UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention : Automatique, traitement du signal et des images

présentée et soutenue par

Lucero Diana LÓPEZ PÉREZ

Regularisation of images defined on non-flat surfaces

Thèse dirigée par **Rachid DERICHE**

et préparée à l'INRIA Sophia Antipolis, projet Odyssée

Date de soutenance : 15 Décembre 2006

Jury :	Pr. Luis Alvarez	Rapporteur
	Pr. Nikos Paragios	Rapporteur
	Pr. Nir Sochen	Rapporteur
	Dr. Rachid Deriche	Examinateur
	Pr. Michel Barlaud	Examinateur
	Pr. Maher Moakher	Examinateur

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention : Automatique, traitement du signal et des images

présentée et soutenue par

Lucero Diana LÓPEZ PÉREZ

Régularisation d'images sur des surfaces non planes

Thèse dirigée par **Rachid DERICHE**

et préparée à l'INRIA Sophia Antipolis, projet Odyssée

Date de soutenance : 15 Décembre 2006

Jury :	Pr. Luis Alvarez	Rapporteur
	Pr. Nikos Paragios	Rapporteur
	Pr. Nir Sochen	Rapporteur
	Dr. Rachid Deriche	Examinateur
	Pr. Michel Barlaud	Examinateur
	Pr. Maher Moakher	Examinateur

Abstract

We are interested in PDE-based approaches for regularization of scalar and vector-valued images defined over non-flat surfaces and their applications for image processing problems. We study the relationship between existing methods and compare them in terms of performance and ease of implementation. We develop new numerical methods to tackle divergence-like operators for regularization of images over triangulated surfaces. We generalize the Beltrami Flow regularization technique to images defined on implicit and explicit surfaces. Implementation schemes for these methods are proposed and validated by experimental results. We also show an application of our work to a concrete retinotopic mapping problem.

Résumé

Nous nous intéressons aux approches par EDP pour la régularisation d'images scalaires et multivaluées définies sur des supports non plans et à leurs applications à des problèmes de traitement des images. Nous étudions la relation entre les méthodes existantes et les comparons en termes de performance et complexité d'implémentation. Nous développons de nouvelles méthodes numériques pour traiter des opérateurs de type divergence utilisés dans les méthodes de régularisation par EDPs sur des surfaces triangulées. Nous généralisons la technique de régularisation du Flot de Beltrami pour le cas des images définies sur des surfaces implicites et explicites. Des implémentations sont proposées pour ces méthodes, et des expériences sont exposées. Nous montrons aussi une application concrète de ces méthodes à un problème de cartographie rétinotopique.

Resumen

Nos interesamos en los métodos de regularización de imágenes escalares y vectoriales sobre superficies no planas basados en EDPs y en sus aplicaciones en el área del tratamiento de imágenes. Estudiamos la relación entre los métodos existentes y los comparamos en términos de funcionamiento y facilidad de implementación. Desarrollamos nuevos métodos numéricos para estimar operadores de tipo divergencia para la regularización de imágenes definidas sobre superficies trianguladas. Generalizamos la técnica de regularización del flujo de Beltrami para imágenes definidas en superficies implícitas y explícitas. Esquemas numéricos para estos métodos son propuestos y validados por resultados experimentales. También demostramos una aplicación de nuestro trabajo a un problema concreto de segmentación de mapas retinotópicos del cortex humano.

Remerciements

Je voudrais remercier tout spécialement mon directeur de thèse Rachid Deriche pour sa direction et sa patience infinie, et Nir Sochen pour sa collaboration sur le Beltrami Flow. Merci à Nikos Paragios, Luis Alvarez et Nir Sochen pour avoir accepté d'être rapporteurs de ma thèse, et à Michel Barlaud et Maher Moakher pour leur participation au Jury. Cette thèse a été financé par le CONACYT (Conseil National de Science et Technologie du Mexique) et l'INRIA.

Merci à Gerardo Hermosillo pour son accueil et son aide au début de mon séjour, à Nicolas Wotawa et Jean-Philippe Pons avec qui j'ai travaillé sur les cartes rétinotopiques. Pour son aide et les échanges que nous avons eu merci à David Tschumperle, Christophe Lenglet, Rami Kanhouche, Robert Fournier, Geoffrey Adde, Facundo Mémoli et Frederic Abad. Merci pour les relectures et corrections de français à Max, Mathias, Raphaël et Mauricio. Merci à Olivier Faugeras pour son accueil (et tous ces signatures) ainsi qu'à Marie-Cecile pour son travail et son amitié, à Pierre pour m'avoir écouté et à Théo pour toutes les fois où je l'ai embêté avec des problèmes techniques. Et bien sûr aux autres membres et ex-membres de Robotvis-Odyssee que j'aurais du mal à tous citer ici tellement ils sont nombreaux!: Maureen, Thierry, Jan, Mikael, Christophe, Manu, Jacques, Diane, Bertrand, Adrien, Lionel, Ivan, Camilo, François, Sylvain, Sandrine, Damian, Marie-José, Mikaela, Monica, Pablo, Lilla, Olivier.

Pour être là dans les meilleurs et les pires moments, mais surtout les pires avec leur soutien precieux, merci à Ricardo et Heike, Raphaël, Fred, Mathias et Marie-Pierre, Miguel et Eunice. Mais aussi à Martine et Bernard, Gerardo et Ale, Gaby et Miguel, Martine Collard, Mauricio, Adrien, Sorel et Anaïs pour leur soutien direct et indirect.

Gracias tambien por su apoyo incondicional a Momia, Darius, Cochesito y Depilopil, a quienes dedico este trabajo.

Contents

List of Figures

Chapter 1

Introduction

Image regularization is a branch of the *digital image processing* field, which refers to processing digital images by means of a digital computer. Hence, the history of digital image processing (see [39]) is intimately tied to the development of the digital computer. Thus the field just appeared in the early 1960s, when the first computers became powerful enough to carry out meaningful image processing tasks for space applications, medical imaging, remote Earth resources observations, and astronomy. Since then, the field has grown vigorously and digital image processing techniques now are used in a broad range of applications, from archeology (to restore blurred pictures that were the only available records of rare artifacts lost or damaged after being photographed) to military (three-dimensional scene reconstruction from aerial and satellite images to locate strategic targets, see [64]). The continuing decline in the ratio of computer price to performance, the broad access of digital cameras and video cameras to the general public and the expansion of networking and communication bandwidth via the Internet have created new opportunities for the growth of digital image processing.

Digital image processes can be characterized from low-level to high-level processes: On low-level processes both inputs and outputs are images, they involve operations such as image preprocessing to reduce noise, contrast enhancement, and image sharpening. On mid-level process inputs generally are images, but outputs are attributes extracted from those images (as edges, contours, individual objects), this kind of processes involves tasks such as segmentation (partitioning an image into regions or objects) and classification (recognition). Finally, higher-level processing involves giving sense of a set of recognized objects, as in *image analysis*, and performing the cognitive

functions normally associated with natural vision. Image regularization is considered a low-level process.

We define by regularization of images a process that associates with a noisy image a resulting image similar to the original one but where the noise has been removed and the important features have been preserved. The interested reader might wonder what exactly do we mean by "similar", "important features" or "noise"? For the moment, we can only find answers to these questions for a single image or a particular class of images, but the more general answer is at the origin of a whole branch of the image processing [1] field. Let us precise that this work has no pretension of going one step further in answering these very difficult questions. Our aim is to extend known regularization methods for images defined on flat surfaces to the case of non-flat surfaces.

Wherever we look at, we see surfaces, colored surfaces, textured surfaces. The visual images we get from our world are no more that flattening views that glitter from the surfaces of the objects we perceive (with the exception of translucent objects). There is an evident interest in digital modeling of objects and scenes for the treatment of data over surfaces. But there are also other less obvious applications, like denoising of electric impulses or diffusion tensors on the human cortex surface, texture on human faces for face recognition, applications on fluid mechanics, pattern formations, to name but a few.

Most of the time, these data suffer from noise coming from numerous sources. For instance, errors originated by the imaging equipment (CT, MRI, ultrasound, 3D laser scanners, etc.), numerical sensitivity (simulations by finite element methods, recovery from photographs, recovery from a lossy data compression). In these cases, some kind of regularization is needed for the visualization or the processing of the image. Depending on the information we might have concerning the noise that perturbs the functions of interest, we can think of different regularization approaches in order to recover exploitable data. Most common methods are based on the optimization of a certain error-energy function. In a number of cases, this error-energy function appears to be in relationship with the error functions produced by stochastic methods or, more often, produced by the solutions of a particular type of Partial Differential Equation (PDE).

[1] General *image processing* also includes other processes such as optical filters, but for the rest of this work, we use the terms image processing and digital image processing indistinctly.

In this thesis, we focus on PDE regularization methods. Most of this work is centered on the reformulation of the regularization problem in terms of surfaces and the way the new surface-intrinsic operators can be estimated. In order to simplify our discussion, we deal with *simple orientable surfaces*, even though several methods can be generalized to more complicated surfaces.

For computer vision matters, surfaces are modeled mathematically as discretized meshes, level sets, algebraic surfaces, tessellated surfaces, or otherwise, depending on the particular application. Nevertheless, there are two widely used representations: the triangulated meshes and the level sets. In this work we use both of them. The PDE regularization methods differ greatly depending on the representation of the surface.

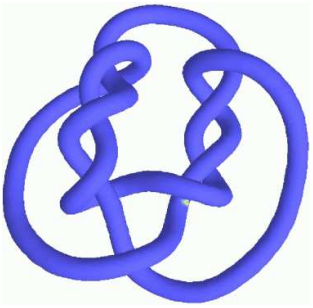
But before going any further, let us ask the following naive question: why spend so much time looking for relatively complicated non-planar operators, intrinsic to the surface, if it is simply possible to apply known 3D operators on the space surrounding the surface? Figure 1.1 illustrates an example of what can happen if we do so.

Let us explain how we obtained the results presented in figure 1.1: we took a knot as the interest surface, and the data set of a single point in this knot. The original image is shown in the first row and in the second row we show the image after application of a simple classical Laplacian operator in the euclidean 3D space (this is the classical space, where the data and the knot are embedded). In the third row, we show the results provided by the application of the Laplace-Beltrami operator, which is a surface intrinsic operator.

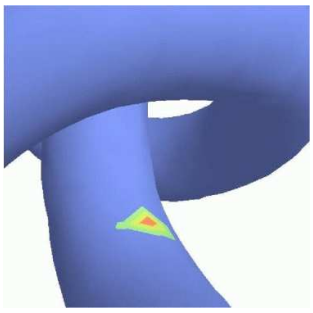
We can directly perceive the faulty results on the set of images corresponding to the simple classical 3-D Laplacian operator: the data on the point has been diffused *out of its vicinity* into a further region of the knot, that is in fact *close* to it if we consider the 3D euclidean distance. But the natural behavior we expect from a diffusion over a surface is the data to be diffused *over the surface*, not out of it. This is what we obtained with the use of the surface-intrinsic operator showed in the third row.

Situations like these are not exclusive to knots, they arise often when working on folded structures as the human cortex. This is why it is interesting to develop regularization methods that take the inner geometry of the surface into account.

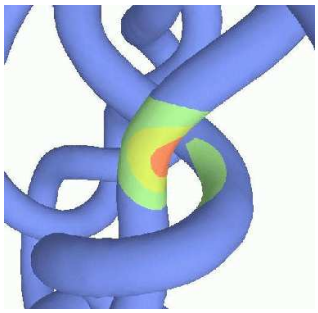
At the time the work presented in this thesis was started, regularization of

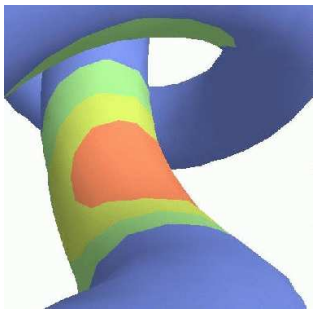


Original Image

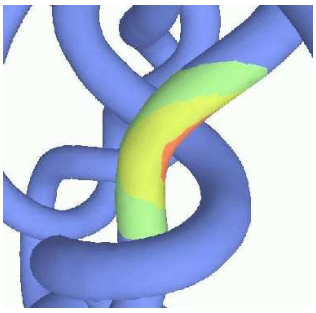


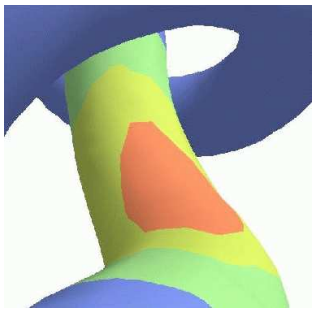
Detail: Point in the knot





$\Delta_{R^3} I$ - Diffusion on the space





$\Delta_S I$ - Diffusion on the surface

Figure 1.1: 3D euclidean Laplacian ($\Delta_{R^3} I$) vs surface-intrinsic Laplace-Beltrami operator ($\Delta_S I$)

images defined on triangulated surfaces was limited to isotropic regularization, and the first methods that make use of the level-set surface representation started to be known.

The main contributions of this thesis are:

- **A numerical method for anisotropic regularization over explicit surfaces.** We propose a method to locally estimate divergence-like operators that gives the same estimation that the finite element method and the discrete exterior calculus for the case of the Laplace-Beltrami. We use this method to propose an implementation for the Beltrami flow on triangulated surfaces.

- **The generalization of the Beltrami flow regularization to images defined on surfaces.** We reformulate the Beltrami Framework for the case of images defined on explicit and implicit surfaces, and propose and implementation. This regularization technique provides a way to overcome the over-smoothing of the isotropic regularization and the stair-casing effects anisotropic regularization by tunning between the two approaches via the parameter β to control the edge-preserving characteristic of the flow. The Beltrami flow is shown to act as the linear isotropic diffusion (L_2-norm) in the limit $\beta \to 0$ and to the strongly edge preserving anisotropic diffusion (L_1-norm) in the limit $\beta \to \infty$. We illustrate the utility of this approach showing synthetic and real examples on triangulated and implicit surfaces.

- **The clarification of the relation between implicit and extrinsic approaches and a comparison of their performances and implementation.** We clarify the link that exists between the the intrinsic Polyakov action of the Beltrami framework and the implicit harmonic energy functional. It is found that although the functionals are basically the same, there are differences in the way various problems are formulated and the way the functionals are applied. We also discussed the passage between the two representations and the advantages and drawbacks of the two associated methods, and compared them for the case of the retinotopic map extraction problem.

- **The application of these methods to a concrete retinotopic mapping problem.** We have detailed a method to obtain a human individual retinotopic map of the occipital cortex using fMRI, which makes use of surfaces-based regularization techniques rather than a

volume regularization method leading to better results. We propose alternative ways to improve the result using vector regularization over the gradient data instead of the scalar regularization over its projected components. We show how the Beltrami flow regularization can improve the resulting maps, tunning the β parameter to erase the noise like the isotropic smoothing while respecting the edges like the isotropic smoothing, without its associated stair-casing effect related to the different resolution of the data sets.

This thesis is organized as follows:

- In **Chapter 2** we present the two main surface representations and discuss the existing bibliography.
- In **Chapter 3** we present a numerical approach to deal with a certain type of PDEs regularization methods for images defined on triangulated surfaces. We use this approach to propose an anisotropic regularization technique. We generalize the Beltrami flow for images defined on triangulated surfaces, and present a numerical implementation based on the previously explained approach.
- In **Chapter 4** we present the generalization of the Beltrami flow for images defined on implicit surfaces.
- In **Chapter 5** we study the relationship between this methods and we discuss their comparative performances and ease of implementation.
- In **Chapter 6** we make an excursion into some regularization methods for vector and color images defined on explicit and implicit surfaces.
- In **Chapter 7** we present the results of our methods applied to images of electrical impulses on the human and macaque cortical surface.
- In **Chapter 8** we conclude and discuss future work and perspectives.

Introduction

La régularisation d'image est une branche du champ de *traitement d'image numérique*. Ce champ s'occupe de traiter des images numériques (aussi appellées digitales) à l'aide d'un ordinateur. Par conséquent, l'histoire du traitement d'image numérique (voir [39]) est intimement liée au développement de l'ordinateur. Ce champ est apparu au début des années 60, quand les premiers ordinateurs sont devenus assez puissants pour réaliser des taches significatives de traitement d'image pour des applications aux images spatiales, médicales, et des observations à distance de ressources terrestres. Depuis lors, le champ s'est développé vigoureusement et des techniques de traitement d'image numérique sont actuellement employées dans une large étendue des applications, de l'archéologie (pour reconstituer les images brouillées qui étaient les seuls registres disponibles des objets antiques perdus ou endommagés après avoir été photographiés) aux militaires (reconstruction tridimensionnelle de scènes à partir des images aériennes et satellites pour localiser les cibles stratégiques, voir [64]). La baisse continue dans le rapport prix/performance de l'ordinateur, l'accès grandissant au grand public des appareils photo numériques et des cameras, et l'expansion des réseaux et de la largeur de bande de communication par l'intermédiaire d'Internet ont créé de nouvelles occasions pour accroître l'intêret du traitement d'image numérique.

Les méthodes de traitement d'images digitales peuvent être caractérisés selon son *niveau* de la manière suivante : aux processus de bas niveau les entrées et les sorties sont des images, elles comportent des opérations telles que le prétraitement d'image pour réduire le bruit ou améliorer le contraste. Aux processus à mi-niveau les entrées sont généralement les images, mais les sorties sont des attributs extraits à partir de ces images (comme ses bords, contours, objets individuels), ce genre de processus implique des tâches comme la segmentation (découper une image dans des régions ou des objets) et la classification (identification). Finalement, le traitement de

plus haut niveau implique donner du sens à un ensemble d'objets identifiés (comme en *analyse d'image*) et en exécutant les fonctions cognitives normalement liées à la vision naturelle. Dans cette classification, la régularisation d'image est considérée comme un processus de bas niveau.

Nous définissons par régularisation d'image un processus qui associe à une image bruitée une image résultante semblable à l'originale mais où le bruit a été enlevé et les attributs importants ont été préservés. Le lecteur intéressé pourrait se demander ce qui nous voulons signifier par "attributs importants", "semblables" ou "bruit"? Pour le moment, nous pouvons seulement trouver des réponses à ces questions pour une image simple ou une classe particulière d'images, mais la recherche des réponses plus générales est à l'origine d'une branche entière du traitement d'image[2]. Précisons que ce travail n'a aucune prétention de répondre à ces questions très difficiles. Notre but est de généraliser des méthodes connues de régularisation d'images définies sur les surfaces planes au cas de surfaces non planes.

Partout où nous regardons, nous voyons des surfaces, colorées, texturées. Les images visuelles que nous obtenons de notre monde ne sont que des vues aplaties qui proviennent des surfaces des objets que nous percevons (à l'exception des objets translucides). Il y a un intérêt évident dans la modélisation numérique des objets et des scènes pour le traitement des données définies sur des surfaces. Mais il y a également d'autres applications moins évidentes, comme le débruitage d'impulsions électriques ou de tenseurs de diffusion sur la surface du cortex humain, la texture sur les visages humains pour son identification, des applications sur la mécanique des fluides, pour n'en nommer que quelques unes.

La plupart du temps, ces données contiennent du bruit venant de nombreuses sources. Par exemple, des erreurs produites par l'équipement d'acquisition de l'image (CT, MRI, ultrasons, scanners laser 3D) ou par la sensibilité numérique des méthodes d'acquisition (simulations par méthodes d'éléments finis, images reconstruites à partir de photographies, images reconstruites à partir d'une image ayant subi une compression avec perte d'information). Dans ces cas, une méthode appropriée de régularisation est nécessaire pour la visualisation ou le traitement de l'image. Selon l'information que nous avons sur la source du bruit qui perturbe les fonctions d'intérêt, nous pouvons procéder à des différentes approches de régularisation afin de récupérer des

[2]Le traitement d'image en général inclut également d'autres processus tels que les filtres optiques, mais pour le reste de ce travail, nous employons indifféremment les termes traitement d'image et traitement d'image numérique.

données exploitables. La plupart des méthodes sont basées sur l'optimisation d'une certaine fonction d'erreur, associée à une énergie. Dans certains cas, cette énergie semble être en rapport avec des fonctions d'erreur produites par des méthodes stochastiques ou, plus souvent, produites par la résolution d'un type particulier d'Équation Différentielle Partielle (EDP).

Dans cette thèse, nous nous concentrons sur les méthodes de régularisation par EDPs. Une partie ce travail est porté sur la reformulation du problème de régularisation pour le cas des supports surfaciques et la manière dont des opérateurs intrinsèques à la surface peuvent être estimés. Afin de simplifier notre discussion, nous traitons des *surfaces orientables simples*, quoique plusieurs méthodes peuvent être généralisées sur des surfaces plus compliquées. En ce qui concerne les problèmes de vision par ordinateur, les surfaces sont modelées mathématiquement en tant que mailles discrétisées, des ensembles de niveau, des surfaces algébriques, des surfaces tessellées, ou autrement, selon l'objet modelé. Néanmoins, le plus souvent une des deux représentations suivantes est utilisée : les maillages triangulés ou les ensembles de niveau.

Dans ce travail nous employons les deux. Les méthodes de régularisation par EDPs diffèrent considérablement selon la représentation de la surface. Mais avant d'aller plus loin, posons-nous la question : pourquoi passer tellement de temps à chercher des méthodes qui travaillent sur la surface, qui utilisent des opérateurs non planaires relativement compliqués, au lieu d'appliquer tout simplement les opérateurs 3D connus sur l'espace qui entoure la surface? La figure 1.2 illustre un exemple de ce qui peut se produire si nous procédons ainsi : nous avons pris un noeud comme example de surface, l'image sur la surface étant un simple point. L'image originale est montrée dans la première ligne, et dans la deuxième ligne nous montrons l'image une fois appliqué l'opérateur Laplacian dans l'espace 3D euclidien. Dans la troisième ligne, nous montrons les résultats fournis par l'application de l'opérateur de Laplace-Beltrami, qui est un opérateur intrinsèque à la surface. Nous l'employons ici au lieu du Laplacien classique pour comparer les différentes images obtenues après application de ces deux méthodes.

Nous pouvons percevoir directement les résultats défectueux sur l'ensemble des images correspondant au Laplacien classique : une region a été atteinte en dehors du voisinage du point d'origine dans une partie du noeud qui se trouve loin en termes de distance à parcourir sur la surface, mais qui est en fait *proche* si nous considérons la distance euclidienne sur l'espace 3D. En effet, le comportement que nous attendons d'une diffusion sur une surface

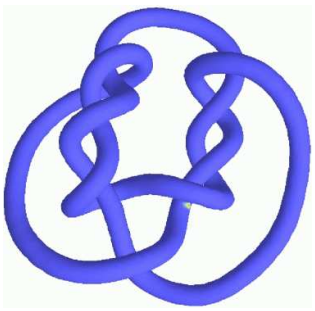
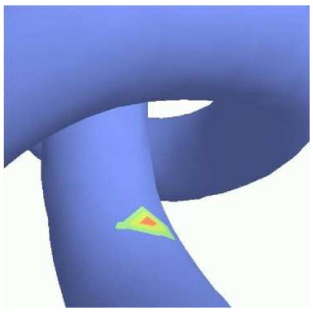
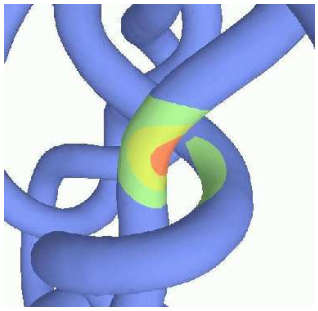
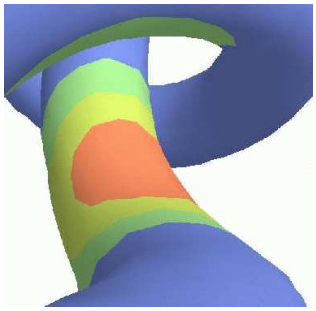


Image originale

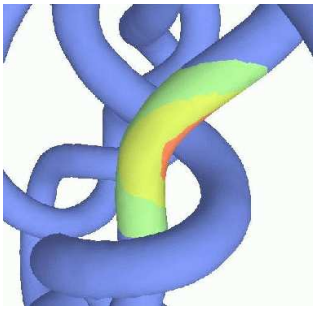


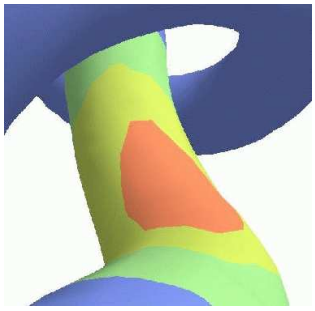
Detail : Point sur le noeud





$\Delta_{R^3} I$ - Diffusion dans l'espace Euclidien





$\Delta_S I$ - Diffusion sur la surface

Figure 1.2: Laplacien sur l'espace Euclidien ($\Delta_{R^3} I$) vs opérateur de Laplace-Beltrami ($\Delta_S I$), intrinsèque à la surface

est que les données soient répandues *sur la surface* et pas hors d'elle. C'est justement ce que l'on obtient avec l'opérateur intrinsèque à la surface dont le résultat est montré dans la troisième ligne.

Cette situation n'est pas spécifique aux noeuds, elle survient souvent lors que l'on travaille sur des surfaces plissées comme le cortex humain. C'est pourquoi il est intéressant de développer des méthodes de régularisation qui tiennent compte de la géométrie de la surface. Lorsque le travail présenté dans cette thèse a été commencé, la régularisation des images définies sur les surfaces triangulées était limité à la régularisation isotrope, et les premières méthodes qui se servent de la représentation par ensembles de niveau commencaient à être connues.

Les principales contributions de cette thèse sont :

- **Une méthode numérique de régularisation anisotrope sur des surfaces explicites.** Nous proposons une méthode pour estimer localement des opérateurs de type divergence qui coïncide avec l'estimation donnée par la méthode d'éléments finis et le calcul extérieur discret pour le cas de l'opérateur de Laplace-Beltrami. Nous employons cette méthode pour proposer une implémentation du flot de Beltrami sur les surfaces triangulées.

- **La généralisation de la régularisation par flot de Beltrami aux images définies sur des surfaces.** Nous reformulons le cadre du flot de Beltrami pour le cas des images définies sur les surfaces explicites et implicites, et proposons des implémentations. Cette technique de régularisation fournit une manière de surmonter le sur-lissage de la régularisation isotrope et l'effet d'escalier de la régularisation anisotrope en coordonnant les deux approches par l'intermédiaire du paramètre β pour contrôler la caractéristique de préservation du bord du flot. Il est montré que le flot de Beltrami se comporte comme la diffusion isotrope linéaire (norme L_2) dans la limite $\beta \to 0$ et comme la diffusion anisotrope (norme L_1) qui préserve fortement les bords dans la limite $\beta \to \infty$. Nous illustrons l'utilité de cette approche avec des exemples synthétiques et réels sur les surfaces triangulées et implicites.

- **La clarification de la relation entre les approches implicite, extrinsèque et explicite et une comparaison de leurs performances et implémentations.** Nous clarifions le lien qui existe entre l'action intrinsèque de Polyakov dans le cadre du Flot de Beltrami et

la fonctionnelle de l'énergie harmonique implicite. On constate que bien que les fonctionnelles soient fondamentalement identiques, il y a des différences dans la manière dont les problèmes sont formulés et la manière dont les fonctionnelles sont appliquées. Nous avons également discuté le transition entre les deux représentations et les avantages et les inconvénients des deux méthodes associées, et les avons comparées pour le cas du problème d'extraction de cartes rétinotopiques.

- **L'application de ces méthodes au problème des cartes rétinotopiques.** Nous avons détaillé une méthode pour obtenir une carte rétinotopique individuelle du cortex occipital humain en utilisant des images fMRI, qui se sert des techniques de régularisation sur des surfaces plutôt que des méthodes de régularisation sur le volume enveloppant, avec des meilleurs résultats. Nous proposons des manières alternatives d'améliorer le résultat en utilisant la régularisation du champ du gradient au lieu de la régularisation scalaire de ses composants projetés. Nous montrons comment la régularisation du flot de Beltrami peut améliorer les cartes résultantes, en utilisant le paramètre β pour effacer le bruit comme avec le lissage isotrope tout en respectant les contours comme avec le lissage anisotrope, mais sans l'effet d'escalier lié à la différence de résolution entre les données.

Cette thèse est organisée comme suit :

- Au **Chapitre 2** nous présentons les deux principales représentations des surfaces et la bibliographie existante.
- Au **Chapitre 3** nous présentons une approche numérique pour des opérateurs de type divergence utilisés par certains méthodes de régularisation par EDPs pour des images définies sur les surfaces triangulées. Nous employons cette approche pour proposer une technique de régularisation anisotrope. Nous généralisons le flot de Beltrami pour des images définies sur des surfaces triangulées, et présentons une implémentation numérique basée sur l'approche précédemment expliquée.
- Au **Chapitre 4** nous généralisons le flot de Beltrami pour des images définies sur des surfaces implicites.
- Au **Chapitre 5** nous étudions le rapport entre l'approche implicite et explicite et nous comparons leurs performances et facilité d'implémentation.

- Au **Chapitre 6** nous discutons quelques méthodes de régularisation pour des images vectorielles et de couleur définies sur des surfaces explicites et implicites.
- Au **Chapitre 7** nous présentons les résultats des méthodes exposées précédemment appliquées aux images d'impulsions électriques sur la surface corticale humaine et du macaque.
- C'est au **Chapitre 8** que nous concluons et discutons les perspectives de ce travail.

Chapter 2

State of the art

We start this chapter by presenting the two most widely used representations of a surface and its data. Then we review the literature available for the two frameworks. In the third section we present these two approaches with the example of the isotropic regularization via the computation of the Laplace-Beltrami operator.

2.1 Surface representations: Introducing the common frameworks

A two dimensional surface embedded on $\mathbb{R}^3$ can be represented in several manners. It can be defined as the solution of a simple equation.

For instance, the set of points

$$S_1 = \left\{ (x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 = 1 \right\}$$

defines the unit sphere and S_1 is an *implicit representation* of this surface.

The unit sphere can also be represented using a parametric function on the space. Take

$$\begin{aligned} f(\theta_1, \theta_2) &= (\cos\theta_1 \sin\theta_2, \sin\theta_1, \cos\theta_1 \sin\theta_2) \\ &= (x, y, z), \end{aligned}$$

then the unit sphere can also be defined by the set

$$S_2 = \{ f(\theta_1, \theta_2) \mid -\pi \leq \theta_1, \theta_2 < \pi \},$$

and in this case S_2 is an *intrinsic representation* of the unit sphere.

Of course S_1 and S_2 represent the same set of points. Thus, we have two different possible representations of the same surface: one implicit and the other intrinsic.

Implicit representations of 3-d surfaces are of the form

$$S_1 = \{ X \in \mathbb{R}^3 \mid \phi(X) = 0 \}.$$

In particular for closed surfaces, in the image processing field, the conventions require that the function ϕ, that defines the implicit representation, should be negative inside and positive outside the (possibly multiply connected) region bounded by the surface. Let us call this region Ω. For simplicity, we may also ask ϕ to be at least C^2. Usually, we use for ϕ the *signed distance* function defined

$$\phi(X) := \begin{cases} -Dist(X, S) & \text{if } X \in \Omega \\ +Dist(X, S) & \text{if } X \notin \Omega \\ 0 & \text{over the surface.} \end{cases}$$

On the other hand, intrinsic representations are given under the general form

$$\begin{aligned} S_2 &= \left\{ (x, y, z) = f(u, v) | (u, v) \in D \subset \mathbb{R}^2 \right\} \\ f(u, v) &= (f_x(u, v), f_y(u, v), f_z(u, v)) \end{aligned}$$

These are the two most common representations of mathematical surfaces.

However, the surfaces that can be found in computer vision problems are rarely known through these ideal representations: most of the time we have to face a discrete representation (a form given by a finite set of points) and the connections between these points. We also may have some additional information like the outing normal vectors at each points of the surface. This is the case for data obtained by laser scanners. Other ways of obtaining

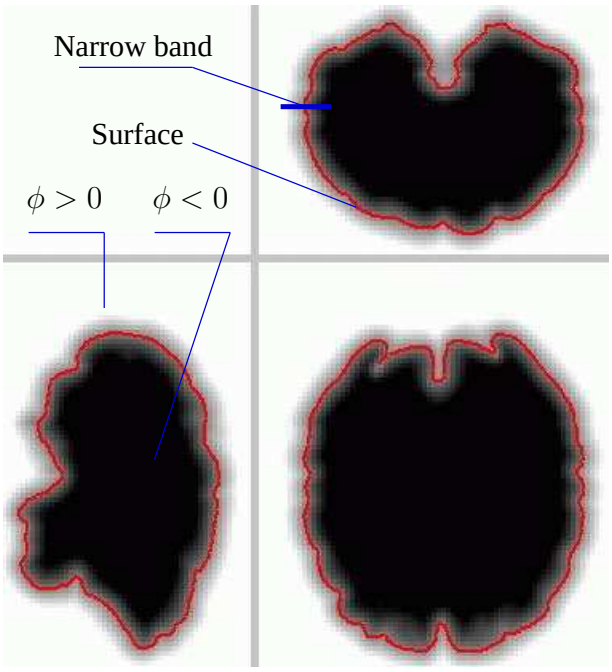


Figure 2.1: Visualization of the signed distance to a brain layer

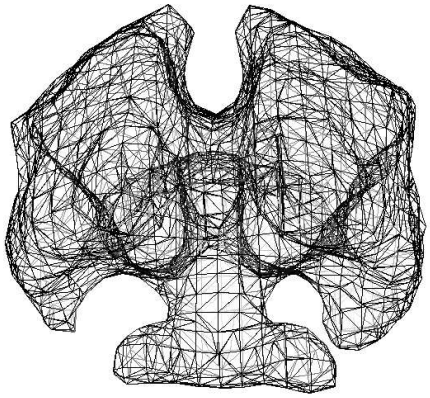


Figure 2.2: Triangulated cortex

digital surfaces -digital approximations of real surfaces- take as input a set of photographs of the surface, which are taken with a calibrated camera (like the one proposed in [18]) or in more general conditions (as done in [73]). In this case, we get as an output a volumetric scalar image whose values approximate the distance to the surface, providing directly an implicit discrete representation on the surface. Surfaces can also be approximated as a result of segmentation from volumetric images (it is the case of most of cortical images) and in this case the quality of the output may depend on the method used in either representations. This list of surface sources is far from being exhaustive and we gave examples of only a few of the intricate ways some data, representing a particular surface of interest, can be given in practice.

Most of the time, digitalized surfaces are given in one of these two forms:

- **Triangulated Surfaces, Surface Meshes or Explicit Surfaces**. In this case, the digitalized surface is given as a set of indexed triplets representing the vertex and a second set of vertex triplets that represent the triangles. Data on the surface is given as a set of values (scalars, vectors, tensors..) where each value corresponds to a vertex of the surface mesh. This is the most common representation for objects in 3D modeling and medical imaging. The traditional way to deal with these data is using the parametric functions given by the intrinsic representation[1]. Only recently, authors discovered (see [41]) that a discrete exterior calculus can be used for such matter (we will touch this matter in section 3.1.2 of this thesis).
- **Implicit Volumes or Level-Sets**. In this case, the digitalized surface is given as a scalar volumetric image, where each pixel value in a narrow band is the *signed distance* function to the surface or some other scalar function that is negative inside the surface and positive outside. Since we are only interested in the zero level-set, we only need the information in a band surrounding the surface. Data is stored in another volumetric image (scalar or multivalued, depending on the nature of the data) and also in pixels corresponding to the narrow band around the surface.

The Implicit Volumes became commonly used as a result of the increasing interest in Level-Set methods. Efforts are being made to find better ways of

[1] In this thesis we use the labels explicit and intrinsic indistinctly whenever there is no confusion between the mathematical and digital representation

storing the surface images. These do not use an entire volume to store the information because this information is in fact only needed in a band around the surface: this is the case of the RLE (Run-Length-Encoded) sparse level-set structure ([43]), unstructured graphs, adaptive grids ([19]), etc. Another inconvenient is that fine surface structures are not always captured by level-sets, although it is possible to use adaptive and triangulated grids (see for example [7] and [74]).

2.2 Bibliographical discussion

In this section we present the main aspects of the classical literature concerning regularization on manifolds. We also present a list of useful works that are strongly related to this subject and that we used for this work.

2.2.1 Explicit Surfaces

We start with a review of the works on *surface regularization*. It is indeed often the surface regularization problem that is placed first and its solution is closely related to the regularization of functions over manifolds when using PDE's methods. Both methods need the estimation of surface-intrinsic operators as the Laplace-Beltrami or Divergence operators: the first estimation is needed over the geometry of the manifold, whereas the second is needed over the set of functions defined on the manifold.

The surface regularization problem appears principally in two areas: error optimization and PDEs. For the first approach we can briefly cite the works from: [38, 42, 44, 60, 78, 101, 51, 49, 102, 56].

As the title of this thesis suggests, we are more concerned by the second category. For a very clear explanation on PDE's based regularization methods for flat multivalued images, we recommend [96], and some references that review the work on this area are [25, 3].

Following the path of the flat image regularization, the different works start from the use of a Laplacian operator over the surface. This PDE-based evolution technique was originally imported for image processing on flat images by [68], [74], and [100].

In [31], [21], and [30] this idea is extended to smooth noised surfaces, propos-

ing to apply the surface-intrinsic Laplacian, namely the Laplace-Beltrami operator over the *geometry* of the manifold, rather than a function over the manifold which is our case of study (we introduce the Laplace-Beltrami operator in the next section, but for a more proper definition we refer to [34]). Numerical issues on the discretization of the Laplace-Beltrami operator are discussed in [95]. In [50] and [51] the authors propose discrete approximations of the Laplace-Beltrami extended to arbitrary connectivity in a way to deal with multi-resolution problems, far from the simple case of triangulations that is studied in this thesis. In [29] an anisotropic technique is presented for denoising height fields and bivariate data.

In [31], the authors used an implicit discretization of geometric diffusion to obtain a strongly stable numerical regularization scheme. Afterward, in [30], they propose finer computational methods of normals and curvatures for discrete data with the objective to enhance and smooth triangulated surfaces. In [21], anisotropic geometric diffusion is introduced to enhance features while the smoothing effect occurs.

Note that we have not said anything about the anisotropic diffusion yet. It is sufficient for the moment to say that this regularization technique permits to *smooth* small irregularities of the function (in this case the geometric representation) while it *preserves* and even accentuates the strong irregularities that define its principal features (corners, edges, etc). This is a major contribution in surface regularization techniques since earlier methods resulted on images that shrunk as the evolution went on and some important edges were not preserved.

The problems of regularizing the geometry of a surface and the data over a surface are closely related, and the way the corresponding discretization problems are solved is very similar. In [6], the authors propose an original method for regularizing the surface and the data on the surface simultaneously, taking advantage of the fact that they share the same stiffness matrix. This is useful when the noise in the surface and the data are related, a situation that is often seen when the surface and functional data is extracted from the same original set, on the same process, or with the same equipment. It is also useful when the data on the surface is related with the geometry on the surface: for example, the color on the lips' surface limits itself to the lips' corners. In [22] they use the data information to regularize the surface *anisotropically* in order to follow discontinuities on the data.

Few works have dealt with the regularization of functions over surfaces,

but we can cite [20], where the authors propose the two estimations of the Laplace-Beltrami operator presented in the next section. More recently, in [82] and [54] (a work that can be found in the second and last chapters of this thesis), we presented the extension of the *Beltrami Flow* technique to the case of triangulated surfaces.

It is also worth mentioning the recent work [106] where the convergence of discrete Laplace-Beltrami operators over surfaces is studied, and where it is concluded that the best discretization known for the Laplace-Beltrami operator is the one obtained in [30], which is precisely the cotangents weights formula (2.7) that we retrieved in the previous section. This formula can also be found in [41] with a new approach: we focus ourselves on this new approach in the last chapter of this work. Finally, in [92, 90] the author shows segmentation of images defined on surfaces by an extension of the geodesic active contours.

2.2.2 Level-Set methods

The level-set methods were first introduced in 1979 by Dervieux and Thomasset for applications on fluid mechanics ([26, 27]) , and later in 1987 by Sethian and Osher ([65]) for computer vision applications. Level-set methods provide a very smart way to deal with moving *interfaces* such as curves and surfaces, as the model allows changes in the topology without major problems. Because of the importance of this discovery, these methods began to be used in many other applications from the image processing field such as segmentation ([45, 23, 62, 70]), motion estimation and tracking ([73, 61, 63]), front propagation ([65, 24, 36, 40, 71]), and many others. The level set method also offers an easier alternative to solve problems on fixed surfaces which before needed the entire and relatively complicated discretization of all the triangles.

Just like what we did for the explicit methods, we start citing the works on *surface smoothing and reconstruction*: [18, 66, 11, 75] (anisotropic geometric diffusion), [107] and [108]. More recently, in [94], an anisotropic diffusion of surface normals is investigated for features preserving surface reconstruction. An evolution PDE (such as the one presented in the previous section) defined this time on the whole volume governs the behavior of the level surface.

Concerning *non-flat image regularization*, the framework showed in [17], [8], [10], and [9] was one of the starting points of this thesis: the authors

present a level-set method to perform isotropic (see [17] for more details) regularization on scalar images that was extended afterward to anisotropic regularization for scalar and directional images in [8].

In [33], the authors show the use of anisotropic diffusion for vector field visualization on surfaces.

In [83], [81] and [82] we have generalized the Beltrami Flow regularization method for flat images for the case on non-flat implicit surfaces, and we presented numerical methods and applications for the case of scalar images. Another interesting contribution of these papers lies in the clarification of the relationship between the implicit and explicit approaches.

2.3 The Laplace Beltrami operator (Δ_S)

A classical method to erase the noise of flat scalar images in order to obtain a nice smoother function is the one based on the following PDE:

$$\frac{\partial u}{\partial t} = \Delta u, \qquad u_{t=0} = u_0. \tag{2.1}$$

Here, the scalar data $u_0 : \mathbb{R}^2 \to \mathbb{R}$ stands for the original noisy image defined on the plane and the solution $u : \mathbb{R}^2 \to \mathbb{R}$ of the PDE stands for the regularized image: this is the one we are looking for.

This PDE is borrowed from physics, it is known as the *heat equation*. In image processing, this regularization method is called *isotropic diffusion*, because the process smooths the image with the same intensity in all spatial directions without considering the fact that it blurs features of the image as well as the noise. The *anisotropic regularization*, which we introduce in section 3.2, on the contrary, privilege the smoothing effect in certain particular directions that are determined by some other PDE regularization model.

The isotropic method is equivalent to the convolution of the image function with a normalized gaussian kernel of variance $\sigma = \sqrt{2t}$. It is also equivalent to the minimization of the *energy* (variational approach) defined as:

$$\frac{1}{2}\int_{\mathbb{R}^2} \|\nabla u\|^2 \, dxdy,$$

This is due to the fact that Equation (2.1) actually defines the differential gradient descent used to minimize the energy.

The regularization algorithm consists in transforming the original image using PDE (2.1). The resulting process progressively smooths the image. Small irregularities first disappear, the image then starts to appear blurred and finally it converges to a constant image as $t \to \infty$ (see fig. 2.3). In practice, a human observer is needed in order to decide when to stop the evolution.

We have chosen this classical regularization method to show how both different frameworks could be applied. Of course, we first have to define this PDE on a manifold, which is the idealistic mathematical object for the representation of surfaces embedded in $\mathbb{R}^3$.

2.3.1 The Laplace-Beltrami on explicit surfaces

A discretized surface is formed of a set of points in $\mathbb{R}^3$ and facets between them. We will consider the case where these facets are triangles: the case of triangulated surfaces. The function on the surface is defined as a value attached to each vertex. These data can be the result of the application of a scanner from an object, numerical simulations retrieved by 2D images or various other extraction methods. It is the most simple and commonly used representation for objects in 3D modeling and medical imaging. Its portability and precision to represent detailed 3D objects are some of its major qualities compared to the implicit methods.

We represent mathematically the surface given by this kind of data using a *parametrization*. A parametrization of a certain surface S is a function

$$X : D \to S, \quad X \in C^2(D),$$

$$X(y) = \left\{x_1(y), x_2(y), x_3(y) : y = (y_1, y_2) \in D \subset R^2\right\}$$

such that $(X_1(p), X_2(p)) = (D_{y_1}X(p), D_{y_2}X(p))$ is a base for the tangent plane T_pS at the point $p \in S$.

Let us define the Riemannian metric g and the tensor l as:

$$g := \begin{pmatrix} \langle X_1, X_1\rangle & \langle X_1, X_2\rangle \\ \langle X_2, X_1\rangle & \langle X_2, X_2\rangle \end{pmatrix} \quad \text{and} \quad l := \begin{pmatrix} \langle X_{1,1}, \mathbf{n}\rangle & \langle X_{1,2}, \mathbf{n}\rangle \\ \langle X_{2,1}, \mathbf{n}\rangle & \langle X_{2,2}, \mathbf{n}\rangle \end{pmatrix}.$$

Original noisy image

Result of (2.1) after 3 iterations

Result of (2.1) after 10 iterations

Result of (2.1) after 50 iterations

Figure 2.3: Heat regularization on scalar flat image

For example, if the surface is of the form

$$x_3 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + 2\beta_4 x_1 x_2 + \beta_5 x_2^2 + \dots$$

the Riemannian metric tensor g and the tensor l become

$$\begin{aligned} g &= \begin{pmatrix} 1+\beta_1^2 & \beta_1\beta_2 \\ \beta_1\beta_2 & 1+\beta_2^2 \end{pmatrix}, \\ l &= \begin{pmatrix} \beta_3 & \beta_4 \\ \beta_4 & \beta_5 \end{pmatrix}. \end{aligned}$$

In this context, the Laplace-Beltrami operator Δ_X can be expressed as ([20]):

$$\begin{aligned} u &: S \to R, \\ \nabla_X\, u &= D_y X^T g^{-1} D_y\, u = \sum_{i,j=1}^{2} \left(g^{-1}\right)_{i,j} \frac{\partial u}{\partial y_j} X_i, \\ \Delta_X\, u &= \nabla_X \cdot \nabla_X\, u = \frac{1}{|g|^{\frac{1}{2}}} \sum_{i,j=1}^{2} \frac{\partial}{\partial y_i} \left(|g|^{\frac{1}{2}} \left(g^{-1}\right)_{i,j} \frac{\partial u}{\partial y_j} \right). \end{aligned}$$

Using the Laplace-Beltrami operator, we can generalize PDE (2.1) from flat images to parametrization in the following manner

$$\frac{\partial u}{\partial t} = \Delta_X\, u,\ u_{t=0} = u_0, \tag{2.2}$$

where again $u_0 : S \to \mathbb{R}$ stands for the original noisy image and $u : S \to \mathbb{R}$ stands for the regularized image. To be able to perform finite space-time differences methods and compute the evolution of the PDE we need to estimate or discretize the Laplace-Beltrami operator. We now briefly explain two known solutions (quoted from [20]).

Parametric method

This is a rather tricky method that makes use of a special intrinsic property of the Laplace-Beltrami operator. Let us explain the idea.

If (y_1, y_2) is a conformal coordinate system, the Laplace-Beltrami operator

becomes locally the planar Laplacian at y,

$$\Delta_X = \frac{1}{\lambda}\left(\frac{\partial^2}{\partial\left(y_1\right)^2}+\frac{\partial^2}{\partial\left(y_1\right)^2}\right)$$

For an arbitrary surface S and a fixed point $p \in S$, we can always find a conformal coordinate system by applying the affine transformation:

$$\begin{aligned} y &:= Q\left(x-p\right) \\ Q &:= \begin{pmatrix} \mathbf{n}_3 & 0 & -\sqrt{\mathbf{n}_1^2+\mathbf{n}_2^2} \\ 0 & 1 & 0 \\ \sqrt{\mathbf{n}_1^2+\mathbf{n}_2^2} & 0 & \mathbf{n}_3 \end{pmatrix} \begin{pmatrix} \frac{\mathbf{n}_1}{\sqrt{\mathbf{n}_1^2+\mathbf{n}_2^2}} & \frac{\mathbf{n}_2}{\sqrt{\mathbf{n}_1^2+\mathbf{n}_2^2}} & 0 \\ -\frac{\mathbf{n}_2}{\sqrt{\mathbf{n}_1^2+\mathbf{n}_2^2}} & \frac{\mathbf{n}_1}{\sqrt{\mathbf{n}_1^2+\mathbf{n}_2^2}} & 0 \\ 0 & 0 & \mathbf{1} \end{pmatrix} \\ y_3 &:= z\left(y_1,y_2\right) \cong \hat{\beta}_0+\hat{\beta}_1 x_1+\hat{\beta}_2 x_2+\hat{\beta}_3 x_1^2+2\hat{\beta}_4 x_1 x_2+\hat{\beta}_5 x_2^2 \end{aligned}$$

where the $\hat{\beta}_i$ are found by a standard least square estimation procedure using the neighboring points. We explain briefly this standard method below:

let $p_1,...,p_m$ be the m neighbors of the point p, and let $y^j = Q\left(p_i-p\right)$. Set

$$\begin{aligned} \mathbf{X} &:= \begin{pmatrix} y_1^1 & y_2^1 & \left(y_1^1\right)^2 & y_1^1 y_2^1 & \left(y_2^1\right)^2 \\ y_1^2 & y_2^2 & \left(y_1^2\right)^2 & y_1^2 y_2^2 & \left(y_2^2\right)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_1^m & y_2^m & \left(y_1^m\right)^2 & y_1^m y_2^m & \left(y_2^m\right)^2 \end{pmatrix} \\ \mathbf{Y} &:= \begin{pmatrix} y_3^1 \\ \vdots \\ y_3^m \end{pmatrix}. \end{aligned}$$

$\hat{\beta}$ is then obtained by solving the linear system

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\left(\mathbf{X}^T\mathbf{Y}\right).$$

Now, if we have the parametrization $W(y_1,y_2) := p + Q'(y_1,y_2,z(y_1,y_2))$, then, applying the linear transformation $V(v_1,v_2) = W(Av)$ where $A = g^{1/2}$, we obtain a conformal coordinate system such that the Laplace-Beltrami operator becomes $\nabla_X = \gamma_1+\gamma_3$ where γ_1, γ_3 are defined as the solutions of

$$z(v_1,v_2) = \gamma_1(v_1)^2 + \gamma_2 v_1 v_1 + \gamma_3(v_2)^2 + ...$$

This allows us to retrieve proper data and start the computations.

Finite element method

A linear barycentric interpolation is used to find an approached solution.

Let $\tilde{u} : (t,x) \rightarrow \tilde{u}(t,x)$ be the approached solution to the diffusion equation.

Take the function $\tilde{u} : (t,x)$ over each triangle:

$$\tilde{u} \rightarrow \tilde{u}(t,x) = \sum_{i=1}^{N_T} \tilde{u}|_{T_i}(t,x) \stackrel{\triangle}{=} \sum_{i=1}^{N_T} \tilde{u}_i(t,x)$$

On each triangle T_i ($i \in \{1 \dots N_T\}$) composed by the nodes $\{p_{i_1}, p_{i_2}, p_{i_3}\}$ we have the barycentric coordinates functions

$$\{\zeta_{i_1}, \zeta_{i_2}, \zeta_{i_3}\}.$$

We look for $\tilde{u}_i(t,.)$ of the form

$$\tilde{u}_i(t,x) = \zeta_{i_1}(x)\tilde{u}(t,p_{i_1}) + \zeta_{i_2}(x)\tilde{u}(t,p_{i_2}) + \zeta_{i_3}(x)\tilde{u}(t,p_{i_3})$$

that is, we express $\tilde{u}_i(t,.)$ in terms of a linear combination of the barycentric coordinates functions, that we will later use as tests functions.

Now, let us focus on the linear system on each triangle T_i.

The functions

$$\phi(x) = \sum_{i=1}^{N_T} \zeta_{i_1}(x)\phi_{p_{i_1}} + \zeta_{i_2}(x)\phi_{p_{i_2}} + \zeta_{i_3}(x)\phi_{p_{i_3}}$$

are used as test functions. We take the integral of the diffusion equation

$$\int_{T_i} \phi \frac{\partial u_i}{\partial t} dT = -\int_{T_i} < \nabla u, \nabla \phi > dT \tag{2.3}$$

where u_i is replaced by the approached solution $\tilde{u}_i$ in (2.3). Since we have the expressions for $\{\zeta_{i_1}, \zeta_{i_2}, \zeta_{i_3}\}$ and the equations holds for all ψ, the linear system is verified for the vector

$$[\tilde{u}_i] = (\tilde{u}_i(t,p_{i_1}), \tilde{u}_i(t,p_{i_2}), \tilde{u}_i(t,p_{i_3}))$$

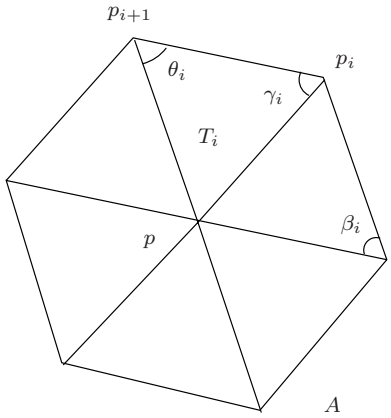


Figure 2.4: Angles intervening in the Laplace-Beltrami estimation

and

$$\frac{d[\tilde{u}_i]}{dt} = -A_i^{-1} C_i[\tilde{u}_i] \tag{2.4}$$

where the matrix A_i et C depend only on the areas and the angles' cotangents of the triangles T_i.

Now we assemble the equations on each node $p = p_0$, taking t small enough to take in account only the first neighbors of p. Taking the sum over all neighboring triangles $\{T_1, \ldots, T_m\}$ near p:

$$\sum_{i=1}^{m} \frac{d[\tilde{u}_i]}{dt} = -\sum_{i=1}^{m} A_i^{-1} C_i[\tilde{u}_i] \tag{2.5}$$

Writing (2.5) in matrix form, we find the importance of each neighbor for the computation of $\tilde{u}(t,p)$. This procedure simplifies considerably the equations and we finally obtain

$$\frac{d[\tilde{u}(t,p)]}{dt} = \sum_{i=1}^{m} w_i \left(\tilde{u}(t,p_i) - \tilde{u}(t,p)\right) \tag{2.6}$$

where the *weights* w_i are

$$w_i = \frac{cot\theta_i + cot\beta_i}{\sum_{i=1}^{m} A_i}, \tag{2.7}$$

where θ_i and β_i are the angles opposite to the edge p, p_i (see fig. 2.4).

This is the first but not the last time we find the cotangents of the interior

angles of the triangles to be in the center of the Laplace-Beltrami estimations. In particular, in Chapter 2, we will prove how this result is obtained again and again with different methods.

The numerical solution of the problem is defined by

$$\tilde{u}(t+\delta,p) = \tilde{u}(t,p) + \delta \sum_{i=1}^{m_p} w_i \left(\tilde{u}(t,p_i) - \tilde{u}(t,p)\right) \tag{2.8}$$

with the initial condition $\tilde{u}(0,p) = f(p)$. To guarantee the stability and convergence of the finite element scheme, The iteration step should be in the order of ([5, 20]):

$$\delta < d_{nodal} \cdot \left| \frac{\text{first derivatives of } u}{\text{second derivatives of } u} \right|$$

where d_{nodal} is the maximal distance between nodes.

The critical value of δ above which the numerical resolution will be unstable for the diffusion equation depends on thermal conductivity, which can be seen as conceptually equivalent to node distance: the closer the pair of nodes is, the stronger their mutual influence, and this mimics the situation of high thermal conductivity and fast heat propagation. Irregular node spacing, thus, parallels a situation of varying thermal conductivity across the field.

The critical temporal iteration step is also directly proportional to the minimal node distance, because instability tends to spread over the whole lattice, and therefore a single pair of nodes that are too close together will be enough to globally affect the result. Lattices possessing nodes in these conditions require very small iteration steps, and this may lead to exceedingly long processing times. Hence, a double constraint operates at the level of node separation: small inter nodal distance imply a small temporal iteration step and long computation times; on the other hand, very large inter nodal distances will eventually affect the reliability of the Laplacian estimations. There is thus, a trade-off between processing speed and estimation accuracy. If computation time is not unreasonably large, a conservative value for δ should be chosen.

Note that the finite element method is *local* in the sense that it is possible to estimate the Laplace-Beltrami operator on a vertex, using the information lying locally on its first ring neighboring triangles. On the other hand, with the parametric method it is necessary to compute the Laplace-Beltrami

estimator for all the vertex at the same time, which can be more costly if we only want to work on part of the surface. The parametric method does not provide local estimators.

For this work, we have implemented both approaches (appendix 10.5) to work on general triangulated surfaces (in [20] an implementation is done for closed triangulations where each vertex has a constant number of neighbors).

2.3.2 The Laplace-Beltrami on implicit surfaces

We will now introduce the implicit framework explaining the generalization of the heat equation for scalar images defined on implicit surfaces, as done by Bertalmio et al in [10] and [9].

Preliminaries for the implicit function approach

For a given vector $w \in \mathbb{R}^3$, we define P_w as the orthogonal projection matrix

$$P_w = I - \frac{w \otimes w}{\|w\|^2}.$$

As a consequence, the components of the matrix are

$$(P_w)_{ij} = \delta_{ij} - \frac{w_i w_j}{\|w\|^2},$$

with δ_{ij} standing as usual for the Kronecker delta.

Let $\mathcal{S} \subset \mathbb{R}^3$ a surface, and ν its normal at $x \in \mathbb{R}^3$. P_ν is an operator that projects vectors onto the tangent plane of $\mathcal{S}$ at point x.

Now, for X a vector field in $\mathbb{R}^3$, we may define the differential operator $P_X \nabla$ as the projection on the X-tangent plane, i.e.

$$(P_X \nabla)_i = \sum_{j=1}^{3} \left(\delta_{ij} - \frac{X_i X_j}{\|X\|^2} \right) \partial_{x_j}$$

where ∂_{x_j} is the gradient vector operator in $\mathbb{R}^3$.

Given a real valued function u on $\mathbb{R}^3$ and given a vector field Y on $\mathbb{R}^3$ we

will indistinctly use the notations

$$(P_X \nabla)\, u = P_X\, (\nabla u) = P_X \nabla u$$

$$P_X \nabla \cdot Y = \sum_{i=1}^{3} (P_X \nabla)_i\, Y_i.$$

Projectors are useful tools when applied for the vector field $X = \nabla\psi$ because if we fix $x \in \mathbb{R}^3$, $P_{\nabla\psi}$ projects vectors into the level-set surface of ψ passing through x. Therefore, if $x \in \mathcal{S}$, $P_{\nabla\psi}$ projects vectors onto $\mathcal{S}$ at x.

Particularly, $P_{\nabla\psi}\nabla u$ evaluated on $\mathcal{S}$ is the projection of the gradient of u onto $\mathcal{S}$. As a matter of fact, we define through this procedure what will be called in the remaining of this thesis the *surface* gradient of u on $\mathcal{S}$ or the *intrinsic* gradient:

$$\nabla_{\mathcal{S}} u := P_{\nabla\psi}\nabla u.$$

Let us now list some useful properties of this tool:

1. $P_w v \cdot z = v \cdot P_w z = P_w v \cdot P_w z$,
2. $(P_X \nabla)_i\, u = \nabla u \cdot P_X e_i$,
3. $P_{\nabla u}\nabla \cdot (P_{\nabla u} X) = \nabla \cdot (P_{\nabla u} X\, \|\nabla u\|)\, \frac{1}{\|\nabla u\|}$.

We represent the surface $\mathcal{S}$ by the zero level-set of a higher dimension function $\psi : \mathbb{R}^3 \to \mathbb{R}$, positive outside $\mathcal{S}$ and negative inside S, so that $\mathcal{S} = \left\{x \in \mathbb{R}^3 : \psi\,(x) = 0\right\}$.

In our implementation the function ψ is the signed distance function to $\mathcal{S}$.

The scalar or vector data $u\,(x)\,,\ x \in \mathbb{R}^3$ is then smoothly extended to a band surrounding $\mathcal{S}$.

Data in Implicit form Usually, the input image data is defined only in the surface, to extend the initial image u_0 over R^3, a method proposed by Chen et al (1999) is to look for a u such as:

$$\nabla u \cdot \nabla\varphi = 0,$$

solving the PDE:

$$\frac{du}{dt} + sign(\varphi)(\nabla u \cdot \nabla\varphi) = 0$$

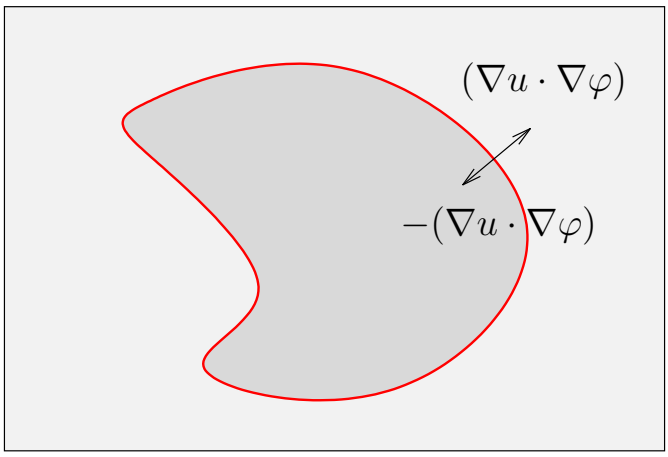


Figure 2.5: Data extension to implicit form

on a narrow band surrounding S.

Computation of the isotropic diffusion equation

If we want to smooth the data $u(x) : \mathcal{S} \to R$, where $\mathcal{S}$ is a surface, a way to do it is to follow the variational approach which consists in minimizing the corresponding *2-harmonic energy* (see [10] and [9]):

$$\frac{1}{2} \int_{\mathcal{S}} \|\nabla_{\mathcal{S}} u\|^2 \, d\mathcal{S},$$

which is equivalent to

$$\frac{1}{2} \int_{\Omega \in \mathbb{R}^3} \|P_{\nabla \psi} \nabla u\|^2 \, \delta(\psi) \, \|\nabla \psi\| \, dx. \tag{2.9}$$

This equivalence can be derived from the fact that $\nabla_{\mathcal{S}} u = P_{\nabla \psi} \nabla u$ (as we noticed in the previous section) combined with the fact that

$$\int_{\Omega} \delta(\psi) \, \|\nabla \psi\| \, dx = \int_{\mathcal{S}} d\mathcal{S}.$$

The equation that minimizes this energy, i.e. the corresponding harmonic

map is

$$\begin{aligned} \frac{\partial u}{\partial t} &= \Delta_{\mathcal{S}} u && (2.10)\\ &= \nabla_{\mathcal{S}} \cdot \nabla_{\mathcal{S}} u && (2.11)\\ &= \frac{1}{\|\nabla\psi\|} \nabla \cdot (P_{\nabla\psi} \nabla u) \|\nabla\psi\| && (2.12) \end{aligned}$$

$\nabla_{\mathcal{S}}$ and $\Delta_{\mathcal{S}}$ stand here as usual for the intrinsic gradient and intrinsic Laplacian (the Laplace-Beltrami operator). This can be obtained computing the gradient descent of (2.9) (See [8] for details on this particular question).

Considering

$$E(u) = \frac{1}{2} \int_{\Omega \in \mathbb{R}^3} \|P_{\nabla\psi} \nabla u\|^2 \delta(\psi) \|\nabla\psi\| \, dx$$

and μ a perturbation of u, we have that (for the complete computation see appendix 10.2):

$$\left. \frac{d}{dt} \right|_{t=0} E(u + t\mu) = - \int_{\mathcal{S}} \frac{1}{\|\nabla\psi\|} \nabla \cdot (P_{\nabla\psi} \nabla u \, \|\nabla\psi\|) \, \mu d\mathcal{S} \qquad (2.13)$$

and since the last term must be identically equal to zero for all μ, we can conclude that at the zero level-set ψ,

$$\frac{1}{\|\nabla\psi\|} \nabla \cdot (P_{\nabla\psi} \nabla u) \|\nabla\psi\| = 0.$$

We can now extend this to the whole domain Ω, as long as we assume that $\|\nabla\psi\| \neq 0$ at least in a band surrounding the surface.

Note that this result could have been intuitively predicted. Indeed, if we replace the expressions in the PDE (2.1) with the intrinsic ones, meaning that we start with the intrinsic problem

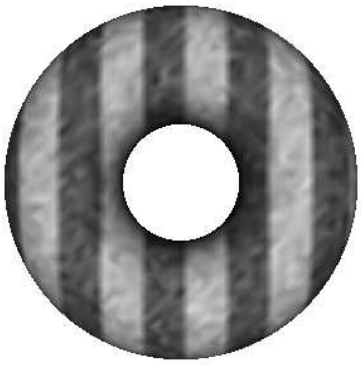
$$\begin{cases} \frac{\partial}{\partial t} u(x,t) = \Delta_s u(x,t) \\ u(x,0) = u_0(x) \end{cases}, \qquad (2.14)$$

where $\Delta_S u(x,t)$ is the Laplace-Beltrami operator:

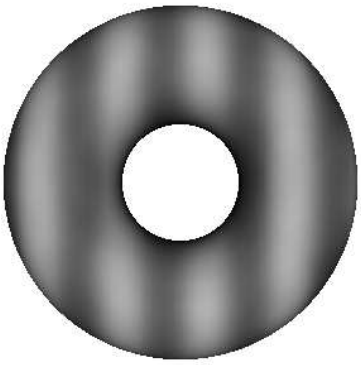
$$\Delta_S u(x) = \nabla_S \cdot \nabla_S u(x),$$

and we arrive at the same result as in the Euclidean space case.

A numerical algorithm can be implemented using widely known numerical schemes for 3D operators without special considerations of the fact that we



Noisy stripes image
over a torus



Regularized image
$dt = 0.1, 10$ iterations

Noisy smiley image
over a sphere

Regularized image
$dt = 0.1, 25$ iterations

Figure 2.6: Examples of isotropic regularization on scalar images

are dealing with surfaces. Everything is made on the cartesian grid. This is one of the reasons that makes this approach so attractive to implement for more complicated PDEs.

2.4 Summary and conclusions

In this chapter we have introduced the implicit and explicit surface representations, and discussed the related literature. We explained the generalization of the isotropic regularization of flat images to surfaces to images defined for the two principal surface representations. We chose to explain in detail the explicit (finite element and parametric approximation, [20]) and

implicit ([8, 10, 9]) Laplace-Beltrami regularization, which in the addition to the implicit anisotropic regularization, were the state of the art of image surface regularization methods when this work began. In the next chapter, we will show a new, simpler estimation for the Laplace-Beltrami operator defined on triangulated surfaces, and two novel regularization methods: the anisotropic regularization and the Beltrami flow for images defined on triangulated surfaces.

Chapter 3

Regularization on explicit surfaces

In this chapter we develop our work done[1] on scalar images regularization over triangulated surfaces. We begin with the isotropic regularization, presenting a new numerical approach for the estimation of the Laplace-Beltrami operator. In the second part of the first section, we introduce the Discrete Exterior Calculus approach ([41]), which results relate to our approach. In the second section we use the same framework to deal with anisotropic regularization, and generalize the numerical method to estimate a whole class of operators. In the third section we introduce the Beltrami framework and generalize it for images defined on triangulated surfaces, proposing an numerical implementation based on the previous sections. We also show results on synthetic images.

We begin with the isotropic diffusion and present a different approach from those shown in chapter 2. We then discuss the anisotropic case together with the Beltrami flow regularization.

3.1 Isotropic diffusion

In this subsection, we present an original procedure to obtain a *discrete* Laplace-Beltrami operator. We later extend this procedure to more general differential operators and show that it is equivalent to the classical estima-

[1] Work published in [83, 82, 54].

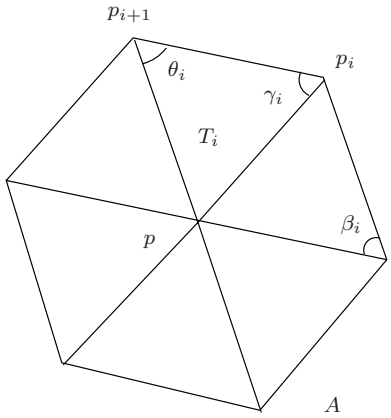


Figure 3.1: Area used for the average estimation and elements involved

tion shown in section ss:fem.

We then compare this procedure to recent results that use the Discrete Exterior Calculus ([41]) as a major tool.

3.1.1 An area-averaged Laplace-Beltrami estimation

Let S be a triangulated surface, $(u, v) \mapsto x \in S$ a conformal parameterization, and $I : S \to \mathbb{R}$ a scalar image painted on the surface.

Using some ideas sketched in [30, 59] for surface fairing, we consider

$$\begin{aligned} \Delta_S \hat{I}(x) &= \frac{1}{A} \int_A \Delta_S I(x) dx \\ &= (\, \Delta_S I(p) \quad \text{averaged over the region } A\,) \\ &\approx \Delta_S I(p) \end{aligned}$$

as our choice for a *discrete Laplace-Beltrami* on the node p, where A is the area defined by the triangles surrounding p (see Figure 3.1). Let p_i for $i = , 1 \ldots, n_{neighbors}$ be the neighboring points of p, ordered counterclockwise. X_i for $i = , 1 \ldots, n_{neighbors}$ stands for the p_i counterparts in the parametric space, for an arbitrary conformal parametrization $X : \mathbb{R}^2 \to S$, i.e. $X(X_i) = p_i$.

By the Gauss Theorem, using the fact that F is linear on each triangle and

that $\nabla I(x)$ is constant, we have:

$$\begin{aligned}\frac{1}{A}\int_A \nabla_S \cdot \nabla_S I(x)dx &= \frac{1}{A}\int_{\partial A} \nabla_S I(x)\cdot \mathbf{n}_{u,v} dl \\ &= \frac{1}{A}\sum_{T\in A}\int_{\partial A\cap T} \nabla_S I(x)\cdot \mathbf{n}_{u,v} dl \\ &= \frac{1}{A}\sum_{T\in A} \nabla_S I(x)\cdot [X_i - X_{i+1}]^{\perp} \\ & \quad x\in \partial A\cap T \text{ for each triangle.}\end{aligned}$$

Now, let us consider a fixed triangle T. Since F is linear on its surface, using the linear basis functions B_l over the triangle T and the fact that $(B_i + B_j + B_k)(u,v) = 1$, we get that

$$\begin{aligned} I(x) &= I(p)B_i(u,v) + I(p_i)B_j(u,v) + I(p_{i+1})B_k(u,v), \\ \nabla_{u,v} I(x) &= I(p)\nabla_{u,v}B_i(u,v) + I(p_i)\nabla_{u,v}B_j(u,v) + I(p_{i+1})\nabla_{u,v}B_k(u,v) \\ &= (I(p_i) - I(p))\nabla_{u,v}B_j(u,v) + (I(p_{i+1}) - I(p))\nabla_{u,v}B_k(u,v).\end{aligned}$$

Computing the gradients of the basis functions we obtain,

$$\begin{aligned}\nabla B_j(u,v) &= \frac{1}{2A_T}(X - X_{i+1})^{\perp} \\ \nabla B_k(u,v) &= \frac{1}{2A_T}(X_i - X)^{\perp}\end{aligned}$$

and as a consequence we obtain

$$\nabla_{u,v} I(x) = \frac{1}{2A_T}\left[(I(p_i) - I(p))(X - X_{i+1})^{\perp} + (I(p_{i+1}) - I(p))(X_i - X)^{\perp})\right]. \tag{3.1}$$

Since A_T is proportional to the sine of any angle of the triangle we get

$$\nabla_{u,v} I(x)\cdot [X_i - X_{i+1}]^{\perp} = [cot(p_{i+1})(I(p_i) - I(p)) + cot(p_i)(I(p_{i+1}) - I(p))] \tag{3.2}$$

and therefore

$$\begin{aligned}\Delta_S I(p) \approx \frac{1}{A}\sum_{T\in A}\frac{1}{A_T}[&(X_i - X)\cdot(X_i - X_{i+1})(I(p) - I(p_{i+1})) \quad + \\ &(X_{i+1} - X)\cdot(X_i - X_{i+1})(I(p_i) - I(p))],\end{aligned}$$

or

$$\Delta_S I(p) \approx \frac{1}{A} \sum_{T \in A} (cot\theta_i + cot\beta_i)(I(p) - I(p_i)). \tag{3.3}$$

where θ_i, β_i are the two opposite angles of the edge (p, p_i) in the area A (see fig. 3.1).
The conformal map preserves angles, so we can use the original triangle $(p, p_i, p_{i+1}) \subset S$ instead of the one in $\mathbb{R}^2$.

Remark As announced in the introduction of this chapter, we arrived to the same expression than the one obtained by Chung et al. in [20] and [41]. This expression has been proved to be the *best estimation* for the Laplace-Beltrami operator in [106] in terms of convergence.

Algorithm We first compute an estimation of the Laplace Beltrami weights on each vertex p by

$$\widehat{\Delta_S I(p)} = \frac{1}{A} \sum_{T \in A} (cot\theta_i + cot\beta_i)(I(p) - I(p_i)) \tag{3.4}$$

where θ_i, β_i are the two opposite angles of the edges (p, p_i) in the area A.

We then apply the following PDE using this approximation,

$$\frac{\partial u(x,t)}{\partial t} = \Delta_S u(x,t) \tag{3.5}$$

where $u(x, 0) = I(x)$, $x \in S$ using finite differences.

3.1.2 Discrete Laplace-Beltrami via Discrete Exterior Calculus (DEC)

Introduction to DEC In [41] a Discrete Exterior Calculus theory is developed using only discrete combinatorial and geometric operations on a simplicial complex and its geometric dual. The derivation of these may require that the objects on the discrete mesh, (not necessarily the mesh itself) are interpolated. This theory includes not only discrete equivalents of differential forms but also discrete vector fields and the operators acting on these objects. Definitions are given for discrete versions of all the usual operators of exterior calculus.

The presence of forms and vector fields allows us to address their various interactions, which are important in applications. In many examples, it is found that the formulas derived from DEC are identical to the existing formulas in the literature. It is shown that the circumcentric dual of a simplicial complex plays a useful role in the metric dependent part of this theory. The appearance of dual complexes leads to a proliferation of the operators in the discrete theory. There are many constraints in numerical algorithms that naturally involve differential forms. Preserving those features directly on the discrete level is an interesting achievement. The purely discrete point of view is pushed as far as possible, using interpolation as a very useful device.

Even if this work ([41]) propose interesting new discrete operators to approximate EDPs over triangulations, a complete DEC theory has not been fully developed. There are still missing some important points. There are several definitions for the interior product, sharps and flats and it is unclear how to combine them to obtain nice properties. The condition for the mesh to be flat for the case of primal vector fields is very imitating. Discrete tensors are still not well defined. And of course, the convergence and stability study is still in progress. In [41] some hints are done on the direction to look for, but the work is still waiting to be done.

Given a smooth manifold M the following spaces and operators are defined:

- p-forms in $\Omega^p(M)$ and vector fields in $\aleph\chi(M)$
- Exterior derivative $\mathbf{d} : \Omega^p(M) \to \Omega^p + 1(M)$
- Hodge star $* : \Omega^p(M) \to \Omega^n - p(M)$
- Wedge product $\wedge : \Omega^k(M) \times \Omega^l(M) \to \Omega^{k+l}(M)$
- Sharp map $\sharp : \Omega^{\mathbf{1}}(M) \to X(M)$
- Flat map $\flat : X(M) \to \Omega^{\mathbf{1}}(M)$
- Interior product (contraction) i_χ of forms with vector field
- Lie derivative $\pounds_\chi$ of forms and vector fields

Simplicial and dual complexes represent the discretized smooth manifold. Cochains represent the discretized differential form. A coboundary operator models the discrete exterior derivative. All these assumptions are common to the other DEC theories.

For metric-dependent operators, they use circumcentric dual complex to define metric-dependent operators of exterior calculus Hodge star ($*$), ($\flat$) and sharp ($\sharp$). Then they use these to define metric-dependent operators gradient, curl and Laplace-Beltrami. $*$ and $\flat$ are used to define divergence, which should depend on metric only via the Lie derivative of volume form.

For the metric-independent operators, they define exterior derivative ($\mathbf{d}$) as coboundary, as usual in this field; the wedge product ($\wedge$). Define interior product, c.a.d. contraction of a vector field and a form, in two ways once via a new definition involving $*$,$\wedge$ and $\flat$, and once via the dynamic definition of Bossavit [2003]. Define Lie derivative ($\pounds$) in two ways once using interior product and once via a dynamic definition. The two definitions of interior product and Lie derivative may be the same.

Review of Smooth Flat and Sharp Smooth sharp ($\sharp$) and flat ($\flat$) are inverses of each other. For form α, vector fields X and V:

$$\begin{aligned} \alpha^\sharp \dot{V} &= \alpha(V) \\ X^\flat(C) &= X\dot{V} \\ (X^\flat)^\sharp \dot{V} &= X^\flat(V) = X\dot{V} \\ (\alpha^\sharp)^\flat(V) &= \alpha^\sharp \dot{V} = \alpha V. \end{aligned}$$

Gradient $\Delta f = (\mathbf{d}f)^\sharp$ or equivalently $(\Delta f)^\flat = \mathbf{d}f$.

Discrete Laplace-Beltrami derivation In the smooth case the Laplace-Beltrami on functions is a special case of the more general Laplace-deRham operator $\nabla : \Omega^k(M) \to \Omega^k(M)$ defined by $\nabla = \mathbf{d}\delta + \delta\mathbf{d}$.

Let us compute here ∇f on a primal vertex σ^0 where $f \in \Omega_d^0(K)$ and K is a (not necessarily flat) triangle mesh in $\mathbb{R}^3$. Suppose that K is oriented by orienting all its triangles counterclockwise. Since $\delta f = 0$ by definition, we have that

$$\begin{aligned} \left\langle \nabla f, \sigma^0 \right\rangle &= \left\langle \delta \mathbf{d} f, \sigma^0 \right\rangle \\ &= \left\langle *\mathbf{d} * \mathbf{d} f, \sigma^0 \right\rangle. \end{aligned}$$

Now by using the definition of discrete Hodge star followed by the discrete

Stokes theorem we get

$$\begin{aligned}\left\langle *\mathbf{d} * \mathbf{d}f, \sigma^0 \right\rangle &= \frac{|\sigma^0|}{-|\star\,\sigma^0|} \left\langle \mathbf{d} * \mathbf{d}f, \star\sigma^0 \right\rangle \\ &= \frac{-1}{|\star\,\sigma^0|} \left\langle *\mathbf{d}f, \partial(\star\sigma^0) \right\rangle .\end{aligned}$$

The explanation for the use of signed volume $-|\star\,\sigma^0|$ was given in Rem. 4.1.2 in [41]. Thus

$$\left\langle \nabla f, \sigma^0 \right\rangle = \frac{1}{|\star\,\sigma^0|} \left\langle *\mathbf{d}f, \partial(\star\sigma^0) \right\rangle .$$

By definition of the dual boundary,

$$\partial(\star\sigma^0) = \sum_{\sigma^1 \succ \sigma^0} \star(s_{\sigma^1}\sigma^1)$$

where $s_{\sigma^1} = \pm 1$ is a sign that depends on the orientation of K and σ^1. In dimension 2, with triangles of K oriented counterclockwise, the definition of dual boundary dictates that the edges $s_{\sigma^1}\sigma^1$, which means that all edges incident on σ^0 are now all pointing outwards. Thus,

$$\begin{aligned}\left\langle *\mathbf{d}f, \partial(\star\sigma^0) \right\rangle &= \left\langle *\mathbf{d}f, \sum_{\sigma^1 \succ \sigma^0} \star\sigma^1 \right\rangle \\ &= \sum_{\sigma^1 \succ \sigma^0} \left\langle *\mathbf{d}f, \star\sigma^1 \right\rangle .\end{aligned}$$

Another use of the definition of discrete Hodge star gives

$$\left\langle *\mathbf{d}f, \star\sigma^1 \right\rangle = \frac{\star|\sigma^1|}{|\sigma^1|} \left\langle \mathbf{d}f, \star\sigma^1 \right\rangle .$$

But then by discrete Stokes theorem we have that

$$\frac{\star|\sigma^1|}{|\sigma^1|} \left\langle \mathbf{d}f, \star\sigma^1 \right\rangle = f(v) - f(\sigma^0)$$

where $\sigma^1 = [\sigma^0, v]$. Putting all this together we get that

$$\left\langle \nabla f, \sigma^0 \right\rangle = \frac{1}{|\star\,\sigma^0|} \sum_{\sigma^1 = [\sigma^0, v]} \frac{|\star\,\sigma^1|}{|\sigma^1|} (f(v) - f(\sigma^0)).$$

The above expression corresponds to eq. 2.7 found using the finite element method.

3.2 Anisotropic diffusion

3.2.1 Anisotropic diffusion on flat images

A method to smooth images while preserving boundaries, called *anisotropic diffusion* has been first proposed by Perona and Malik in [68]. This non-linear method overcomes some limitations of the isotropic smoothing. Since then, the idea has been widely used and extended by many authors, see for instance [52, 86, 4].

These algorithms use a diffusion process, in which the diffusion coefficient is chosen to vary spatially in a way that it encourages intraregion smoothing in preference to interregion smoothing. It is called *anisotropic* because the diffusion is made with different weights for different spatial directions.

In [68], the anisotropic diffusion flow for an scalar image $I : \mathbb{R}^2 \to \mathbb{R}$ is given in the form:

$$I_t = \operatorname{div}\left(c(x, y, t)\, \nabla I\right) \tag{3.6}$$

where $c(x, y, t)$ can be chosen to be a function of the image gradient, for instance,

$$\begin{aligned} c_1(x, y, t) &= \exp(-(|\nabla I|/K)^2) \\ c_2(x, y, t) &= \frac{1}{|\nabla I|}\ . \end{aligned}$$

Choosing these kind of functions leads to a diffusion flow that not only preserves the edges, but sharpens them. See fig 3.2 for an example. In this section, we study the simple anisotropic diffusion that corresponds to use c_2 on the flat image case.

The Beltrami flow, which is presented below on section 3.4, provides a way to perform image regularization *between* the isotropic and anisotropic diffusion, using a parameter that tunes its behavior.

In [96], the idea of anisotropic regularization has been extended for far more general models and for the case of multivalued images (including color images), defined on $\mathbb{R}^n$ (flat and volumetric images). With the approach de-

veloped in that work, the *anisotropy* of the diffusion flow can be controlled by defining a *tensor field*.

3.2.2 Area-averaged estimation

We want to obtain an approximation of $\nabla_S \cdot (\frac{1}{|\nabla_S I(p)|} I(p))$, ($p$ being one of the nodes of the triangulation), using a mean of this quantity in the area A defined by the triangles immediately surrounding p,

$$\nabla_S \cdot \left(\frac{1}{|\nabla_S I(p)|} \nabla_S I(p) \right) \approx \frac{1}{A} \int_A \nabla_S \cdot \left(\frac{1}{|\nabla_S I(x)|} \nabla_S I(x) \right) dx. \qquad (3.7)$$

Using again the Gauss Theorem and the fact that F is linear on the triangle combined with the fact that $\nabla I(x)$ is constant we obtain,

$$\begin{aligned} \frac{1}{A} \int_A \nabla_S \cdot \left(\frac{1}{|\nabla_S I(x)|} \nabla_S I(x) \right) dx &= \frac{1}{A} \int_{\partial A} \frac{\nabla_S I(x) \cdot \mathbf{n}_{u,v}}{|\nabla_S I(x)|} \, dl \\ &= \frac{1}{A} \sum_{T_i \in A} \int_{\partial A \cap T_i} \frac{\nabla_S I(x) \cdot \mathbf{n}_{u,v}}{|\nabla_S I(x)|} \, dl \\ &= \frac{1}{A} \sum_{T_i \in A} \frac{\nabla_S I(x) \cdot \mathbf{n}_{u,v}}{|\nabla_S I(x)|} \, dl \\ &\quad for\, x \in \partial A \cap T_i \end{aligned}$$

Using Equation (3) we get

$$\nabla_S \cdot (\frac{1}{|\nabla_S I(p)|} I(p)) \approx \frac{1}{A} \sum_{T_i \in A} \frac{1}{|\nabla_S I(x)|} \nabla_S I(x) \cdot [p_i - p_{i+1}]^{\perp}$$

$$= \frac{1}{A} \sum_{T_i \in A} \frac{1}{|\nabla_S I(x)|} \left[\cot \theta_i (I(p_i) - I(p)) + \cot \gamma_i (I(p_{i+1}) - I(p)) \right].$$

for x in the triangle T_i, where θ_i is the internal angle of the node p_{i+1} and

Original noisy image

Isotropic Diffusion

Anisotropic Diffusion
using c_2 in (3.6)

Figure 3.2: Isotropic vs Anisotropic diffusion

γ_i is the internal angle of the node p_i, and

$$\begin{aligned}
\cot\theta_i &= \frac{< p_{i+1}-p, p_{i+1}-p_i >}{2A_{T_i}}, \\
\cot\gamma_i &= \frac{< p_i-p_{i+1}, p_i-p >}{2A_{T_i}}, \\
A_{T_i} &= \frac{1}{2}|(p_{i+1}-p)\times(p_i-p)|, \\
|\nabla_{u,v}I(x)| &= \frac{1}{2A_T}\left|(I(p_i)-I(p))(p-p_{i+1})+(I(p_{i+1})-I(p))(p_i-p)\right|.
\end{aligned}$$

In the resulting algorithm, we compute the cotangents of the angles and the triangles areas once and store the resulting values. At each time step, we recompute the gradient's norm. Unlike in the isotropic diffusion algorithm, where the Laplace-Beltrami weights depend only on the surface geometry, in the anisotropic case, the weights also depend on the image function. As a consequence the weights need to be updated at each iteration, since the image function itself changes at each iteration. For more details on the algorithm, we reproduced our c++ code in the Appendix 10.5.

3.3 Extension to a more general case

We look this time for a local approximation of $\nabla_{\mathcal{S}}\cdot(\phi(|\widehat{\nabla_{\mathcal{S}}I(p)}|)\nabla_{\mathcal{S}}I(p))$ on the nodes of S . As usual, p stands for a node of the triangulated surface and ϕ denotes a standard real function.

Following exactly the same method than before, we take the spatial mean of this quantity in the area A defined by the triangles immediately surrounding p (see fig 3.1)

$$\nabla_{\mathcal{S}}\cdot(\phi(|\nabla_{\mathcal{S}}I(p)|)\nabla_{\mathcal{S}}I(p)) \approx \frac{1}{A}\int_A \nabla_{\mathcal{S}}\cdot(\phi(|\nabla_{\mathcal{S}}I(x)|)\nabla_{\mathcal{S}}I(x))dx$$

and we retrieve that

$$\frac{1}{A}\int_A \nabla_{\mathcal{S}}\cdot(\phi(|\nabla_{\mathcal{S}}I(x)|)\nabla_{\mathcal{S}}I(x))dx = \frac{1}{A}\int_{\partial A}\phi(|\nabla_{\mathcal{S}}I(x)|)\nabla_{\mathcal{S}}I(x)\cdot\mathbf{n}_{u,v}dl$$

$$
\begin{aligned}
&= \frac{1}{A}\sum_{T_i\in A}\int_{\partial A\cap T}\phi(|\nabla_{\mathcal{S}}I(x)|)\nabla_{\mathcal{S}}I(x)\cdot\mathbf{n}_{u,v}dl \\
&= \frac{1}{A}\sum_{T_i\in A}\phi(|\nabla_{\mathcal{S}}I(x)|)\nabla_{\mathcal{S}}I(x)\cdot[X_i-X_{i+1}]^{\perp}\mathcal{I}_{\partial A\cap T_i}(x)
\end{aligned}
$$

where p, p_i, p_{i+1} stand for the vertex of the triangle T_i. X, X_i, X_{i+1} represent the correspondences of the vertex in the (u,v) space; and $\mathcal{I}_{\partial A\cap T_i}(x)$ stands for the indicator function over the set $\partial A\cap T_i$.

Let us now take a fixed triangle T_i. Let B_l, $l=1,2,3$ be the linear basis functions over the triangle T_i. Because of $(B_1+B_2+B_3)(u,v)=1$,

$$
\begin{aligned}
\nabla_{\mathcal{S}}I(x) &= I(p)\nabla_{\mathcal{S}}B_1 + I(p_i)\nabla_{\mathcal{S}}B_2 + I(p_{i+1})\nabla_{\mathcal{S}}B_3 \\
&= (I(p_i)-I(p))\nabla_{\mathcal{S}}B_2(u,v) + (I(p_{i+1})-I(p))\nabla_{\mathcal{S}}B_3(u,v) \\
&= \frac{1}{2A_{T_i}}[(I(p_i)-I(p))(X-X_{i+1})^{\perp} \\
&+ (I(p_{i+1})-I(p))(X_i-X)^{\perp})]
\end{aligned}
$$

and we find that

$$
\begin{aligned}
\nabla B_2(u,v) &= \frac{1}{2A_{T_i}}(X-X_{i+1})^{\perp} \\
\nabla B_3(u,v) &= \frac{1}{2A_{T_i}}(X_i-X)^{\perp}
\end{aligned}
$$

where A_{T_i} is the area of the triangle T_i. Using the fact that A_{T_i} is proportional to the sine of any angle of the triangle,

$$
\nabla_{\mathcal{S}}I(x)\cdot[X_i-X_{i+1}]^{\perp} = [cot(p_{i+1})(I(p_i)-I(p)) + cot(p_i)(I(p_{i+1})-I(p))].
$$

We get

$$
\begin{aligned}
\widehat{\Delta_g I(p)} &= \frac{\phi(|\widehat{\nabla_{\mathcal{S}}I(p)}|)}{A}\sum_{T_i\in A}\{\,\phi(|\widehat{\nabla_{\mathcal{S}}I(p)}|)\,\cdot \\
&\quad [\cot\theta_i(I(p_i)-I(p)) + \cot\gamma_i(I(p_{i+1})-I(p))]\,\}
\end{aligned}
$$

for x in the triangle T_i, where θ_i is the internal angle of the node p_{i+1} and γ_i is the internal angle of the node p_i. Moreover,

$$
|\widehat{\nabla_{\mathcal{S}}I(p)}| = \frac{1}{2A_T}\sum_{T_i\in A}|(I(p_i)-I(p))(p-p_{i+1}) + (I(p_{i+1})-I(p))(p_i-p)|
$$

Note that the approximation is independent from the chosen parametrization, and can be used to approximate other differential operators of the form

$$\nabla_{\mathcal{S}} \cdot (\phi(|\nabla_{\mathcal{S}} I(p)|) \nabla_{\mathcal{S}} I(p)) \tag{3.8}$$

for ϕ being a function $\phi : \mathbb{R} \rightarrow \mathbb{R}$.

3.4 Beltrami flow

3.4.1 Introducing the Beltrami framework

The problem of regularizing noisy data defined on non flat - either implicit or intrinsic - surfaces has been tackled with two functionals that operate on the space of embedding maps of Riemannian manifolds (see [48] for a non-variational approach.)

- **The Polyakov action:** It was introduced in the Beltrami framework for images over flat surfaces [85, 86, 89, 47, 84]. It uses a local and intrinsic-parametric description of the manifolds and treats the metric as a dynamic variable.
- **Harmonic maps:** It has been recently used to denoise images on manifolds [10, 58, 12, 14, 79]. It relies on an implicit description of the surface within which the noisy data are constrained to live. The metric is a parameter of the process.

We clarify later in this chapter the relationship between the intrinsic Polyakov action of the Beltrami framework [85, 86, 89, 47, 84] and the implicit harmonic energy functional [10, 58, 12, 14, 79].

Although the functionals are very similar, there are differences in the way various problems are formulated. It has direct consequences on the way the functionals should be applied. Specifically, for the case of denoising images on non-flat surfaces, there are differences in the definition of the manifolds and the embedding functions.

In many problems in computer vision, we usually have an underlying manifold - that can be either flat or curved - over which the features are defined. A typical situation is when the image rectangle is the underlying flat manifold and when at each pixel, we assign values such as intensity, color, gradient

value, gradient direction, motion vector, disparity vector, texture characteristics, etc. This is easily described mathematically as a **fiber bundle** in which a space is attached to each point in the base manifold. The spaces at different points of the manifolds are isomorphic. A choice of one point in the attached space for every point in the base manifold is called a **section**. If the attached space is a vector space, then this section is called a vector field.

In order to better understand the difference between the two formulations, we consider the case of a gray-value image defined on a surface (flat or not). In the harmonic energy approach, it is usually assumed that the map is defined from the 2D surface (the base manifold) to the real axis (the fiber). It means that **the metric of the base manifold** is used for the derivatives and the fiber's 1D flat metric is used for the values of the scalar field. If the chosen norm is L_2, we get a linear diffusion process whether the base manifold is flat or not. In the case of L_1 norm or the Φ formulation [52, 77, 25] we get non-linear flows. These flows only depend on the absolute value of the gradient, defined over the base manifold [14, 93].

In the Beltrami framework [85, 86, 47], the basic object is **the section embedded in the fiber bundle**. For a flat gray-value image the graph of the intensity function is thereby the section of this 3D fiber bundle and is the primary object of interest. The metric of the fiber bundle induces a metric on this section and both are used in the functional. The flow depends on the **geometry of the data** and not only on the geometry of the base manifold. This means that **the image metric (i.e. the section metric) evolves with the flow**. It also means that the flow may depend on the direction of the gradients and not only on their amplitudes. The Beltrami flow which has been shown to interpolate, for flat gray-value images, between the L_2 norm and the L_1 norm [86, 88] is generalized in this work to images over non-flat surfaces. This work show the way the Beltrami flow interpolates between the L_2 and L_1 flows on non-flat surfaces (see [14, 93] for references). A common feature of both frameworks is the ability to deal with non-flat feature spaces as done in [89, 93, 84, 97, 15].

3.4.2 Beltrami flow over non-flat surfaces

Assume that we have a surface Σ and a Riemannian structure on Σ i.e a metric $(\tilde{g}_{ij})$. For cases where the surface is given as an embedding in $\mathbb{R}^3$ we can take, for example, the induced metric of Σ as in the previous

Section. The regularization acts on the image only and does not change the geometry of the surface. The metric $(\tilde{g}_{ij})$ is constant with respect to the time. In addition we are given a scalar field (image) i.e. a function U, defined on Σ. In a local coordinate system (x, y) of the surface Σ, the line element is

$$ds_\Sigma = \tilde{g}_{11}dx^2 + 2\tilde{g}_{12}dxdy + \tilde{g}_{22}dy^2$$

We add now a third dimension, perpendicular to the surface and we describe the scalar field $U(x, y)$ as a surface S embedded in the 3D fiber bundle $\mathcal{M} = \Sigma \times \mathbb{R}$. The surface S is a section of the fiber bundle. The embedding coordinates are

$$\left(x(u, v) = u, y(u, v) = v, X^3(u, v) = U(x, y)\right)$$

or by abuse of notations: $(x, y, U(x, y))$, where (x, y) are coordinates on S .

The line element in the manifold/fiber bundle $\mathcal{M}$ is

$$ds_\mathcal{M} = \tilde{g}_{11}dx^2 + 2\tilde{g}_{12}dxdy + \tilde{g}_{22}dy^2 + \beta^2 dU^2$$

or, in equivalent way, the metric of the 3D manifold $\mathcal{M}$ is

$$(h_{ij}) = \begin{pmatrix} \tilde{g}_{11} & \tilde{g}_{12} & 0 \\ \tilde{g}_{21} & \tilde{g}_{22} & 0 \\ 0 & 0 & \beta^2 \\ . & & \end{pmatrix}$$

Assuming an isometric embedding i.e. $ds_\mathcal{S} = ds_\mathcal{M}$, we obtain the induced metric on the section S :

$$ds_\mathcal{S} = \tilde{g}_{11}dx^2 + 2\tilde{g}_{12}dxdy + \tilde{g}_{22}dy^2 + \beta^2 \left(U_x dx^2 + 2U_x U_y dxdy + U_y^2 dy^2\right) \ ,$$

where β^2 takes into account the differences in dimensions between the spatial directions and the intensity one. The metric elements on S , are therefore:

$$g_{ij} = \tilde{g}_{ij} + \beta^2 U_i U_j \ . \tag{3.9}$$

Denote by G the matrix whose elements are g_{ij}. Using this metric we can calculate the Laplace-Beltrami operator:

$$\Delta_g U = \frac{1}{\sqrt{g}} \mathrm{Div}\left(\sqrt{g} G^{-1} \nabla U\right) \ , \tag{3.10}$$

where $g = \det(G)$. Note that here Div and ∇ are two-dimensional. The equation of motion that results from the Polyakov action is:

$$U_t = \Delta_g U + \Gamma^3_{ab}(\partial_i X^a)(\partial_j X^b)g^{ij} \ ,$$

where

$$\Gamma^c_{ab} = \frac{1}{2}h^{cd}\left(\partial_a h_{db} + \partial_b h_{ad} - \partial_d h_{ab}\right) \ .$$

In our case

$$\Gamma^3_{ab} = \frac{1}{2}h^{3d}\left(\partial_a h_{db} + \partial_b h_{ad} - \partial_d h_{ab}\right) = \frac{1}{2}h^{33}\left(\partial_a h_{3b} + \partial_b h_{a3} - \partial_3 h_{ab}\right) = 0,$$

where the second equality derives from the decoupling of the third dimension from the other two (i.e. the independence of the fiber on the base manifold) and the last equality comes from the fact that $h_{33} = const$ and that the other elements do not depend on U (note that $\partial_3 \equiv \partial/\partial X^3 = \partial/\partial U$) .

The motion equation is therefore

$$U_t = \Delta_g U \ ,$$

where $\Delta_g U$ is given in (3.10) with the induced metric we derived in (3.9).

3.4.3 Example: Gray image on the sphere

In order to apply the above equations to an image "painted" on the sphere, we need to compute the metric $\tilde{g}_{ij}$ of the sphere. We choose to represent the sphere by stereographic coordinates from the south pole. The metric in this case is (derivation on appendix 10.1)

$$\tilde{g}_{ij} = \frac{4}{1 + x^2 + y^2}\delta_{ij} \ . \tag{3.11}$$

In general we write

$$g_{ij} = \tilde{g}_{ij} + \beta^2 U_i U_j \ ,$$

where $U_i = \partial U/\partial x^i$ (we denoted here $x^1 = x$ and $x^2 = y$ for convenience).Define

$$A(x, y) = \frac{4}{1 + x^2 + y^2} \ .$$

We find, consequently, that

$$g = det(g_{ij}) = A^2 + A\beta^2(U_x^2 + U_y^2) \ ,$$

and the motion equation becomes

$$U_t = \frac{1}{\sqrt{g}}\text{Div}\left(\frac{1}{\sqrt{g}}\begin{pmatrix} A+\beta^2U_y^2 & -\beta^2U_xU_y \\ -\beta^2U_xU_y & A+\beta^2U_x^2 \end{pmatrix}\nabla U\right) \ .$$

3.4.4 Estimation of the differential operator

Let S be a surface parametrized by

$$P(\mathbf{u}) = \{x(\mathbf{u}), y(\mathbf{u}), z(\mathbf{u}) : \mathbf{u} = (u^1, u^2) = (u, v) \in D\}.$$

Let $G = (G_{i,j})$, $G_{i,j} = \langle P_i, P_j \rangle$ be the Riemanian metric tensor for this parametrization.
Then, for a function $I : S \to R$,

$$\nabla_P I(x) = \nabla_{u,v} I(x) = \sum_{i,j=1}^{2} G^{i,j}\frac{\partial I}{\partial u^i}P_i$$

where $(G^{i,j}) = G^{-1}$. We will use the (u, v)-space with the metric induced by the local parameterization so that (3.10) can be expressed as:

$$I_t = \Delta_g I \ = \ \frac{1}{\sqrt{1+\beta|\nabla_{u,v}I|^2}}\,\text{Div}\left(\sqrt{1+\beta|\nabla_{u,v}I|^2}\,\nabla_{u,v}I\right) \ .$$

Note that if we take the limit

$$\begin{aligned} \lim_{\beta\to 0}\Delta_g I &= \Delta_{u,v} I \\ \lim_{\beta\to \text{inf}}\Delta_g I &= \frac{1}{|\nabla_{u,v}I|}Div(|\nabla_{u,v}I|\nabla_{u,v}I) \end{aligned}$$

we find the L_2 and L_1 flows respectively.

Let us call $\nabla_S I(x) = \nabla_{u,v} I(x)$ for $x \in S$. We then simply take

$$\phi(|\nabla_S(I(x))|) = \frac{1}{\sqrt{1+\beta|\nabla_{u,v} I(x)|^2}} \tag{3.12}$$

and replace it in the algorithm described in section 3.3.

Remark Note that if we take $\beta \to 0$, we recover the expression 2.7 which approximates the Laplace Beltrami operator in order to perform isotropic smoothing:

$$\begin{aligned}
\lim_{\beta\to 0} \widehat{\Delta_g I(p)} &= \lim_{\beta\to 0} \frac{1}{A\sqrt{1+\beta|\nabla_{u,v} I(x)|^2}} \sum_{T_i \in A} \{ \frac{1}{\sqrt{1+\beta|\nabla_{u,v} I(x)|^2}} \cdot \\
&\quad [\cot\theta_i (I(p_i) - I(p)) + \cot\gamma_i (I(p_{i+1}) - I(p))] \} \\
&= \frac{1}{A} \sum_{T_i \in A} [\cot\theta_i (I(p_i) - I(p)) + \cot\gamma_i (I(p_{i+1}) - I(p))].
\end{aligned}$$

This last expression can be found in [20] just like the expression of $\widehat{\Delta_{u,v} I(p)}$ given below

$$\begin{aligned}
\widehat{\Delta_{u,v} I(p)} &= \sum_{T_i \in A} w_i (I(p_i) - I(p)) \\
\text{with} \qquad & w_i = \frac{\cot\theta_i + \cot\beta_i}{A_{T_i}},
\end{aligned}$$

which is obtained using common trigonometric identities. As before, θ_i, β_i are the two opposite angles of the edge (p, p_i) in the area A (see fig. 3.1). Also note that, when the surface is flat, we find that the Laplace-Beltrami estimation becomes a Laplacian discretization.

3.4.5 Implementation details

The implementation was done using the 3D visualization package *plouvis* developed at *Odyssée Lab.* We compute the value of I_p^n, the value of I on the vertex p of Σ at the n^{th} iteration based on the values of $I_{p_i}^{n-1}$. As usual, $p_i, .., p_{nb\_of\_neigh}$ denote the vertex neighboring p.

First, we compute the $\cot\theta_i$,$\cot\gamma$, and A_{T_i} for each node p only once. Then,

for each iteration n, we actualize the flow as follows:

$$\begin{aligned} I_p^n &= I_p^{n-1} + dt\widehat{\Delta_g I(p)}, \\ \widehat{\Delta_g I(p)} &= \frac{\phi(|\widehat{\nabla_{\mathcal{S}} I(p)}|)}{A} \sum_{T_i \in A} \{\phi(|\widehat{\nabla_{\mathcal{S}} I(p)}|) \cot\theta_i (I(p_i) - I(p)) \\ &\quad + \cot\gamma_i (I(p_{i+1}) - I(p))\}. \end{aligned}$$

3.4.6 Example

For this example (fig. 3.3) we use a triangulation from the Stanford Computer Graphics Laboratory's Data Base with the Japanese word for *peace* as the data function. We added a gaussian noise with $\sigma = 40$. The first stage of the algorithm takes about 2 minutes to be computed and the iterations a few more minutes (from 1 to 5 depending on the value of our parameter β). More examples on real images are presented in chapter 7.

For our experiments using the explicit framework, we used and developed (with the exception of lapack and vtk) the c++ libraries:

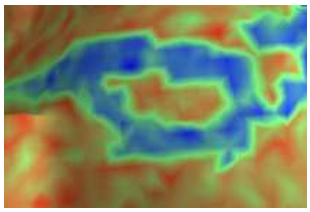
- **plouvis**, an Image processing and 3D rendering library developed by Robert Fourier at the Odyssée lab.
- **lapack**, fortran based linear algebra package.
- **vtk**, for the 3D visualization.

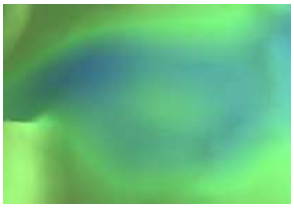
3.5 Summary and conclusions

In this chapter we have shown a novel numerical method for locally estimating operators for images defined on triangulated surfaces that can be used for a range of regularization techniques which involve operators of the form (3.8. This estimation has successfully been used by Adde et al. in [1, 2] to deal with the MEG/EEG inverse problem. For the case of isotropic smoothing, it yields to the same expression for the estimation of the Laplace-Beltrami operator obtained by Chung et al. in [20] and [41], which has been proved to be the *best estimation* for the Laplace-Beltrami operator in [106] in terms of convergence. In the second section we present a novel anisotropic regularization method based on the same technique.

Original noisy image

$\beta = 0$ (Isotropic diffusion)





$\beta = 0.01$

$\beta = 1$ (Anisotropic diffusion)

Figure 3.3: Note the stair-casing effect on the anisotropic diffusion regularization result and the blurry image obtained with the isotropic one.

In the last section, we reformulate the Beltrami Framework for the case of images defined on explicit surfaces, and propose an implementation. This regularization technique provides a way to overcome the over-smoothing of the isotropic regularization and the stair-casing effects anisotropic regularization by tunning between the two approaches via the parameter β which encodes the ratio between the data and spatial units. This parameter controls the edge-preserving characteristic of the flow. The Beltrami flow is shown to act as the linear diffusion (L_2-norm) in the limit $\beta \to 0$ and to the strongly edge preserving diffusion (L_1-norm for the scalar field) in the limit $\beta \to \infty$ (up to time scaling). We have also illustrated the utility of this approach showing synthetic examples. Examples on real images are shown in chapter 7.

Chapter 4

Regularization on implicit surfaces

Since we have already introduced the implicit framework and the isotropic diffusion using this method in the first chapter, we begin with the anisotropic diffusion in the simplest case of the Perona and Malik regularization. In the second section we generalize the Beltrami flow regularization for images defines on implicit surfaces[1]. We then present some results on synthetic images.

4.1 Anisotropic diffusion

We start by replacing the expressions on the P&M known PDE by the surface versions explained in chapter 2, and using the properties given in subsection 2.3.2, we find that for the surface version of the anisotropic diffusion we have

$$\frac{\partial u}{\partial t} = \nabla \cdot \frac{\nabla u}{\|\nabla u\|}.$$

The gradient descent is

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla_{\mathcal{S}} \cdot \frac{\nabla_{\mathcal{S}} u}{\|\nabla_{\mathcal{S}} u\|} \\ &= \frac{1}{\|\nabla \psi\|} \nabla \cdot \left(\frac{P_{\nabla \psi} \nabla u}{\|P_{\nabla \psi} \nabla u\|} \|\nabla \psi\| \right). \end{aligned}$$

[1] Work published in [83, 81, 82]

This is quoted from [9, 10, 8, 58]. We implemented this method, and we propose an alternative numerical scheme more stable that those given in the cited works that can be found in appendix (10.4) .

4.2 Beltrami flow

In the previous chapter, we formulated the Beltrami flow for images defined over non flat surfaces given in an explicit parametric form. In the following section, we focus on the case where the surface is given in an implicit form. In the implicit setup we are given an implicit surface $\Psi(X^1, X^2, X^3) = 0$ and an implicit scalar image on the surface $U(X^1, X^2, X^3)$. We postpone the treatment of this case to the next section and study first the easier case of a one-dimensional manifold i.e. a curve on which a scalar field is given.

Although the derivation of the flow equations for scalar data defined over a curve is very similar to the surface case, it is easier to get a clear image of the situation for curves. We therefore describe first the derivations in details in this simpler situation. The derivation in the first subsection is purely geometric, and once the flow equation is established, it can be easily generalized to higher dimensional manifolds, e.g. surfaces. An alternative derivation can be found in [83].

4.2.1 Scalar field defined on a curve: Geometric derivation

The curve is given in the x-y plane and the data is pictured as the height in the z direction. The data can be seen in that framework as a curve in $\mathbb{R}^3$. Note that this curve lies on the cylinder-like surface defined by $\Psi(x, y) = 0$. We want to produce a curvature flow of this data such that it is confined to the surface, which means that it is defined on the base curve. The projected curvature on the surface is the geodesic curvature. Note however that the fact that the data curve is on the cylinder-like surface, generated by the base curve, is not enough in general to ensure that the data curve can be represented as a function on the base curve since the evolution is not done in the z direction only (compare with [12]). For a general flow, it may happen that two or even more points of the data curve share the same x-y values at different times. In order to avoid such problems and to get a flow that respects discontinuities, we confine the flow by projecting the geodesic curvature on the z direction. This procedure is analogue to the Beltrami flow

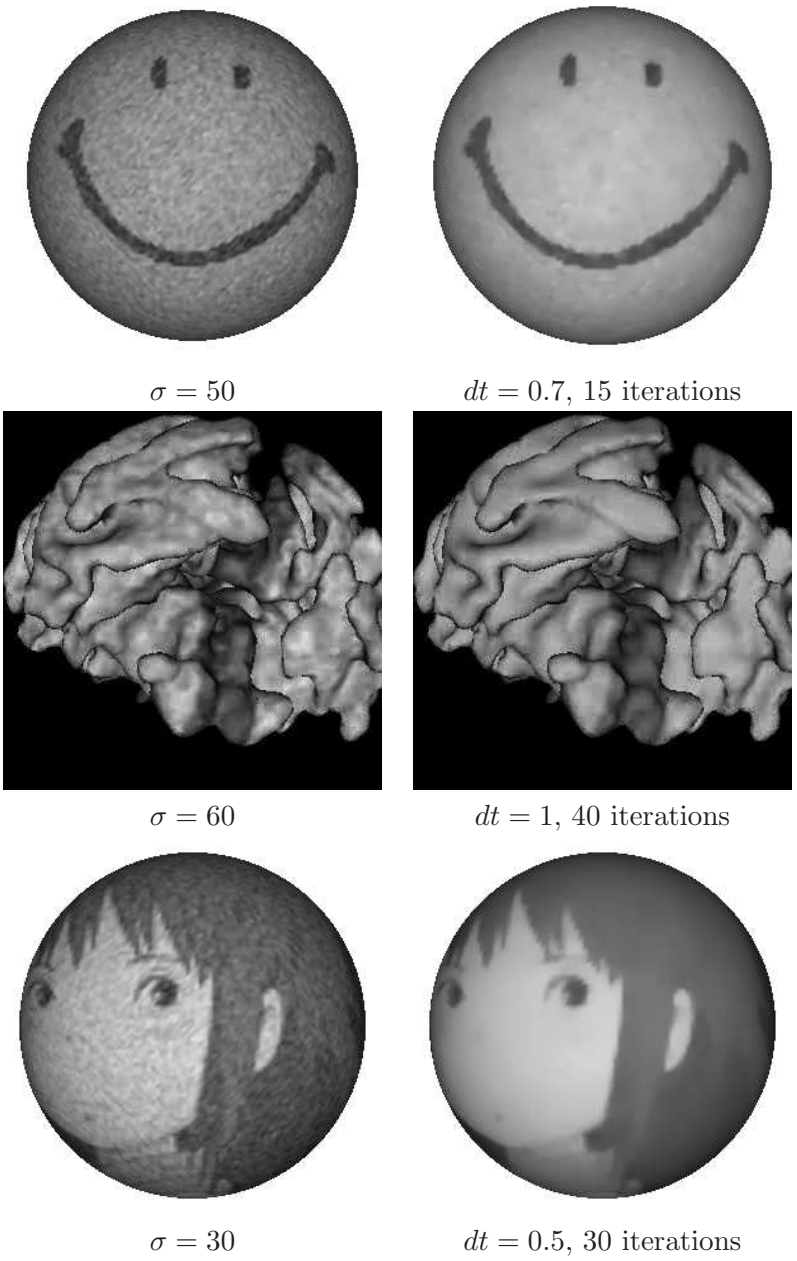


Figure 4.1: Examples of anisotropic diffusion of scalar images over implicit surfaces

which is a projection of the mean curvature flow on the intensity direction.

Formulating this geometric understanding, we represent the curve as the intersection of two surfaces:

$$\begin{aligned} \Psi(x,y) &= 0, \\ \Phi(x,y,z) &= z - \beta U(x,y) = 0. \end{aligned} \tag{4.1}$$

The surface defined by Ψ is a cylinder-like surface defined by the base curve, which lies in the x-y plane. The second surface is the graph of the function U which is defined over the x-y plane (and by restriction over the base curve). The data function is modified along the flow in the z direction only. Thus, it remains a function along the flow (see Fig. 4.2). We denote by D the 3D gradient $D = (\partial_x, \partial_y, \partial_z)^t$, and by ∇ the 2D gradient $\nabla = (\partial_x, \partial_y)^t$. We define the tangent vector to the curve by

$$T = \frac{D\Psi \times D(z - \beta U)}{||D\Psi \times D(z - \beta U)||} = [T_1, T_2, T_3]^t.$$

The curvature is then given by the second derivative of the curve.

$$\begin{aligned} kN &= \partial_s T = x_s \partial_x T + y_s \partial_y T + z_s \partial_z T = T^t DT \\ &= (T \cdot DT_1, T \cdot DT_2, T \cdot DT_3)^t. \end{aligned} \tag{4.2}$$

The projection of this vector on the surface defined by $\Psi(x,y) = 0$ is obtained by applying the projection operator $P_\eta kN$. The normal to the surface is

$$\eta = \frac{D\Psi}{||D\Psi||} .$$

We will write, abusing of the notation, $P_{D\Psi}$ and $P_{\nabla\Psi}$ for the 3D and 2D projections respectively. Keeping only the z component in (4.2) we get

$$U_t = \left(P_{D\Psi} kN\right)_z , \tag{4.3}$$

which is the non-flat analogue of (5.14).

Let us calculate it explicitly. We have the following:

$$\begin{aligned} (D\Psi \times D(z - \beta U))^t &= (\Psi_y, -\Psi_x, \beta(\Psi_y U_x - \Psi_x U_y)) \\ D(z - \beta U)^t &= (-\beta U_x, -\beta U_y, 1) \\ D\Psi^t &= (\Psi_x, \Psi_y, 0). \end{aligned} \tag{4.4}$$

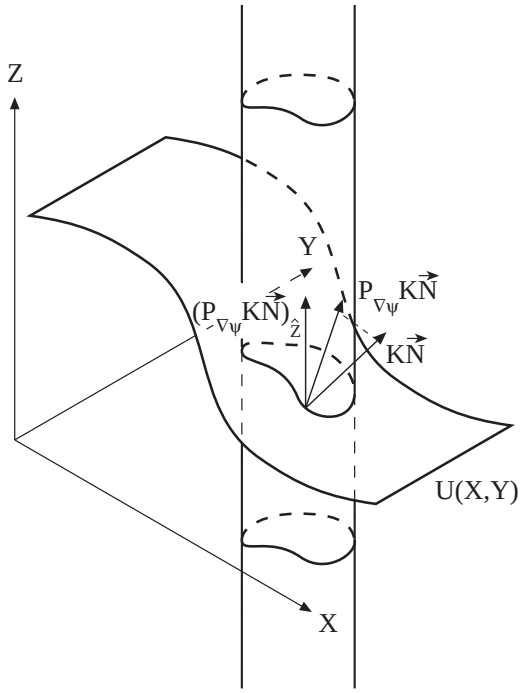


Figure 4.2: The data curve as the intersection of the cylinder-like surface induced by the base planar curve, extended to $\mathbb{R}^3$, and the graph of the function U.

Note that T does not depend on z. The projection operator is

$$\begin{aligned} P_{D\Psi} &= I - \frac{D\Psi D\Psi^t}{||D\Psi||^2} \\ &= \frac{1}{||D\Psi||^2}\begin{pmatrix} \Psi_y^2 & -\Psi_x\Psi_y & 0 \\ -\Psi_x\Psi_y & \Psi_x^2 & 0 \\ 0 & 0 & ||D\Psi||^2 \end{pmatrix} \end{aligned}$$

The equation of motion now reads

$$U_t = \left(P_{D\Psi} \begin{pmatrix} T \cdot DT_1 \\ T \cdot DT_2 \\ T \cdot DT_3 \end{pmatrix} \right)_z = T \cdot DT_3.$$

Since T does not depend on z, only the x and y components enter the calculation:

$$U_t = T_1 \partial_x T_3 + T_2 \partial_y T_3. \tag{4.5}$$

The gradient notation ∇ and all other vector notations are from now on two-dimensional. We also denote the perpendicular gradient by

$$\nabla^{\perp}\Psi = (\Psi_y, -\Psi_x)^t.$$

Notice also that in the x-y plane we have the identities

$$\begin{aligned} ||\nabla\Psi||^2 P_{\nabla\Psi}\nabla U &= (\Psi_y U_x - \Psi_x U_y)\nabla^{\perp}\Psi \\ ||\nabla\Psi||^2 ||P_{\nabla\Psi}\nabla U||^2 &= (\Psi_y U_x - \Psi_x U_y)^2. \end{aligned} \tag{4.6}$$

Define

$$\begin{aligned} g &= ||\nabla\Psi||^2 + \beta^2(\Psi_y U_x - \Psi_x U_y)^2 \\ &= ||\nabla\Psi||^2 \left(1 + \beta^2 ||P_{\nabla\Psi}\nabla U||^2\right). \end{aligned} \tag{4.7}$$

We can finally write (4.5) as follows

$$\begin{aligned} U_t &= \frac{\nabla^{\perp}\Psi}{\sqrt{g}} \cdot \nabla \left(\frac{\beta(\Psi_y U_x - \Psi_x U_y)}{\sqrt{g}} \right) \\ &= \frac{1}{\sqrt{g}} \mathrm{Div} \left(\frac{\beta ||\nabla\Psi||^2 P_{\nabla\Psi}\nabla U}{\sqrt{g}} \right) \end{aligned} \tag{4.8}$$

where we used the identities (4.6) and the fact that $\mathrm{Div}\nabla^{\perp}\Psi = \partial_x(\Psi_y) + \partial_y(-\Psi_x) = 0$.

In conclusion, this last equation allows us to denoise the scalar signal U lying on the implicitly defined curve $\Psi(x, y) = 0$. This Beltrami flow defined on implicit curve can also be generalized to the case of scalar field defined on implicit surface. The form of the flow is identical. The only difference is the dimension of the level set functions. For an n-dimensional manifold we use an $n + 2$-dimensional level sets whose intersection gives the data section of interest. An alternative derivation that coincides with this result is given in [83].

4.2.2 The L_1 and L_2 limits

We show in this subsection that (4.8) interpolates between the non-flat L_1 flow (5.8) and the L_2 flow (5.5). Note that

$$\lim_{\beta\to 0} g = ||\nabla\Psi||^2$$

$$\lim_{\beta \to \infty} g \quad = \quad \beta^2 ||\nabla \Psi||^2 ||P_{\nabla \Psi} \nabla U||^2. \tag{4.9}$$

It is easily seen now that when $\beta \to 0, \quad t \to \infty$ (maintaining $\tau = t\beta$ finite), the implicit Beltrami flow defined in (4.8) tends to

$$u_\tau = \frac{1}{||\nabla \Psi||} \text{Div} \left(||\nabla \Psi|| P_{\nabla \Psi} \nabla U \right),$$

which is the non-flat L_2 flow. This fact was derived in a different manner by Bertalmio and al [10].

The other case of interest, when $\beta \to \infty, \quad t \to \infty$ (this time maintaining $\tau = t/\beta$ finite), gives another flow as a result:

$$u_\tau = \frac{1}{||\nabla \Psi|| ||P_{\nabla \Psi} \nabla U||} \text{Div} \left(\frac{||\nabla \Psi|| P_{\nabla \Psi} \nabla U}{||P_{\nabla \Psi} \nabla U||} \right)$$

which is the non-flat L_1 flow used in [10].

In one limit, we find the L_2 norm which over-smooths the image while in the other limit, we find the L_1 flow with the notorious stair-casing effect. Choosing for β intermediate values brings more degree of freedom in the regularization of noisy data defined on surfaces. It opens new perspectives with the advantage of smoothing anisotropically the image and conserving the edges while avoiding the disadvantages of the total variation type of flow. This is well illustrated with various synthetic and real images in the next subsections that present our numerical results (figures 4.4, 4.5).

4.2.3 Implementing the regularization of scalar fields on surfaces

We compute the value of $u^n_{i,j,k}$, the value of u in the pixel (i, j, k) at the n^{th} iteration, based on the values of u^{n-1} at the neighboring pixels. First, we compute the vector $\vec{N} \simeq \nabla \Psi$ (this is needed only once) using central differences. Then, for each iteration n, we visit all pixels to compute:

- The gradient, its projection, and g.
- The divergence.
- The actualization of u.

For the gradient $\vec{v}^n_{i,j,k}$ we used backward differences

$$\vec{v}^n_{i,j,k} = \nabla_+ u^n_{i,j,k} = \begin{pmatrix} u^n_{i+1,j,k} - u^n_{i,j,k} \\ u^n_{i,j+1,k} - u^n_{i,j,k} \\ u^n_{i,j,k+1} - u^n_{i,j,k} \\ , \end{pmatrix}$$

its projection on the surface

$$\left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k} = \vec{v}^n_{i,j,k} - \frac{\sum_{m=1}^{3} \vec{N}_{i,j,k}[m] \cdot \vec{v}^n_{i,j,k}[m]}{\left\|\vec{N}_{i,j,k}\right\|^2} \vec{N}_{i,j,k},$$

and g,

$$g^n_{i,j,k} = ||\vec{N}_{i,j,k}||^2 \left(1 + \beta^2 || \left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k} ||^2\right),$$

where square brackets represent the component of the vector. To compute the divergence, we use backward differences,

$$\begin{aligned} \nabla_- \cdot \vec{w}_{i,j,k} \quad = \quad & \vec{w}_{i,j,k}[1] - \vec{w}_{i-1,j,k}[1] + \\ & \vec{w}_{i,j,k}[2] - \vec{w}_{i,j-1,k}[2] + \\ & \vec{w}_{i,j,k}[3] - \vec{w}_{i,j,k-1}[3] \end{aligned}$$

We switch forward differences and backward differences to avoid numerical problems. Finally, the resulting flow implementation is:

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t \frac{1}{\sqrt{g^n_{i,j,k}}} \nabla_- \cdot \left(\frac{\beta \left\|\vec{N}_{i,j,k}\right\| \left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k}}{\sqrt{g^n_{i,j,k}}} \right).$$

We use a time step $\frac{dt}{\beta}$, adjusted accordingly as in the previous subsection.

For our experiments we used and developed (for yar++) the C++ libraries:

- **yar++**, an Image processing C++ library made by Gerardo Hermosillo at the *Odyssée lab.*
- The Visualization Tool Kit (VTK), for the 3D rendering.

For the visualization, we used the marching cubes algorithm [55] to obtain a triangulation from our implicit representation of the surface, then to draw the data on this surface.

For the images defined on curves, the experiments were carried on *matlab*.

4.2.4 Examples

We present in this subsection some figures that illustrate the regularization of noisy data on various implicitly defined curves and surfaces. The results are given with various values of the parameter β. Note how the regularization of the data is done isotropically or anisotropically depending on the value of this parameter.

In the first set of images (fig. 4.3), the red curve is the original function, the green curve is obtained by adding noise, and the blue curve is obtained after regularization. For images (a) and (b), we added a Gaussian noise with $\sigma = 15$.

Regarding the CPU time, for the tori (12500 pixels), it took less than 2 seconds for 20 iterations. Note that we used a color map from red to blue in order to better see how the the discontinuities are treated. Regarding the noise level, we have added a Gaussian noise with ($\sigma = 40$) to the grey level images which scalar range is (0,256).

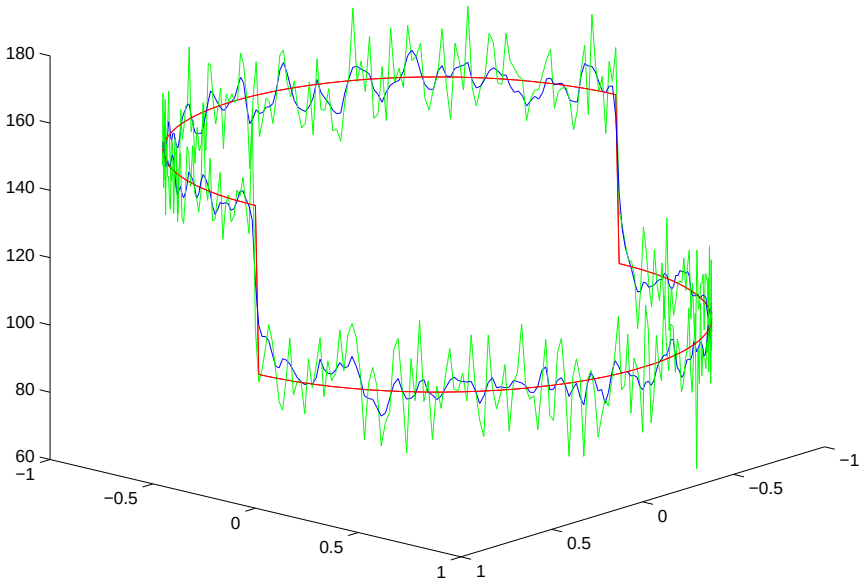
For the Ella image, illustrated in fig. 4.5 and defined in a solid that looks like a quadratic, the image is much bigger (2744000 pixels). The time to compute 20 iterations was almost 3.5 minutes in a 386 sun, 260 MB in RAM.

In strongly noised images such as these anisotropic regularization treat some noise as part of the image discontinuities (the points under the left eye, for example). The intermediate regularization with $\beta = 0.1$ gives a better result.

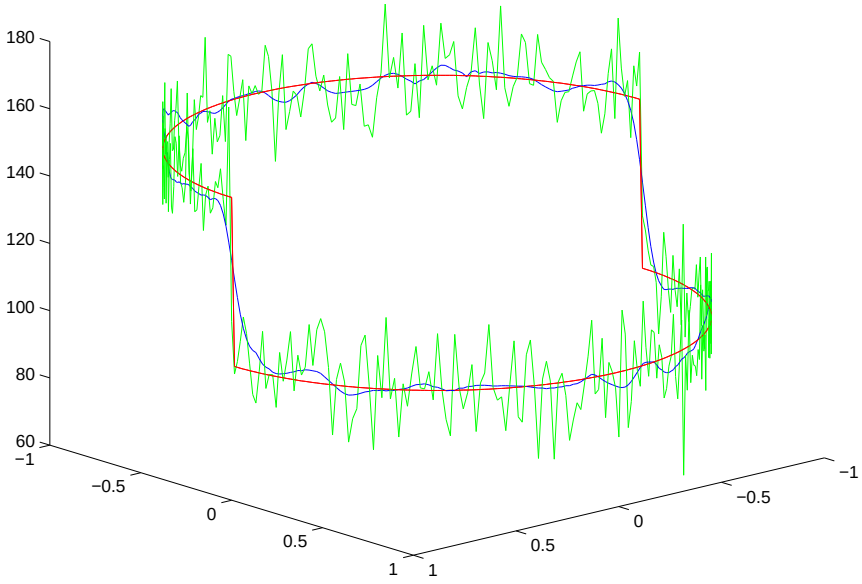
4.3 Summary and conclusions

At this point we have reformulated the Beltrami framework for the intrinsic - parametric method, the implicit - level set method, and the explicit - triangulated form of a surface.

We used the geometrical understanding of the flat Beltrami flow and generalized it to a denoising flow over implicitly defined curves and surfaces. It is shown that the resulting flows depend on β which encodes the ratio between the data and spatial units. This parameter controls the edge-preserving characteristic of the flow. The Beltrami flow is shown to act as the linear

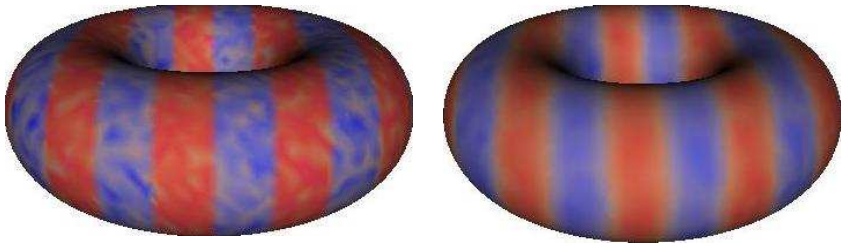


(a) $\beta = 0.1$

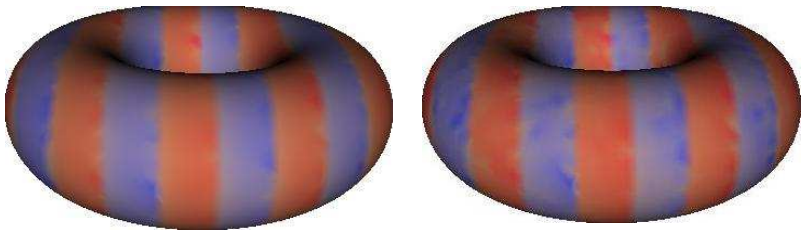


(b) $\beta = 0$ (Isotropic diffusion)

Figure 4.3: Beltrami flow regularization for scalar data defined on curves. Results for different values of β



Original noisy image | Isotropic diffusion



$\beta = 0.1$ | L_1 anisotropic diffusion

Figure 4.4: Beltrami flow regularization of synthetic scalar data defined on the torus, for different values of β

diffusion (L_2-norm, isotropic regularization) in the limit $\beta \to 0$ and to the strongly edge preserving diffusion (L_1-norm) in the limit $\beta \to \infty$ (up to time scaling).

The implementation scheme of this flow is presented and various experimental results obtained (subsection 4.2.4) on a set of various synthetic images (examples on real images are shown in chapter 7) illustrate the performances of the approach as well as the differences with the harmonic map flows.

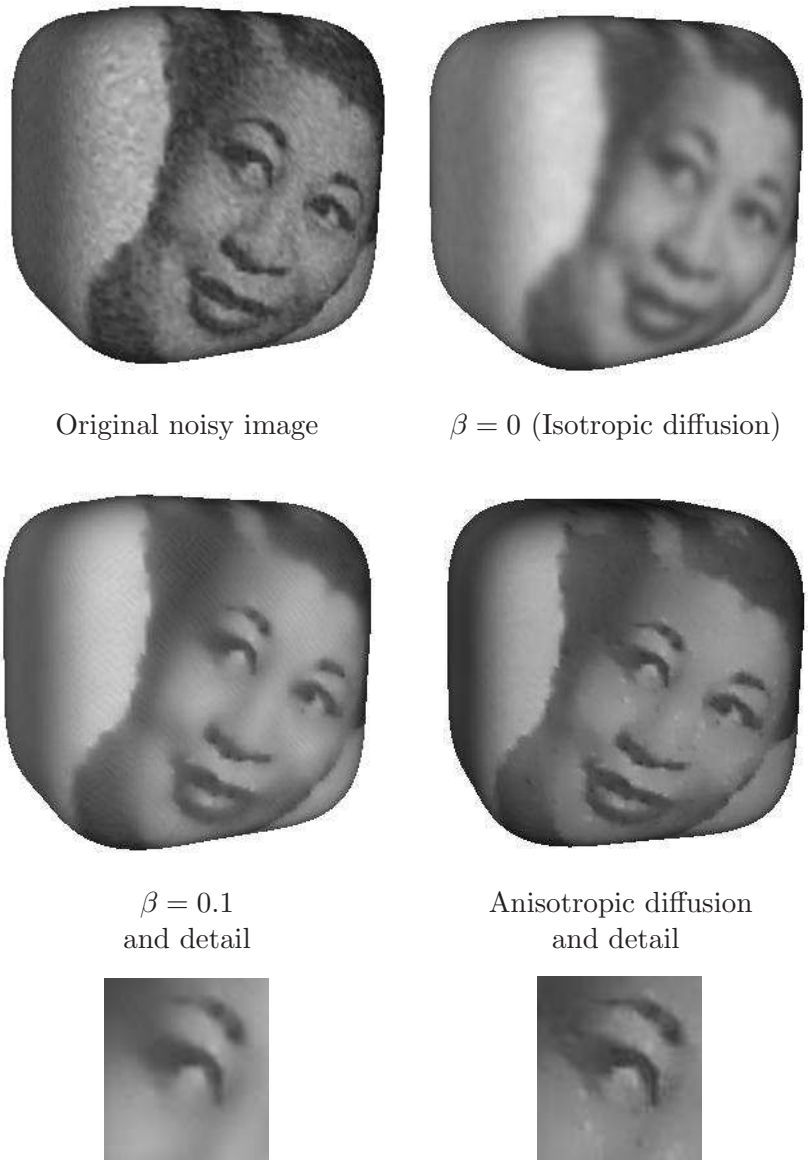


Figure 4.5: Beltrami flow regularization with different values of β for an Ella Fitzgerald photograph on a quadratic surface. Note the excellent result for $\beta = 0.1$

Chapter 5

Comparison of methods and relationships

In this chapter we first clarify the relationship between both the explicit and implicit approaches. In the second section, we discuss the passage between the explicit and implicit representation and vice versa, and then examine the respective advantages and drawbacks of the two principal frameworks, in particular for an a application of analysis[1].

5.1 From explicit to implicit via the Beltrami flow

There are two ways to write the geometric functional which is called harmonic energy or Polyakov action. Both functionals are defined over the space of embedding maps of Riemannian manifolds. We focus our attention to the case of a gray-value image defined over a two- (or one-) dimensional manifold i.e. a surface (or a curve). The first way is to represent the surface implicitly i.e. as a zero section of a distance function, defined over $\mathbb{R}^3$. The other way is to choose a local coordinate system and define the surface intrinsically in a parametric way. These ideas are developed in more details in the next two subsections.

[1] Work published in [105]

5.1.1 Implicit formulation

Suppose that we have a scalar data U defined on a known surface Σ, that is we have a mapping $\tilde{U} : \Sigma \to \mathbb{R}^1$ where Σ is a known surface, given by its implicit representation

$$\Psi(X^1, X^2, X^3) = 0. \tag{5.1}$$

We extend $\tilde{U}$ to $U : \mathbb{R}^3 \to \mathbb{R}$ such that $\tilde{U}$ is the restriction of U to Σ. The *Harmonic maps* approach [10, 58, 12, 14] looks for the minimizer of the following energy

$$S_{imp}[U] = \int_{\mathbb{R}^3} || \nabla_\Sigma U ||^2 \, d\Sigma \tag{5.2}$$

where $d\Sigma$ is the surface element $d\Sigma = \delta(\Psi)||\nabla\Psi||dX^1dX^2dX^3$. We denote here by δ the Dirac function and $\nabla_\Sigma U$ denotes the gradient intrinsic to the surface Σ i.e. the projection of the 3D Cartesian gradient of U on $T\Sigma$ – the tangent space to Σ. Clearly, $\nabla\Psi$ is normal to its zero level set, which is Σ. It follows then that $\nabla_\Sigma U = P_\eta \nabla U$ where $P_{\vec{\eta}}$ is the orthogonal projection operator on the surface Σ. Here ∇U is the 3D Cartesian gradient of U:

$$\nabla U = \left(\frac{\partial U}{\partial X^1}, \frac{\partial U}{\partial X^2}, \frac{\partial U}{\partial X^3} \right)^t ,$$

and η denotes the unit normal vector to the surface Σ with

$$\eta = \frac{\nabla\Psi}{||\nabla\Psi||}.$$

We can rewrite now the harmonic energy as follows:

$$\begin{aligned} & S_{imp}[U] = \int_{\mathbb{R}^3} || P_\eta \nabla U ||^2 \, d\Sigma \\ = & \int_{\mathbb{R}^3} || P_\eta \nabla U ||^2 \, \delta(\Psi) \, || \nabla\Psi || \, dX^1dX^2dX^3 \ . \end{aligned}$$

All the expressions above are considered in the sense of distributions. For a given surface with a unit normal vector η, the orthogonal projection operator $P_{\vec{\eta}}$ on that surface is given by

$$P_\eta = I - \eta\eta^t \ , \tag{5.3}$$

where I is the 3×3 Identity matrix and η^t denotes the transposed of the

vector η. The functional can rewritten as follows:

$$S_{imp}[U] = \int_{\mathbb{R}^3} \nabla U^t (I - \eta\eta^t) \nabla U d\Sigma \ . \tag{5.4}$$

This functional is the generalization of the L_2 norm from flat to non-flat manifolds. The modified gradient descent equation reads:

$$U_t = \frac{1}{||\nabla\Psi||} \mathrm{Div} \left(||\nabla\Psi|| P_\eta \nabla U \right) \ , \tag{5.5}$$

where Div stands for the 3D Divergence operator, and ∇ is in $\mathbb{R}^3$ as well. The restriction of U to the surface Σ is the regularized image. This equation can be generalized via the Φ formulation (see [52, 77, 25]) to

$$S_{imp}[U] = \int_{\mathbb{R}^3} \Phi \left(||\nabla_\Sigma U|| \right) d\Sigma. \tag{5.6}$$

In particular, for the generalized L_1 norm

$$S_{imp}[U] = \int_{\mathbb{R}^3} ||\nabla_\Sigma U|| d\Sigma \tag{5.7}$$

we obtain the following minimization flow

$$U_t = \frac{1}{||\nabla\Psi|| ||P_\eta \nabla U||} \mathrm{Div} \left(\frac{||\nabla\Psi|| P_\eta \nabla U}{||P_\eta \nabla U||} \right) \ . \tag{5.8}$$

For the interested reader, we refer to [10, 58, 12, 14] for more details and results on this implicit formulation and its various applications.

5.1.2 Intrinsic formulation

Suppose we have a 2-dimensional manifold Σ with local coordinates σ^1, σ^2 embedded in an 3-dimensional manifold M with coordinates X^1, X^2, X^3, the embedding map $X : \Sigma \to M$ is given explicitly by the 3 functions of 2 variables

$$X : (\sigma^1, \sigma^2) \longrightarrow (X^1(\sigma^1, \sigma^2), X^2(\sigma^1, \sigma^2), X^3(\sigma^1, \sigma^2)) \ .$$

Note that the X^is are the solution of (5.1). A gray-value image is represented, in this framework, as the embedding $(X^1 = \sigma^1, X^2 = \sigma^2, X^3 = \beta U(\sigma^1, \sigma^2))$ (see Fig. 1).

Denote by (Σ, g) the image manifold and its metric and by (M, h) the space-feature manifold and its metric, then the map $X : \Sigma \to M$ has the following weight :

$$S_{int}[U, g] = \int d\sigma^1 d\sigma^2 \sqrt{g} g^{\mu\nu} \partial_\mu X^i \partial_\nu X^j h_{ij}(X) \tag{5.9}$$

where g is the determinant of the image metric, $g^{\mu\nu}$ is the inverse of the image metric, the range of indices is $\mu, \nu = 1, .., 2$ and $i, j = 1, ..3$ and h_{ij} is the metric of the embedding space. We use the summation convention: indices that appear twice are being summed over. This functional was first proposed by Polyakov in the context of high energy physics [69] .

Let us formulate the Polyakov action in the matrix form: (Σ, G) is the image manifold and its metric as before. Similarly, (M, H) is the spatial-feature manifold and its metric (it is simply $\mathbb{R}$ with the usual Euclidean distance for gray-value images). Define

$$A^{ij} = (\partial_\Sigma X^i)^t G^{-1} \partial_\Sigma X^j \ ,$$

where $\partial_\Sigma X^i = (\partial_{\sigma^1} X^i, \partial_{\sigma^2} X^i)^t$.

The map $X : \Sigma \to M$ has a weight

$$S[X^i, G, H] = \int d\sigma^1 d\sigma^2 \sqrt{g} \mathrm{Tr}(AH),$$

where $g = \det(G)$.

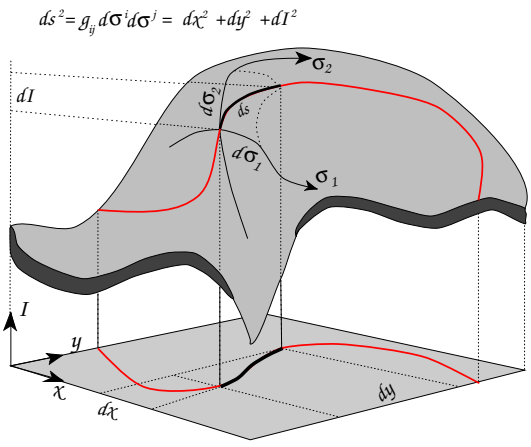


Figure 5.1: The graph of the intensity map as a surface embedded in $\mathbb{R}^3$

The gradient descent equations, with respect to the embedding functions, result in

$$X_t^i = frac1\sqrt{g}\partial_\mu(\sqrt{g}g^{\mu\nu}\partial_\nu X^i) + \Gamma^i_{jk}(\partial_\mu X^j)(\partial_\nu X^k)g^{\mu\nu}. \tag{5.10}$$

In our case X^1 and X^2 are fixed and we only change $X^3 = \beta U$. The embedding space is Euclidean and therefore the Levi-Civita Γ^3_{jk} coefficients are zero.

The flow, in matrix form, is therefore

$$U_t = -\frac{1}{2\sqrt{g}}\frac{\delta S}{\delta U} = \underbrace{\frac{1}{\sqrt{g}}\mathrm{div}\left(\sqrt{g}G^{-1}\nabla U\right)}_{\Delta_g U}. \tag{5.11}$$

The extension for non-Euclidean embedding space is treated in [85, 86, 47, 88, 89]. Note that we did not specify the metric yet. If we choose for a metric the first fundamental form of the base and non-flat manifold, then this flow is the analog of (5.5). Both equations describe a **linear operator** that acts on the image function U. One major difference between the Beltrami framework and the harmonic map formulation is the way one chooses the metric. In the harmonic map approach it is treated as data which is given beforehand. The metric then is **constant in time**. In the Beltrami framework **the metric is a dynamic variable to be found by minimizing the functional**. Carrying out the calculation, we find the following equation (see [85] for the derivation):

$$\frac{1}{\sqrt{g}}\frac{\delta S}{\delta g_{\mu\nu}} = -\frac{1}{2}g_{\mu\nu}g^{\lambda\rho}\partial_\lambda X^i\partial_\rho X^j h_{ij} + \partial_\mu X^i\partial_\nu X^j h_{ij} = 0. \tag{5.12}$$

Albeit the complicated appearance of the minimization equation, **it can be solved analytically**. The result is simply

$$g_{\mu\nu} = \partial_\mu X^i\partial_\nu X^j h_{ij}\ .$$

This equation has a clear geometric meaning: it is the induced metric of the section! In the simplest situation of flat domain with a scalar field i.e. the intensity, the metric is find to be [86]:

$$(g_{\mu\nu}) = \begin{pmatrix} 1+U_x^2 & U_xU_y \\ U_xU_y & 1+U_y^2 \end{pmatrix}. \tag{5.13}$$

where U denotes the intensity level. In the Beltrami framework, the manifold of interest is taken to be **the data surface** i.e. the graph of the intensity function. In other words we are interested in the structure of the section of the fiber bundle and not in the base manifold. It means that the metric itself depends on the data. This is the way non-linearities are introduced in the flow.

Equation (5.11), with the induced metric (5.13), has a clear geometric meaning. It is the projection of the mean curvature vector in the $X^3 = U$ direction:

$$U_t = KN_{\hat{U}} \tag{5.14}$$

where K is the mean curvature magnitude. N is the normal to the surface and $\hat{U}$ is a unit vector in the $X^3 = U$ direction (see Fig. 2). This choice of the induced metric gives a simple geometric meaning to the functional as well. It becomes simply the volume of the section (area or length for the two- and one- dimensional cases). It is important for the study below to have the explicit form of the functionals for flat domain in the scalar and vectorial cases:

$$S_s[U] \quad = \quad \int dxdy\sqrt{g} = \int dxdy\sqrt{1+\beta^2||\nabla U||^2}$$

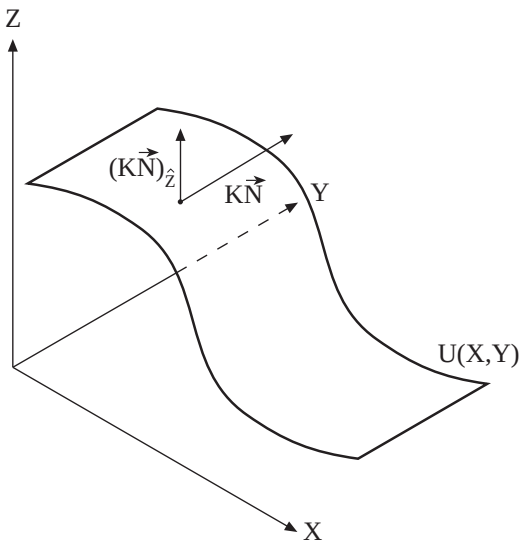


Figure 5.2: The Beltrami flow velocity as a projection of the mean curvature on the data = z axis

$$S_v[U^i] = \int dxdy \sqrt{1+\beta^2 \sum_i ||\nabla U^i||^2 + \beta^4 \sum_{i,j} ||\nabla U^i \times \nabla U^j||^2} \quad (5.15)$$

where S_s corresponds to the functional for the scalar case, S_v for the vectorial case, and β is the ratio between the distances taken in the spatial and intensity directions.

We will use this geometric understanding in order to derive the Beltrami flow on non-flat surfaces from pure geometric consideration. We also use this simplified functional in our quest for implicit formulation of the Beltrami flow on non-flat manifolds. But let us explain first the implicit-intrinsic correspondence.

5.1.3 The implicit-explicit correspondence

We have shown two ways of writing the geometric functional, which is called either harmonic energy or the Beltrami functional, and weights embedding maps of Riemannian manifolds.

Let us work out the case of a painted surface. The mapping is $U : \Sigma \to \mathbb{R}^1$ where Σ is a known surface with metric G (first fundamental form).

One way is to represent the surface implicitly i.e. as a zero section of an Osher-Sethian function, defined over $\mathbb{R}^3$. The other way is to choose a local coordinate system and define the surface intrinsically in a parametric way.

The implicit method consist of the equation

$$\Psi(X^1, X^2, X^3) = 0. \quad (5.16)$$

This equation is an implicit description of a surface. The functional is written in that case as

$$S_{imp}[U] = \int_{\mathbb{R}^3} \nabla U^t (\mathrm{I} - \eta\eta^t) \nabla U \delta(\Psi) ||\nabla \Psi|| dX^1 dX^2 dX^3 \quad (5.17)$$

where

$$\eta = \frac{\nabla \Psi}{||\nabla \Psi||}$$

and the gradient is

$$\nabla U = \left(\frac{\partial U}{\partial X^1}, \frac{\partial U}{\partial X^2}, \frac{\partial U}{\partial X^3} \right)^t$$

The intrinsic way relies on the choice of (σ^1, σ^2) as local coordinates on (possibly part of) the manifold. The surface is then given by the functions

$$\left(X^1(\sigma^1,\sigma^2), X^2(\sigma^1,\sigma^2), X^3(\sigma^1,\sigma^2)\right). \tag{5.18}$$

Note that these functions are the solutions of (5.16).

The functional is defined as

$$S_{int}[U] = \int \partial_\Sigma U^t G^{-1} \partial_\Sigma U \sqrt{g} d\sigma^1 d\sigma^2 \tag{5.19}$$

where $g = \det(G)$ and

$$\partial_\Sigma U = \left(\frac{\partial U}{\partial \sigma^1}, \frac{\partial U}{\partial \sigma^2}\right)^t.$$

We show the equivalence between these two ways of writing by deriving the intrinsic formulation from the implicit one. To that purpose, we perform the integration over the delta function and express all the three dimensional objects intrinsically. Explicitly we prove the following technical lemma:

LEMMA:

1. $$G = J^t J$$

   where J is the Jacobian – a 3×2 matrix whose elements are

   $$\frac{\partial X^i}{\partial \sigma^\mu} \qquad i = 1,2,3 \quad \mu = 1,2$$

   That is

   $$J = \begin{pmatrix} \frac{\partial X^1}{\partial \sigma^1} & \frac{\partial X^1}{\partial \sigma^2} \\ \frac{\partial X^2}{\partial \sigma^1} & \frac{\partial X^2}{\partial \sigma^2} \\ \frac{\partial X^3}{\partial \sigma^1} & \frac{\partial X^3}{\partial \sigma^2} \end{pmatrix}$$

2. $$\nabla U^t J = \partial_\Sigma U^t$$

3. $$||\nabla \Psi|| = \sqrt{g}$$

   where $g = \det G$.

4.

$$J^t \cdot \nabla\Psi = 0$$

Proof:

1. For an intrinsic surface the induced metric is

$$G_{\mu\nu} = e_\mu \cdot e_\nu$$

where e_1, e_2 are the tangent vectors to the surface. In our case

$$e_\mu = \begin{pmatrix} \frac{\partial X^1}{\partial \sigma^\mu} \\ \frac{\partial X^2}{\partial \sigma^\mu} \\ \frac{\partial X^3}{\partial \sigma^\mu} \end{pmatrix}$$

The Jacobian is simply written as $J = (e_1, e_2)$ and we find

$$(J^t J)_{\mu\nu} = e_\mu \cdot e_\nu = G_{\mu\nu}$$

2.

$$\partial_{\sigma^\mu} U = U_{x^1} \frac{\partial X^1}{\partial \sigma^\mu} + U_{x^2} \frac{\partial X^2}{\partial \sigma^\mu} + U_{x^3} \frac{\partial X^3}{\partial \sigma^\mu} = \nabla U^t e_\mu \qquad \mu = 1, 2$$

This is written as

$$\partial_\Sigma U^t = \nabla U^t J.$$

3. The normal to the surface $\nabla\Psi$ is perpendicular to the two tangent vectors e_μ. The components of the vector product of two vectors V_1 and V_2 is given by $(V_1 \times V_2)_i = \epsilon_{ijk} V_1^j V_2^k$ where $\epsilon_{ijk} = (-1)^p$ and p is the number of permutations needed to bring (ijk) to (123). We can now compute

$$||\nabla\Psi||^2 = ||e_1 \times e_2||^2 = \epsilon_{ijk} e_1^j e_2^k \epsilon_{ilm} e_1^l e_2^m \ .$$

We use the identity $\epsilon_{ijk}\epsilon_{ilm} = \delta_{jl}\delta_{km} - \delta_{jm}\delta_{kl}$ which can be verified directly, and find

$$\begin{aligned} ||\nabla\Psi||^2 &= (\delta_{jl}\delta_{km} - \delta_{jm}\delta kl)\, e_1^j e_2^k e_1^l e_2^m \\ &= e_1 \cdot e_1 e_2 \cdot e_2 - (e_1 \cdot e_2)^2 \\ &= G_{11}G_{22} - G_{12}^2 = \det G \end{aligned}$$

4.

$$J^t \cdot \nabla\Psi = (e_1, e_2)^t \nabla\Psi$$

but

$$e_i^t \cdot \nabla\Psi = \partial_{\sigma^i}\Psi = 0 \quad i = 1, 2,$$

Since Ψ is constant in the tangent directions.

Inserting these results in 5.19 we find

$$S_{int}[U] = \int \nabla U^t J G^{-1} J^t \nabla U \delta(\Psi) ||\nabla\Psi|| dX^1 dX^2 dX^3 \tag{5.20}$$

All is needed now is to prove the identity

$$JG^{-1}J^t = Id - \eta\eta^t$$

In order to do that we start from property 1 in the Lemma: $G = J^t J$. We take now the inverse of the two sides. Here we must proceed with care. While G is positive definite and there is no problem, J has a non-trivial kernel as we can see from property 4 of the Lemma. The Kernel is one dimensional and is the normal direction to the surface. Its inversion should be followed by a projection on the tangent space of the surface where J^{-1} is well defined. We get therefore

$$G^{-1} = (J^t J)^{-1} = J^{-1}(Id - \eta\eta^t)^t (Id - \eta\eta^t) J^{-t} = J^{-1}(Id - \eta\eta^t) J^{-t}$$

and the identity follows.

5.2 Comparison of the two methods

5.2.1 From explicit to implicit

For simplicity, let us take the pixel length as the unity. To apply the implicit method, first we need to obtain the distance function to the sphere. A question that arises immediately is, which size are we gone to choose for the box that will contain the zero-level surface? To properly conserve the original topology, the resolution should be as fine as the shortest path in the mesh. For a mesh such as the one illustrated in fig. 7.8 of 18979 vertex and 37954 triangles this can yield to a $10000^3 = 1000000000000$ pixels not very convenient volume. But if we have homogeneous meshes with near-to-

constant edges length, the volume size is friendlier, this suggest to choose the size in relation with the homogeneity of the mesh. What we have chosen to do for this work, is to take the product of the number or pixels and a function of the variability of the edges length, as an indicator for the box size.

Then how to obtain the level-set function? This can be done in several ways: The *brute force* (compute directly the distance function), the *closest point transform* ([57]), using a the classical Hamilton Jacobi equation $\|\nabla\psi\| = 1$ ([98]), etc. This last method is very fast but has the inconvenience of changing the topology for folding surfaces as the human cortex. Computing the distance function directly assure a good result but is very very slow: It has to compute for each pixel the minimal distance to the surface, the straightforward computations gives $N_{pixels} \times N_{triangles}$ distance computations. In this work we use a kind of brute force: We first create a binary volume with the pixels "touching" the surface that is then use to look for the distance function in its neighborhood. We use octree searches too to accelerate the process. This way wee guarantee to maintain the topology.

Once we have our implicit surface, we'd like to pass the data too to an *implicit* form, i.e. define the value of the function in each point of the narrow band. A method first proposed in [16] consists in defining the function in the pixels touching the surface as the interpolated value of the triangle touching the pixel, then to apply the PDE

$$\frac{\partial u}{\partial t} + sign(\psi)(\nabla u \dot{\nabla} \psi) = 0$$

to extend this data orthogonally to the surface.

5.2.2 From implicit to explicit

The passage to the mesh representation is mandatory at least once for an implicit surface to be visualized. There exists several methods, the most commonly used is the *Marching cubes* ([55]). Passing the data to this representation can be done more easily, just taking the interpolated value of the function in each vertex. The interpolation method used (linear, cubic, etc.) can make great difference.

5.2.3 Comments on the geometry of the mesh

In the case when our surface representation is a mesh, we observed that the way the diffusion behaves on the data over the surface depends on the geometry on the mesh.

A same surface can be triangulated in different manners: the triangles' edges can have all the same approximated size or not at all; some regions in the surface may be very detailed and need tiny triangles while others may not need but a single triangle; each vertex can have a fixed number of neighbors or not at all, etc. It depends basically on the technique used to produce the mesh.

The next example shows how the mesh influences a Laplace-Beltrami diffusion where the discretization depends on a parametrization.

Take as base surface a simple plane, where it would be easy to paint the flat image (in this example an uncolored Mondrian). We choose naturally the vertex to be in the center of each pixel of the image. As a first meshing, we join the pixels by squares and then divide the squares with a diagonal. As second meshing, we connect the vertex in a more symmetrical manner.

We can see in image 5.4 the correspondent images and the results of the Laplace-Beltrami diffusion using the parametric method explained in subsection 2.3.1. Because of the interpolation of the function over the triangles, before we apply the diffusion it is already possible to determine which mesh is symmetrical and which one is not. After the diffusion, the influence of the connectivity between the vertex is very much more accentuated.

In theory, if the edges of the triangulation are small, this kind of behavior will not be noticeable but very locally. However, it adds a supplementary consideration: a re mesh of the triangulation may be needed before performing regularization.

Work has been done to avoid these problems with more careful discretization: In [28], Desbrun et al. avoided the problem of the edge size variability proposing a suitable parametrization. This particular parametrization can be used in order to estimate the Laplace-Beltrami operator as explained in [59]. But for more complicated regularization models, the task is not that easy. That is why the regularity of the mesh can be an important issue to

Figure 5.3: Original image on a flat triangulation

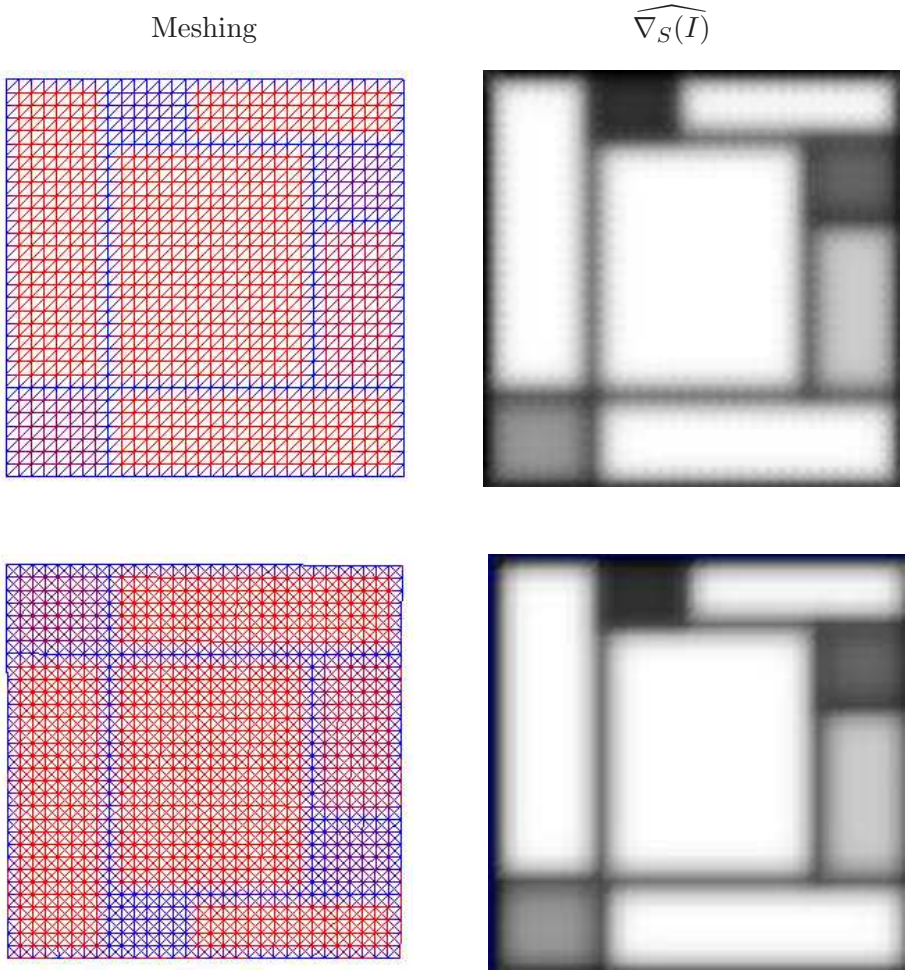


Figure 5.4: Different meshes give different diffusion results

fix before doing any treatment of the data: for complicated PDE models it may be better to go straight to the level-set representation.

5.2.4 Further comments and comparison for a particular application

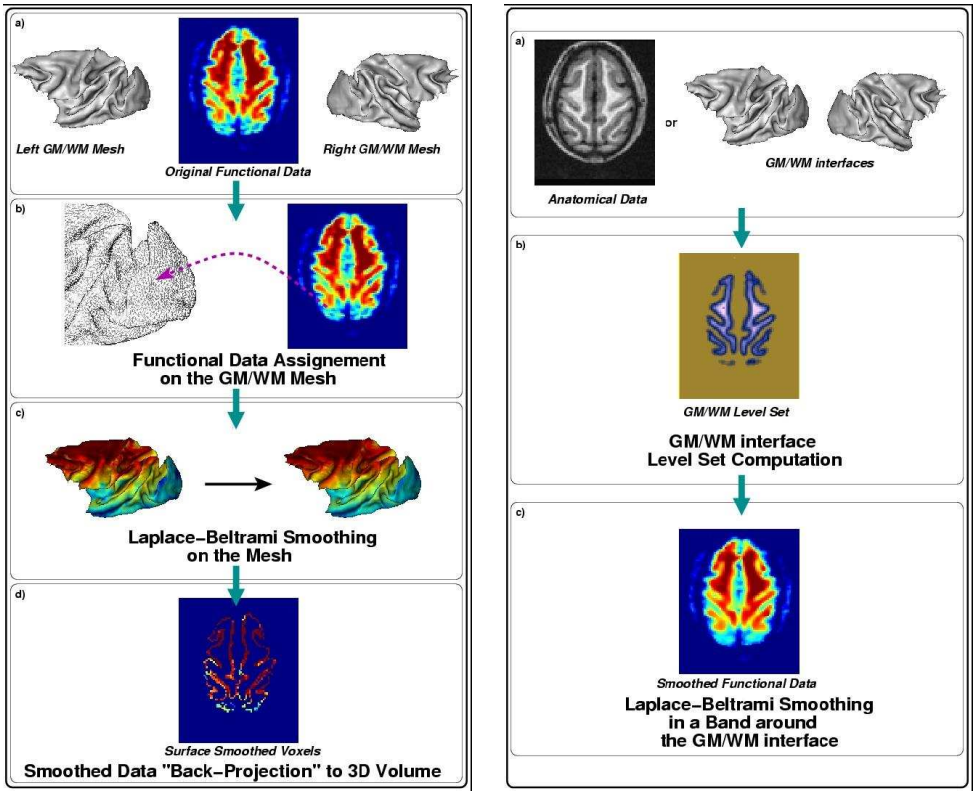
To decide which method is better for a particular application, we have to ask several questions: What is the form of my input? What kind of regularization do I need, a simple isotropic diffusion or more complicated PDE's? Is the surface going to be processed afterward itself? In some problems we want to keep track of certain points during the process, in that case it may be better to keep the mesh intact and use explicit methods straight on the mesh. If the mesh is going to be transformed or for other reason an implicit method will be used anyway, it may be simpler to use that framework from the beginning, and make use of approaches that take care of point track and correspondence as [72].

For a triangulated mesh, and a level set method, it will be needed the computation of the distance function and the extension of the data to a grid. If this is done the straight way, called *brute force*, it can be considerably long, and longer if it is needed for a big set of surfaces. A lot of work has been done to take care of this task and to reduce the computer time required to produce appropriate level sets.

For the reverse process, to produce a triangulated mesh from a level-set representation, the most common method is the one known as *marching cubes* [55], but there too, there exists several alternative methods, for instance: [76].

For the particular application of regularization as a intermediate process in the task of analyzing fMRI data (to produce retinotopical maps of the brain, for instance), a comparison has been made and presented in [105] (see fig. 5.5) by the author in joint work with Odyssée lab colleagues Nicolas Wotawa and Jean Philippe Pons. More details can be found in [103].

This work is to our knowledge the first implementation of an fMRI smoothing method based on the level set framework. The conclusion is that, since the input and output is given in terms of volumetric images, the level set method presents more advantages: the data does not need intermediate processes as for the mesh-based approach, and it is numerically more robust. We find expected differences between both anisotropic methods that



Mesh based approach Level Set based approach

Figure 5.5: Level set vs triangulation based regularization for fMRI data

can be attributed to the use of different numerical schemes and, to a more important degree, to the fact that the mesh-based approach is restricted to projected voxels only whereas the level set based approach solves the PDE equation continuously within a band around the GM/WM interface. The approach is independent of a method assigning functional data to the cortical surface, consequently, the choice of a projection method to visualize the statistical results on the cortical mesh can be made a posteriori.

5.3 Summary and conclusions

We have first clarified the link that exists between the the intrinsic Polyakov action of the Beltrami framework and the implicit harmonic energy functional. It is found that although the functionals are basically the same, there are differences in the way various problems are formulated and consequently in the way the functionals are applied.

In the last section, we discussed the passage between the two representations and the advantages and drawbacks of the two associated methods, and presented the results of a comparison made for retinotopic map extraction. We conclude that it is preferable to keep the representation used by the data whenever it is possible to avoid the error associated with the passage between representations. Nevertheless, the implementation of new PDEs methods remains easier using the implicit framework, and can be advised for explicit surfaces whenever the discretization of the PDE on the triangulation becomes too itchy, or the triangulation itself is too coarse or non-homogeneous (see section 5.2.3).

Chapter 6

Vector image regularization

In this chapter we make an excursion into some vector image regularization techniques using the explicit and explicit frameworks. In the first section we generalize a method to perform unconstrained vector regularization using the explicit approach, and apply it on synthetic examples. In the second section we present some methods for color images regularization, starting by the Beltrami Flow using an explicit framework, and then methods using the implicit framework[1]. In the last section we show synthetical examples.

6.1 Unconstrained vector regularization

In this section we basically generalize some methods and discretization techniques showed in Chapter 2, to the case of the target function being a vector field, then we show implementation and examples for 3-D vector field synthetic images. Unlike the color image case, which is much more complex, the generalization to unconstrained regularization is relatively straight-forward.

6.1.1 Vector regularization: Explicit approach

Starting with the isotropic regularization problem, let us consider S be a triangulated surface, $X : (u, v) \mapsto x \in S$ a conformal parameterization, and $U : S \to \mathbb{R}^n$ a vector image on the surface. Our data consisting on a set of vectors where each vector is attached to a vertex of the triangulated surface.

[1] Work published in [81, 82]

We are then looking to apply the next evolution PDE to the noisy data U_0

$$\frac{\partial U}{\partial t} = \Delta_X U,\ U_{t=0} = U_0, \tag{6.1}$$

$$\begin{aligned} \Delta_S \hat{I}(x) &= \frac{1}{A}\int_A \Delta_S I(x) dx \\ &= (\,\Delta_S I(p) \quad \text{average over the region } A\,) \\ &\approx \Delta_S I(p) \end{aligned}$$

If we look carefully at the discretization method presented in 3.1.1, we see that it is possible to repeat every step considering $U : S \to \mathbb{R}^n$ instead of $I : S \to \mathbb{R}$ since we only use properties fulfilled by vector spaces. We obtain

$$\Delta_S U(p) \approx \frac{1}{A}\sum_{T\in A}(cot\theta_i + cot\beta_i)(U(p) - U(p_i)) \tag{6.2}$$

where θ_i, β_i are the two opposite angles of the edge (p, p_i) in the area A. (see image 3.1).

Image 6.1 shows an example of isotropic regularization over a triangulated sphere using the PDE 6.2. An application of this method is shown at subsubsection 7.2.3.

6.1.2 Bi-dimensional vector field on a surface: Implicit approach

We have in this case an N-dimensional manifold which is described by an implicit function on $\mathbb{R}^{n+1}$: $\Psi(x_1, \ldots, x_{n+1})$. On this manifold we have a bi-dimensional vector field with components U_1 and U_2 extended to $\mathbb{R}^{n+1}$ as well. We describe the data manifold as the intersection of the following functions:

$$\begin{aligned} \Psi(x_1, \ldots, x_{n+1}) &= 0 \\ \Phi_1(x_1, \ldots, x_{n+3}) &= x_{n+2} - \beta U_1(x_1, \ldots, x_{n+1}) = 0 \\ \Phi_2(x_1, \ldots, x_{n+3}) &= x_{n+3} - \beta U_2(x_1, \ldots, x_{n+1}) = 0 \end{aligned} \tag{6.3}$$

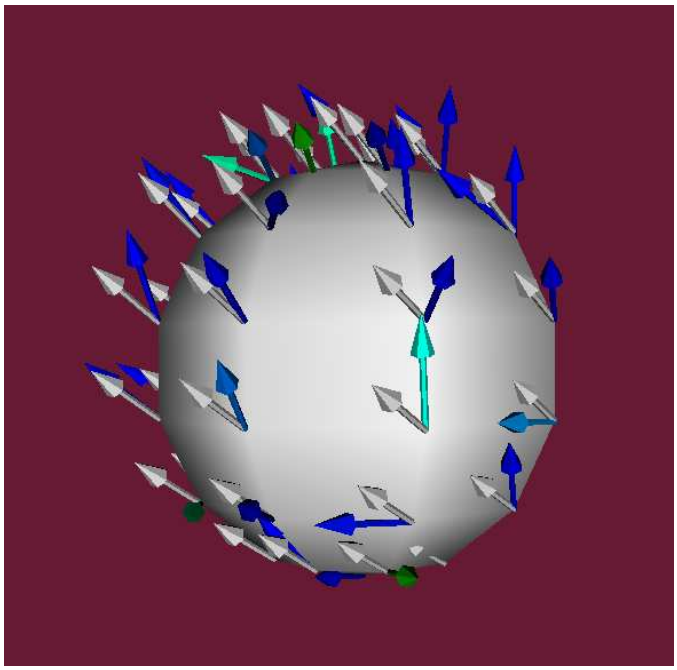


Figure 6.1: Isotropic regularization. The regularized vector field is in gray, while the original noisy one is in a color that codes its angle displacement from the original constant vector image.

The functional is the volume of the manifold of intersection of these three functions. It is given by

$$S[U_1, U_2] = \int \delta(\Psi)\delta(\Phi_1)\delta(\Phi_2)||D\Psi||||P_{D\Psi}D\Phi_1||||P_{P_{D\Psi}D\Phi_1}P_{D\Psi}D\Phi_2|| \tag{6.4}$$

In order to write it in a simpler form we use the following identity. Let v_1 and v_2 be two vectors. then

$$||v_1||^2||P_{v_1}v_2||^2 = ||v_1||^2||v_2||^2 - (v_1 \cdot v_2)^2 \tag{6.5}$$

It follows that

$$||P_{D\Psi}D\Phi_1||^2||P_{P_{D\Psi}D\Phi_1}P_{D\Psi}D\Phi_2||^2 = ||P_{D\Psi}D\Phi_1||^2||P_{D\Psi}D\Phi_2||^2 - (P_{D\Psi}D\Phi_1 \cdot P_{D\Psi}D\Phi_2)^2$$

Next we recall our distinction between the gradient D in $\mathbb{R}^{n+3}$ and ∇ which is the gradient in the first $n+1$ coordinates. It follows that

$$\begin{aligned} P_{D\Psi}D\Phi_1 &= (-\beta\nabla U_1, 1, 0)^t \\ P_{D\Psi}D\Phi_2 &= (-\beta\nabla U_2, 0, 1)^t \end{aligned} \tag{6.6}$$

We arrive finally to a functional that reads

$$S[U_1, U_2] = \int \delta(\Psi)||D\Psi||\sqrt{1 + \beta^2\sum_i ||P_{\nabla\psi}\nabla U_i||^2 + \beta^4||P_{\nabla\psi}\nabla U_1 \times P_{\nabla\psi}\nabla U_2||^2} \tag{6.7}$$

This functional is the obvious generalization of the functional Eq. (5.15).

6.2 Constrained regularization: Color images

Color images are an important case of multivalued images that have to be treated separately. Colors in digital imaging are usually modeled as triplets of RED, GREEN and BLUE channels (RGB, see fig. 6.2), seen as vectors in $\mathbb{R}^3$. There exist others, as the CMYK (cyan, magenta, yellow,) model used by printers, RYB (red, yellow, blue) used traditionally by artists, HSV (hue, saturation, brightness), etc.

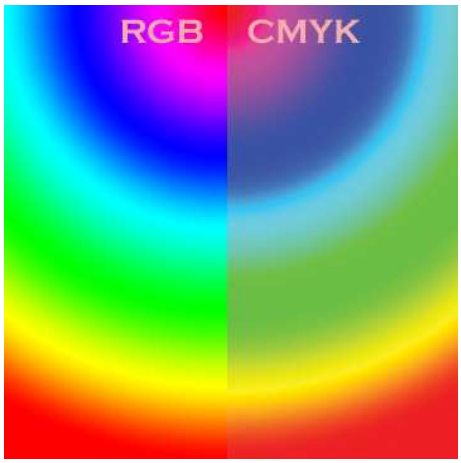


Figure 6.2: RGB and CMYK color models

In order to regularize color images, we could try to apply scalar regularization to each channel. This naif approach leads to a *color blending effect* that segments the image by the color channels (see image 6.3). The first column shows the original noisy color image, the second is the result of a channel by channel approach, and the third row is the result of a vector regularization approach (image from [96]). To get an acceptable result, we need to define local vector geometries that describe vector-valued image variations and structures. One way to do this is to represent colors using the unit direction vector and its norm, performing scalar regularization on the norm image and regularizing the unit vector image separately, taking into account the unit norm constraint. This is why color image regularization is considered as constrained regularization.

6.2.1 Beltrami flow, explicit-intrinsic approach

We consider as the base manifold a two-dimensional surface. The fiber is the color space. We will treat here, for simplicity, the case where the color space is taken to be a Euclidean $\mathbb{R}^3$ space with Cartesian coordinates (R, G, B). One can also consider the color space to be a 3D Riemannian manifold [89].

Here we describe the color image on a surface as embedding of a 2D surface S embedded in 5D manifold $\mathcal{M} = \Sigma \times \mathbb{R}^3$. The metric in the embedding

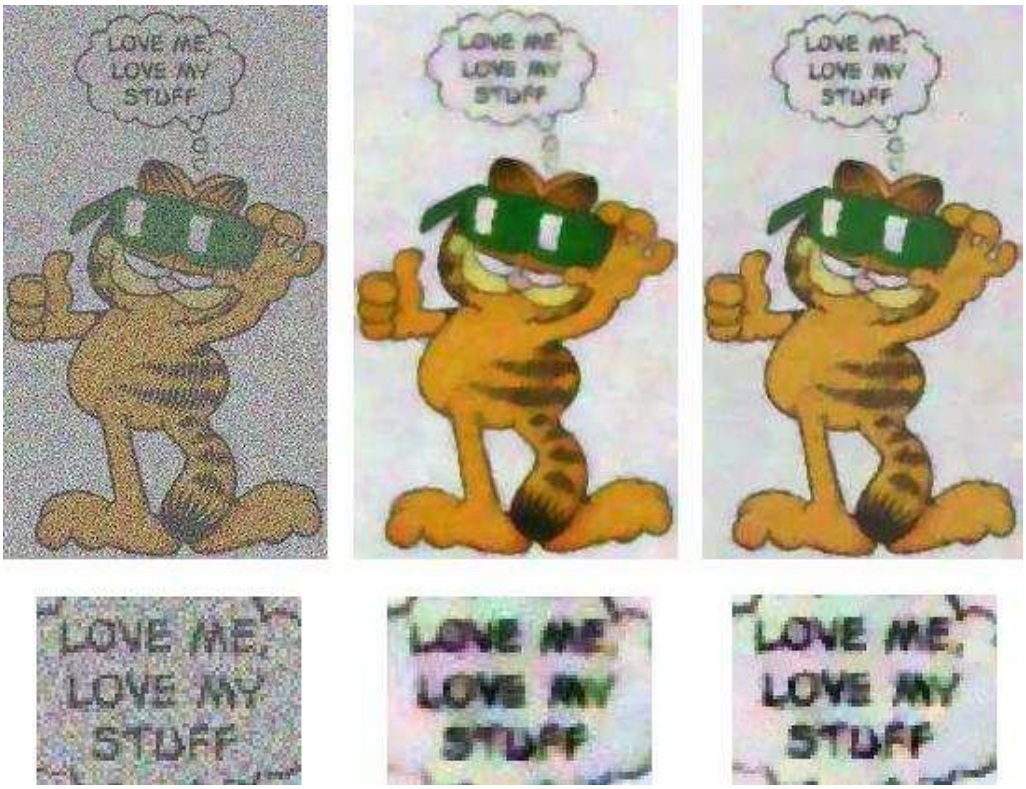


Figure 6.3: Channel by channel approach vs vector-valued PDE's

space is

$$(h_{ij}) = \begin{pmatrix} \tilde{g}_{11} & \tilde{g}_{12} & 0 & 0 & 0 \\ \tilde{g}_{21} & \tilde{g}_{22} & 0 & 0 & 0 \\ 0 & 0 & \beta^2 & 0 & 0 \\ 0 & 0 & 0 & \beta^2 & 0 \\ 0 & 0 & 0 & 0 & \beta^2 \end{pmatrix}.$$

The line element is therefore

$$ds_{\mathcal{M}} = \tilde{g}_{11}dx^2 + 2\tilde{g}_{12}dxdy + \tilde{g}_{22}dy^2 + \beta^2\left(dR^2 + dG^2 + dB^2\right).$$

and the induced metric is

$$G = (g_{ij})$$

$$= \begin{pmatrix} \tilde{g}_{11} + \beta^2\left(R_x^2 + G_x^2 + B_x^2\right) & \tilde{g}_{12} + \beta^2\left(R_xR_y + G_xG_y + B_xB_y\right) \\ \tilde{g}_{21} + \beta^2\left(R_xR_y + G_xG_y + B_xB_y\right) & \tilde{g}_{22} + \beta^2\left(R_y^2 + G_y^2 + B_y^2\right) \end{pmatrix}.$$

It is easy to see that the Levi-Civita connection coefficients vanish in this case for the same reason it does in the scalar case. We finally obtain the equations of motion

$$U_t^a = \Delta_g U^a = \frac{1}{\sqrt{g}}\mathrm{Div}\left(\sqrt{g}G^{-1}\nabla U^a\right) \qquad a = r, g, b$$

Example: Color image on the sphere

In order to apply the above equations to a color image "painted" on the sphere we need the metric $\tilde{g}_{ij}$ of the sphere. We choose to represent the sphere by stereographic coordinates from the south pole. The metric in this case was calculated already in the scalar case (see appendix 10.1) and it is

$$\tilde{g}_{ij} = \frac{4}{1 + x^2 + y^2}\delta_{ij}\ .$$

The induced metric of the two-dimensional section of this five-dimensional fiber bundle is

$$G = (g_{ij}) = \left(\tilde{g}_{ij} + \beta^2 \sum_{a=R,G,B} I_i^a I_j^a\right)\ ,$$

where $I_i^a = \partial U^a / \partial x^i$ with $x^1 = x$ and $x^2 = y$. Define

$$A(x,y) = \frac{4}{1+x^2+y^2} \ .$$

With this notation the determinant read

$$g = det(g_{ij}) = A^2 + A\beta^2 \sum_a ((U_x^a)^2 + (U_y^a)^2) + \beta^4 \sum_{ab} |\nabla U^a \times \nabla U^b|^2 \ .$$

The evolution equations are

$$U_t^a = \frac{1}{\sqrt{g}} \mathrm{Div} \left(\frac{1}{\sqrt{g}} \begin{pmatrix} A + \beta^2 \sum_b (U_y^b)^2 & -\beta^2 \sum_b U_x^b U_y^b \\ -\beta^2 \sum_b U_x^b U_y^b & A + \beta^2 \sum_b (U_x^b)^2 \end{pmatrix} \nabla U^a \right) \ .$$

6.2.2 Isotropic and anisotropic regularization of color images on implicit surfaces

Here we represent the color images as functions $\mathbf{u} : \mathcal{S} \to S^2 \times \mathbb{R}$. These functions take as values unit vectors on the $\mathbb{R}^3$ sphere whom represent chromaticity vectors, and a real component that represents the brightness (the magnitude of the color vector) which we treat separately following the previous results. So we have to focus on the smoothing of the S^2 component, minimizing an harmonic energy of the form

$$\int_{\mathcal{S}} \|\nabla_{\mathcal{S}} u\|^p \, d\mathcal{S}, \quad s.t. \ \|u\| = 1.$$

for $u : \mathcal{S} \to S^2$.

The gradient descent is given by

$$\frac{\partial u_i}{\partial t} = div_{\mathcal{S}} \left(\|\nabla_{\mathcal{S}} u\|^{p-2} \nabla_{\mathcal{S}} u_i \right) + u_i \|\nabla_{\mathcal{S}} u\|^p, \quad 1 \le i \le n. \tag{6.8}$$

with $p = 2$ for isotropic and $p = 1$ for anisotropic diffusion. The first term is obtained in the same way that the one for the scalar case, and the second comes from the unit norm constraint (see appendix 10.3).

Implementation and examples

The program takes as input a color volume: to each pixel a triplet is assigned. It first normalizes the data to obtain the chroma vector, and saves the norm

in another volume. Then apply the numerical schemes explained in 10.4 to the chroma vector, and the ones of 10.4 to the brightness, to obtain the diffused result.

Our first example (image 6.5) we use a normalized data, and apply an additive gaussian noise only to the chroma vector (before normalization), keeping the brightness component unchanged. The second example (image 6.5) shows isotropic and anisotropic diffusion, applied only in the chroma vector. The noise is so strong that even after the anisotropic diffusion it succeeds at breaking a border and change the color in blue. Despite the lost of the contours with the diffusion, the original colors are very well restored. In the next example the surface is the bombed "odyssee" word (surface modeled using VTK), the data is again artificially generated as intersection of sets of the form

$$\left\{ u_{i,j,k}[m] = c : \frac{1}{3}\dim_m(u) \leq e_m \cdot (i,j,k) \leq \frac{2}{3}\dim_m(u) \right\},$$

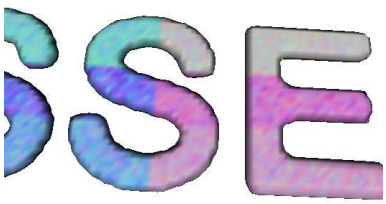
$c \in \{1, ..., 255\}$ and $\dim_m(u)$ is the m^{th} component of the dimension of the volume u. Here we have applied separately a gaussian noise to the brightness vector, the same σ.

The last example (6.6) is a flower image painted over a quadratic surface, both the chroma vector and brightness have been passed by the anisotropic regularization filter.

6.3 Summary and conclusions

In this chapter, we show an implementation of the regularization techniques showed before extended to multivalued images and some synthetic examples to illustrate this techniques are presented. We covered some theoretical aspects and presented methods for the Beltrami flow over color images and the isotropic regularization on vector images. An application of this technique for retinotopic mapping is illustrated later on chapter 7. We show synthetic examples of the isotropic and anisotropic implicit regularization over vector images.

Original image

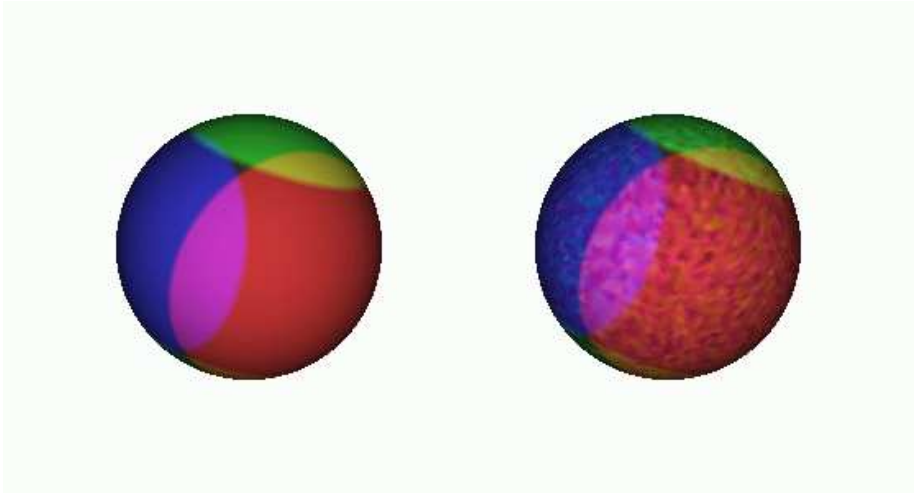


Added noise (detail)

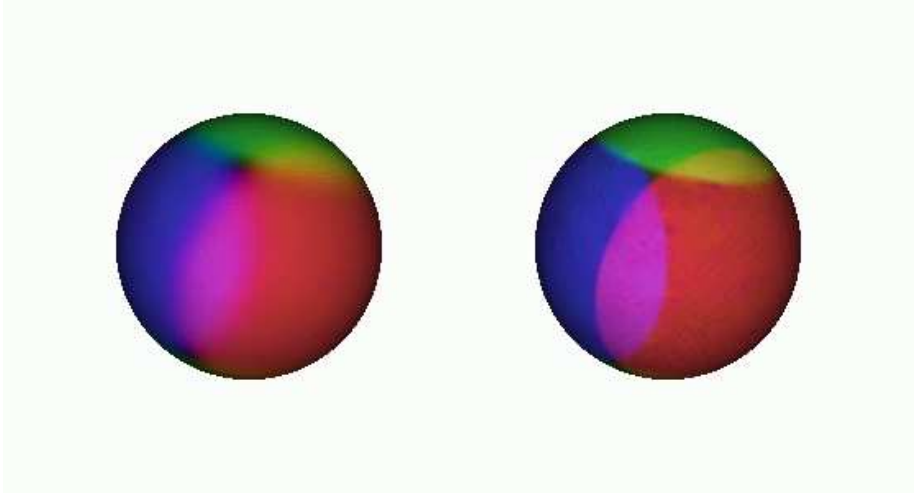
Isotropic diffusion

Anisotropic diffusion

Figure 6.4: Example of isotropic and anisotropic diffusion over the colored "Odyssee" surface



Original image Image with noise added



Isotropic diffusion Anisotropic diffusion

Figure 6.5: Example of isotropic and anisotropic diffusion over a colored sphere

Original noisy image over quadratic surface

After anisotropic regularization

Figure 6.6: Example of anisotropic regularization of a color image: flower painted on a quadratic surface

Chapter 7

Application to cortical images

In the first section of this chapter we explain the way the extraction of the retinotopic areas is done (for more detail see [103], [104], [37]), starting by a brief explanation of the visual system process on primates and then resuming the extraction methodology. In the second section we show the importance of the data regularization and our efforts to improve the way it is done to obtain better retinotopic maps.

7.1 Occipital retinotopic areas extraction

The visual cortex is the most massive system in the human brain and is responsible for higher-level processing of the visual image. It lies at the rear of the brain, above the cerebellum. The portion of the cortex involved in vision processing has been estimated to be around 50% in the macaque monkey, with over twenty distinct areas identified. Some of these areas are quite well understood, others are still a complete mystery.

Nearly all visual information reaches the cortex via the so called V1, the largest and most important visual cortical area. Because of its stripy appearance this area is also known as striate cortex, among other things. Other areas of visual cortex are known as extrastriate visual cortex; the more important areas are V2, V3, V4 and MT, also known as V5 (see fig. 7.1)

For this application, we are interested in labeling some areas of the visual

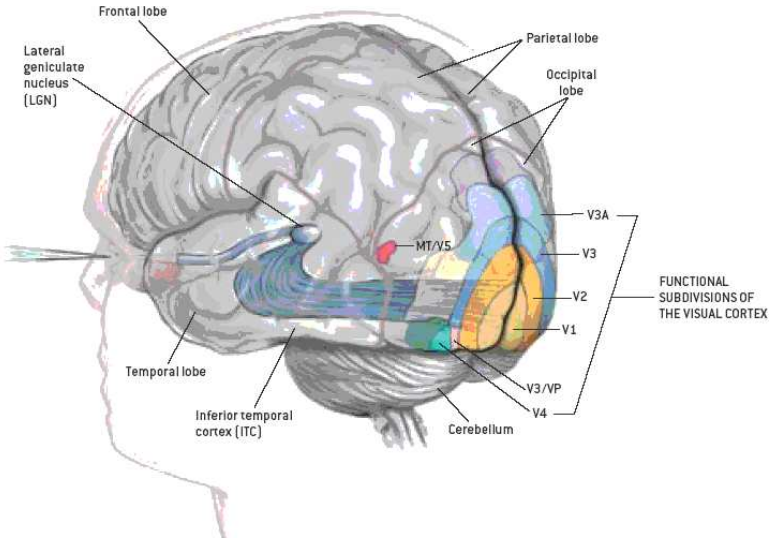


Figure 7.1: Visual cortex and some of its principal areas (image from[53])

cortex using the retinotopic representation of the visual field. This criterion has been used to reveal the "low level" areas first by Engel et al. in [35] and later by [80, 32, 99, 104].

We now briefly explain the biological visual system process in primates, more comprehensive explanations can be found in [104, 67, 13, 46].

7.1.1 Biological visual system

The visual process begins with the light entering the eye, first passing through the cornea, then the aqueous humor, the pupil (controlled by the iris) and is refracted by the lens before passing trough vitreous humor to finally project an image onto the *retina*. The retina is covered with over 125 million photosensitive receptors of two families: the cones (around 8 millions, mainly concentrated in the center of the retina, the fovea) are responsible for chromatic and normal lighting condition vision (called photopic), and the rods (around 120 millions, found everywhere except in the fovea), dealing with black and white perception and low-lighting conditions (called scotopic). These receptors translate lighting information into electrical information, transmitted to the optical nerves via the ganglion cells. The two optical nerves cross, forming the optic chiasm, after which information is transmitted by visual hemifield (separated vertically with respect to the head position): the information from photons striking the left parts of both

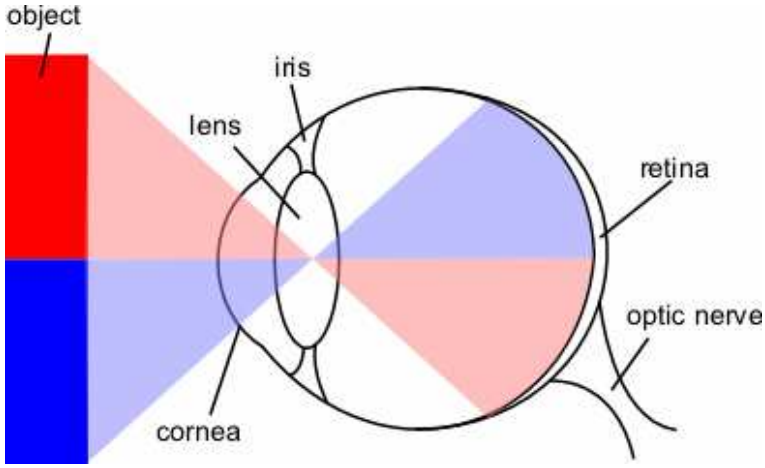


Figure 7.2: Visual process: eye

retina and corresponding to the right visual field is brought together to form the left optical tractus.

Following some rudimentary processing (mostly involving color boundaries), the information about the image received by the eye is transmitted to the brain via the optic nerve. In humans, the optic nerve is the only sensory system that is connected directly to the brain and does not connect through the medulla, due to the necessity of processing the complex visual information quickly.

The optic nerves from both eyes meet and cross at the *optic chiasm*, at the base of the frontal lobe of the brain. At this point the information from both eyes is combined and split according to the field of view (see figure 7.3). The corresponding halves of the field of view (right and left) are sent to the left and right halves of the brain, respectively (the brain is cross-wired), to be processed. That is, though we might expect the right brain to be responsible for the image from the left eye, and the left brain for the image from the right eye, in fact, the right brain deals with the left half of the field of view, and similarly for the left brain. The right eye actually perceives part of the left field of view, and vice versa. Information from the right visual field (now on the left side of the brain) travels in the left optic tract. Information from the left visual field travels in the right optic tract. Each optic tract terminates in the lateral geniculate nucleus (LGN) in the thalamus.

The *lateral geniculate nucleus* (LGN) is a sensory relay nucleus in the thalamus of the brain. The LGN consists of six layers in humans and some other

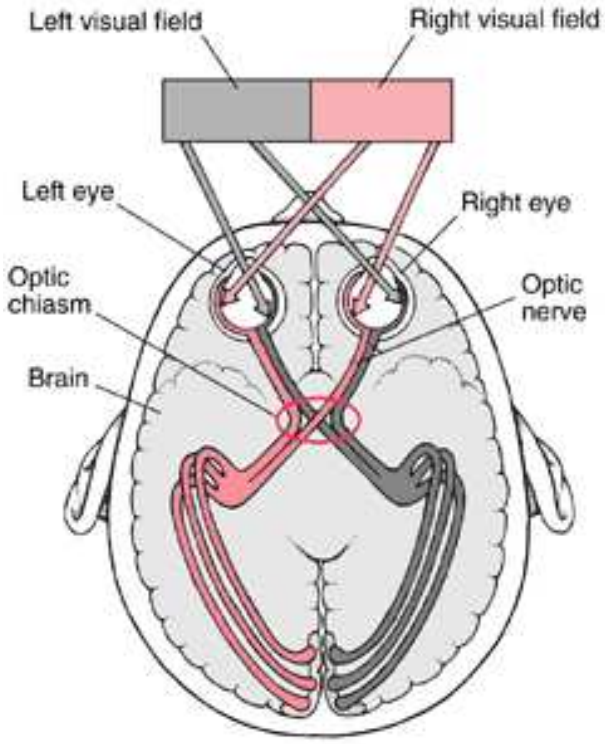


Figure 7.3: Optic chiasma and lateral geniculate nucleus.

primates such as macaques. The neurons of the LGN relay the visual image to the primary visual cortex, and we arrive where we wanted to.

7.1.2 Extraction method

Occipital retinotopic areas are delineated using the classical fMRI method used first in [35]. Phase-encoded stimulus for the retinotopic mapping consisted of a 9Hz flickering black-and-white checkerboard into a 80° rotating wedge or a varying size (depending on the eccentricity) ring. To get the polar angle maps, 8 complete rotations, either clockwise or counter-clockwise, were performed in each wedge scan. 4 complete rotations, either inward or outward, were performed in each ring scan, to get the eccentricity maps (see figs. 7.4 and 7.5). Retinotopic stimulus signals for each rotating sens were modeled through a General Linear Model with a cosine and sine regressors having the same period as the stimulus, followed by a voxelwise signal-phase computation (including a hemodynamic response delay estimation) closely related to the stimulus position in the visual field.

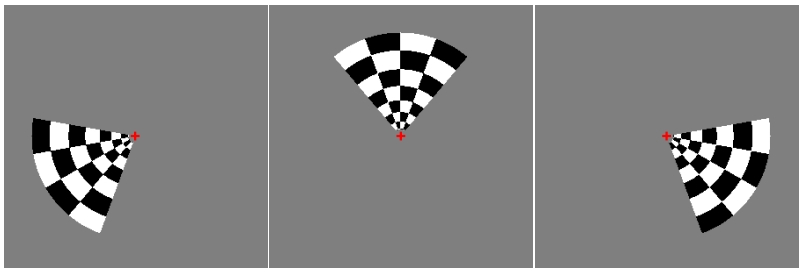


Figure 7.4: The *wedge* stimulus seen in different positions. It encodes the polar angle coordinate θ in the visual field.

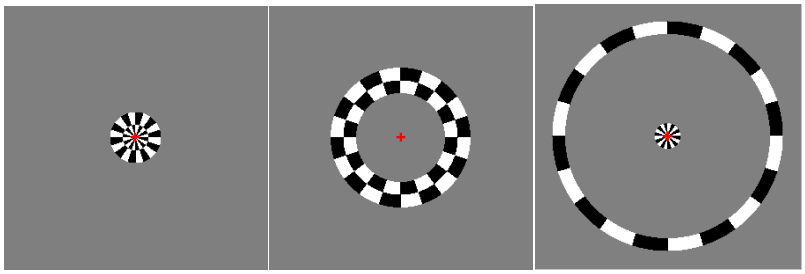


Figure 7.5: The *ring* stimulus seen in different positions. It encodes the eccentricity coordinate ρ in the visual field.

The resulting phase maps were masked by a $P < 0.001$ threshold F-test on

the estimates computed at each voxel. These maps were then projected as color-coded maps onto models of the white-matter/gray-matter interfaces (one topologically spherical surface by hemisphere) using a combination of the Brainvisa software (http://brainvisa.info) and algorithms developed in the Odyssée laboratory. Then the scalar maps of the ring and wedge data are regularized using first an isotropical diffusion over the cortical surface (see fig. 7.6)

For the purpose of ease the visualization, the models were further inflated, and V1, V2v, V3v, V4v, V2d, V3d and V3A could be identified based on the angular maps pattern.

The output of the process are the Visual field sign (VFS) maps. The VFS is computed from the cortical surface-intrinsic gradients of the eccentricity and polar angle (ring and wedge) using this formula:

$$VFS = sign(\det(\nabla\rho, \nabla\theta, N))$$

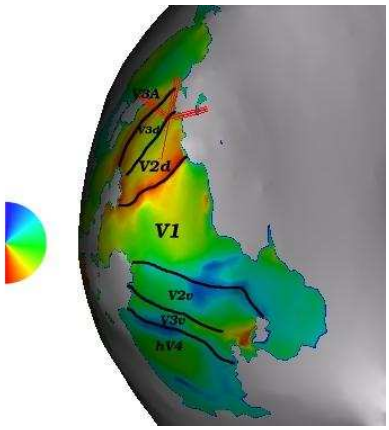
The data is from the MRI scanner located in the Timone fMRI center in Marseille in France.

Results using isotropic smoothing in the regularization process are shown in fig. 7.6: (A) shows a polar angle map projected on an inflated left occipital cortex. The area boundaries were drawn by hand, based on the angular gradients. Images (B) and (C) represent respectively the results of the visual field sign computing, based on the original polar angle and eccentricity maps and after a surface-based smoothing of the angular maps with a gaussian kernel ($\sigma = 3mm$) in (C). Red (green) color indicates a non-mirror (mirror) local representation. The smoothing does not globally improve the global result; the red cross even shows a position where the original VFS (left) performed better.

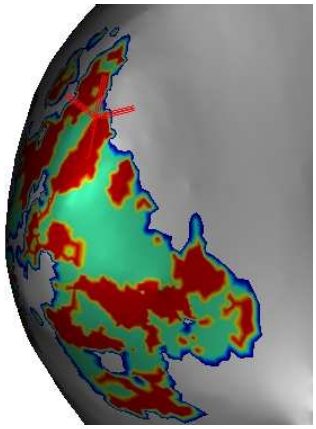
In the next subsection we show results of our experiments done on different regularization methods used on these maps.

7.2 Regularization methods

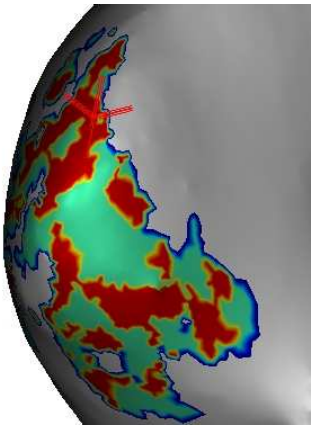
In this subsection we will show different ways to perform regularization on this problem. The experiments are done on different sets of data, unlike the comparison experiment mentioned on subsection 5.2 where the purpose



A) Polar angle map



B) VFS map



C)VFS map after angular maps smoothing

Figure 7.6: Visual field signed maps

was to compare the explicit and implicit scalar diffusion methods on the extraction problem (the same set of data was used), here we simply want to see for different sets of data which regularization method can perform better. The aim is to obtain a final map with clear boundaries between the visual regions in terms of the prior knowledge we have from them. The final contour extraction is supposed to look like the hand-drawings from fig. 7.6-A or fig. 7.1.

In the first two subsections we make use of the Beltrami Flow approach applied on the scalar maps. In the third we are more interested on applying the regularization on the gradient vector data rather than on the resulting scalar maps only, with surprising results.

7.2.1 Implicit Beltrami flow scalar regularization

The surface is a slice from a cortex (97x222x143). It took less than 10 minutes to compute these results (fig 7.7). This is a retinotopic map i.e a neural representation within the visual cortex that preserves the spatial layout of the retina image. Notice the red holes in the noisy image. The small holes have to be filled in the map. Our approach allows to fill in these holes while preserving the important borders. While the isotropic smoothing also performs this task, it does not allow to preserve important information like the blue zone in the extreme right. With the L_1 anisotropic smoothing, the outer borders are thinner, but the inner holes rest. Our approach allows us to deal with more degree of freedom in the process to manage the weighting between the isotropy and anisotropy processes. Choosing beta in a range from 0.1 to 0.5 allows to produce a whole set of interesting and better results since the blue zone is kept and the holes are filled in.

7.2.2 Explicit regularization

The scalar data used in this experiment is very noisy because of the measurement errors. Already from the beginning, we note a strong stair-casing effect due to the differences of resolution between the anatomic (fig. 7.8) and functional (fig. 7.9) brain images.

Even though we have no access to the original non-noisy image, we can select a region where we know that the original image has null variance. We can do this because we know that there should be no electric impulsions

outside a certain region of the cortex. To compare the performance with different values of β, we applied the flow until the variance measurement in the selected region reaches a fixed threshold. Then we observed how different values of β act on discontinuities on the image for the same amount of noise reduction on the selected region.

7.2.3 Explicit vector regularization

For the examples in the two previous subsections we made the regularization on the scalar wedge and ring maps. Here we try it on the $\nabla\rho$ and $\nabla\theta$ fields later used to compute the VFS, using the isotropic regularization method proposed in section 6.1.1.

We show the VSF results using the vector regularization and compared them with the polar angle scalar regularization on fig. 7.11: A) Polar angle map overlaid on the in inflated left occipital cortex. The areas boundaries were drawn by hand, based on the angular variations pattern. B) VFS results based on the original polar angle and eccentricity maps. C) VFS results based on the surface-based smoothing of the angular maps (equivalent Gaussian kernel with $\sigma = 3mm$. D) VFS results based on the surface-based angular gradients field smoothed along the cortical geometry.

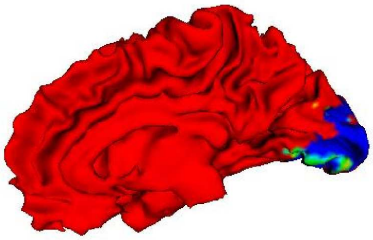
This 3D vector field smoothing step is followed by the classical VFS computation. The resulting VFS maps are once again far less noisy than the classical ones and also better than that obtained with the phase maps surface smoothing, especially preserving better the stripes shape of V2 and V3.

7.3 Summary and conclusions

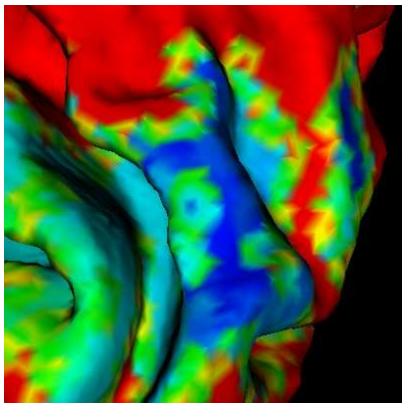
We have detailed a method to obtain a human individual retinotopic map of the occipital cortex using fMRI, which makes use of our surfaces-based regularization techniques rather than a volume regularization method (see image 1.1) leading to better results. The acquisition time is below what is generally described and the resulting maps are consistent with those published. These results furthermore show a reliable reproducibility, across and among subjects.

In the second section we propose alternative ways to improve the result using

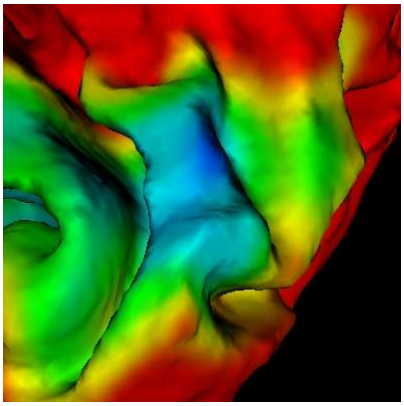
vector regularization over the gradient data instead of the scalar regularization over its projected components. We also show how the Beltrami flow regularization can improve the resulting maps, tunning the β parameter to erase the noise like the isotropic smoothing without letting the stair-casing effect related to the different resolution of the data sets give faulty results.



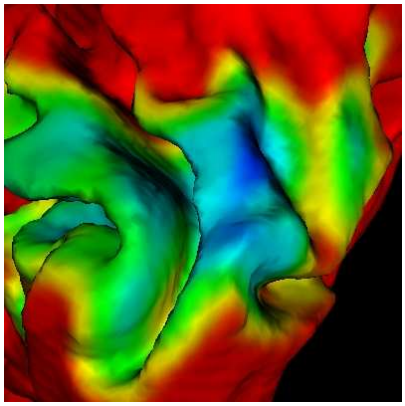
Original noisy image



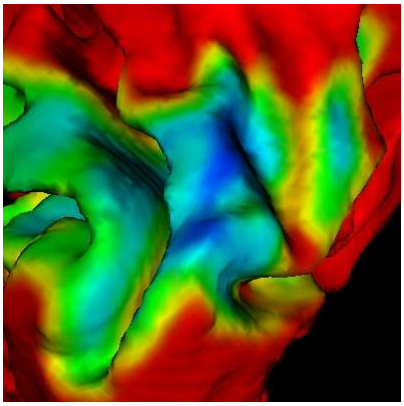
Detail



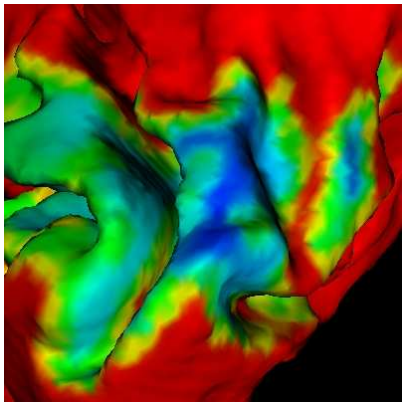
$\beta = 0$ (Isotropic diffusion)



$\beta = 0.1$



$\beta = 0.5$



L_1 (Anisotropic diffusion)

Figure 7.7: Retinotopic images regularized with different β

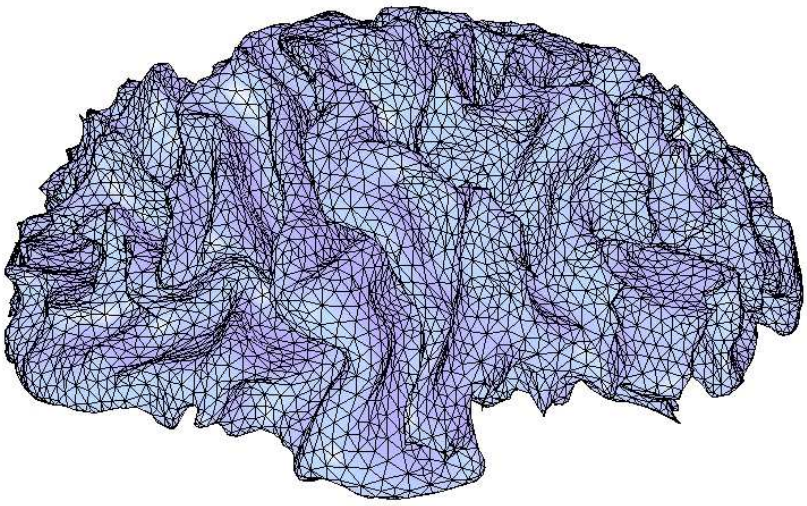
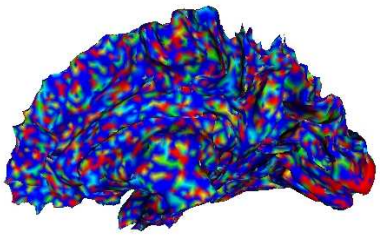
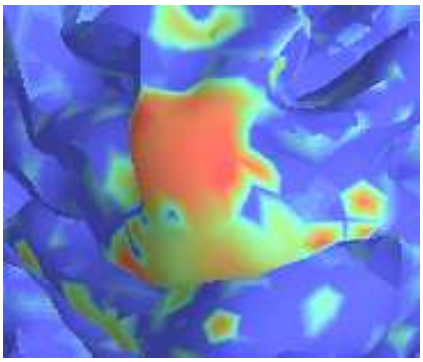


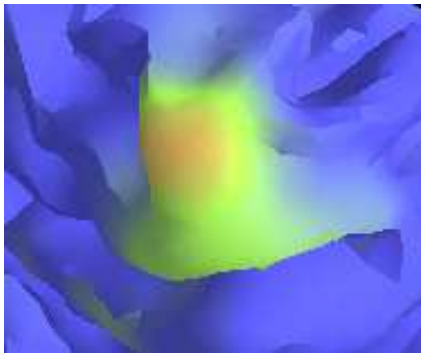
Figure 7.8: Triangulation used: 18979 vertex, 37954 triangles



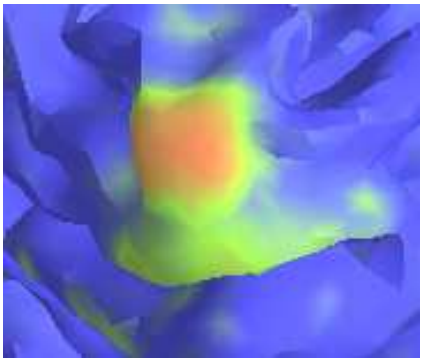
Original noisy image



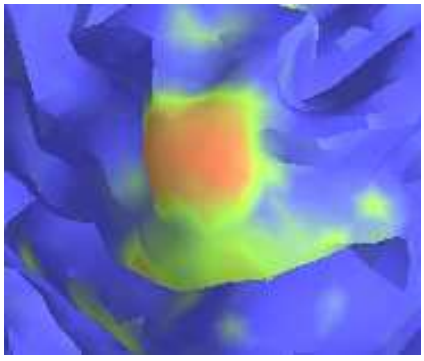
Area of interest



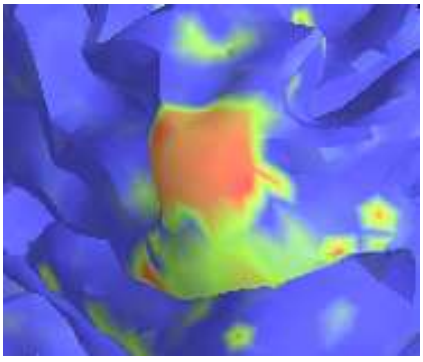
$\beta = 0$ (Isotropic diffusion)



$\beta = 0.0001$

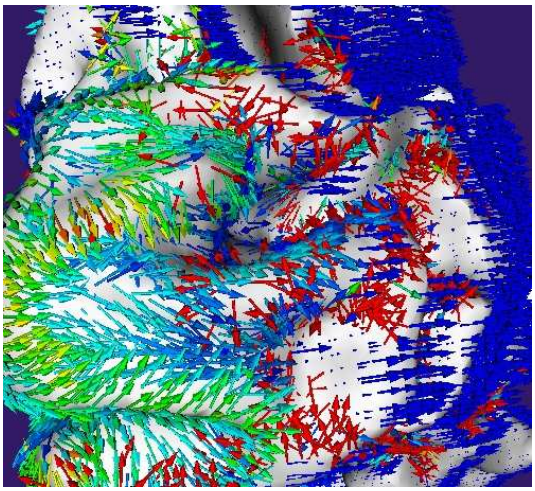


$\beta = 0.1$

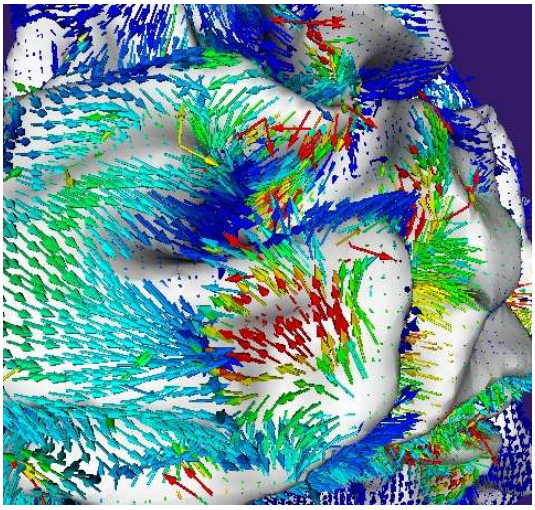


$\beta = 1$ (Anisotropic diffusion)

Figure 7.9: Eccentricity image of the visual areas on the cortex

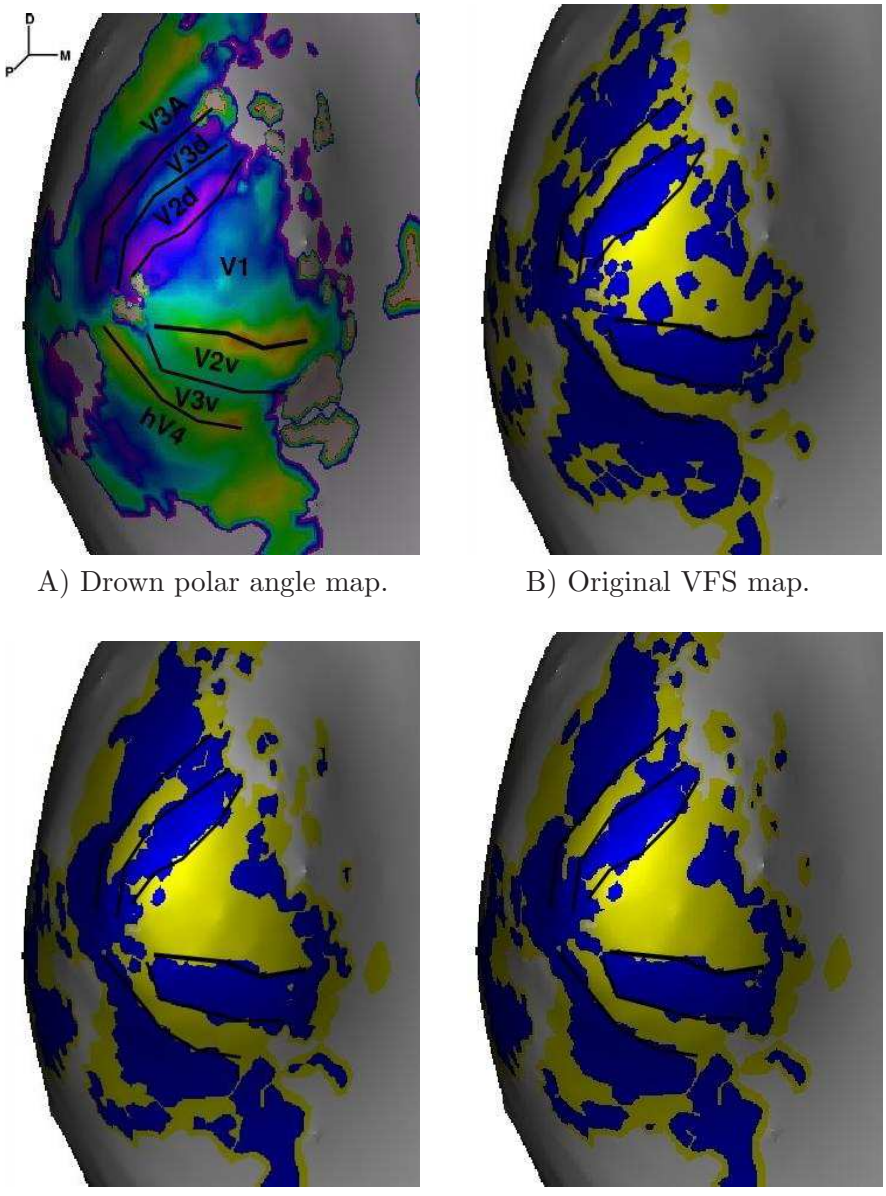


Original noisy image



Isotropic regularization
$dt = 0.1$, $n = 10$

Figure 7.10: Isotropic regularization on the vector field. The color of the vectors is associated with their norm.



A) Drown polar angle map. B) Original VFS map.

C) VFS map after 1D smoothing D) VFS map after 3D smoothing .

Figure 7.11: Comparison of the results: scalar vs vector regularization.

Chapter 8

Conclusions

Contributions

In this thesis, we studied regularization methods for images defined on surfaces, focusing on PDE-based approaches.

First we introduced the most used surface representations, presented the Laplace-Beltrami operator and the state of the art regularization methods, and discussed the bibliography of this domain. Later we propose novel regularization techniques, starting by the anisotropic regularization on explicit surfaces, then the Beltrami flow scalar regularization on implicit and explicit surfaces.

This new regularization technique overcomes the over-smoothing of the L_2 flow and the stair-casing effects of the L_1 flow, that were suggested via the harmonic map methods. We clarify the link between the intrinsic *Polyakov action* and the implicit *Harmonic energy functional* and then use the geometrical understanding of the Beltrami flow to generalize it to images on explicitly and implicitly defined non flat surfaces. It is shown that, as in the case of flat images ([47]), the Beltrami flow interpolates between the L_2 and L_1 flows on non-flat surfaces.

The implementation scheme of this flow is presented and various experimental results obtained on a set of various real images illustrate the performances of the approach as well as the differences with the harmonic map flows.

We also compare the performance and ease of implementation of scalar regularization methods given the surface representation used. Then we made an

excursion on some vector image regularization methods for the constrained and unconstrained case, in particular the Beltrami flow regularization on color images. Last we showed an application of our work to a concrete problem, the extraction of the visual areas of the Cortex.

Our algorithms are also used for the estimation of differential operators defined on triangulated surfaces in applications such as the the MEG/EEG inverse problem (see [1, 2]). In [92, 91], the author propose an alternative numerical method to the PDE gradient descent presented in this work for the Beltrami flow, based on a short time kernel. This kernel enables the implementation of the Beltrami Flow by a convolution of the image with the kernel, similarly to the implementation of the heat equation by a convolution with the gaussian kernel.

The main contributions of this work are:

- Comparison of scalar regularization methods given the surface support.
- Stable numerical schemes for isotropical and anisotropical regularization on implicit surfaces.
- A novel discretization method for divergence-like operators for regularization over explicit surfaces. Its relation to known methods.
- The extension to non-flat images of the Beltrami flow Regularization technique.
- Implicit and Explicit numerical methods for the Beltrami flow regularization.
- The clarification of the relation between implicit and extrinsic approaches.
- The application of these methods to a concrete retinotopic mapping problem.

Future work

This thesis opened new questions that could be considered for future works in the theoretical and applicative aspects. The discrete approximation for a divergence-like operator proposed in section 3.3 and its relation to DEC

operators points interest in this direction. Convergence and uniqueness of our proposed schemes need to be covered. Work has been started on segmentation methods over surfaces using our proposed PDEs.

For the multivalued image case, we have covered some theoretical aspects and presented methods for the Beltrami flow over color images and the isotropic regularization over vector images. There is still much work for the case of tensor images, which gain interest because of their growing number of applications in brain structure research. Numerical methods for more general vector image regularization methods have to be developed.

For future work, it could be of interest to make further use of the Beltrami flow techniques for the regularization step of segmentation applications such as the cortex mapping and the MEG/EEG inverse problem. We have use the Beltrami flow technique in the scalar regularization step of the retinotopic mapping problem, but it could also be used for the regularization of the gradient vector field.

Conclusions

Contributions

Dans cette thèse, nous avons étudié des méthodes de régularisation pour des images définies sur des surfaces, nous concentrant sur des approches basées sur des EDPs.

Nous avons d'abord introduit les représentations des surfaces les plus utilisées, présenté l'opérateur de Laplace-Beltrami et les méthodes de régularisation existantes, et discuté la bibliographie de ce domaine. Nous avons proposé ensuite des techniques de régularisation innovantes, commencant par la régularisation anisotrope sur les surfaces explicites, puis la régularisation scalaire du flot de Beltrami sur les surfaces implicites et explicites.

Cette nouvelle technique de régularisation surmonte le sur-lissage du flot L_2 et l'effet d'escalier du flot L_1, qui ont été suggérés par les méthodes de fonctionnelles harmoniques. Nous clarifions le lien entre *l'action de Polyakov* intrinsèque et *la fonctionnelle d'énergie harmonique* et puis employons l'interprétation géométrique du flot de Beltrami pour le généraliser aux images définies sur des surfaces non planes explicites et implicites. Nous montrons que, comme dans le cas des images planes ([47]), le flot de Beltrami interpole entre les flots L_2 et L_1.

Une implémentation de ce flot est présentée ainsi que divers résultats expérimentaux obtenus sur un ensemble d'images réelles que montrent les performances de cette approche ainsi que ses différences avec les méthodes de fonctionnelles harmoniques.

Nous avons comparé également les performances et les complexités d'implémentation des méthodes de régularisation de données scalaires selon la représentation de surface utilisée. Après, nous avons fait une discussion

de quelques méthodes de régularisation d'images vectorielles avec et sans contraintes, en particulier la régularisation du flot de Beltrami sur des images de couleur. Nous avons aussi montré une application de notre travail à un problème concret, l'extraction des régions du cortex relatives à la vision.

Nos algorithmes sont également employés pour l'estimation d'opérateurs différentiels définis sur des surfaces triangulées dans des applications telles que le problème inverse de la MEG/EEG (voir [1, 2]). Dans [92, 91], l'auteur propose une méthode numérique alternative à la descente de gradient présentée dans ce travail pour le flot de Beltrami, basée sur un noyau à court terme. Ce noyau permet l'implémentation du flot de Beltrami par une convolution avec l'image, de manière équivalente à l'implémentation de l'équation de la chaleur par une convolution avec un noyau gaussien.

Les contributions principales de ce travail sont :

- Comparaison des méthodes de régularisation des données scalaires selon la représentation de la surface d'appui.
- Des schémas numériques stables pour la régularisation isotrope et anisotrope sur des surfaces implicites.
- Une méthode de discrétisation des opérateurs de type divergence pour la régularisation sur des surfaces explicites, et sa relation avec les méthodes existantes.
- La généralisation du flot de Beltrami aux images définies sur des surfaces non planes, implicites et explicites.
- Des méthodes numériques explicites et implicites pour la régularisation du flot de Beltrami.
- La clarification de la relation entre les approches implicites et extrinsèques.
- L'application de ces méthodes au problème de segmentation des cartes rétinotopiques du cortex humain.

Perspectives

Cette thèse a posé des nouvelles questions qui pourraient être considérées pour des travaux futurs, tant sur les aspects théoriques que sur les applications. L'approximation discrète des opérateurs de type divergence proposée

dans la section 3.3 et son lien avec les opérateurs du Calcul extérieur discrèt éveille notre intérêt dans cette direction. La convergence et l'unicité des schémas proposés sont à étudier. Un travail sur des méthodes de segmentation des images sur des surfaces qui utilise les EDPs proposées dans cette thèse a été commencé.

Pour des travaux futurs, il serait avantageux d'utiliser davantage des techniques du flot de Beltrami pour l'étape de régularisation dans des applications de segmentation telles que les cartes du cortex et le problème inverse de la MEG/EEG. Nous avons utilisé le flot de Beltrami dans l'étape de régularisation scalaire du problème des cartes rétinotopiques, mais il pourrait également être employé pour la régularisation du champ de vecteurs du gradient.

Pour le cas des images multi-valuées, nous avons couvert quelques aspects théoriques et présenté des implémentations pour le flot de Beltrami sur des images de couleur et de régularisation isotrope pour des images vectorielles. Il reste beaucoup de travail pour le cas des images de tenseurs, qui gagnent en intérêt en raison du nombre de plus en plus important de leurs applications dans la recherche du fonctionnement du cerveau. Des méthodes plus générales de régularisation d'image vectorielles doivent être développées.

Chapter 9

Author publications

Author's related publications

- N. Sochen, R. Deriche, L. Lopez-Perez, "The Beltrami flow over Implicit Manifolds", In proceedings of the International Conference in Computer Vision (ICCV): 832-839, October 2003, Nice.
- N. Sochen, R. Deriche, L. Lopez-Perez, "Variational Beltrami flows over Manifolds", In proceedings of the International Conference in Image Processing (ICIP): 861-864, September 2003, Barcelona.
- N. Sochen, R. Deriche, L. Lopez-Perez, "The Beltrami flow over Manifolds", INRIA Technical report No. 4897, July 2003.
- L. Lopez-Perez, N. Sochen, R. Deriche, "Beltrami flows over Triangulated Manifolds", In proceedings of the Computer Vision Approaches to Medical Image Analysis (CVAMIA) and Mathematical Methods in Biomedical Image Analysis (MMBIA) Workshop, European Conference in Computer Vision (ECCV): 135-144, May 2004 Prague.
- N. Wotawa, J. P. Pons, L. Lopez-Perez, R. Deriche, O. Faugeras, "fMRI data smoothing constrained to the cortical surface: a comparison of the level-set and mesh-based approaches", Neuroimage HBM, June 2004, Budapest.

Other publications from the author

- R. Martinez, L. Lopez-Perez, Nicolas Pasquier, Claude Pasquier, Martine Collard, "CGGA: An automatic tool for the interpretation of gene expression experiments", accepted to the Journal of Integrative Bioinformatics 2006 (JIB).
- R. Martinez, Nicolas Pasquier, Claude Pasquier, L. Lopez-Perez, "Interpreting Microarray Experiments Via Co-expressed Gene Groups Analysis", Discovery Science 2006: 316-320, from the Lecture Notes in Artificial Intelligence series, Springer-Verlag.
- R. Martinez, Nicolas Pasquier, Claude Pasquier, L. Lopez-Perez, Martine Collard, "Analyse des groupes de gènes co-exprimés (AGGC): un outil automatique pour l'interprétation des expériences de biopuces". Revue des Nouvelles Technologies d'Information 2006, RNTI-E-6.

Publications de l'auteur

Publications relatives à la thèse

- N. Sochen, R. Deriche, L. Lopez-Perez, "The Beltrami flow over Implicit Manifolds", In proceedings of the International Conference in Computer Vision (ICCV): 832-839, October 2003, Nice.
- N. Sochen, R. Deriche, L. Lopez-Perez, "Variational Beltrami flows over Manifolds", In proceedings of the International Conference in Image Processing (ICIP): 861-864, September 2003, Barcelona.
- N. Sochen, R. Deriche, L. Lopez-Perez, "The Beltrami flow over Manifolds", INRIA Technical report No. 4897, July 2003.
- L. Lopez-Perez, N. Sochen, R. Deriche, "Beltrami flows over Triangulated Manifolds", In proceedings of the Computer Vision Approaches to Medical Image Analysis (CVAMIA) and Mathematical Methods in Biomedical Image Analysis (MMBIA) Workshop, European Conference in Computer Vision (ECCV): 135-144, May 2004 Prague.
- N. Wotawa, J. P. Pons, L. Lopez-Perez, R. Deriche, O. Faugeras, "fMRI data smoothing constrained to the cortical surface: a comparison of the level-set and mesh-based approaches", Neuroimage HBM, June 2004, Budapest.

Autres publications de l'auteur

- R. Martinez, L. Lopez-Perez, Nicolas Pasquier, Claude Pasquier, Martine Collard, "CGGA: An automatic tool for the interpretation of gene expression experiments", accepted to the Journal of Integrative Bioinformatics 2006 (JIB).

- R. Martinez, Nicolas Pasquier, Claude Pasquier, L. Lopez-Perez, "Interpreting Microarray Experiments Via Co-expressed Gene Groups Analysis", Discovery Science 2006: 316-320, from the Lecture Notes in Artificial Intelligence series, Springer-Verlag.

- R. Martinez, Nicolas Pasquier, Claude Pasquier, L. Lopez-Perez, Martine Collard, "Analyse des groupes de gènes co-exprimés (AGGC): un outil automatique pour l'interprétation des expériences de biopuces". Revue des Nouvelles Technologies d'Information 2006, RNTI-E-6.

Chapter 10

Appendix

10.1 Stereographic direction diffusion

Refers to the derivation of 3.4.3, for more details see [87].

Every hypersphere S^n can be isometrically embedded in Re^{n+1}. The hypersphere is realized as the place of all the points in Re^{n+1} that satisfy $\sum_{i=1}^{n+1} U^i U^i = 1$. We denote by Y^i for $i = 1, \ldots, n$ the cartesian coordinate system on the subspace $\mathbb{R}^n$ that passes through the equator of S^n i.e. $\left\{\vec{U} \in \mathbb{R}^{n+1} | U^{n+1} = 0\right\}$. The stereographic transformation gives the values of Y^i as functions of the points on the north (south) hemispheres of the hypersphere. Explicitly it is given as

$$Y^i = \frac{U^i}{1 - U^{n+1}} \quad i = 1, \ldots, n.$$

Inverting these relations we find

$$\begin{aligned} U^i &= \frac{2Y^i}{1 + \sum_{i=1}^n Y^i} \quad i = 1, \ldots, n \\ U^{n+3} &= \frac{-1 + \sum_{i=2}^n Y^i}{1 + \sum_{i=2}^n Y^i} \end{aligned}$$

Now we can compute the induced metric of our feature space

$$g_{ij} = \sum_{k=1}^{n+1} \frac{\partial U^k \partial U^k}{\partial Y^i \partial Y^j} = \frac{4}{(1 + A)^2} \delta_{ij} \quad i, j = 1 \ldots, n,$$

where $A = \sum_{k=1}^{n} (Y^k)^2$.

10.2 Computation of the implicit isotropic smoothing PDE

Refers to the derivation of equation 2.3.2 (see [9]).

$$\begin{aligned}
\left.\frac{d}{dt}\right|_{t=0} E(u+t\mu) &= \frac{d}{dt}\int_\Omega \left(P_{\nabla\psi}\nabla(u+t\mu)\cdot P_{\nabla\psi}\nabla\mu\right)\delta(\psi)\,\|\nabla\psi\|\,dx \\
&= \int_\Omega \left(P_{\nabla\psi}\nabla u\cdot\left(\nabla\mu - \frac{\nabla\psi\cdot\nabla\mu}{\|\nabla\psi\|^2}\nabla\psi\right)\right)\delta(\psi)\,\|\nabla\psi\|\,dx \\
&= \int_\Omega \left(P_{\nabla\psi}\nabla u\cdot\nabla\mu\right)\delta(\psi)\,\|\nabla\psi\|\,dx \\
&\quad - \int_\Omega \left(P_{\nabla\psi}\nabla u\cdot\nabla\psi\right)\frac{\nabla\psi\cdot\nabla\mu}{\|\nabla\psi\|^2}\delta(\psi)\,\|\nabla\psi\|\,dx \\
&= \int_\Omega \left(P_{\nabla\psi}\nabla u\cdot\nabla\mu\right)\delta(\psi)\,\|\nabla\psi\|\,dx \\
&= -\int_\Omega \nabla\cdot\left(P_{\nabla\psi}\nabla u\;\|\nabla\psi\|\right)\delta(\psi)\,\mu dx \\
&\quad - \int_\Omega \left(P_{\nabla\psi}\nabla u\cdot\nabla\mu\right)\delta'(\psi)\,\|\nabla\psi\|\,\mu dx \\
&= -\int_\Omega \nabla\cdot\left(P_{\nabla\psi}\nabla u\;\|\nabla\psi\|\right)\delta(\psi)\,\mu dx \\
&= -\int_{\mathcal{S}} \frac{1}{\|\nabla\psi\|}\nabla\cdot\left(P_{\nabla\psi}\nabla u\;\|\nabla\psi\|\right)\mu d\mathcal{S}
\end{aligned}$$

10.3 Implicit isotropic smoothing for color images

Refers to the derivation of the equation 6.2.2, for more details see [9].

We consider u to be data of the form $u : S \longrightarrow S^{n-1}$. For $n = 3$ we have unite vector on the sphere: $u = (u_1, u_2, u_3), \|u\| = 1$.

We define the norm of the vectorial gradient of u :

$$\|\nabla u\| = \left(\|\nabla u_1\|^2 + \|\nabla u_2\|^2 + \|\nabla u_3\|^2\right)^{\frac{1}{2}},$$

and also the norm of the intrinsic vectorial gradient:

$$\|P_{\nabla\psi}\nabla u\| = \left(\|P_{\nabla\psi}\nabla u_1\|^2 + \|P_{\nabla\psi}\nabla u_2\|^2 + \|P_{\nabla\psi}\nabla u_3\|^2\right)^{\frac{1}{2}}.$$

We want to minimize the energy

$$E_A(u) := \frac{1}{2}\int_{\Omega\in\mathbb{R}^3} \|P_{\nabla\psi}\nabla u_1\|^2\,\delta(\psi)\,\|\nabla\psi\|\,dx,$$

with the constraint that u is of unit norm:

$$E_B(u) := \frac{1}{2}\gamma\int_{\Omega\in\mathbb{R}^3} (u^2-1)\delta(\psi)\,\|\nabla\psi\|\,dx,$$

so in practice we want to minimize the energy $E(u) = E_A(u) - E_B(u)$.

Applying to $E_A(u)$ the method described in Appendix A,

$$\begin{aligned}\frac{d}{dt}\,|_{t=0}\,E(u+t\mu) &= \int_\Omega (P_{\nabla\psi}\nabla u_1 \cdot P_{\nabla\psi}\nabla\mu_1)\delta(\psi)\,\|\nabla\psi\|\,dx \\ &\quad + \int_\Omega (P_{\nabla\psi}\nabla u_2 \cdot P_{\nabla\psi}\nabla\mu_2)\delta(\psi)\,\|\nabla\psi\|\,dx \\ &\quad + \int_\Omega (P_{\nabla\psi}\nabla u_3 \cdot P_{\nabla\psi}\nabla\mu_3)\delta(\psi)\,\|\nabla\psi\|\,dx\end{aligned}$$

we obtain that the gradient descent for E_A is given by

$$\frac{du_i}{dt} = \frac{1}{\|\nabla\psi\|}\nabla\cdot\left(P_{\nabla\psi}\nabla u_1\,\|\nabla\psi\|\right),$$

for each of the three components of u.

The gradient descent for E_B is simply

$$\frac{du_i}{dt} = -\gamma u_i.$$

So The composed gradient descent, for E, is

$$\frac{du_i}{dt} = \frac{1}{\|\nabla\psi\|}\nabla\cdot\left(P_{\nabla\psi}\nabla u_i\,\|\nabla\psi\|\right) - \gamma u_i.$$

We must find the value of γ. Multiplying both sides of the last equation by

u_i, making a summation over i and bearing in mind that $\|u\| = 1$, we get

$$\gamma = \frac{1}{\|\nabla\psi\|} \sum u_i \nabla \cdot \left(P_{\nabla\psi} \nabla u_i \, \|\nabla\psi\|\right).$$

Using the equalities

$$\begin{aligned}
u_i \nabla \cdot \left(P_{\nabla\psi} \nabla u_i \, \|\nabla\psi\|\right) &= \nabla \cdot \left(u_i P_{\nabla\psi} \nabla u_i \, \|\nabla\psi\|\right) - \nabla u_i P_{\nabla\psi} \nabla u_i \, \|\nabla\psi\|, \\
u_i P_{\nabla\psi} \nabla u_i &= u_i (\nabla u_i - \frac{\nabla\psi \nabla u_i}{\|\nabla\psi\|^2} \nabla\psi) = \frac{1}{2} (\nabla u_i^2 - \frac{\nabla\psi \nabla u_i^2}{\|\nabla\psi\|^2} \nabla\psi), \\
\nabla u_i \cdot P_{\nabla\psi} \nabla u_i \, \|\nabla\psi\| &= \|\nabla\psi\| \, \nabla u_i \cdot P_{\nabla\psi} \nabla u_i = \|\nabla\psi\| \, \|P_{\nabla\psi} \nabla u_i\|^2
\end{aligned}$$

and the fact that

$$\sum \nabla u_i^2 = \nabla \left(\sum u_i^2\right) = \nabla (1) = 0,$$

we finally obtain

$$\gamma = - \|P_{\nabla\psi} \nabla u\|^2$$

so the gradient descent for E is

$$\frac{du_i}{dt} = \frac{1}{\|\nabla\psi\|} \nabla \cdot \left(P_{\nabla\psi} \nabla u_1 \, \|\nabla\psi\|\right) - u \, \|P_{\nabla\psi} \nabla u\|^2 .$$

10.4 Numerical schemes for implicit regularization

We work in a cubic grid, with $\Delta x = \Delta y = \Delta z = 1$. We compute the value $u^n_{i,j,k}$, the value of u in the pixel (i, j, k) at the n^{th} iteration, based on the previous values of its neighbors, i.e. forward time differences.

First, we compute the vector $\vec{N}$ (this is needed only once), the outward normal to the surface by central differences,

$$\begin{aligned}
\vec{N}_{i,j,k} &= \nabla\psi_{i,j,k} \\
&= \frac{1}{2} \left(\psi_{i+1,j,k} - \psi_{i-1,j,k} \, , \, \psi_{i,j+1,k} - \psi_{i,j-1,k} \, , \, \psi_{i,j,k+1} - \psi_{i,j,k-1}\right)
\end{aligned}$$

Then, for the iteration n, we first compute the gradient of u

$$\vec{v}^n_{i,j,k} = \nabla_+ u^n_{i,j,k} = \left(u^n_{i+1,j,k} - u^n_{i,j,k} \, , \, u^n_{i,j+1,k} - u^n_{i,j,k} \, , \, u^n_{i,j,k+1} - u^n_{i,j,k}\right)$$

and its projection to the on the surface, to obtain the intrinsic gradient

$$\left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k} = \vec{v}^n_{i,j,k} - \left(\frac{\sum_{m=1}^{3} \vec{N}_{i,j,k}[m] \cdot \vec{v}^n_{i,j,k}[m]}{\left\|\vec{N}_{i,j,k}\right\|^2} \vec{N}_{i,j,k} \right)$$

where square brackets represent the component of the vector.

Finally, we use backward time differences to compute the divergence

$$\nabla_- \vec{w}_{i,j,k} = \vec{w}_{i,j,k}[1] - \vec{w}_{i-1,j,k}[1] + \vec{w}_{i,j,k}[2] - \vec{w}_{i,j-1,k}[3] + \vec{w}_{i,j,k}[3] - \vec{w}_{i,j,k-1}[3]$$

We switch forward differences and backward differences to avoid numerical problems.

Then the numerical implementation of the heat flow is

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t \left(\frac{1}{\left\|\vec{N}_{i,j,k}\right\|} \nabla_- \cdot \left(\left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k} \left\|\vec{N}_{i,j,k}\right\| \right) \right) \tag{10.1}$$

and for the embedding function being the distance function,

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t \left(\nabla_- \cdot \left(\left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k} \right) \right)$$

Implicit anisotropic diffusion

For the anisotropic diffusion, we found that using

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t \left(\frac{1}{\left\|\vec{N}_{i,j,k}\right\|} \nabla_- \cdot \left(\frac{\left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k}}{\left\|\left(P_{\vec{N}}\vec{v}\right)^n_{i,j,k}\right\|} \left\|\vec{N}_{i,j,k}\right\| \right) \right)$$

rather than the scheme proposed in [9] gives better results. This is illustrated in fig.

Numerical schemes for implicit color regularization

We use here the same notation as in 10.4, but now we have that u^n has three components: $u^n[m]$. For the isotropic diffusion, we apply the next scheme:

$$\begin{aligned} u^{n+1}_{i,j,k}[m] &= u^n_{i,j,k}[m] + \Delta t \frac{1}{\left\|\vec{N}_{i,j,k}\right\|} \nabla_- \left(\frac{\left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k}[m] \right) \right)^n_{i,j,k} \right\|}{\left\|\vec{N}_{i,j,k}\right\|} \right) \\ &\quad + u^n_{i,j,k}[m] \left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k} \right) \right)^n_{i,j,k} \right\|^2 \end{aligned}$$

where

$$\begin{aligned} \left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k} \right) \right)^n_{i,j,k} \right\| &= \Big(\left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k}[1] \right) \right)^n_{i,j,k} \right\| \\ &\quad + \left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k}[2] \right) \right)^n_{i,j,k} \right\| \\ &\quad + \left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k}[3] \right) \right)^n_{i,j,k} \right\| \Big)^{\frac{1}{2}}. \end{aligned}$$

And for the anisotropic scheme we have

$$\begin{aligned} u^{n+1}_{i,j,k}[m] &= u^n_{i,j,k}[m] \\ &\quad + \Delta t \frac{1}{\left\|\vec{N}_{i,j,k}\right\|} \nabla_- \left(\frac{\left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k}[m] \right) \right)^n_{i,j,k} \right\|}{\left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k} \right) \right)^n_{i,j,k} \right\|} \frac{1}{\left\|\vec{N}_{i,j,k}\right\|} \right) \\ &\quad + u^n_{i,j,k}[m] \left\| \left(P_{\vec{N}} \left(\nabla_+ u^n_{i,j,k} \right) \right)^n_{i,j,k} \right\| \end{aligned}$$

10.5 Algorithm for explicit anisotropic smoothing

Refers to the algorithm presented in section 3.2. Coded in C++, uses vtk, lapack and the yar++ image library from the Odyssée project.

```
/*
INPUT:
   "in": *.vtk vtkPolyData file containing a mesh and the
data related.
   "dt": time step (float).
   "n_time": number of steps (integer).
   "param": in (0,1,2), to choose between the parametric
isotropic diffusion (param=1), the FEM isotropic diffusion
(param=0) or the anisotropic diffusion (param=2).
   "mkmpg": choose mkmpg=1 to save all evolution images
in directory Seq/.

OUTPUT:
   "out": *.vtk vtkPolyData file containing the same mesh
as "in" but where the data has been regularized using the
parameters dt and n_time and the param method.

#include <math.h>
#include <Usage.H>
#include <Matrix.H>
#include <LinAlg.H>
#include "vtkPolyDataReader.h"
#include "vtkCellArray.h"
#include "vtkPolyData.h"
#include "vtkRenderer.h"
#include "vtkRenderWindow.h"
#include "vtkConeSource.h"
#include "vtkPolyDataMapper.h"
#include "vtkActor.h"
#include "vtkPolyDataWriter.h"
#include "vtkDataArray.h"
#include "vtkFloatArray.h"
#include "vtkPointData.h"
```

```
#include "vtkDataSetAttributes.h"

typedef Math::Matrix<double> MAT;

int main(int argc, char **argv) {

    char *in,*out;
    float dt;
    int n_time,param,mkmpg;
    Usage Call(argc,argv,"in out param mkmpg dt n_time",
    in,out,param,mkmpg,dt,n_time); Call();

// Options
    int normaliza = 0;
    int i;

// Read the polydata structure
    vtkPolyDataReader *pr = vtkPolyDataReader::New();
    pr ->  SetFileName(in);
    vtkPolyData  *p = vtkPolyData::New();
    p = pr -> GetOutput();

  vtkRenderer *ren2 = vtkRenderer::New();
  vtkRenderWindow *renWindow = vtkRenderWindow::New();
    renWindow->AddRenderer(ren2);
  vtkRenderer *ren1 = vtkRenderer::New();
    renWindow->AddRenderer(ren1);

// Read the polydata structure again

  vtkPolyDataReader *pro = vtkPolyDataReader::New();
    pro -> SetFileName(in);

  vtkPolyDataMapper *coneMapper = vtkPolyDataMapper::New();
    coneMapper->SetInput(p);
  vtkPolyDataMapper *coneMappero = vtkPolyDataMapper::New();
    coneMappero->SetInput(pro->GetOutput());

  vtkActor *coneActor = vtkActor::New();
    coneActor->SetMapper(coneMapper);
```

```
  vtkActor *coneActoro = vtkActor::New();
    coneActoro->SetMapper(coneMappero);

  // assign our actor to both renderers
  ren1->AddActor(coneActor);
  ren2->AddActor(coneActoro);
  // set the size of our window
  renWindow->SetSize(600,300);
  // set the viewports and background of the renderers
  ren1->SetViewport(0,0,0.5,1);
  ren1->SetBackground(0.2,0.3,0.5);
  ren2->SetViewport(0.5,0,1,1);
  ren2->SetBackground(0.2,0.5,0.3);
  renWindow->Render();

    char name[250];
    int ph = 0;

// Access the triangles

    p -> Update();
    p -> BuildCells();
    int n_points = p -> GetNumberOfPoints();
    vtkCellArray *tri = p -> GetPolys ();
    int n_tri = p -> GetNumberOfPolys();

// Connectivity info

    cout<<"Constructing the connectivity array"<<endl;

    const int max_n_nbr = 15;

    int j,k; //i,j contadores de puntos, k contador de vecinos
    int i_tri;
    int npts;
    int *pts;
    int yaesta;
    int nbr1[n_points][max_n_nbr];
    int m_nbr[n_points];
    //Inicializar en -1
```

```
for (i=0;i<n_points;i++) {
    m_nbr[i] = 0;
    for (j=0;j<max_n_nbr;j++)
        nbr1[i][j] = -1;
}


cout<<"# points:    "<<n_points<<endl;
cout<<"# triangles: "<<n_tri<<endl;

tri -> InitTraversal();

for(i_tri=0;tri -> GetNextCell(npts,pts)!=0;i_tri++)
    for(i=0;i<npts;i++)
        for(j = 0;j<3;j++)
            if (pts[j] != pts[i]) {
                yaesta = 0;
                // Ya esta pts[j] en nbr[pts[i]][.] o no?
                for (k=0;k<m_nbr[pts[i]];k++)
                    if (pts[j] == nbr1[pts[i]][k])
                     yaesta = 1;
                if (yaesta == 0) { // o sea que noesta
                    if (m_nbr[pts[i]] == max_n_nbr)
                        cout<<"Maximum surpassed"<<endl;
                    nbr1[pts[i]][m_nbr[pts[i]]] = pts[j];
                    m_nbr[pts[i]]++;
                }
            }

int max_nbr = 0;
for (i=0;i<n_points;i++)
    if (max_nbr < m_nbr[i]) max_nbr = m_nbr[i];

cout<<"Maximum number of neighbors: "<<max_nbr<<endl;
int nbr[n_points][max_nbr];
for(i=0;i<n_points;i++)
    for(j=0;j<max_nbr;j++)
        nbr[i][j] = nbr1[i][j];

cout<<"Reading Data"<<endl;
```

```
    float f[n_points];

    float * actf;
    actf = (float *) malloc (n_points * sizeof (float));
    if (actf == NULL) {printf ("out of memory\n"); exit (-1);}
    for (j = 0; j < n_points; j++) actf[j] = 0;

    const float FWHM = 4*sqrt(log(2.0)*n_time*dt);
    cout<<"FWHM = "<<FWHM<<endl;
    float lap;
    vtkDataArray *sca = p -> GetPointData() -> GetScalars();
    int t;

    for(i=0;i<n_points;i++) f[i] = sca -> GetComponent(i,0);


/*************************************************************
*************************************************************
***    WEIGHTS COMPUTATION FOR THE Laplace-Beltrami   *******
*************************************************************
*************************************************************/

    cout<<"Calculating the linear weights for the LB"<<endl;
    MAT weight(n_points,max_nbr);

    if (param == 1) {

// Parametric method


    float *pt,*nbr_pt,*normal;
    float deno;
    int n_nbr;

    for (i=0;i<n_points;i++) {
        n_nbr = m_nbr[i];
        MAT coord_nbr(3,n_nbr);
        //Translate the origin
        pt = p -> GetPoints() -> GetPoint(i);
```

```
for (j=0;j<m_nbr[i];j++){
    nbr_pt = p -> GetPoints() -> GetPoint(nbr[i][j]);
    for (k=0;k<3;k++)
        coord_nbr(k+1,j+1) =  nbr_pt[k]-pt[k];
}
//Rotation by Eulerian Axis
MAT Q(Identity(3));
normal = p->GetPointData()->GetNormals()->GetTuple(i);
deno = sqrt(normal[0]*normal[0]+normal[1]*normal[1]);

if (deno >= 0.1) {
    MAT Q1(3,3);
    Q1(1,1)=normal[0]/deno;
    Q1(1,2)=normal[1]/deno;
    Q1(1,3)=0.0;
    Q1(2,1)=-normal[1]/deno;
    Q1(2,2)=normal[0]/deno;
    Q1(2,3)=0.0;
    Q1(3,1)=0.0;
    Q1(3,2)=0.0;
    Q1(3,3)=1.0;

    MAT Q2(3,3);
    Q2(1,1)=normal[2];
    Q2(1,2)=0.0;    Q2(1,3)=-deno;
    Q2(2,1)=0.0;
    Q2(2,2)=1.0;    Q2(2,3)=0.0;
    Q2(3,1)=deno;
    Q2(3,2)=0.0;    Q2(3,3)=normal[2];

    Q = Q2*Q1;
}

MAT rot = Q*coord_nbr;
MAT X(n_nbr,5);
MAT z(n_nbr,1);

// Contruction of X, the design matrix and z
for (k=1;k<=n_nbr;k++) {
    X(k,1) = rot(1,k);
```

```
    X(k,2) = rot(2,k);
    X(k,3) = rot(1,k)*rot(1,k);
    X(k,4) = 2*rot(1,k)*rot(2,k);
    X(k,5) = rot(2,k)*rot(2,k);
    z(k,1) = rot(3,k);
}

Math::LinearAlgebra<double> LinAlg;
MAT beta;

if (n_nbr >= 5) {
    beta =
    (LinAlg.Inverse(Transpose(X)*X)*Transpose(X))*z;
    beta = LinAlg.SolveLinear(X,z);
}

else {

       MAT beta2(5,1);
       beta2(1,1) = 0;
       beta2(2,1) = 0;
       beta2(3,1) = 0;
       beta2(4,1) = 0;
       beta2(5,1) = 0;
       beta = beta2;
    MAT X2(n_nbr,n_nbr);
    for (k=1;k<=n_nbr;k++)
        for (j=1;j<=n_nbr;j++)
            X2(k,j)=X(k,j);
    beta = (LinAlg.Inverse(Transpose(X2)*X2)
    *Transpose(X2))*z;

}

MAT g(2,2);
g(1,1) = 1 + beta(1,1)*beta(1,1);
g(1,2) = beta(1,1)*beta(2,1);
g(2,1) =  beta(1,1)*beta(2,1);
g(2,2) =  1 + beta(2,1)*beta(2,1);
```

```
    MAT g_half; // principal sqroot of g
    MAT D_vect,VR;
    D_vect = LinAlg.eig(g,VR); //(2X1)
    MAT D(2,2);

    D(1,1) = sqrt(fabs(D_vect(1)));
    D(1,2) = 0.0;
    D(2,1) = 0.0;
    D(2,2) = sqrt(fabs(D_vect(2)));

    g_half =
    LinAlg.SolveLinear(Transpose(VR),D*Transpose(VR));

    MAT temp1(2,n_nbr);
    for (k=1;k<=n_nbr;k++) {
        temp1(1,k) = rot(1,k);
        temp1(2,k) = rot(2,k);
    }
    MAT conformal_coord = g_half*temp1;
    MAT X_act (n_nbr,5);
    for (k=1;k<=n_nbr;k++) {
        const double x_aux = conformal_coord(1,k);
        const double y_aux = conformal_coord(2,k);
        X_act(k,1) = x_aux;
        X_act(k,2) = y_aux;
        X_act(k,3) = 0.5*x_aux*x_aux;
        X_act(k,4) = (x_aux)*(y_aux);
        X_act(k,5) = 0.5*y_aux*y_aux;
    }
    MAT gamma = LinAlg.PseudoInverse(X_act);
    // gamma is 5xn_nbr
    // Saving the weigths
    for (k=1;k<=n_nbr;k++)
        weight(i+1,k) = gamma(3,k)+gamma(5,k);

    if (max_nbr > n_nbr)
        for (k=n_nbr+1;k<=max_nbr;k++)
            weight(i+1,k) = 0.0;
}
```

```
}

if ((param == 0) || (param == 2)) {

// FEM

    cout<<"Ordering neighbors"<<endl;

    int n_nbr;
    //int n_pts;
    short unsigned int n_cells;
    int *tri_ady1;
    int *tri_ady2;
    int *tris1;
    int flag1,flag2,flag3,flag4;
    p -> BuildLinks();
    for (i=0;i<n_points;i++) {
        n_nbr = m_nbr[i];
        for (int k_nbr=0;k_nbr<n_nbr-1;k_nbr++) {
            // For each neighbor look for triangles
            // that contain point i and k_nbr
            p -> GetPointCells(i,n_cells,tris1);
            // check on triangles neighboring point i
            flag3 = 0; flag4 = 0;
            tri -> InitTraversal();
            for(i_tri=0;(i_tri<n_cells)
                        &&(flag4==0);i_tri++)
            {
              p -> GetCellPoints(tris1[i_tri],npts,pts);
              flag1 = 0; flag2 = 0;
              for(j=0;j<npts;j++)
                  if (i == pts[j]) flag1 = 1;
                  else if (nbr[i][k_nbr] == pts[j])
                       flag2 = 1;
              if ((flag1 == 1) && (flag2 == 1))
              // The triangle has both
                  if (flag3 == 0) {
                  // Is the first triangle it finds
                      tri_ady1 = pts;
                      flag3 = 1; }
```

```
      else {
      // Is the second (there can't be
      // more than 2 triangles adyacents
      // to a vertex)
          tri_ady2 = pts;
          flag4 = 1;
      }
}
int nei = 0;
if (flag3 == 0) // unclosed surface
    cout<<"no tri for the pair "
        <<i<<","<<nbr[i][k_nbr]<<"!"<<endl;
else if (flag4 == 0)
    cout<<"unclosed surface!"
        <<i<<","<<nbr[i][k_nbr]<<endl;

else {
    for (int k1=1;k1<=3;k1++)
    for (int k2=1;k2<=3;k2++)
        if ((tri_ady1[k1-1]==i) &&
            (tri_ady1[k2-1]==nbr[i][k_nbr]))
            if (((k1-k2)==-1)||((k1-k2)==2))
            {
                // is tri_a, neighbor is
                int k3;
                if ((k1!=1) && (k2!=1))
                 k3 = 1;
                else if ((k1!=2) & (k2!=2))
                 k3 = 2;
                else k3 = 3;
                nei = tri_ady1[k3-1]; }
    if (nei == 0)
        for (int k1=1;k1<=3;k1++)
        for (int k2=1;k2<=3;k2++)
            if ((tri_ady2[k1-1]==i) &&
            (tri_ady2[k2-1]==nbr[i][k_nbr]))
                {
                int k3;
                if ((k1!=1) & (k2!=1))
                 k3 = 1;
```

```
                        else if ((k1!=2) & (k2!=2))
                         k3 = 2;
                        else k3 = 3;
                        nei = tri_ady2[k3-1]; }
        }
        nbr[i][k_nbr+1]=nei;
    }
}

// Ordering finished
cout<<"Cotan computation"<<endl;

float *pt;
float *nbr_pt;

for (i=0;i<n_points;i++) {
    n_nbr = m_nbr[i];
    MAT cotan(1,n_nbr);
    MAT coord_nbr1(3,n_nbr);
    //Translate the origin
    pt = p -> GetPoints() -> GetPoint(i);
    for (j=0;j<n_nbr;j++) {
      nbr_pt = p->GetPoints()->GetPoint(nbr[i][j]);
      for (k=0;k<3;k++)
            coord_nbr1(k+1,j+1) =  nbr_pt[k]-pt[k];
    }
    MAT coord_nbr(3,n_nbr+2);
    for (k=1;k<=3;k++) { //circular array..
        coord_nbr(k,1) = coord_nbr1(k,n_nbr);
        for (j=1;j<=n_nbr;j++)
            coord_nbr(k,j+1) = coord_nbr1(k,j);
        coord_nbr(k,n_nbr+2) = coord_nbr1(k,1);
    }

    float area = 0;
    for (int l=2;l<=n_nbr+1;l++) {
        MAT a1 = coord_nbr.GetSubMatrix(1,l,3,l);
        //xi
        MAT a2 = coord_nbr.GetSubMatrix(1,l+1,3,l+1);
        //xj
```

```
            MAT a3 = coord_nbr.GetSubMatrix(1,l-1,3,l-1);
            //xk
            MAT a4 = a2 - a1; // xj - xi
            MAT a5 = a3 - a1; // xk - xi
            MAT p = CrossProd(a2,a4);
            MAT q = CrossProd(a3,a5);
            const double area_p = sqrt(DotProd(p,p))/2;
            const double area_q = sqrt(DotProd(q,q))/2;
            const double dot_p = DotProd(a2,a4);
            const double dot_q = DotProd(a3,a5);
            if (param==2) {
                MAT an = ((f[l]-f[l-1])*a4
                 + (f[l+1]-f[l])*a5);
                const double normag = an.Norm();
                if (normag>0)
                    cotan(1,l-1) = (dot_p/(2*area_p) +
                              dot_q/(2*area_q))/normag;
                else
                    cotan(1,l-1)=0;
            }
            else {
                if (area_p*area_q > 0)
                    cotan(1,l-1) = dot_p/(2*area_p) +
                                   dot_q/(2*area_q);
                else
                    cotan(1,l-1)=0;
                }
            area+=area_p;
        }

        for (j=1;j<=max_nbr;j++)
            if (j<n_nbr)
                weight(i+1,j) = cotan(1,j)/area;
            else
                weight(i+1,j) = 0;

    }
}
```

```
/************************************************************
************************************************************
***************   FINITE DIFFERENCES ***********************
************************************************************
************************************************************/

    cout<<"Finite Differences scheme"<<endl;

    for (t=0;t<n_time;t++) {

        if ((param==2)&&(t>0))  {
        // WEIGHTS ACTUALIZATION
        float *pt;
        float *nbr_pt;
        for (i=0;i<n_points;i++) {
            int n_nbr = m_nbr[i];
            MAT cotan(1,n_nbr);
            MAT coord_nbr1(3,n_nbr);
            //Translate the origin
            pt = p -> GetPoints() -> GetPoint(i);
            for (j=0;j<n_nbr;j++) {
                nbr_pt = p->GetPoints()->GetPoint(nbr[i][j]);
                for (k=0;k<3;k++)
                    coord_nbr1(k+1,j+1) =  nbr_pt[k]-pt[k];
            }
            MAT coord_nbr(3,n_nbr+2);
            for (k=1;k<=3;k++) { //circular array..
                coord_nbr(k,1) = coord_nbr1(k,n_nbr);
                for (j=1;j<=n_nbr;j++)
                    coord_nbr(k,j+1) = coord_nbr1(k,j);
                coord_nbr(k,n_nbr+2) = coord_nbr1(k,1);
            }
            float area = 0;
            for (int l=2;l<=n_nbr+1;l++) {
                MAT a1 = coord_nbr.GetSubMatrix(1,l,3,l);
                //xi
                MAT a2 = coord_nbr.GetSubMatrix(1,l+1,3,l+1);
                //xj
                MAT a3 = coord_nbr.GetSubMatrix(1,l-1,3,l-1);
                //xk
```

```
        MAT a4 = a2 - a1; // xj - xi
        MAT a5 = a3 - a1; // xk - xi
        MAT p = CrossProd(a2,a4);
        MAT q = CrossProd(a3,a5);
        const double area_p = sqrt(DotProd(p,p))/2;
        const double area_q = sqrt(DotProd(q,q))/2;
        const double dot_p = DotProd(a2,a4);
        const double dot_q = DotProd(a3,a5);
        if (param==2) {
            MAT an = ((f[l]-f[l-1])*a4 +
                      (f[l+1]-f[l])*a5);
            const double normag = an.Norm();
            cotan(1,l-1)=(dot_p/(2*area_p) +
                          dot_q/(2*area_q))/normag;
        }
        else
            cotan(1,l-1)=dot_p/(2*area_p)
                        +dot_q/(2*area_q);
        area+=area_p;
    }

    for (j=1;j<=max_nbr;j++)
        if (j<n_nbr)
            weight(i+1,j) = cotan(1,j)/area;
        else
            weight(i+1,j) = 0;
} }
// END WEIGHT ACTUALIZATION

for (j=0;j<n_points;j++) {
    lap = 0;
    for (k=0;k<m_nbr[j];k++)
        lap += weight(j+1,k+1)*(f[nbr[j][k]] - f[j]);
    actf[j] = f[j] + dt*lap;
}



float * minf;
minf = (float *) malloc (n_points * sizeof (float));
```

```
if (minf == NULL)
 {printf ("out of memory\n"); exit (-1);}
for (j = 0; j < n_points; j++) minf[j] = 1000000.0;

float * maxf;
maxf = (float *) malloc (n_points * sizeof (float));
if (maxf == NULL)
 {printf ("out of memory\n"); exit (-1);}
for (j = 0; j < n_points; j++) maxf[j] = 1000000.0;


for (j=0;j<n_points;j++) {
    minf[j]=1000000.0;
    maxf[j]=-1000000.0;
}
for (j=0;j<n_points;j++)
    for (k=0;k<m_nbr[j];k++) {
        if (minf[j] > f[nbr[j][k]])
            minf[j] = f[nbr[j][k]];
        if (maxf[j] < f[nbr[j][k]])
            maxf[j] = f[nbr[j][k]];
    }
for (j=0;j<n_points;j++) {
    if (actf[j] < minf[j]) actf[j] = minf[j];
    if (actf[j] > maxf[j]) actf[j] = maxf[j];
    f[j] = actf[j];
    sca -> SetComponent(j,0,f[j]);
}
p -> GetPointData() -> SetScalars(sca);

// Visualization
renWindow -> RemoveRenderer(ren2);
ren2 -> Delete();
vtkRenderer *ren2 = vtkRenderer::New();
renWindow -> AddRenderer(ren2);
ren2 -> AddActor(coneActoro);

renWindow -> RemoveRenderer(ren1);
ren1 -> Delete();
vtkRenderer *ren1 = vtkRenderer::New();
```

```
    renWindow -> AddRenderer(ren1);
    ren1 -> AddActor(coneActor);

    ren1->SetViewport(0.5,0,1,1);
    ren1->SetBackground(0.2,0.5,0.3);
    ren2->SetViewport(0,0,0.5,1);
    ren2->SetBackground(0.2,0.3,0.5);

    renWindow -> Render();

    // END Visualization

    cout<<t+1<<endl;
    if ( mkmpg == 1 ) {
        sprintf(name,"Seq/imagen\%03d.ppm",t+ph);
        //renWindow->SetFileName(name);
        renWindow->MappedOn();
        //renWindow->SaveImageAsPPM();
    }
}

ph += t;

if (normaliza == 1) {
    float maxf = -1000000.0;
    float minf = 1000000.0;
    for (i=0;i<n_points;i++) {
        if (maxf<f[i]) maxf = f[i];
        if (minf>f[i]) minf = f[i];
    }
    if (maxf-minf>0.0001)
        for (i=0;i<n_points;i++)
            f[i]=255.0*(f[i]-minf)/(maxf - minf);
    for (j=0;j<n_points;j++)
        sca -> SetComponent(j,0,f[j]);
}

cout<<"Saving result"<<endl;
vtkPolyDataWriter *pw = vtkPolyDataWriter::New();
pw -> SetInput(p);
```

```
    pw -> SetFileName(out);
    pw -> Write();

  // Clean up
  p -> Delete();
  ren1->Delete();
  ren2->Delete();
  renWindow->Delete();
  pr->Delete();
  pro->Delete();
  coneMapper->Delete();
  coneMappero->Delete();
  coneActor->Delete();
  coneActoro->Delete();

}
```

Bibliography

[1] G. Adde. *Méthodes de Traitement d'Image Appliquées au Problème Inverse en Magnéto-Electro-Encéphalographie*. PhD thesis, École Nationale des Ponts et Chaussées, 2005.

[2] M. Adde, G. Clerc, , and R. Keriven. Imaging methods formeg/eeg inverse problem. In *Joint Meeting of the 5th International Conference on Bioelectromagnetism and the 5th International Symposium on Noninvasive Functional Source Imaging*, Minneapolis, USA, May 2005.

[3] Luis Alvarez. Images and PDE's. In *In proceedings of the 12th International Conference on analysis and optimization of Systems. Images, Wavelets and PDE's.*, volume 219 of *Lecture Notes in Control and Information Sciences*, pages 3–15. Springer Verlag, june 1996.

[4] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel. Image selective smoothing and edge detection by nonlinear diffusion II. *SIAM Journal on Numerical Analysis*, 29(3):845–866, june 1992.

[5] A. Andrade, F. Kherif, J.-F. Mangin, K. Worsley, A.-L. Paradis, O. Simon, S. Dehaene, and J.-B. Poline. Detection of fMRI activation using cortical surface mapping. *Human Brain Mapping*, 12:79–93, 2001.

[6] Chandrajit L. Bajaj and Guoliang Xu. Anisotropic diffusion of surfaces and functions on surfaces. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 4–32, New York, NY, USA, 2003. ACM Press.

[7] E. Bansch and K. Mikula. Adaptivity in 3d image processing (manuscript). Manuscript, 2001.

[8] M. Bertalmío. *Processing of flat and non-flat image information on arbitrary manifolds using partial differential equations*. PhD thesis, University of Minnesota, 2001.

[9] M. Bertalmío, L. T. Cheng, S. Osher, and G. Sapiro. A framework for solving surface partial differential equations for computer graphics applications. Technical Report 00-23, UCLA Research Report, 2000.

[10] M. Bertalmío, L. T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *Journal of Computational Physics*, 174:759–780, December 2001.

[11] David E. Breen and Ross T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.

[12] Paul Burchard, Li-Tien Cheng, Barry Merriman, and Stanley Osher. Motion of curves in three spatial dimensions using a level set approach. Technical Report 00-32, Department of Mathematics, UCLA, September 2000.

[13] L. M. Chalupa and J.S. Werner, editors. *The visual neurosciences*. MIT Press, 2004.

[14] Tony Chan and Jianhong Shen. Variational restoration of nonflat image features: Models and algorithms. *SIAM Journal on Applied Mathematics*, 61(4):1338–1361, 2001.

[15] C. Chefd'hotel, D. Tchumperle, R. Deriche, and O. Faugeras. Constrained flows of matrix-valued functions: Application to diffusion tensor regularization. In *Proceedings of 7th European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.

[16] S. Chen, B. Merriman, S. Osher, and P. Smereka. A simple level set method for solving stefan problems. *Journal of Computational Physics*, 135:8, 1995.

[17] Li-Tien Cheng. *The Level Set Method Applied to Geometrically Based Motion, Materials Science, and Image Processing*. PhD thesis, Caltech, 2000.

[18] D. Chopp and J. Sethian. Motion by intrinsic laplacian of curvature. In *Interfaces and Free Boundaries*, page 1:18, 1999.

[19] David L. Chopp. Computing minimal surfaces via level set curvature flow. *Journal of Computational Physics*, 106(1):77–91, May 1993.

[20] M. K. Chung, J. Taylor, K. J. Worsley, J. O. Ramsay, S. Robbins, and A. Evans. Diffusion smoothing on the cortical surface via the laplace-beltrami operator. In *IEEE Transactions on Medical Imaging*, 2000.

[21] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings Visualization 2000*, pages 397–405. IEEE Computer Society Technical Committee on Computer Graphics, 2000.

[22] U. Clarenz, U. Diewald, and M. Rumpf. Processing textured surfaces via anisotropic geometric diffusion. In *IEEE Transactions on Image Processing*, volume 13, pages 248–261, 2004.

[23] Eric Debreuve, Michel Barlaud, Gilles Aubert, Ivan Laurette, and Jacques Darcourt. Space-time segmentation using level set active contours applied to myocardial gated spect. *IEEE Transactions on Medical Imaging*, 20(7):643–659, 2001.

[24] Rachid Deriche, Christophe Bouvin, and Olivier D. Faugeras. Front propagation and level-set approach for geodesic active stereovision. In *Third Asian Conference On Computer Vision (ACCV)*, pages 640–647, Bombay, India, January 1998.

[25] Rachid Deriche and Olivier Faugeras. Les EDP en traitement des images et vision par ordinateur. Technical Report 2697, INRIA Sophia Antipolis, Novembre 1995.

[26] A. Dervieux and F. Thomasset. *A finite element method for the simulation of Rayleigh-Taylor instability.* Number 771 in Lecture Notes in Mathematics. Springer Verlarg, 1979.

[27] A. Dervieux and F. Thomasset. Multifluid incompressible flows by a finite element method. In *Seventh International Conference on Numerical Methods in Fluid Dynamics*, volume 141 of *Lecture Notes in Physics*, pages 158–163. Stanford University, Stanford, California and NASA/Ames (U.S.A.), 1981.

[28] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. In *Proceedings of the Computer Graphics Forum*, volume 21, 2002.

[29] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Graphics Interface*, 2000.

[30] M. Desbrun, M. Meyer, P. Shroeder, and A. H. Barr. Discrete differential-geometry operators in nD. In *Visualization and Mathematics*, 2003.

[31] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, 33(Annual Conference Series):317–324, 1999.

[32] E. DeYoe, G. Carman, P. Bandettini, et al. Mapping striate and extrastriate visual areas in human cerebral cortex. *Neurobiology*, 93:2382–2386, March 1996.

[33] U. Diewald, T. Preußer, and M. Rumpf. Anisotropic diffusion in vector field visualization on euclidean domains and surfaces. In *IEEE Transactions on Visualization and Computer Graphics*, volume 6, pages 139–149, 2000.

[34] M. P. Do Carmo. *Riemannian Geometry*. Birk, Boston, USA, 1992.

[35] S. Engel, D. Rumelhart, B. Wandell, A. Lee, G. Glober, E-J. Chichilnisky, and M. Shadlen. fMRI of human visual cortex. *Nature*, 369:525–529, June 1994.

[36] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. In *IEEE Transactions on Image Processing*, volume 7, pages 336–344, 1998.

[37] Olivier Faugeras, Geoffray Adde, Guillaume Charpiat, Christophe Chefd'Hotel, Maureen Clerc, Thomas Deneux, Rachid Deriche, Gerardo Hermosillo, Renaud Keriven, Pierre Kornprobst, Jan Kybic, Christophe Lenglet, Lucero Lopez-Perez, Théo Papadopoulo, Jean-Philippe Pons, Florent Segonne, Bertrand Thirion, David Tschumperlé, Thierry Viéville, and Nicolas Wotawa. Variational, geometric, and statistical methods for modeling brain anatomy and function. *Neuroimage*, 23S1:S46–S55, 2004.

[38] Greiner. G. Variational design and fairing of spline surface. In *Proceedings of the Computer Graphics Forum*, volume 13, pages 143–154, 1994.

[39] Rafael C. Gonzales and Richard E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, New Jersey 07458, 2002.

[40] G. Hermosillo, O. Faugeras, and J. Gomes. Unfolding the cerebral cortex using level set methods. In *Proceedings of the 2nd International Conference on Scale-Space Theories in Computer Vision (SCALE-SPACE '99)*, pages 58–69, London, UK, 1999. Springer-Verlag.

[41] A. N. Hirani. *Discrete Exterior Calculus*. PhD thesis, Caltech, 2003.

[42] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93)*, pages 19–26, New York, NY, USA, 1993. ACM Press.

[43] Ben Houston, Mark Wiebe, and Christopher Batty. RLE sparse level sets. In *Proceedings of the SIGGRAPH '04 Conference on Sketches & Applications*, Los Angeles, California, 2004. ACM Press.

[44] Andreas Hubeli and Markus Gross. Fairing of non-manifolds for visualization. In *Proceedings of the Conference on Visualization (VIS '00)*, pages 407–414, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[45] Stéphanie Jehan-Besson, Michel Barlaud, and Gilles Aubert. Detection and tracking of moving objects using a new level set based method. In *ICPR*, pages 7112–7117, 2000.

[46] E.R. Kandel, J.H. Schwartz, and T.M. Jessel. *Principles of Neural Science*. McGraw-Hill, 4th edition, 2000.

[47] R. Kimmel, R. Malladi, and N. Sochen. Images as embedding maps and minimal surfaces: Movies, color, texture, and volumetric medical images. *International Journal of Computer Vision*, 39(2):111–129, 2000.

[48] Ron Kimmel. Intrinsic scale space for images on surfaces: The geodesic curvature flow. *Graphical models and image processing*, 59(5):365–372, 1997.

[49] L. Kobbelt, T. Hesse, H. Prautzsch, and Schweizerhof K. Iterative mesh generation for fe-computations on free form surfaces. *Engineering Computations*, 14, 1997.

[50] Leif Kobbelt. Discrete fairing. In T. Goodman and R. Martin, editors, *In The Mathematics of Surfaces VII*, pages 101–129. Information Geometers, 1996.

[51] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114, New York, NY, USA, 1998. ACM Press.

[52] P. Kornprobst, R. Deriche, and G. Aubert. Nonlinear operators in image restoration. In *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 325–331, San Juan, Puerto Rico, June 1997. IEEE Computer Society.

[53] N. Logothetis. Vision: A window on consciousness. *Scientific American*, November 1999.

[54] L. Lopez-Perez, R. Deriche, and N. Sochen. The beltrami flow over triangulated manifolds. In *Proceedings of the European Conference on Computer Vision*, pages 135–144, 2004.

[55] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[56] J. L. Mallet. Discrete smooth interpolation in geometric modelling. *Computer Aided Design*, 24(4):178–191, 1992.

[57] Sean Mauch. *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology, 2003.

[58] F. Memoli, G. Sapiro, and S. Osher. Solving variational problems and partial differential equations mapping into general target manifolds. *Journal of Computational Physics*, 195:263–292, 2004.

[59] M. Meyer, M. Desbrun, P. Schroder, and Barr A. H. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, 2003.

[60] Henry P. Moreton and Carlo H. Squin. Functional optimization for fair surface design. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 167–176, New York, NY, USA, 1992. ACM Press.

[61] Paragios Nikos and Deriche Rachid. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280, 2000.

[62] Paragios Nikos and Deriche Rachid. Geodesic active regions and level set methods for supervised texture segmentation. *International Journal of Computer Vision*, 46(3):223–247, 2002.

[63] Paragios Nikos and Deriche Rachid. Geodesic active regions and level set methods for motion estimation and tracking. *Computer Vision and Image Understanding*, 97(3):259–282, 2005.

[64] M. Ortner, X. Descombes, and J. Zerubia. Building outline extraction from digital elevation models using marked point processes. *International Journal of Computer Vision*, 2006.

[65] S. Osher and J.A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *Jounal of Computational Physics*, 79:12–49, 1988.

[66] S. J. Osher and R.P. Fedkiw. Level set methods. Technical Report 00-07, UCLA, Mathematics Department, 2000.

[67] S. Palmer. *Vision Science : Photons to Phenomenology.* MIT Press, 1999.

[68] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[69] A. M. Polyakov. Quantum geometry of bosonic strings. *Physics Letters B*, 103:207–210, jul 1981.

[70] J. P. Pons. *Methodological and applied contributions to the deformable models framework.* PhD thesis, Ecole Nationale des Ponts et Chausses, 2005.

[71] J. P. Pons, G. Hermosillo, and O. Faugeras. Maintaining the point correspondence in the level set framework. *Journal of Computational Physics*, 2006.

[72] Jean-Philippe Pons, Gerardo Hermosillo, Renaud Keriven, and Olivier Faugeras. How to deal with point correspondences and tangential velocities in the level set framework. In *Proceedings of the International Conference on Computer Vision*, 2003.

[73] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[74] T. Preußer and M. Rumpf. An adaptive finite element method for large scale image processing. *Journal of Visual Communication and Image Representation*, 11:183–195, 2000.

[75] T. Preußer and M. Rumpf. A level set method for anisotropic geometric diffusion in 3D image processing. *SIAM Journal on Applied Mathematics*, 62(5):1772–1793, 2002.

[76] Bruno Rodrigues de Araujo and Joaquim Armando Pires Jorge. Curvature dependent polygonization of implicit surfaces. In *Proceedings of the XVII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'04)*, pages 266–273. IEEE Computer Society, 2004.

[77] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[78] Nicholas S. Sapidis. *Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994.

[79] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis.* Cambridge University Press, 2001.

[80] M.I. Sereno, A.M. Dale, J.B. Reppas, K.K. Kwong, J.W. Belliveau, T.J. Brady, B.R. Rosen, and R.B.H. Tootell. Borders of multiple visual areas in human revealed by functional magnetic resonance imaging. *Science*, pages 889–893, 1995.

[81] N. Sochen, R. Deriche, and L. Lopez-Perez. The beltrami flow over implicit manifolds. In *Proceedings of the International Conference on Computer Vision*, pages 832–839, 2003.

[82] N. Sochen, R. Deriche, and L. Lopez-Perez. The beltrami flow over manifolds. Technical Report 4897, INRIA, June 2003.

[83] N. Sochen, R. Deriche, and L. Lopez-Perez. Variational beltrami flows over manifolds". In *Proceedings of the International Conference on Image Processing*, pages 861–864, 2003.

[84] N. Sochen and R. Kimmel. Combing a porcupine via stereographic direction diffusion. In *In proceedings of the 4th Int. Conf. on Scale-Space*, Vancouver, Canada, 2001.

[85] N. Sochen, R. Kimmel, and R. Malladi. From high energy physics to low level vision. In *SCALE-SPACE '97: Proceedings of the First International Conference on Scale-Space Theory in Computer Vision*, pages 236–247, London, UK, 1997. Springer-Verlag.

[86] N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. In *IEEE Tran. on Image Processing*, volume 7, pages 310–318, 1998.

[87] N. Sochen, C. Sagiv, and R. Kimmel. Stereographic combing a porcupine or studies on orientation diffusion. *SIAM Journal on Applied Mathematics*, 64:1477–1508, 2004.

[88] N. Sochen and Y. Y. Zeevi. Representation of images by surfaces embedded in higher dimensional non-Euclidean space. In *International Conference on Mathematical Methods for Curves and Surfaces*, Lillehamer, Norway, 1997.

[89] N. Sochen and Y. Y. Zeevi. Representation of colored images by manifolds embedded in higher dimensional non-Euclidean space. In *International Conference on Ima Processing*, Chicago, 1998, 1998. IEEE.

[90] A. Spira and R. Kimmel. Enhancement of images painted on manifolds. In *Scale Space Methods in Computer Vison*, Lecture Notes in Computer Science, pages 492–502, Hofgeismar, Germany, April 2005. Springer-Verlag.

[91] A. Spira, R. Kimmel, and N. Sochen. Efficient beltrami flow using a short time kernel. In *Scale Space Methods in Computer Vison*, Lecture

Notes in Computer Science, pages 511–522, Isle of Skye, Scotland, UK, June 2003.

[92] Alon Spira. *Geometric Image Evolution on Parametric Surfaces*. PhD thesis, Israel Institute of Technology, 2005.

[93] B. Tang, G. Sapiro, and V. Caselles. Diffusion of general data on non-flat manifolds via harmonic maps theory : The direction diffusion case. *International Journal on Computer Vision*, 36(2):149–161, February 2000.

[94] Tolga Tasdizen and Ross Whitaker. Anisotropic diffusion of surface normals for feature preserving surface reconstruction. Technical Report UUCS-03-007, University of Utah, School of Computing, 2003.

[95] Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1995. ACM Press.

[96] D. Tschumperle. *PDE-Based Regularization of Multivalued Images and Applications*. PhD thesis, University of Nice-Sophia Antipolis, 2002.

[97] D. Tschumperle and R. Deriche. Orthonormal vector sets regularization with pde's and aplications. *International Journal on Computer Vision, Special Issue VLSM*, 50(3):237–252, 2002.

[98] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. In *IEEE Transactions on Automatic Control*, pages 1528–1538, 1995.

[99] J. Warnking, M. Dojat, et al. fMRI retinotopic mapping - step by step. *NeuroImage*, 17:1665–1683, 2002.

[100] Joachim Weickert. *Anisotropic Diffusion in Image Processing*. Teubner-Verlag, Stuttgart, Germany, Esta igual y quitarla, el libro ya ni esta en venta 1998.

[101] William Welch and Andrew Witkin. Variational surface modeling. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 157–166, New York, NY, USA, 1992. ACM Press.

[102] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 247–256, New York, NY, USA, 1994. ACM Press.

[103] N. Wotawa. *Low-level visual cortex and motion perception: an MRI study*. PhD thesis, University of Nice-Sophia Antipolis, 2006.

[104] N. Wotawa, B. Thirion, E. Castet, J.L. Anton, and O. Faugeras. Human retinotopic mapping using fMRI. Technical Report 5472, INRIA, January 2005.

[105] Nicolas Wotawa, Jean-Philippe Pons, Lucero Lopez-Perez, Rachid Deriche, and Olivier Faugeras. fMRI data smoothing constrained to the cortical surface: a comparison of the level-set and mesh-based approaches. In Jean-Baptiste Poline Susan Y. Bookheimer and Balazs Gulyas, editors, *In proceedings of NeuroImage (HBM'04)*, Budapest, Hungary, 2004. Academic Press.

[106] Guoliang Xu. Discrete laplace-beltrami operators and their convergence. *Computer Aided Geometric Design*, 21(8):767–784, 2004.

[107] H. K. Zhao, S. Osher, B. Merriman, and H. Kang. Implicit and nonparametric shape reconstruction from unorganized points using variational level set method. *Computer Vision and Image Understanding*, 80:295–319, 2000.

[108] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01)*, page 194, Washington, DC, USA, 2001. IEEE Computer Society.