I_{vx}(x,y)}. \tag{2.15}$$

Finally, one obtain:

$$D_x = \frac{\frac{I_{hx}}{I_{hy}}\left(1 + \frac{I_{vy}}{I_{vx}}\tan(\theta)\right)}{\sin(\theta)\left(\frac{I_{hx}}{I_{hy}}\frac{I_{vy}}{I_{vx}} - 1\right)} \quad \text{and} \quad D_y = \frac{\frac{I_{vy}}{I_{vx}}\left(1 + \frac{I_{hx}}{I_{hy}}\frac{1}{\tan(\theta)}\right)}{\cos(\theta)\left(\frac{I_{hx}}{I_{hy}}\frac{I_{vy}}{I_{vx}} - 1\right)}. \tag{2.16}$$

We conclude that the gradient of the disparity ($D$) can be calculated in a simple manner as a function of the derivatives of the (horizontal and vertical) Fourier components of the image $I(x,y)$ acquired by the camera.

After obtaining the partial derivatives $\partial D/\partial x$ and $\partial D/\partial y$, we must integrate them to obtain $D(x,y)$ ($\approx h\frac{Bf}{H^2}$). In the next chapter, we will discuss the integration problem in detail and address practical solutions to it.

### 2.2.1. Geometrical interpretation and method overview

Complementing the analytical description performed previously, we will address now a geometrical interpretation that can be carried out for the presented approach. This intuitive derivation of the method helps the understanding of the proposed approach, but also it allows us to detect some aspects we may take into account when implementing the method.

Let us consider a synthetic example; Fig. 2.16(a) shows a test surface with height $z(x,y)$ that contains a ramp and Gaussian peaks. If we project the patterns shown in Fig. 2.16(b-d) over the test surface, we would obtain deformed patterns as simulated in Fig. 2.16(e-g). In this set of simulations, we supposed that the projector was translated the same distance along $x$ and $y$ coordinates and the constant $K = Bf/H^2$ was arbitrary set $K = 1/5$.

The patterns shown in Fig. 2.16(b-d) illustrates different ways of combining horizontal and vertical fringes in a single pattern. In the first case (Fig. 2.16(b)) vertical and horizontal fringes are multiplied, in the second case (Fig. 2.16(c)) fringes are summed [35], and finally, in the last examples (Fig. 2.16(d)) fringes are coded in the blue and red channels. As we can see, these patterns are deformed

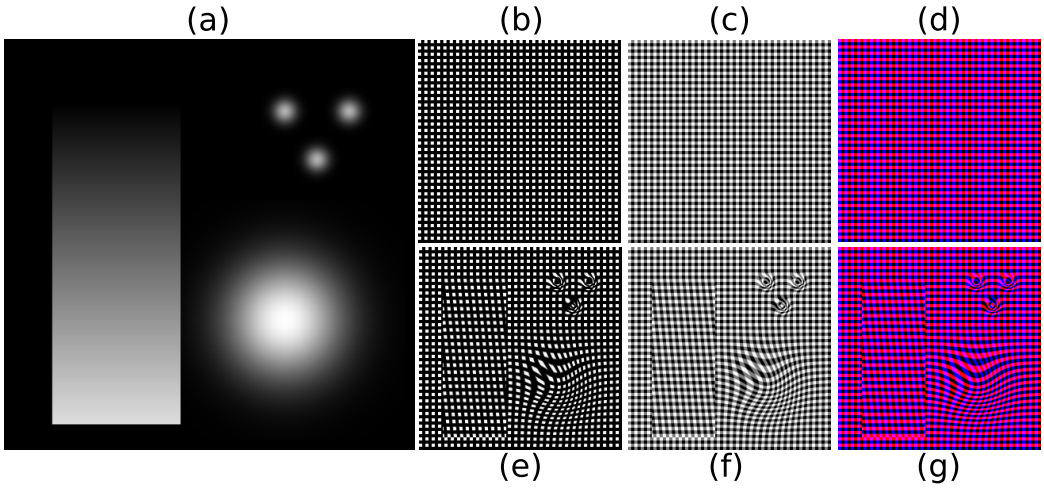


Figure 2.16: Example of simulated data. (a) illustrates the height map of the test surface (where the gray level is proportional to surface's height), (b-d) show different patterns projected over the reference plane, (e-g) obtained (deformed patterns) when patterns are now projected over the test surface (a).

.

when they are projected over a surface according to its height. For instance let us assume that we are analyzing just the deformation of the horizontal fringes. As we shown before, each pixel in the original pattern will be shifted a distance proportional to surface height. Figure 2.17 illustrates the shift of a point $(x_1, y_1)$ compared to its neighbor $(x_2, y_2)$.

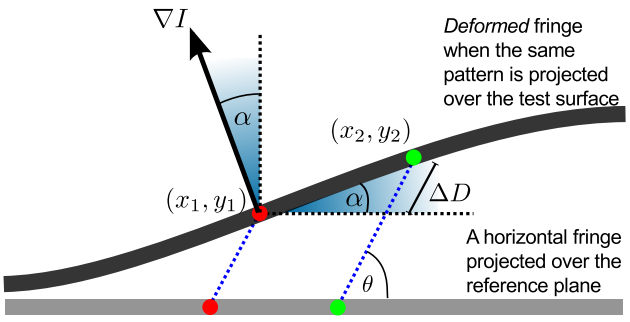


Figure 2.17: Illustration of the local bending of projected fringes due to surface's depth variation.

.

We can relate the image gradient and the disparity partial derivative by:

$$\frac{-I_{hx}}{I_{hy}} = \tan(\alpha) = \frac{(D(x_2, y_2) - D(x_1, y_1))\sin(\theta)}{x_2 - x_1}. \tag{2.17}$$

If we define the vector $\vec{u} = (x_2 - x_1, y_2 - y_1)$ the previous expression can be rewritten as (where we divide and multiplied by $|\vec{u}|$):

$$\frac{-I_{hx}}{I_{hy}} = \tan(\alpha) = \left.\frac{D(\vec{x} + \vec{u}) - D(\vec{x})}{|\vec{u}|}\right|_{\vec{x}=(x_1,y_1)} \frac{|\vec{u}|}{x_2 - x_1}\sin(\theta) \tag{2.18}$$

By taking the limit when $|\vec{u}| \to 0$, we obtain:

$$\frac{-I_{hx}}{I_{hy}} = \tan(\alpha) = \frac{\partial D(x_1, y_1)}{\partial u}\frac{1}{\cos(\alpha)}\sin(\theta) \tag{2.19}$$

$$\Rightarrow \tan(\alpha) = \frac{-I_{hx}}{I_{hy}} = (D_x + D_y \tan(\alpha))\sin(\theta) \tag{2.20}$$

$$\Rightarrow \quad \tan(\alpha) = \frac{I_{hx}}{I_{hy}} = \frac{-D_x \sin(\theta)}{1 - D_y \sin(\theta)}. \tag{2.21}$$

Notice that the expressions matches with those obtained in Eq. 2.14. Analogously we can relate vertical fringes deformation with $D$ partial derivatives recovering.

$$\frac{I_{vy}}{I_{vx}} = \frac{-D_y \cos(\theta)}{1 - D_x \cos(\theta)}. \tag{2.22}$$

Expressions 2.21 and 2.22 allow us to directly relate the variations on surface depth to the local bending of the fringes. This geometrical understanding of the method is also very useful to deeply understand some implementation details and some cares we must have.

Figure 2.18 shows (a) the simulated image with horizontal fringes, (b) a zoomed region of the image, where overlapped we display in red the vectors $(D_x \sin(\theta), 1 - D_y \sin(\theta))$ (as illustrated in Fig. 2.17) and in blue the gradient of the image $\nabla I$. Finally (c) Shows the angular absolute error (measured in degrees) between the angle $\alpha$ obtained from image gradients and the value obtained from surface depth variations. As we can see, as the depth partial derivatives are computed from the gradient's direction, large errors may occur in those areas with very weak variations (illustrated with green dotted ellipses in Fig. 2.18(b)). In such areas, as the gradient is very weak its direction becomes noisy, and the estimation of

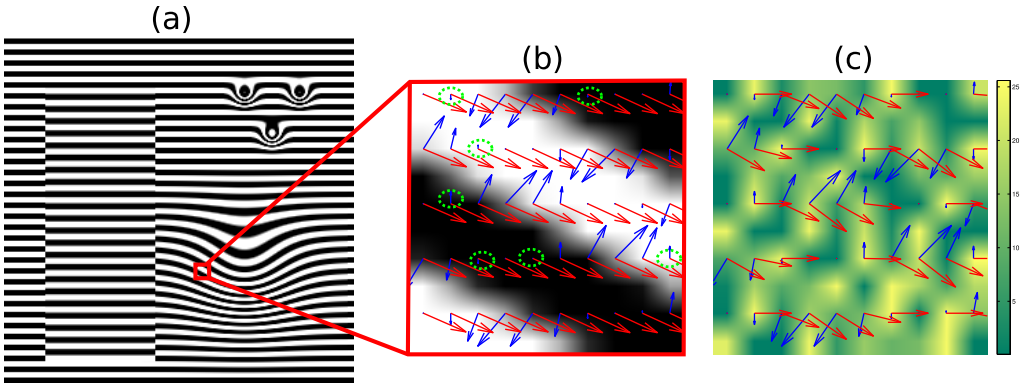


Figure 2.18: (a) Shows the simulated image with horizontal fringes (obtained from the red channel of the color image illustrated in Fig. 2.16(g)). (b) Shows one zoomed region of the image, where overlapped we display in red the vectors $(D_x \sin(\theta), 1 - D_y \sin(\theta))$ (as illustrated in Fig. 2.17) and in blue the gradient of the image $\nabla I$. (c) Shows the angular absolute error (measured in degrees) between the angle $\alpha$ obtained from image gradients and the value obtained from surface depth variations.

surface depth variations inaccurate. To solve this problem we may sacrifice some resolution and smooth fringes transitions, by this, we can increase the number of points that provide a reliable gradient estimation. Figure 2.19 shows the smoothed input image, its gradients and the new errors.

The second source of error in gradient estimation are discontinuities of the test surface, as illustrated in the first example on Fig. 2.20. When there is a discontinuity in the surface, the projected pattern suffers a discontinuous deformation in which image gradients are not any more orthogonal to depth partial derivatives (which are not even defined in such points). This kind of errors may be also present in continuous but very heavy variations of the test surface, as illustrated in the second example shown in Fig. 2.20. There, a narrow peak produces the merge of two different fringes causing errors in gradient estimation.

Finally, one can identify more common sources of error which are: changes in fringes shape due to a variable surface reflectance, shadows, other sources of illumination that may be present in the scene or noise introduced by the camera. Below we summarize the different possible error sources and we give some insight of how overcome or mitigate them.

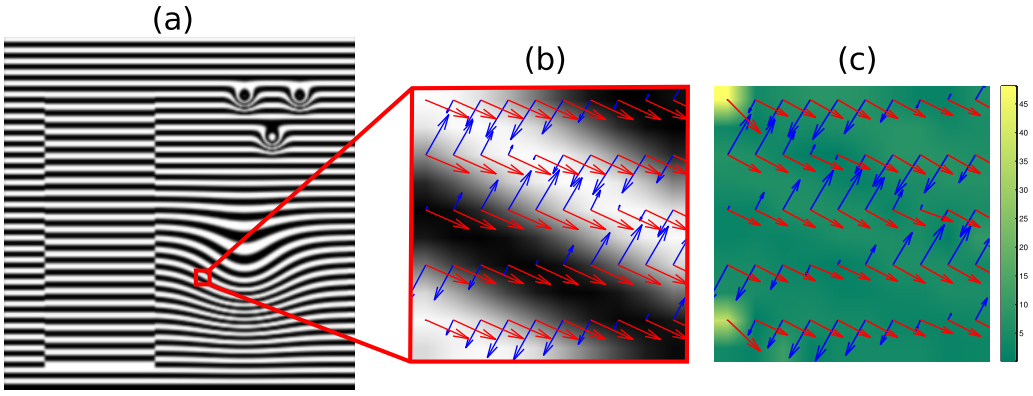


Figure 2.19: (a) Smooth version of 2.18(a) obtained applying a convolution between the binary fringes and a Gaussian kernel with a standard deviation $\sigma = 0{,}8px$. (b) Shows one zoomed region of the image, were overlapped we display in red the vectors $(D_x \sin(\theta), 1 - D_y \sin(\theta))$ (as illustrated in Fig. 2.17) and in blue the gradient of the image $\nabla I$. (c) Shows the angular absolute error (measured in degrees) between the value of the angle $\alpha$ obtained from image gradients and one obtained from surface depth variations).

Summary of the possible error sources and possible solutions we can take to minimize them:

- Low gradients in the interior regions of the fringes.

  *We can slightly smooth acquired images, in addition we can also use the module of image's gradient to estimate a quality map for the estimated depth gradients. Another interesting idea*[2] *is to dynamically change fringes position or period and acquire more patterns. By means of this we can obtain redundant estimations of the gradient map, with the advantage that the location of corrupt pixels change from one frame to the other. Hence by the combination of this set of redundant measurements it could be possible to remove those noisy points. Of course if we project different patterns the method will no longer be* one-shot *and other aspects and limitations (e.g. the need of synchronization and static scenes) will have to be faced.*

- Surface discontinuities or extreme height variations.

  *This is the most challenging source of error. If we can estimate the position of*

[2]thanks to Rafael Grompone for the fruitful discussions and suggestions.

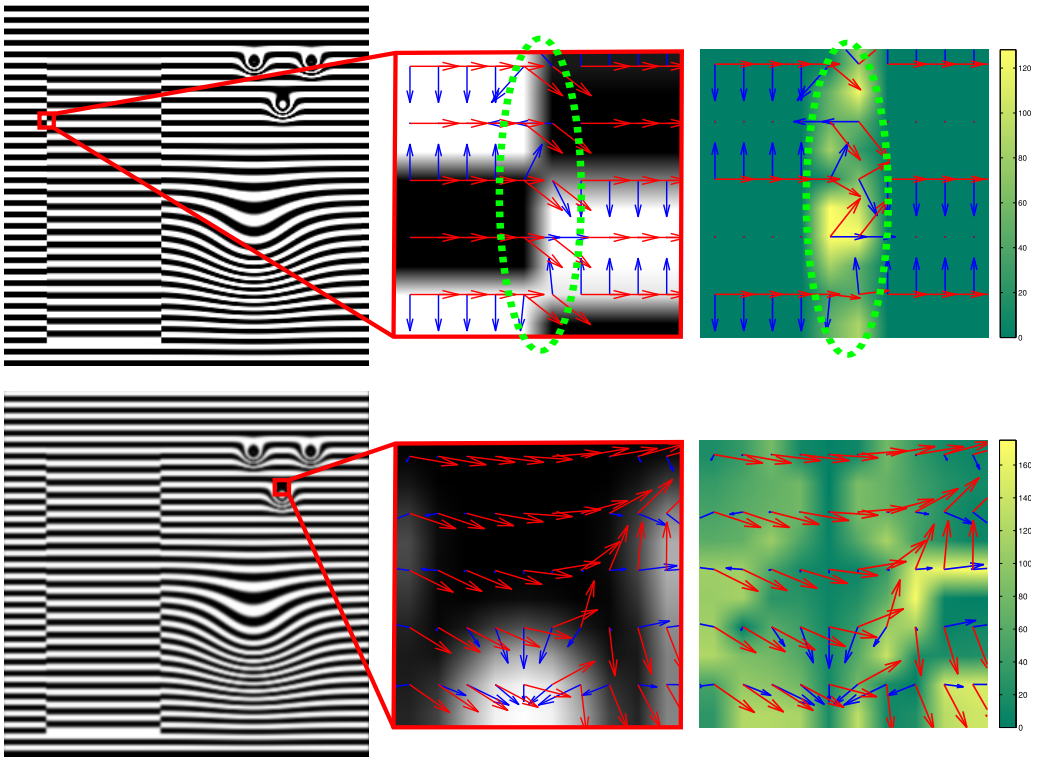

Figure 2.20: As in Figs. 2.18 and 2.19 we illustrate the simulated image with horizontal fringes, image gradients compared with depth level sets orientation, and finally the angular errors. We illustrate in this particular image two particular areas where errors may be expected, in the first row we zoomed an area where a discontinuity in the depth map occurs. And in the second row we zoomed a region where we have a narrow peak.

*these points, we could integrate the gradient field through the smooth regions avoiding discontinuities. In the next section we will propose a simple way to estimate a discontinuity map which used as binary weights allows us to retrieve discontinuities following a simple weighted least squares approach. Furthermore in the following chapter, other techniques for the estimation of quality maps will be introduced.*

- Surface reflectance and ambient illumination.

  *If surface reflectance varies slowly (compared with fringes variations) this should not be a problem; if not, one can take an additional image (with just the ambient illumination) and compensate surface reflectance and ambient*

*illumination. In the next section (pag. 41) we will explore the impact of uncontrolled ambient illumination and we will estimate the minimum intensity of projected light that is required.*

- Projector or camera deformation.
  *We are assuming that the projector and the camera do not introduce significant geometrical deformation. If they do, we can calibrate [53, 104] both camera and projector and then process the rectified images.*

### 2.2.2. Evaluation and comparison with other methods.

In this section we will experimentally analyze the main features of the proposed technique (which we will call from now on *D3D* acronym for Differential 3D). In particular we will address the following questions: How can we handle the different sources of error described in previous section? When we want to retrieve test surface from its gradient: Which is the impact of the integration step? In addition, we will compare the accuracy of the proposed technique with some popular methods for 3D shape retrieval, e.g. the projection of binary coded stripes (from now on *BCS*), Takeda's approach [91] (from now on *Tak*) and a Three-steps Phase Shifting approach (from now on *PS*). After this we will explore the performance of the proposed technique, when we acquire under uncontrolled ambient illumination while varying the intensity of the projected pattern; furthermore the impact of fringes width will be evaluated. Finally we will test the proposed method scanning a person face while it is moving which will show the potential of the proposed technique to analyze dynamic and complex scenes.

#### Proposed method analysis

As was mention in the previous sub-section, due to empirical errors and surface discontinuities one obtain after measuring fringes deformation a noisy -in general non integrable[3]- gradient field. In order to retrieve the target surface, we must

[3] i.e. the irrotational constraint does not hold

either follow a robust integration approach (see e.g. Chapter 3) or pre-process the input gradient field in order to transform it into an integrable one. As we analyze in subsection 2.2.1, the errors in the input gradient field are essentially caused by: (a) low gradients in the areas of local maximum and minimum of fringes, and (b) surface discontinuities. Figure 2.21 illustrates in red and green pixels where

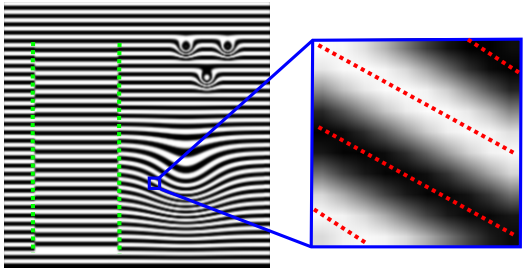

Figure 2.21: Result of projecting horizontal fringes over the test surface whose depth map is illustrated in 2.16(a). In this example, we highlight in green and red those points where we expect to have poor gradient estimations due to: surface discontinuities and fringes' local extremum respectively.

we expect to have errors in gradient field measurement because of (a) and (b) respectively.

In a first experiment, we will explore the impact of these two sources of errors; to that end, we will analyze again the Ramp-Peaks example explored in the previous section. Figure 2.22-(a) shows the ground truth surface, sub-figures (b) and (c) show the true $x$ and $y$ partial derivatives of the test surface respectively. Sub-figures (d) and (e) show the $x$ and $y$ estimated partial derivatives by following the proposed approach. As we can see, sparse errors (due to low gradients) and errors due to surface discontinuities are present in the retrieved gradient field. The first kind of errors, do not represent a real problem as they are too sparse, hence we found that a nonlinear median-like filter is enough to remove them and achieve accurate reconstructions. On the other hand, the errors due to surface discontinuities do represent a more challenging problem as they lead to large and connected regions of outliers. Because of that, even robust integration approaches tend to miss surface discontinuities; see e.g. the results illustrated in Figure 2.22

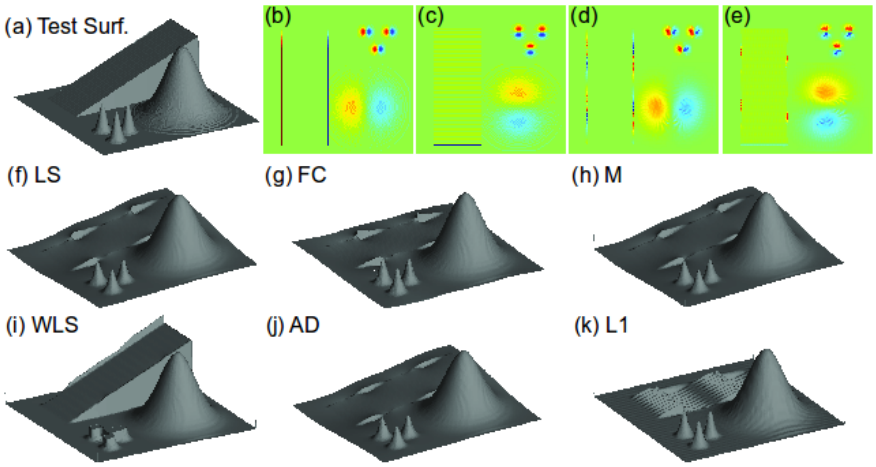


Figure 2.22: (a) Original test surface. (b) Ground truth x-partial derivative. (c) Ground truth y-partial derivative. (d) Estimated x-partial derivative following D3D method. (e) Estimated y-partial derivative following D3D method. (f) Surface obtained by integrating the gradient field illustrated in (c-d) following Least Squares (LS) approach (3.2.1). (g) Surface obtained following Fankot-Chellappa (FC) integration approach (3.2.2). (h) Surface obtained following M-estimator (M) integration approach (3.2.4). (i) Surface obtained following LS integration approach including binary weight. (j) Surface obtained following Anisotropic Diffusion (AD) integration approach (3.2.5). (k) Surface obtained following $L_1$-norm minimization (L1) integration approach (3.2.3)

where we show the surfaces obtained by integrating[4] with: (f) Least Squares (LS) approach (3.2.1), (g) Fankot-Chellappa (FC) (3.2.2), (h) M-estimator (M) integration approach (3.2.4), (j) surface obtained following Anisotropic Diffusion (AD) integration (3.2.5) and (k) L1-norm minimization (L1). To handle errors due to surface discontinuities we propose to detect these points, and assign zero weight to them during the integration process. For example, in the present experiment we estimate a *discontinuities map* by comparing on the estimated gradient field each pixel with its neighborhood. If the difference was above certain threshold we set the integration weight to zero. In addition we found that the discontinuity map can be improved if we apply dilatation and erosion morphological operations (as dis-

[4]In this chapter we briefly describe and use different gradient field integration techniques, leaving to the following chapter a detailed and formal analysis of this problem.

continuities areas tend to be connected segments of several pixels); implementation details are presented in App. D.1. We then use these weights to integrate the input gradient maps following a weighted Least Squares approach (we will refer to this approach as WLS). WLS can be seen as modification of AD approach (Sec. 3.2.5, Eq. 3.13), where the weight map is computed taking into account the specific properties of the noise present in D3D gradient fields. For instance, we have to handle a non-sparse and non-linear impulsive kind of noise, which is distributed along the discontinuity regions, therefore integration approaches that assume additive Gaussian noise plus sparse impulsive noise are not suitable for the reconstruction of scenes with height discontinuities.

Figure 2.22-(i) shows the result obtained following the previous described approach. As we can see, if we consider the proposed weights we can improve the reconstruction precision specially for surfaces that have large discontinuities. Table 2.2 summarizes some numerical results obtained from the previously described approaches.

| Method | LS | FC | M | WLS | AD | L1 |
|---|---|---|---|---|---|---|
| RMSE (%) | 19,1 | 19,8 | 19,1 | **4,3** | 19,0 | 20,8 |
| Rel. Time | 2,3 | **1,0** | 16,5 | 2,6 | 65,8 | 50,8 |

Table 2.2: Comparison of integration techniques for the Ramp-Peaks experiment. The root mean squared error is display as a percentage of surface amplitude, and execution time of the algorithms are normalized with respect to the fastest one.

We can now move a step forward and analyze in a second series of experiments more complex and realistic data such as human faces. To that end we considered a set of examples of the Texas 3D Face Recognition Database [44,45], some examples of faces' depth obtained with a commercial scanner can be seen in Figure 2.23, in the second row are shown examples of the simulated data that we would obtain if we project the proposed pattern over each test surface. Figure 2.24 illustrates the relative root mean square error for different preprocessing filters over thirty human faces. As we can see, a median like filter improves the accuracy on surface

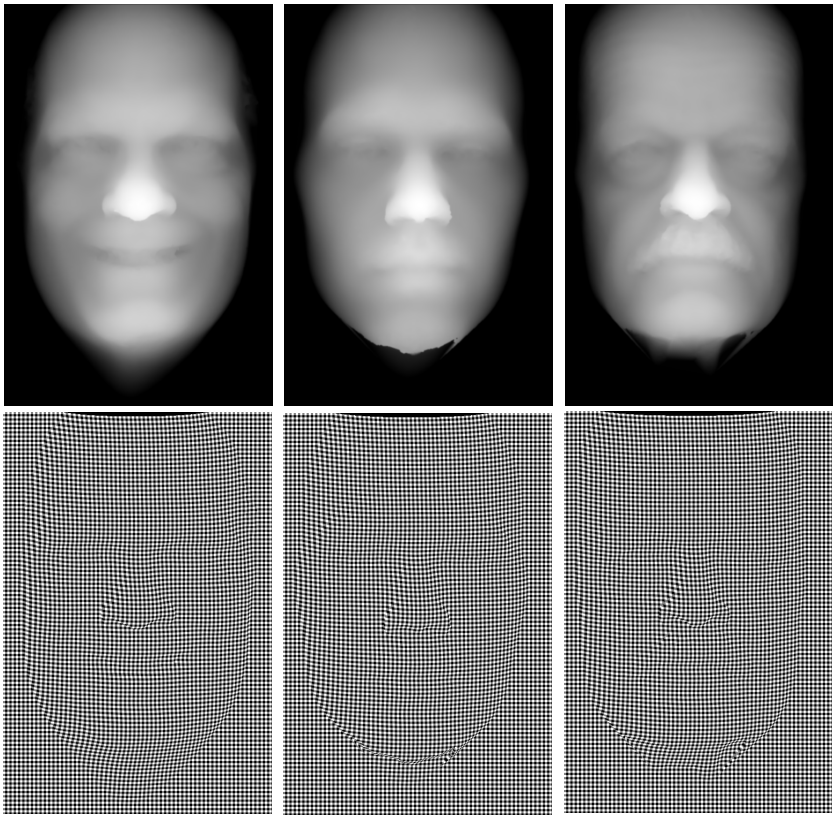

Figure 2.23: The first row shows examples of real faces' depth of the Texas Database. Second row shows examples of the simulated data that we would obtain if we project the fringes pattern over each test surface.

reconstruction while fine details of the faces are kept as can be seen in Fig. 2.25.

Table 2.3 summarizes the numerical results obtained for the test face display in Fig. 2.25. As can be seen, the proposed 3D gradient approach is capable of retrieving most face details, even if the surface presents discontinuities and noise.

### Comparison with other 3D retrieving techniques

In this second series of experiments, we repeat some of the previous examples including other 3D scanning techniques -described above- such as Binary Stripe Projection (BSP), Phase Shifting (PS) and Takedas' approach (Tak). Figure 2.26 shows the simulated images for each scanning technique after projecting each pattern over a test surface. Figure 2.27 shows the Ramp-Peaks reconstructed surfaces

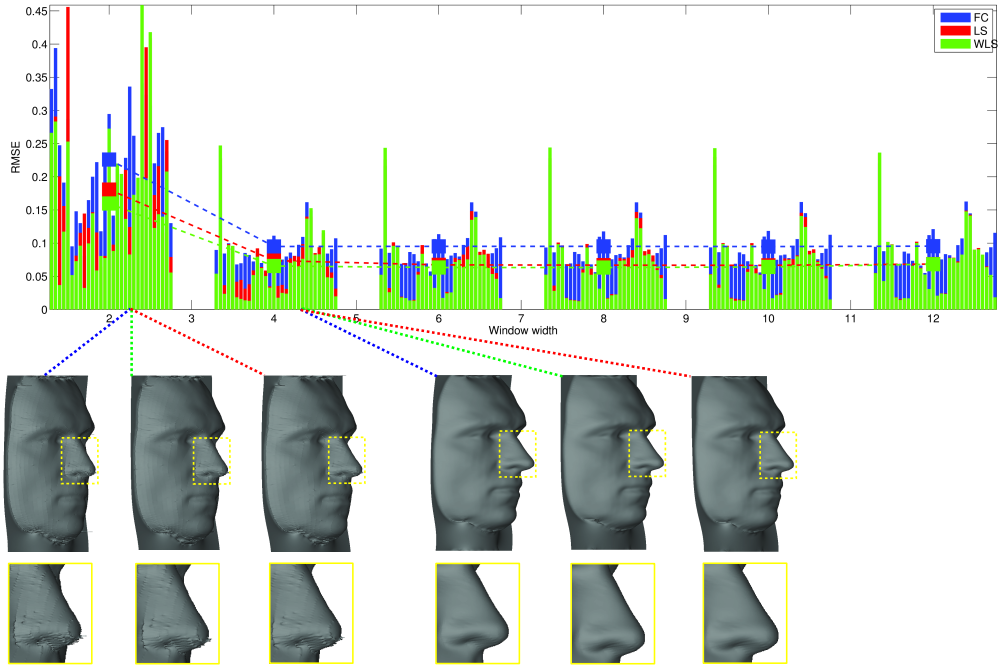


Figure 2.24: Root mean squared error (relative to surface amplitude) for a set of *fast algorithms* (LS, WLS, FC) varying the width of the median filter used to preprocess the input gradient field. In the graphic is shown the median value and each individual RMSE error obtained for 30 faces of the Texas database.

| Method | LS | FC | M | WLS | AD | L1 |
|---|---|---|---|---|---|---|
| RMSE (%) | 3,7 | 8,5 | 3,4 | 2,4 | 3,2 | **0,81** |
| Ex. Time | $12s$ | **2s** | $24s$ | $6s$ | $5m32s$ | $5m23s$ |

Table 2.3: Comparison of integration techniques for one face of the Texas Database. The root mean squared error is displayed as a percentage of surface amplitude, and execution time corresponds to an experiment performed on MATLAB and a Intel Core i5 CPU M 430 @ 2.27GHz x 4, the size of the discrete input gradients ($p$ and $q$) was $750 \times 500$ pixels.

obtained by: (a) proposed approach (D3D), (b) Takedas' approach (Tak), (c) a three steps phase shifting technique (PS) and (d) a 8 bit Binary codded stripes approach (BCS). The RMSE is normalized with respect to surface amplitude and the number of projected patterns that each method requires is also specified below each image.

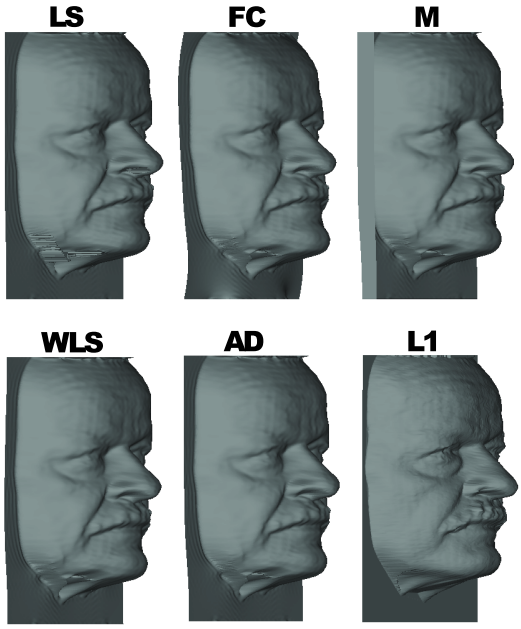


Figure 2.25: Example of a face surface of the Texas Database reconstructed by different integration approaches. The retrieved gradient field was preprocessed width a $4 \times 4$ median filter.

In a second set of experiments, real data from the Texas Database is considered. Figure 2.28 illustrates the results obtained for some arbitrary test faces. As can be seen the proposed technique is capable to extract faces shape (although deformation can appear in the background plane). While Takedas' and PS approaches perform extremely well for simple examples, in more complex scenarios the unwrapping steps become more challenging and can lead to errors (see e.g. the third and fourth columns of Fig. 2.28). On the other hand Binary coded stripes methods tend to be very robust and as the measurements are absolute (each pixel's depth is determined without considering the other ones), this technique is very useful when we have to deal with heavy discontinuities or isolated regions. In these experiments quantization error can be notice. This is because our implementation of BCS can detect disparities at least of one pixel (subpixel techniques can improve quantization resolution). Of course the main drawback of PS and BCS techniques is that they require multiple images to achieve one single reconstruction, hence they are in practice more complicated to implement. For example, a synchroniza-

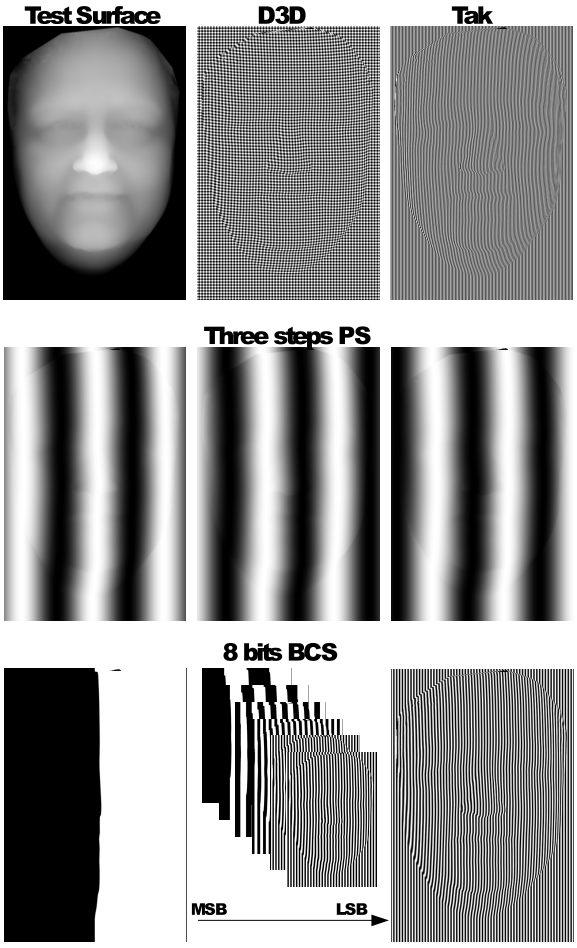


Figure 2.26: Illustration of different input images for the same test face (of the Texas Database). The first image (upper left) illustrates the height map (proportional to the gray level of the image), and the other images illustrates the deformed patterns obtained for different methods.

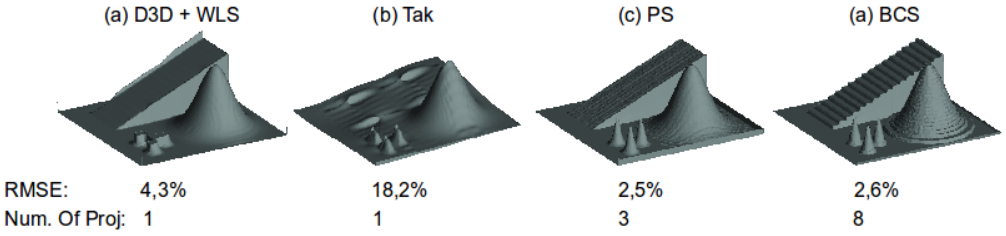


Figure 2.27: Results for the Ramp-Peaks example

tion between the camera and the projector is needed and also the scene must be rigid (or quasi-static). The code of our implementation of each algorithm as well

as additional implementation concerns (discussed in the comments of the code) can be found in Appendix D.1.

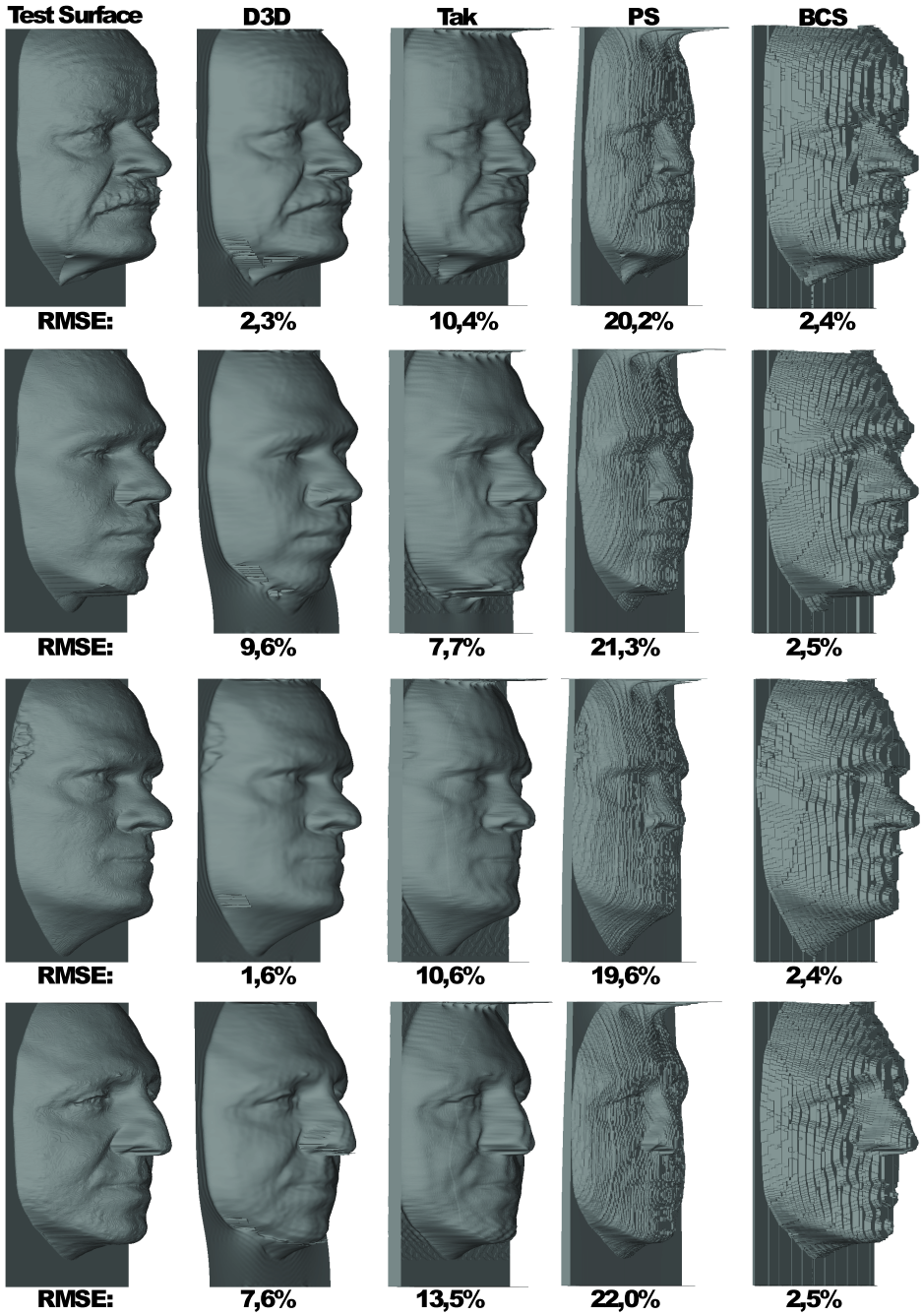


Figure 2.28: Results over a representative set of faces of Texas Database.

### On field parameters evaluation

Until now, we have identified the most important sources of error and characteristics of the proposed technique. In addition, we have explored and evaluated the impact of the integration step and compared D3D with other popular 3D scanning techniques. In this new set of experiments we propose to evaluate the impact of varying fringes' period and the power of the projected light. To that end, we manufacture an hemispherical surface (with a radius of $40mm$) using a lathe with which we could achieve a precision of the order of $0{,}1mm$. Figure 2.29 shows some steps of the manufacturing process. We chose this kind of surface because we can easily fit an analytical expression to it which make the quantitative evaluation easier.

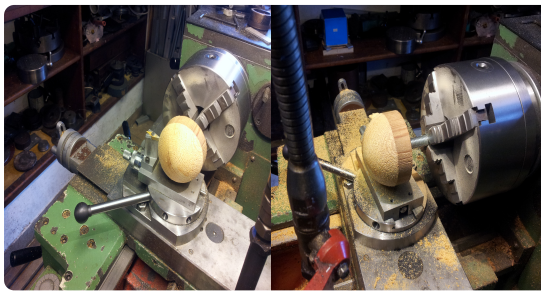

Figure 2.29: Illustration of the manufacturing process of the test surface.

In a first experiment, we projected several patterns over our test surface varying fringes' period as illustrated in Fig. 2.30. We explore two different codification strategies of vertical and horizontal fringes, multiplying and summing it. Figure 2.31 shows an example of (a) the ground truth gradient field, (b) the estimated gradient gradient field, (c) the error between true and estimated gradient field, and (d) the surface reconstructed from the estimated gradient field. For the quantitative analysis we isolated the borders of the sphere, because this region presents discontinuities in $x$ and $y$ partial derivatives and we already investigate such kind of errors. In the present set experiments we want to identify and quantify the impact of new sources of error.

In a second set of experiments, we introduce a arbitrary ambient illumination

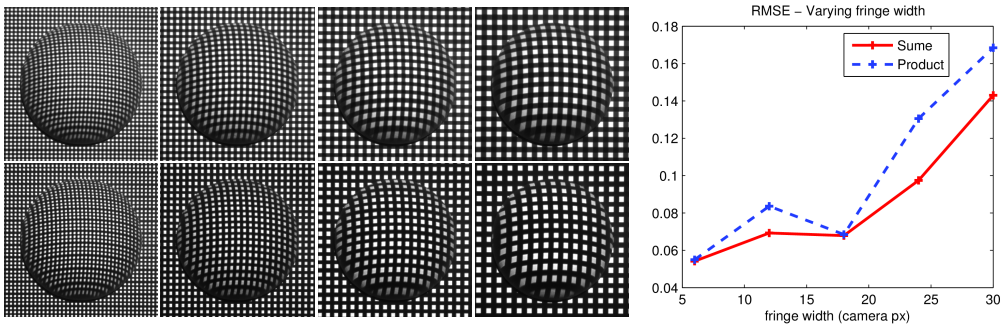


Figure 2.30: Left images, illustrates some of the acquired images when we projected summed (first row) and multiplied (second row) vertical and horizontal fringes. On the right we show the obtained RMSE over the gradients fields for different fringes widths.

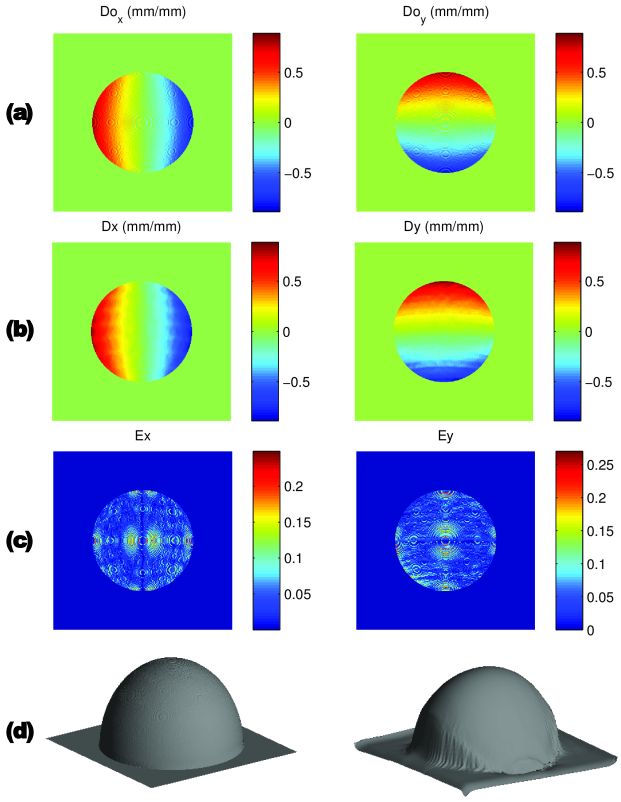


Figure 2.31: (a) Ground truth gradient field, (b) estimated gradient gradient field, (c) error between true and estimated gradient field and (d) surface reconstructed from the estimated gradient field.

and varied the power of the projected pattern. The idea was to investigate the impact of uncontrolled sources of light that may affect the perceived deforma-

tion of the projected pattern. We measure the intensity of the incident ambient illumination which was approximately $0{,}9\mu W/mm^2$, we project patterns starting from an intensity of $0{,}01\mu W/mm^2$ up to $3{,}1\mu W/mm^2$. Figure 2.32 shows some of the acquired images when we projected summed (first row) and multiplied (second row) vertical and horizontal fringes while varying patterns' intensity; the right plot shows the obtained RMSE over the gradients fields.

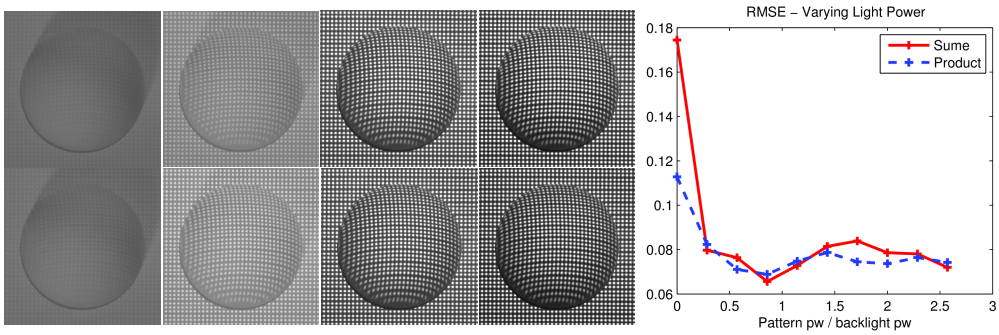


Figure 2.32: Left images, illustrates some of the acquired images when we projected summed (first row) and multiplied (second row) vertical and horizontal fringes. On the right we show the obtained RMSE over the gradients fields for different relations between the background light and pattern's light power (background light intensity was approximately $0{,}9\mu W/mm^2$) .

From the first experiment, we can see that fringes' width up to $18px$ leads[5] to an error below the 10 %, after this value errors increase significantly. On the other hand, concerning pattern intensity, the experimental results suggest that a pattern with an intensity of the order 20 % of the ambient illumination intensity is enough to achieve a root-mean-squared error below the 10 %. This relatively low amount of required intensity may be due to the fact that we are looking at high spacial frequencies of acquired images, while ambient illumination uses to be smooth and hence its power is concentrated in the low frequencies.

[5] 18 pixels measured in the camera sensor, correspond in this setup to $4{,}5mm$ in the reference plane and 6 pixels on projector sensor.

### Last experiment: scanning a dynamic surface

In this last experiment we present the results of projecting the proposed rectangular pattern over the face of a subject that was speaking and moving. As the proposed technique requires the projection of a unique binary pattern, in order to reconstruct a dynamic scene we just need to acquire a standard video sequence of the test surface while the pattern is being projected. Figure 2.33 shows some frames of the acquired video, while Fig. 2.34 shows the estimated depth gradient field by measuring local fringes deformation as explained previously. Figure 2.35

Figure 2.33: Different frames of the input video.

shows the reconstruction results after integrating the estimated gradient field (following a weighted least squares approach). As we can see, the proposed technique is capable to capture fine details of the 3D shape of the observed surface for a moving and complex scenario, while keeping very low hardware and computational requirements.

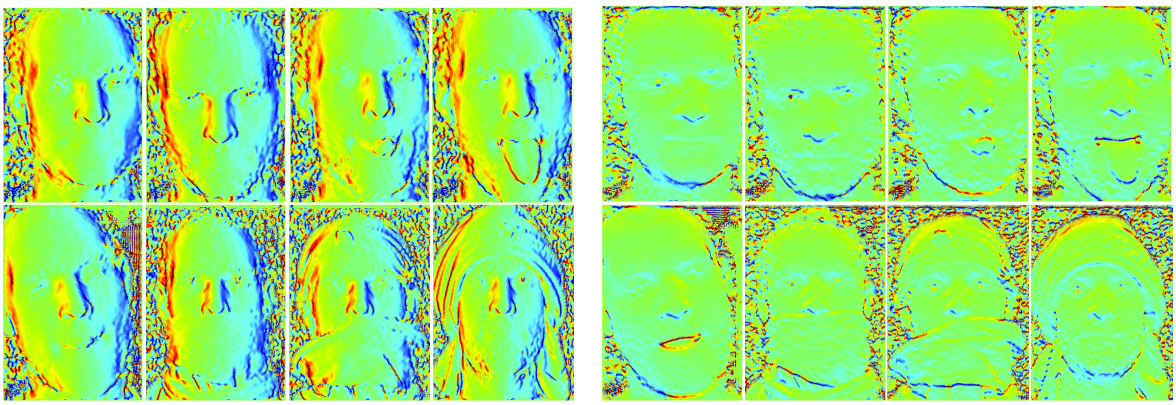

Figure 2.34: Raw frames of estimated gradient field.

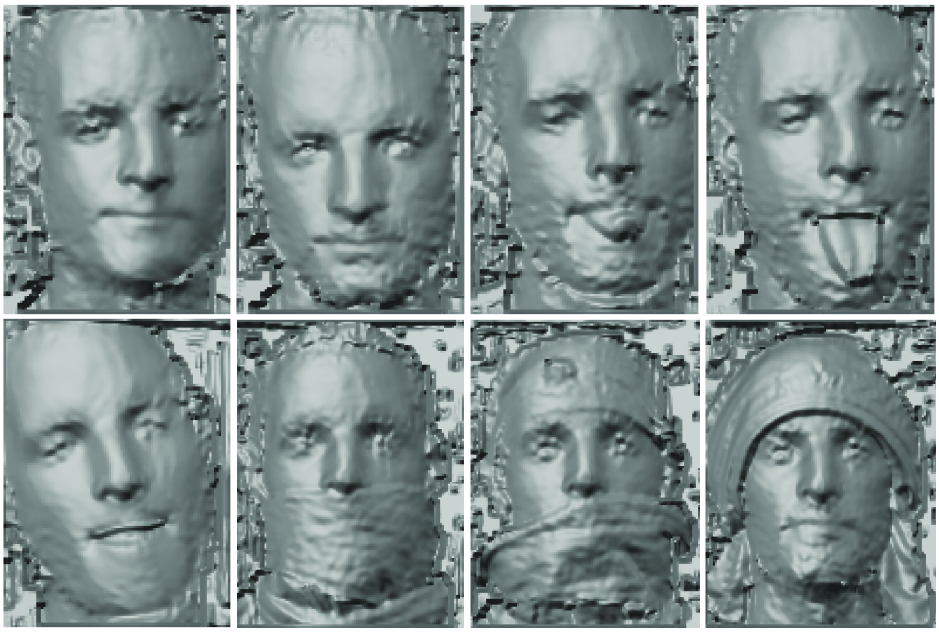

Figure 2.35: Frames of the retrieved surface.

## 2.3. Conclusions

In this chapter a subset of the most popular 3D retrieval methods have been reviewed, particularly those methods based on the projection of structured light (e.g. stripe coding or phase shifting techniques). In addition a novel gradient approach was proposed which allows us to retrieve in a very straightforward manner the depth gradient map. An analytical description and a geometrical interpretation of the proposed technique were provided which allows us to identify practical error sources and limitations the presented approach has. As we have shown, the two main error sources are: (a) smooth regions of the image (where fringes present a local maximum or minimum), and (b) surface discontinuities. Experiments suggests that the first kind of errors can be overcome by preprocessing the input gradient field with a nonlinear filter (e.g. a simple median like filter). On the other hand, as the proposed technique measures depth gradient maps, erroneous measurements will be obtained in those points where the surface presents discontinuities (points where derivatives are not even defined). We showed that the previous problem can be handled by estimating a discontinuity map which guides the integration process.

In order to obtain the depth map from the retrieved gradient field, it is necessary to integrate it. To this end we can follow different integration approaches, each of them with its own advantages and drawback. In this chapter we briefly described and used popular integration techniques, leaving to the following chapter a detailed address of this problem.

Experiments were presented with simple and complex synthetic data, as well as real dynamic data. The presented approach was tested and compared to other common scanning techniques such as Binary Stripes Projection, Phase Shifting and Takeda's techniques.

The procedure for 3D-shape retrieval described in the present work has various advantages with respect to other fringe projection methods presented in the literature. Firstly, image partial derivatives can be computed in an extremely fast and efficient manner. Secondly, as high frequency fringes are projected the proposed

method is robust under smooth ambient illumination and/or surface's reflectivity variations. Thirdly, instead of the disparity itself, we are computing partial derivatives which are extremely useful to extract 3D features of the scene. This is particularly important in those applications [61, 88] in which gradient depths are required, e.g. to have invariant SIFT descriptors [57], perform 3D face recognition [15,61,103], gesture detection [92] or objects segmentation [11]. Finally as only one binary pattern is used, no complicated calibration steps are required (such as color calibration or gamma correction), nor synchronization between the projector device and the camera.

# Chapter 3

# Surface reconstruction from gradient fields

In many applications such as Photometric Stereo [9,36,98], Shape from Shading [48, 90, 99], Differential 3D reconstruction [20, 25] and Image Editing in gradient domain [12,77] it is important to integrate a retrieved gradient field. In most cases, the retrieved gradient fields correspond to non integrable fields (i.e. they are not irrotational on every point of the domain). Robust approaches have been proposed to deal with noisy non integrable gradient fields. In this chapter we review some of these techniques -Sec. 3.2-, and we adapt them to the case of dynamic scenes when the gradient field in the $x-y$ domain can be estimated over time -Sec. 3.3-. We exploit temporal consistency of scenes to ensure integrability and improve the accuracy of integration methods.

## 3.1. Introduction

The main problem when dealing with estimated gradient fields[1] is that, due to empirical errors and noise, these gradient fields are not usually integrable, i.e.

[1] We will represent the input gradient field by $(p, q)$ where $p$ and $q$ represents the $x$ and $y$ components of the raw input gradient field, respectively.

they do not represent irrotational fields.

Several approaches were proposed to enforce integrability: (i) Frankot and Chellappa [37] represent the gradient field by a finite set of basis functions and integrability is enforced by calculating the orthogonal projection onto a vector subspace spanning the set of integrable slopes. In particular the case of Fourier basis functions is addressed in [37] because it has an intuitive interpretation and is computationally very efficient, e.g. by using Fast Fourier Transforms. Many variants of this approach have been proposed by either choosing different basis functions e.g. Cosine functions or using a redundant non-orthogonal set of basis functions (shapelets) [59]. (ii) Other approaches are based on least squares solutions [51,87] where the $L_2$-norm of the error[2] is minimized. This minimization process leads to a Poisson Euler Lagrange equation and because of that this methods are sometimes referred as Least Squares (LS) or Poisson methods. (iii) More recently Du et al. [28] propose a robust surface reconstruction algorithm by minimizing the absolute error between the input and the estimated gradient field ($L_1$-norm). This approach is less sensitive to outliers than LS and projection approaches. (iv) Wei and Reinhard [95] presented a generalization of Frankot-Chellappa algorithm were constraints on higher order derivatives are imposed in order to handle noisy and corrupt data. (v) Agrawal et al. [4] proposed a generalized equation to represent a continuous of surface reconstruction solutions. They generalized a set of popular approaches (such as LS and Frankot-Chellappa) by introducing anisotropic weights in the integration process, examples of these approaches are: (a) $\alpha-$surfaces using binary weights, (b) M-estimators and edge preserving regularization using continuous weights and ( c) Diffusion using affine transformation of gradients (based on Anisotropic Diffusion ideas of Weickert et al. [96]). Agrawal's thesis is an excellent review of some of the previously mentioned methods [2]. (vi) Ng et al. [73] proposed the use of kernel basis functions, which turn the continuous surface reconstruction problem into a high-dimensional space, where a closed form solution exists.

In the next section, a selected set of the previously mentioned methods are

[2] i.e. $E[u] = \int\int \left(|u_x - p|^2 + |u_y - q|^2\right) dxdy$

reviewed and important implementations details are also discussed.

## 3.2. Integration approaches

### 3.2.1. Least Squares method

Suppose that $Z : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ is the surface obtained by integrating a given gradient field $(p, q) : \Omega \to \mathbb{R}^2$ using a particular algorithm. One always can write,

$$(Z_x, Z_y) = (p, q) + (\epsilon_1, \epsilon_2) \tag{3.1}$$

where the subindex $x$ and $y$ denotes $x$ and $y$ partial derivatives respectively and $(\epsilon_1, \epsilon_2)$ represents a residual gradient field which is added to the given non-integrable field to make it integrable.

The least squares solution is obtained by minimizing the energy function:

$$E_{LS}[u] = \int_\Omega \left((u_x - p)^2 + (u_y - q)^2\right)\, dx\, dy = \int_\Omega \left(\epsilon_1^2 + \epsilon_2^2\right)\, dx\, dy \tag{3.2}$$

and hence $Z_{LS} = argmin\{E_{LS}[u]\}$ is the solution with minimum squared error. This solution can be seen as the orthogonal projection of the gradient field $(p, q)$ onto the subspace of integrable gradient field. The Euler-Lagrange equation of (3.2) gives the Poisson equation [Appendix A]:

$$\begin{cases} \nabla^2 Z = \texttt{div}(p, q) \\ (Z_x - p)\nu_1 = 0 \text{ on } \partial\Omega \\ (Z_y - q)\nu_2 = 0 \text{ on } \partial\Omega \end{cases} \tag{3.3}$$

where $\nu_i$ is the $i$-th component of the outward unit normal on $\partial\Omega$.

We can find a numerical solution of Eqs. 3.3 using finite differences and vectoring the 2D matrices in lexicographical ordering (we share our implementation of the method in Appendix D.2.1, pag. 123). Equations 3.2 can be transformed into a linear and sparse equation system $Ax = b$, where matrix $A$ has not full rank because the problem defined by minimizing the energy 3.2 is an ill-posed problem, particularly because $E_{LS}[u] = E_{LS}[u + K]$ for any arbitrary constant $K$. This

problem can be solved adding one additional equation to the linear system $Ax = b$, e.g. we can set the value of one pixel or impose that $\sum_i x_i = 0$ to define a problem with a unique solution. In practice we found that robust numerical methods such as the presented in [46] achieves accurate solutions even when the matrix $A$ has not full rank.

### 3.2.2. Fankot-Chellappa method

Frankot and Chellappa (FC) proposed an integration method that enforces integrability by projecting the input gradient field $(p, q)$ onto the subspace of integrable gradient fields. This method is mathematically equivalent to the minimization procedure described in the previous section, but in this case the minimization is achieved by a projection in the Fourier domain which can be done in a very efficient manner using Fast Fourier Transform algorithm and Fourier properties.

Using Fourier Basis, a surface $Z : \Omega \to \mathbb{R}$ can be represented by:

$$Z(x, y) = \int \mathcal{F}\{Z\}_{(\omega_x,\omega_y)} \, e^{j(\omega_x,\omega_y)\cdot(x,y)} \, d\omega_x d\omega_y. \tag{3.4}$$

where $\mathcal{F}\{Z\}$ is the Fourier transform of $Z$ defined as:

$$\mathcal{F}\{Z(x, y)\} = \frac{1}{2\pi} \int Z(x, y) e^{-j(\omega_x,\omega_y)\cdot(x,y)} \, dxdy. \tag{3.5}$$

The derivatives of the Fourier basis functions possess the following useful properties,

$$\mathcal{F}\left\{\frac{\partial Z}{\partial i}\right\} = j\omega_i \mathcal{F}\{Z\} \quad i \;=\; x, y. \tag{3.6}$$

Thus it is straightforward that the solution of the Poisson equation which minimizes 3.2 can be computed as:

$$Z = \mathcal{F}^{-1}\left\{\frac{-j\omega_x \mathcal{F}\{p\} - j\omega_y \mathcal{F}\{q\}}{\omega_x^2 + \omega_y^2}\right\}. \tag{3.7}$$

In this approach, the ill-posedness of the integration problem produces a numerical 0/0 division which occurs when $\omega_x = \omega_y = 0$, this simply means that we cannot recover the average value of $Z$ without additional information. Appendix D.2.2 shares a possible numerical implementation for Frankot-Chellappa methods.

### 3.2.3. $L_1$-min method

Methods that minimizes $L_2$-norm of the error e.g. LS and FC are very sensitive to the presence of outliers because of the quadratic growth of the penalty function. To solve this problem we can either use a different error norm so that outliers have less influence in the global error, or keep the $L_2$-norm but introduce weight to handle outliers and corrupt measurements. In this section we present a $L_1$-norm method and in the following section we shall explore weighted techniques.

Instead of minimizing the Energy function defined in Eq. 3.2 Du et al. [28] define the energy function:

$$E_{L1}[u] = \int_{\Omega} (|u_x - p| + |u_y - q|) \; dx\, dy = \int_{\Omega} (|\epsilon_1| + |\epsilon_2|) \; dx\, dy. \tag{3.8}$$

The main challenge of minimizing the $L_1$ error is that it is not differentiable and hence more complicated optimization techniques should be considered. Du et al. redefine the minimization problem of the functional 3.8 as,

$$\begin{cases} \{u, a, b\} = argmin \left\{ \displaystyle\int_{\Omega} (a(x,y) + b(x,y)) dx\, dy \right\} \\ st. \quad u_x - p \leq a \\ \qquad u_x - p \geq -a \\ \qquad u_y - q \leq b \\ \qquad u_y - q \geq -b \\ \qquad a \geq 0 \quad b \geq 0 \end{cases} \tag{3.9}$$

Using finite differences we can transform the previous minimization problem in a constrained linear problem (LP) where the target function and the constraints are both linear in the variables $u$, $a$ and $b$. In the Appendix D.2.3 we share one possible implementation of the $L_1$-norm approach.

### 3.2.4. M-Estimators method

As we mention in the previous sections, the $L_2$-norm is very sensitive to the presence of outliers, we can reduce the effect of outliers by introducing weight on

the error functional i.e.

$$E_M^k[u] = \int_\Omega \left( w(\epsilon_1^{k-1})(u_x - p)^2 + w(\epsilon_2^{k-1})(u_y - q)^2 \right) \, dx\, dy. \tag{3.10}$$

The weights ($w_x = w(\epsilon_1^{k-1})$, $w_y = w(\epsilon_2^{k-1})$) at step $k$ depend on the residual error at $k-1$ step using some function $w(\cdot)$ that is: symmetric, positive-definite and chosen to be less increasing than square. In this iterative method, we solve in each step the weighted Poisson equation:

$$\texttt{div}(w_x\, Z_x, w_y\, Z_y) = \texttt{div}(w_x\, p, w_y\, q). \tag{3.11}$$

With the solution of Eq. 3.11 we obtain the new weights by evaluating at each point the residuals $\epsilon_1$, $\epsilon_2$ and we repeat this procedure until convergence. In the Appendix D.2.4 we share the implementation of the M-Estimators approach.

### 3.2.5. Anisotropic Diffusion method

Another alternative to handle outliers and errors while keeping the $L_2$-norm to penalize errors, is to calculate a weight map from the available information in the input gradient field $(p, q)$.

The technique we will describe is based on Anisotropic Diffusion (AD) ideas proposed in the context of image restoration [78] and multi-scale image analysis [6]. We will focus on the ideas proposed by J. Weickert [96, 97] and we will adapt the general framework to the integration problem. Instead of solving the Poisson equation given by Eq. (3.3), the differential equation

$$\texttt{div}\left(D \cdot \nabla u\right) = \texttt{div}\left(D \cdot (p, q)\right). \tag{3.12}$$

is considered. $D$ represents an affine transformation ($2 \times 2$ symmetric matrix for each $x$, $y$). The above equation can be thought as the Euler-Lagrange equation of the following energy function:

$$\begin{aligned} E[u] = \int_\Omega (d_{11}(u_x - p)^2 + (d_{12} + d_{21})(u_x - p)(u_y - q) \\ + d_{22}(u_y - q)^2) dx dy \end{aligned} \tag{3.13}$$

where $d_{ij}(x, y)$ can be interpreted as weights on the gradient field. Intuitively, the main idea is to identify features such as borders and exploit local coherence to guide the integration through smooth and coherent areas, avoiding large discontinuities and noisy points. To this end, the affine transformation $D$ is obtained considering $p$ and $q$ and calculating (for each $(x, y)$) the $2 \times 2$ matrix

$$H(x, y) = (p\ q)^T (p\ q). \tag{3.14}$$

The resulting matrix $H$ has an orthonormal basis of eigenvectors $v_1$ and $v_2$, where $v_1$ is parallel to the local gradient and $v_2$ is perpendicular to it. $H$ can be rewritten as

$$H = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix}. \tag{3.15}$$

In order to penalize large gradients, $H$ eigenvectors are modified following [97],

$$\begin{cases} \lambda_2 = 1 \\ \lambda_1 = \begin{cases} 1 & \text{if } \mu_1 = 0 \\ \beta + 1 - \exp\left(-3{,}315/\mu_1^4\right) & \text{if } \mu_1 > 0 \end{cases} \end{cases} \tag{3.16}$$

and finally the tensor is obtained preserving $H$ orientation but with the modified eigenvalues:

$$D = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix}. \tag{3.17}$$

The parameter $\beta$ in Eq. (3.16) was introduced in [2] to ensure positive-definiteness of $D$ but was not mentioned in [6] and [96]. We set $\beta = 0{,}02$ in order to maintain consistency with [2]. In addition, we compared the results to those obtained for $\beta = 0$ (as is described in [6] and [96]); for the set of experiments we performed, the results were equivalent.

Appendix D.2.5 details AD integration method implementation.

### 3.2.6. Methods evaluation

We compared Least Squares (LS) [Sec.3.2.1], Frankot-Chellappa (FC) [Sec.3.2.2], $L_1$-norm minimization (L1) [Sec.3.2.3], M-estimator using Huber weight function

(M) [Sec.3.2.4] and Anisotropic Diffusion (AD) [Sec.3.2.5] approaches. Figure 3.1 shows the input gradient field which was obtained from the Ramp-Peaks surface [2–4] and to whom it was added Gaussian noise (with $\sigma = 0{,}05g$, $g$ = maximum gradient magnitude) and uniformly distributed outliers. As the main reason

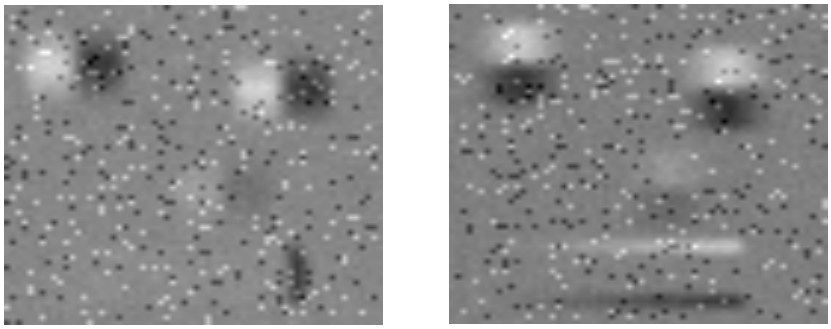

Figure 3.1: Illustration of the input gradient field (left image corresponds to $p(x, y)$ and right image to $q(x, y)$) which contains white Gaussian noise and outliers.

to consider more complex integrations approaches (e.g. L1 and AD) is to be able to handle noise and outliers on input gradient fields, we decide to evaluate also the impact of performing a denoising preprocessing step. For example, we evaluate the performance of each integration technique after applying very simple and common denoising filters such as Gaussian filter (which is very effective on removing Gaussian noise) or a non-linear median filter (which is particularly useful to remove outliers). Figure 3.2 compare the MSE obtained after different (linear and non-linear) preprocessing steps for different integration methods. Figure 3.3 shows the reconstructions obtained from the raw input gradient field and the filtered ones. As we can see, robust approaches (like $L_1$-norm minimization, M-estimator or Anisotropic Diffusion) obtain better results when we integrate a gradient field with substantial power of noise and outliers (see e.g. the results in Fig. 3.2). Despite this, accurate results can also be achieved with very trivial integration techniques if before the integration, a non linear preprocessing step is considered (Fig. 3.2). In addition, there are also other aspects that must be taken into account when evaluating different integration approaches, e.g.: (a) implementation complexity, and

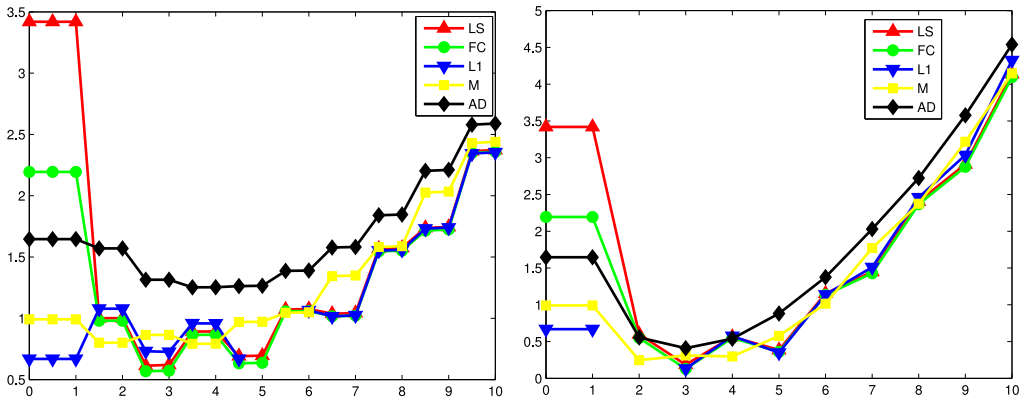


Figure 3.2: Comparison of the Mean Squared Error for the set of presented integration techniques. Left: MSE obtained from the filtered gradient field, the abscissa correspond to the variance of the Gaussian Kernel used for averaging. Right: MSE obtained from the filtered gradient field using a non linear median filter, squared windows where used, the abscissa indicates the side length on pixels.

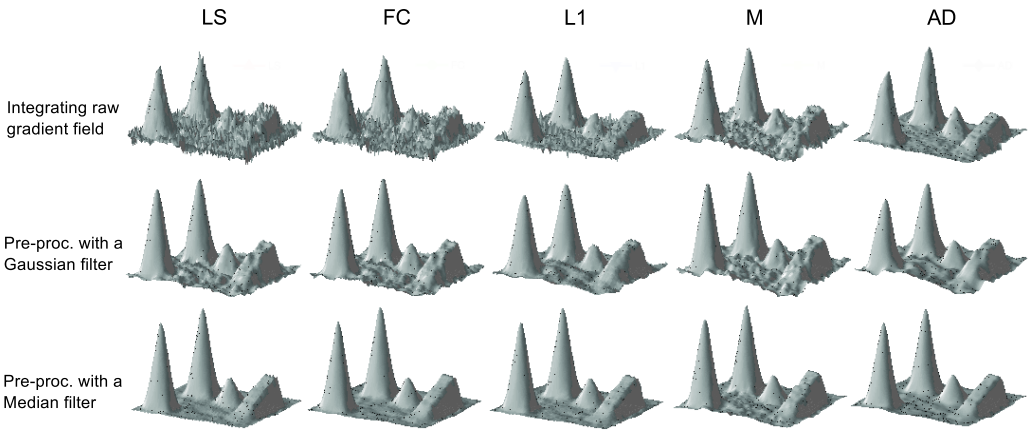


Figure 3.3: First row: reconstructions without preprocessing. Second row: reconstructions using the smoothed gradient field (where the width of the Gaussian Kernel was the one that minimizes the MSE for each integration technique). Third row: reconstructions using a median filter as a preprocessing step (the window size was the one that minimizes the MSE for each integration technique).

(b) execution time and computational cost. To have an idea of the implementation complexity of the described algorithms, we share the code of our Matlab implementation of each technique in Appendix **??**. Table 3.1 shows the mean execution time of each integration approach (relative to the fastest algorithm). For example,

while $L_1$ approach presents a remarkable robustness and keeps edge definition, it takes four orders of magnitude more time.

| Method | LS | FC | L1 | M | AD |
|---|---|---|---|---|---|
| Rel. Time | 36 | 1 | 25925 | 170 | 1637 |

Table 3.1: Mean execution time of each algorithm (relative to the fastest one).

## 3.3. Gradient field of dynamic scenes: exploiting temporal consistency

In the previous section, we reviewed a representative set of classical and state of the art integration methods. They were developed for two-dimensional problems where one have a estimation of the $x$ and $y$ partial derivatives $(p(x, y), q(x, y))$ of an unknown function $z(x, y)$. In this section we present a generalization for the case of dynamic scenes and we exploit the consistency in the movement of the scene to guide the integration process and make the algorithms more robust to noise and outliers. It is important to highlight that this is not a simple extension of integration approaches from $2D$ to $3D$ (as in [94]); we will focus on three-dimensional problems (where we have two spatial coordinates plus a temporal variable) in which we still have $x$ and $y$ partial derivatives but no information about the temporal partial derivative is available. Examples of such problems are Differential 3D reconstruction and Photometric Stereo where one wants to integrate the gradient field $(p(x, y, \mathbf{t}), q(x, y, \mathbf{t}))$ and do not have additional information regarding the temporal partial derivative of the target function. This asymmetry between $x$, $y$ and $t$ variables will give to the time variable the special role of constraining the space of solutions as we will demonstrate.

In the following, we will describe two possible ways of regularize the space of solutions. Firstly we will concentrate on those cases in which the image of the scene varies in a continuous way, and secondly, we will extend the analysis to those

scenes where the observed objects are moving.

### 3.3.1. Constraining scene variation

In the rest of the section, we will assume that it is possible to estimate $x$ and $y$ partial derivatives ($p(x,y,t)$ and $q(x,y,t)$ respectively) for a unknown function $z : \Omega \times [0,T] \rightarrow \mathbb{R}$ where $\Omega \subset \mathbb{R}^2$.

As was described in Sec. 3.2.1, to enforce integrability one can find the surface that minimizes the energy function:

$$E[u] = \int_{\Omega} \left( |u_x - p|^2 + |u_y - q|^2 \right) dxdy \tag{3.18}$$

whose Euler-Lagrange equation leads to the Poisson equation[3].

$$u_{xx} + u_{yy} = \mathtt{div}\,(p,q)\,. \tag{3.19}$$

The straightforward way to extend the previous solution to dynamic scenes, consists in solving Eq. (3.19) for each time $t$ independently. Instead, we consider the new functionals

$$E[u] = \int_{\Omega \times [0\ T]} \left( (1-\lambda) \left( |u_x - p|^2 + |u_y - q|^2 \right) + \lambda \left| \mathbf{d}^{\alpha}\mathbf{u}/\mathbf{dt}^{\alpha} \right|^{\mathbf{2}} \right) dxdydt \tag{3.20}$$

and

$$\begin{aligned} E[u] = \int_{\Omega \times [0\ T]} ((1-\lambda) \left( d_{11}(u_x - p)^2 + (d_{12} + d_{21})(u_x - p)(u_y - q) \right. \\ \left. + d_{22}(u_y - q)^2 \right) + \lambda \left| \mathbf{d}^{\alpha}\mathbf{u}/\mathbf{dt}^{\alpha} \right|^{\mathbf{2}}) dxdydt \end{aligned} \tag{3.21}$$

where $\alpha$ is a natural number that denotes the order of the temporal derivative to be considered. Expressions (3.20) and (3.21) can be seen as a generalization of LS and AD methods for the case of dynamic scenes and adding a temporal constraint. In both cases by adding a regularization term over the time domain, we are enforcing the solution to have continuous variations between frames (case $\alpha = 1$) or low acceleration (for $\alpha = 2$) depending on the nature of the scene. In all the cases,

[3] in this section, $\nabla$ and $\mathtt{div}\,()$ operators are those defined in the previous sections: $\nabla f(x,y,t) = (f_x, f_y)^T$ and $\mathtt{div}\,(f(x,y,t)) = f_x + f_y$.

$\lambda$ is a parameter that must be empirically set (this is a standard procedure in image restoration [6]) and determines the weight of the regularization term. The proposed framework allows us to introduce different kinds of restrictions over the dynamics of the scene, but also provides a method for solving the problem using all the data simultaneously. As we will show, the proposed approach is more robust and flexible than solving the integration problem frame by frame and then trying to average or combine the results obtained for each time independently.

The function that minimizes Eqs. (3.20) and (3.21) must satisfy,

$$(1-\lambda)\left(\mathtt{div}\left(\nabla u\right)\right)+(-1)^{\alpha-1}\lambda\frac{d^{2\alpha}u}{dt^{2\alpha}}=(1-\lambda)\left(\mathtt{div}\left((p\ q)^T\right)\right) \tag{3.22}$$

$$\langle\nabla u,n\rangle=0\ \text{ on }\partial\Omega\times[0\ T] \tag{3.23}$$

$$\frac{d^iu(x,y,0)}{dt^i}=\frac{d^iu(x,y,T)}{dt^i}=0\ \text{ on }\Omega\text{ and }i=1..(2\alpha-1) \tag{3.24}$$

and

$$(1-\lambda)\left(\mathtt{div}\left(D\nabla u\right)\right)+(-1)^{\alpha-1}\lambda\frac{d^{2\alpha}u}{dt^{2\alpha}}=(1-\lambda)\left(\mathtt{div}\left(D(p\ q)^T\right)\right) \tag{3.25}$$

$$\langle D\nabla u,n\rangle=0\ \text{ on }\partial\Omega\times[0\ T] \tag{3.26}$$

$$\frac{d^iu(x,y,0)}{dt^i}=\frac{d^iu(x,y,T)}{dt^i}=0\ \text{ on }\Omega\text{ and }i=1..(2\alpha-1) \tag{3.27}$$

respectively. The detailed deduction of these conditions is presented in the Appendix B. The implementation of these methods is presented in Appendix. D.2.6 where we address and described the most important implementation details.

### 3.3.2. Constraining objects movement

A different way to enforce integrability, could be constrain objects movement instead of the variations on the view of the scene. The main difference and motivation of this second regularization approach is illustrated in the toy examples (a) and (b) shown in Fig. 3.4. Even if we assume that the objects in the scene are moving in a continuous way, if the scene present discontinuities or regions with large spatial gradients, constraining scene variation (i.e. constraining $|Z_t|$) does not seems an adequate approach. Instead it could be desired to constrain the *real* spatial velocity of the objects in the scene. To this end, we will use the ideas behind

optical flow techniques to derive a new regularization term which is more adequate when we have borders and large spacial gradients.

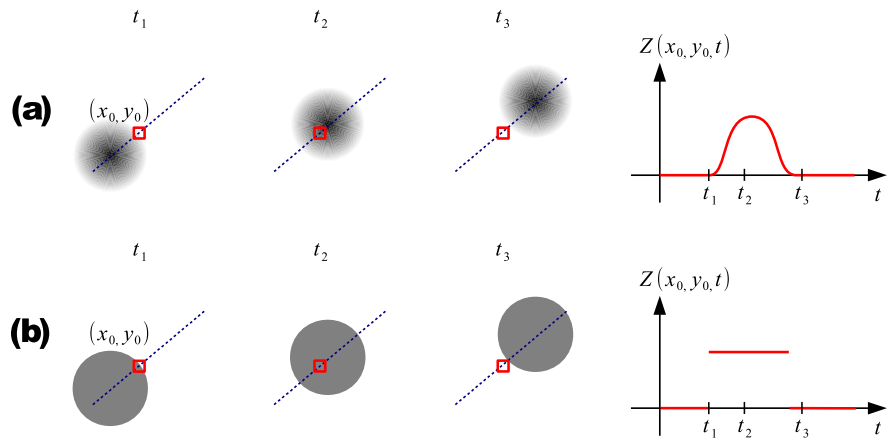


Figure 3.4: Illustration of two examples that show the motivation of the different kind of regularization. The firs example (a) illustrates a smooth height map varying along the time. The second example (b) illustrates a second scene which -also varies smoothly over time- and presents discontinuities over the height map.

We will firstly assume that all the objects present in the scene move with a unique (and constant) velocity, and in a second analysis we will present a more general approach in which different objects are allowed to move at different (and variable) velocity.

Let $Z(x, y, t)$ be the height of the test surface and let $(x(t), y(t))$ be the trajectory of a material point projected to the image plane. The depth constancy (we restrict to movements parallel to the image plane) states that $Z(x(t), y(t), t)$ remains constant over time and hence [71]

$$\frac{dZ}{dt} = 0 \quad \Rightarrow \quad \frac{\partial Z}{\partial t} + \frac{\partial Z}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial Z}{\partial y}\frac{\partial y}{\partial t} = 0. \tag{3.28}$$

The previous equation relates the temporal partial derivative to the spatial partial derivatives of the height function and the $vx$ and $vy$ (horizontal and vertical) components of the velocity. We use this relation to regularize the integration problem taking into account that we have a estimation of the spatial variations given by the input field $(p, q)$, hence we propose to constrain the integration problem by

considering the modified energy function:

$$E[u(x,y,t),vx,vy] = \int_{\Omega\times[0\ T]} \left( |u_x - p|^2 + |u_y - q|^2 + \lambda\,|u_t + vx\,p + vy\,q|^2 \right) dxdydt. \tag{3.29}$$

Until now we assumed that $vx$ and $vy$ are (unknown) constants. The Euler Lagrange equations for the minimization problem of the energy in Eq. (3.29) are

$$\begin{cases} u_{xx} + u_{yy} + \lambda u_{tt} + \lambda(vx\,p + vy\,q)_t = p_x + q_y \\ p^T\,u_t + p^T p\,vx + p^T q\,vy = 0 \\ q^T\,u_t + q^T p\,vx + q^T q\,vy = 0 \end{cases} \tag{3.30}$$

If the size of the discrete inputs $p$ and $q$ is $H \times W \times T$, the previous set of equations can be transform into a linear (sparse) problem of the form:

$$Ax = b \tag{3.31}$$

with $A \in \mathbb{R}^{(HWT+2)\times(HWT+2)}$ and $x \in \mathbb{R}^{HWT+2\times 1}$ (which contains the $u$, $vx$ and $vy$ unknowns). Implementation details can be found in Appendix D.2.7.

In practical applications, the hypothesis of a unique uniform velocity can be too restrictive. The Energy proposed in Eq. (3.30) can be extended if we assume now that $vx$ and $vy$ are not uniform i.e. $vx = vx(x,y,t)$ and $vy = vy(x,y,t)$. In that case, as many unknowns are introduced to the problem it is usual to introduce additional constraints over $vx$ and $vy$ [71] e.g.:

$$E[u(x,y,t),vx(x,y,t),vy(x,y,t)] = \\ \int_{\Omega\times[0\ T]} \left( |u_x - p|^2 + |u_y - q|^2 + \lambda\,|u_t + vx\,p + vy\,q|^2 + \mu\left( |\nabla vx|^2 + |\nabla vy|^2 \right) \right) dxdydt. \tag{3.32}$$

Of course now the regularization weight $\lambda$ and $\mu$ must be set and will depend on the level of noise and the intrinsic characteristics of the scene. The Euler Lagrange equations for the functional described in Eq. (3.32) are (see App. C for more details),

$$\begin{cases} u_{xx} + u_{yy} + \lambda u_{tt} + \lambda(vx\,p + vy\,q)_t = p_x + q_y \\ \lambda p\,(u_t + p\,vx + q\,vy) - \mu\Delta vx = 0 \\ \lambda q\,(u_t + p\,vx + q\,vy) - \mu\Delta vy = 0 \end{cases} \tag{3.33}$$

The previous set of equations can be again transformed into a linear (sparse) problem of the form:

$$Ax = b \tag{3.34}$$

with $A \in \mathbb{R}^{(3HWT)\times(3HWT)}$ and $x \in \mathbb{R}^{3HWT\times 1}$ (which contains the $u$, $vx$ and $vy$ unknowns). Implementation details can be found in Appendix D.2.7.

### 3.3.3. Evaluation

We will start the evaluation with a very simple synthetic example, which was generated by translating the *Ramp Peaks* surface -evaluated in the previous section- with a constant speed $\vec{v}_0$. The gradient field was computed for this new (dynamic) test surface and we added Gaussian noise and outliers to each frame. We now have a dynamic non-integrable gradient field given by $(p(x,y,t), q(x,y,t))$ as illustrated in Fig. 3.5.

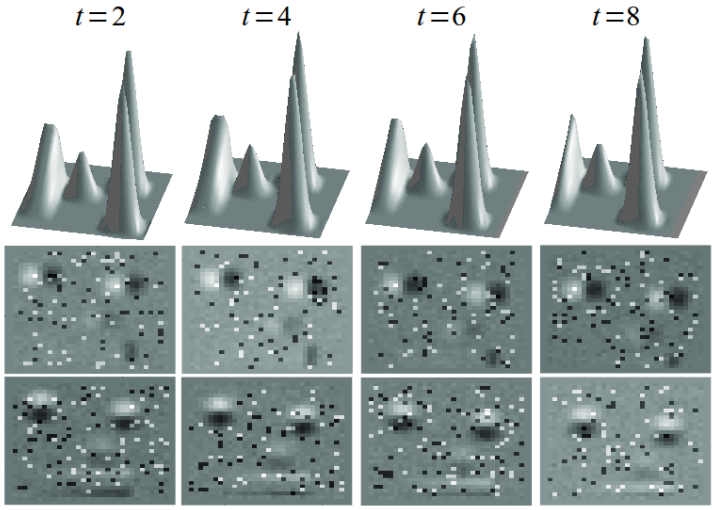


Figure 3.5: First row shows some frames of the test surface. Second row shows the first component of the input (noisy) gradient field ($p(x,y,t)$), and the third row shows the second component of the input (noisy) gradient field ($q(x,y,t)$).

In order to evaluate and understand the main differences between the presented approaches, we will compare five different reconstructions. Firstly we reconstruct the surface by integrating independently each frame of the sequence $(p(t), q(t))$

following the standard LS and AD approaches presented in Sects. 3.2.1 and 3.2.5; we named these reconstructions $Z_{LS0}$ and $Z_{AD0}$ respectively. Secondly we integrate the whole gradient field imposing the temporal regularization described in Sec. 3.3.1 for LS and AD approaches; we named these reconstructions $Z_{LSI}$ and $Z_{ADI}$ respectively. Finally, we integrate the dynamic gradient field restricting objects movement as explained in Sec. 3.3.2. We explore three different strategies; (i) we set the velocity coefficients to the real values $(vx, vy) = \vec{v}_0$; (ii) we estimated $vx$ and $vy$ from the input gradient field (assuming they are scalars) by solving Eqs. (3.30); and (iii), we assume arbitrary velocities -that may or may not vary with $x$, $y$ and $t$- and we compute $vx(x, y, t)$, $vy(x, y, t)$ and $Z(x, y, t)$ from the input gradient field solving Eqs. (3.33).[4] We named these last three reconstructions as $Z_{II1}$, $Z_{II2}$ and $Z_{II3}$ respectively. Table 3.2 summarize the approaches evaluated.

| | |
|---|---|
| $Z_{LS0}$ | Each frame is integrated independently following standard 2D LS approach. |
| $Z_{AD0}$ | Each frame is integrated independently following standard 2D AD approach. |
| $Z_{LSI}$ | Solution obtained by minimizing Eq. (3.20). |
| $Z_{ADI}$ | Solution obtained by minimizing Eq. (3.21). |
| $Z_{II1}$ | Solution obtained by minimizing Eq. (3.29), but instead of considering $vx$ and $vy$ as unknowns, they are set to the real values. This reconstruction is preformed mainly to check and test the two following cases. |
| $Z_{II2}$ | Solution obtained by minimizing Eq. (3.29) (unknown but uniform velocity) . |
| $Z_{II3}$ | Solution obtained by minimizing Eq. (3.32) (unknown and arbitrary velocity). |

Table 3.2: Summary of the compared reconstructions.

Figure 3.6 shows the MSE obtained for the different reconstructions approaches when varying the regularization weight $\lambda$. In Fig. 3.7 we show some frames of the reconstructed surfaces using the optimal value of $\lambda$ for each integration strategies. Interesting conclusions can be extracted from these results: (i) The introduction of any of the proposed temporal regularization to the standard LS approach reduces the integration error, and outperform the results obtained by processing each

[4] In the experiments we set $\mu = 0{,}4\lambda$ which we found is a reasonable value for the evaluated experiments.

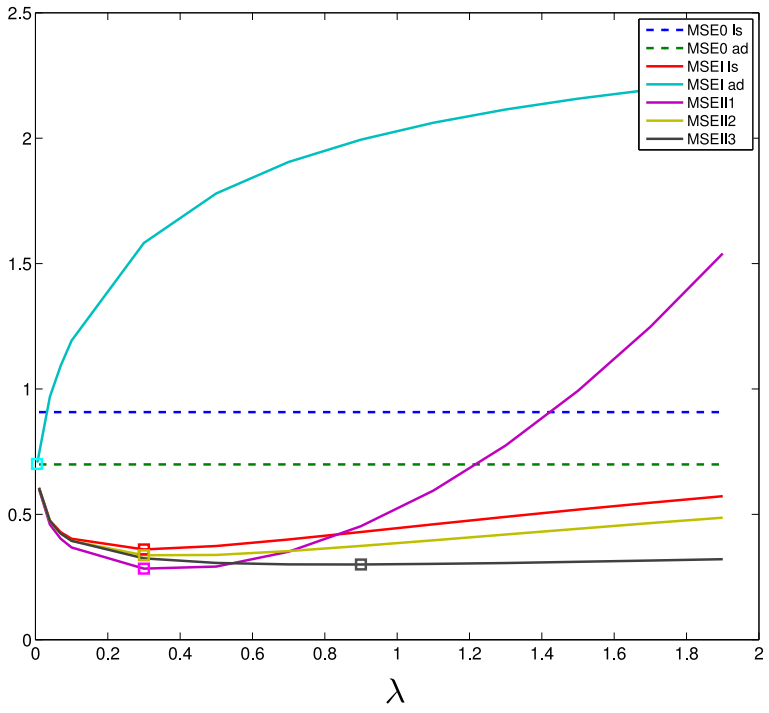


Figure 3.6: Mean Squared Error for the different reconstructions: $Z_{LS0}$, $Z_{AD0}$, $Z_{LSI}$, $Z_{ADI}$, $Z_{ZII1}$, $Z_{II2}$, $Z_{II3}$ as a function of the regularization weight $\lambda$. Case of constant and uniform velocity.

frame independently. (ii) In this example, if we introduce a temporal regularization over AD approaches, the results seems to be less accurate than using AD method for each frame independently (for instance, compare in Fig. 3.6 the green dashed line with the solid cyan curve). We think that one possible explanation for this, is that AD approach already constrains the problem (by the use of a local tensor $D$ that guides the integration paths). It is possible that when we add an additional regularization to the problem it becomes overconstrained. (iii) In this experiment we obtained slightly better results constraining objects movements (in comparison of constraining scene variations). This is an expected result as we generate this synthetic example by translating with fixed velocity the rigid Ramp Peaks surface, and so the hypothesis that holds is the one that assumes a continuous objects movement and not the one that assume a continuous image variation. (iv) If we

compare the results obtained by considering $vx$ and $vy$ as the true known parameters with respect to those obtained by estimating $vx$ and $vy$ as scalar unknowns, we obtain similar results (e.g. compare yellow and violet solid curves in Fig. 3.6). This verifies that it is possible to estimate the velocity coefficients from the input data (this is necessary because in practical problems we do not know the true $vx$ and $vy$ values). (v) If we compare the results obtained by assuming that $vx$ and $vy$ are scalar with respect to the case when we assume that they are arbitrary fields, we can see that they are very similar. These is also a consistent result because the data was generated by translating a rigid surface with a unique velocity pattern and hence assuming that the velocity is a constant vector or a vector field should not lead to significant differences. In the next experiment, we generate a new example with non-uniform velocity in which we can evaluate further this aspect.

In a second experiment, we generate a new synthetic dynamic scene by rotating with constant angular velocity the Ramp Peaks surface. The new generated surface and the noisy input gradient field is illustrated in Fig. 3.8. Figure 3.9 shows the MSE obtained for the different reconstruction approaches when varying the regularization weight $\lambda$. In Fig. 3.10 we show some frames of the reconstructed surfaces using the optimal value of $\lambda$ for each integration strategies. Most of the conclusions extracted for the first set of experiments can be confirmed in this second example. In addition we can see how the difference increase in this case if we compare the reconstructions $Z_{II2}$ (where a scalar $vx$ and $vy$ is estimated for the whole sequence) and $Z_{II3}$ (where arbitrary and variable velocities are allowed). This difference was expected because in this new example, as the scene rotates, different points will have different velocities and hence the hypothesis of constant velocity does not hold anymore.

The numerical results are summarized in Table 3.3 for the two previously presented synthetic examples.

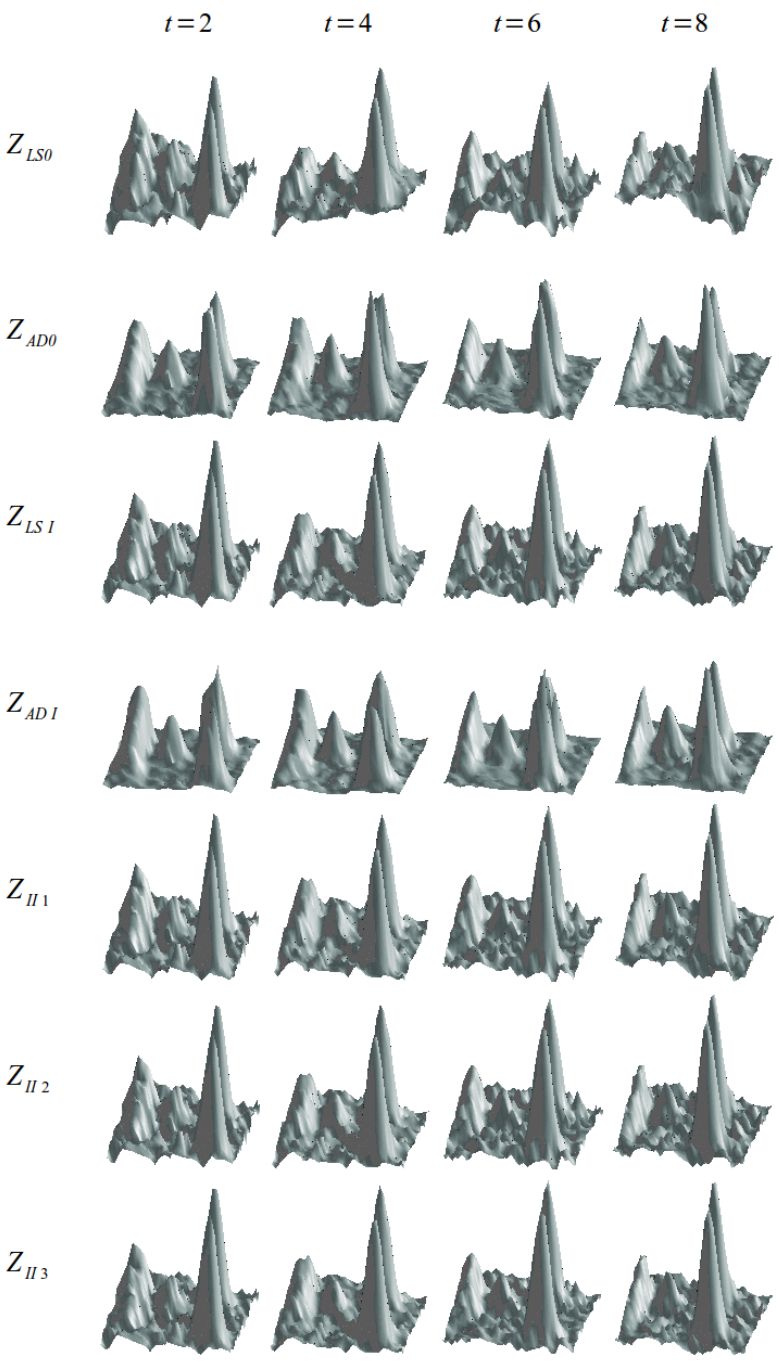


Figure 3.7: Frames of the surface retrieved by each strategy using the value of $\lambda$ that minimizes the MSE.

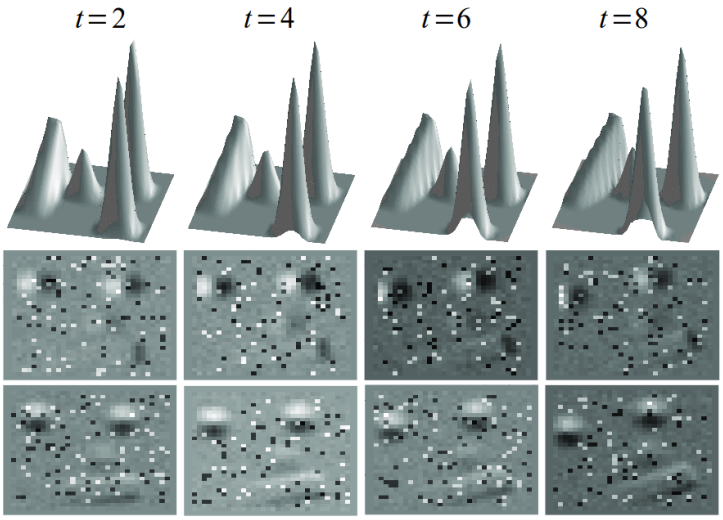


Figure 3.8: First row shows some frames of the (rotated) test surface. Second row shows the first component of the input (noisy) gradient field ($p(x, y, t)$), and the third row shows the second component of the input (noisy) gradient field ($q(x, y, t)$).

| Experiment with constant velocity | | | | | | | |
|---|---|---|---|---|---|---|---|
| | $Z_{LS0}$ | $Z_{AD0}$ | $Z_{LSI}$ | $Z_{ADI}$ | $Z_{II1}$ | $Z_{II2}$ | $Z_{II3}$ |
| min. MSE | 0,91 | 0,70 | 0,36 | 0,82 | **0,28** | 0,34 | 0,30 |
| Opt. $\lambda$ | – | – | 0,30 | 0,01 | 0,30 | 0,30 | 0,90 |
| Mean Ex. Time | **0,08s** | 7,89$s$ | 0,71$s$ | 0,99$s$ | 0,97$s$ | 12,03$s$ | 34,36$s$ |
| Experiment with constant angular velocity | | | | | | | |
| | $Z_{LS0}$ | $Z_{AD0}$ | $Z_{LSI}$ | $Z_{ADI}$ | $Z_{II1}$ | $Z_{II2}$ | $Z_{II3}$ |
| min. MSE | 0,90 | 0,86 | 0,35 | 0,94 | – | 0,35 | **0,30** |
| Opt. $\lambda$ | – | – | 0,40 | 0,01 | – | 0,40 | 1,00 |
| Mean Ex. Time | **0,07s** | 7,79$s$ | 0,74$s$ | 0,99$s$ | – | 11,51$s$ | 32,69$s$ |

Table 3.3: Minimum MSE, optimal value of $\lambda$ and mean execution time using different regularization techniques.

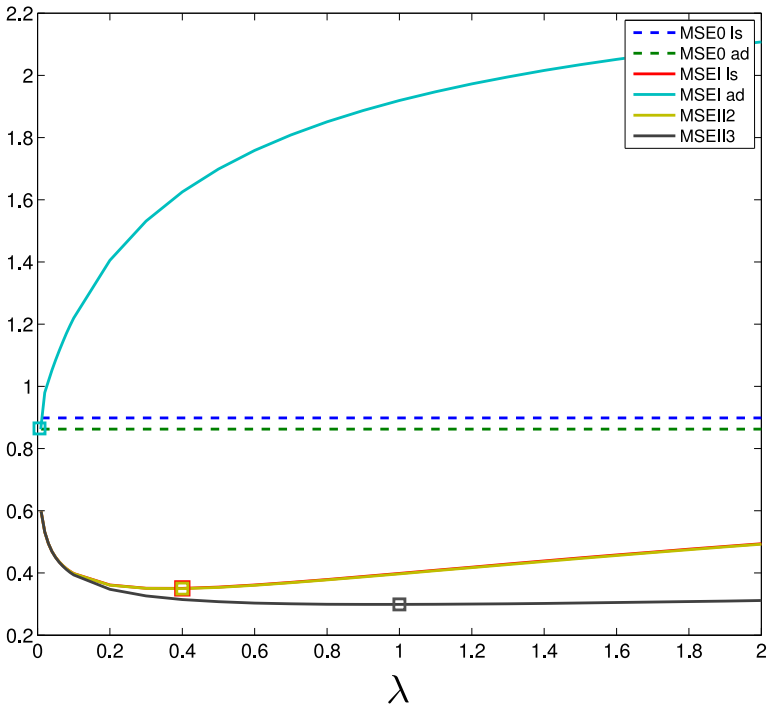


Figure 3.9: Mean Squared Error for the different reconstructions of the rotated surface: $Z_{LS0}$, $Z_{AD0}$, $Z_{LSI}$, $Z_{ADI}$, $Z_{ZII1}$, $Z_{II2}$, $Z_{II3}$ as a function of the regularization weight $\lambda$. Case of constant angular speed.

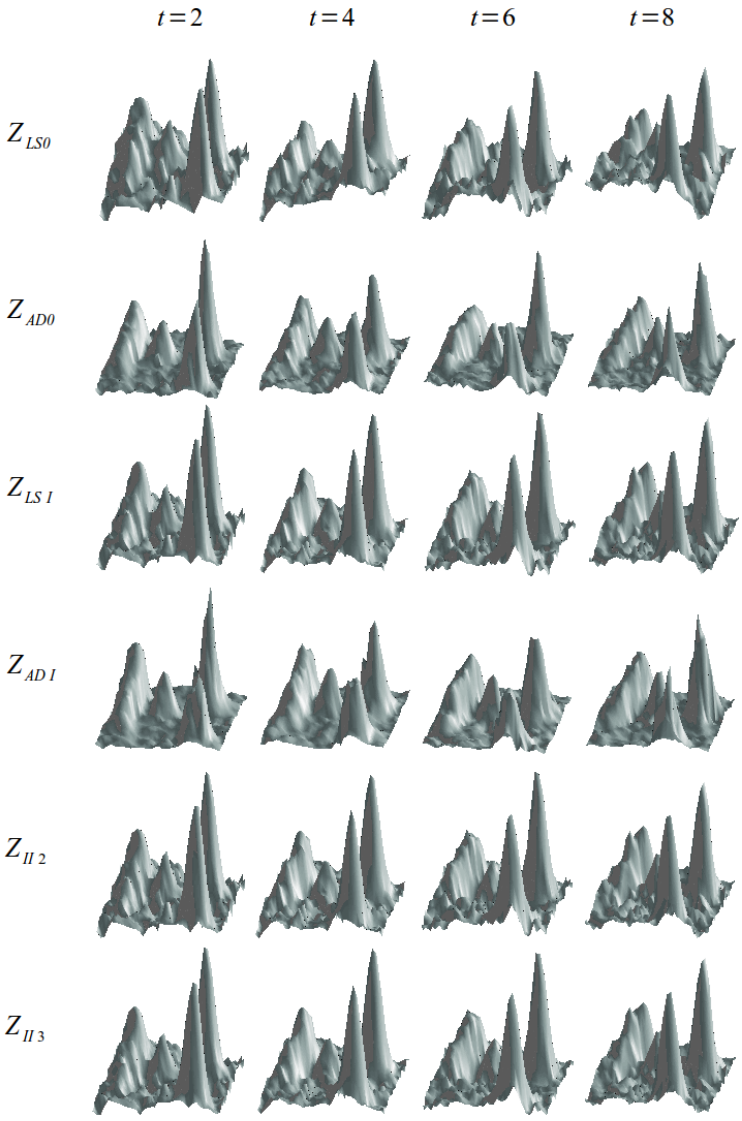


Figure 3.10: Frames of the rotated surface retrieved by each strategy using the value of $\lambda$ that minimizes the MSE.

Additional experiments with complex and real data.

We conclude this section presenting additional experiments over (a) real scenes extracted from -publicly available- surveillance records (Video 1-5) [80]; and (b) over a retrieved dynamic gradient field obtained by following the 3D scanning approach (D3D) presented in Chapter 2. In the first case, we computed the sequences of $x$ and $y$ partial derivatives of the surveillance records and we add to them Gaussian noise (with $\sigma = 0{,}1g$, $g$ = maximum gradient magnitude) and uniformly distributed outliers. In the second case, we integrate a retrieved gradient field (obtained by D3D) which already has empirical errors and outliers due to surface discontinuities (as discussed in Sec. 2.2.1).

Figures 3.11-3.15 show a set of frames of the achieved results obtained after integrating the noisy gradient fields with some of the previously presented integration approaches. As we can see, temporal regularization reduces the final level of noise without introducing distortion on edges' definition or ghost effects on movings parts of scenes.

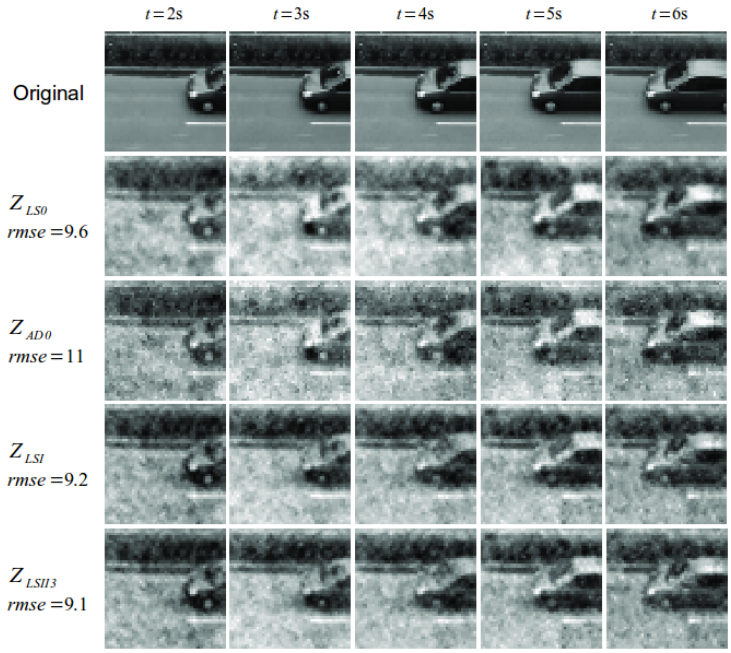


Figure 3.11: Results for Video 1.

In the second experiments, we integrated a dynamic depth's gradient field

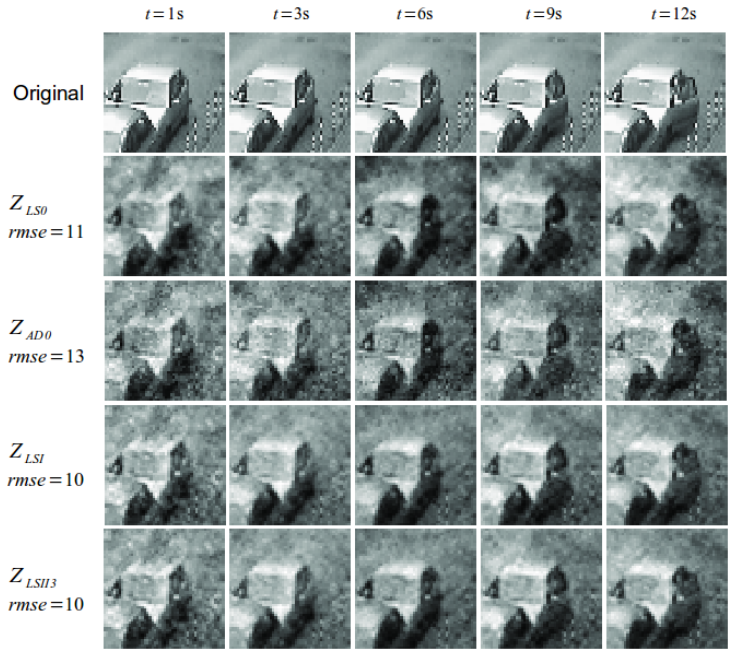


Figure 3.12: Results for Video 2.

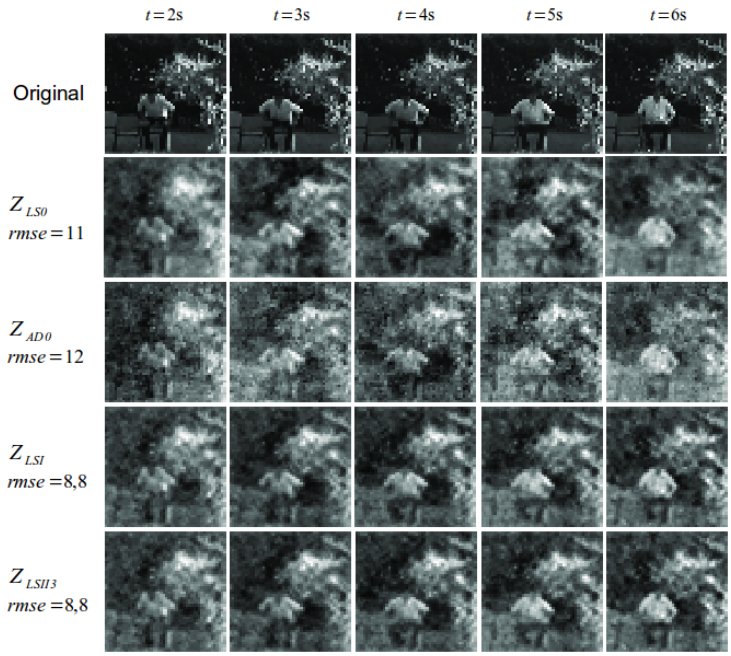


Figure 3.13: Results for Video 3.

obtained by measuring the deformation of a pattern projected over the target surface. Figure 3.16 shows (left images) some frames of the input images and

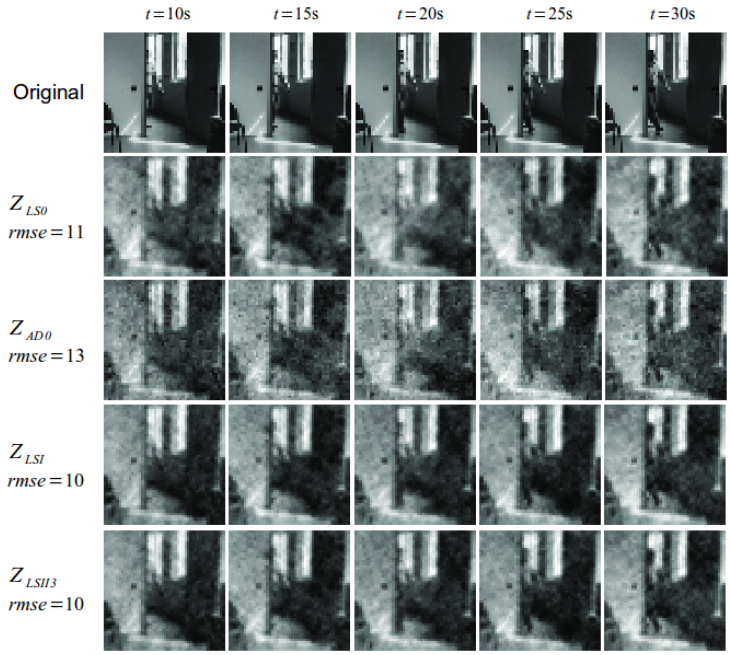


Figure 3.14: Results for Video 4.

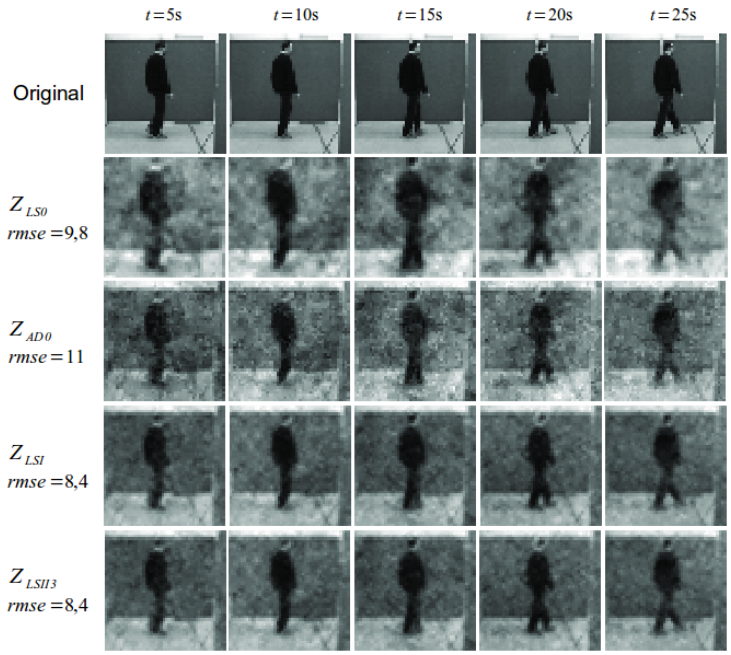


Figure 3.15: Results for Video 5.

(right images) some examples of the retrieved gradient field.

Figure 3.17 illustrates the reconstructed 3D surfaces from the input gradient

map following the described regularization techniques. As can be seen by taking advantage of the temporal relationship of the data, one can reduce noise (see e.g. the zoomed area of the forehead) and obtain smoother surfaces without sacrificing facial expression details.

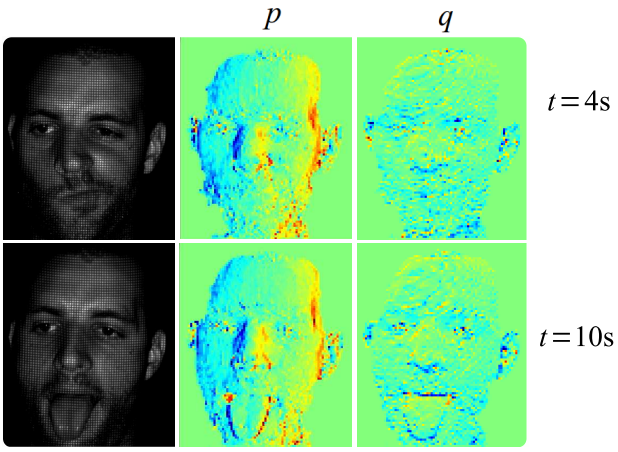


Figure 3.16: Examples of frames of the input video (left) and the estimated gradient field from them.

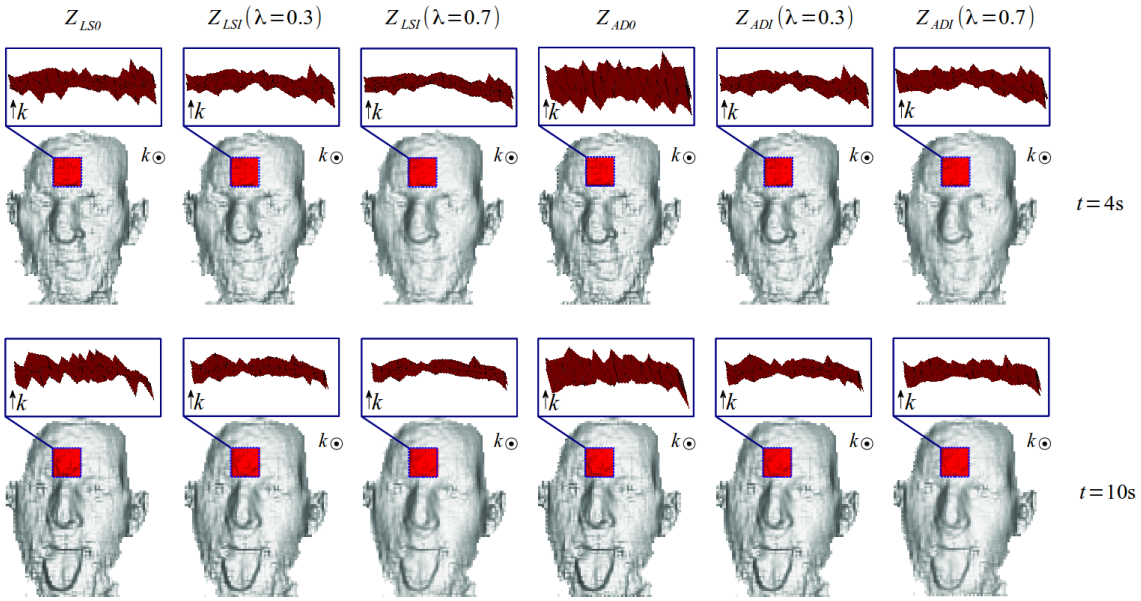


Figure 3.17: Some frames of the obtained 3D surfaces.

## 3.4. Conclusions

In this chapter we reviewed a selected set of the most popular integration techniques and we extended some of them for the case of dynamic gradient fields. We study two possible strategies to regularize the space of solutions depending on the temporal nature of the scene.

As we saw in Sec. 3.2 many different approaches have been developed for surface reconstruction from a gradient field. The simplest and fastest techniques are LS and FC methods; besides their simplicity, for particular problems they may let to an accurate solution and therefore they should always be tried first. When LS or FC methods do not achieve successful results it may be worth considering more complex strategies as: AD, M and $L_1$-min. Recall that AD and M methods handle outliers and noise by introducing weights to the standard $L_2$ minimization approach, while $L_1$ uses a penalty norm with linear (instead of quadratic) growth. As we saw at the end of Sec. 3.2 complex techniques require orders of magnitude more computation time and their implementations are much more difficult to code. An interesting result that we showed in Sec. 3.2.6 is that the difference between the performance of integration techniques is significantly reduced if we pre-process the input gradient field with a non-linear median filter. This should be taken into account when facing a practical integration problem, because a simple preprocessing denoising step plus a simple integration algorithm can save much computational cost compared with any of the robust approaches.

In chapter 2 we presented a novel projection technique, capable of provide us with dynamic gradient fields of the form $(p(x, y, t), q(x, y, t))$. In this chapter we show how the temporal relation between frames can be exploited to guide the integration process, and obtain robust integration techniques. We present two different regularization schemes; In the first case, we assumed that the image of the scene varies (in some temporal order) continuously. In the second case we focus in cases in which the material objects present in the scene have regular velocity patterns. As we saw in the presented experiments there are not general rules concerning methods' performance and we must explore for each particular

problem which of the hypothesis is more feasible. In addition, in practical examples the regularization weights must be set (which depends on the amount of noise and outliers present in the input gradient field).

# Chapter 4

# Differential-3D applied to face recognition

In this chapter we will show how the proposed D3D scanning technique can be applied to face recognition. As we achieve a fast gradient field representation of the scene, D3D can also be applied to other common problems on the field of computer vision and pattern recognition, particularly those related to face analysis e.g. gesture interpretation, pose and mood recognition.

## 4.1. Introduction

Face recognition is one of the most popular and challenging problems in the field of pattern recognition and computer vision [1]. It has many applications such as security control and prevention, medical and biometrical analysis or gesture understanding. In the last decade, lot of research included three-dimensional (3D) face information to improve recognition rates and make the methods more robust to pose, gesture and illumination variations [13]. Bronstein et al. [15] used 3D facial scanners and achieved a robust recognition framework by modeling facial expressions as surface isometries, and constructing an expression-invariant face representation using the canonical forms approach. Chang et al. [17] was one of

the first works that combined scores obtained from matching multiple overlapping regions around the nose. A similar method was proposed by Faltemier et al. [32] where 28 different regions around the face were selected and then a score-based fusion approach was followed. Kakadiaris et al. [58] presented a 3D deformable model approach where the face was parametrized by the annotated face model (AFM). Mahoor et al. [66] used the principal curvature to represent the face image as a 3D binary image, then Hausdorff distance and iterative closest points (ICP) [10] were used for matching. More recently, Li et al. [63] used facial curves to form a rejection classifier and produce a facial deformation mapping from which adaptively select different face regions for matching. In Y. Zhang et al. [103] each 3D facial surface was mapped onto a 2D lattice that represents local 3D geometrical or textural properties, after this traditional 2D face recognition techniques were applied. Recently, Lei et al. [61] presented a 3D face recognition approach based on low-level geometric features collected from the eyes-forehead and nose regions, after that a Support Vector Machine (SVM) algorithm was used to separate different subjects' representations.

In the present chapter, a 2D+3D face recognition approach is proposed. Instead of using a 3D scanner we acquire a standard color photo of the face while a rectangular pattern is been projected. We will show that from this unique image, it is possible to extract low-level geometrical features without the explicit 3D reconstruction, as well as texture information. To handle expression variations and occlusions that may occur by wearing a scarf or a bonnet, we will extract information just from the eyes-forehead and nose regions which tend to be less influenced by facial expressions, as discussed by Lei et al. [61]. To that end, in this chapter we will also describe how eyes and nose location can be automatically find following a simple but robust approach.

The rest of this chapter is organized as follows. Section 4.2 starts with a general description of the proposed framework and then each specific step is detailed. Section 4.3 presents experimental results, firstly using the Texas 3D Face Recognition Database [44], and secondly, using our own set of images where all the steps of the

proposed framework are involved. In section 4.4 we present the conclusions and discuss future work.

## 4.2. Description of the proposed framework

We will start by presenting an overview of the proposed framework, highlighting the main steps and properties. After that, each individual step will be described in detail.

The main steps of the proposed technique can be identified as: (i) acquisition; (ii) extraction of geometric and texture information; (iii) localization of the eyes and nose position; (iv) extraction of features from the rigid and semi-rigid regions of the face, and finally; (v) train/classify.

Unlike most of the work published in the literature, to perform 2D+3D face recognition we are not using a commercial scanner plus a standard photo of the subject. Instead, we propose an acquisition setup in which fringes are projected over the face of the test subject while a standard photo is taken. As we have demonstrated in Chapter 2, this single picture allows us to extract both texture and 3D geometrical information of the face [20, 25].

Once the texture and the 3D geometrical information is retrieved, we perform a curvature analysis to localize nose tip and eyes corners candidates points [23]. After this we implemented a *message passing* approach to remove those false positive nose and eyes candidates.

From eyes and nose positions, we extract two different areas of the face. The first is a rectangular region that contains both eyes, and the second one is a trapezoidal region that contains the nose. As it was demonstrated in several works, these are the more rigid face portions, and also the less affected by facial expressions (see e.g. [13, 61] and reference therein).

The next step consists in defining features using the information available in the selected portions of the face. We analyze features computed from the geometrical 3D information and from the texture information. By measuring the projected

pattern deformation, we can compute the partial derivatives of the face depth, then we condensate that geometrical 3D information in histograms of fixed dimension. For the texture information, we compute a Local Binary Pattern (LBP) transformation, which is a simple yet very efficient texture descriptor [5].

Until now, we briefly described how we can move from an acquired image to a descriptor vector. Once this is achieved, one can think of an acquired image as a point in a $m$-dimensional space ($m$ is of course the length of the descriptor vector). Using a training dataset, i.e. a set of acquired images in which we know who is the present subject, we can build the *best* boundaries in the $m$-dimensional space. This means, looking for each person in the database those boundaries that separates with less error and more distance the feature vectors that belong to the same person from the other ones. To that end, we use Support Vector Machine (SVM) algorithm, training one classifier for each subject in the database (this is commonly known as one-vs-all approach). Once this classifiers are trained, one can define a partition of the $m$-dimensional space and *classify* new samples by observing its position on the feature space.

Figure 4.1 illustrates the main steps of the proposed framework and the kind of data processed at each step. Before presenting the experimental results, additional explanations and details of each step are covered in the following subsections.

### 4.2.1. Acquisition and preprocessing

Unlike most of the approaches that perform 3D face recognition, the required hardware of the proposed method is a standard camera, a source of light (e.g. a led or flashlight), a rectangular binary amplitude pattern and a lens to project the image of the pattern over the face being captured.

The acquired (2D) image is filtered in the Fourier domain, and three new images are synthesized. Firstly, by keeping the low frequencies of the input image, a texture image is estimated ($I_0(x, y)$). Then, by filtering the acquired image around the first peak in the $x$ and $y$ directions, images with deformed vertical ($I_v(x, y)$) and horizontal ($I_h(x, y)$) fringes respectively are obtained. As demonstrated in

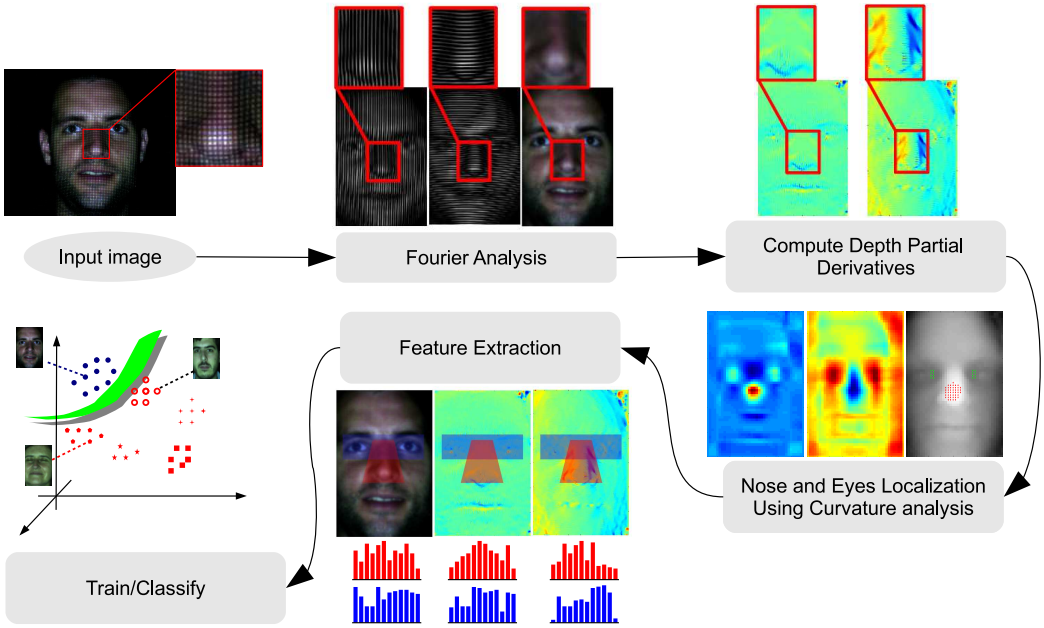


Figure 4.1: Illustration of the main steps of the proposed framework. Note: due to image resolution artifacts can appear, the full images, the code and a demo can be found at www.fing.edu.uy/~matiasdm.

Chapter 2, by measuring the deformation of these fringes, it is possible to estimate the $x$ and $y$ partial derivatives of the surface depth.

Once these three images are synthesized, the texture image is used to segment the face from the background. In our experiments, we used a black background and therefore the segmentations was trivially performed by thresholding using e.g. Otsu's method [75, 86]. For experiments in uncontrolled conditions, more robust approaches may be followed (e.g. segmenting those pixels with skin color in the HSV space) while keeping unchanged the following steps of the proposed method.

### 4.2.2. Nose and eyes localization

To achieve a robust localization of the eyes and nose position, we follow a curvature approach (see e.g. [23] and references therein). This methods is very efficient for detecting those possible nose and eyes candidates, despite that, it is important to solve the problem of removing false positive candidates. To solve this

issue we will follow a message passing methodology; the main idea behind it is to use nose candidates to estimate those regions where is expected to find the eyes, and vice versa.

### Find nose and eyes candidates

Let $S$ be the surface defined by a twice differentiable real valued function $D : \Omega \to \mathbb{R}$, defined on an open set $\Omega \subseteq \mathbb{R}^2$:

$$S = \{(x, y, z) \,|\, (x, y) \in \Omega;\ z \in \mathbb{R}; D(x, y) = z\}. \tag{4.1}$$

For every point $(x, y, D(x, y)) \in S$ we consider two different curvature measures, the Mean ($H$) and the Gaussian ($K$) curvature defined as [18, 27]:

$$H(x, y) = \frac{(1 + D_y^2)\, D_{xx} - 2 D_x D_y D_{xy} + (1 + D_x^2)\, D_{yy}}{2\left(1 + D_x^2 + D_y^2\right)^{3/2}}, \tag{4.2}$$

$$K(x, y) = \frac{D_{xx} D_{yy} - D_{xy}^2}{\left(1 + D_x^2 + D_y^2\right)^2}. \tag{4.3}$$

Following the procedure described in Sec. 2.2 we compute the first derivatives of the scene depth. After that, to calculate $H(x, y)$ and $K(x, y)$ it is only necessary to compute the second order derivatives ($D_{xx}$, $D_{yy}$ and $D_{xy}$) e.g. using finite differences (i.e. $D_{xx}[i, j] = (D_x[i, j+1] - D_x[i, j-1])/2$) or any equivalent numerical method).

Once the Mean and Gaussian curvatures are computed, it is possible to classify the different areas of the face according to its shape [11]. Depending on $H$ and $K$ values, points on the surface are classified following the rule shown in Table 4.1. To remove smooth regions from the areas of interest a thresholding approach was

| | $K < 0$ | $K = 0$ | $K > 0$ |
|---|---|---|---|
| $H < 0$ | Hyperbolic concave | Cylindrical concave | Elliptical concave |
| $H = 0$ | Hyperbolic symmetric | Planar | Impossible |
| $H > 0$ | Hyperbolic convex | Cylindrical convex | Elliptical convex |

Table 4.1: $HK$ classification.

followed [18, 42], and then those points with high absolute value of $K$ and $H$ were isolated. For those points with $K > 0$ and $H > 0$ the eye candidate label was assigned, while those with $K > 0$ and $H < 0$ were selected as nose candidates.

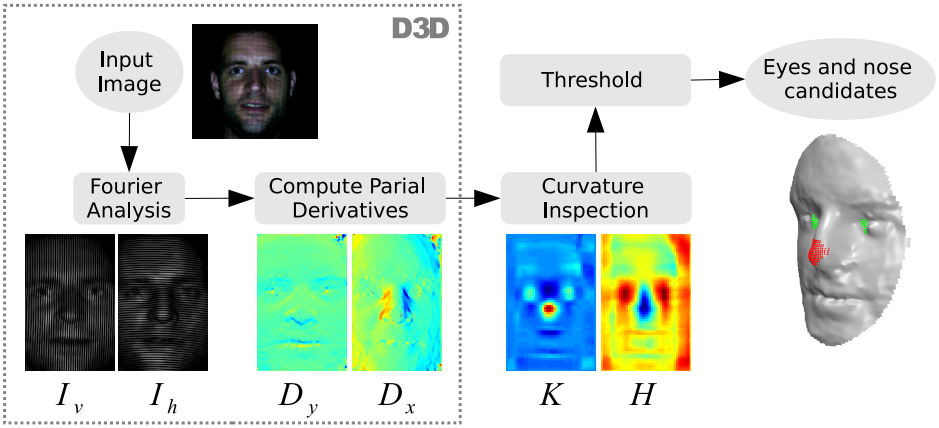


Figure 4.2: Main steps to identify nose and eyes candidates.

Figure 4.2 summarizes the main steps involved in nose and eyes candidates detection, while Fig. 4.3 illustrates the results in uncontrolled conditions.

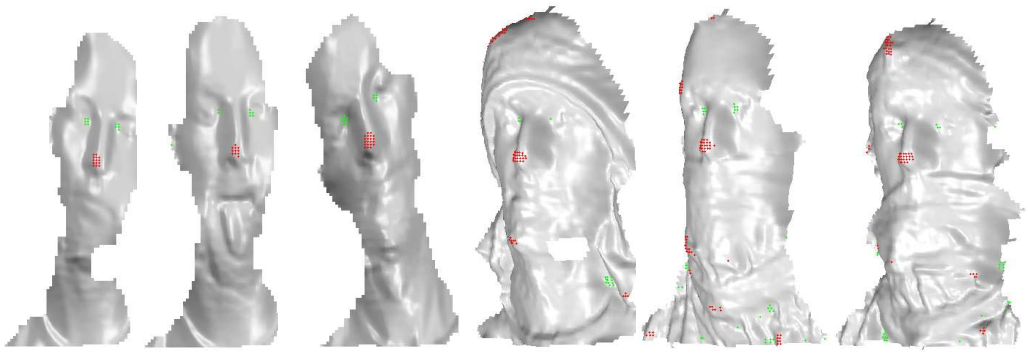

Figure 4.3: Examples of nose (red dots) and eyes (green dots) candidates point detected.

### Remove false positive detections

In order to remove possible false detections many approaches has been proposed; for example, Li and Da [63] fitted a nose template to each nose candidate. Bronstein et al. [15] impose geometric relations between the candidates (e.g. that

the nose apex is located between the two eyes, above the nose tip and within certain distance intervals). In the work of Faltemier et al. [32] the nose tip is found as the highest $z$ value after the face surface is aligned using ICP to a template face. Chang et al. [17] impose that the eyes regions must present similar $y$ and $z$ value, and that the nose tip is found starting between the eyes landmark and moving *down* (i.e. along the $x$ direction) along the face. Another example is the work of Colombo et al. [18] in which the idea is to look from all the nose and eyes candidates, all the possible triangles formed by one nose candidate and two eyes candidates. Then, each triangle is described by the distances between the three regions composing it; finally, triangles with abnormal distances are rejected.

While some of the previously described methods make hypothesis regarding the position or orientation of the face (i.e. assuming than the nose is down the two eyes), others are computationally expensive (i.e. requiring ICP registration or checking all possible combination of nose and eyes triangles). But what these methods have in common is that they use the explicit three-dimensional representation of the face, in the present work we aim to bypass the 3D mapping of the face and instead take advantage of the information available in the Depth gradient map of the face. To this end we will introduce a simple false positive rejection approach -that do not need the explicit 3D mapping of the face- and instead restricts the distance (in the image plane) between the eyes, and between the eyes and the nose.

In first place, we estimate the distance between the eyes and between each eye and the nose tip using a training set. We approximate each distance distributions by a Gaussian distributions i.e.

$$d_{ee}(u) = \frac{1}{\sigma_{ee}\sqrt{2\pi}}\, e^{-\left(\frac{(u-\mu_{ee})^2}{2\sigma_{ee}^2}\right)} \tag{4.4}$$

$$d_{ne}(u) = \frac{1}{\sigma_{ne}\sqrt{2\pi}}\, e^{-\left(\frac{(u-\mu_{ne})^2}{2\sigma_{ne}^2}\right)}. \tag{4.5}$$

where $d_{ee}$ and $d_{ne}$ denote the distance between eyes and the distance between the nose and each eye, respectively. The mean and variance parameters ($\mu$ and $\sigma^2$) were estimated by fitting the distributions given by Eqs. (4.4) and (4.5) to the distributions obtained from a given training set.

Once distance distributions are obtained, they are used to *propagate* where it is feasible to find the nose/eyes considering each of the others nose and eyes candidates. To this end, two kernels with Gaussian section (defined by Eqs. (4.4) and (4.5) respectively), and rotational symmetry are considered

$$\mathcal{K}_{ne}(x,y) = d_{ne}\left(\sqrt{x^2+y^2}\right) \quad \text{and} \quad \mathcal{K}_{ee}(x,y) = d_{ee}\left(\sqrt{x^2+y^2}\right). \tag{4.6}$$

The next step consists in estimating in which areas is more suitable to have the nose and eyes candidates so we can keep the set of points with higher probability to be the true ones. For this we follow four basic steps -illustrated in Fig. 4.4-:

1. Estimate initial *nose-likelihood* ($\mathcal{N}$) and *eye-likelihood* ($\mathcal{E}$) functions:

$$\mathcal{N}_0(x,y) = \frac{1}{\#e}\sum_{i=1}^{\#e}\delta_{x_i,y_i} * \mathcal{K}_{ne}(x,y) \tag{4.7}$$

$$\mathcal{E}_0(x,y) = \left(\frac{1}{\#e}\sum_{i=1}^{\#e}\delta_{x_i,y_i} * \mathcal{K}_{ee}(x,y)\right) \cdot \left(\frac{1}{\#n}\sum_{j=1}^{\#n}\delta_{x_j,y_j} * \mathcal{K}_{ne}(x,y)\right) \tag{4.8}$$

   where $\#e$, $\#n$ is the number of eyes and nose candidates respectively, $\delta_{x_i,y_i}$ is the Dirac delta function at each nose/eye candidate location $(x_i, y_i)$; and $\mathcal{K}_{ne/ee}$ the Kernels defined above.

2. Compute the nose-likelihood and eyes-likelihood distributions as,

$$\mathcal{N}(x,y) = \frac{1}{\#e}\sum_{i=1}^{\#e}\mathcal{E}_0(x_i,y_i)\delta_{x_i,y_i} * \mathcal{K}_{ne}(x,y) \tag{4.9}$$

$$\begin{aligned}\mathcal{E}(x,y) = &\left(\frac{1}{\#e}\sum_{i=1}^{\#e}\mathcal{E}_0(x_i,y_i)\delta_{x_i,y_i} * \mathcal{K}_{ee}(x,y)\right) \\ &\cdot \left(\frac{1}{\#n}\sum_{j=1}^{\#n}\mathcal{N}_0(x_i,y_i)\delta_{x_j,y_j} * \mathcal{K}_{ne}(x,y)\right).\end{aligned} \tag{4.10}$$

   This second step is illustrated in the third row of Fig. 4.4, the size of each nose/eye candidate point illustrates the value of $\mathcal{N}_0/\mathcal{E}_0$ and the information of each candidate will be propagated with different weight.

3. Finally, the nose/eyes candidate points with higher $\mathcal{N}/\mathcal{E}$ are kept as the true nose/eyes locations.

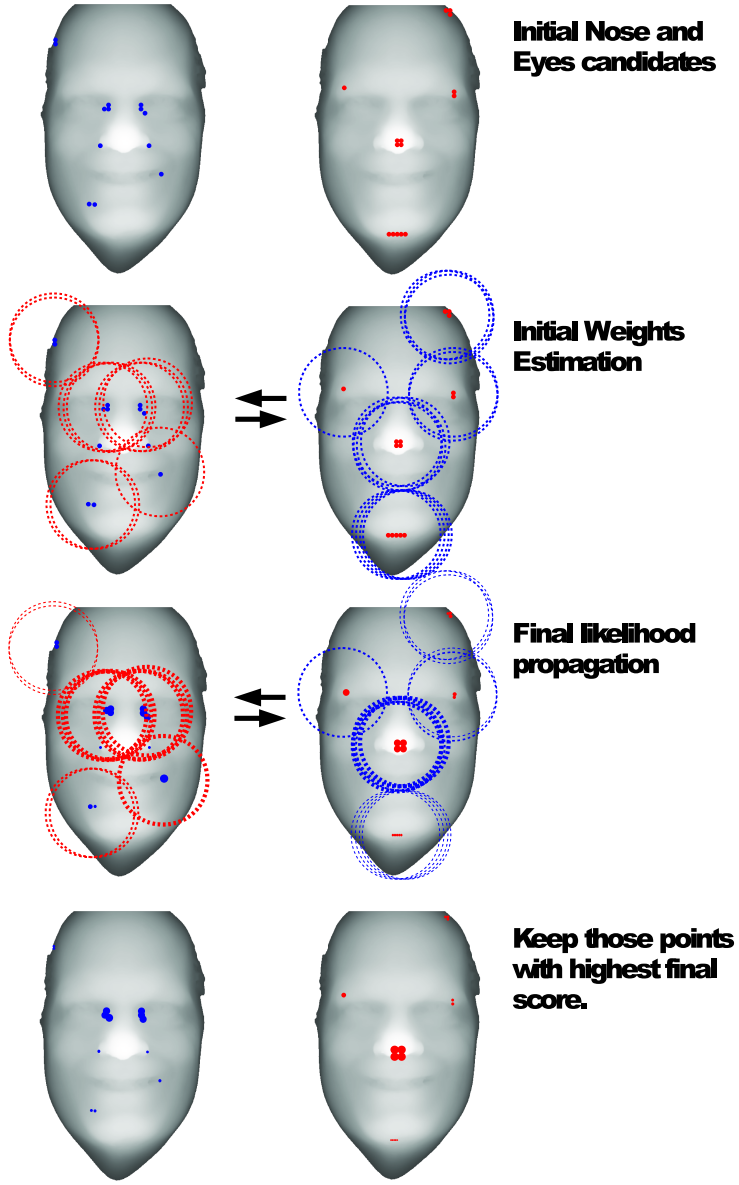


Figure 4.4: Illustration of the steps followed to remove false positive detections. The size of each candidate point illustrates the likelihood of that point at each step, i.e. the larger the size, the higher the consistency with respect to the other candidate points.

Implementation details and the code described approach are provided in Appendix D.3. Figure 4.5 shows an example of the input image (a), depth gradient field (b-c) [estimated following D3D], eyes candidate points (display as blue dots) overlapped to the computed $\mathcal{E}(x, y)$ (d), nose candidate points (display as blue dots) overlapped to the computed $\mathcal{N}(x, y)$ (e), and finally (f) the output nose and eyes locations obtained .

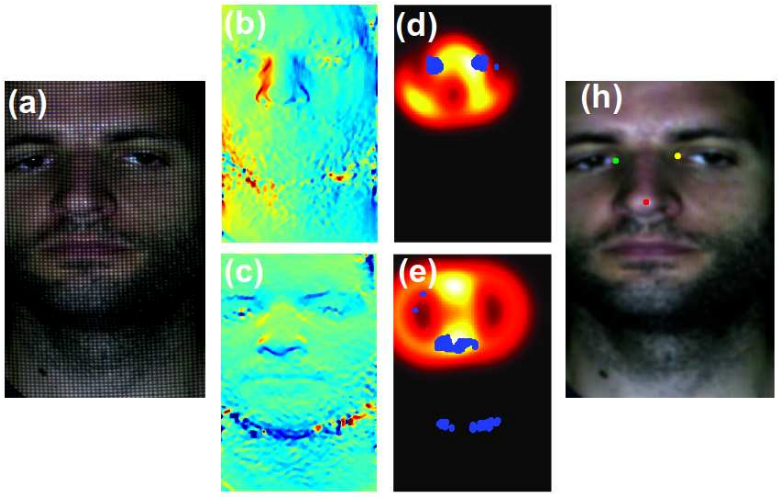


Figure 4.5: This figure illustrates the nose and eyes detection procedure. (a) Input image. (b-c) $x$ and $y$ depth partial derivatives obtained by measuring fringes' deformation. (d-e) eyes and nose candidates obtained by curvature thresholding (as blue dots) overlapped with the estimated functions $\mathcal{E}(x, y)$ and $\mathcal{N}(x, y)$ respectively. (h) Nose and eyes located in the image.

## Evaluation

Before proceeding with the description of the last steps of face recognition framework, let us briefly analyze the performance of the previously described nose and eyes detection approach. For that we will use the Texas 3D Face Recognition Database [44, 45]. This database contains 1149 pairs of high resolution, pose normalized, preprocessed, and perfectly aligned color and range images of 118 subjects. Additionally, it includes the locations of manually marked anthropometric facial fiducial points which will be used as ground truth. Examples of this database are shown in Figure 4.11. For the experiments, we split the database in two sets:

Train and Test sets. The first one was used to estimate the parameters, and the test set was used for evaluation once all the parameters were trained.

In a first experiment, we want to evaluate the robustness of the proposed methodology when we have different power of noise in the input gradient field. Figure 4.6 shows the accuracy obtained (over the test set) for different levels of noise . The noise added to the input gradient field was Gaussian with zero mean; as we can see, when the variance of noise distribution was below the 20 % of the maximum of the signal, the error in both nose and eyes detection was below the 10 %.

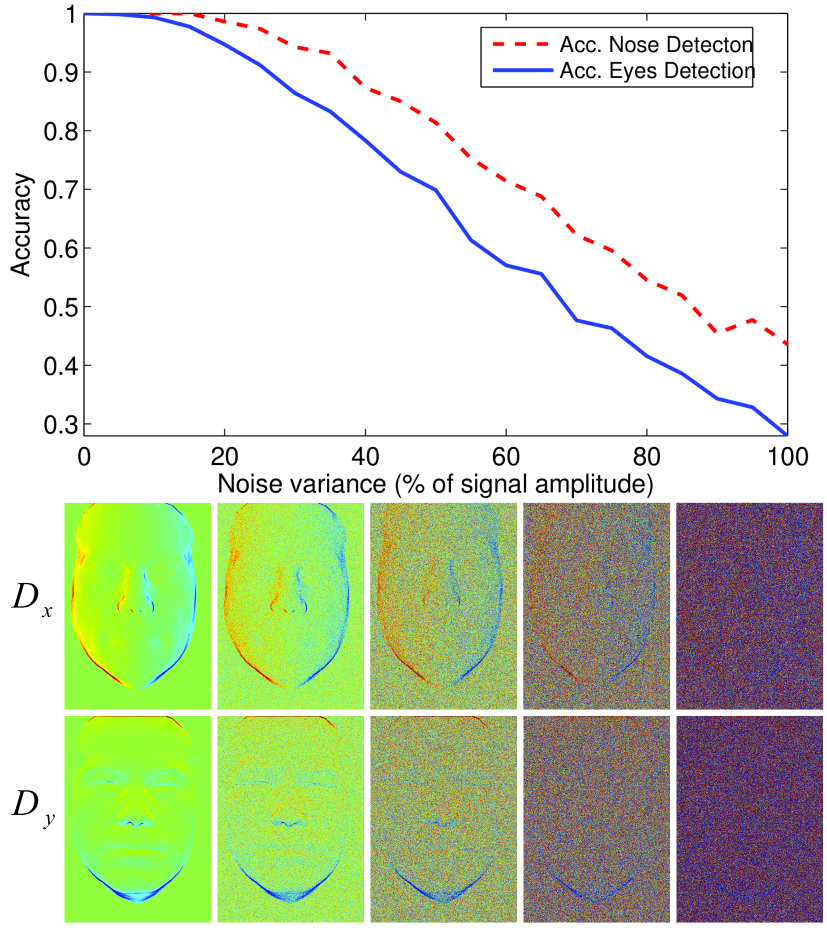


Figure 4.6: Result obtained over the test set varying the power of the noise added to the input gradient field.

In a second experiment, we want to test the robustness of the proposed tech-

nique when the faces are in a non-frontal position with respect to the camera. We define the angles $\theta_1$, $\theta_2$ and $\theta_3$ as illustrated in Figure 4.7. Recall that we are measuring the projected euclidean distance in the 3D space over the image plane, because of this it is expected that for databases with larger pose variations, $K_{ne}$ and $K_{ee}$ kernels will become wider and hence more errors should be expected[1]. A

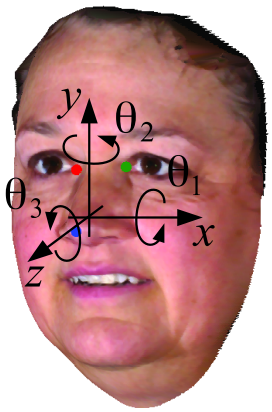


Figure 4.7: Definition of $\theta_1$, $\theta_2$ and $\theta_3$.

new dataset was generated by randomly rotating faces (of the Texas Database) and projecting them back to the image plane. Again this database was spitted in train and test sets to avoid over fitting. The test set was divided according to faces' position, hence we were able to measure the accuracy for different face orientation (i.e. different values of $\theta_1$, $\theta_2$ and $\theta_3$). Figure 4.8 shows the accuracy for test sets with faces in different positions. Accuracies on eyes and nose recognition are displayed for $|\theta_i| \in [0^o, 55^o]$ $i = 1.,3$, the range images below the $x$ axis illustrates the pose obtained with the corresponding value of $\theta_i$.

### 4.2.3. Feature extraction

Two different regions of the face will be considered for local feature extraction (as illustrated in Fig. 4.1). The first patch (named PatchA and illustrated with a

[1]In this direction we think there is room for interesting future work. For example, an improvement could be to estimate the distance in the 3D space from the distances in the image plane plus the information available in the Depth gradient field ($D_x$ and $D_y$).

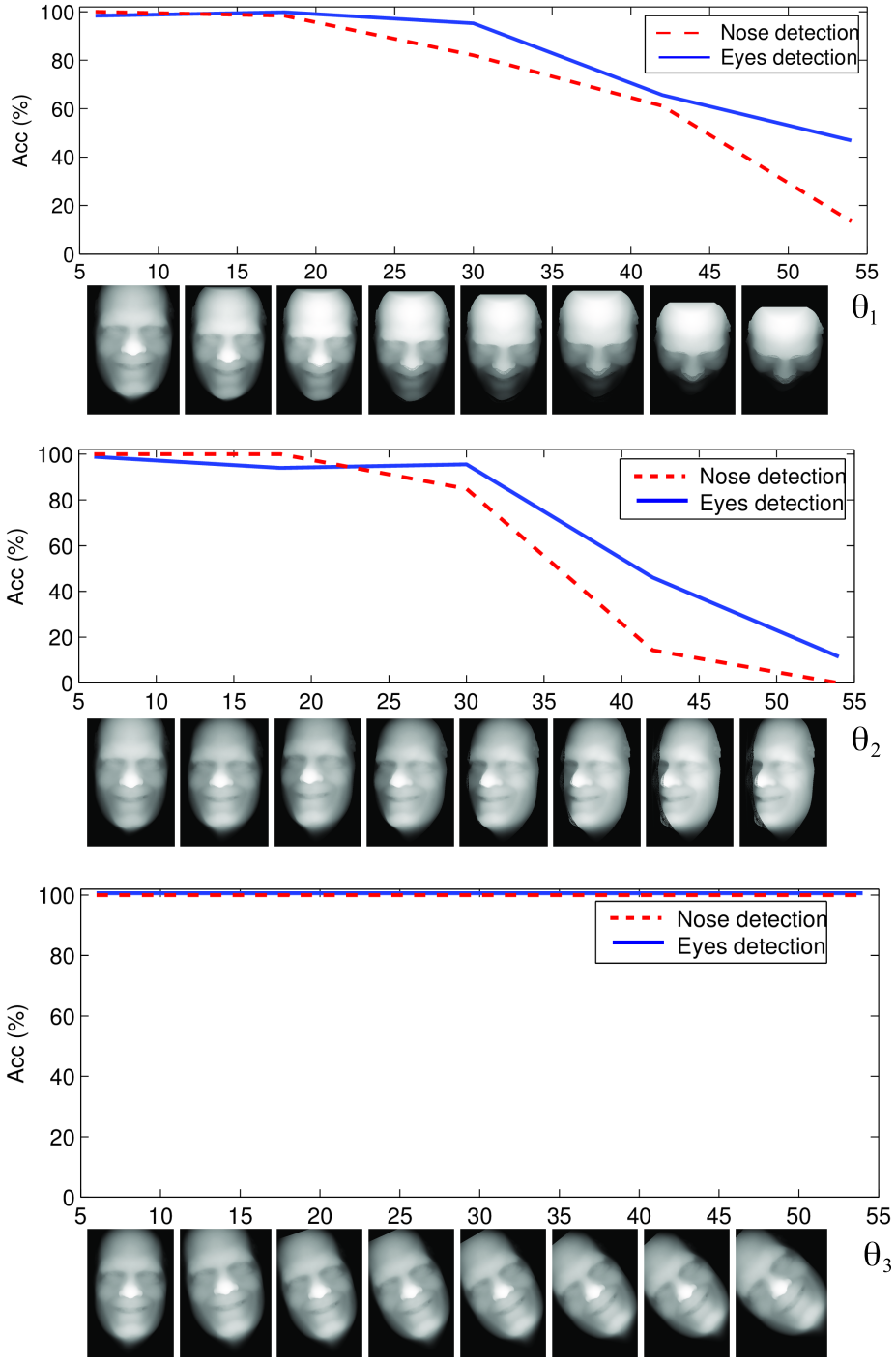


Figure 4.8: Results when varying face position.

blue rectangle in Fig. 4.9) corresponds to the regions around eyes and the second one (PatchB illustrated in red in Fig. 4.9) corresponds to the region around the nose. The region of the nose, is the portion of the face less distorted by facial expressions, thus is called the rigid region. The area of the eyes is called semi-rigid region [61], and finally, the rest of the face is much more affected due to facial expression, and therefor was not considered in the present work.

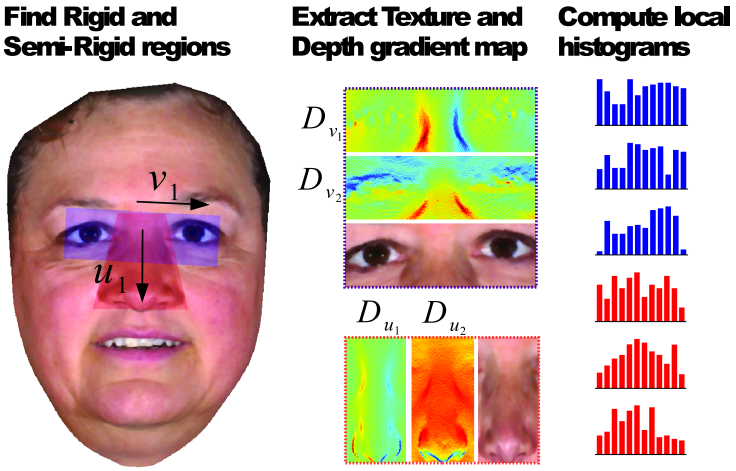


Figure 4.9: Illustration of the regions used for *expression-invariant* features extraction.

Following Sec. 2.2, we calculate the partial derivatives $D_x$ and $D_y$ of the face depth ($D$); then, using these partial derivative, over the $PatchA$ we have to calculate the directional derivatives $D_{v_1}$ and $D_{v_2}$, where $v_1$ is a direction parallel to the largest side of $PatchA$ and $v_2 \perp v_1$, as illustrated in Fig. 4.9. Analogously $D_{u_1}$ and $D_{u_2}$ are calculated over $PatchB$, where $u_1$ is a direction parallel to the height of $PatchB$ and $u_2 \perp u_1$. Histograms for these quantities are computed to construct the 3D low level features descriptor, in addition, the Local Binary Pattern [74] (LBP) descriptor over the Texture image is also taken into account.

Examples of extracted features are illustrated in Fig. 4.10, where some features are displayed using different colors for different subjects.

Intuitively it can be seen that this feature vector will capture both the geometrical 3D information of the nose and eyes regions as well as its texture information. We decide to use the LBP descriptor for the texture image because it probes to

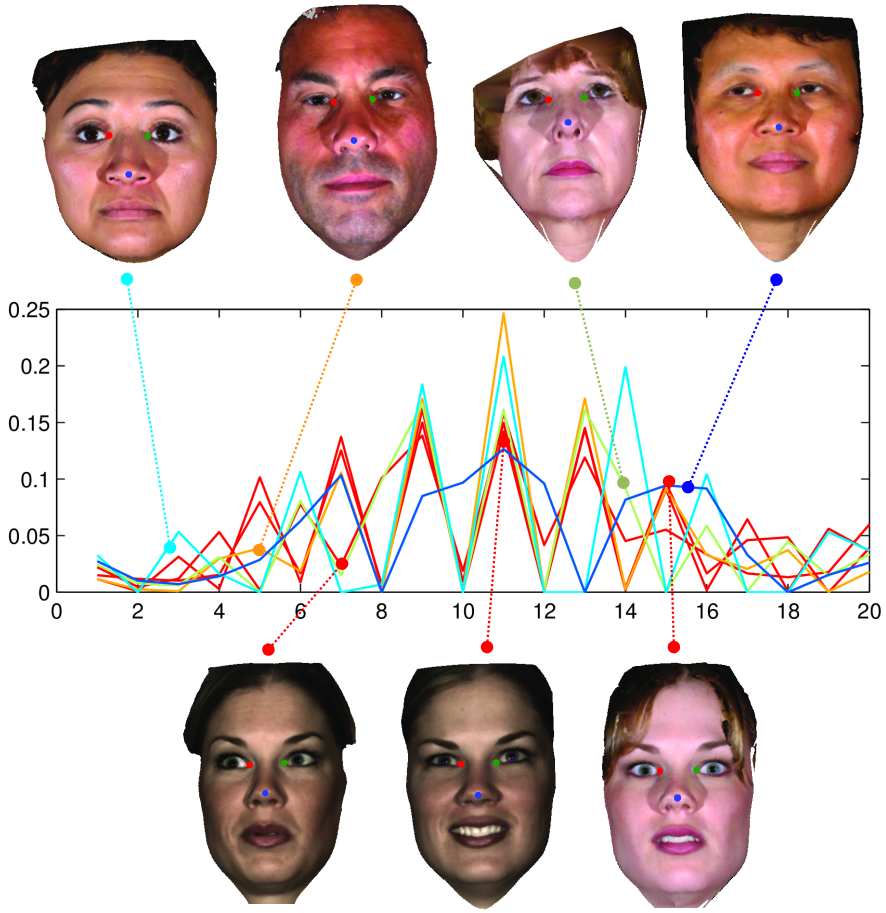


Figure 4.10: Example of some features extracted for different subject of Texas-Database. The colored curves correspond to a portion of the feature vector obtained for each of the subjects shown. The x-axis indicates the number of the feature, and the y-axis the value of each of the features (for each subject) in the normalized features space.

be a very efficient yet simple texture descriptor [5]. [Of course any other texture representation may be applied]. On the other hand, the histograms built directly from the depth partial derivatives represent itself low level geometrical features which can capture the main characteristics of face's geometry.

The proposed feature vector, is clearly scale invariant because, it is defined in terms of the own face dimension. It is also robust under facial expressions because only the rigid and semi-rigid portions of the face are taken into account. In addition, the geometrical descriptors (those histograms extracted from the $D_x$

and $D_y$ components) are invariant to ambient illumination conditions. Concerning pose variations, the proposed feature vector is invariant with respect to rotations along the $z$ axis (angle $\theta_3$ illustrated in Fig. 4.7) but it is not invariant under rotations along the other two axis ($\theta_2$ and $\theta_3$)[2].

### 4.2.4. Training and classifying

The last step of the proposed framework consists in training a classifier for each subject we want to recognize. We decide to use Support Vector Machine (SVM) because it shows to be an efficient and robust algorithm for the sake of face recognition [61].

Recall that for a binary classification problem, where we assume that are known $m$ training samples $x_k \in \mathbb{R}^N$ $(k = 1..m)$, with labels $y_k \in \{-1, 1\}$, SVM finds the hyper-plane with largest margin by solving the optimization problem:

$$\begin{aligned} &\min_{\omega,b,\xi} \left( \frac{1}{2}\omega^T\omega + C\sum_i \xi_i \right) \\ &\text{s.t. } y_i\left(\omega^T x_i + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0. \end{aligned} \tag{4.11}$$

The parameter $C$ is a penalty parameter of the error term and must be set; $\omega$ is a vector orthogonal to the hyper-plane, $b$ is a constant that sets the location of the hyper-plane and $\xi_i$ are auxiliary variables that allow to handle non separable problems. Depending on the problem, we may replace the constraint $y_i\left(\omega^T x_i + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0$ by $y_i\left(\omega^T \phi(x_i) + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0$ (Kernel SVM) which allows us to find the hyper-plane in a higher dimensional space. Once the SVM hyper-plane is obtained, one can classify new instances according to its positions with respect to the hyper-plane, i.e. by measuring the signed distance defined as:

$$d(x) = \frac{\omega^T x + b}{\|\omega\|}. \tag{4.12}$$

For a detailed explanation of the SVM algorithm we refer the reader to [16, 29, 52, 89].

[2]Build a feature representation -with the available differential information- invariant under rotations along any axis is an interesting problem in which we are working on.

As in the work of Y. Lei et all. [61], we opted by a *one versus all* approach. That means, that for each subject in the database, we will solve a two class problem in which we try to separate those feature vectors that belong to a given subject (named *positive* class) from the rest of the feature vectors (*negative* class). It is clear that, for each subject, we will have an hyper-plane that *best separates* it from the rest of the subjects in the database. To find each of those hyper-planes, we used the implementation of SVM given in [16] and we tried both linear SVM and Kernel-SVM [with a Radial Basis Function kernel (SVM-RBF)]. The cost parameter of SVM algorithm ($C$) and the kernel parameter ($\gamma$) (when the kernel was considered) were estimated performing 5-fold cross validation.

## 4.3. Experiments and evaluation

In this section, we perform two different sets of experiments; firstly, experiments with the Texas 3D Face Recognition Database [44, 45], and secondly, experiments with acquired images where the whole framework is tested. One of the challenges we may address for the evaluation lies in the fact that the proposed approaches do not use the explicit 3D model of the face, instead of this, a photo under structured light illumination is used from which 3D geometrical and texture information is extracted (without performing the explicit 3D reconstruction). For that reason, most of the standard *public* databases were not useful for the entire framework evaluation. In order to test all the steps of the proposed one-shot framework, specific data was acquired by us (and presented in the second part of this section). Despite of this, we include an evaluation with the Texas Database in which we already have each 3D face scan plus a texture image and a set of fiducial points coordinates. We use this public database to evaluate feature extraction and classification steps.

### 4.3.1. Evaluation over the Texas 3D Face Recognition database

This database contains 1149 pairs of high resolution, pose normalized, preprocessed, and perfectly aligned color and range images of 118 subjects. Additionally,

it includes the locations of 25 manually marked anthropometric facial fiducial points. From those 25 fiducial points, just three are used (the nose tip and the corner of the eyes). Examples of this database are shown in Figure 4.11.

Figure 4.11: Examples of the preprocessed color, and range images of the Texas 3D Face Recognition Database. Extracted From [44].

From the range image, $D_x$ and $D_y$ were computed using centered finite differences. Then, using the nose and eyes positions, PatchA and PatchB were extracted, and finally the feature vector from each subject was obtained following the steps described in section 4.2.3. Once feature vectors were computed for each pair of range and color images, the set was split in two. One part was used for training (estimating SVM hyper-plane parameters) and the other set was reserved for evaluation. Train and test sets contain (each) 486 samples obtained from scans of 25 different subjects. Once the 25 hyper-planes were obtained, the signed distance to each hyper-plane is measured for all the samples in train and test datasets. To each pair of range and color images, the assigned class is the one associated to the hyper-plane that has the higher distance to the respective feature vector.

| (Acc %) | $D_v^A$ $D_{v_\perp}^A$ | $D_u^B$ $D_{u_\perp}^B$ | $LBP_{(Tex.)}^A$ | $LBP_{(Tex.)}^B$ | All feat. PatchA | All feat. PatchB | All feat. |
|---|---|---|---|---|---|---|---|
| Train | $99{,}3 \pm 0{,}4$ | $98{,}4 \pm 0{,}5$ | $77{,}0 \pm 4{,}2$ | $51{,}6 \pm 3{,}6$ | $99{,}8 \pm 0{,}2$ | $99{,}4 \pm 0{,}3$ | $\mathbf{100 \pm 0{,}0}$ |
| Test | $91{,}9 \pm 1{,}7$ | $93{,}6 \pm 1{,}3$ | $64{,}9 \pm 3{,}8$ | $44{,}3 \pm 3{,}8$ | $95{,}2 \pm 0{,}8$ | $95{,}1 \pm 1{,}8$ | $\mathbf{99{,}4 \pm 0{,}2}$ |

Table 4.2: Accuracy (percentage) over Train and Test Sets. The superscript (A or B) recalls the patch from which features were extracted. Recall that the patch A corresponds to the region of the eyes while patch B corresponds to the region of the nose. The first two columns shows the results obtained just considering the geometrical information (represented by histograms of depth partial derivatives values). The last three columns shows the accuracy obtained by considering all the features extracted from the patch A, patch B and the union of both, respectively.

The results are summarized in Table 4.2 for different subsets of features. As was expected the area of the nose is the more robust region of the face for the sake of 3D face recognition when we have expression variations. This fact is in agreement with recent research in this field (see e.g. [17, 32, 61, 63]). As we see in Table 4.2, the geometrical information seems to be more effective than the texture analysis, this is also an expected result as the texture is easily affected, e.g. by illumination conditions and camera parameters. Finally, the best results were achieved by the fusion of all the features.

### 4.3.2. Evaluation of the entire system

In a second series of experiments our own one-shot database was used. This database was generated by illuminating different subjects with structured light, as described in section 4.2. For the pattern projection we used a commercial LCD-projector (model CP-RX80, Hitachi) with $1024 \times 768$ pixels, and the images were acquired with an 8-bit digital color camera (model DCU224C, Thorlabs) with $1280 \times 1024$ pixels. The distance of the subjects to the camera was about $50\,cm$, the camera pixels width is $4{,}65\,\mu m$ and camera's lens focal distance $f = 16\,mm$. We projected a rectangular pattern on each subject while a standard color picture was taken (obtaining images like those shown in Fig. 4.12). The images were

collected over different days, and therefore, illumination conditions, pose and face expressions present significant variations.

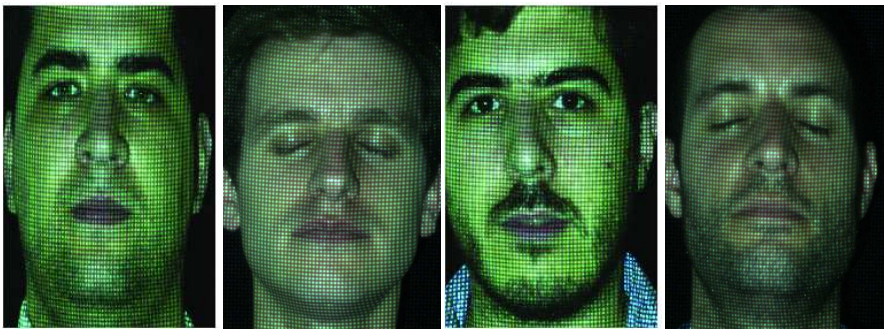

Figure 4.12: Example of some of the acquired images by us.

The database contains pictures of approximately 120 different subjects, and in most cases there is available only one picture for each person. As the main objective of this second series of experiment is to evaluate the robustness of the proposed framework under different pose and facial expressions, we will focus on the two class problem where multiples images of one particular subject play the role of the positive class and the rest of the subjects represents the negative class.

As training set we used 219 images, with multiple images of the target subject (positive class) and the rest of the images from different subjects representing the negative class. Figure 4.13 shows the euclidean distance between each sample to all the other samples on the training set. The black dashed line, shows the frontier of the images that belong to the same subject (positive samples) with respect to the images that belong to different subjects (all conforming the negative class). [This figure is just for illustrative proposes, recall that the classification is performed by obtaining SVM hyper-planes and not by a nearest neighbor approach].

Once the classifier is trained and we have a model for the target subject (positive class) we measure the distance of new data with respect to SVM hyper-plane. The test set is composed of 48 new negative instances (new pictures of people that were not present in the training set) plus 46 new positive instances (pictures of the target subject). In this test dataset, a wide range of facial expressions and pose variations are included as well as some pictures of the subjects with the face par-

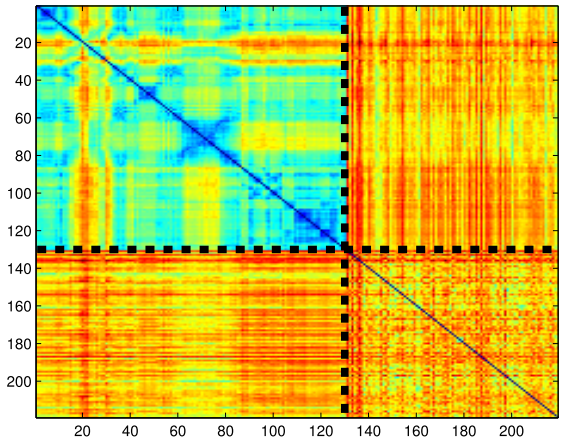


Figure 4.13: Illustrative figure that displays the euclidean distance (in the feature space) of each of the samples with respect to all the other samples in the database.

tially occluded (i.e. wearing a scarf or a bonnet as illustrated in Fig. 4.14). Figure 4.14 shows red/green histograms obtained by measuring the signed distance to the hyper-plane of each negative/positive sample on the test set.

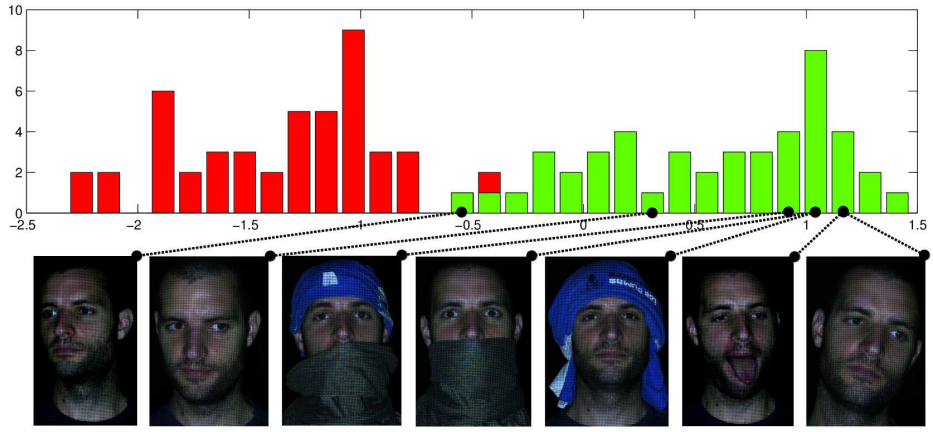


Figure 4.14: Number of test instances versus its signed distance to SVM hyperplane. Red and green histograms were obtained by considering the negative and positive samples respectively. Under the histogram, examples of positive samples are shown. The dashed lines shows the particular distance obtained for each of the example images.

By considering the signed distance of each test sample to the hyper-plane, one can classify each sample as positive if the signed distance [Eq. (4.12)] is higher than

certain threshold and negative in the other case. Figure 4.15 shows the Precision[3], the Recall[4], and the Precision-Recall curve for different threshold values.

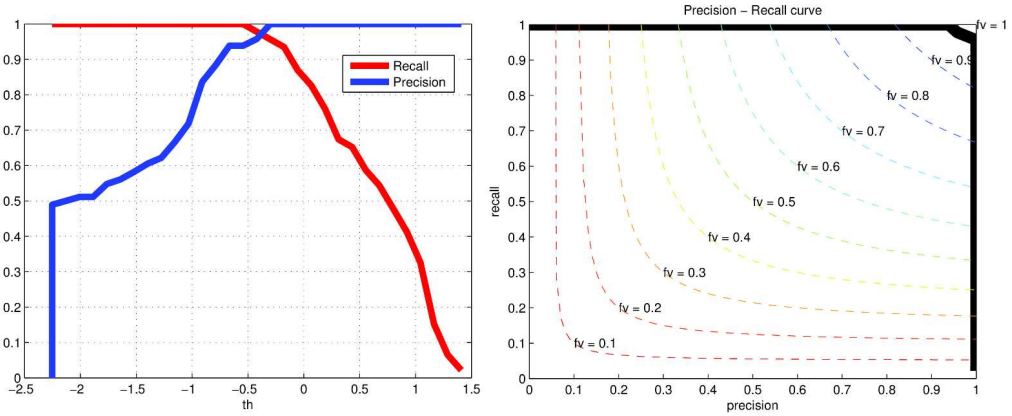


Figure 4.15: Left: Recall and Precision for different threshold values. Right: Precision-Recall curve obtained varying the threshold value used for classification. The colored dashed lines shows the lines of constant F-measure (which represents the geometrical mean between Recall and Precision).

A larger database is required to perform more exhaustive experiments (e.g. by repeating the previous two-class experiment for several subjects) before extracting quantitative conclusions. Despite this, interesting aspects of the proposed technique can be distinguish. Firstly, the proposed approach shows to be robust to facial expression (even exaggerated ones). Secondly, also promising results were achieved when the areas of the face (other than the nose or forehead) were occluded (e.g. by a scarf or bonnet), which shows that the approach is also robust to some intentional variations that a subject may produce such as variations of facial hair (e.g. shave his beard off). Thirdly, the framework is robust to face rotations with respect to the axis that goes from the center of the face to the center of the camera (e.g. see the last (from left to right) example image of those shows in Fig. 4.14). When the face rotates along the other two orthogonal axes, the distance to the hyper-plane becomes smaller (as can be seen in Fig. 4.14) because the proposed

[3]Portion of the samples labeled as positive that actually belong to the positive class.
[4]Portion of samples of the positive class correctly classified.

feature representation is not invariant under these kinds of pose variations -as was mentioned in Sec. 4.2.3-.

## 4.4. Conclusions

In this chapter a one-shot 2D+3D face recognition approach was presented which unlike previous work extract three-dimensional low level geometrical features of the face just from its Depth gradient map bypassing the three-dimensional facial reconstruction. The proposed methods uses the location of the eyes and nose to build a robust representation of a face which is invariant to changes in facial expression, scale, ambient illumination and some pose variations. The position of the eyes and nose are automatically detected using gradient information and constraining distance between landmarks.

Instead of using a 3D scanner, we show how it is possible to take advantage of the Differential method introduced in Chapter 2 to build a feature representation without the need of an explicit 3D reconstruction of the face. Moreover the use of SVM to find the regions of the feature space that better represents each subjects has different advantages; for example, to classify new instances we just need to know the set of hyper-planes that represent each class, and not the whole set of images used for training. This could save lot of memory resources and allows to perform a fast and efficient face recognition system. Furthermore matching can be done just performing a inner product in the feature space which is computationally extremely fast. In second place, the proposed framework can be implemented with low hardware resources, e.g. one just need a standard camera and a simple projector device such the one illustrated in Fig. 4.16.

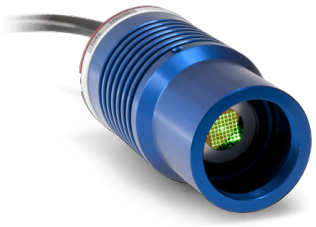

Figure 4.16: Example of a simple projector device [30].

On the other hand, the proposed framework is likely limited to indoor scenarios

(as the projected pattern may lost most of its contrast if the ambient illumination is too high). Additionally the presented approach requires a set of images of the subjects we want to be able to identify (for the negative class one single image of many different subjects should be enough) as a SVM algorithm must be trained [61].

In the present chapter we presented a particular application of some of the ideas introduced along this thesis. In particular we showed how D3D (introduced in Chapter 2) can be applied bypassing the integration step (described in Chapter 3). The particular framework for face recognition presented in this chapter do not intend to be optimal and we think there is plenty of room for improvements and further analysis which could leads to interesting future work.

Besides we addressed in this chapter one particular application, the main ideas can be extended to other popular problems in the field of pattern recognition and computer vision such as Gesture Detection or Objects Recognition. As the proposed technique can be easily extended for moving scenes, the use of D3D for Gesture Detection could also be an interesting future work.

# Chapter 5

# Conclusions

In this thesis, we have first reviewed different popular 3D scanning techniques, focusing on those that use the projection of structured light, and measure pattern deformation to extract information about the height map of the scene. We have then introduced a novel 3D scanning approach, which provides us with the depth field's gradient map using very simple hardware and computational resources. Following, we have analyzed the two main sources of error we may have on gradient estimations: (i) areas with very low gradient i.e. local maxima and minima of the fringes, and (ii) surface discontinuities or needle points. After presenting a geometrical interpretation of the proposed method, we were able to develop different approaches to mitigate these sources of error and handle erroneous measurements during the integration step.

The introduction of a *gradient* approach, led us to the question: how this gradient field should be integrated? This motivates Chapter 3 where once again, we started by reviewing the most popular and practical integration techniques and then we moved a step forward and extended some particular techniques to our problem. As we can estimate a 2D gradient field along time, we have explored how temporal correlation can be exploited to enforce the integrability in a more robust way. In that direction we have proposed two different ways of regularizing the integration process of a temporal 2D gradient field.

Finally, we have presented an approach for robust face recognition that takes

advantage of some of the characteristics the proposed Differential 3D technique has. For example, in most of the problems of computer vision and pattern recognition, it is necessary to build feature vectors to describe reality and move to a mathematically more convenient feature space. In that process, the use of gradient information is very frequent (SIFT is one very popular example). This idea is exploited in Chapter 4 where we have presented an efficient and robust face recognition approach which can be achieved bypassing the explicit 3D reconstruction of the face.

## List of publications.

### Articles directly related to the topics presented in this thesis.

The main ideas behind the proposed Differential 3D approach (presented in Sec. 2.2) were published in Optics and Lasers in Engineering Journal [20, 25].

Some of the regularization approaches presented in Chapter 3 were also published in Optics and Lasers in Engineering Journal [24].

The step for automatic nose tip and eyes candidates detection presented in Chapter 4 was recently published and presented in IEEE International Conference on Image Processing (ICIP-2014) [23].

### Other works.

During the first years of academic and research activities, I worked with other members of Applied Optics Group in different methods that help to compensate the nonlinear response of projectors and cameras in order to improve 3D phase-shifting techniques. Some examples of these works are [7] and [8]. In the first one we have proposed the use of a Pulse Width Modulation technique which after defocusing, allows us to project a very precise sinusoidal pattern (using just black and white levels of the projector). This method shows to be very practical to minimize the projector's nonlinear response effects. In the second work we have proposed to code the sinusoidal profile in binary patterns, this technique overcomes

both camera's and projector's nonlinear response.

In the field of pattern recognition I also worked on improving classification performance for imbalanced problems. These problems are very frequent in fields such as information retrieval [68], fraud detection [21] or medical diagnosis [33,34]; where one of the classes has orders of magnitude more instances than the others, and hence the typical *error minimization* approaches fails. In that direction two different approaches were developed and published in Pattern Recognition and Pattern Recognition Letters journals [22, 26].

## 5.1. Future lines of research

Developments presented in this thesis open a new branch of interesting questions that may be addressed in future projects. We will conclude this work by enumerating what we think are the most interesting and promising lines of future research.

- In Chapter 2, we discuss the two main sources of error the proposed method has on 3D gradient estimation. If we, instead of projecting a fixed fringe pattern, project additional patterns in which fringes' phase -or period- changes; the set of noisy points will be different from one frame to the other. This would help to reduce the effects of noise and outliers on gradient integration. However if we use multiple patterns the new approach will no longer be one-shot and other concerns must be faced (e.g. synchronization between projection and acquisition, and registration in the case of moving scenes.).

- As was discussed in Chapter 2, stereo and matching techniques have the advantage of providing -sparse- absolute depth measurements when it is possible to find correspondences. On the other hand the proposed technique can provide a dense gradient map estimation that may fail around discontinuity regions. It would be very interesting to merge these two approaches into one single method. For example, we are thinking on projecting the proposed pattern coded in one of the color channels of a projector, while projecting sparse

landmarks in another channel. By acquiring this unique -colored- pattern, we can find the disparity of a sparse set of landmarks (which can be used as anchorages) and fill the space between them by propagating the dense gradient information.

- In Chapter 3 other regularization techniques can be studied, in particular we focused on $L_2$-norm and $L_1$-norm techniques but there are other alternatives that may be applied. In addition, improvements can be done to achieve computationally more efficient implementations of some of the discussed methods.

- The framework described for Face Recognition could be analyzed thoroughly if a bigger and more general dataset is collected.

- A more robust feature representation should be explored, for instance one that is invariant under rotations of the face in any direction.

- Many of the ideas presented in Chapter 4 can be modified and adapted for the sake of gesture recognition, mood estimation or micro expressions detection. One potential and very interesting application would be the use of a high speed camera combined with the proposed Differential 3D technique to design a system that automatically detects involuntary facial micro-expressions [79].

# Appendix A

# LS approach Euler-Lagrange equation:

The function $u : \Omega \to \mathbb{R}$ that minimizes:

$$E_{LS}[u] = \int_{\Omega} \left((u_x - p)^2 + (u_y - q)^2\right)\ dx\, dy \tag{A.1}$$

must satisfy

$$\lim_{\epsilon \to 0} \frac{E[u + \epsilon v] - E[u]}{\epsilon} = 0 \quad \forall\ v \tag{A.2}$$

which leads to,

$$\lim_{\epsilon \to 0} \frac{1}{\epsilon} \left[\int_{\Omega} \left(\left((u_x + \epsilon v_x - p)^2 + (u_y + \epsilon v_y - q)^2\right) - \left((u_x - p)^2 + (u_y - q)^2\right)\right)\ dx\, dy\right] = 0 \tag{A.3}$$

$$\Rightarrow \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left[\int_{\Omega} \left(2\epsilon v_x(u_x - p) + 2\epsilon v_y(u_y - q) + \mathcal{O}(\epsilon^2)\right)\ dx\, dy\right] = 0 \tag{A.4}$$

$$\Rightarrow \left[\int_{\Omega} \left(2v_x(u_x - p) + 2v_y(u_y - q)\right)\ dx\, dy\right] = 0 \tag{A.5}$$

which can be transform using integration by parts formula (see e.g. [6, Pag.62]) in

$$-2\int_{\Omega} \left(v\left((u_x - p)_x + (u_y - q)_y\right)\right)\ dx\, dy + \int_{\partial\Omega} v(u_x - p)\nu_1 ds + \int_{\partial\Omega} v(u_y - q)\nu_2 ds = 0 \tag{A.6}$$

where $\nu_i$ is the $i$-th component of the outward unit normal of $\partial\Omega$. In order to satisfy Eq. (A.6) for any arbitrary function $v$ we obtain:

$$\begin{cases} \nabla^2 u = \texttt{div}(p, q) \\ (u_x - p)\nu_1 = 0 \text{ on } \partial\Omega \\ (u_y - q)\nu_2 = 0 \text{ on } \partial\Omega \end{cases} \tag{A.7}$$

# Appendix B

# Generalized LS and AD approaches Euler-Lagrange equation:

The function $u : \Omega \times [0\ T] \to \mathbb{R}$ that minimize the functional:

$$E[u] = \int_{\Omega\times[0\ T]} \Big( (1-\lambda)\left((\nabla u - \nabla g)^T D(\nabla u - \nabla g)\right) + \lambda \left| d^\alpha u/dt^\alpha \right|^2 \Big)\, dxdydt \tag{B.1}$$

must satisfy,

$$\delta E_v = \lim_{\epsilon\to 0^+} \frac{E[u+\epsilon v] - E[u]}{\epsilon} = 0 \quad \forall\ v : \Omega \times [0\ T] \to \mathbb{R} \tag{B.2}$$

which is equivalent to,

$$\begin{aligned} \delta E_v = \lim_{\epsilon\to 0^+} \frac{1}{\epsilon} \int_{\Omega\times[0\ T]} \big\{ & (1-\lambda)\left((\nabla(u+\epsilon v) - \nabla g)^T D(\nabla(u+\epsilon v) - \nabla g)\right) \\ & + \lambda \left| d^\alpha (u+\epsilon v)/dt^\alpha \right|^2 - (1-\lambda)\left(\nabla u - \nabla g)^T D(\nabla u - \nabla g)\right) \\ & - \lambda \left| d^\alpha u/dt^\alpha \right|^2 \big\}\, dxdydt. \end{aligned} \tag{B.3}$$

The previous expression is equal to,

$$\begin{aligned} \delta E_v = \lim_{\epsilon\to 0^+} \int_{\Omega\times[0\ T]} \Big\{ \Big( & 2(1-\lambda)\left((\nabla v)^T D(\nabla u - \nabla g)\right) \\ & + 2\lambda \frac{d^\alpha v}{dt^\alpha}\frac{d^\alpha u}{dt^\alpha} + \mathcal{O}(\epsilon) \Big) \Big\}\, dxdydt \end{aligned} \tag{B.4}$$

where the term $\mathcal{O}$ vanishes when we take the limit. Applying Green's formulas we arrive to,

$$\begin{aligned} \delta E_v = \int_{\Omega\times[0\ T]} \Big\{ \Big( & -2v^T(1-\lambda)\left(\mathtt{div}\left(D(\nabla u - \nabla g)\right)\right) + 2\lambda \frac{d^\alpha v}{dt^\alpha}\frac{d^\alpha u}{dt^\alpha} \\ & + \mathcal{O}(\epsilon)) \Big\}\, dxdydt + (1-\lambda)\left( \int_{\partial\Omega\times[0\ T]} \nabla u \cdot \hat{n}\, v\ dsdt. \right) \end{aligned} \tag{B.5}$$

## Appendix B. Generalized LS and AD approaches Euler-Lagrange equation:

In the previous equation $\hat{n}$ denotes a unitary vector normal to $\partial\Omega$. The last steps consists in apply the integration by parts formula $\alpha$ times leading to

$$\delta E_v = \int_{\Omega\times[0\ T]} \left\{ -2v(1-\lambda)\left( (\texttt{div}\,(D(\nabla u - \nabla g))) + (-1)^{\alpha-1}\lambda \frac{d^{2\alpha}u}{dt^{2\alpha}} \right) \right\} dxdydt$$
$$+(1-\lambda)\left( \int_{\partial\Omega\times[0\ T]} \nabla u\cdot\hat{n}\, v\ dsdt \right) + \lambda \sum_{p=1}^{\alpha} (-1)^{p-1} \int_{\Omega} \frac{d^{\alpha-p}v}{dt^{\alpha-p}} \frac{d^{\alpha+p-1}u}{dt^{\alpha+p-1}} \Big|_0^T\ dxdy. \tag{B.6}$$

The conditions given by Eqs. (3.26) and (3.27) are sufficient conditions to guarantee that the second term of Eq. (B.6) vanishes and, the condition $\delta E_v = 0$ for all $v$ leads to

$$(1-\lambda)\,(\texttt{div}\,(D(\nabla u - \nabla g))) + (-1)^{\alpha-1}\lambda \frac{d^{2\alpha}u}{dt^{2\alpha}} = 0 \tag{B.7}$$

which correspond to the conditions stated in Eq. (3.25) as we want to prove.

The deduction of equation (3.22) is a particular case of the previous one when $D$ is the identity $2\times 2$ matrix for all $x,\ y,\ t$.

# Appendix C

# Integration constraining objects' movement E-L equations:

Given the energy function (Eq. 3.33),

$$E[u(x,y,t), vx(x,y,t), vy(x,y,t)] = \int_{\Omega\times[0\,T]} \left( |u_x - p|^2 + |u_y - q|^2 + \lambda\,|u_t + vx\,p + vy\,q|^2 + \mu\left(|\nabla vx|^2 + |\nabla vy|^2\right)\right) dxdydt. \tag{C.1}$$

where $p : \mathbb{R}^3 \to \mathbb{R}$ (estimation of the x-partial derivative), $q : \mathbb{R}^3 \to \mathbb{R}$ (estimation of the y-partial derivative), $\lambda$ and $\mu$ are known. We want to find the conditions that $u : \mathbb{R}^3 \to \mathbb{R}$, $vx : \mathbb{R}^3 \to \mathbb{R}$ and $vy : \mathbb{R}^3 \to \mathbb{R}$ must satisfy when $E[u, vx, vy]$ is minimized.

We know that the solution $\{\tilde{u}, \tilde{vx}, \tilde{vy}\} = argmin\,\{E[u, vx, vy]\}$ must satisfy,

$$\begin{cases} \lim\limits_{\epsilon\to 0} \dfrac{E[\tilde{u} + \epsilon A_1, \tilde{vx}, \tilde{vy}] - E[\tilde{u}, \tilde{vx}, \tilde{vy}]}{\epsilon} = 0 \quad \forall\ A_1 : \mathbb{R}^3 \to \mathbb{R} \\ \lim\limits_{\epsilon\to 0} \dfrac{E[\tilde{u}, \tilde{vx} + \epsilon A_2, \tilde{vy}] - E[\tilde{u}, \tilde{vx}, \tilde{vy}]}{\epsilon} = 0 \quad \forall\ A_2 : \mathbb{R}^3 \to \mathbb{R} \\ \lim\limits_{\epsilon\to 0} \dfrac{E[\tilde{u}, \tilde{vx}, \tilde{vy} + \epsilon A_3] - E[\tilde{u}, \tilde{vx}, \tilde{vy}]}{\epsilon} = 0 \quad \forall\ A_3 : \mathbb{R}^3 \to \mathbb{R} \end{cases} \tag{C.2}$$

The first condition leads to,

$$\lim_{\epsilon\to 0} \int_{\Omega\times[0,T]} \left(2A_{1x}(u_x - p) + 2A_{1y}(u_y - q) + 2\lambda A_{1t}(u_t + vx\,p + vy\,q) + \mathcal{O}(\epsilon)\right) dxdydt = 0$$

after taking the limit and using Green's formula the previous equation imply,

$$\begin{aligned} &- \int_{\Omega\times[0,T]} 2A_1\left((u_x - p)_x + (u_y - q)_y + \lambda(u_t + vx\,p + vy\,q)_t\right) dxdydt \\ &+ \int_{\partial(\Omega\times[0,T])} 2A_1\left((u_x - p)\hat{x} + (u_y - q)\hat{y} + \lambda(u_t + vx\,p + vy\,q)\hat{t}\right) \nu ds = 0. \end{aligned} \tag{C.3}$$

## Appendix C. Integration constraining objects' movement E-L equations:

$\nu$ is the outward unit normal of $\partial(\Omega \times [0,T])$, and $\hat{x}$, $\hat{y}$ and $\hat{t}$ denotes unitary vectors in the direction of the axis $x$, $y$ and $t$ respectively. Assuming Neumann boundary conditions equal to zero, Eq. C.3 is holds for all $A_1$ if,

$$u_{xx} + u_{yy} + \lambda u_{tt} + \lambda(vx\, p + vy\, q)_t = p_x + q_y. \tag{C.4}$$

Continuing with the second condition of Eq. C.2, we obtain

$$\lim_{\epsilon \to 0} \int_{\Omega \times [0,T]} \left( 2\lambda A_2\, p(u_t + vx\, p + vy\, q) + 2\mu (\nabla A_2)^T \nabla vx + \mathcal{O}(\epsilon) \right) dxdydt = 0$$

and again by taking the limit and using Green's formula we obtain,

$$\begin{aligned} &\int_{\Omega \times [0,T]} 2A_2 \left( \lambda\, p(u_t + vx\, p + vy\, q) - \mu \Delta vx \right) dxdydt \\ &+ \int_{\partial(\Omega \times [0,T])} 2\mu\, A_2 \left( \frac{\partial vx}{\partial \nu} \right) ds = 0. \end{aligned} \tag{C.5}$$

Assuming Neumann border conditions equal to zero, Eq. C.5 holds for any arbitrary $A_2$ if,

$$\lambda\, p(u_t + vx\, p + vy\, q) - \mu \Delta vx. \tag{C.6}$$

Analogously we obtain a similar condition for the third condition of Eq. C.2, and if we join all the equations we obtain the set of Euler-Lagrange equations

$$\begin{cases} u_{xx} + u_{yy} + \lambda u_{tt} + \lambda(vx\, p + vy\, q)_t = p_x + q_y \\ \lambda p\, (u_t + p\, vx + q\, vy) - \mu \Delta vx = 0 \\ \lambda q\, (u_t + p\, vx + q\, vy) - \mu \Delta vy = 0. \end{cases} \tag{C.7}$$

# Appendix D

# Code and implementation details

## D.1. Some 3D scanning techniques

### D.1.1. D3D

```
%% [D,[Texture],[Dx],[Dy],[Qx],[Qy]] = mt_D3D(I,parameters)
% function that performs differential 3D retrieval (I = image
% acquired with projected fringe patterns)
%
% Inputs:
%   - I (mxnxc) gray or color image with horizontal and
%               vertical fringes. Fringes can be multiplied,
%               summed of coded in the blue and red channels
%               of the projector. Indicate the kind of
%               codification used in "parameters.CodeType".
%               By default it is assumed that CodeType
%               ='sum'.
%
%   - parameters, struct that may contain:
%       .CodeType [def sum] {'sum','prod','color'}; in the
%                           first two, the fringes are
%                           overlapped in all the channels.
%                           In the case of 'color' we assume
%                           that VERTICAL fringes are coded
%                           in the blue channel and HORIZONTAL
%                           fringes in the red channel.
%       .verbose [def 0]: 1 display some text, 2 also graphics.
%       .IntegrationMethod
%           - 'LS' : Least Squares Integration
%           - 'WLS': Weighted Least Squares Integration (Qx,Qy)
%           - 'FC' : Frankot-Chellapa approach (Fourier)
%           - 'AD' : Anisotropic diffusion approach,
%           - 'L1' : L1-error minimization,
%           - 'M'  : M-estimator integration approach,
```

```
30 %       .FiltWidth [def [1 1]] (width of the median filter
31 %                               window in (in terms of the
32 %                               fringes width), this filter
33 %                               is applied to the gradient
34 %                               field before integration).
35 %       .TextureIm: texture image (used for normalization,
36 %                   if it is not provided, we estimate it from
37 %                   the low frequencies of I.
38 %       .Theta: [def pi/4] angle in which the projector
39 %                          was translated (with respect to
40 %                          the camera)
41 %
42 % Outputs:
43 %   - D (mxnx1)
44 %   - Texture (mxnxc); [optional]
45 %   - Dx (mxnx1) [optional]
46 %   - Dy (mxnx1) [optional]
47 %   - Qx (mxnx1) [optional] Dx quality map
48 %   - Qy (mxnx1) [optional] Dy quality map
49 %
50 % Refs:
51 %   [1] Matias Di Martino, Gaston Ayubi, Alicia Fernandez and
52 %       Jose Ferrari."Differential 3d shape retrieval", Optics
53 %       and Lasers in Engineering, 58C:114-118, 2014.
54 %
55 % see also: integration, PSI, Takeda
56 %
57 % ------------------------------------------------------------
58 % matias di martino (c) - matiasdm@fing.edu.uy - 2014   v.0.3
59 % ------------------------------------------------------------
60 function [D,varargout] = mt_D3D(Iin,parameters)
61
62 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63 %% Load and set general parameters, %%%%%%%%%%%%%%%%%%%%%%%%%
64 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65 list    = {'CodeType' ,'verbose'  ,'IntegrationMethod',...
66            'FiltWidth','TextureIm','Theta'};
67 defval = {'sum'      ,0          ,'LS'                ,...
68            1         ,[]         ,pi/4   };
69 [CodeType,verbose,IntegrationMethod,FiltWidth,TextureIm,Theta]...
70 = GetParameters(parameters,list,defval);
71
72 Iin = double(Iin);
73
74 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
75 %% Calculate Fringe Images %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77 [fw] = EstimateFringeWidth(Iin,verbose);
78
79 % if we don't have a TextureIm estimate it from low freq.
80 if isempty(TextureIm);
```

```
81      [TextureIm] = EstimateTextureImage(Iin,fw,verbose);
82      if strcmpi(CodeType,'color');
83          TextureIm = mean(TextureIm,3);
84      end
85  end
86  varargout{1} = TextureIm;
87
88  % Compute Ih and Iv image components.
89  switch lower(CodeType),
90      case 'color',
91          Ih   = Iin(:,:,1)./TextureIm(:,:,1);
92          if size(TextureIm,3)>1,
93              Iv   = Iin(:,:,3)./TextureIm(:,:,3);
94          else
95              Iv   = Iin(:,:,3)./TextureIm(:,:);
96          end
97          k    = fspecial('gaussian',round(.5*[fw fw]),fw/3);
98          Ih   = conv2(Ih,k,'same'); Iv = conv2(Iv,k,'same');
99      otherwise % if the image has the fringes overlapped:
100                % filter in the Fourier domain,
101         I = Iin./TextureIm; I = mean(I,3);
102         [Ih,Iv] ...
103         = ExtractVerticalAndHorizontalFringes(I,fw,verbose);
104 end
105
106 if verbose>1, % show Ih and Iv;
107     figure('name','[mt_D3D] Ih and Iv');
108     subplot(121), imshow(Ih,[]), title('Ih');
109     subplot(122), imshow(Iv,[]), title('Iv');
110 end
111 % %%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ////////////////////////////////////////////////////////////
function [varargout] = GetParameters(parameters,list,defval)
% ////////////////////////////////////////////////////////////

% first check that all the parameters are in the list.
names = fieldnames(parameters);
for par = 1:length(names);
    f        = names{par};
    isinlist = 0;
    for j = 1:length(list);
        if strcmpi(f,list{j});
            isinlist = 1;
        end
    end
    if isinlist == 0;
        warning(['[mt_D3D] input parameter ' f ' in unknown ']);
    end
end

% now load input parameters or assign default values,
for par = 1:length(list);
    if isfield(parameters,list{par})
        varargout{par} = getfield(parameters,list{par});
    else % Set default value,
        varargout{par} = defval{par};
    end
end

if isfield(parameters,'verbose');
    if parameters.verbose>0;
        % show parameters values,
        display('[mt_D3D] Input parameters: ');
        for par = 1:length(list);
            if length(varargout{par})>20;
                auxstr = 'ToLargeToBeDisplayed';
            else
                auxstr = num2str(varargout{par});
            end
            fprintf(['\t' list{par} ' > ' auxstr '\n']);
        end
    end
end
end

% ////////////////////////////////////////////////////////
function [fw] = EstimateFringeWidth(I,verbose)   % ////////
% ////////////////////////////////////////////////////////
[m,n] = size(I);
% Define some shortcuts for image transformations ------
ft2  = @(u) fftshift(fft2(u));
```

```
183 ift2 = @(u) real(ifft2(ifftshift(u)));
184 % ------------------------------------------------------------
185
186 % A0) Estimate fringes width ----------------------
187 FT = ft2(mean(I,3)); % Fourier transform of the gray image
188 % Remove zero frec,
189 waux = 25;
190 [aux,i0] = max(abs(FT));
191 [~,j0] = max(aux'); i0 = i0(j0);
192 FT(i0+[-waux:waux],:) = 0;
193
194 % find max (correspond to fringes)
195 [aux,i1] = max(abs(FT));
196 [~,j1] = max(aux'); i1 = i1(j1);
197 di = abs(i1(1)-i0(1));
198 fw = round( 1/di * m );
199
200 if verbose>0,
201   disp(['[mt_D3D] fringes width :  ' num2str(fw) '(pixels)']),
202 end
203 % ------------------------------------------------
204 end
205
206 % ////////////////////////////////////////////////////
207 function [T] = EstimateTextureImage(I,fw,verbose) % /////
208 % ////////////////////////////////////////////////////
209 [m,n,col] = size(I);
210
211 % Define some shortcuts for image transformations -----
212 ft2  = @(u) fftshift(fft2(u));
213 ift2 = @(u) real(ifft2(ifftshift(u)));
214 % ------------------------------------------------------------
215
216 % A1) Filter on Fourier domain ------------------------------
217 f0x = round(n/2)+1; f0y = round(m/2)+1; % center of spectrum
218 % Filter in Fourier domain ----------------------------------
219
220 % Gaussian low pass filter ----------------------
221 LPF          = zeros(m,n); % LowPassFilter
222 LPF(f0y,f0x) = 1;
223 KSIZE        = [round(m/fw) round(n/fw)];
224 KSIGMA       = .5 * (round(m/fw)+round(n/fw)) / 2 ;
225 K            = fspecial('gaussian',KSIZE,KSIGMA);
226 LPF          = conv2(LPF,K,'same');
227 % ----------------------------------------------------
228
229 T = I; % initialization;
230 for c = 1:col;
231     T(:,:,c) = ift2(LPF.*ft2(I(:,:,c)));
232 end
233
```

```
234 if verbose>1, % show spectrum, filter and Texture image.
235     figure('name','[mt_D3D] ft(Iin) + LPF'),
236     imagesc(log(abs(mean(ft2(I),3)))), colormap jet;
237     AuxLPFImage = cat(3,zeros(m,n),zeros(m,n),ones(m,n));
238     hold on,
239     imagesc(AuxLPFImage,'AlphaData',LPF/max(LPF(:))*.5),
240     hold off,
241     figure('name','[mt_D3D] Estimated TextureIm'),
242     imshow(uint8(mt_Normalize(T,[0 255])));
243 end
244 end
245
246 % ////////////////////////////////////////////////////////////
247 function [Ih,Iv] = ...
248          ExtractVerticalAndHorizontalFringes(I,fw,verbose)
249 % ////////////////////////////////////////////////////////////
250 I = mean(I,3); % convert to gray
251 [m,n] = size(I);
252
253 % Define some shortcuts for image transformations ------
254 ft2  = @(u) fftshift(fft2(u));
255 ift2 = @(u) real(ifft2(ifftshift(u)));
256 % ----------------------------------------------------------
257
258 % A1) Filter on Fourier domain ------------------------------
259 f0x = round(n/2)+1; f0y = round(m/2)+1; % center of spectrum
260 % ----------------------------------------------------------------
261
262 % Gaussian Band pass filter ---------------------------
263 BPFv         = zeros(m,n); % Band Pass Fil. (horizontal direc.)
264 BPFh         = zeros(m,n); % Band Pass Fil. (vertical direc.)
265 BPFv(f0y,f0x+round(n/fw)) = 1; BPFv(f0y,f0x-round(n/fw)) = 1;
266 BPFh(f0y+round(m/fw),f0x) = 1; BPFh(f0y-round(m/fw),f0x) = 1;
267 KSIZE        = [round(m/fw) round(n/fw)];
268 KSIGMA       = .5 * ( round(m/fw)+round(n/fw) ) / 2;
269 K            = fspecial('gaussian',KSIZE,KSIGMA);
270 %K            = hanning(round(m/fw))*hanning(round(n/fw))';
271 BPFv         = conv2(BPFv,K,'same');
272 BPFh         = conv2(BPFh,K,'same');
273 % ----------------------------------------------------------
274
275 Ih = ift2(ft2(I).*BPFh); Iv = ift2(ft2(I).*BPFv);
276 if verbose>1, % show spectrum and filters,
277     figure('name','[mt_D3D] ft(Iin) + BPFs'),
278     imagesc(log(abs(mean(ft2(I),3)))), colormap jet;
279     AuxBPFhImage = cat(3,ones(m,n),zeros(m,n),zeros(m,n));
280     AuxBPFvImage = cat(3,zeros(m,n),ones(m,n),zeros(m,n));
281     hold on,
282     imagesc(AuxBPFhImage,'AlphaData',BPFh/max(BPFh(:))*.5),
283     imagesc(AuxBPFvImage,'AlphaData',BPFv/max(BPFv(:))*.5),
284     hold off,
```

```
285 end
286 end
287
288 % ////////////////////////////////////////////////////////////
289 function [Dx,Dy,varargout]...
290       = ComputeGradientField(Ih,Iv,Theta,FiltWidth,fw,verbose)
291 % ////////////////////////////////////////////////////////////
292 % define some shortcuts
293 dx = @(U) 1/2*[0*U(:,1)  U(:,3:end)-U(:,1:end-2)  0*U(:,1)];
294 dy = @(U) 1/2*[0*U(1,:); U(3:end,:)-U(1:end-2,:); 0*U(1,:)];
295
296 %ta1 = dx(Ih)./dy(Ih);  ta2 = dy(Iv)./dx(Iv);
297 %Dx = ta1.*( 1+ta2*tan(Theta) )./( sin(Theta) * (ta1.*ta2-1) );
298 %Dy = ta2.*( 1+ta1/tan(Theta) )./( cos(Theta) * (ta2.*ta1-1) );
299
300 % instead of the previous, we compute an equivalent expression
301 % (which requires less divisions)
302 Dx = ( dx(Ih) .* ( dx(Iv) + dy(Iv)*tan(Theta) ) ) ./ ...
303      ( sin(Theta) * ( dx(Ih).*dy(Iv) - dy(Ih).*dx(Iv) ) );
304
305 Dy = ( dy(Iv) .* ( dy(Ih) + dx(Ih)/tan(Theta) ) ) ./ ...
306      ( cos(Theta) * ( dx(Ih).*dy(Iv) - dy(Ih).*dx(Iv) ) );
307
308
309 % Remove NaN and inf due to 0/0 or x/0;
310 % these are produced in local minimum or
311 % maximum of the fringes. These errors are
312 % corrected after we apply the median filter.
313 Dx(isnan(Dx(:))|isinf(Dx(:))) = 0;
314 Dy(isnan(Dy(:))|isinf(Dy(:))) = 0;
315
316 if FiltWidth>0,
317     Dx = medfilt2(Dx,ceil(FiltWidth*fw)*[1 1]);
318     Dy = medfilt2(Dy,ceil(FiltWidth*fw)*[1 1]);
319 end
320
321 % Find the discontinuity map (this quality map
322 % is used just when we chose WLS integration
323 % approach).
324 [Qx,Qy] = ComputeQualityMap(Dx,Dy,fw,verbose);
325 varargout{1} = Qx;
326 varargout{2} = Qy;
327
328 if verbose>1,
329      figure('name','[mt_D3D] atan(Dx) (with med filt)')
330      imshow(atan(Dx),[]), colormap jet,
331      figure('name','[mt_D3D] atan(Dy) (with med filt)')
332      imshow(atan(Dy),[]), colormap jet,
333      figure('name','[mt_D3D] Qx')
334      imshow(Qx,[]), colormap jet,
335      figure('name','[mt_D3D] Qy')
```

```
336       imshow(Qy,[]), colormap jet,
337       drawnow,
338   end
339   end
340
341   % ////////////////////////////////////////////////////
342   function [Qx,Qy] = ComputeQualityMap(Dx,Dy,fw,verbose)
343   % ////////////////////////////////////////////////////
344   [m,n] = size(Dx);
345
346   % 0) Clean Data ---------------------------
347   Qx = ones(m,n);
348   Qy = ones(m,n);
349
350   Qx(isnan(Dx) | isinf(Dx)) = 0;
351   Qy(isnan(Dy) | isinf(Dy)) = 0;
352
353   Dx2 = Dx;
354   Dy2 = Dy;
355
356   Dx2(isnan(Dx) | isinf(Dx)) = 0;
357   Dy2(isnan(Dy) | isinf(Dy)) = 0;
358   % ----------------------------------------
359
360   % 1) Find the set of outliers ------------
361   auxX = abs(Dx2 - medfilt2(Dx2, round([1 fw*2])));
362   auxY = abs(Dy2 - medfilt2(Dy2, round([fw*2 1])));
363   % ----------------------------------------
364
365   % 2) Apply morphological operations -----
366   % this is based in the idea that discontinuity regions
367   % tend to be connected and extended regions. When
368   % we find the outliers, we can miss some particular
369   % points when the discontinuity produce a shift which
370   % is multiple of the period of the fringes.
371   th2  = 2e-1;
372   w1   = 3;
373   w2   = 30;
374   w3   = w2-5;
375
376   % Mask for Qx,
377   AuxMask = auxX>th2;
378   se      = strel('square',w1);
379   AuxMask = imerode(AuxMask,se);   %imagesc(AuxMask)
380   se      = strel('square',w2);
381   AuxMask = imdilate(AuxMask,se);  %imagesc(AuxMask)
382   se      = strel('square',w3);
383   AuxMask = imerode(AuxMask,se);   %imagesc(AuxMask)
384   Qx(AuxMask) = 0;
385
386   % Mask for Qy
```

```
AuxMask = auxY>th2;
se      = strel('square',w1);
AuxMask = imerode(AuxMask,se);
se      = strel('square',w2);
AuxMask = imdilate(AuxMask,se);
se      = strel('square',w3);
AuxMask = imerode(AuxMask,se);
Qy(AuxMask) = 0;
% ------------------------------------------


if verbose>1,
    figure('name','[mt_D3D] Qmap'),
    subplot(1,2,1), imagesc(Qx);
    subplot(1,2,2), imagesc(Qy);
end
end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

### D.1.2. PS

```
%% [Disparity] = PSI(Iin,Plane,parameters);
% three-steps Phase shifting algorithm.
%
% Inputs:
%   -Iin: 1x3 cell, Iin{n} code the N image of the PS sequence
%   -Plane: 1x3 cell where P{n} (sequence for the ref. plane)
%   -parameters: struct that may contain,
%       .verbose {def 0} 1-display some text, 2- also graphics
%
% Outputs:
%   - Disparity: is prop. to surf. height, matrix of the same
%                size of Iin{1}.
%
% Refs:
%   [1] D.C. Ghiglia and Mark D. Pritt. "Two dimensional Phase
%       Unwrapping: Theory, Algorithms and Software".
%       Wile Blackwell, 1998.
%   [2] Daniel Malacara. "Optical shop testing". John Wiley &
%       Sons, 2007.
% ------------------------------------------------------------------------
% matias di martino, matiasdm@fing.edu.uy                        2014
% ------------------------------------------------------------------------

function [D] = PSI(I,P,parameters);

% load input parameters,
if isfield(parameters,'verbose');
```

```
28      verbose = parameters.verbose;
29  else % set default value,
30     verbose = 0;
31  end
32
33  % I) Surface phase ------------------------------------------------
34  % Three Step Algorithms (with shift -2pi/3,0,2pi/3)
35  % see [2] pag. 569. in particular eq. (14.46)
36  DeltaPhi    = atan( sqrt(3)*(I{1}-I{3})./(2*I{2}-I{1}-I{3}) );
37  [m n]    = size(DeltaPhi);
38  if verbose>1,
39      figure('name','[PSI] DeltaPhi'),
40      imagesc(DeltaPhi), colormap jet;
41  end
42  % Unwrapping,
43  % Use any of the codes provided on [1]:
44  D = unwrap(DeltaPhi); % this function can contain any
45                          % the methods provided in [1].
46  % ----------------------------------------------------------------
47  % II) Ref plane phase ---------------------------------------------
48  DeltaPhi    = atan( sqrt(3)*(P{1}-P{3})./(2*P{2}-P{1}-P{3}) );
49  % Unwrapping,
50  % Use any of the codes provided on [1]:
51  plane_phase = unwrap(DeltaPhi);
52  % ----------------------------------------------------------------
53
54  % III) Remove plane phase,
55  D = D - plane_phase;
56  if verbose>1,
57      figure('name','D'); imagesc(D); colormap jet;
58  end
59  end % end function,
```

### D.1.3. Takeda's method

```
1   %% [Disparity] = Takeda(Iin,Plane,parameters);
2   %
3   % Inputs:
4   %   - Iin: image with horizontal fringes
5   %   - Plane: projection over the ref. plane
6   %   - parameters: struct that may contain,
7   %       .verbose {def 0} 1-display some text, 2- also graphics
8   %
9   % Outputs:
10  %   - Disparity: is prop. to surf. height, matrix of the same
11  %                size of Iin.
12  %
13  % Refs:
```

```
14 %   [1] Mitsuo Takeda and Kazuhiro Mutoh. "Fourier transform
15 %       profilometry for the automatic measurement of 3-d
16 %       object shape". Applied Optics, 22(24):3977-3982, 1983.
17 %   [2] D.C. Ghiglia and Mark D. Pritt. "Two dimensional Phase
18 %       Unwrapping: Theory, Algorithms and Software".
19 %       Wile Blackwell, 1998.
20 % ------------------------------------------------------------------
21 % matias di martino, matiasdm@fing.edu.uy                    2014
22 % ------------------------------------------------------------------
23
24 function [D] = Takeda(I,I0,parameters);
25
26 % load input parameters,
27 if isfield(parameters,'verbose');
28     verbose = parameters.verbose;
29 else % set default value,
30     verbose = 0;
31 end
32
33 % I) Compute phase variations (following [1])
34 I  = mean(I,3);  % convert to gray if is colored image,
35 I0 = mean(I0,3); % convert to gray if is colored image,
36 [m n] = size(I);
37
38 % Define some shortcuts
39 ft2  = @(u) fftshift(fft2(u));
40 ft   = @(u) fftshift(fft(u));
41 ift2 = @(u) ifft2(ifftshift(u));
42 ift  = @(u) ifft(ifftshift(u));
43
44 % center of spectrum
45 f0x = round(n/2)+1; f0y = round(m/2)+1;
46
47 for x = 1:n, % for each column, perform fft,
48     G(:,x) = ft(I(:,x));
49     G0(:,x) = ft(I0(:,x));
50 end
51
52 %% Filter to keep just Q1
53 % find secondary peak:
54 AuxMask = ones(size(G0));
55 AuxMask(f0y+[-10:10],:) = 0; % remove DC
56
57 % find secondary peak (higher after DC value)
58 [~,inds] = max(abs(G0.*AuxMask));
59
60 % to make this more robust, keep the mode
61 fy1 = mode(inds( inds(:)<f0y ));
62
63 % filter;
64 width   = abs( (f0y-fy1) ); % filter width
```

```
AuxMask = zeros(m,n);
AuxMask(f0y+round(width/2)+[1:width],:) = 1;

G0 = G0.*AuxMask;
G  = G .*AuxMask;

% Inversion (see [1]) ———————
for x = 1:n,  % for each column, perform fft,
    g(:,x) = ift(G(:,x));
    g0(:,x) = ift(G0(:,x));
end

DeltaPhi = imag( log( g.*conj(g0) ) );
% ——————————————————————————

% II) Perform phase unwrapping
% Use any of the codes provided on [2]:
D = unwrap(DeltaPhi);  % this function can contain any
                       % the methods provided in [2].

if verbose>1,
    figure('name','D'); imagesc(D); colormap jet;
end
end  % end function
```

### D.1.4. BCS

```
%% [Disparity] = BinaryCodedStripes(Iin,Plane,parameters);
% Computes the disparity of the binary coded stripes.
%
% Inputs:
%   – Iin: 1xN cell where Iin{n} code the N bits of the
%          stripes projected over the surface.
%   – Plane: 1xN cell where P{n} code the N bits of the
%            stripes projected over the ref. plane.
%   – parameters: struct that may contain,
%       .verbose {def 0} 1–display some text, 2– also graphics
%
% Outputs:
%   – Disparity: is prop. to surf. height, matrix of the same
%                size of Iin{i}.
%
% ————————————————————————————————————————————————————————
% matias di martino, matiasdm@fing.edu.uy                2014
% ————————————————————————————————————————————————————————

function [D] = BinaryCodedStripes(Ic,Pc,parameters);
%% I) Recover coded stripes ——————————
```

```
22 N       = length(Ic);
23 [m n]   = size(Ic{1});
24 P       = zeros(m,n);
25 I       = zeros(m,n);
26
27 % I.a binarization,
28 for k = 1:N;
29     Ic{k} = mt_Normalize(Ic{k},[0 1]);
30     th    = graythresh(Ic{k});
31     Ic{k} = Ic{k}>th;
32     Pc{k} = mt_Normalize(Pc{k},[0 1]);
33     th    = graythresh(Pc{k});
34     Pc{k} = Pc{k}>th;
35 end
36 % I.b decodification
37 for k = 1:N;
38     P = P + 2^(N-k) * Pc{k};
39     I = I + 2^(N-k) * Ic{k};
40 end
41 % ----------------------------------------
42
43 %% II) Find correspondences ------------
44 max_d = 70;  % half of the width of the "search window"
45 % add borders;
46 I = [zeros(m,max_d) I zeros(m,max_d)];
47 P = [zeros(m,max_d) P zeros(m,max_d)];
48 D = zeros(m,n);  % initialization;
49 for i = 1:m;
50     for j = 1:n;
51         j2     = j+max_d;
52         aux    = P(i,j2+[-max_d:max_d]);
53         [~,d]  = min(abs(I(i,j2)-aux));
54         D(i,j) = d-max_d;
55     end
56 end
57
58 if verbose>1,
59     figure('name','D'); imagesc(D); colormap jet;
60 end
61 % ----------------------------------------
```

## D.2. Integration techniques

### D.2.1. LS integration approach

```
1 % Least Squares Integration method
2 %
```

```
% Inputs:
%   - gx (mxnx1): estimated x-partial derivative,
%   - gy (mxnx1): estimated y-partial derivative,
%
% Outputs:
%   - Z (mxnx1) retrieved surface
%
% ------------------------------------------------------------
% Matias Di Martino (c)                              2014
%                                      matiasdm@fing.edu.uy
% ------------------------------------------------------------

function Z = LS_integration(gx,gy)

[H,W] = size(gx); p = H*W;

% Define Dx and Dy matrices such that
% {d(U)/di}(:) = Di*U(:), i = x,y,
N                               = (H+2)*(W+2);
mask                            = zeros(H+2,W+2);
mask(2:end-1,2:end-1)           = 1;
idx                             = find(mask==1);

Dx = 1/2 * (sparse(idx,idx+(H+2),1,N,N) ...
          - sparse(idx,idx-(H+2),1,N,N));
Dy = 1/2 * (sparse(idx,idx+1    ,1,N,N) ...
          - sparse(idx,idx-1    ,1,N,N));
Dx = Dx(idx,idx); Dy = Dy(idx,idx);
Dx = Dx   - sparse(1:p,1:p,sum(Dx,2) ,p,p);
Dy = Dy   - sparse(1:p,1:p,sum(Dy,2) ,p,p);

% Laplacial operator
L  = sparse(idx,idx,     -4,N,N)  ...
   + sparse(idx,idx+1,    1,N,N) ...
   + sparse(idx,idx-1,    1,N,N) ...
   + sparse(idx,idx+(H+2),1,N,N) ...
   + sparse(idx,idx-(H+2),1,N,N);
L  = L(idx,idx);
L  = L     - sparse(1:p,1:p,sum(L ,2) ,p,p);

% ////////////////////////////
% // Poisson Eq //////////////
% ////////////////////////////
% Eqs Zxx + Zyy = div(gx,gy)
A = L;
b = Dx*gx(:) + Dy*gy(:);

% ////////////////////////////
% // Border Conditions, /////
% ////////////////////////////
Abord  = []; b_bord = [];
```

```
54
55 % Eq Zx-gx = 0 on x = 1, x = n,
56 Abord  = [Abord; Dx([1:m],:)];
57 b_bord = [b_bord; gx(transpose(1:m))];
58 Abord  = [Abord; Dx(m*(n-1) + [1:m],:)];
59 b_bord = [b_bord; gx(m*(n-1) + transpose(1:m) )];
60
61 % Eqs Zy-gy = 0 on y = 1, y = m,
62 Abord  = [Abord; Dy( ([1:n] - 1) * m+1,:)];
63 b_bord = [b_bord; gy((transpose(1:n)-1)*m+1)];
64 Abord  = [Abord; Dy( [1:n]*m,:)];
65 b_bord = [b_bord; gy( transpose(1:n)*m )];
66
67 % Add border conditions to the linear system of equations,
68 A = [A; Abord]; b = [b; b_bord];
69
70
71 % ////////////////////////////
72 % // Additional constraints /
73 % ////////////////////////////
74 % the integration process is an Ill-Posed problem,
75 % we have a set of surfaces that differ by a constant
76 % and are solution of the system. This causes that
77 % rank(A) = mn-1. We can solve this issue by adding an
78 % additional constrain, e.g. looking for the solution
79 % with zero mean. Other option is arbitrary set the
80 % value of some pixel.
81 A = [A; ones(1,mn) ]; b = [b; 0];
82
83 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%
84 % Finally, solve the linear (sparse) problem
85 x = A\b;
86 % Other option to solve the ill-posed condition
87 % of the problem removing one unknown and solving
88 % x = A(:,2:end)\b;
89 % x = [0; x];
90 Z = reshape(x,[m n]);
91 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%
92 end % end function
```

### D.2.2. FC integration approach

```
1 % Frankot-Chellapa
2 %
3 % Inputs:
4 %   - gx (mxnx1): estimated x-partial derivative,
5 %   - gy (mxnx1): estimated y-partial derivative,
6 %
```

```
7  % Outputs:
8  %   - Z (mxnx1) retrieved surface
9  %
10 % Refs:
11 %  [1] Fankot and Chellappa. "A method for enforcing
12 %      integrability in shape from shading algorithms"
13 %      TPAMI, 1988.
14 %  [2] Agrawal et. al. "What is the range of surface
15 %      reconstructions from a gradient field? ".
16 %      ECCV 2006.
17 % ------------------------------------------------------------
18 % Matias Di Martino (c)                                  2014
19 %                                      matiasdm@fing.edu.uy
20 % ------------------------------------------------------------
21
22 function Z = FC_integration(gx,gy)
23
24 [m,n] = size(gx);
25
26 % Define frequency domain,
27 [wx,wy] = meshgrid(0:n-1,0:m-1);
28 wx = 2*pi*wx/n;
29 wy = 2*pi*wy/m;
30
31 % Gradient fourier transform,
32 Gx = fft2(gx);
33 Gy = fft2(gy);
34
35 % FT(Z) = (ax* Gx + ay* Gy) ./ ( |ax|^2 + |ay|^2 );
36 % Assuming we approximate the derivatives by finite
37 % central differences [1]:
38 % ax = (1/2) e^{j wx} - (1/2) e^{-j wx} = j sin(wx)
39 % ay = (1/2) e^{j wy} - (1/2) e^{-j wy} = j sin(wy)
40 j  = sqrt(-1);
41 ax = j * sin(wx); ay = j * sin(wy);
42
43 FTZ      = (conj(ax).*Gx + conj(ay).*Gy ) ...
44            ./ ( abs(ax).^2 + abs(ay).^2 );
45
46 % Set zero component (which is undefined due to
47 % 0/0 division) ,
48 FTZ(1,1) = 0;
49 Z = real(ifft2(FTZ));
50 end % end function
```

### D.2.3. L1 integration approach

```
1  % L1-error minimization,
```

```
2  %
3  % Inputs:
4  %   - gx (mxnx1): estimated x-partial derivative,
5  %   - gy (mxnx1): estimated y-partial derivative,
6  %
7  % Outputs:
8  %   - Z (mxnx1) retrieved surface
9  %
10 % Refs:
11 %  [1] Zhouyu Du et al. "Robust surface reconstruction
12 %      from gradients field using the L1 norm".
13 %      Digital Image Computing Techniques and
14 %      Applications, IEEE, 2007.
15 % ------------------------------------------------------------
16 % Matias Di Martino (c)                              2014
17 %                                      matiasdm@fing.edu.uy
18 % ------------------------------------------------------------
19
20 function Z = L1_integration(gx,gy)
21
22 [m,n]       = size(gx);
23 mn          = m*n;
24
25 % we want to minimize Sum fi * xi;
26 % the vector xi contains in the first mn
27 % entries Z, then u and v. To impose
28 % min {Sum (ui + vi)} f must be defined as:
29 f           = zeros(3*mn,1);
30 f(mn+1:end) = 1;
31
32 % Define Dx and Dy operators,
33 se = speye(mn);
34 Dx = 1/2 * ( circshift(se,[0 m]) - circshift(se,[0 -m]) );
35 Dy = 1/2 * ( circshift(se,[0 1]) - circshift(se,[0 -1]) );
36
37 % Now we set the set of inequalities, as
38 % A*x < b.
39 A = speye(6*mn,3*mn);
40 b = zeros(6*mn,1);
41
42 for i = 1:mn,
43     % Z_x - p < u
44     % ==> 1/2 (Zij+1 - Zij-1) - uij < pij
45     i2 = i;
46     A(i2,1:mn)    = Dx(i,:);
47     A(i2,i+mn)    = -1;
48     b(i2)         =  gx(i);
49
50     % Z_x - p > -u
51     % ==> -1/2 (Zij+1 - Zij-1) - uij < -pij
52     i2 = i+mn;  % Stack after the previous mn eqs.
```

```
53      A(i2,1:mn)     =  -Dx(i,:);
54      A(i2,i+mn)     =  -1;
55      b(i2)          =  -gx(i);
56
57      % Z_y - q < v
58      % ==> 1/2 (Zi+1j - Zi-1j) - vij < qij
59      i2 = i+2*mn;  % Stack after the previous 2mn eqs.
60      A(i2,1:mn)     = Dy(i,:);
61      A(i2,i+2*mn)   = -1;
62      b(i2)          =  gy(i);
63
64      % Z_y - q > -v
65      % ==> -1/2 (Zi+1j - Zi-1j) - vij < -qij
66      i2 = i+3*mn;  % Stack after the previous 3mn eqs.
67      A(i2,1:mn)     =  -Dy(i,:);
68      A(i2,i+2*mn)   =  -1;
69      b(i2)          =  -gy(i);
70
71      % u > 0 ==> - uij < 0
72      i2 = i+4*mn;  % Stack after the previous 4mn eqs.
73      A(i2,i+mn)     = -1;
74      b(i2)          = 0;
75
76      % v > 0 ==> - vij < 0
77      i2 = i+5*mn;  % Stack after the previous 5mn eqs.
78      A(i2,i+2*mn)   = -1;
79      b(i2)          = 0;
80  end
81
82  x = linprog(f,A,b);  % minimize sum(fi*xi) st. Ax<b
83
84  % keep the first mn elements, and reshape it.
85  Z      = reshape(x(1:mn),[m n]);
86
87  end % end function
```

### D.2.4. M integration approach

```
1   % M-estimator integration approach,
2   %
3   % this code is based on the source code provided
4   % by Amit Agrawal, 2006
5   % http://www.umiacs.umd.edu/~aagrawal/
6   % Permitted for personal use and research purpose only
7   % Refer to the following citations:
8   %
9   %   1.  A. Agrawal, R. Raskar and R. Chellappa, "What
10  %       is the Range of Surface Reconstructions
```

```
11 %       from a Gradient Field? European Conference on
12 %       Computer Vision (ECCV) 2006
13 %   2.  A. Agrawal, R. Chellappa and R. Raskar,
14 %       "An Algebraic approach to surface reconstructions
15 %        from gradient fields? International Conference
16 %        on Computer Vision (ICCV) 2006
17 %
18 %=========================================================
19 % Inputs:
20 %   - gx (mxnx1): estimated x-partial derivative,
21 %   - gy (mxnx1): estimated y-partial derivative,
22 %
23 % Outputs:
24 %   - Z (mxnx1) retrieved surface
25 %
26 % ------------------------------------------------------------
27 % Matias Di Martino                                    2014
28 %                                    matiasdm@fing.edu.uy
29 % ------------------------------------------------------------
30
31 function Z1 = M_integration(gx,gy)
32
33 [m,n] = size(gx); mn = m*n;
34
35 % Use LS solution as seed,
36 Z0    = WLS_integration(gx,gy,ones(m,n),ones(m,n));
37 Z0    = Z0 - mean(Z0(:));
38
39 % ////////////////////////////////////////////
40 % chose weight function ///////////////////////
41 % ////////////////////////////////////////////
42 option = 1; %{ 1 - Huber, 2 - Fair, 3 - Cauchy }
43 switch option,
44     case 1,  % Huber,
45         k   = 1.345;
46         rho = @(e) (abs(e)<=k) .* 1 + ...
47                    (abs(e)>k)  .* (k ./ (abs(e)+eps) );
48     case 2,  % Fair,
49         c   = 1.3998;
50         rho = @(e) 1 ./ ( 1 + abs(e)/c );
51     case 3,  % Cauchy,
52         c   = 1.4;
53         rho = @(e) 1 ./ (1 + (e/c).^2 );
54 end
55
56 % Initialization,
57 tol      = 1e-6;
58 diff     = 2*tol+1;
59 iter     = 0;
60 max_iter = 50;
61 dx       = @(U) [U(:,2:end)-U(:,1:end-1)  0*U(:,1)];
```

```
62  dy        = @(U) [U(2:end,:)-U(1:end-1,:); 0*U(1,:)];
63  wx        = rho(gx-dx(Z0));
64  wy        = rho(gy-dy(Z0));
65
66  % ////////////////////////////////////////
67  % // Begin iteration ////////////////////
68  % ////////////////////////////////////////
69  while diff>tol && iter<max_iter;
70      Z1        = WLS_integration(gx,gy,wx,wy);
71      Z1        = Z1 - mean(Z1(:));
72      wx        = rho(gx-dx(Z1));
73      wy        = rho(gy-dy(Z1));
74      diff      = mean(abs(Z1(:)-Z0(:)));
75      Z0        = Z1;
76      iter      = iter+1;
77  end
78
79  end % end function,
80
81  % %%%%
```

```
113     disp('All weights are one in diffusion tensor');
114 end
115
116 D21 = D12;
117 D11 = padarray(D11,[1 1],0,'both');
118 D12 = padarray(D12,[1 1],0,'both');
119 D21 = padarray(D21,[1 1],0,'both');
120 D22 = padarray(D22,[1 1],0,'both');
121
122 N = (H+2)*(W+2);
123 mask = zeros(H+2,W+2);
124 mask(2:end-1,2:end-1) = 1;
125 idx = find(mask==1);
126
127 A = sparse(idx,idx+1,-D22(idx),N,N);
128 A = A + sparse(idx,idx+H+2,-D11(idx),N,N);
129 A = A + sparse(idx,idx-1,-D22(idx-1),N,N);
130 A = A + sparse(idx,idx-H-2,-D11(idx-H-2),N,N);
131
132 A = A + sparse(idx,idx+1,-D12(idx),N,N);
133 A = A + sparse(idx,idx-H-2,-D12(idx-H-2),N,N);
134 A = A + sparse(idx,idx-H-2+1,D12(idx-H-2),N,N);
135 A = A + sparse(idx,idx+H+2,-D21(idx),N,N);
136 A = A + sparse(idx,idx-1,-D21(idx-1),N,N);
137 A = A + sparse(idx,idx-1+H+2,D21(idx-1),N,N);
138
139 A = A(idx,idx);
140 N = size(A,1);
141 dd = sum(A,2);
142 idx = [1:N]';
143 A = A + sparse(idx,idx,-dd,N,N);
144 end
145
146 % // f tensor ////////////////////////
147 function [f] = calculate_f_tensor(gx,gy,d11,d12,d21,d22)
148 [H,W] = size(gx);
149 gx(:,end) = 0;
150 gy(end,:) = 0;
151
152 if(~(exist('d11','var') & exist('d12','var') ...
153    & exist('d21','var') & exist('d22','var')))
154     disp('Weights are all zeros')
155     d11 = ones(H,W);
156     d21 = d11;
157     d12 = d11;
158     d22 = d11;
159 end
160
161 d21 = d12;
162 gx1 = gx.*d11;
163 gy1 = gy.*d22;
```

```
164 gx1 = padarray(gx1,[1 1],0,'both');
165 gy1 = padarray(gy1,[1 1],0,'both');
166 gxx = zeros(size(gx1)); gyy = gxx;
167 j = 1:H+1;
168 k = 1:W+1;
169 % Laplacian
170 gyy(j+1,k) = gy1(j+1,k) - gy1(j,k);
171 gxx(j,k+1) = gx1(j,k+1) - gx1(j,k);
172 f = gxx + gyy;
173 f = f(2:end-1,2:end-1);
174 clear gx1 gy1 gxx gyy
175
176 gx1 = gx.*d12;
177 gy1 = gy.*d21;
178
179 gx2 = gy.*d12;
180 gy2 = gx.*d21;
181
182 gx2(end,:) = gx1(end,:);
183 gy2(:,end) = gy1(:,end);
184
185 gx2(:,end) = 0;
186 gy2(end,:) = 0;
187
188 gx2 = padarray(gx2,[1 1],0,'both');
189 gy2 = padarray(gy2,[1 1],0,'both');
190 gxx = zeros(size(gx2)); gyy = gxx;
191 j = 1:H+1;
192 k = 1:W+1;
193
194 % Laplacian
195 gyy(j+1,k) = gy2(j+1,k) - gy2(j,k);
196 gxx(j,k+1) = gx2(j,k+1) - gx2(j,k);
197 f2 = gxx + gyy;
198 f2 = f2(2:end-1,2:end-1);
199
200 f = f + f2;
201 end
```

### D.2.5. AD integration approach

```
1 % Anisotropic diffusion integration approach,
2 %
3 % Inputs:
4 %   - gx (mxnx1): estimated x-partial derivative,
5 %   - gy (mxnx1): estimated y-partial derivative,
6 %
7 % Outputs:
```

```
8  %   - Z (mxnx1) retrieved surface
9  %
10 % Refs:
11 %   [1] Gilles Aubert, Pierre Kornprobst,
12 %       "Mathematical Problems In Image Processing".
13 %       Springer, sec. edition
14 %   [2] A. Agrawal. "Scene analysis under variable
15 %       illumination using gradient domain methods".
16 %       PhD Thesis, University of Mariland 2006.
17 %   [3] Joachim Weickert. "Anisotropic Diffusion in
18 %       Image Processing". Published by B. G.
19 %       Teubner Stuttgart. 2008.
20 %   [1] Agrawal et. al. "What is the range of surface
21 %       reconstructions from a gradient field? ".
22 %       ECCV 2006.
23 % ------------------------------------------------------------
24 % Matias Di Martino (c)                              2014
25 %                                      matiasdm@fing.edu.uy
26 % ------------------------------------------------------------
27
28 function Z = AD_integration(gx,gy)
29
30 % Set some parameters,
31 verbose     = 2;
32 KernelSigma = 1/2;  % Sigma of the gaussian kernel
33                     % used to compute the smoothed
34                     % gradient field from which
35                     % the tensor is estimated.
36
37 [m,n] = size(gx);
38 mn    = m*n;
39
40 % Calculate diffusion tensor
41 [d11, d12, d22] = ...
42     CalculateDiffusionTensor(gx,gy,verbose,KernelSigma);
43
44 A = laplacian_matrix_tensor(m,n,d11,d12,d12,d22);
45 b = calculate_f_tensor(gx,gy,d11,d12,d12,d22);
46
47 b = b(:);  % convert to column vector
48
49 x = A(:,2:end)\b;  % remove one variable to make
50                    % the solution unique (we have
51                    % a family of functions differing
52                    % on a constant
53 x = [0;x];  % arbitrary set as 0 the fist pixel.
54
55 Z = reshape(x,[m n]);  % reshape the vector solution
56                        % to the original shape.
57
58 end % end function
```

```
59
60  % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61  % Aux functions, %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
62  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63  % ////////////////////////////////////////////
64  function [D11, D12, D22] = ...
65      CalculateDiffusionTensor(gx,gy,verbose,sigma);
66  % ////////////////////////////////////////////
67  if verbose>0,
68      tic, fprintf('\t Computing diffusion tensors '),
69  end
70
71  [m,n] = size(gx);
72
73  % Set tensor parameters ------------------------------
74  KernelSigma = sigma;      %0.5 is a typical value;
75  KernelWidth = max(3,round(2*KernelSigma));
76  Beta        = 0.02;  % see ref [2]
77  % ----------------------------------------------------
78
79  GaussianKernel = fspecial('gaussian',KernelWidth,KernelSigma);
80
81  H11 = zeros(m,n); H12 = zeros(m,n); H22 = zeros(m,n);
82  H11(:,:)     = conv2(gx(:,:).^2,GaussianKernel,'same');
83  H12(:,:)     = conv2(gx(:,:).*gy(:,:),GaussianKernel,'same');
84  H22(:,:)     = conv2(gy(:,:).^2,GaussianKernel,'same');
85
86  D11 = zeros(m,n); D12 = zeros(m,n); D22 = zeros(m,n);
87  % compute largest eigenvalue,
88  mu1 = 1/2*( H11+H22 + sqrt( (H11-H22).^2 + 4 * H12.^2 ) );
89  th  = max(mu1(:))/100;  % this small threshold is
90                          % computed as the tenth of the mean
91                          % of H11+H22 which is in the order
92                          % of magnitude of the eigenvalues
93                          % of matrix H.
94
95  for i = 1:m,
96      for j = 1:n,
97          AuxH = [H11(i,j) H12(i,j); H12(i,j) H22(i,j)];
98          [V,D] = eigs(AuxH);
99          % return the eigenvalues in the diagonal of D
100         % in descendant order of magnitude and they
101         % corresponding eigenvectors as columns in V.
102         %SANITY CHECK ~~~~~~~~~~~~~~~~~~~~~~
103         if D(2,2)>D(1,1),
104             error('there is an error in eigenvalues')
105         end
106         % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
107         % modify the diagonal according to ([2],pag 55)
108         D(2,2) = 1;  % the smallest eigenvalue (mu2)
109                      % is modify by 1 the largest eigenvalue
```

```
                        % (mu1) is set to 1 if it is small
        if D(1,1) < th,
            D(1,1) = 1;
        else % if it is ">0" is modified according [1] pag 115:
            D(1,1) = Beta + 1 - exp( -3.315 / (D(1,1))^4 );
        end

        auxD = V*D*V';
        D11(i,j) = auxD(1,1);
        D12(i,j) = auxD(1,2);
        D22(i,j) = auxD(2,2);
    end
end
% release memory
clear auxD auxH D V
end

function [A] = laplacian_matrix_tensor(H,W,D11,D12,D21,D22)
% See M integration approach
end

function [f] = calculate_f_tensor(gx,gy,d11,d12,d21,d22)
% See M integration approach
end
```

### D.2.6. Integration approach constraining scene temporal variation

```
%% [I] = Int_Const_SceneVariation(Ix,Iy,parameters)  %
% this function returns the integral a video sequence,
% from it spatial gradients Ix,Iy. Input images are of
% size mxnxp where p is the number of frames of the video.
% The integration is performed taking into account the
% temporal continuity between consecutive frames.
%
% Input:
%   - Ix (mxnxp) [double image] (x-partial derivative)
%   - Iy (mxnxp) [double image] (y-partial derivative)
%   - parameters: struct that may contain
%       .verbose [def 0] {1-display some text,
%                         2- also some graphics}
%       .Lambda  [def 0] set the "time" regularization
%                        weight.
%       .Type [def Definition]
%           *LS:   (Least Squares)
%           *WLS:  (Weighted Least Squares)
%           *AD:   (see [2, pag 50-60],
%                       [1, pag 113-117] and [3]
%       .TemporalOrder [def 1] Set the order of the
```

```
22 %                              temporal regularization
23 %           *1: we minimize |du/dt|
24 %           *2: we minimize |ddu/dtdt| and so on.
25 %
26 % Output:
27 %   - I (mxnxp)
28 %
29 % refs:
30 %   [1] Gilles Aubert, Pierre Kornprobst,
31 %       "Mathematical Problems In Image Processing".
32 %        Springer, sec. edition
33 %   [2] A. Agrawal. "Scene analysis under variable
34 %       illumination using gradient domain methods".
35 %       PhD Thesis, University of Mariland 2006.
36 %   [3] Joachim Weickert. "Anisotropic Diffusion in
37 %       Image Processing". Published by B. G.
38 %       Teubner Stuttgart. 2008.
39 %
40 % ------------------------------------------------------
41 % Matias Di Martino (c)                          2014
42 % matiasdm@fing.edu.uy                  (20/02/2014)
43 % ------------------------------------------------------
44
45 function I = Int_Const_SceneVariation(Ix,Iy,p);
46 %% Load and set general parameters, %%%%%%%%%%%%%%%%%%%%%
47 [m n k] = size(Ix);
48 list    = {'verbose','Type','Lambda','TemporalOrder'}
49 defval = {0        ,'LS'  ,1       ,1              }
50 [verbose,Type,Lambda,,TemporalOrder] ...
51             = GetParameters(parameters,list,defval);
52
53 %% Integration %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54 switch Type,
55     % ////////////////////////////////////////
56     case 'LS'  % ///////////////////////////////////
57     % ////////////////////////////////////////////
58     % to integrate we are minimizing the energy functional:
59     %   E[u(x,y,t)] = (1-Lambda)[||du/dx - gx||2
60     %      + ||du/dy - gy||2] + Lambda ||d^{o}u/dt^{o}||2
61     % where gx(x,y,t) and gy(x,y,t) are the know functions
62     % ans o the Temporal Order.
63     %
64     % The E-L equation of this problem is,
65     %  (1-Lambda)[u_xx + u_yy] + (-1)^{o-1} Lambda u_{tt}^o =
66     %              (1-Lambda)[gxx+gyy].                    (1)
67     % In this case we solve Eq. (1) by rewriting it as
68     % a linear problem in the form of Au(:) = b
69     % ------------------------------------------------
70     Wx  = ones(size(Ix)); Wy = Wx;
71     [I] = WLS(Ix,Iy,Wx,Wy,verbose,Lambda,TemporalOrder);
72
```

```
73      % ////////////////////////////////////////
74      case 'WLS'   % ///////////////////////////////
75      % ////////////////////////////////////////
76      % to integrate we are minimizing the energy functional:
77      %   E[u(x,y,t)] = int{(1-Lambda)[Wx (du/dx - gx)2
78      %    + Wy (du/dy - gy)2] + Lambda ||d^{o}u/dt^{o}||2
79      %                    }dxdydt
80      % where gx(x,y,t) and gy(x,y,t) are the know functions
81      % o is the Temporal Order and Wx-y the gx and gy weights.
82      %
83      % The E-L equation of this problem is, (see [2] pag 48)
84      %
85      % (1-Lambda)[(Wx*u_x)_x + (Wy*u_y)_y] + (-1)^{o-1}
86      % Lambda u_{tt}^o = (1-Lambda)[(Wx*gx)_x + (Wy*gy)_y].   (1)
87      % In this case we solve Eq. (1) by rewriting it as
88      % a linear problem in the form of
89      %   Au(:) = b   % WARNING, depending on Wx and Wy
90      % matrix A can become singular
91      % ------------------------------------------------------
92      [I] = WLS(Ix,Iy,p.Wx,p.Wy,verbose,Lambda,TemporalOrder);
93
94      % ////////////////////////////////////////////////
95      case 'AD'         % ////////////////////////////////
96      % ////////////////////////////////////////////////
97      % to integrate we are minimizing the energy functional:
98      %   E[u(x,y,t)] = int{ (1-Lambda)[d11 (du/dx - gx)2
99      %                 + (d12+d21)*(du/dx - gx)(du/dy - gy)
100     %                 +    d22 (du/dy - gy)2]
101     %                 + Lambda ||d^{o}u/dt^{o}||2
102     %                    }dxdydt
103     % where gx(x,y,t) and gy(x,y,t) are the know functions,
104     % o is the Temporal Order and D the diffusion tensor
105     % (see [2, pag 50-60] and [1, pag 113-117]).
106     %
107     % The E-L equation of this problem is,
108     % (1-Lambda)[(d11 u_x + d12 u_y)_x
109     % + (d12 u_x + d22 u_y)_y] + (-1)^{o-1} Lambda u_{tt}^o =
110     % (1-Lambda)[(d11 gx + d12 gy)_x + (d12 gx + d22 gy)_y] (1)
111     %
112     % In this case we solve Eq. (1) by rewriting it as
113     % a linear problem in the form of
114     %   Au(:) = b
115     % ------------------------------------------------------
116       [I] = AD(Ix,Iy,verbose,Lambda,TemporalOrder);
117     otherwise,
118         error('[mt_ImIntegrate] Type Unknown')
119 end
120
121 end %end function
122
123 % %%%%
```

```
124 % Aux functions, %%%%
```

```
175
176 % Generate sparse matrices with the diffusion coefficients
177 % on the diagonal
178 D11  = sparse(1:mnk,1:mnk,d11(:),mnk,mnk);
179 D12  = sparse(1:mnk,1:mnk,d12(:),mnk,mnk);
180 D22  = sparse(1:mnk,1:mnk,d22(:),mnk,mnk);
181
182 % compute the diffusion laplacian operator
183 o = TemporalOrder;
184 L = (1-lambda)*( Dx*(D11*Dx+D12*Dy) + Dy*(D12*Dx+D22*Dy) ) ...
185     + (-1)^(o-1)*lambda*(Dt*Dt)^o;
186
187 % compute the data term
188 b = (1-lambda)*( Dx*(D11*gx(:)+D12*gy(:)) ...
189     + Dy*(D12*gx(:)+D22*gy(:)) );
190
191 b = b(:);  % convert to column vector
192 % Know we must solve the linear problem L u = b
193 u = L\b;
194 % reshape the vector solution to the original shape.
195 u = reshape(u,[m n k]);
196 end % end AD
197
198 %% ////////////////////////////////////////////////////
199 function [D11, D12, D22] = ...                          % //
200      CalculateDiffusionTensor(gx,gy,verbose);           % //
201 % ////////////////////////////////////////////////////
202 % Is the same procedure explained in
203 % "AD integration approach" applied frame by frame.
204 end % end CalculateDiffusionTensor.
```

### D.2.7. Integration approach constraining objects velocity

```
1  %% [Z,[vx],[vy]] = Int_Const_ObjectMovement(gx,gy,[parameters])
2  %
3  % Least Squares Integration method using temporal
4  % regularization,
5  %
6  % Inputs:
7  %   - gx (HxWxT): estimated x-partial derivative,
8  %   - gy (HxWxT): estimated y-partial derivative,
9  %   - parameters: (OPTIONAL) struct that may contain
10 %       .lambda: weight of temporal reg
11 %       .mu: smoothness of vx and vy fields
12 %       .algOp:
13 %           0 - process each frame indep.
14 %           1 - uses input scalars vx and vy
15 %           2 - estimates scalar vx and vy
```

```
16 %           3 - estimates fields vx and vy
17 %       .vx: (scalar) velocity along x axis
18 %       .vy: (scalar) velocity along y axis
19 %            vx and vy must be provided just for
20 %            algOp = 1.
21 %
22 % Outputs:
23 %   - Z (HxWxT) retrieved surface
24 %   - vx (optional)
25 %   - vy (optional)
26 %
27 % Refs:
28 %   [1] M. Di Martino, A. Fernandez and J. Ferrari
29 %       "Gradient domain methods with application to
30 %        4D scene reconstruction". Optics and Lasers
31 %        in Engineering, 2014
32 %   [2] Agrawal et. al. "What is the range of surface
33 %       reconstructions from a gradient field? ".
34 %       ECCV 2006.
35 %
36 % ------------------------------------------------------------
37 % Matias Di Martino (c)                              2014
38 %                                      matiasdm@fing.edu.uy
39 % ------------------------------------------------------------
40
41 function [Z,varargout] = ...
42               Int_Const_ObjectMovement(gx,gy,varargin)
43
44 % ////////////////////////////////////////////
45 % // Load and set general parameters //////////
46 % ////////////////////////////////////////////
47 if nargin>2, par = varargin{1}; else p = []; end,
48 % list of input par.
49 list    = {'lambda','mu','algOp','vx','vy'};
50 % list of def. values.
51 defval = {.1      ,5   ,2      ,0   , 0};
52 [lambda,mu,algOp,vx,vy] = GetParameters(par,list,defval);
53 [H,W,T] = size(gx);
54 % set some derivation operators as global
55 % as they are used inside various algorithms.
56 global Dx Dy Dt L L_lambda;
57 % ------------------------------------------------------
58
59 % Now perform the integration following
60 % the method indicated by algOp,
61 switch algOp,
62     % ////////////////////////////////////////////
63     case 0,  % Process frame by frame indep.        ///
64     % ////////////////////////////////////////////
65     % this methods follows standard LS approach
66     % frame by frame,
```

```
67          Z = IntegrateEachFrame(gx,gy);
68
69      % ////////////////////////////////////////////
70      case 1,  % impose temporal constraint with    ///
71               % known velocity.                     ///
72      % ////////////////////////////////////////////
73      % In this case, Z is the dynamic surface that
74      % minimizes
75      %
76      % E1[u] = int{ (u_x-gx)^2 + (u_y-gy)^2 ...
77      %            + lambda ( u_t + (vx*gx + vy*gy))^2
78      %            }dxdydt
79      % whose euler lagrange eq. is:
80      %    u_xx + u_yy + lambda u_tt = ...
81      %    g_xx + g_yy - lambda (vx*gx+vy*gy)_t
82      %
83      % if we use vx = 0 and vy = 0 this approaches
84      % is a particular case of the techniques presented
85      % in ref [1].
86          DefineLapAndDiOperators(H,W,T,lambda);
87          Z = IntegrateGiven_vxvy(gx,gy,lambda,vx,vy);
88
89      % ////////////////////////////////////////////
90      case 2,  % impose temporal constraint with    ///
91      % unknown (but uniform) velocity.            ///
92      % ////////////////////////////////////////////
93      % In this case, Z, vx and vy are those who
94      % minimizes
95      %
96      % E1[u,vx,vy] = int{ (u_x-gx)^2 + (u_y-gy)^2 ...
97      %             + lambda ( u_t + (vx*gx + vy*gy))^2
98      %                  }dxdydt
99      % whose euler lagrange eqs. are:
100     % (ai) u_xx + u_yy + lambda u_tt ...
101     %      + lambda (vx*gx+vy*gy)_t = g_xx + g_yy;
102     % (b)  gx'*u_t + gx'*gx * vx + gx'*gy * vy = 0
103     % (c)  gy'*u_t + gy'*gx * vx + gy'*gy * vy = 0
104         DefineLapAndDiOperators(H,W,T,lambda);
105         [Z,vx,vy] = IntegrateAndEtimate_vxvy(gx,gy,lambda);
106
107     % ////////////////////////////////////////////
108     case 3,  % impose temporal constraint with    ///
109     % a unknown velocity field.                   ///
110     % ////////////////////////////////////////////
111     % In this case, Z, vx and vy are those who
112     % minimizes
113     %
114     % E1[u,vx,vy] = int{ (u_x-gx)^2 + (u_y-gy)^2 ...
115     %             + lambda ( u_t + (vx*gx + vy*gy))^2
116     %             + mu * [|grad(vx)|^2 + |grad(vy)|^2]}dxdydt
117     % whose euler lagrange eqs. are:
```

```
118      % (ai) u_xx + u_yy + lambda u_tt ...
119      %      + lambda (vx.*gx+vy.*gy)_t = g_xx + g_yy;
120      % (bi) lambda* gx( u_t + gx.*vx + gy.*vy)- mu lap(vx) = 0
121      % (ci) lambda* gy( u_t + gx.*vx + gy.*vy)- mu lap(vy) = 0
122          DefineLapAndDiOperators(H,W,T,lambda);
123          [Z,vx,vy] = IntegrateAndEtimate_VXVY(gx,gy,lambda,mu);
124      otherwise
125          error('algOp value is invalid.')
126  end
127
128  % set additional optional outputs.
129  varargout{1} = vx; varargout{2} = vy;
130  end % end function
131
132  % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
133  % Aux functions, %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
134  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
135  % /////////////////////////////////////////////////
136  function [varargout] = ...                    % ///////
137      GetParameters(parameters,list,defval)  % ///////
138  % /////////////////////////////////////////////////
139  % see D3D code.
140  end % end GetParameters
141
142  % /////////////////////////////////////////////////
143  function DefineLapAndDiOperators(H,W,T,lambda) % //
144  % /////////////////////////////////////////////////
145  p = H*W*T;
146  global Dx Dy Dt L L_lambda;
147
148  % Define Dx and Dy matrices such that
149  % {d(U)/di}(:) = Di*U(:), i = x,y,
150  N                               = (H+2)*(W+2)*(T+2);
151  mask                            = zeros(H+2,W+2,T+2);
152  mask(2:end-1,2:end-1,2:end-1)   = 1;
153  idx                             = find(mask==1);
154
155  Dx = 1/2 * ( sparse(idx,idx+(H+2),1,N,N) ...
156             - sparse(idx,idx-(H+2),1,N,N) );
157  Dy = 1/2 * ( sparse(idx,idx+1    ,1,N,N) ...
158             - sparse(idx,idx-1    ,1,N,N) );
159  Dt = 1/2 * ( sparse(idx,idx+(H+2)*(W+2),1,N,N)...
160             - sparse(idx,idx-(H+2)*(W+2),1,N,N) );
161  Dx = Dx(idx,idx); Dy = Dy(idx,idx); Dt = Dt(idx,idx);
162  Dx       = Dx       - sparse(1:p,1:p,sum(Dx,2)      ,p,p);
163  Dy       = Dy       - sparse(1:p,1:p,sum(Dy,2)      ,p,p);
164  Dt       = Dt       - sparse(1:p,1:p,sum(Dt,2)      ,p,p);
165
166  % Define Lap Op.
167  L  = sparse(idx,idx      ,-4,N,N)  ...
168     + sparse(idx,idx+1    ,1,N,N) ...
```

```
169     + sparse(idx,idx-1    ,1,N,N) ...
170     + sparse(idx,idx+(H+2),1,N,N) ...
171     + sparse(idx,idx-(H+2),1,N,N);
172
173 L_lambda = L + lambda * ...
174               ( sparse(idx,idx,-2,N,N) ...
175               + sparse(idx,idx+((H+2)*(W+2)),1,N,N) ...
176               + sparse(idx,idx-((H+2)*(W+2)),1,N,N) );
177
178 L  = L(idx,idx); L_lambda = L_lambda(idx,idx);
179 clear mask idx
180 L        = L        - sparse(1:p,1:p,sum(L ,2)      ,p,p);
181 L_lambda = L_lambda - sparse(1:p,1:p,sum(L_lambda,2),p,p);
182 end % end DefineLapAndDiOperators
183
184 % ////////////////////////////////////////////////
185 function Z0 = IntegrateEachFrame(gx,gy)   % ///////
186 % ////////////////////////////////////////////////
187 % Case 0: each frame independently
188 [H,W,T] = size(gx);
189
190 Z0 = zeros(H,W,T);
191 tic,
192 for t = 1:T;
193     [Z0(:,:,t)] = LS_integration(gx(:,:,t),gy(:,:,t));
194     Z0(:,:,t)   = Z0(:,:,t) - mean(mean(Z0(:,:,t)));
195 end
196 fprintf('Case 0, took: \n')
197 mt_printtime(toc)
198 end % end IntegrateEachFrame
199
200 % ////////////////////////////////////////////////
201 function Z1 = ...                                 % //
202     IntegrateGiven_vxvy(gx,gy,lambda,vx0,vy0)  % //
203 % ////////////////////////////////////////////////
204 % // Case 1: vx and vy are known scalar //////////
205 % solve,
206 % Eq Z_xx + Z_yy + lambda Z_tt = ...
207 % gx_x + gy_y + lambda (vx*gx +vy*gy)_t
208 %  => A = L_lambda ;
209 %  => b = Dx*gx+Dy*gy+lambda*Dt*(vx*gx + vy*gy);
210 [H,W,T] = size(gx);
211 global Dx Dy Dt L L_lambda;
212
213 A = L_lambda;
214 b = Dx*gx(:)+ Dy*gy(:)- lambda*Dt*( vx0*gx(:)+vy0*gy(:) );
215 fprintf('Solving first case ... \n')
216 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
217 % Finally, solve the linear (sparse) problem
218 tic
219 x1 = A(:,2:end)\b;
```

```
x1 = [0; x1];
fprintf('Case 1, A\b took: \n')
mt_printtime(toc)
Z1 = reshape(x1,[H W T]);
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end % end IntegrateGiven_vxvy

% ////////////////////////////////////////////////////
function [Z,vx,vy] = ...                          % ///////
    IntegrateAndEtimate_vxvy(gx,gy,lambda)        % ///////
% ////////////////////////////////////////////////////
% // Case 2: vx and vy are unknown scalar /////////////
% Euler lagrange eqs. are:
% (ai) u_xx + u_yy + lambda u_tt ...
%      + lambda (vx*gx+vy*gy)_t = g_xx + g_yy;
% (b)  gx'*u_t + gx'*gx * vx + gx'*gy * vy = 0
% (c)  gy'*u_t + gy'*gx * vx + gy'*gy * vy = 0% solve Eqs:
%
%  => A = [L_lambda | lambda*Dt*gx | lambda*Dt*gy;
%          gxTDt    | gx'*gx       | gx'*gy      ;
%          gyTDt    | gy'*gx       | gy'*gy      ];
%  => b = [Dx*gx+Dy*gy;
%          0;
%          0]
[H,W,T]   = size(gx); p = H*W*T;
global Dx Dy Dt L L_lambda;
aux_gxTDt = zeros(1,p);
aux_gyTDt = zeros(1,p);
for i = 1:p
    aux_gxTDt = aux_gxTDt + gx(i)*Dt(i,:);
    aux_gyTDt = aux_gyTDt + gy(i)*Dt(i,:);
end

A  = [L_lambda, lambda*Dt*gx(:), lambda*Dt*gy(:);
      lambda*([aux_gxTDt, gx(:)'*gx(:), gx(:)'*gy(:)]);
      lambda*([aux_gyTDt, gy(:)'*gx(:), gy(:)'*gy(:)])
     ];

b  = [Dx*gx(:) + Dy*gy(:); 0; 0];

fprintf('Solving second case ... \n')
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Finally, solve the linear (sparse) problem
tic
x = A(:,2:end)\b;
x = [0;x];
fprintf('Case 2, A\b took: \n')
mt_printtime(toc)
Z = reshape(x(1:p),[H W T]);
vx = x(p+1);
vy = x(p+2);
```

```
271 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
272 end % end IntegrateAndEtimate_vxvy
273
274 % ////////////////////////////////////////////////////
275 function [Z,vx,vy] = ...                          % ///////
276     IntegrateAndEtimate_VXVY(gx,gy,lambda,mu)  % ///////
277 % ////////////////////////////////////////////////////
278 % // Case 3: vx and vy are unknown scalar //////////////
279 % Euler lagrange eqs. are:
280 % (ai) u_xx + u_yy + lambda u_tt ...
281 %      + lambda (vx.*gx+vy.*gy)_t = g_xx + g_yy;
282 % (bi)  lambda * gx ( u_t + gx.*vx + gy.*vy ) - mu lap(vx) = 0
283 % (ci)  lambda * gy ( u_t + gx.*vx + gy.*vy ) - mu lap(vy) = 0
284 [H,W,T] = size(gx); p = H*W*T;
285 global Dx Dy Dt L L_lambda;
286 % Define some additional operators,
287 Gx  = sparse(1:p,1:p,gx(:),p,p);
288 Gy  = sparse(1:p,1:p,gy(:),p,p);
289 Oo  = 0*speye(p,p); %null sparse matrix (pxp)
290
291 A = [L_lambda, lambda*Dt*Gx, lambda*Dt*Gy;
292     lambda*(gx(:)*ones(1,3*p)).*[Dt, Gx, Gy] - mu*[Oo, L, Oo];
293     lambda*(gy(:)*ones(1,3*p)).*[Dt, Gx, Gy] - mu*[Oo, Oo, L]
294     ];
295
296 b  = [Dx*gx(:) + Dy*gy(:); zeros(p,1); zeros(p,1)];
297
298 fprintf('Solving third case ... \n')
299 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
300 % Finally, solve the linear (sparse) problem
301 tic
302 x = A(:,2:end)\b;
303 x = [0; x];
304 fprintf('Case 3, A\b took: \n')
305 mt_printtime(toc)
306 Z  = reshape(x(1:p)      ,[H W T]);
307 vx = reshape(x(p+[1:p])  ,[H W T]);
308 vy = reshape(x(2*p+[1:p]),[H W T]);
309 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
310 end % end IntegrateAndEtimate_VXVY
```

## D.3. Nose and eyes detection

```
1 %% [NosePos,LeftEyePos,RightEyePos] = ...
2 %               NoseAndEyesLocation(Dx,Dy,parameters)
3 %
4 % Inputs:
```

```
5  %   - Dx (mxnx1 double) x-partial derivative
6  %   - Dy (mxnx1 double) y-partial derivative
7  %   - parameters: struct that may contain,
8  %       .verbose {1- display some text, 2- also some graphics}
9  %       .TextureImage (mxnxc) used just for visualization
10 %       .Mask (mxnx1 binary) select those face points (mask=1)
11 %             and background (mask = 0);
12 %       .r0_ee     : mean distance between eyes,
13 %       .r0_ne     : mean distance between nose-eyes
14 %       .sigma_ee  : variance of the distance between eyes
15 %       .sigma_ne  :     "    distance between eyes and nose,
16 %
17 % Outputs:
18 %   - NosePos: a row vector containing [x y] nose position
19 %   - LeftEyePos: "
20 %   - RightEyePos: "
21 %
22 % Ref:
23 % [1] M. Di Martino, A. Fernandez and J. Ferrari. "3D
24 %     curvature analysis with a novel one-shot technique".
25 %     IEEE-ICIP 2014.
26 % ------------------------------------------------------------------
27 % matias di martino, matiasdm@fing.edu.uy       v.0.1   (2014)
28 % ------------------------------------------------------------------
29 function [NosePos, LeftEyePos, RightEyePos] = ...
30                                     NoseAndEyesLocation(Dx,Dy,p);
31
32 % ////////////////////////////////////////////////
33 %% Load input parameters,
34 % ////////////////////////////////////////////////
35 [m, n] = size(Dx);
36 % list of input par.
37 list   = {'verbose','TextureImage','Mask'    ,'DiskRadius', ...
38           'r0_ee','r0_ne', 'sigma_ee', 'sigma_ne'};
39 % list of def. values.
40 defval = {0        ,ones(m,n,3)   ,ones(m,n) ,1           , ...
41           70     , 90    , 30        , 30};
42
43 [verbose, Texture, Mask, DiskRadius, r0_ee, r0_ne, ...
44  sigma_ee, sigma_ne] = GetParameters(p,list,defval);
45
46 clear m n
47 % ////////////////////////////////////////////
48 % Use a coarse scale, ////////////////////////
49 % ////////////////////////////////////////////
50 % Erode the mask so border point are eliminated
51 % [in these regions gradient information tend to
52 % contain several outliers.]
53 se   = strel('disk',15);
54 Mask = imerode(Mask,se);
55
```

```
56  % We are assuming that the order of magnitude of
57  % the a face is 600x600 px
58  Ratio = .15;
59  fx    = imresize(Dx.*Mask, Ratio, 'bilinear');
60  fy    = imresize(Dy.*Mask, Ratio, 'bilinear');
61  M     = imresize(Mask, Ratio);
62
63  KSize = [10 10]; KSigma = 5;
64  Ker   = fspecial('Gaussian',KSize,KSigma);
65  fx    = conv2(fx,Ker,'same');
66  fy    = conv2(fy,Ker,'same');
67
68  % ////////////////////////////////////////////////////
69  % /// Nose and eyes detection  ///////////////////////
70  % ////////////////////////////////////////////////////
71  [Nose,Eyes] = FindNoseAndEyesCandidates(fx,fy,verbose);
72
73  % ////////////////////////////////////////////////////
74  % /// Remove false positive points //////////////////
75  % ////////////////////////////////////////////////////
76  r0_ne    = Ratio*r0_ne;    r0_ee    = Ratio*r0_ee;
77  sigma_ne = Ratio*sigma_ne; sigma_ee = Ratio*sigma_ee;
78
79  [NosePos, LeftEyePos, RightEyePos] = ...
80      RemoveFalsePositive(Nose,Eyes,r0_ne,r0_ee,...
81                          sigma_ne,sigma_ee,verbose);
82
83  % Step back to the original scale;
84  NosePos      = round(NosePos/Ratio);
85  LeftEyePos   = round(LeftEyePos/Ratio);
86  RightEyePos  = round(RightEyePos/Ratio);
87
88  % ////////////////////////////////////////////////////
89  % /// Output results               ///////////////////
90  % ////////////////////////////////////////////////////
91  if verbose>1,
92   figure('name','Nose (B) and Eyes (Left R, Right G) detected')
93   imshow(uint8(Texture));  hold on
94   scatter(NosePos(1),     NosePos(2),     50,[0 0 1],'filled');
95   scatter(LeftEyePos(1),  LeftEyePos(2),  50,[1 0 0],'filled');
96   scatter(RightEyePos(1), RightEyePos(2), 50,[0 1 0],'filled');
97  end
98
99  end %end main function
100 %
101
102 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
103 % Aux functions, %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
104 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
105 % ////////////////////////////////////////////////////
106 function [varargout] = ...                % ///////
```

```
107     GetParameters(parameters,list,defval)  % ///////
108  % ////////////////////////////////////////////////
109  % see D3D code.
110  end  % end GetParameters
111
112  % ////////////////////////////////////////////////////////
113  function [Nose,Eyes] = FindNoseAndEyesCandidates(fx,...
114                           fy,verbose)
115  % ////////////////////////////////////////////////////////
116  % Define some shortcuts,
117  dx = @(U) 1/2 * [0*U(:,1)  U(:,3:end)-U(:,1:end-2)  0*U(:,1)];
118  dy = @(U) 1/2 * [0*U(1,:); U(3:end,:)-U(1:end-2,:); 0*U(1,:)];
119
120  % ////////////////////////////////////////////
121  % compute mean and Gaussian curvature /////////
122  % ////////////////////////////////////////////
123  H = ( (1+fy.^2) .* dx(fx) ...
124      - 2 .*fx.*fy.* dx(fy)  ...
125      + (1+fx.^2) .* dy(fy) ) ...
126      ./ ( 2 * (1 + fx.^2 + fy.^2 ).^(2/3) ) ;
127
128  K = ( dx(fx).*dy(fy) - dx(fy).^2 ) ...
129      ./ ( (1 + fx.^2 + fy.^2 ).^2 );
130  % ////////////////////////////////////////
131  % /// Check for noses on elliptical /////////
132  % /// concave regions (K>0 and H<0) /////////
133  % ////////////////////////////////////////
134  % define thresholds
135  H_th   = 1.5;
136  K_th_n = 15;  % the nose presents higher
137  K_th_e = 1.5;   % values than the eyes.
138
139  Nose = ( abs(H) > H_th*mean( abs(H(:)) )    ) & ...
140         ( abs(K) > K_th_n*mean( abs(K(:)) ) ) & ...
141         ( K      > 0                        ) & ...
142         ( H      < 0                        ) ;
143
144  %//////////////////////////////////////////////////////////
145  %/// Check for eyes on elliptical convex regions (K>0 and H>0)
146  %//////////////////////////////////////////////////////////
147  Eyes = ( abs(H) > H_th*mean( abs(H(:))   ) ) & ...
148         ( abs(K) > K_th_e*mean( abs(K(:)) ) ) & ...
149         ( K      > 0                        ) & ...
150         ( H      > 0                        ) ;
151
152  end  % end FindNoseAndEyesCandidates
153
154  % ////////////////////////////////////////////////////
155  function [NP,LEP,REP] = RemoveFalsePositive(Nose,Eyes,...
156                     r0_ne,r0_ee,sigma_ne,sigma_ee,verbose)
157  % ////////////////////////////////////////////////////
```

```
158
159 % compute the location of Nose and Eyes candidates
160 [X,Y] = meshgrid(1:size(Nose,2), 1:size(Nose,1));
161 CandidateNoseLoc = [X(Nose(:)==1) Y(Nose(:)==1)];
162 CandidateEyesLoc = [X(Eyes(:)==1) Y(Eyes(:)==1)];
163
164 % ------------------------------------------------------------
165 % I) Propagate the probabilities to impose
166 %    the geometrical constrains
167 % ------------------------------------------------------------
168
169 Gne = @(x,y)   1/(sigma_ne*sqrt(2*pi)) ...
170              * exp(- (sqrt(x.^2+y.^2)-r0_ne).^2 ...
171              / (2*sigma_ne^2) );
172 Gee = @(x,y)   1/(sigma_ee*sqrt(2*pi)) ...
173              * exp(- (sqrt(x.^2+y.^2)-r0_ee).^2 ...
174              / (2*sigma_ee^2) );
175
176 % ======================================
177 % A) First step of the estimation, =======
178 % ======================================
179 % initialization
180 ZoneNE  = zeros(size(Eyes));
181 % the NE represents the zone where is
182 ZoneEN  = zeros(size(Nose));
183 % probable to find a nose because of the
184 ZoneEE  = zeros(size(Eyes)); % potential eyes candidates
185 for k = 1:size(CandidateNoseLoc,1);
186     ZoneEN = ZoneEN + ...
187      Gne(CandidateNoseLoc(k,1)-X,CandidateNoseLoc(k,2)-Y);
188 end
189 ZoneEN0 = ZoneEN/max(ZoneEN(:)); % normalization
190 for k = 1:size(CandidateEyesLoc,1);
191     ZoneNE = ZoneNE + ...
192      Gne(CandidateEyesLoc(k,1)-X,CandidateEyesLoc(k,2)-Y);
193
194     ZoneEE = ZoneEE + ...
195      Gee(CandidateEyesLoc(k,1)-X,CandidateEyesLoc(k,2)-Y);
196 end
197 ZoneNE0 = ZoneNE/max(ZoneNE(:)); % normalization
198 ZoneEE0 = ZoneEE/max(ZoneEE(:)); % normalization
199 ZoneE0  = ZoneEE0.*ZoneEN0;      % combine EE and EN;
200 % ====================================
201 % B) Now compute weight again but =====
202 %    using cross information      =====
203 % ====================================
204 ZoneNE  = zeros(size(Eyes));
205 % the NE represents the zone where is
206 ZoneEN  = zeros(size(Nose));
207 % probable to find a nose because of the
208 ZoneEE  = zeros(size(Eyes)); % potential eyes candidates
```

```
for k = 1:size(CandidateNoseLoc,1);
 ZoneEN = ZoneEN + ...
 ZoneNE0(CandidateNoseLoc(k,2),CandidateNoseLoc(k,1)) ...
 * Gne(CandidateNoseLoc(k,1)-X,CandidateNoseLoc(k,2)-Y);
end
ZoneEN = ZoneEN/max(ZoneEN(:)); % normalization
for k = 1:size(CandidateEyesLoc,1);
  ZoneNE = ZoneNE + ...
  ZoneE0(CandidateEyesLoc(k,2),CandidateEyesLoc(k,1)) ...
  * Gne(CandidateEyesLoc(k,1)-X,CandidateEyesLoc(k,2)-Y);

  ZoneEE = ZoneEE + ...
  ZoneE0(CandidateEyesLoc(k,2),CandidateEyesLoc(k,1)) ...
 * Gee(CandidateEyesLoc(k,1)-X,CandidateEyesLoc(k,2)-Y);
end
ZoneNE = ZoneNE/max(ZoneNE(:)); % normalization
ZoneEE = ZoneEE/max(ZoneEE(:)); % normalization
ZoneE  = ZoneEE.*ZoneEN;        % combine EE and EN;


if verbose>1,
    figure('name','ZNE'), imagesc(ZoneNE); axis image,
    axis off, colormap hot, hold on
    scatter(CandidateNoseLoc(:,1),CandidateNoseLoc(:,2)...
    ,36,[0 0 1],'filled')
    figure('name','ZEE*ZEN'), imagesc(ZoneE);
    axis image, axis off,
    colormap hot, hold on
    scatter(CandidateEyesLoc(:,1),CandidateEyesLoc(:,2)...
    ,36,[0 0 1],'filled')
end

% ------------------------------------------------------
% II) Detect the centroid of candidate points with
%     higher "ZoneNE" and "ZoneEE" value,
% ------------------------------------------------------
[X,Y] = meshgrid([1:size(Nose,2)],[1:size(Nose,1)]);

% ////////////////////////////////////////////
% II.A) Nose location ////////////////////////
% ////////////////////////////////////////////
[maxvalue] = max(ZoneNE(Nose(:)==1));
if ~isempty(maxvalue) & ~isnan(maxvalue);
                        % maxvalue is empty if we don't have
                        % any point candidate,
    [y0,x0]    = find(ZoneNE==maxvalue & Nose==1,1,'first');
    % keep the cloud centered at the extracted points
    Nose = ( (Nose(:)==1 & abs(Y(:)-y0)<(r0_ne/2)) ...
           & (abs(X(:)-x0)<(r0_ne/2)) );
    % Compute the position of the centroid of the most
    % likely cloud of points.
```

```
260     NP    = [mean(X(Nose(:)==1)) mean(Y(Nose(:)==1))];
261 else
262     if verbose>0,
263         warning('No nose candidate was detected!!'),
264     end
265     NP    = [NaN NaN];
266 end
267
268 % ////////////////////////////////
269 % II.B) Eyes location //////////////
270 % ////////////////////////////////
271 AuxEyes = Eyes;
272 [maxvalue] = max(ZoneE(AuxEyes(:)==1));
273 if ~isempty(maxvalue) && ~isnan(maxvalue),
274                     % maxvalue is empty if we don't
275                     % have any eye candidate
276     [y0,x0]    = find(ZoneE==maxvalue & AuxEyes==1,1,'first');
277     % remove this cloud of points from the set of candidates
278     AuxEyes(  (Y(:)-y0).^2 + (X(:)-x0).^2 < (r0_ee/4)^2 ) = 0;
279     % now look for the second maximum
280     [maxvalue] = max(ZoneE(AuxEyes(:)==1));
281     if ~isempty(maxvalue),
282     % max value is empty if we have a single
283     % cloud of points as eyes candidates.
284     % if we reach this point, means that we have (at least)
285     % two different clouds of points which corresponds to
286     % eyes candidates, we now need to find the centroid of
287     % each cloud.
288         [y1,x1] = find(ZoneE==maxvalue & AuxEyes==1,1,'first');
289         % keep just the two cloud of points around [x0,y0] and
290         % [x1,y1]
291         Eyes = Eyes(:)==1 ...
292                & ( ( (  abs(Y(:)-y0)<(r0_ee/8) ...
293                       & abs(X(:)-x0)<(r0_ee/8) ) ...
294                   )|( (  abs(Y(:)-y1)<(r0_ee/8) ...
295                        & abs(X(:)-x1)<(r0_ee/8) ) ) );
296         % compute the middle point between the eyes,
297         CenterPos = [mean(X(Eyes(:)==1)) mean(Y(Eyes(:)==1))];
298         CandidateEyesLoc = [X(Eyes(:)==1) Y(Eyes(:)==1)];
299
300         if ~isnan(NP(1)); % use the vector from the point
301             % between the eyes and the nose to define right
302             % and left directions,
303             auxvect    = CenterPos - NP;
304             auxvect2   = [auxvect(2) -auxvect(1)];
305             % vector orthogonal to the
306             % center-nose direction
307             % compute the inner product and classify as
308             % a function of the sign
309          Side = sign( ( CandidateEyesLoc - ...
310                         ones(size(CandidateEyesLoc,1),1)*NP) ...
```

```
311                     * auxvect2');
312         else %if we don't have the location of the nose
313          %available, just compare with the middle point,
314          Side = sign( ( CandidateEyesLoc - ...
315             ones(size(CandidateEyesLoc,1),1) * CenterPos ) ...
316           * ([x0 y0]-CenterPos)' );
317         end
318         LEP   = mean(CandidateEyesLoc(Side>0,:),1);
319         REP   = mean(CandidateEyesLoc(Side<0,:),1);
320
321     else % if we don't detect a second cloud of eye candidates
322          % means that just one eye is detected, set the
323          % centroid as the eye position and set the other
324          % eye position to NaN.
325      if verbose>0, warning('Just one eye was detected.'), end
326          REP = [NaN NaN];
327          LEP = [mean(X(Eyes(:)==1)) mean(Y(Eyes(:)==1))];
328     end
329 else % any eye candidate,
330     if verbose>0, warning('No eye was detected.'), end
331     LEP = [NaN NaN]; REP = [NaN NaN];
332 end
333 % ////////////////////////////////////////////////////////////
334 end % end RemoveFalsePositive
```

# References


[1] Andrea F. Abate, Michele Nappi, Daniel Riccio, and Gabriele Sabatino. 2d and 3d face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885 – 1906, 2007. Image: Information and Control. Cited in page 75.

[2] Amit Agrawal. *Scene Analysis under Variable Illumintation using Gradient Domain Methods*. PhD thesis, University of Maryland, 2006. Cited in pages 48, 53 and 54.

[3] Amit Agrawal, Rama Chellappa, and Ramesh Raskar. An algebraic approach to surface reconstruction from gradient fields. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 174–181 Vol. 1, 2005. Cited in page 54.

[4] Amit Agrawal, Ramesh Raskar, and Rama Chellappa. What is the range of surface reconstructions from a gradient field? In *Computer Vision–ECCV 2006*, pages 578–591. Springer, 2006. Cited in pages 48 and 54.

[5] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041, Dec 2006. Cited in pages 78 and 90.

[6] Gilles Aubert and Pierre Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations (second edition)*, volume 147 of *Applied Mathematical Sciences*. Springer-Verlag, 2006. Cited in pages 52, 53, 58 and 105.

[7] Gastón A Ayubi, Jaime A Ayubi, J Matías Di Martino, and José A Ferrari. Pulse-width modulation in defocused three-dimensional fringe projection. *Optics letters*, 35(21):3682–3684, 2010. Cited in pages 9, 10, 19 and 102.

[8] Gastón A. Ayubi, J. Matías Di Martino, Julia R. Alonso, Ariel Fernández, César D. Perciante, and José A. Ferrari. Three-dimensional profiling with binary fringes using phase-shifting interferometry algorithms. *Appl. Opt.*, 50(2):147–154, Jan 2011. Cited in pages 9, 10, 19 and 102.

[9] Ronen Basri, David Jacobs, and Ira Kemelmacher. Photometric stereo with general, unknown lighting. *International Journal of Computer Vision*, 72(3):239–257, 2007. Cited in pages 9 and 47.

[10] Paul Besl and Neil McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992. Cited in page 76.

[11] Paul J. Besl and Ramesh C. Jain. Invariant surface characteristics for 3d object recognition in range images. *Computer Vision, Graphics, and Image Processing*, 33(1):33 – 80, 1986. Cited in pages 45 and 80.

[12] Pravin Bhat, Brian Curless, Michael Cohen, and CL Zitnick. Fourier analysis of the 2D screened Poisson equation for gradient domain problems. *Computer Vision ECCV 2008*, 2008. Cited in page 47.

[13] Kevin W. Bowyer, Kyong Chang, and Patrick Flynn. A survey of approaches and challenges in 3D and multi-modal 3D+2D face recognition. *Computer Vision and Image Understanding*, 101(1):1–15, January 2006. Cited in pages 75 and 77.

[14] Kim L Boyer and Avinash C Kak. Color-encoded structured light for rapid active ranging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):14–28, 1987. Cited in pages 9, 10 and 15.

[15] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Three-Dimensional Face Recognition. *International Journal of Computer Vision (IJCV), vol 64 (1)*, pages 5–30, 2005. Cited in pages 45, 75 and 81.

[16] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. Cited in pages 91 and 92.

[17] Kyong I Chang, Kevin W Bowyer, and Patrick J Flynn. Multiple nose region matching for 3D face recognition under varying facial expression. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1695–700, October 2006. Cited in pages 75, 82 and 94.

[18] Alessandro Colombo, Claudio Cusano, and Raimondo Schettini. 3D face detection using curvature analysis. *Pattern Recognition*, 39(3):444–455, March 2006. Cited in pages 80, 81 and 82.

[19] Salil Desai and Bopaya Bidanda. Reverse engineering: A review & evaluation of contact based systems. In Ali Kamrani and EmadAbouel Nasr, editors, *Rapid Prototyping*, volume 6 of *Manufacturing Systems Engineering Series*, pages 107–131. Springer US, 2006. Cited in pages 8 and 10.

[20] J. Matías Di Martino, Gasón Ayubi, Alicia Fernández, and José A. Ferrari. Differential 3d shape retrieval. *Optics and Laser in Engineering*, 58C:114–118, 2014. Cited in pages 47, 77 and 102.

[21] J. Matías Di Martino, Federico Decia, Juan Molinelli, and Alicia Fernández. A novel framework for nontechnical losses detection in electricity companies. In Pedro Latorre Carmona, J. Salvador Sánchez, and Ana L.N. Fred, editors, *Pattern Recognition - Applications and Methods*, volume 204 of *Advances in Intelligent Systems and Computing*, pages 109–120. Springer Berlin Heidelberg, 2013. Cited in page 103.

[22] J. Matías Di Martino, A. Fernández, P. Iturralde, and F. Lecumberry. Novel classifier scheme for imbalanced problems. *Pattern Recognition Letters*, 34(10):1146–1151, 2013. Cited in page 103.

[23] J. Matías Di Martino, Alicia Fernández, and José A. Ferrari. 3d curvature analysis with a novel one-shot technique. *IEEE International Conference on Image Processing (ICIP)*, 2014. Cited in pages 77, 79 and 102.

[24] J. Matías Di Martino, Alicia Fernández, and José A. Ferrari. Gradient domain methods with application to 4d scene reconstruction. *Optics and Lasers in Engineering*, 66(0):223 – 232, 2015. Cited in page 102.

[25] J Matías Di Martino, Alicia Fernández, and José A Ferrari. One-shot 3d gradient field scanning. *Optics and Lasers in Engineering*, 72:26–38, 2015. Cited in pages 47, 77 and 102.

[26] J. Matías Di Martino, G. Hernández, M. Fiori, and A. Fernández. A new framework for optimal classifier design. *Pattern Recognition*, 46(8):2249–2255, 2013. Cited in page 103.

[27] Manfredo do Carmo. *Differential Geometry of Curves and Surfaces*. Pearson Education Canada, 1976. Cited in page 80.

[28] Zhouyu Du, Antonio Robles-Kelly, and Fangfang Lu. Robust surface reconstruction from gradient field using the l1 norm. In *Digital Image Computing Techniques and Applications, 9th Biennial Conference of the Australian Pattern Recognition Society on*, pages 203–209. IEEE, 2007. Cited in pages 48 and 51.

[29] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. Wiley, New York, 2. edition, 2001. Cited in page 91.

[30] Opto engineering: The Telecentric Company. http://www.opto-engineering.com/. Accessed: 2014-12-26. Cited in pages 99 and 176.

[31] John Ens and Peter Lawrence. An investigation of methods for determining depth from focus. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(2):97–108, Feb 1993. Cited in page 9.

[32] Timothy C Faltemier, Kevin W Bowyer, Patrick J Flynn, and Senior Member. A Region Ensemble for 3-D Face Recognition. 3(1):62–73, 2008. Cited in pages 76, 82 and 94.

[33] Marcelo Fiori, Pablo Musé, Sergio Aguirre, and Guillermo Sapiro. Automatic colon polyp flagging via geometric and texture features. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC*, volume 10, pages 3170–3173, 2010. Cited in page 103.

[34] Marcelo Fiori, Pablo Musé, and Guillermo Sapiro. A complete system for polyps flagging in virtual colonoscopy. Technical report, DTIC Document, 2011. Cited in page 103.

[35] Jorge L. Flores, Beethoven Bravo-Medina, and José A. Ferrari. One-frame two-dimensional deflectometry for phase retrieval by addition of orthogonal fringe patterns. *Appl. Opt.*, 52(26):6537–6542, Sep 2013. Cited in pages 22 and 24.

[36] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002. Cited in page 47.

[37] Robert Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(4):439–451, Jul 1988. Cited in page 48.

[38] C.a. García-Isáis and Noé Alcalá Ochoa. One shot profilometry using a composite fringe pattern. *Optics and Lasers in Engineering*, 53:25–30, February 2014. Cited in page 13.

[39] Jason Geng. Rainbow three-dimensional camera: new concept of high-speed three-dimensional vision systems. *Optical Engineering*, 35(2):376–383, 1996. Cited in page 15.

[40] Jason Geng. Structured-light 3d surface imaging: a tutorial. *Adv. Opt. Photon.*, 3(2):128–160, Jun 2011. Cited in pages 13, 17, 21 and 169.

[41] Dennis Ghiglia and Marck Pritt. *Two Dimensional Phase Unwrapping: Theory, Algorithms and Software*. WileyBlackwell, 1998. Cited in pages 19 and 21.

[42] Gaile G. Gordon. Face recognition based on depth maps and surface curvature. *SPIE 1570, Geometric Methods in Computer Vision*, pages 234–247, September 1991. Cited in page 81.

[43] Paul M Griffin, Lakshmi S Narasimhan, and Soung R Yee. Generation of uniquely encoded light patterns for range data acquisition. *Pattern recognition*, 25(6):609–616, 1992. Cited in pages 16, 17 and 169.

[44] Shalini Gupta, Kenneth Castleman, Mia Markey, and Alan Bovik. Texas 3d face recognition database. In *Image Analysis Interpretation (SSIAI), 2010 IEEE Southwest Symposium on*, pages 97–100, May 2010. Cited in pages 33, 76, 85, 92, 93 and 176.

[45] Shalini Gupta, Mia Markey, and Alan Bovik. Anthropometric 3d face recognition. *International Journal of Computer Vision*, 90(3):331–349, 2010. Cited in pages 33, 85 and 92.

[46] Per Christian Hansen. *Rank-deficient and discrete ill-posed problems*. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Numerical aspects of linear inversion. Cited in page 50.

[47] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. Cited in page 10.

[48] Hideki Hayakawa. Photometric stereo under a light source with arbitrary motion. *JOSA A*, 11(11):3079–3089, 1994. Cited in page 47.

[49] Carrie L Heike, Kristen Upson, Erik Stuhaug, and Seth M Weinberg. 3d digital stereophotogrammetry: a practical guide to facial image acquisition. *Head & face medicine*, 6(1):18, 2010. Cited in page 15.

[50] Thang Hoang, Bing Pan, Dung Nguyen, and Zhaoyang Wang. Generic gamma correction for accuracy enhancement in fringe-projection profilometry. *Optics letters*, 35(12):1992–1994, 2010. Cited in page 19.

[51] Berthold Horn. Height and gradient from shading. *International Journal of Computer Vision*, 5(1):37–75, 1990. Cited in page 48.

[52] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. *National Taiwan University, Taipei 106, Taiwan*, 2010. Cited in page 91.

[53] Lei Huang, Patrick SK Chua, and A Asundi. Least-squares calibration method for fringe projection profilometry considering camera lens distortion. *Applied optics*, 49(9):1539–1548, 2010. Cited in pages 19 and 30.

[54] Peisen S Huang, Chengping Zhang, and Fu-Pen Chiang. High-speed 3-d shape measurement based on digital fringe projection. *Optical Engineering*, 42(1):163–168, 2003. Cited in page 19.

[55] Seiji Inokuchi, Kosuke Sato, and F. Matsuda. Range-imaging for 3-d object recognition. In *International Conference on Pattern Recognition*, pages 806–808. International association for pattern recognition., 1984. Cited in page 16.

[56] Idaku Ishii, Kenkichi Yamamoto, T Tsuji, et al. High-speed 3d image acquisition using coded structured light projection. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 925–930. IEEE, 2007. Cited in page 16.

[57] Chao Jiang, Ze-xun Geng, Xiao-feng Wei, and Chen Shen. Sift implementation based on gpu. *Proc. SPIE*, 8913:891304–891304–7, 2013. Cited in page 45.

[58] Ioannis a Kakadiaris, Georgios Passalis, George Toderici, Mohammed N Murtuza, Yunliang Lu, Nikos Karampatziakis, and Theoharis Theoharis. Three-dimensional face recognition in the presence of facial expressions: an annotated deformable model approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(4):640–9, April 2007. Cited in page 76.

[59] Peter Kovesi. Shapelets correlated with surface normals produce surfaces. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 994–1001. IEEE, 2005. Cited in page 48.

[60] Shuangyan Lei and Song Zhang. Flexible 3-d shape measurement using projector defocusing. *Optics letters*, 34(20):3080–3082, 2009. Cited in page 19.

[61] Yinjie Lei, Mohammed Bennamoun, and Amar a. El-Sallam. An efficient 3D face recognition approach based on the fusion of novel local low-level features. *Pattern Recognition*, 46(1):24–37, 2013. Cited in pages 45, 76, 77, 89, 91, 92, 94 and 100.

[62] Intuon Lertrusdachakul. *A Novel 3D recovery method by dynamic (de)focused projection*. PhD thesis, Ecole Doctorale Environnement - Sante, 2011. Cited in page 9.

[63] Xiaoli Li and Feipeng Da. Efficient 3D face recognition handling facial expression and hair occlusion. *Image and Vision Computing*, 30(9):668–679, September 2012. Cited in pages 76, 81 and 94.

[64] Lowery and Associates Land Surveying Services. http://lowerylandsurveys.com/surveying-services/3d-laser-scanning/. Accessed: 2015-01-20. Cited in pages 8 and 10.

[65] F Jessie MacWilliams and Neil JA Sloane. Pseudo-random sequences and arrays. *Proceedings of the IEEE*, 64(12):1715–1729, 1976. Cited in page 15.

[66] Mohammad H. Mahoor and Mohamed Abdel-Mottaleb. Face recognition based on 3d ridge images obtained from range data. *Pattern Recognition 42*, pages 445–451, 2009. Cited in page 76.

[67] Daniel Malacara. *Optical shop testing*, volume 59. John Wiley & Sons, 2007. Cited in page 18.

[68] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Cited in page 103.

[69] Mortimer Marks. System and devices for time delay 3d, 1992. US Patent 5,151,821. Cited in pages 9 and 10.

[70] Gerald F . Marshall and Glenn E . Stutz. *Handbook of Optical and Laser Scanning, Second Edition*. CRC Press, 2011. Cited in pages 9 and 10.

[71] Enric Meinhardt-Llopis, Javier Sánchez Pérez, and Daniel Kondermann. Horn-Schunck Optical Flow with a Multi-Scale Strategy. *Image Processing On Line*, 3:151–172, 2013. Cited in pages 59 and 60.

[72] Shree K Nayar and Yasuo Nakagawa. Shape from focus. *Pattern analysis and machine intelligence, IEEE Transactions on*, 16(8):824–831, 1994. Cited in page 9.

[73] Heung-Sun Ng, Tai-Pang Wu, and Chi-Keung Tang. Surface-from-gradients without discrete integrability enforcement: a gaussian kernel approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(11):2085–2099, 2010. Cited in page 48.

[74] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, Jul 2002. Cited in page 89.

[75] Nobuyuki Otsu. A threshold selection method from gray level histograms. *IEEE Transactions on Systems, Man and Cybernetics.*, 9(1):62–66, Jan 1979. Cited in page 79.

[76] Mesures par méthods optiques. www.gom.com. http://www.gom.com, Accessed: 2013-09-20. Cited in pages 8, 9 and 10.

[77] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, July 2003. Cited in page 47.

[78] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990. Cited in page 52.

[79] Senya Polikovsky, Yoshinari Kameda, and Yuichi Ohta. Facial micro-expressions recognition using high speed camera and 3d-gradient descriptor. In *Crime Detection and Prevention (ICDP 2009), 3rd International Conference on*, pages 1–6. IET, 2009. Cited in page 104.

[80] ViSOR: Video Surveillance Online Repository. http://www.openvisor.org/. Accessed: 2014-12-20. Cited in page 69.

[81] Giovanna Sansoni, Matteo Carocci, and Roberto Rodella. Three-dimensional vision based on a combination of gray-code and phase-shift light projection: analysis and compensation of the systematic errors. *Applied Optics*, 38(31):6565–6573, 1999. Cited in page 19.

[82] David 3D scanner. http://www.david-3d.com/en. Accessed: 2015-01-20. Cited in pages 8, 9 and 10.

[83] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *Computer Vision and Pattern Recognition (CVPR'03)*, 2003. Cited in page 16.

[84] Steven Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction

algorithms. *Computer Vision and Pattern Recognition*, pages 519 – 528, 2006. Cited in pages 9 and 10.

[85] Wikipedia: De Bruijn sequence. http://en.wikipedia.org/wiki/De_Bruijn_sequence. Accessed: 2014-12-01. Cited in pages 16 and 169.

[86] Mehmet Sezgin and Bulent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004. Cited in page 79.

[87] Tal Simchony, Rama Chellappa, and Min Shao. Direct analytical methods for solving poisson equations in computer vision problems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(5):435–446, May 1990. Cited in page 48.

[88] Dirk Smeets, Peter Claes, Dirk Vandermeulen, and John Gerald Clement. Objective 3D face recognition: Evolution, approaches and challenges. *Forensic science international*, 201(1-3):125–32, September 2010. Cited in page 45.

[89] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, USA, 1. edition, 2006. Cited in page 91.

[90] Richard Szeliski. Fast shape from shading. *CVGIP: Image Understanding*, 53(2):129–153, 1991. Cited in page 47.

[91] Mitsuo Takeda and Kazuhiro Mutoh. Fourier transform profilometry for the automatic measurement of 3-d object shapes. *Appl. Opt.*, 22(24):3977–3982, Dec 1983. Cited in page 30.

[92] Filareti Tsalakanidou and Sotiris Malassiotis. Real-time 2D+3D facial action and expression recognition. *Pattern Recognition*, 43(5):1763–1775, 2010. Cited in page 45.

[93] Robert J Valkenburg and Alan M McIvor. Accurate 3d measurement using a structured light system. *Image and Vision Computing*, 16(2):99–110, 1998. Cited in page 16.

[94] Hongcheng Wang, Ning Xu, Ramesh Raskar, and Narendra Ahuja. Videoshop: A new framework for spatio-temporal video editing in gradient domain. *Graph. Models*, 69(1):57–70, January 2007. Cited in page 56.

[95] Tiangong Wei and Reinhard Klette. A new algorithm for gradient field integration. Technical report, CITR, The University of Auckland, New Zealand, 2001. Cited in page 48.

[96] Joachim Weickert. *Anisotropic Diffusion in Image Processing*. PhD thesis, University of Kaiserslautern, Germany, 1996. Cited in pages 48, 52 and 53.

[97] Joachim Weickert. *Anisotropic Diffusion in Image Processing*. G.B. Teubner Stuttgart, New York, 2008. Cited in pages 52 and 53.

[98] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):191139–191139–, 1980. Cited in page 47.

[99] Ruo Zhang, P.-S. Tsai, J.E. Cryer, and M. Shah. Shape-from-shading: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):690–706, Aug 1999. Cited in page 47.

[100] Song Zhang. Recent progresses on real-time 3d shape measurement using digital fringe projection techniques. *Optics and Lasers in Engineering*, 48(2):149 – 158, 2010. Cited in pages 9 and 10.

[101] Song Zhang. *Handbook of 3D machine vision: optical metrology and imaging*. Series in Optics and Optoelectronics. CRC Press, Hoboken, NJ, 2013. Cited in page 9.

[102] Song Zhang and Peisen S Huang. Phase error compensation for a 3-d shape measurement system based on the phase-shifting method. *Optical Engineering*, 46(6):063601–063601, 2007. Cited in page 19.

[103] Yan-Ning Zhang, Zhe Guo, Yong Xia, Zeng-Gang Lin, and David Dagan Feng. 2D representation of facial surfaces for multi-pose 3D face recognition. *Pattern Recognition Letters*, 33(5):530–536, 2012. Cited in pages 45 and 76.

[104] Zhengyou Zhang. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In *ICCV99*, pages 666–673, 1999. Cited in page 30.

[105] Zonghua Zhang. Review of single-shot 3D shape measurement by phase calculation-based fringe projection techniques. *Optics and Lasers in Engineering*, 50(8):1097–1106, August 2012. Cited in page 13.

# Tables index

2.1. General features of the most popular three-dimensional scanning techniques. The objective of this table is to summarize some of the most common strength and weakness of each family of methods. Of course as it is a general description, some particular implementations in each class may not verify all the characteristics here summarized. . . . . 10

2.2. Comparison of integration techniques for the Ramp-Peaks experiment. The root mean squared error is display as a percentage of surface amplitude, and execution time of the algorithms are normalized with respect to the fastest one. . . . . 33

2.3. Comparison of integration techniques for one face of the Texas Database. The root mean squared error is displayed as a percentage of surface amplitude, and execution time corresponds to an experiment performed on MATLAB and a Intel Core i5 CPU M 430 @ 2.27GHz x 4, the size of the discrete input gradients ($p$ and $q$) was $750 \times 500$ pixels. . . . . 35

3.1. Mean execution time of each algorithm (relative to the fastest one). 56

3.2. Summary of the compared reconstructions. . . . . 62

3.3. Minimum MSE, optimal value of $\lambda$ and mean execution time using different regularization techniques. . . . . 66

4.1. $HK$ classification. . . . . 80

Tables index

4.2. Accuracy (percentage) over Train and Test Sets. The superscript (A or B) recalls the patch from which features were extracted. Recall that the patch A corresponds to the region of the eyes while patch B corresponds to the region of the nose. The first two columns shows the results obtained just considering the geometrical information (represented by histograms of depth partial derivatives values). The last three columns shows the accuracy obtained by considering all the features extracted from the patch A, patch B and the union of both, respectively. . . . . 94

# Figures index

2.1. Classification of 3D shape recovery methods according to their main characteristics. . . . . 8
2.2. Classification of the subset of triangulation based methods. . . . . 10
2.3. Principle of stereoscopic vision. The figure shows two cameras $C_{1,2}$ with lenses of focal distance $f$ and translated a distance $B$. . . . . 11
2.4. Example of a stereo pair of images. . . . . 12
2.5. Shift between the projection over the reference plane and the projection over the test surface. . . . . 13
2.6. Classification of most popular structured light frameworks according to the pattern/s they use. . . . . 14
2.7. Stripe projection example, figure (a) illustrates the projected pattern, (b) illustrates the scene height map (proportional to the gray level), and finally, figure (c) illustrates the image we would obtain when the pattern (a) is projected over the scene (b). . . . . 15
2.8. Examples of stripes indexing techniques. (a) Gray code (different columns are codded with different gray levels). (b) De Bruijn sequence (a specif sequence of colors is used to code each column). . 16
2.9. Example of De Brujin sequence for $k = 2$ and $n = 2$. Extracted from: [85] . . . . . 16
2.10. Example of 2D grid indexing pattern [43]. . . . . 17
2.11. Example of a sequential binary-coded pattern; each image represents one bit of the binary codification of the pattern. Figure extracted from [40]. . . . . 17

2.12. (a) Illustration of a sinusoidal pattern projected over a reference plane, and (b) the same pattern projected over a test surface. . . . 18

2.13. Example of a wrapped phase; the plot illustrate the original phase $\varphi$ (solid blue line) and the wrapped phase $\psi$ (dotted red line) . . . 20

2.14. Illustration of the acquisition setup. . . . . 22

2.15. (a) Acquired image $I(x, y)$. The colored figures show the 2D-spatial spectrum of the image. By filtering the horizontal and vertical Fourier components (red in the figures), one obtains the images $I_v(x, y)$ and $I_h(x, y)$ shown in (b) and (c), respectively. . . . . 23

2.16. Example of simulated data. (a) illustrates the height map of the test surface (where the gray level is proportional to surface's height), (b-d) show different patterns projected over the reference plane, (e-g) obtained (deformed patterns) when patterns are now projected over the test surface (a). . . . . 25

2.17. Illustration of the local bending of projected fringes due to surface's depth variation. . . . . 25

2.18. (a) Shows the simulated image with horizontal fringes (obtained from the red channel of the color image illustrated in Fig. 2.16(g)). (b) Shows one zoomed region of the image, where overlapped we display in red the vectors $(D_x \sin(\theta), 1 - D_y \sin(\theta))$ (as illustrated in Fig. 2.17) and in blue the gradient of the image $\nabla I$. (c) Shows the angular absolute error (measured in degrees) between the angle $\alpha$ obtained from image gradients and the value obtained from surface depth variations. . . . . 27

2.19. (a) Smooth version of 2.18(a) obtained applying a convolution between the binary fringes and a Gaussian kernel with a standard deviation $\sigma = 0{,}8px$. (b) Shows one zoomed region of the image, were overlapped we display in red the vectors $(D_x \sin(\theta), 1 - D_y \sin(\theta))$ (as illustrated in Fig. 2.17) and in blue the gradient of the image $\nabla I$. (c) Shows the angular absolute error (measured in degrees) between the value of the angle $\alpha$ obtained from image gradients and one obtained from surface depth variations). . . . . 28

2.20. As in Figs. 2.18 and 2.19 we illustrate the simulated image with horizontal fringes, image gradients compared with depth level sets orientation, and finally the angular errors. We illustrate in this particular image two particular areas where errors may be expected, in the first row we zoomed an area where a discontinuity in the depth map occurs. And in the second row we zoomed a region where we have a narrow peak. . . . . 29

2.21. Result of projecting horizontal fringes over the test surface whose depth map is illustrated in 2.16(a). In this example, we highlight in green and red those points where we expect to have poor gradient estimations due to: surface discontinuities and fringes' local extremum respectively. . . . . 31

2.22. (a) Original test surface. (b) Ground truth x-partial derivative. (c) Ground truth y-partial derivative. (d) Estimated x-partial derivative following D3D method. (e) Estimated y-partial derivative following D3D method. (f) Surface obtained by integrating the gradient field illustrated in (c-d) following Least Squares (LS) approach (3.2.1). (g) Surface obtained following Fankot-Chellappa (FC) integration approach (3.2.2). (h) Surface obtained following M-estimator (M) integration approach (3.2.4). (i) Surface obtained following LS integration approach including binary weight. (j) Surface obtained following Anisotropic Diffusion (AD) integration approach (3.2.5). (k) Surface obtained following $L_1$-norm minimization (L1) integration approach (3.2.3) . . . . . 32

2.23. The first row shows examples of real faces' depth of the Texas Database. Second row shows examples of the simulated data that we would obtain if we project the fringes pattern over each test surface. 34

2.24. Root mean squared error (relative to surface amplitude) for a set of *fast algorithms* (LS, WLS, FC) varying the width of the median filter used to preprocess the input gradient field. In the graphic is shown the median value and each individual RMSE error obtained for 30 faces of the Texas database. . . . . 35

2.25. Example of a face surface of the Texas Database reconstructed by different integration approaches. The retrieved gradient field was preprocessed width a $4 \times 4$ median filter. . . . . 36

2.26. Illustration of different input images for the same test face (of the Texas Database). The first image (upper left) illustrates the height map (proportional to the gray level of the image), and the other images illustrates the deformed patterns obtained for different methods. . . . . 37

2.27. Results for the Ramp-Peaks example . . . . . 37

2.28. Results over a representative set of faces of Texas Database. . . . . 38

2.29. Illustration of the manufacturing process of the test surface. . . . . 39

2.30. Left images, illustrates some of the acquired images when we projected summed (first row) and multiplied (second row) vertical and horizontal fringes. On the right we show the obtained RMSE over the gradients fields for different fringes widths. . . . . 40

2.31. (a) Ground truth gradient field, (b) estimated gradient gradient field, (c) error between true and estimated gradient field and (d) surface reconstructed from the estimated gradient field. . . . . 40

2.32. Left images, illustrates some of the acquired images when we projected summed (first row) and multiplied (second row) vertical and horizontal fringes. On the right we show the obtained RMSE over the gradients fields for different relations between the background light and pattern's light power (background light intensity was approximately 0,9$\mu W/mm^2$) . . . . . 41

2.33. Different frames of the input video. . . . . 42

2.34. Raw frames of estimated gradient field. . . . . 43

2.35. Frames of the retrieved surface. . . . . 43

3.1. Illustration of the input gradient field (left image corresponds to $p(x, y)$ and right image to $q(x, y)$) which contains white Gaussian noise and outliers. . . . . 54

3.2. Comparison of the Mean Squared Error for the set of presented integration techniques. Left: MSE obtained from the filtered gradient field, the abscissa correspond to the variance of the Gaussian Kernel used for averaging. Right: MSE obtained from the filtered gradient field using a non linear median filter, squared windows where used, the abscissa indicates the side length on pixels. . . . . 55

3.3. First row: reconstructions without preprocessing. Second row: reconstructions using the smoothed gradient field (where the width of the Gaussian Kernel was the one that minimizes the MSE for each integration technique). Third row: reconstructions using a median filter as a preprocessing step (the window size was the one that minimizes the MSE for each integration technique). . . . . 55

3.4. Illustration of two examples that show the motivation of the different kind of regularization. The firs example (a) illustrates a smooth height map varying along the time. The second example (b) illustrates a second scene which -also varies smoothly over time- and presents discontinuities over the height map. . . . . 59

3.5. First row shows some frames of the test surface. Second row shows the first component of the input (noisy) gradient field ($p(x, y, t)$), and the third row shows the second component of the input (noisy) gradient field ($q(x, y, t)$). . . . . 61

3.6. Mean Squared Error for the different reconstructions: $Z_{LS0}$, $Z_{AD0}$, $Z_{LSI}$, $Z_{ADI}$, $Z_{ZII1}$, $Z_{II2}$, $Z_{II3}$ as a function of the regularization weight $\lambda$. Case of constant and uniform velocity. . . . . 63

3.7. Frames of the surface retrieved by each strategy using the value of $\lambda$ that minimizes the MSE. . . . . 65

3.8. First row shows some frames of the (rotated) test surface. Second row shows the first component of the input (noisy) gradient field ($p(x, y, t)$), and the third row shows the second component of the input (noisy) gradient field ($q(x, y, t)$). . . . . 66

3.9. Mean Squared Error for the different reconstructions of the rotated surface: $Z_{LS0}$, $Z_{AD0}$, $Z_{LSI}$, $Z_{ADI}$, $Z_{ZII1}$, $Z_{II2}$, $Z_{II3}$ as a function of the regularization weight $\lambda$. Case of constant angular speed. . . . . 67

3.10. Frames of the rotated surface retrieved by each strategy using the value of $\lambda$ that minimizes the MSE. . . . . 68

3.11. Results for Video 1. . . . . 69

3.12. Results for Video 2. . . . . 70
3.13. Results for Video 3. . . . . 70
3.14. Results for Video 4. . . . . 71
3.15. Results for Video 5. . . . . 71
3.16. Examples of frames of the input video (left) and the estimated gradient field from them. . . . . 72
3.17. Some frames of the obtained 3D surfaces. . . . . 72

4.1. Illustration of the main steps of the proposed framework. Note: due to image resolution artifacts can appear, the full images, the code and a demo can be found at www.fing.edu.uy/~matiasdm. . . . . 79
4.2. Main steps to identify nose and eyes candidates. . . . . 81
4.3. Examples of nose (red dots) and eyes (green dots) candidates point detected. . . . . 81
4.4. Illustration of the steps followed to remove false positive detections. The size of each candidate point illustrates the likelihood of that point at each step, i.e. the larger the size, the higher the consistency with respect to the other candidate points. . . . . 84
4.5. This figure illustrates the nose and eyes detection procedure. (a) Input image. (b-c) $x$ and $y$ depth partial derivatives obtained by measuring fringes' deformation. (d-e) eyes and nose candidates obtained by curvature thresholding (as blue dots) overlapped with the estimated functions $\mathcal{E}(x, y)$ and $\mathcal{N}(x, y)$ respectively. (h) Nose and eyes located in the image. . . . . 85
4.6. Result obtained over the test set varying the power of the noise added to the input gradient field. . . . . 86
4.7. Definition of $\theta_1$, $\theta_2$ and $\theta_3$. . . . . 87
4.8. Results when varying face position. . . . . 88
4.9. Illustration of the regions used for *expression-invariant* features extraction. . . . . 89

4.10. Example of some features extracted for different subject of Texas-Database. The colored curves correspond to a portion of the feature vector obtained for each of the subjects shown. The x-axis indicates the number of the feature, and the y-axis the value of each of the features (for each subject) in the normalized features space. . . . . 90
4.11. Examples of the preprocessed color, and range images of the Texas 3D Face Recognition Database. Extracted From [44]. . . . . 93
4.12. Example of some of the acquired images by us. . . . . 95
4.13. Illustrative figure that displays the euclidean distance (in the feature space) of each of the samples with respect to all the other samples in the database. . . . . 96
4.14. Number of test instances versus its signed distance to SVM hyperplane. Red and green histograms were obtained by considering the negative and positive samples respectively. Under the histogram, examples of positive samples are shown. The dashed lines shows the particular distance obtained for each of the example images. . . . . 96
4.15. Left: Recall and Precision for different threshold values. Right: Precision-Recall curve obtained varying the threshold value used for classification. The colored dashed lines shows the lines of constant F-measure (which represents the geometrical mean between Recall and Precision). . . . . 97
4.16. Example of a simple projector device [30]. . . . . 99

Esta es la última página.
Compilado el 26 de mayo de 2015.
http://iie.fing.edu.uy/