Geometric constraints and variational approaches to image analysis

Daniel Antunes

Abstract

Several problems in image processing belong to the class of inverse problems, and hypotheses should be made to have a well-defined formulation, i.e., a formulation in which a solution exists and it is unique. A possibility is to use geometric criteria to regularize the problem, e.g., to favor solutions with smooth contours or short perimeters. This process is called regularization.

However, it is likely the case that the objects in the scene have unknown mathematical representations and that such geometric measurements should be computed in place, considering only their visual representation: In the case of image processing, a digital image. Usually, such measurements are computed without considering the nature of the digital domain, and consequently, are not guaranteed to converge neither approximate the expected Euclidean quantity. The regularization is thus incorrect or not precise, and the solutions biased.

Recently, several digital estimators of geometric properties such as tangent and curvature were proven multigrid convergent. In other words, the estimated values computed in the digital representation of a shape converges towards the values computed in its Euclidean representation as the digital mesh becomes finer and finer. However, there exist few models in the image processing literature that make use of them. That is because such estimators are more difficult to integrate in an optimization framework.

In this thesis, we investigate the use of multigrid convergent estimators and their applications in image processing. In particular, we aim to integrate regularizers based on convergent estimators of curvature in image segmentation problems. We present four combinatorial models based on the elastica energy (a classical geometric regularization term combining perimeter and curvature) with applications in image segmentation. Next, we evaluate our results and compare with similar methods. The results have shown to be very competitive with the state of art.

Résumé

Beaucoup de problèmes en analyse d'images sont caractérisés comme des problèmes inverses, et des hypothèses s'avèrent nécessaire pour obtenir un formulation bien posée, c'est-à-dire que le problème ait une solution et que celle-ci soit unique. Une approche possible consiste à utiliser des critères géométriques pour régulariser le problème, par exemple pour favoriser des solutions avec des contours lisses ou de faible périmètre.

Cependant, dans le cadre de l'analyse d'image ; nous ne disposons pas de la représentation mathématique des objets dans une scène observée. Nous devons utiliser les seules données discrètes (les couleurs des pixels de l'image) qui approchent ces objets et les mesures géométriques sont alors délicates. Les méthodes classiques prennent peu en compte la nature discrète des données dans leur mesure. En conséquence, nous n'avons pas de garanties de convergence ou même d'approximation des mesures effectuées par rapport aux mesures euclidiennes attendues. La régularisation dans le processus de traitement d'image est alors incorrecte ou peu précise, et les solutions trouvées sont alors biaisées.

Récemment, plusieurs estimateurs discrets de propriétés géométriques, notamment liés à la longueur, à la tangente et à la courbure, ont été prouvé convergents multigrilles. Autrement dit la valeur mesurée par ces estimateurs sur la représentation discrète d'une forme converge vers la valeur mesurée sur sa forme euclidienne quand on utilise des grilles de discrétisations de plus en plus fines. Néanmoins, on constate que la littérature d'analyse d'image comporte peu de modèles qui utilisent des estimateurs convergents multigrille. Cela vient du fait qu'il est plus difficile des les intégrer dans les algorithmes de résolution.

Dans cette thèse, nous explorons l'utilisation d'estimateurs convergents multigrille dans des applications en analyse d'image. Plus spécifiquement nous cherchons à intégrer des régularisations basées sur des estimateurs convergents de courbure dans des processus de segmentation d'image. Nous présentons quatre modèles variationnels combinatoires basés sur l'énergie dite "Elastica" (combinaison classique de régularisation géométrique utilisant la longueur et la courbure) avec application en segmentation d'image. Nos résultats sont ensuite évaluées et comparées avec des méthodes similaires, et nos modèles s'avèrent très compétitifs avec l'état de l'art.

Contents

Notation

Domains

$\mathbb{Z}$	Integers
$\mathbb{R}$	Reals
$\mathbb{R}_+$	Non negative reals

Vectors and matrices

$\mathbf{A}$	Matrix
$\mathbf{A}_i$	i-th column of matrix $\mathbf{A}$
$\mathbf{A}_{i,j}$	Element in the ith row and jth column of matrix $\mathbf{A}$
$\mathbf{x}$	Vector
$\mathbf{x}_i$	ith coefficient of vector $\mathbf{x}$

Functions

f, g	Functions
x, y, z	Variables
a, b, c	Constants
$\hat{f}$	Estimation function
$E_{\boldsymbol{\theta}}$	Functional E with parameter vector $\boldsymbol{\theta}$

Sets and points

A, B, C	Sets
p, q	Points
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Families of sets

$\mathbb{1}_A$	Active vector for set A

Image

$\boldsymbol{I}$	Image matrix
$f_{\boldsymbol{I}}$	Image function
Ω	Image domain
$\widetilde{\boldsymbol{I}}$	Input image
$\mathbb{F}$	Finite set of gray level values

Graphs

$\mathcal{G}(\mathcal{V}, \mathcal{E})$	A graph and its vertices and edge sets
$\{v_p, v_q\}$	An undirect edge
(v_p, v_q)	A direct edge
$\mathcal{N}_4$	Four-adjacency neighborhood
$\mathcal{N}_8$	Eight-adjacency neighborhood

Markov random fields

$\mathcal{H}$	A Markov random field
$\boldsymbol{X}$	A random variable vector
$Pr(\boldsymbol{X}_i = \boldsymbol{x}_i \mid \boldsymbol{Y}_j = \boldsymbol{y}_j)$	Conditional probability
Γ	A label set
ϕ	A clique potential function
$\mathcal{G}_{\boldsymbol{I}}$	Image grid graph
$\mathcal{G}_{\boldsymbol{I}+}$	Capacitated image grid graph

Digital and continuous shapes

S	Bidimensional Euclidean shape, i.e., $S \subset \mathbb{R}^2$
$D_h(S)$	Digitization of Euclidean shape S, i.e., $D_h(S) \subset h\mathbb{Z}^2$
D	Bidimensional digital shape, i.e., $D \subset \mathbb{Z}^2$
$B_r(p)$	Digital ball of radius r in a grid of size $h = 1$

Notations for chapter 5 (A combinatorial model for digital elastica ...)

d_D	Signed distance transformation of shape D
$I(D)$	Inner pixel boundary of D
$R_m(D)$	m-ring set of digital set D
$\mathcal{W}_n(D)$	n-neighborhood of shape D

Notations for chapter 6 (A 2-step evolution model driven by ...)

$w(p)$	Variable index function for pixel p
$D^{(k)}$	The k-th digital shape in the evolution
$I^{(k)} = I(D^{(k)})$	The inner pixel boundary of the k-th digital shape
$X^{(k)} = X(I(D^{(k)}))$	The k-th set of optimization variables in the evolution
$F^{(k)} = D^{(k)} \setminus I^{(k)}$	The k-th trust foreground
$F_r^{(k)}(p)$	Intersection of k-th set $F^{(k)}$ with $B_r(p)$
H_F, H_B	Mixed Gaussian distributions

Notations for chapter 7 (A single step evolution model driven by ...)

$O^{(k)}$	Incident pixels (inner,outer) to $\partial D^{(k)}$
$u_r(D, p)$	Balance coefficient for digital shape D centered at p

Notations for chapter 8 (Digital elastica minimization via graph cuts)

$E_{\boldsymbol{\gamma}}^{(gcut)}$	Standard graph cut energy
$E_{\boldsymbol{\gamma}}^{(gflow)}$	Candidate selection energy
$E_{(\boldsymbol{\theta},\boldsymbol{\gamma})}^{(val)}$	Validation energy
$\mathcal{P}_a(D)$	The a-probe set of D
$\mathcal{G}_D$	The candidate graph for D
D^{+a}	Dilation of D by a disk of radius a
D^{-a}	Erosion of D by a disk of radius a
$sol(D)$	Solution candidates set
O_n	Optimization band of width n

Abbreviations

MAP	Maximum a posteriori
TV	Total variation
PDE	Partial differential equations
MRF	Markov random field
HMM	Hidden Markov model
PBF	Pseudo-boolean function
CS	Curve-shortening
DSS	Digital straight segment
MLP	Minimum length polygon
MDCA	Maximal digital circular arcs
II	Integral invariant
LS	LocalSearch
FF	FlipFlow
BF	BalanceFlow
GF	GraphFlow

Introduction

Is the universe continuous or discrete? The analyst would certainly say that it is continuous and it will point out the natural phenomena modeled by calculus and the successful applications in engineering. The computer scientist would surely not accept this idea because there is not such a thing as an unlimited space or unlimited time. The statistician would rather say that both are correct with a probability of 50%.

This thesis has no pretension to answer this intricate question and its philosophical consequences. Our contributions are much humbler than that. Applied mathematics have to face this duality every time an analytic solution is not available and this is not different with image processing, except that in the latter we have to deal with a third ingredient: *digital objects*.

An image is a 2D discrete representation of a projection of a much more complex 3D world. The unit of an image is the pixel, and pixels lie in the digital grid, which is a regular sampling of the plane. Differently from most discretization schemes in numerical analysis, whose discretization points are allowed to be located anywhere, points in an image are constrained to a subset of $\mathbb{Z}^2$. This restriction lead to a serious problem if our image processing model is based on *geometric measurements* of objects in the scene. How to compute geometric measurements of objects described by points in $\mathbb{Z}^2$?

A key argument of this thesis is that general discretizations of geometric measurements, e.g. perimeter, tangent, curvature, do not extend well to the digital world. A linear discretization of curvature proven convergent (in some sense) to the continuous definition do not say too much when this measurement is done in digital objects. First because we are not allowed to position the discretization points anywhere, and second because the convergence theorems usually tells us about convergence when the number of discretization points goes to infinity, which is very frustrating when coping with finite data. More appropriate would be a convergence theorem that relates the convergence speed with the resolution of the digital grid. This concept exists and it is called the *multigrid convergence*.

Recently, several digital estimators of tangent and curvature were proven multigrid convergent, but there is a lack of image processing models using such estimators.

This thesis investigates the use of multigrid convergent estimators in models of image processing relying on a combinatorial optimization framework. The developed work is mainly concerned with image segmentation models using a digital version of the *elastica energy*. The thesis is grouped in two parts:

Image Processing and Digital Geometry. We review classical models of image processing (continuous and discrete), and we dedicate particular attention to those employing geometric properties as regularizers, mainly the curvature. At the end of this part, we briefly introduce basic concepts from digital geometry and the multigrid convergence definition, as well as some examples of multigrid convergent estimators.

Contributions. In this part we grouped all our contributions. It is composed of four combinatorial models aiming elastica energy minimization. Some of them can be applied in image segmentation. The last model is the fastest one and is based on graph cuts.

Chapters outline

Part I: Image Processing and Digital Geometry

1. Variational models in Image Processing. As many applied fields, image processing drinks from the source of continuum models, in particular those emerging from partial differential equations and signal processing. The image is then modeled as an infinitely smooth function and, based on the principle of *least energy*, energies (functionals) are proposed such that the image of minimum (or maximum) value gives us the answer of the problem. That is the so called *variational approach*, that eventually is solved by the computation of its *Euler-Lagrange equation* and it boils down to find the steady-state of a diffusion process. The classical *Tikhonov* and *total variation* for image denoising and inpainting follow this principle. Similarly, we have curve evolution approaches to segment objects of interest in the scene.

2. Discrete methods in Image Processing. The continuous point of view is popular because it is, indeed, very powerful and produce satisfactorily results for several imaging problems as discussed in Chapter 1. An important drawback, however, is that it is still very difficult for a continuous model to preserve the discontinuities of an image along the edges of the objects present on it, which is the most important feature of an image (the human perception targets discontinuities in images). That is one of the reasons why discrete (combinatorial) approaches appear as an alternative. In this chapter, we review some combinatorial models inspired by *Markov random fields* and

we will be interested in a special class of energies emerging from its Gibbs energy: the *submodular pseudo-boolean functions* class. In particular, we are going to point out its relationship with *graph cuts* and present some graph cut based models applied in imaging at the end of the chapter.

3. Curvature as regularizer. We focus on models that employ curvature as a regularization term, both in continuous and discrete settings. In a second moment, we turn to the elastica energy and examine imaging models based on it for segmentation and inpainting. Finally, we describe combinatorial models that attempts to minimize the squared curvature.

4. Digital Geometry. It is often the case that we do not know a priori the mathematical expressions modeling the objects in an image. Sometimes, it is not even possible to get such expressions, or at least, it is very complicated. Therefore, we have to recognize shapes from their digital representation, possibly by recognizing primitives as lines, circles, etc. However, we should remember that the sampling on a regular grid imposes special conditions on how the computation of geometric properties is done on digital objects. Digital geometry offers the proper tools to do such measurements and evaluate their convergence towards the measurements on the continuous representation of the object. In this chapter, we give a brief introduction to digital geometry and we define the multigrid convergence property for digital estimators of geometric properties.

Part II: Contributions

5. A combinatorial model for digital elastica shape optimization. We propose a local combinatorial model to minimize the elastica energy using multigrid convergent estimators of length and curvature. We validate the model through experiments and we observe convergence to the shape of minimum elastica value in the *free elastica problem*. At the end of the chapter, we sketch some global optimization models.

6. A 2-step evolution model driven by digital elastica minimization. We propose to iteratively minimize a quadratic non-submodular pseudo boolean function to evolve an initial shape to another with lower elastica energy. The model is specially conceived for the *curvature integral invariant estimator* described in Chapter 4. We present an application to image segmentation at the end of the chapter.

7. A single step evolution model driven by digital elastica minimization. It can be seen as an improved version of the previous model, though with some differences. In particular, the model is singled step and has an

easier implementation. In this chapter we introduce the *balance coefficient* and we set up the terrain for a graph cut based model.

8. Digital elastica minimization via graph cuts. The balance coefficient defined in the previous chapter is used to set up a cost function on the edges of *candidate graphs*. The candidate graphs are derived from a neighborhood of shapes, and the solution, at each iteration, is chosen as the source component of a minimum cut in the candidate graphs with lowest digital elastica value. We observe convergence to the global optimum shape in the free elastica problem and we show how to use it in image segmentation.

9. Result analysis. We make a summary of the models developed in this thesis and we point out its pros and cons. At the end of the chapter, we present a comparison of them with a competitor model that uses curvature regularization in image segmentation.

Part I

Image Processing and Digital Geometry

Chapter 1

Variational methods in Image Processing

A track should be constructed to connect some point in space p to a lower altitude point q. Which form the track should take if we wish that a ball released at p reaches q in the shortest time? The curve known as brachistochrone or tautochrone is the answer of this puzzle solved by Jean Bernoulli and a classical problem of the *calculus of variations*.

The main object of calculus of variations are called *functionals* or *energies*, and a simple way to describe it is as a function whose variable is itself a function. Minimizing functionals is a more intricate problem than minimizing an usual function, as the variable in a functional has infinite dimension. Nonetheless, by means of the so called *variations*, one can model infinitely small variations in the functional and do a rigorous analysis of its extremum, the main tool of which is the *Euler-Lagrange* equation.

The calculus of variations found in image processing a fertile field of applications, as images themselves can be seen as functions, and image processing tasks can be modeled as being the results of some functional minimization. In this chapter we present some popular variational techniques to approach image processing tasks, with a particular focus on image segmentation.

1.1 Inverse problems in imaging

An archaeological museum decided to digitize some of its collections and make them available for digital visits over the internet. The chosen method of digitization consists in taking a set of pictures for each object, in different camera positions, execute a *stereo* algorithm to estimate point depths and finally reconstruct the 3D object. The stereo and reconstruction are examples of *inverse problems* in imaging.

Inverse problem	Forward problem
Projection: Compute vector $\boldsymbol{v} \in \mathbb{R}^3$ whose projection is $P(\boldsymbol{v}) \in \mathbb{R}^2$	Compute the projection $P(\boldsymbol{v}) \in \mathbb{R}^2$ of vector $\boldsymbol{v} \in R^3$
Parameters inference: Given a set of observations Γ, infer the parameters (μ, σ) of the Gaussian distribution that describes Γ	Given a random variable X following a Gaussian distribution with parameters $(\mu = 0, \sigma = 1)$, compute the probability $P(X \leq 0.42)$
Image denoising: Given noisy image $\widetilde{\boldsymbol{I}}$, compute the original image $\boldsymbol{I}$, i.e., the image without noise	Add some random noise to a given image $\boldsymbol{I}$ to produce noisy image $\widetilde{\boldsymbol{I}}$
Image inpainting: Given image $\widetilde{\boldsymbol{I}}$ with a missing patch, reconstruct the removed patch	Remove a patch from image $\boldsymbol{I}$
Image segmentation: Given image I, find the labeled partition $\mathcal{I}$	Given a labeled partition $\mathcal{I}$ of some image I, assemble the pieces to create image I

Table 1.1: Examples of inverse problems and its direct versions. Inverse problems are characterized by uncertainty and parameter inference.

Usually, inverse problems are characterized by a degree of *uncertainty* or *lack of information*. The 2D pictures in the problem above miss depth information, that should be *inferred* by the stereo algorithm. On the other hand, if the shape geometry was known, e.g., the values of mean curvature were known for every infinitesimal point of the shape, then constructing a digital 3D representation would be a *forward problem*.

We can find examples of inverse problems in several branches of mathematics [**kirsch96**], geophysics [**zhdanov15**], natural language processing [**stroppa05**], astronomy [**lucy94**] and the list goes on. The image processing field itself is plenty of them [**bertero98**]. In fact, a great part of real world applications consists in inferring parameters of some model, i.e., an inverse problem. In Table 1.1 we list some examples of inverse problems and its corresponding forward version.

Another characteristic of inverse problems is that they are usually *ill-posed*. A problem is said to be ill-posed if at least one of the properties below is not respected

1. A solution exists and it is unique;

2. The solution changes continuously with its parameters

In order to solve ill-posed problems one should include additional information, i.e., create assumptions over the properties of the sought solution. In the museum problem, for example, one may assume that missing patches of the reconstructed surface should be filled by patches of minimal area. The process of including additional information in ill-posed problems is called *regularization* and its goal is to

better condition an ill-posed problem and in the best scenario, transform it into a well-posed one.

Next, we describe the image model used in this thesis and give a precise definition of the main image problems discussed further on. Examples of such applications can be seen in Figures 1.1 and 1.2.

Image model

For matters of simplicity, we limit our discussion to grayscale images, the concepts being mostly extendable to multichannel images. It is convenient to have in mind two different representations of an image.

$$\begin{aligned} \text{Discrete:} \quad & \boldsymbol{I} \in \mathbb{F}^{m \times n} \\ \text{Continuous:} \quad & f_{\boldsymbol{I}} : \Omega \subset \mathbb{R}^2 \to [0,1], \end{aligned}$$

where $\mathbb{F}$ is a finite set. In this thesis, we define such set as

$$\mathbb{F} = \{\frac{i}{255} \mid i \in \mathbb{N}, i \leq 255\}. \tag{1.1}$$

The discrete representation is interpreted as a sampling of $m \times n$ elements (pixels) of the continuous representation $f_{\boldsymbol{I}}$.

Image denoising

Given an image $f_{\tilde{\boldsymbol{I}}}$ corrupted with some noise from an external source, *image denoising* consists in finding an estimation $f_{\hat{\boldsymbol{I}}}$ of the original image that respects some quality criteria, usually encoded by the minimum of a functional E.

Given $f_{\tilde{\boldsymbol{I}}}$, find estimation $f_{\hat{\boldsymbol{I}}}$ such that

$$f_{\hat{\boldsymbol{I}}} = \arg\min_{f} E(f, f_{\tilde{\boldsymbol{I}}})$$

Applications: Restoration of old pictures; enhancement of satellite images.

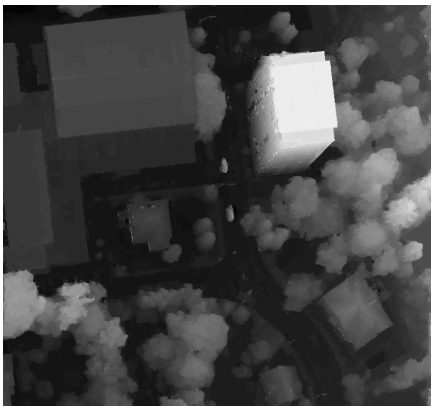
Image segmentation

Given an image $f_{\boldsymbol{I}}$, the *image segmentation* problem consists in finding a partition $\mathcal{I}$ of $f_{\boldsymbol{I}}$ such that each element of $\mathcal{I}$ is identified with some desired property, usually encoded by the minimum of some functional E. Given $f_I : \Omega \to [0,1]$ and a positive integer n, find partition $\mathcal{I}^\star = \{\Omega_i \subset \Omega \mid i \leq n\}$ such that

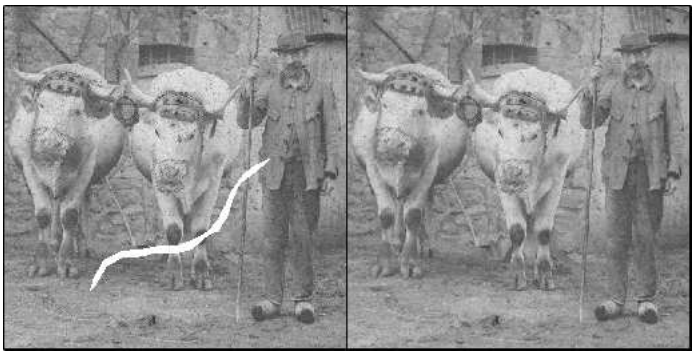
$$\mathcal{I}^\star = \arg\min_{\mathcal{I}} E(\mathcal{I}, f_{\boldsymbol{I}}) \quad \text{subject to} \quad \begin{array}{l} \forall i \neq j : \ \Omega_i \cap \Omega_j = \emptyset \\ \bigcup_i^n \Omega_i = \Omega \end{array}$$

Applications: enhance blood vessels in angiograms; track roads in satellite images; identify objects in a scene.

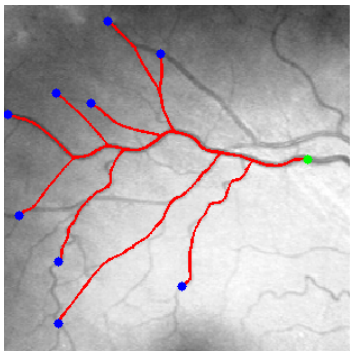
(a) 1 of 2 aerial images



(b) Depth reconstruction [**pock08**]

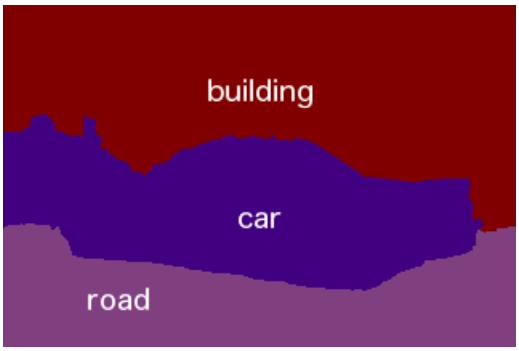


(c) Photo restoration [**masnou98inpainting**]



(d) Vessel segmentation [**peyre10**; **cohen97**].

(e) Input image



(f) Multilabel segmentation [**souiai2013co**]

Figure 1.1: **Real applications of imaging problems**.

Image inpainting

Given an image $f_{\widetilde{\boldsymbol{I}}}$ with a collection of missing patches $\mathcal{P}$, *image inpainting* consists in creating an image $f_{\boldsymbol{I}}$ with the reconstructed missing patches such that a quality criteria, encoded as the minimum of some functional, is respected

Given $f_{\widetilde{\boldsymbol{I}}}$ and missing patches $\mathcal{P}$, find image $f_{\boldsymbol{I}}$ such that

$$f_{\boldsymbol{I}} = \arg\min_{f} E(f, f_{\widetilde{\boldsymbol{I}}}) \quad \text{subject to } f(\Omega \setminus \mathcal{P}) = f_{\widetilde{\boldsymbol{I}}}(\Omega \setminus \mathcal{P}).$$

Applications: removal of undesired objects in a scene; restoration of old pictures.

1.2 Bayesian rationale and total variation

As remarked in the previous section, inverse problems involve some level of uncertainty about the solution. In order to solve an ill-posed problem we need to regularize it by including additional information, otherwise said, make assumptions.

The maximum a posteriori method was first introduced in the image processing community in the work of [**geman84**] and we are going to reproduce here the rational for image denoising. We make two assumptions

1. The noisy image $\widetilde{\boldsymbol{I}}$ was obtained by addition of a normal Gaussian noise with $\mu = 0, \sigma = \lambda^{-1/2}$ $(\lambda > 0)$ to the original image, i.e.,

$$\widetilde{\boldsymbol{I}} = \boldsymbol{I} + \boldsymbol{N}, \tag{A.1}$$

   where $\boldsymbol{N}$ is a $(m \times n)$ matrix of random variables $\boldsymbol{N}_{i,j}$ and $Pr\big(\boldsymbol{N}_{i,j} = n\big) = \frac{1}{Z_1}\exp(-\lambda\frac{n^2}{2})$, with Z_1 being the normalization constant.

2. Given some function ρ, a candidate image estimation $\boldsymbol{C}$ has probability

$$Pr(\boldsymbol{C}) = \frac{1}{Z_2}\exp(-\rho(\boldsymbol{C})), \tag{A.2}$$

   where Z_2 is the normalization constant.

$$\widehat{\boldsymbol{I}} = \arg\max_{\boldsymbol{C}} Pr(\boldsymbol{C} \mid \widetilde{\boldsymbol{I}}) = \arg\max_{\boldsymbol{C}} \frac{Pr(\widetilde{\boldsymbol{I}} \mid \boldsymbol{C})Pr(\boldsymbol{C})}{Pr(\widetilde{\boldsymbol{I}})}. \tag{1.4}$$

We have already all the elements to expand Equation (1.4). The probability of having the corrupted image $\widetilde{\boldsymbol{I}}$ given a candidate image $\boldsymbol{C}$ is derived from Equation (A.1), i.e.,

$$Pr(\widetilde{\boldsymbol{I}} \mid \boldsymbol{C}) = Pr(\boldsymbol{N} = \widetilde{\boldsymbol{I}} - \boldsymbol{C}) = \frac{1}{Z_1}\exp\Big(-\frac{\lambda\|\widetilde{\boldsymbol{I}} - \boldsymbol{C}\|^2}{2}\Big). \tag{1.5}$$

(a) Original image

(b) 10-partition as in [**chambolle08**]

(c) Noisy image

(d) Image denoised with FISTA [**beck09a**]

(e) Inpainting mask

(f) Inpainted image with [**fedorov15**]

Figure 1.2: **Imaging problems applications**. From top to bottom row, an example of image segmentation, denoising and inpainting.

The denominator term is computed as the joint probability

$$Pr(\widetilde{\boldsymbol{I}}) = \sum_{\boldsymbol{J}\in\mathbb{F}^{m\times n}} Pr(\widetilde{\boldsymbol{I}} \mid \boldsymbol{J}) Pr(\boldsymbol{J}) = \frac{1}{Z_1 Z_2} \sum_{\boldsymbol{J}\in\mathbb{F}^{m\times n}} \exp\left(-\frac{\lambda}{2}\|\widetilde{\boldsymbol{I}} - \boldsymbol{J}\|^2 - \rho(\boldsymbol{J})\right). \tag{1.6}$$

Substituting Equations (1.5) and (1.6) in Equation (1.4) we obtain

$$\widehat{\boldsymbol{I}} = \arg\max_{\boldsymbol{C}} \frac{\exp\left(-\frac{\lambda}{2}\|\widetilde{\boldsymbol{I}} - \boldsymbol{C}\|^2 - \rho(\boldsymbol{C})\right)}{\sum_{\boldsymbol{J}\in\mathbb{F}^{m\times n}} \exp\left(-\frac{\lambda}{2}\|\widetilde{\boldsymbol{I}} - \boldsymbol{J}\|^2 - \rho(\boldsymbol{J})\right)} \tag{1.7}$$

Finally, solving Equation (1.7) is equivalent to solve

$$\widehat{\boldsymbol{I}} = \arg\min_{\boldsymbol{C}} \frac{\lambda}{2}\|\widetilde{\boldsymbol{I}} - \boldsymbol{C}\|^2 + \rho(\boldsymbol{C}). \tag{1.8}$$

The first term appears so often in imaging problems that it has a special name: *data fidelity*. In the denoising problem, the data fidelity term appeared as a consequence of the Gaussian noise model assumption. The second term is also a regularization term and it favors images that respect some desirable property for the problem to be solved. Since natural images has a higher spatial dependency, a reasonable guess for ρ would be a function that has lower value for piecewise smooth data, i.e., images composed by closed regions with smooth variations in its interior but possibly strong discontinuities in their boundaries.

1.2.1 Tikhonov regularization

The classical way to optimize Equation (1.8) is to shift it to a continuous setting, analytically derive some optimization properties and then use this properties to solve the problem in a discrete setting. The continuous reformulation of Equation (1.8) consists in optimizing the energy functional below

$$f_{\widehat{\boldsymbol{I}}} = \arg\min_{f} F(f) = \frac{\lambda}{2}\int_{\Omega} \|f_{\widetilde{\boldsymbol{I}}} - f\|^2 dx + R(f), \tag{1.9}$$

where R is a functional derived from the choice of ρ. A popular choice for R is to define it as the $L2$ norm of ∇f, also called the *Tikhonov* regularization term. Equation (1.9) is rewritten as

$$f_{\widehat{\boldsymbol{I}}} = \arg\min_{f} F(f) = \frac{\lambda}{2}\int_{\Omega} \|f_{\widetilde{\boldsymbol{I}}} - f\|^2 dx + \int_{\Omega} \|\nabla f\|^2 dx. \tag{1.10}$$

1.2.2 Euler-Lagrange equation

We can establish some necessary optimal conditions for Equation (1.10) by deriving its *Euler-Lagrange* equation. Assume that function g minimizes functional F, i.e.,

$$g = \arg\min_f F(f).$$

Further, assume that there exists a function w that agrees with g at the boundary of f' domain, i.e., $w(x) = 0$, $\forall x \in \partial\Omega$. Define the function h as

$$h(\epsilon) = F(g + \epsilon w)$$

Therefore, h has a minimum at $\epsilon = 0$. Thus,

$$\begin{aligned} 0 = \frac{dh}{\partial\epsilon}_{|\epsilon=0} &= \frac{d}{\partial\epsilon}_{|\epsilon=0} \int_\Omega \frac{\lambda}{2}\|f_{\widetilde{\boldsymbol{I}}} - g - \epsilon w\|^2 + \|\nabla(g + \epsilon w)\|^2 dx \\ &=_{|\epsilon=0} \int_\Omega \lambda\|f_{\widetilde{\boldsymbol{I}}} - g - \epsilon w\| \frac{(f_{\widetilde{\boldsymbol{I}}} - g - \epsilon w)}{\|f_{\widetilde{\boldsymbol{I}}} - g - \epsilon w\|} w + 2\|\nabla(g + \epsilon w)\| \frac{(\nabla g + \epsilon w)}{\|\nabla(g + \epsilon w)\|} \nabla w dx \\ &= \int_\Omega \lambda(f_{\widetilde{\boldsymbol{I}}} - g)w + (\nabla g)\nabla w dx. \end{aligned}$$

Applying integration by parts and using the fact that $w(x) = 0$, $\forall x \in \partial\Omega$.

$$0 = \int_\Omega \big(\lambda(f_{\widetilde{\boldsymbol{I}}} - g) - \Delta g\big) w dx$$

Since w could be any function, we can write

$$\lambda(f_{\widetilde{\boldsymbol{I}}} - g) - \Delta g = 0 \tag{1.11}$$

Therefore, if g is a minimum of Equation (1.9), then it respects the convex Equation (1.11). Hence, given an initial solution f, one can execute a descent method (gradient descent, for example) to find its minimum. In practice, Equation (1.9) is discretized using the samplings $\widehat{\boldsymbol{I}}, \widetilde{\boldsymbol{I}}$ of $f_{\widehat{\boldsymbol{I}}}, f_{\widetilde{\boldsymbol{I}}}$ and a finite differences scheme is defined to estimate the Laplacian Δ.

The Tikhonov term favors images with smooth variations in color, but the smoothness is not restricted to the interior of regions. Thus, Tikhonov tends to obfuscate the discontinuities that will likely be present in the contour of regions and we have the impression that the image is blurred (see Figure 1.3). Nonetheless, Tikhonov term is attractive due to its optimization properties.

(a) Noisy (b) Tikhonov (c) Total Variation

Figure 1.3: **Tikhonov x Total variation denoising**. Total variation is capable to better preserve discontinuities across edges than Tikhonov.

1.2.3 Total variation regularization

An alternative to Tikhonov regularization is to use the so called *total variation* of the image function. For a smooth function f, its total variation is computed as

$$TV(f) = \int_\Omega \|\nabla f\|.$$

For a more general (possibly not differentiable) locally integrable function $f : \Omega \to \mathbb{R}^n$, its total variation is defined as

$$\begin{aligned} TV(f) &= \sup \left\{ \int_\Omega \nabla u \cdot \phi \mid \phi \in C_c^1(\Omega, \mathbb{R}^n) \text{ and } \|\phi\|_\infty \leq 1 \right\} \\ &= \sup \left\{ -\int_\Omega u \nabla \cdot \phi \mid \phi \in C_c^1(\Omega, \mathbb{R}^n) \text{ and } \|\phi\|_\infty \leq 1 \right\}, \end{aligned}$$

where ϕ is vector-valued infinitely differentiable function with compact support. The image denoising total variation model is written as

$$f_{\tilde{\boldsymbol{I}}} = \arg\min_f \frac{\lambda}{2} \int_\Omega \|f_{\tilde{I}} - f\|^2 dx + TV(f). \tag{1.12}$$

The ROF model [**rudin92**] assumes the image representation is smooth, and from the Euler-Lagrange equation of Equation (1.12) the following gradient flow is derived:

$$\frac{\partial f}{\partial t} = \nabla \cdot \left(\frac{\nabla f}{\|\nabla f\|} \right) - \lambda (f_{\tilde{\boldsymbol{I}}} - f)$$

The left term is not differentiable, and a small $\epsilon > 0$ is added to the denominator in order to avoid numerical instability. However, the calibration of ϵ might be delicate, since a small ϵ might not be sufficient to avoid instability and a larger ϵ may disfigure the model. In [**chambolle04**] the total variation definition is exploited to create a convergent algorithm that works by successive projections and that solves model Equation (1.12). In [**beck09a**] a modified version of the previous algorithm has proven to have faster convergence.

The total variation term is characterized by its smooth properties while partially preserving some discontinuities across the edges. In this sense, total variation models produce results with sharper edges than those produced by the Tikhonov term (see Figure 1.3) .

1.3 Standard techniques

In this section we give an overview of the key techniques in variational models for problems in image processing. We start by describing the most influential model in this category.

Mumford-Shah

The $L2$-norm regularization has nice optimization properties, but it does not preserve discontinuities along the object boundaries. This effect is attenuate using a $L1$-norm, but it it not sufficient to avoid blurred edges. The*Mumford-Shah* functional [**mumford89**] handles this issue by incorporating the edges in its formulation in the form of a set of discontinuities $\mathcal{K}$ and limiting the $L2$-norm regularization to points in the interior of objects, i.e., $\Omega \backslash \mathcal{K}$. Moreover, the set $\mathcal{K}$ itself is compelled to be of small length. The Mumford-Shah model consists in minimizing the following functional

$$(f_{\widehat{\boldsymbol{I}}}, \widehat{\mathcal{K}}) = \arg\min_{f,\mathcal{K}} \alpha \int_{\Omega} \|f_{\widetilde{\boldsymbol{I}}} - f\| dx + \beta \int_{\Omega \backslash \mathcal{K}} \|\nabla f\| dx + \lambda Per(\mathcal{K}) \tag{1.13}$$

The functional can be seen as a model for both denoising and segmentation problems. The function $f_{\widehat{\boldsymbol{I}}}$ being the denoising solution and $\widehat{\mathcal{K}}$ the segmentation solution. Equation (1.13) is proven to have a minimizer [**degiorgi89**], and in the case $\mathcal{K}$ is fixed, the minimizer is unique (see chapter 25 of [**bar11**]). However, to find a minimizer of Equation (1.13) is a challenging task due to its non-convexity.

Nonetheless, there exist several approximations to the Mumford-Shah functional. We refer to the phase-field model of [**ambrosio90**]; the finite-differences scheme of [**chambolle99**]; the level-set method of [**vese02**]; the convex relaxations of [**pock09**; **strekalovskiy14**]; and the discrete calculus approach of [**foare17**].

1.3.1 Curve evolution

Active contours

Active contours or snakes is a supervised method for doing image segmentation. In the original work [**kass88**], an initial parametric curve $C_0(q) \rightarrow (x(q), y(q))$ is evolved towards the local minimum of the snakes energy

$$\begin{aligned} F(C) &= \alpha Length(C) + \beta Smoothness(C) + \gamma Edge(C) \\ F(C) &= \alpha \int_0^1 \left\| \frac{dC}{\partial q} \right\|^2 dq + \beta \int_0^1 \left\| \frac{d^2C}{\partial q^2} \right\|^2 dq - \gamma \int_0^1 \|\nabla f_I(C(q))\|^2 dq \end{aligned} \tag{1.14}$$

The length and smoothness regularization term favors curves of smooth variations and small length while the edge term compels the curve to stop at regions of high variation of color intensity.

The snakes method was devised having an interactive framework in mind. First of all, the user must set the initial curve close to the object to be segmented, and besides that, a set of additional tools as anchor points, repulsion and spring forces are available for online modification of the problem. The user can make use of these tools to conveniently perturb the current solution and force the curve to evolve to the expected local optimum.

The active contours is an influential paradigm for image segmentation and it was particularly popular for segmenting medical images [**mcinerney99**]. Variations of the original model include extension to 3D-segmentation [**mcinemey99**] and topologically adaptable snakes [**mcinerney95**].

Some drawbacks in the active contours formulation include its non-intrinsic definition, i.e., the curve is not defined in terms of its geometric properties and its representation depends on the chosen parametrization; and, partially as consequence of the latter, its inability to change the initial curve topology. One needs to initialize several snakes in order to correctly segment a picture with several holes, for example.

Geometric active contours

Parametric models as snakes are often criticized because of their non-intrinsic definition, i.e., the energy is not defined in terms of the geometric properties of the contours. That makes the theoretical analysis of the snakes model harder, as the evolution of the contour itself depends of the chosen parametrization. In [**caselles93**] the authors propose a model based on the mean curvature motion of the level-sets of a C^2 function u.

Let $u : \Omega \subset \mathbb{R}^2 \to [0, 1]$ be a C^2 function. The curvature at its k-th level set is given by

$$\kappa(x, y) = \nabla \cdot \left(\frac{\nabla u}{\|\nabla u\|} \right), \quad \forall (x, y) \in \{ (x, y) \mid u(x, y) = k \}$$

The *geometric active contour* model consists in evolving an extension of u, by including an artificial time parameter t, and compute the steady solution of the flow

$$\begin{aligned} u(0, x, y) &= u(x, y) \\ \frac{du}{dt} &= g(\|\nabla f_{\boldsymbol{I}}\|)\|\nabla u\|\nabla \cdot \left(\frac{\nabla u}{\|\nabla u\|} + v \right), \end{aligned} \tag{1.15}$$

where g is a non-increasing function that plays the role of an edge-detector, e.g., $g(x) = 1/(1 + x)^2$. The function u can be initially defined as a smoothed version of $1 - \chi_C$, where χ_C is the characteristic function of some set $C \in \Omega$ that contains the objects to be segmented.

Following Equation (1.15), the gray level at some point (x, y) changes proportionally to the curvature of its belonging level set. The constant v forces the change in u to be always positive, i.e., pixels gets lighter, never darker. The term $\|\nabla u\|$ allows u to evolve only at some neighborhood of the 0-level-set boundary and the term $g(\|\nabla f_{\boldsymbol{I}}\|)$ makes the evolution to stop if an edge is reached. At the steady solution of Equation (1.15) the segmented objects of $\boldsymbol{I}$ corresponds to the 0-level set of u.

Differently from the snakes, the geometric active contours handle changes in topology of the initial curve. In figure Figure 1.4, the initial 0-level set of u splits in three disjoint sets at the steady solution of Equation (1.15). However, the geometric active contour models cannot segment objects with holes without including a region-based term [**chen06**].

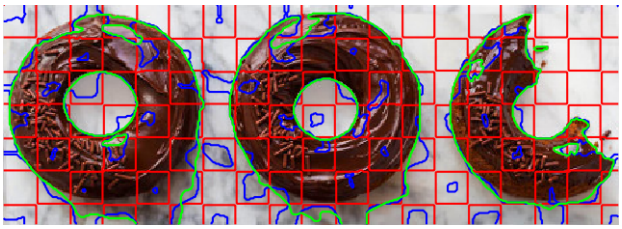
1.3.2 Level set

The active contour and its geometric version are both edge-based methods, a natural strategy for image segmentation but with limitations, e.g., the models may encounter some difficulties to segment objects with holes. The *Chan-Vese* method proposes the inclusion of a region-based term and it generalizes the level-set approach already presented in the geometric active contour model.

Let $f_{\boldsymbol{I}} : \Omega \subset \mathbb{R}^2 \to \mathbb{U}^2$ a grayscale image and $F \subset \Omega$ an open set such that the pair $(F, \Omega \setminus F)$ is the searched binary partition. Further, assume that there exists a function $\phi : \Omega \to \mathbb{R}$ with bounded first derivative. The image partitions are

(a) Active contours

(b) Geometric active contours



(c) Chan-Vese

Figure 1.4: **Curve evolution models**. Segmentation results of three curve evolution models. From top row to bottom: active contours, geometric active contours and Chan-Vese. The initial curve (0-level set) is colored in red and the final one is colored in green. The blue curve highlights an intermediate iteration.

identified in the following fashion

$$\begin{aligned}\phi(x) &< 0, \quad \forall x \in F \\ \phi(x) &= 0, \quad \forall x \in \partial F \\ \phi(x) &> 0, \quad \forall x \in \Omega \setminus \overline{F}\end{aligned}$$

In possession of the partition descriptor ϕ, the following energy is proposed

$$\begin{aligned}F(\phi, x) &= \mu Length(\phi, x) + \nu Area(\phi, x) + \lambda_1 Foreground(\phi, x) + \lambda_2 Background(\phi, x) \\ &= \mu \int_\Omega \delta_0(\phi(x)) \|\nabla \phi(x)\| dx + \nu \int_\Omega H(\phi(x)) dx \\ &+ \lambda_1 \int_\Omega \big(1 - H(\phi(x))\big) \|f_{\boldsymbol{I}}(x) - c_F\|^2 dx + \lambda_2 \int_\Omega H(\phi(x) \|f_{\boldsymbol{I}}(x) - c_B\|^2 dx,\end{aligned} \tag{1.16}$$

where $H(x)$ is the Heaviside function and δ_0 the standard Dirac delta function, i.e.,

$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise,} \end{cases} \qquad \delta_0(x) = \begin{cases} +\infty, & x = 0 \\ 0, & \text{otherwise.} \end{cases} \quad \text{and } \int_{-\infty}^{+\infty} \delta_0(x) dx = 1.$$

The parameters c_f, c_b are defined as the average color intensity in the interior of the foreground and background regions, respectively

$$c_F = \frac{\int_\Omega \big(1 - H(\phi(x))\big) f_{\boldsymbol{I}}(x) dx}{\int_\Omega \big(1 - H(\phi(x))\big) dx}, \quad c_B = \frac{\int_\Omega H(\phi(x)) f_{\boldsymbol{I}}(x) dx}{\int_\Omega H(\phi(x)) dx}.$$

Next, the Euler-Lagrange equation of Equation (1.16) is calculated and used to define a gradient flow to minimize Equation (1.16), in a similar fashion as done in Section 1.2.1. In order to be numerically tractable, the Heaviside and Dirac delta function are regularized as

$$H_\epsilon(x) \quad = \frac{1}{2}\left(1 + \frac{2}{\pi} \arctan(\frac{x}{\epsilon})\right), \qquad \delta_\epsilon(x) \quad = \frac{\epsilon}{\pi(\epsilon^2 + x^2)}.$$

The initial level set function can be set as any function with bounded first derivative, but it is recommended to use the checkerboard function $\phi = \sin(\pi/5x_1)\sin(\pi/5x_2)$ which is reported [**getreuer12**] to have fast convergence. In [**vese02**], the Chan-Vese authors extended their method to contemplate colored images and multisegmentation. An illustration is presented in Figure 1.4

1.3.3 Minimum path

In a sequel work, the authors of geometric active contours established a link between their geometric model in [**caselles93**] and the computation of geodesics in a regular surface [**caselles97**].

The length of a parametric curve $C(q)$ according to an isotropic metric of potential $W(C)$ is calculated as

$$L(C) = \int W(C)\|C_q\|dq. \tag{1.17}$$

Equation (1.17) is used to compute shortest paths between two points according to the given metric. By properly setting the potential W, we can make object boundaries in an image $f_{\boldsymbol{I}}$ to match the curves of shortest length, for example, letting $W = g(\|\nabla f_I\|)$ as in Equation (1.15) we obtain

$$L(C) = \int g\big(\|\nabla f_I(C(q))\|\big)\|C_q\|dq. \tag{1.18}$$

In [**caselles97**] the authors show that the snakes model without the smoothness term($\beta = 0$) is equivalent to geodesics computations, the metric changing accordingly with the models parameters. The isotropic metric above is equivalent to the case in which the length and image terms of the snakes model are equal.

Given an initial curve $C_0(q)$, a local minimizer for Equation (1.18) can be computed by finding the steady solution of the following flow derived from its Euler-Lagrange equation

$$C(0, q) = C_0(q) \tag{1.19}$$

$$\frac{dC}{dt} = (g\kappa - \nabla g \cdot \boldsymbol{n})\boldsymbol{n}, \tag{1.20}$$

where $\boldsymbol{n}$ is the normal vector to the curve C at (t, q). One can show that, given an initial function $u \in \mathcal{C}^2$ such that u is negative (positive) in the interior (exterior) of its 0-level set, the solution of Equation (1.20) equals the steady solution of

$$u(0, x, y) = u_0(x, y) \tag{1.21}$$

$$\frac{du}{dt} = g\|\nabla u\|\kappa + \nabla g \cdot \nabla u \tag{1.22}$$

Comparing Equation (1.22) with Equation (1.15) we notice that the $\nabla g \cdot y$ term was included while the v parameter was removed. The geometric active contour stops as soon as an ideal edge is found (a threshold should be set), which is particularly bad for real images segmentation, as it is likely that the flow will stop at the first

variation of color intensity. The new term allows the flow to evolve even in those cases. Nonetheless, one can include again the v parameter, as it permits to increase the convergence in some cases.

The relation between segmentation and geodesic computation inspired several works. Elongated and thin objects as blood vessels or roads in satellite images are the global optimum of a geodesic computation in which the minimum path is constrained to lie between two points [**cohen97**]. Further development of this work reduced the initialization to just a single point [**benmansour09**]. Anisotropic metrics aligned to the image edges are reported to return improved solutions for blood vessels segmentation [**jbabdi08**; **benmansour11**]. Ideas from Chan-Vese and Geodesic models are put together in [**chen06**]. Finally, an elucidating review of geodesic methods in computer vision can be found in [**peyre10**].

1.3.4 Convex relaxation

A set Ω is convex if for all $a, b \in \Omega$, every element in the line connecting a, b also lie in Ω. A function $f : \Omega \to \mathbb{R}$ is convex if its domain Ω is convex and the following inequality holds

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y).$$

Convexity is a desirable property in optimization problems because any local minimizer (maximizer) is also a global one. In other words, methods that optimizes f do not depend on its initialization. Therefore, it is of great interest to find convex formulations for image processing problems.

Ideally, one would look for the so called convex envelope of f, which is the tightest convex function $\widetilde{f}$ such that $\widetilde{f} \leq f$. In fact, one can compute the envelope of a function f by taking its convex biconjugate.

Definition 1(Convex conjugate): Let $f : \Omega \to \mathbb{R} \cup \{+\infty, -\infty\}$. Its convex conjugate is defined as

$$f^*(y) = \sup_x y^T x - f(x)$$

The biconjugate f^{**} is the convex envelope of f. In fact, if f is convex and lower-semicontinuous, $f^{**} = f$ (Frenchel's inequality). Unfortunately, the computation of the biconjugate is known only for a few functions. Nonetheless, the conjugate is key to prove properties on convex optimization algorithms as those based on the proximal operator [**chambolle04**; **beck09a**].

In order to use tools from convex optimization one needs to define a convex energy. Very often in imaging problems the functionals are defined over a non-convex function space, for example, in the binary denoising or the multilabeling

problem, in which the optimization function has a discrete range. A straightforward approach is to simply relax the range to a continuous set and execute a standard optimization method. The discrete solution is then obtained by simple rounding. However, fewer are the cases in which the projected solution is optimum or even meaningful to the original problem. A technique that gives guarantees with respect to the quality of the back projected solution is *functional lifting*.

Functional lifting

Consider the binary image denoising problem. Let $f_{\tilde{\boldsymbol{I}}}$ the observed noisy image and consider the total variation model for binary denoising

$$\min_{f_{\boldsymbol{I}}:\Omega\to\{0,1\}} E^{2-den}(f_{\boldsymbol{I}}) = \min_{f_{\boldsymbol{I}}} \int_{\Omega} \|\nabla f_{\boldsymbol{I}}\| + \lambda \int_{\Omega} (f_{\boldsymbol{I}} - f_{\tilde{\boldsymbol{I}}})^2, \tag{1.23}$$

This model is not convex, as the optimization variable $f_{\boldsymbol{I}}$ belongs to the non-convex domain of binary functions. The corresponding level-set formulation of Equation (1.23) (in the same spirit of the Chan-Vese model) is given by

$$\min \int_{\Omega} \|\nabla H(\phi(x))\| + \lambda \int_{\Omega} \big(H(\phi(x)) - f_{\tilde{\boldsymbol{I}}}(x)\big)^2. \tag{1.24}$$

Whose a local minimum is a steady solution of

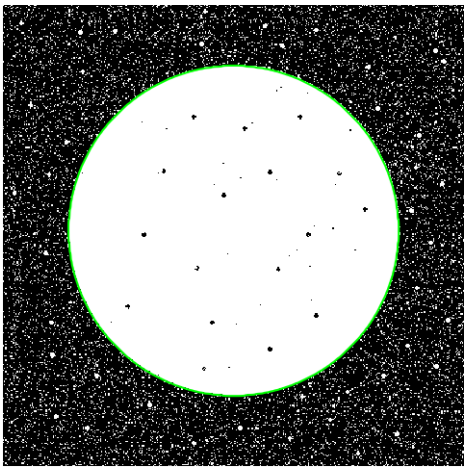
$$\frac{d\phi}{dt} = H'_{\epsilon}(\phi)\left(\nabla \cdot \left(\frac{\nabla\phi}{\|\nabla\phi\|}\right) + 2\lambda(f_{\tilde{\boldsymbol{I}}}(x) - H_{\epsilon}(\phi))\right) \tag{1.25}$$

We reproduce the example given in [**chan06**] to illustrate a situation in which the Chan-Vese method returns a local optimum solution. Assume that the observed image is the characteristic function of a ball of radius R, i.e., $f_{\tilde{\boldsymbol{I}}}(x) = \mathbf{1}_{B_R}(x)$. Moreover, we set the initial guess of the level-set function to be precisely the observed image, i.e., $\phi_0(x) = f_{\tilde{\boldsymbol{I}}}(x)$. The evolution given by Equation (1.25) maintains the radial symmetry of ϕ_0, which means that $\phi^{(t)}$ will represent the characteristic function of some ball of radius r. In other words, for this particular setting, Equation (1.24) is equivalent to

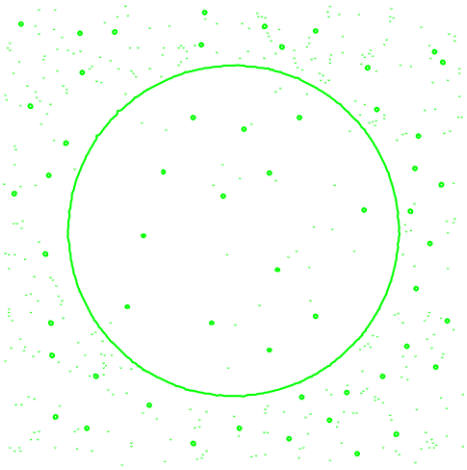
$$\min_r g(r,\lambda) = \min_r 2\pi r + \lambda\pi|R^2 - r^2|, \quad r \geq 0. \tag{1.26}$$

Equation (1.26) has a local minimum at $r = R$ and a local maximum at $r = 1/\lambda$. Depending on the value of λ, its global minimizer will be at $r = 0$ or $r = R$. For a fixed λ, let's consider the case in which $g(0,\lambda) < g(R,\lambda)$.

$$\begin{aligned} g(0,\lambda) &\leq g(R,\lambda) \\ \lambda\pi R^2 &\leq 2\pi R \\ R &\leq \frac{2}{\lambda} \end{aligned}$$



(a) Input image and contour of lowest energy for $\lambda = 1$



(b) Result of level-set method for $\lambda = 1$

Figure 1.5: **Level-sets return local optimum solutions.** Level-set result of Equation (1.23) stops at a local minimum in the right, instead of the green contour in the left.

Therefore, for $R \in (1/\lambda, 2/\lambda)$, the flow of Equation (1.25) is going to stop at the local minimum $r = R$, even the global minimum being located at $r = 0$. The parameter λ is set by the user to calibrate the denoise capabilities of the model. In the example of Figure 1.5, it indicates whether a ball of certain radius should be considered as an object or as a noise element. In the illustrated case, with $\lambda = 1$, the optimal solution consists in segmenting only the ball of largest radius in the picture, but Chan-Vese fails to converge to this solution.

In [**chan06**] the authors use an upper-level set representation to derive an equivalent model to Equation (1.23) but with the particular property that global optimum solution can be recovered by simple thresholding. The function u can be rewritten in terms of its upper level representation as

$$f_{\boldsymbol{I}}(x) = \int_0^1 \varphi(x, \mu) d\mu,$$

where $\varphi(x, \mu)$ is its μ-th upper level set, i.e.,

$$\varphi(x, \mu) = \mathbf{1}_{\{f_{\boldsymbol{I}} > \mu\}} = \begin{cases} 1, & f_{\boldsymbol{I}}(x) > \mu \\ 0, & \text{otherwise.} \end{cases}$$

Using the co-area formula we rewrite the total variation term as

$$\int_\Omega \|\nabla f_{\boldsymbol{I}}\| = \int_\Omega \int_0^1 \|\nabla \varphi(x, \mu)\| dx d\mu$$

and we can rewrite Equation (1.23) as

$$\min_{\varphi:\Omega\to\{0,1\}} E^{2-den} = \min_{\varphi} \int_{\Omega}\int_{0}^{1} \|\varphi(x,\mu)\| + (\mu - f_{\widetilde{\boldsymbol{I}}}(x))^2 \delta(f_{\boldsymbol{I}}(x) - \mu) dx d\mu \quad (1.27)$$

$$= \min_{\varphi} \int_{\Sigma} \|\varphi(x,\mu)\| + (\mu - f_{\widetilde{I}}(x))^2 |\partial_\mu \varphi(x,\mu)| d\Sigma, \quad (1.28)$$

where $\Sigma = [\Omega \times [0,1]]$. It happens that in the new formulation Equation (1.28), one can recover a binary solution from simple thresholding. From its relaxation

$$\min_{\varphi:\Omega\to[0,1]} E^{2-den} \quad (1.29)$$

we can once again rewrite E^{2-den} in terms of the upper level set representation of φ

$$E^{2-den} = \int_{\Sigma}\int_{0}^{1} \|\mathbf{1}_{\{\varphi>\gamma\}}\| + (\mu - f_{\widetilde{I}}(x))^2 |\partial_\mu \mathbf{1}_{\{\varphi>\gamma\}}| d\Sigma d\gamma$$

$$E^{2-den} = \int_{0}^{1} E^{2-den}(\mathbf{1}_{\{\varphi>\gamma\}}) d\gamma$$

Therefore, if $\varphi^\star$ is the solution of the convex relaxed problem Equation (1.29), $\mathbf{1}_{\{\varphi^\star>\gamma\}}$ is also an optimal solution of E^{2-den} for almost every choice of γ.

The functional lifting technique creates an equivalent higher dimensional model with the property that binary solutions can be easily recovered from its relaxed solution. In [**pock08**] this strategy is extended for multilabeling problems and in [**pock09**; **strekalovskiy12**] they are used to create a convex relaxation of the Mumford-Shah model.

To optimize the higher dimensional energy one could regularize the indicator and dirac delta functions in the same spirit of the Chan-Vese method, but it is usually preferable to use a convex optimization method that is suitable for non-differentiable functions as the proximal gradient [**chambolle04**], FISTA [**beck09a**] or the primal-dual [**chambolle11**] algorithm. The results are very satisfactory, but the running times very high.

Chapter 2

Discrete methods in Image Processing

In the beginning of the last chapter we described the Bayesian rational for the denoising problem. We turn once again to probability thinking, but restricting our analysis to discrete probabilistic models. Markov Random Fields are in the foundations of the methods discussed in this chapter, and an inspiration for many others. Image pixels are naturally interpreted as Markov states, and image properties, as spatial coherence, are encoded as potentials stored in the edges of the image grid graph. The MAP inference boils down to minimize the challenging class of pseudo-boolean functions.

In some fortunate cases, the functions can be minimized exactly and efficiently by a reduction to a max-flow (min-cut) problem, and it happens that such cases model imaging problems, as segmentation, nicely well. In fact, the minimum cut defines a partition, and one can interpret the cut as the contour separating two objects. It is the key for fruitful research that has followed. One can abstract the MRF machinery and define potentials on vertices and edges of the grid graph such that its minimum cut answers the problem being posed. Moreover, the potentials can model geometric properties of objects embedded in the grid graph, and one can use cuts to estimate the objects perimeters, for example.

We start this chapter by giving a brief description of Markov Random Fields and the minimization problem arising from the MAP inference. In the second section we present some properties of pseudo-boolean functions and how to optimize them. In the third section we describe the special class of submodular functions and efficient algorithms to compute the minimum of such functions. Finally, we describe successful models in the image processing community based on graph cuts and how one can inject geometric information in them.

2.1 Markov Random Fields

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ an undirected graph with vertices set $\mathcal{V}$ and edges set $\mathcal{E}$. The set of adjacent vertices to $v \in V$ is denoted $\mathcal{N}(v)$. Given two subsets $S, Q \subset \mathcal{V}$, a (S, Q)-cut is any subset of edges $\mathcal{E}' \subset \mathcal{E}$ such that S, Q are in different connected components in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E} \setminus \mathcal{E}')$. We denote $cut(S, Q)$ the set of all (S, Q) cuts.

For each vertex $v \in \mathcal{V}$ we associate a discrete random variable X_v that take values from a label set Γ_v according with some distribution P. We group all random variables in vector $\boldsymbol{X}$ and we write $\boldsymbol{X}_S$ to refer to the set of associated variables with vertex set $S \subset \mathcal{V}$. We also group all the label sets in the collection Γ. We denote $W_{\boldsymbol{X}}$ the set of all configurations for the random vector $\boldsymbol{X}$. We say that $\mathcal{H} = (\mathcal{G}, \boldsymbol{X}, \Gamma, P)$ is a *Markov Random Field* (MRF) if for any non-adjacent states X_u and X_v, the probability distribution P satisfies the independence conditions below

$$\textbf{Pairwise independencies:} \quad \Big\{ X_u \perp X_v \mid \{X_i,\ \forall i \in \mathcal{V} \setminus \{u, v\}\} \Big\} \tag{2.1}$$

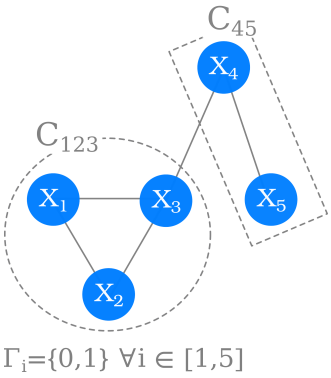
$$\textbf{Local independencies:} \quad \Big\{ X_u \perp X_v \mid \{X_i,\ \forall i \in \mathcal{N}(u)\} \Big\} \tag{2.2}$$

$$\textbf{Global independencies:} \quad \Big\{ \boldsymbol{X}_S \perp \boldsymbol{X}_Q \mid \boldsymbol{X}_Z,\ \text{where } \boldsymbol{X}_Z \in cut(S, Q) \Big\}, \tag{2.3}$$

where $X_u \perp X_v \mid \boldsymbol{X}_S$ means that variable X_u is independent of X_v given an assignment of variables in $\boldsymbol{X}_S$. For example, the MRF in Figure 2.1 respects the following expressions

$$\begin{aligned}
P\Big(X_1 = w_1 \,|\{X_i = w_i,\ i \neq 1\}\Big) &= P\Big(X_1 = w_1 \mid X_2 = w_2, X_3 = w_3\Big) \\
P\Big(X_3 = w_3 \,|\{X_i = w_i,\ i \neq 3\}\Big) &= P\Big(X_3 = w_3 \mid X_1 = w_1, X_2 = w_2, X_4 = w_4\Big) \\
P\Big(\boldsymbol{X}_{C_{123}} = \boldsymbol{w} \,|X_4 = w_4, X_5 = w_5\Big) &= P\Big(\boldsymbol{X}_{C_{123}} = \boldsymbol{w} \mid X_4 = w_4\Big),
\end{aligned}$$

where we use the shorter notation $\boldsymbol{X}_{C_{123}} = \boldsymbol{w}$ to denote some assignment of the random variables associated with the nodes of clique C_{123}. A clique is any complete subgraph of $\mathcal{G}$. Given a graph, it may be quite difficult to sort out a probability distribution that satisfies Equations (2.1) to (2.3). However, for the class of MRF that can be *factorized* in terms of the maximal cliques of $\mathcal{G}$, creating a valid probability distribution is straightforward with the help of *clique potential* functions.



ω					
X_1	X_2	X_3	X_4	X_5	$P(\omega)$
0	0	0	0	0	0.0238
0	0	0	0	1	0.0023
0	0	0	1	0	0.0011
0	0	0	1	1	0.0059
0	0	1	0	0	0.0238
0	0	1	0	1	0.0023
0	0	1	1	0	0.0011
0	0	1	1	1	0.0059
0	1	0	0	0	0.119
0	1	0	0	1	0.0119
0	1	0	1	0	0.0059
0	1	0	1	1	0.0297
0	1	1	0	0	0.0476
0	1	1	0	1	0.0047
0	1	1	1	0	0.0023
0	1	1	1	1	0.0119

ω					
X_1	X_2	X_3	X_4	X_5	$P(\omega)$
1	0	0	0	0	0.119
1	0	0	0	1	0.0119
1	0	0	1	0	0.0059
1	0	0	1	1	0.0297
1	0	1	0	0	0.0476
1	0	1	0	1	0.0047
1	0	1	1	0	0.0023
1	0	1	1	1	0.0119
1	**1**	**0**	**0**	**0**	**0.238**
1	1	0	0	1	0.0238
1	1	0	1	0	0.0119
1	1	0	1	1	0.0595
1	1	1	0	0	0.0952
1	1	1	0	1	0.0095
1	1	1	1	0	0.0047
1	1	1	1	1	0.0238

Figure 2.1: **Example of a Markov Random Field**. The nodes X_1, X_2, X_3 forms the 3-clique C_{123}.

2.1.1 Clique factorization and Gibbs energy

A distribution P_Φ is a *Gibbs distribution* if it can be parameterized by a set of factors $\Phi = \{\phi_1, \phi_2, \ldots \phi_m\}$, i.e.,

$$P_\Phi(\boldsymbol{X} = \boldsymbol{w}) = \frac{1}{Z} \prod_{i=1}^{m} \phi_i(\boldsymbol{w})$$

$$Z = \sum_{\boldsymbol{w} \in W_{\boldsymbol{X}}} \prod_{i=1}^{m} \phi_i(\boldsymbol{w})$$

For strictly positive distributions ($P(\boldsymbol{X} = \boldsymbol{w}) > 0 \ \forall \boldsymbol{w} \in W_{\boldsymbol{X}}$), the *Hammersley-Clifford* theorem [**koller09**] states that $(\mathcal{G}, \boldsymbol{X}, \Gamma, P)$ is a MRF if and only if P is a Gibbs distribution parameterized over complete subgraphs (cliques) of $\mathcal{G}$. Therefore, for MRF with strictly positive distributions P we can write

$$P(\boldsymbol{X} = \boldsymbol{w}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{w}) \tag{2.4}$$

$$Z = \sum_{\boldsymbol{w} \in W_{\boldsymbol{X}}} \prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{w}), \tag{2.5}$$

where $\mathcal{C}$ is the set of all cliques of $\mathcal{G}$. We define the *order* of such MRF as $\max_{C \in \mathcal{C}} |\mathcal{C}| - 1$, i.e., the size of the highest clique in $\mathcal{C}$ minus one. The second order MRF in Figure 2.1 was constructed by defining the following clique potentials

X_1	X_2	X_3	ϕ_{123}
0	0	0	1
0	0	1	5
0	1	0	5
0	1	1	10
1	0	0	5
1	0	1	10
1	1	0	10
1	1	1	20

X_3	X_4	ϕ_{34}
0	0	100
0	1	50
1	0	20
1	1	10

X_4	X_5	ϕ_{45}
0	0	100
0	1	10
1	0	10
1	1	50

For strictly positive distributions we also have that the global independencies in Equation (2.3) are equivalent to the pairwise and local independencies [**koller09**].

2.1.2 Hidden Markov model

Imaging problems are naturally coupled with a set of observations, namely the color intensities of each pixel. We can expect that such observations play a role in any probabilistic model pretending to solve an imaging problem. The *Hidden Markov Model* (HMM) is a subclass of MRF that incorporates such observed variables in its definition and is quite often used in the image processing literature.

Definition 1(Hidden Markov Model): A Hidden Markov Model is a MRF $\mathcal{H} = (\mathcal{G}, \boldsymbol{X} \cup \boldsymbol{Y}, \Gamma_X \cup \Gamma_Y, P)$ such that

$$\begin{aligned} Y &= \{Y_i \mid X_i \in \boldsymbol{X}\} \\ \forall i \neq j, \quad & Y_i \perp X_j \mid X_i \\ \forall i \neq j, \quad & Y_i \perp Y_j. \end{aligned}$$

We are going to be interested in problems arising from the setting in which the states of random variables in $\boldsymbol{Y}$ are known, and one wishes to infer the states of random variables in $\boldsymbol{X}$. In other words, we are interest to find $\boldsymbol{X}^\star$ such that

$$\boldsymbol{w}^\star = \arg\max_{\boldsymbol{w}} = P(\boldsymbol{X} = \boldsymbol{w} \mid \boldsymbol{Y}). \tag{2.6}$$

The vector $\boldsymbol{Y}$ is called the vector of observed variables, and in image problems they are usually associated to the color intensity of pixels.

The set of labels $\Gamma_{\boldsymbol{X}}$ is defined according to the problem. In segmentation it could represent the different partitions in which to segment the image, e.g. a label to encode vehicles, another to encode pedestrians, a third to encode the sky and so on. In stereo, the labels could mean the relative depth of the object with respect to the others. In denoising and reconstruction problems in general, it could be the color intensities themselves.

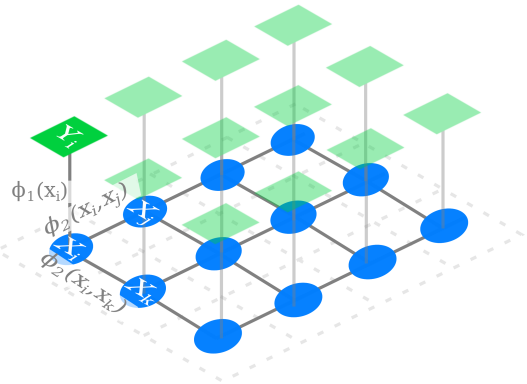


Figure 2.2: **HMM and grid graphs**. A typical HMM in imaging using a 4-neighborhood system.

2.1.3 Grid graph and Tikhonov denoising revisited

Let $\boldsymbol{I} \in \mathbb{F}^{m \times n}$ a grayscale bidimensional image. We denote $p \in \boldsymbol{I}$ a pixel of $\boldsymbol{I}$. We define its undirected *grid graph* $\mathcal{G}_{\boldsymbol{I}}(\mathcal{V}, \mathcal{E})$ as

$$\begin{aligned} \mathcal{V} &= \{v_p \mid p \in \boldsymbol{I}\} \\ \mathcal{E} &= \big\{\{v_p, v_q\} \mid p \in \boldsymbol{I} \text{ and } q \in \mathcal{N}_k(p)\big\}, \end{aligned}$$

where $\mathcal{N}_k(p)$ is the k-neighborhood of pixel p. Common values for k are 4 and 8, i.e.,

$$\begin{aligned} \mathcal{N}_4(p) &= \{p + (i,j) \mid |i| \leq= 1, |j| \leq 1, |i+j| = 1\} \\ \mathcal{N}_8(p) &= \{p + (i,j) \mid |i| \leq= 1, |j| \leq 1\}. \end{aligned}$$

In the grid graph $\mathcal{G}_I$ we have 1 and 2-cliques only. We attach to $\mathcal{G}_I$ the HMM $\mathcal{H} = (\mathcal{G}_I, \boldsymbol{X} \cup \boldsymbol{Y}, \Gamma_{\boldsymbol{X}}, \Gamma_{\boldsymbol{Y}}, P)$ in which the random variables take values over the grayscale levels of the image, grouped in $\Gamma_{\boldsymbol{X}} = \Gamma_{\boldsymbol{Y}} = \{i \mid 0 \leq i \leq 255\}$. The Gibbs distribution P is given by Equations (2.4) and (2.5) with clique potentials defined as

$$\begin{aligned} \psi_p(\gamma_p) &= \tfrac{\lambda}{2}\left(f_{\widetilde{\boldsymbol{I}}}(p) - \gamma_p\right)^2. \\ \psi_{pq}(\gamma_p, \gamma_q) &= (\gamma_p - \gamma_q)^2. \end{aligned}$$

$$\begin{aligned} \phi_p(\boldsymbol{x}_p = \gamma_p) &= \exp\big(-\psi_1(\gamma_p)\big), && \forall p \in \boldsymbol{I} \\ \phi_{pq}(\boldsymbol{x}_p = \gamma_p, \boldsymbol{x}_q = \gamma_q) &= \exp\big(-\psi_2(\gamma_p, \gamma_q)\big), && \forall p \in \boldsymbol{I} \text{ and } q \in \mathcal{N}_k(p). \end{aligned}$$

In Figure 2.2 we have a representation of this HMM, which encodes the Tikhonov denoising model of Section 1.2. The estimated image $\widehat{I}$ is computed as the maximum

likelihood of P, i.e.,

$$\begin{aligned}\widehat{\boldsymbol{I}} &= \arg\max_{\boldsymbol{I}'} P(\boldsymbol{X} = \boldsymbol{I}' \mid \boldsymbol{Y} = \boldsymbol{I}) \\ &= \arg\min_{\boldsymbol{I}'} E(\boldsymbol{I}') \\ &= \arg\min_{\boldsymbol{I}'} \sum_{p\in\boldsymbol{I}} \psi_p\big(\boldsymbol{I}'(p)\big) + \frac{1}{2} \sum_{\substack{p\in\boldsymbol{I} \\ q\in\mathcal{N}_k(p)}} \psi_{pq}\big(\boldsymbol{I}'(p), \boldsymbol{I}'(q)\big),\end{aligned} \tag{2.7}$$

the $\frac{1}{2}$ factor is necessary in order to compensate double counting of ψ_{pq}. The energy E to be minimized is called the *Gibbs energy*.

The success of this approach depends on our capacity to minimize the Gibbs energy in Equation (2.7). For the Tikhonov multilabel HMM, it can be computed exactly and efficiently [**ishikawa03**]. However, MAP inference in general multilabel HMMs is NP-hard. The scenario is a little bit better for binary HMMs, as we are going to see in the next section.

2.1.4 Potts and Ising models

Let $\boldsymbol{I} \in \mathbb{F}^{m\times n}$ a grayscale image to which we associated its grid graph $\mathcal{G}_I(\mathcal{V}, \mathcal{E})$. We wish to denoise image $\boldsymbol{I}$, but instead of using the Tikhonov regularization term, we want to use one that preserves discontinuities. An intermediate step would be to minimize the naive discrete version of total variation, which can also be efficiently minimized. In this case, the Gibbs energy is given by

$$E(\boldsymbol{x}) = \frac{\lambda}{2} \sum_{p\in\boldsymbol{I}} (\widetilde{I}(p) - x_p)^2 + \sum_{\substack{p\in\boldsymbol{I}, \\ q\in\mathcal{N}(p)}} |x_p - x_q|. \tag{2.8}$$

The term in the right of Equation (2.8) is called the discrete anisotropic total variation. As its continuous version, the discrete total variation performs better than Tikhonov for imaging problems, but it still not considered a discontinuity preserving function and it presents some undesirable side effects (see Figure 2.3). On the other hand, truncated functions are discontinuity preserving. For some $K > 0$, consider the following Gibbs energy

$$\phi_{pq}(x_p, x_q) = \begin{cases} K, & \text{if } x_p \neq x_q \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

$$E(\boldsymbol{x}) = \sum_{p\in\boldsymbol{I}} \phi_p(x_p) + \sum_{\substack{p\in\boldsymbol{I}, \\ q\in\mathcal{N}(p)}} \phi_{pq}(x_p, x_q). \tag{2.10}$$

(a) Noisy image (b) $|x_p - x_q|$ (c) $\min(3, |x_p - x_q|)$

Figure 2.3: **Comparison between convex and truncated potentials [blake11markov](chapter 3)**. Truncated potentials are discontinuity preserving functions, and are suitable for imaging problems.

Equations (2.9) and (2.10) defines a *Potts model*. Minimizing Equation (2.10) is NP-hard [**boykov01fast**] if variables x_p, x_q are not binary, i.e., the underneath HMM is a multilabel one. In the binary case, Equations (2.9) and (2.10) are referred as the *Ising model*, and this one can be minimized efficiently.

In fact, MAP inference in multilabel HMMs are much more difficult than MAP inference in binary HMMs. At first glance, the multilabeling extension does not seem to be an issue, as we can always transform a multilabel problem in a binary one by including as many as $\log_2 |\Gamma_{\boldsymbol{X}}|$ new variables. The difficulty is that the resulting energy is very likely to lie in a class of binary energies whose minimization is NP-hard [**ramalingam08**]. Therefore, MAP inference in multilabel HMMs is not likely to be solved exactly in an efficient manner. Instead, approximation algorithms as (α, β)-swap [**boykov01fast**] or range moves [**veksler07graph**] are used.

In the next two sections we are going to focus on the analysis of binary first-order HMMs, i.e., MRF that are encoded by $1, 2$-cliques potentials and binary random variables. This restriction is justified by two reasons: first because there exist efficient algorithms to solve them (see Table 2.1); and second because of its generality. A general MRF can be transformed into an equivalent binary first-order MRF by including a sufficient number of auxiliary variables. Naturally, such transformations are not always useful. In the multilabel case the derived energy is almost surely non-submodular and in the case of high-order cliques, the inclusion of auxiliary variables may turn the minimization problem impractical [**ishikawa10**].

In binary first-order MRFs, the clique potentials maps binary vectors to real values and the energy belongs to the class of *quadratic pseudo-boolean* functions, which can be written as

$$f(x_1, \cdots, x_n) = \sum_{j<n} c_j x_j + \sum_{j<k<n} c_{jk} x_j x_k.$$

MRF Order	Graph topology	Function class	
		Submodular	**Non-submodular**
0^{th} order	All topologies	Greedy algorithm	
1^{st} order	1-connected	Viterbi algorithm [**viterbi67**], Belief Propagation [**pearl82**]	
	$\geq$ 2-connected	Max-Flow: $O(\|V\|^2\|E\|)$ [**kolmogorov04whatenergies**]	NP-Hard [**nemhauser81**]
2^{nd} order	All topologies	Max-Flow: $O(\|V\|^2\|E\|)$ [**billionnet85**]	
Higher order	All topologies	$O(n^5Q + n^6)$ [**orlin09faster**]	

Table 2.1: **MAP inference algorithms for binary MRF of different orders**. In 0^{th} order HMM, a simple Greedy algorithm computes the MAP inference. In 1^{st} order HMM we have linear algorithms based on dynamic programming for 1-connected graphs and max-flow for other topologies. For orders greater than 2, MAP inference can be computed efficiently if the 2-clique potential is submodular. Notice, however, that the best algorithm for orders greater than 2 is quite expensive (Q denotes the time to evaluate the function). Submodular functions is a special class of Pseudo-Boolean functions discussed in Section 2.2 that can be minimized in polynomial time, in constrast with non-submodular, which in this case is proven to be a problem in NP-hard. There are other classes of pseudo-boolean functions that are efficiently optimized, some of them are described in [**boros02pseudo**]

In the next section, we explore the class of pseudo-boolean functions and how to optimize them.

2.2 Pseudo-boolean functions

Definition 1(Pseudo-boolean function): A pseudo boolean function $f : \{0,1\}^n \to \mathbb{R}$ is a function that maps binary vectors to real values.

Pseudo-boolean functions (PBF) can be interpreted as *set functions*, i.e., functions that map sets to real values. Indeed, a binary vector of n elements is in bijection with the power sets of $V = \{1, 2, \cdots, n\}$. Therefore, the PBF $f : \{0,1\}^n \to \mathbb{R}$ can be written as

$$f(x_1, \cdots, x_n) = \sum_{S \subset 2^V} c_S \prod_{i \in S} x_i. \tag{2.11}$$

where $c_S \in \mathbb{R}$ is a coefficient associated to each subset S. Equation (2.11) is denoted the *polynomial form* of PBF f. The *order* of a PBF is defined as the cardinality of the largest subset S in which $c_S > 0$. The polynomial form is **unique**.

Sometimes it is convenient to express f in its so called *posiform* representation, usually denoted by ϕ instead of f. In its posiform representation we use literals

$x_i, \bar{x}_i \in L$ to represent states $1, 0$ of indexed variable i and all coefficients are positive except the one associated to the empty set.

$$\phi(f) = \phi(x_1, \cdots, x_n) = \sum_{T \subset 2^L} a_T \prod_{p \in T} p, \tag{2.12}$$

where $a_T \geq 0$ whenever $T \neq \emptyset$. The posiform representation can be seen as a truth table with positive values for all configurations, except the one in which all variables are set to zero. One can pass from posiform to polynomial representation by exchanging $\bar{x}_i$ and $(1 - x_i)$. A posiform is said *maximum-constant* if its constant term $C(\phi)$ is maximum among all posiforms representing f. A maximum-constant posiform is denoted $\phi^\star$. Both general and maximum-constants posiforms are **not unique** representations of the PBF f.

Example 1: The polynomial form and two possible posiform representations for the same PBF.

$$\begin{aligned} f(x_1, x_2) &= a_\emptyset + a_{\bar{1}} + a_{\bar{2}} + a_{\bar{1}\bar{2}} + \big(a_1 - a_{\bar{1}} + a_{1\bar{2}} - a_{\bar{1}\bar{2}}\big)x_1 \\ &\quad + \big(a_2 - a_{\bar{2}} + a_{\bar{1}2} - a_{\bar{1}\bar{2}}\big)x_2 \\ &\quad + \big(a_{12} - a_{\bar{1}2} - a_{1\bar{2}} + a_{\bar{1}\bar{2}}\big)x_1 x_2 \end{aligned}$$

$$\begin{aligned} \phi_1(x_1, x_2) &= a_\emptyset + a_1 x_1 + a_{\bar{1}}\overline{x_1} + a_2 x_2 + a_{\bar{2}}\overline{x_2} \\ &\quad + a_{12} x_1 x_2 + a_{\bar{1}2}\overline{x_1} x_2 + a_{1\bar{2}} x_1 \overline{x_2} + a_{\bar{1}\bar{2}} x_1 \overline{x_2} \end{aligned}$$

$$\begin{aligned} \phi_2(x_1, x_2) &= (a_\emptyset - c) + (a_1 + c)x_1 + (a_{\bar{1}} + c)\overline{x_1} + a_2 x_2 + a_{\bar{2}}\overline{x_2} \\ &\quad + a_{12} x_1 x_2 + a_{\bar{1}2}\overline{x_1} x_2 + a_{1\bar{2}} x_1 \overline{x_2} + a_{\bar{1}\bar{2}} x_1 \overline{x_2} \end{aligned}$$

Classical problems in combinatorial optimization as vertex cover, maximum independent set, 3-SAT and many others are formulated as PBF optimization problems. The mentioned problems are in NP-complete, so we can expect that the optimization of a general PBF is a task that is unlikely to be solved efficiently. Nonetheless, we can investigate subclasses of PBF in which the optimum can be found efficiently.

2.2.1 PBF optimization

We first observe that optimizing a PBF of order n can be transformed into an equivalent quadratic PBF optimization problem by creating extra variables and penalty terms. For example, let $x, y, w, z \in \{0, 1\}$. Then,

$$\begin{aligned} z = xy &\leftrightarrow xy - 2xz - 2yz + 3x = 0 \\ z \neq xy &\leftrightarrow xy - 2xz - 2yz + 3x > 0 \end{aligned}$$

Therefore

$$\begin{aligned} \min f(x,y,w) &= \min g(x,y,w,z) \\ \min xyw &= \min zw + xy - 2xy - 2yz + 3x. \end{aligned}$$

This procedure can be extended and formally described in a polynomial time algorithm that transforms an arbitrary PBF f in a quadratic PBF g with the property that $\min f = \min g$ [**boros02pseudo**]. We remark, however, that the transformation may add a prohibitive number of auxiliary variables, turning the minimization problem impractical. With that in mind, we are going to restrict our analysis to the optimization of quadratic PBFs.

Roof duality

A quite natural and naive approach to optimize Equation (2.11) would be to formulate the quadratic PBF as a continuous linear programming. Consider the quadratic PBF

$$f(\boldsymbol{x}) = \left\{ \begin{array}{rl} \min & c_0 + \sum c_i x_i + \sum c_{ij} x_i x_j \\ \text{subject to} & \boldsymbol{x} \in \{0,1\}^n. \end{array} \right. \tag{2.13}$$

Equation (2.13) can be linearized by substituting each pairwise term $x_i x_j$ with binary variable z_{ij} and including the following set of constraints

$$R(\boldsymbol{x},\boldsymbol{z}) = \left\{ \begin{array}{l} z_{ij} \leq x_i, \\ z_{ij} \leq x_j, \\ z_{ij} \geq x_i + x_j - 1 \end{array} \middle| \; \forall 0 < i < j < n \right\}$$

Therefore, an equivalent linear integer programming formulation is

$$\begin{array}{rl} \min & c_0 + \sum c_i x_i + \sum c_{ij} z_{ij} \\ \text{subject to} & \boldsymbol{x}, \boldsymbol{z} \in \{0,1\}^n \\ & R(\boldsymbol{x},\boldsymbol{z}) \end{array} \tag{2.14}$$

Finally, the relaxation of Equation (2.14) gives

$$g(\boldsymbol{x},\boldsymbol{z}) = \left\{ \begin{array}{rl} \min & c_0 + \sum c_i x_i + \sum c_{ij} z_{ij} \\ \text{subject to} & \boldsymbol{x}, \boldsymbol{z} \in [0,1]^n \\ & R(\boldsymbol{x},\boldsymbol{z}) \end{array} \right.$$

Clearly, formulation G is a lower bound of f, i.e., $g(\boldsymbol{x},\boldsymbol{z}) \leq f(\boldsymbol{x})$. Such lower bound is called the *roof dual* (it was originally defined for a maximization problem) and it is shown [**hammer84**] to be equivalent to

$$g(\boldsymbol{x},\boldsymbol{z}) = C_{\phi^\star(f)},$$

i.e., the constant in the max-constant posiform representation of f. In this same work, the so called *strong persistency* theorem is proven. It says that for every unary term $a_p p$ (p a literal) in a max-constant posiform representation of f, we have that $p = 0$ in every solution of $\min f$.

Example 2: Consider the quadratic PBF

$$f(x_1, x_2, x_3) = 6 - x_1 - 4x_2 - x_3 + 3x_1x_2 + x_2x_3.$$

It can be shown that its roof dual equals 2. A possible max-constant posiform representation for f is

$$\phi^\star(f) = 2 + x_1 + \bar{x}_2 + x_1x_2 + 2\bar{x}_1\bar{x}_2 + \bar{x}_2\bar{x}_3$$

According with the strong persistency theorem, $x_1 = 0, \bar{x}_2 = 0$ for every solution of $\min f$. Replacing this values in f we have

$$\begin{aligned} \min f(x_1, x_2, x_3) &= \min f(x_1 = 0, \bar{x}_2 = 0, x_3) \\ &= \min 6 - 4 - x_3 + x_3 \\ &= 2. \end{aligned}$$

The strong persistency theorem allow us to fix some variables of the quadratic PBF, which could possibly result in a simpler optimization problem. Clearly, the main difficult is to find the max-constant posiform $\phi^\star(f)$ that results in a maximum number of variable elimination. Such posiform, called the *master posiform*, can be found by computing the maximum flow in some capacitated graph.

Master posiform and max-flow reduction

We present a construction given in [**boros91**; **boros02pseudo**] of a capacitated graph $G_\phi(\mathcal{V}, \mathcal{E}, c)$ that encodes some posiform ϕ with constant $C(\phi) = 0$. The authors showed how to derive the max-constant posiform from the computation of the maximum flow of G_ϕ. Let ϕ be a posiform of the form

$$\phi = \sum_{p \in L} a_p p + \sum_{p,q \in L} a_{pq} pq,$$

with $a_p > 0, a_{pq} > 0 \; \forall p, q \in L$. We construct the capacitated graph $G_\phi(\mathcal{V}, \mathcal{E}, c)$ such that each term in the sum is encoded by two edges. Each unary term with literal p have one edge from the source s to its negated literal vertex $v_{\bar{p}}$ and one edge from v_p to the target vertex $\bar{s}$. The source and target vertices are identified with the 1 and 0 constants, respectively.

Similarly, pairwise terms with literals p, q have edges $(v_p, v_{\bar{q}})$ and $(v_q, v_{\bar{p}})$. The full graph is given by

$$\mathcal{V} = \{v_p \mid p \in L\} \cup \{s, \bar{s}\}$$

$$\mathcal{E} = \big\{(s, v_{\bar{p}}), (v_p, t) \mid \forall a_p > 0\big\} \cup \big\{(v_p, v_{\bar{q}}), (v_q, v_{\bar{p}}) \mid \forall a_{pq} > 0\big\}$$

$$c(\,(v_p, v_q)\,) = c_{pq} = \begin{cases} a_p/2, & \text{if } v_p \notin \{s, \bar{s}\} \text{ or } v_q \notin \{s, \bar{s}\} \\ a_{pq}/2, & \text{if } v_p, v_q \notin \{s, \bar{s}\} \\ 0, & \text{otherwise.} \end{cases}$$

A construction of graph G_ϕ is illustrated in Figure 2.4a. Therefore, the posiform ϕ can also be written as $\phi = \sum_{(v_p, v_q) \in \mathcal{E}} c_{p\bar{q}}$. In fact, it is possible to show that there is a bijection between the posiform with zero constant ϕ and the graph G_ϕ.

A flow is a function $\varphi : \mathcal{E} \to \mathbb{R}_+$ that satisfies

$$\begin{aligned} \varphi(\,(v_p, v_q)\,) = \varphi(p, q) < \;& c_{pq}, && \forall v_p, v_q \in \mathcal{V} \\ \sum_{v_p \in \mathcal{V}} \varphi(p, q) = \;& \sum_{v_p \in \mathcal{V}} \varphi(q, p), && \forall v_q \in \mathcal{V}. \end{aligned}$$

A flow $\varphi^\star$ is said to be maximum if

$$\varphi^\star = \arg\max_{\varphi} \sum_{v_p \in \mathcal{V}} \varphi(s, v_p),$$

i.e., if the flow leaving the source is maximum. The residual graph of G_ϕ with respect to some flow φ is denoted $G_{\phi[\varphi]}(\mathcal{V}, \mathcal{E}^+, r)$ and owns the same set of vertices of G_ϕ. The set of edges is extended to include returning edges as well, i.e.,

$$\mathcal{E}^+ = \mathcal{E} \cup \{(v_q, v_p) \mid (v_p, v_q) \in \mathcal{E}\}.$$

The edges cost is given by the residual function r

$$r(\,(v_p, v_q)\,) = r_{pq} = \begin{cases} c_{pq} - \varphi(p, q), & (v_p, v_q) \in \mathcal{E} \\ \varphi(p, q), & (v_p, v_q) \in \mathcal{E}^+ \setminus \mathcal{E}. \end{cases}$$

One can also construct a posiform from the residual graph $G_{\phi[\varphi]}$. In this case, the posiform is denoted $\phi_{G_{\phi[\varphi]}}$. We remark that edges arriving in the source or leaving the target are all mapped to 0, as the source and the target are identified with the constants $1, 0$ respectively. For example, (p, s) is encoded as $p\bar{s} = 0$.

The *Ford-Fulkerson* algorithm computes the maximum flow by incrementing an initial zero flow function $\varphi_0 = 0$ every time an *augmenting path* is found. The k-th augmenting path is a path $\pi_k = (p_0 = s, p_1, p_2, \cdots, p_n, p_{n+1} = \bar{s})$ in the residual graph $G_{\phi[\varphi_{k-1}]}$ in which all edges of π_k have a positive residual value. We say that π_k is an ϵ-augmenting path if

$$\epsilon = \min_{p_i \in \pi_k \setminus \bar{s}} r_{p_i, p_{i+1}}.$$

Proposition 1(Residual graph to posiform): Given a posiform

$$\phi = C(\phi) + \sum_{p \in L} a_p p + \sum_{p,q \in L} a_{pq} pq,$$

construct its corresponding capacitated graph G_ϕ and compute its maximum flow executing the Ford Fulkerson algorithm. Then, for every step k of the algorithm we have that

$$\phi = C(\phi) + \nu(\varphi_k) + \phi_{G_{\phi[\varphi_k]}}.$$

Proof:

We observe that every ϵ-augmenting path $\pi_k = s, v_{p_1}, v_{p_2}, \cdots, v_{p_n}, \bar{s}$ in $G_{\phi[\varphi_k]}$ encodes an alternating sum of literals of the form

$$\begin{aligned}\phi_{\pi_k} =& a_1\bar{p}_1 + a_{1\bar{2}}p_1\bar{p}_2 + a_{2\bar{3}}p_2\bar{p}_3 + \cdots + a_{n-1\bar{n}}p_{n-1}\bar{p}_k + a_n p_n \\ =& \epsilon(\bar{p}_1 + p_1\bar{p}_2 + p_2\bar{p}_3 + \cdots + p_{n-1}\bar{p}_n + p_n) + \phi',\end{aligned}$$

where

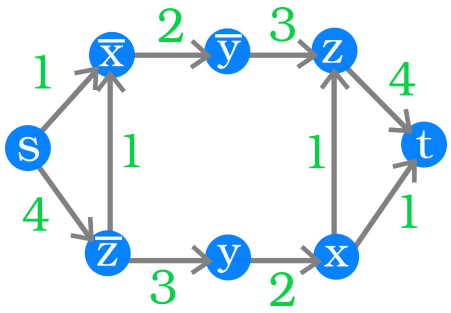
$$\phi' = (a_{\bar{1}} - \epsilon)\bar{p}_1 + (a_{1\bar{2}} - \epsilon)p_1\bar{p}_2 + (a_{2\bar{3}} - \epsilon)p_2\bar{p}_3 + \cdots + (a_{n-1\bar{n}} - \epsilon)p_{n-1}\bar{p}_n + (a_n - \epsilon)p_n.$$

By observing that

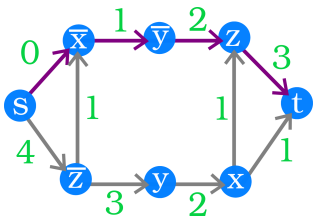
$$\begin{aligned}\bar{p}_1 + p_1\bar{p}_2 &= 1 - p_1p_2 \\ -\bar{p}_{j-1}p_j + p_j\bar{p}_{j+1} &= \bar{p}_{j-1}p_j - p_jp_{j+1} \\ -p_{n-1}p_n + p_n &= \bar{p}_{n-1}p_n,\end{aligned}$$

we can rewrite the alternating sum as

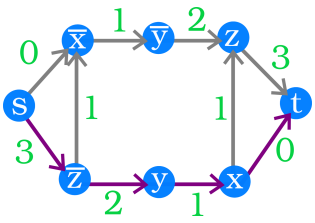
$$\begin{aligned}\phi_\pi &= \epsilon + \epsilon(\bar{p}_1p_2 + \bar{p}_2p_3 + \cdots + \bar{p}_{n-1}p_n) + \phi' \\ &= \psi + \phi',\end{aligned}$$



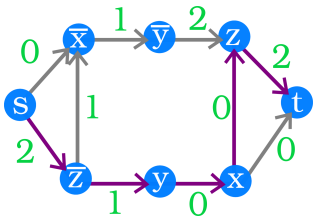
(a) Initial graph G_ϕ for $\phi = 2x + 8z + 2x\bar{z} + 4\bar{x}y + 6\bar{y}\bar{z}$



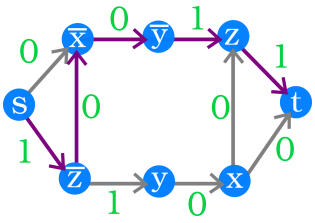
(b) 1-Augmenting path 1



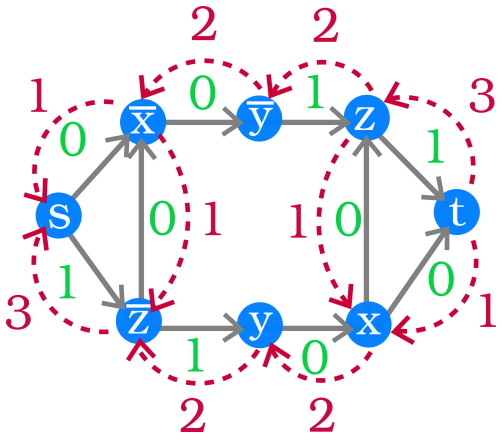
(c) 1-Augmenting path 2



(d) 1-Augmenting path 3



(e) 1-Augmenting path 4



(f) Final residual graph $G_{\phi[\varphi^\star]}$. The master posiform is written as $\phi^\star = 4 + 2z + 2\bar{z}\bar{y} + 4zy + 4\bar{y}x + 2z\bar{x}$

Figure 2.4: **Example of max-flow to find the master posiform**. The Ford Fulkerson algorithm is executed for the capacitated graph representation of posiform ϕ. In (a), the initial capacitated graph; a sequence of augmenting paths (we omit the returning edges) are shown in figures (b-e); the final residual graph is shown in figure (f).

Note that ϕ' and ψ corresponds, respectively, to the update of the residual costs for the edges of π_k and the updates of its returning edges. Therefore, we can write

$$\phi = C(\phi) + \nu(\varphi_k) + \phi_{G_{\phi[\varphi_k]}},$$

i.e., the initial posiform ϕ can be rewritten as a constant plus the posiform corresponding to the residual graph at step k of the Ford Fulkerson algorithm. ■
It follows that the master posiform is given by

$$\phi^\star = C(\phi) + \nu(\varphi^\star) + \phi_{G_{\phi[\varphi^\star]}}.$$

Example 3: The posiform $\phi = 2x+8z+2x\bar{z}+4\bar{x}y+6\bar{y}\bar{z}$ is represented by the graph G_ϕ in Figure 2.4. Its maximum flow value equals to 4 and the master posiform is given by $\phi^\star = \nu(\varphi^\star) + \phi_{G_{\phi[\varphi^\star]}} = 4 + 2z + 2\bar{z}\bar{y} + 4zy + 4\bar{y}x + 2z\bar{x}$.

From the strong persistency theorem we conclude that $z = 0$, and we have

$$\min \phi = \min \phi^\star(z = 0) = \min 2\bar{y} + 4\bar{y}x.$$

We can say more. Let U be the set of literals that are reached from the source by a path of positive residual in the final residual graph. Then, a solution $\boldsymbol{x} \in \arg\min f$ must agree with $x(U = 1)$ [**boros02pseudo**]. Therefore,

$$\min \phi = \min \phi^\star(\bar{z} = 1, y = 1) = \min 4 = 4.$$

In fact, by looking at all configurations of ϕ, we observe that $(z = 0, y = 1)$ in every minimum configuration of f.

x	y	z	f
0	0	0	6
0	0	1	8
0	**1**	**0**	**4**
0	1	1	12
1	0	0	10
1	0	1	10
1	**1**	**0**	**4**
1	1	1	10

The construction above is the core of the *QPBO* algorithm [**boros91**] that finds partial solutions of general quadratic PBF. The labeled variables by QPBO are guaranteed to belong to an optimum solution. This last property is usually referred as the *partial optimality property* of QPBO, and it is a consequence of the strong persistency theorem. In the most cases, the master posiform allows us to eliminate only few variables, but if the function f is a *submodular* function, QPBO finds the minimum of f.

2.2.2 Submodularity

Definition 2(Submodular set function):
Let V be a set with n elements, e.g., $V = \{1, 2, \cdots, n\}$. A set function $f : 2^V \to \mathbb{R}$ is submodular if

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y), \quad \forall X, Y \subset V. \tag{2.15}$$

An equivalent local definition is given by

$$f(X \cup \{x_1\}) + f(X \cup \{x_2\}) \geq f(X \cup \{x_1, x_2\}) + f(X), \ \forall X \subset V \text{ and } \{x_1, x_2\} \not\subset X. \tag{2.16}$$

Proposition 2(Quadratic submodular PBF): Let $f : 2^V \to \mathbb{R}$ a quadratic PBF written as

$$f(x_1, \cdots, x_n) = C + \sum_{i<n} f_i(x_i) + \sum_{i<j<n} f_{ij}(x_i, x_j).$$

Then, the statements below are equivalent

i Function f is submodular.

ii $f_{ij}(0,1) + f_{ij}(1,0) \geq f_{ij}(0,0) + f_{ij}(1,1), \quad \forall i < j < n$

iii $\frac{\partial^2 f}{\partial x_i \partial x_j} \leq 0, \quad \forall i < j < n$

Proof:
(**i** $\to$ **ii**): Immediately from Equation (2.16).
(**ii** $\to$ **iii**): Writing down the terms in f for variables x_i, x_j we have

$$f_{ij}(0,0)(1-x_i)(1-x_j) + f_{ij}(1,1)x_i x_j + f_{ij}(0,1)(1-x_i)x_j + f_{ij}(1,0)x_i(1-x_j).$$

Taking its second derivative

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = f_{ij}(0,0) + f_{ij}(1,1) - f_{ij}(0,1) - f_{ij}(1,0) \leq 0.$$

(**iii** $\to$ **i**): We define

$$\begin{aligned}\Delta_{x_i} &= f(x_1, \cdots, x_{i-1}, x_i = 1, x_{i+1}, \cdots, x_n) - f(x_1, \cdots, x_{i-1}, x_i = 0, x_{i+1}, \cdots, x_n) \\ &= f(X \cup \{x_i\}) - f(X), \quad \text{where } x_i \notin X.\end{aligned}$$

Therefore

$$f(X \cup \{x_1, x_2\}) = f(X) + \Delta_{x_1} + \Delta_{x_2} + \frac{\partial^2 f}{\partial x_i \partial x_j}$$
$$f(X \cup \{x_1, x_2\}) = -f(X) + f(X \cup \{x_1\}) + f(X \cup \{x_2\}) + \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

Finally,

$$f(X \cup \{x_1, x_2\}) + f(X) - f(X \cup \{x_1\}) - f(X \cup \{x_2\}) \leq 0.$$

■

Supermodular functions, on the other hand, are functions that satisfy

$$f(X) + f(Y) \leq f(X \cup Y) + f(X \cap Y), \quad \forall X, Y \subset V.$$

Therefore, if f is submodular, $-f$ is supermodular. The QPBO algorithm efficiently computes the minimum (maximum) of submodular (supermodular) functions. The opposite problem, i.e., maximizing (minimizing) a submodular (supermodular) function is in NP-hard.

There are functions which are neither submodular nor supermodular and are called *non-submodular*. Optimizing a non-submodular function is in NP-hard but QPBO can be used to find a partially optimal solution, as noticed previously. Nonetheless, it is likely the case that QPBO will leave several variables unlabeled while minimizing a non-submodular energy. To attenuate this problem, two variations of QPBO were proposed in [**rother07qpbo**]

> **QPBO-Probe**: Implements a branch-and-bound technique in an attempt to increase the number of variables labeled by QPBO. It keeps the partial optimality property.
>
> **QPBO-Improve**: An heuristic that is guaranteed to return a labeling of lower or equal value than the labeling giving by QPBO. The partial optimality property is lost.

As discussed in Sections 2.1 and 2.2, several problems in image processing can be modeled in terms of an HMM $\mathcal{H}$ defined over the image grid graph. The hidden states of $\mathcal{H}$ are often estimated as the maximum likelihood of the induced probability distribution given the set of observed variables $\boldsymbol{Y}$. In that occasion, we mention that the success of this approach depended on the difficulty of the maximum likelihood computation. We observed that the general problem is NP-Hard for multilabel

HMM, and we turn our attention to binary ones. In this case, we have to solve a PBF optimization problem.

As we have seen, the optimization of a PBF is closely related to the computation of maximum flows, or equivalently, minimum cuts in a graph. In fact, in some applications such as binary segmentation, it is more practical to think in terms of a minimum cut problem than in terms of an HMM one. Both interpretations are equivalent and trigger different insights, but the graph cut mindset fits very nicely in the binary segmentation framework and it gives to us an implicit representation of a digital contour, which can be exploited to incorporate geometric information in the model.

2.3 Graph cut models

Graph cut techniques in imaging were pioneered by [**greig89exact**] and became popular after the interactive binary segmentation model proposed by [**boykov01graphcut**]. In this section, we are going to describe the latter and two other applications of graph cut techniques. The first explains how to set the cost function to estimate perimeter of segmented shapes and the second one extends the first by incorporating a constraint that helps in the segmentation of thin and elongated objects.

2.3.1 Binary segmentation

Let $\boldsymbol{I} \in \mathbb{F}^{m\times n}$ a discrete grayscale image and $\mathcal{G}_{\boldsymbol{I}}(\mathcal{V},\mathcal{E})$ its grid graph. We denote $\boldsymbol{x}$ the vector of binary variables indexed by image pixels, i.e., x_p is associated to pixel p. We consider the capacitated grid graph $\mathcal{G}_{\boldsymbol{I}+}(\mathcal{V}^+,\mathcal{E}^+,c)$ where

$$\begin{aligned}\mathcal{V}^+ &= \mathcal{V}\cup\{s,t\}\\ \mathcal{E}^+ &= \mathcal{E}\cup\big\{\{s,v_p\},\{v_p,t\}\mid p\in\boldsymbol{I}\big\}.\end{aligned}$$

The cost function c is defined later. The members of the additional set of edges are referred to as terminal edges. Next, let sets $\mathcal{V}_{fg},\mathcal{V}_{bg}\subset\mathcal{V}$ to represent foreground and background seeds furnished by the user. A (s,t)-cut set of $\mathcal{G}_{\boldsymbol{I}+}$ partitions the graph in connected components S,T. The first connected to the source and the other to the target vertex. Vertices in $\mathcal{V}_{fg}$ ($\mathcal{V}_{bg}$) will be forced to be in the source (target) component after the removal of a (s,t)-cut set.

The data term is modeled by

$$\psi_1(x_p)=\begin{cases}-\ln H_{bg}\big(I(p)\big), & \text{if } x_p=0\\ -\ln H_{fg}\big(I(p)\big), & \text{if } x_p=1,\end{cases}$$

where H_{bg}, H_{fg} are mixed Gaussian distributions derived from foreground and background seeds. We associate the 0 value to the background label and the 1 value to the foreground label. The space coherence term is expressed as

$$\psi_2(x_p, x_q) = \begin{cases} \exp\left(-\dfrac{1}{d_E(p,q)}\dfrac{(I(p)-I(q))^2}{2\sigma^2}\right), & q \in \mathcal{N}_k(p) \\ 0, & \text{otherwise,} \end{cases}$$

where $d_E(p,q)$ is the Euclidean distance between the pixel coordinates of p, q .The value σ is interpreted as a parameter to configure the noise level of the input image; and k denotes the chosen neighborhood cardinality of the graph (e.g. 8).

Finally, given weights $\gamma_r \geq 0$ and $\gamma_b \geq 0$, we define the cost function $c : \mathcal{E}^+ \to \mathbb{R}$ as

edge e	**c(e)**	**for**
$\{v_p, v_q\}$	$\gamma_b \cdot \psi_2(x_p, x_q)$	$\forall p \in \boldsymbol{I}$ and $q \in \mathcal{N}_k(p)$
$\{v_p, s\}$	$\gamma_r \cdot \psi_1(x_p = 0)$	$\forall p \in \boldsymbol{I}$ and $p \notin \mathcal{V}_{fg} \cup \mathcal{V}_{bg}$
	M	$p \in \mathcal{V}_{fg}$
	0	$p \in \mathcal{V}_{bg}$
$\{v_p, t\}$	$\gamma_r \cdot \psi_1(x_p = 1)$	$\forall p \in \boldsymbol{I}$ and $p \notin \mathcal{V}_{fg} \cup \mathcal{V}_{bg}$
	0	$p \in \mathcal{V}_{fg}$
	M	$p \in \mathcal{V}_{bg}$.

$$\text{where,} \qquad M = 1 + \max_{p \in \boldsymbol{I}} \gamma_b \sum_{q \in \mathcal{N}_k(p)} \psi_2(x_p, x_q).$$

Notice that each vertex in $\mathcal{V}$ must have one and only one of its terminal edges in a cut set. Let $\mathcal{E}'$ be a cut set partitioning G_{I+} in connected components (S, T), the first connected to the source and the other connected to target. Its cut value is written as

$$E^{gcut}(\mathcal{G}_{\boldsymbol{I}+}, \mathcal{E}') = \gamma_r \left(\sum_{v_p \in S} \psi_1(1) + \sum_{v_p \in T} \psi_1(0) \right) + \gamma_b \sum_{(v_p, v_q) \in \mathcal{E}'} \psi_2(0,1). \tag{2.17}$$

Figure 2.5: **Graph cut segmentation**. Foreground seeds are colored in green and background seeds are colored in blue.

We observe that Equation (2.17) can also be written in the form of the Gibbs energy

$$\gamma_r \sum_{x_p \in \boldsymbol{X}} \psi_1(x_p) + \frac{\gamma_b}{2} \sum_{\substack{x_p, x_q \in \boldsymbol{X} \\ x_p \neq x_q}} \psi_2(x_p, x_q). \tag{2.18}$$

Therefore, a minimum (s,t)-cut of $\mathcal{G}_{\boldsymbol{I}+}$ induces a labeling of minimum value for Equation (2.18). Vertices connected to the source component of the cut are labeled as foreground and those connected to the target are labeled as background. This interpretation is quite natural for the binary segmentation problem, as a minimum (s,t)-cut of the grid graph gives a binary partition of the image. In this sense, one could abstract the HMM machinery underneath and simply construct a graph such that its minimum cut separates the desired objects.

Given the particular topology of grid graphs, specific versions of max-flow algorithms were conceived for them. A scalable version of graph cut algorithm [**delong08scalable**] can be used for images of high resolution; if several flows should be computed for similar graphs (video sequence segmentation), one can use the flow recycling algorithms [**kohli05efficiently**; **juan06active**]. Although the general complexity of these alternatives may be higher than Ford-Fulkerson algorithm, they are reported [**szeliski06comparative**] to compute minimum cuts in lower time for grid graphs.

2.3.2 Geodesics computation

In the previous section, we have seen the natural connection between graph cuts and binary segmentation. The removal of a cut $\mathcal{E}'$ set partitions the grid graph in two disjoint connected components, one connected to the source and associated to the foreground and the other connected to the target and associated to the background. The cut $\mathcal{E}'$ itself can be related with the contour ∂S of the foreground

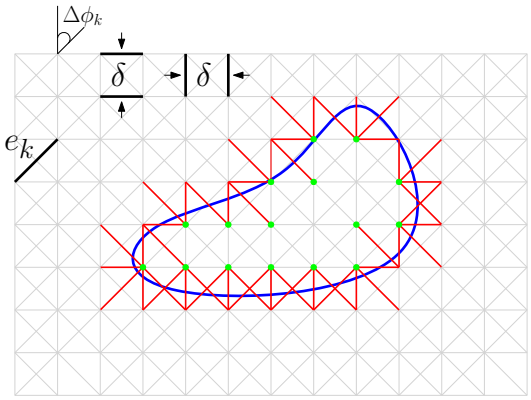


Figure 2.6: **Computing perimeter via graph cuts**. The perimeter of a shape S can be encoded as the value of some cut in the grid graph. In the figure, the contour of shape S intersects a set $C \in \mathcal{E}$ of the grid graph with neighborhood system $\mathcal{N}_8$. One can set a cost function on $\mathcal{E}$ such that the cost of C converges to the length of ∂S as the neighborhood system goes to infinity.

shape (see Figure 2.6). In [**boykov03geodesics**] it was shown that one can define a cost function for the edge set $\mathcal{E}$ such that the cost of $\mathcal{E}'$ is arbitrarily close to the length of ∂S.

The key idea is to use the Cauchy-Crofton formula from integral geometry. In $2D$, let $\mathcal{L}$ be the set of all straight lines in the plane and $d\mathcal{L}$ a Lebesgue measure on this set. Then, the perimeter of a shape S is given by

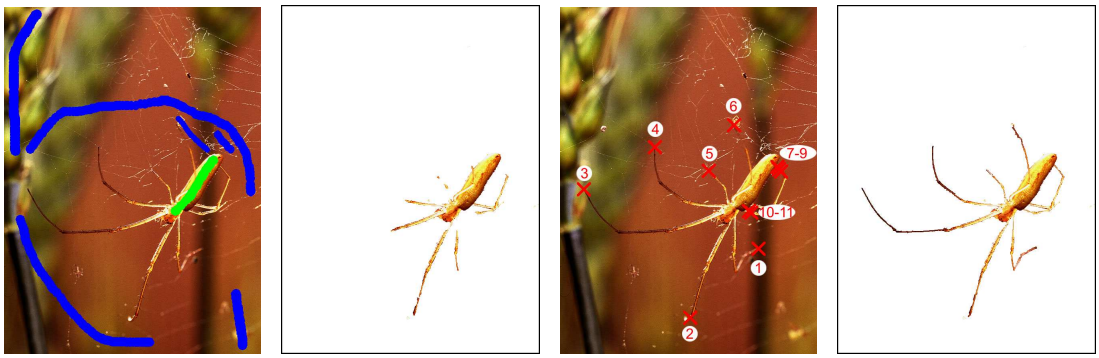
$$\int n_c d\mathcal{L} = 2|\partial S|,$$

where n_c is the number of intersections of some line in $\mathcal{L}$ with the shape contour. To compute the Euclidean length, the cost function should be set as

$$w_k = \frac{\delta^2 \Delta\phi_k}{2|e_k|},$$

where δ is the distance between two lines in the same family and $\Delta\phi_k$ is the angle difference between two consecutive lines (counterclockwise orientation) of different directions in the neighborhood system (see Figure 2.6).

The authors proved equivalent results for an arbitrary Riemannian metric in two and three dimensions. A drawback of this approach is that the quality of the results are very sensitive to the neighborhood system. Small neighborhoods are prone to metrication errors and the convergence theorem, although of theoretical importance, it does not possess the *multigrid convergence* property. The 4-neighborhood system returns a poor estimation of length no matter the image resolution. We discuss multigrid convergence in Chapter 4.



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

Figure 2.7: **Graph cut with connectivity priors [vicente08graph]**. In (a) the foreground (green) and background (blue) seeds. In (b) the graph cut segmentation. In (c) the points selected by the user in which the connectivity constraint is going to be applied. In (d) the final segmentation.

Another interesting contribution in the attempt to inject geometric information in the graph cut framework is the connectivity priors of [**vicente08graph**]. This work provides an additional tool to the graph cut algorithm in which the user select points of the image that should be connected to the foreground component. An heuristic based on the Dijkstra algorithm for shortest paths is computed using a metric based on the color intensities. The method proved very useful to the segmentation of thin and elongated objects (see Figure 2.7).

Chapter 3

Curvature as a regularizer

The curvature is a geometric property of curves and it measures the rate of change in curve orientation. A straight line has zero curvature while the curvature at a corner of a square is infinity. The curvature is measured with respect to the arc-length parametrization, and it is invariant to translations, rotations and reflections (rigid transformations).

In imaging, the most widely use of curvature is, perhaps, as a smooth regularization term. That was the case in the Geometric Active Contour and Chan-Vese image segmentation models of Chapter 1. The curvature action in these models is closed related with the *curve-shortening flow* applied to the image level curves. Additionally, the curvature also appears in the derivation of the TV denoising model.

Beyond the smoothing role, curvature can be used to favor connectivity, suggesting that its use could be valuable in the segmentation of thin and elongated objects, a difficult class to standard segmentation models. In image inpainting, the *elastica curve* revealed to be a suitable model to mimic the amodal completion phenomenon, believed to be the process behind the human vision in the perception of occluded objects (e.g., see Figure 3.4a). However, elastica minimization is a challenging task, mainly because of its non-convexity and the $4th$ order expression emerging from its Euler-Lagrange equation.

In Sections 3.1 and 3.2, we recall the definition of curvature and point out its role in some earlier imaging models. In Section 3.3, we describe the elastica curve, and the challenges involved in the minimization of the elastica energy. Section 3.4 describes discrete models aiming at the minimization of the elastica energy, which is an important topic of this thesis.

3.1 Curvature and the curve-shortening flow

The curvature is a fundamental geometric property of curves and is widely present in geometric imaging models. In this section, we start by recalling the curvature definition and a few of its mathematical representations. Next we describe the *curve-shortening flow* for planar curves, the one dimensional case of the mean curvature flow.

3.1.1 Definitions

In the remainder of this section, we assume that every curve is regular, planar and counter-clockwise oriented. Let $C : [0, L(C)] \to \mathbb{R}^2$ a curve parameterized by arc-length, i.e.,

$$C(s) = (x(s), y(s)).$$

Let $T(s)$ and $N(s)$ two orthonormal vectors such that they are respectively, the unitary tangent and normal to $C(s)$. Writing $C(s)$ using these orthonormal vectors we obtain

$$C(s) = (C(s) \cdot T(s))\, T(s) + (C(s) \cdot N(s))\, N(s)$$

Notice that $\frac{\partial C}{\partial s} = T(s)$, therefore

$$\frac{\partial C}{\partial s} \cdot \frac{\partial C}{\partial s} = \left\| \frac{\partial C}{\partial s} \right\|^2 = 1.$$

Derivating the last expression in both sides we obtain

$$\begin{aligned} 0 &= 2\frac{\partial^2 C}{\partial s^2} \cdot \frac{\partial C}{\partial s} \\ &= 2\frac{\partial^2 C}{\partial s^2} \cdot T(s). \end{aligned}$$

We conclude that $\frac{\partial^2 C}{\partial s^2}$ is orthogonal to $T(s)$, and that the curvature is given by the normal component of $\frac{\partial^2 C}{\partial s^2}$, i.e.,

$$\frac{\partial^2 C}{\partial s^2} = \kappa(s) N(s).$$

We have several formulations for curvature, some of them listed below (for a full derivation see the appendix).

$$\begin{aligned}
\textbf{Arc-length parameterization:} \quad \kappa(s) &= \left\| \frac{\partial^2 C}{\partial s^2} \right\|. \\
\textbf{Arbitrary parameterization:} \quad \kappa(u) &= \frac{y'x'' - x'y''}{(x'^2 + y'^2)^{3/2}}. \\
\textbf{Implicit function:} \quad \kappa(x,y) &= -\frac{f_{xx}^2 - 2f_x f_y f_{xy} + f_{yy}^2}{\|\nabla f\|^3} \\
&= \nabla \cdot \left(\frac{\nabla f}{\|\nabla f\|} \right).
\end{aligned}$$

3.1.2 Curve-shortening flow

Let $t \geq 0$ and $u \in [0, U]$, for some $U > 0$. Next, let $C^{(0)} : [0, U] \to \mathbb{R}^2$ and $\mathcal{C}(t, u)$ a family of curves such that

$$\mathcal{C}(0, u) = C^{(0)}(u) \tag{3.1}$$

$$\frac{\partial \mathcal{C}}{\partial t}(t, u) = v(t, u) N(t, u), \tag{3.2}$$

where v is an arbitrary smooth function. The curve $C^{(t)}$ is deformed at each point p at speed v in the normal direction at p. Note that any tangent component is irrelevant to curve deformation. From Equation (3.2) we compute the first variation of curve length of family $\mathcal{C}$

$$\begin{aligned}
\frac{\partial}{\partial t} L(t) &= \frac{\partial}{\partial t} \int_0^U \|\mathcal{C}_u\| du \\
&= \int_0^U \frac{\mathcal{C}_u \cdot \mathcal{C}_{ut}}{\|\mathcal{C}_u\|} du \\
&= \int_0^U T \cdot (\mathcal{C}_t)_u \, du.
\end{aligned}$$

Changing from an arbitrary parameterization u to an arc-length one and recalling that $ds = \|\mathcal{C}_u\| du$, hence $\frac{\partial C_t}{\partial u} = \|C_u\| \frac{\partial C_t}{\partial s}$, we obtain

$$\frac{\partial}{\partial t} L(t) = \int_0^U T \cdot (\mathcal{C}_t)_s \, \|\mathcal{C}_u\| du = \int_0^{L(\mathcal{C})} T \cdot (\mathcal{C}_t)_s \, ds.$$

Using Equation (3.2) and $N_s = -\kappa T$, we obtain

$$\frac{\partial}{\partial t}L(t) = \int_0^{L(\mathcal{C})} T \cdot (-\kappa v T + v_s N) ds = -\int_0^{L(\mathcal{C})} \kappa v ds = - < \kappa, v > .$$

From which we conclude that length decreases the fastest by choosing $v = \kappa$. We define the *curve-shortening flow* (CS flow) of curve $C^{(0)}(u)$ as the family $\mathcal{C}(t,u)$ such that

$$\mathcal{C}(0,u) = C^{(0)}(u) \tag{3.3}$$

$$\frac{\partial \mathcal{C}}{\partial t}(t,u) = \kappa(t,u)N(t,u). \tag{3.4}$$

Example: Let $C^{(0)}$ be a circle of radius R_0, i.e., $C^{(0)} = R_0(\cos t, \sin t)$. From Equation (3.4) we conclude that for every $t \geq 0$, $C^{(t)}$ is a circle of radius $R(t)$ where

$$\frac{\partial R}{\partial t} = -\frac{1}{R} \rightarrow R(t) = \sqrt{R_0^2 - 2t}. \tag{3.5}$$

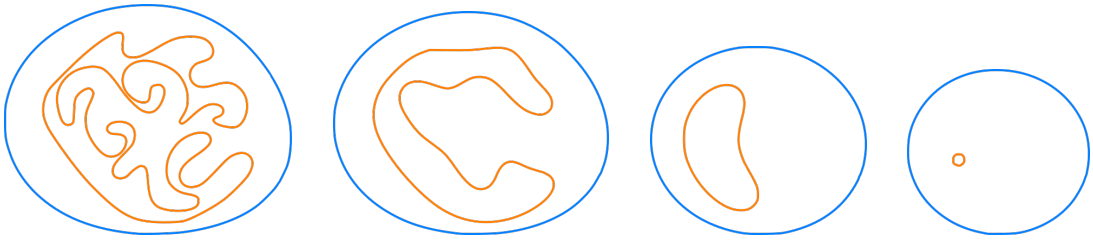
Moreover, the curve collapses to a single point in time $t = \frac{R_0^2}{2}$. The curve-shortening flow has many interesting properties [**huisken84flow**; **gage86heat**; **ecker08heat**]. Among them (for planar curves),

> **Comparison principle:** Let C_1,C_2 two closed curves such that $C_1^{(0)}$ is in the interior of $C_2^{(0)}$. Then $C_1^{(t)}$ is in the interior of $C_2^{(t)}$ for every t.
>
> **Convexity preserving:** A convex curve $C^{(0)}$ stays convex for all t.
>
> **Point collapsing:** Let $C^{(0)}$ a closed curve. There exists a time t in which $C^{(t)}$ describes a circle and it follows Equation (3.5) until collapsing into a single point.
>
> **Perimeter minimization:** The curvature flow is the continuous deformation that decreases the perimeter of a single closed curve at the fastest speed.

The CS flow appeared in the Chan-Vese and Geometric Active Contour models for image segmentation in Chapter 1 in its level set formulation [**osher88fronts**]

$$\frac{\partial f}{\partial t} = \|\nabla f\| \nabla \cdot \left(\frac{\nabla f}{\|\nabla f\|} \right). \tag{3.6}$$

In this case, each of the image level curves describes a CS flow.



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

Figure 3.1: **CS flow in action.** CS flow at different times in evolution. In this example we observe the CS flow properties listed in the text.

3.2 Diffusion and level curves motion

Several models in Chapter 1 are stated as diffusion processes, but quite often a level curve evolution interpretation is more convenient. In this section we analyze the role of curvature, its properties in earlier models of image processing and how the dual interpretation of these models helps us to gain extra insight on them.

3.2.1 Curvature in denoising and image segmentation

The necessary optimality condition of the total variation denoising energy is given by (see Section 1.2.1)

$$\nabla \cdot \left(\frac{\nabla f}{\|\nabla f\|} \right) + \lambda (f_{\tilde{I}} - f) = 0. \tag{3.7}$$

The solution of Equation (3.7) is the steady state of the anisotropic diffusion

$$u(0, x) = f(x) \tag{3.8}$$

$$\frac{\partial u}{\partial t} = \nabla \cdot \left(\frac{\nabla f}{\|\nabla f\|} \right) + \lambda (f_{\tilde{I}} - f). \tag{3.9}$$

Note the similarity between Equation (3.9) and the level set formulation of the CS flow in Equation (3.6). We are interested in the evolution of Equations (3.8) and (3.9) without considering the data fidelity term, i.e., $\lambda = 0$, and we are going to call it the *TV flow*[**bellettini02total**]. Interestingly, both CS flow and TV flow decrease total variation of f, but in different ways. The CS flow tends to deform the boundary of objects and decreasing total variation by perimeter minimization. The TV flow, on the other hand, preserves the boundary for a longer time and it decreases the TV by decreasing the surface height, i.e., the color intensity of the pixels(see Figure 3.2).

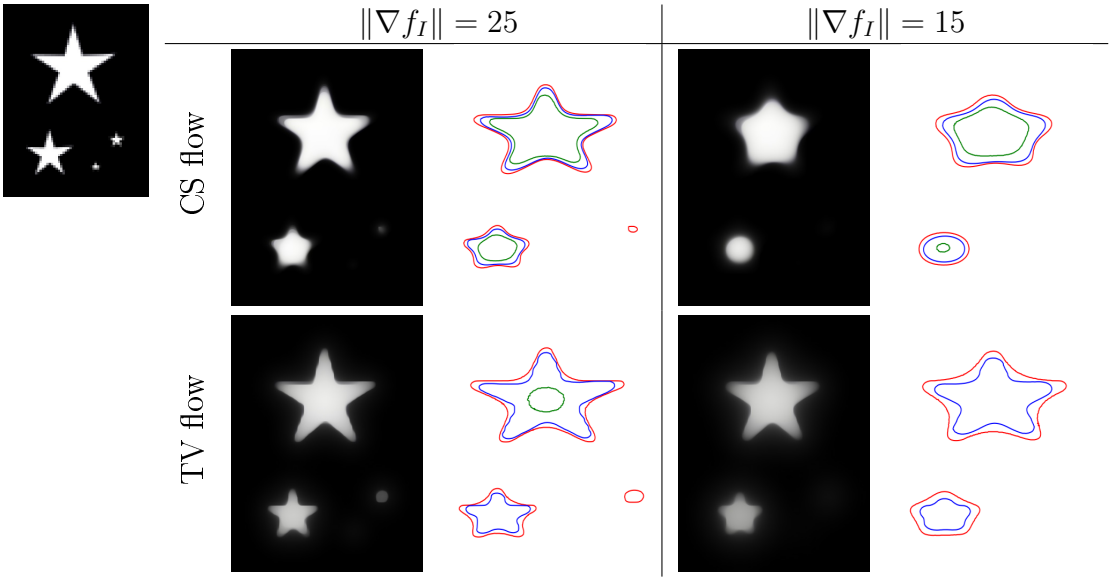


Figure 3.2: **Minimizing TV energy with TV flow and CS flow.**Both flows decrease the image total variation, but CS flow tends to deform object boundaries while doing so. On the other hand, TV flow reduces the image contrast.

Naturally, the CS flow can also be used to do image denoising. If we think of noise as small artifacts with high gradient of color, the level curves corresponding to noise will have a very small radius, and will collapse faster than other level curves (see Figure 3.3).

In the geometric active contours model for image segmentation, the level sets of a predefined function u (e.g., a smoothed version of function $1-\chi_C$ where C is a set that contains the object to be segmented and χ_C its characteristic funtion) describes a CS flow motion modulated by an edge detector function g (e.g., $g = (1+\|\nabla f\|)^{-1}$).

$$\begin{aligned} u(0,x,y) &= u(x,y) \\ \frac{du}{dt} &= g(\|\nabla f\|)\|\nabla u\|\nabla \cdot \left(\frac{\nabla u}{\|\nabla u\|} + v\right), \end{aligned}$$

3.2.2 Curvature and the connectivity principle applied to inpainting

We recall that the inpainting problem consists in reconstructing a collection of missing patches $\mathcal{P}$ of the image. The earlier inpainting model [**bertalmio00image**] diffuses image information through the patches boundaries. The particularity of this

(a) Noisy Image (b) CS flow (c) TV flow

Figure 3.3: **TV denoising using CS and TV flow.**Results are quite similar, but TV flow image is sharper, although with lower contrast than CS flow. Images (b) and (c) have the same TV value.

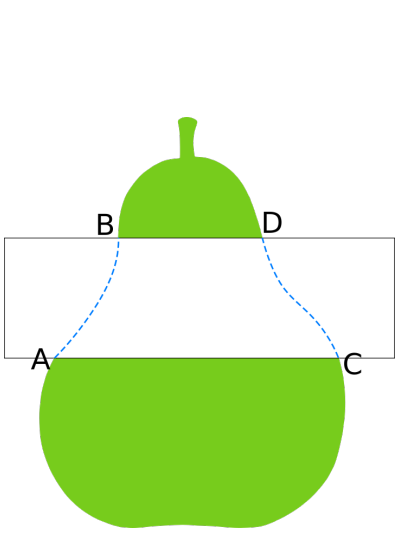
model is that the Laplacian (heat equation) is diffused along the image level lines. Despite its impressive results, this models does not possess an important property which is expected in inpainting models: the *connectivity principle*.

Studies of the Gestalt school of psychology suggest that the human vision, by a process called amodal completion [**mumford94elastica**], completes the boundary of partially occluded object with a short and low curvature curve, as illustrated in Figure 3.4a. The model proposed in [**chan01nontexture**] is a TV flow modulated by the curvature of the image level curves.

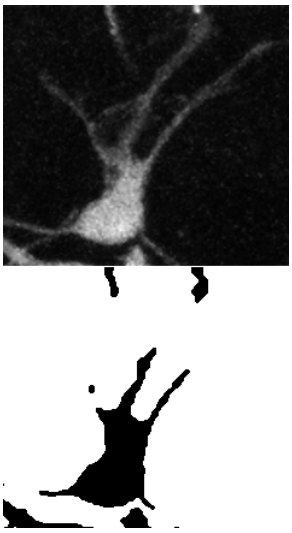
$$\begin{aligned} u(0,x,y) &= f(x,y) \\ \frac{\partial u}{\partial t} &= \nabla \cdot \left(|\kappa(t)| \frac{\nabla f}{\|f\|} \right). \end{aligned} \tag{3.10}$$

The diffusion in Equation (3.10) is stronger at points of high curvature with respect to its level curves. This model partially achieves the connectivity principle (see Figure 3.5), but the completion is done only for very small regions.

The connectivity principle suggests that the curve connecting the endpoints of an occluded object should minimize some energy with respect to the curvature. Moreover, the segmentation of thin and elongated objects may also benefit from a curvature term, since classical methods have difficulties to give connected solutions. In the next section we describe the Elastica, the curve that minimizes the squared curvature along its length.



(a) Amodal completion.



(b) Thin and elongated object segmentation.

Figure 3.4: **Connectivity principle.** In Figure 3.4a, an illustration of amodal completion, the process followed by the human vision to complete the boundary of occluded object with curves of low curvature and that conform the most with the direction of the endpoints. In Figure 3.4b a vessel segmentation with length penalization term only. Curvature favors connected objects.

Figure 3.5: **Inpainting by curvature driven diffusions.**[**chan01nontexture**] The diffusion process modeled by Equation (3.10) completes the features disconnected by the inpainting mask.

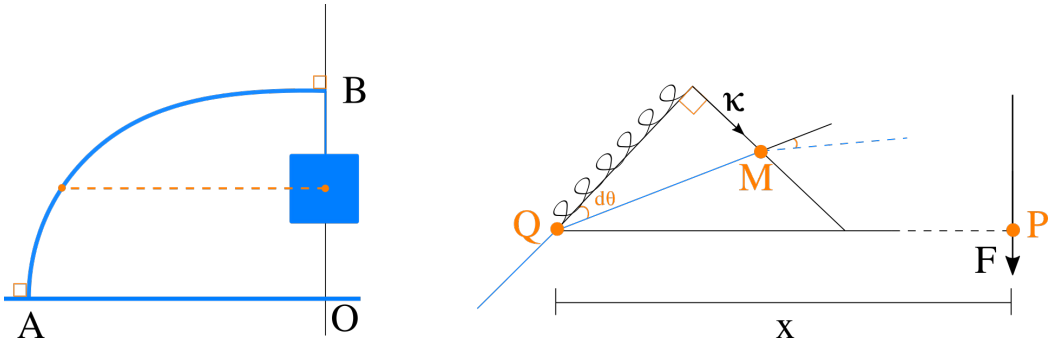


Figure 3.6: **Elastic beam and James Bernoulli's solution.** In equilibrium, the moments at the two ends of the virtual lever PQM should cancel, implying that the curvature is a linear function of x if we assume the spring behaves in accordance with Hooke's law.

3.3 Elastica curve

In 1691 James Bernoulli pose a challenge to his colleagues: What is the form taken by an elastic beam whose endpoint A is kept fixed in the ground while a weight is attached to its other endpoint B such that the tangent at each of the endpoints are perpendicular? A simplified scheme is presented in Figure 3.6.

By using mechanical principles, James concluded that the curvature of such curve at some point Q can be written as a linear function of the distance from Q to the line BO, i.e., $\kappa(x) = cx$. One can show that [**levien08elastica**; **truesdell60rational**],

$$\frac{\partial y}{\partial x} = \frac{cx^2}{\sqrt{1 - c^2x^4}}. \tag{3.11}$$

The curve modeled by Equation (3.11) is called the *rectangular elastica*. The expression in the right cannot be integrated using elementary functions and is a member of the class of elliptic integrals. Fortunately, Euler has shown that one can express not only the rectangular elastica, but its general version in the form of a minimization principle.

The generalized elastica in 2D is the planar curve C of fixed length L whose endpoints have known tangents $C'(0) = \theta_0, C'(L) = \theta_L$ and minimizes the energy $\int_C \kappa^2 ds$.

$$\begin{aligned} &\min_C \int_C \kappa^2 \\ &\text{subject to} \quad \begin{aligned} &\|C\| = L, \\ &C'(0) = \theta_0, \\ &C'(L) = \theta_L. \end{aligned} \end{aligned}$$

Given parameters $\alpha \geq 0, \beta \geq 0$ we incorporate the isoperimetric constraint in the objective function (can be interpreted as a Lagrange multiplier) to define the Elastica energy as

$$E(C) = \int_C \alpha + \beta\kappa^2 ds. \tag{3.12}$$

If we do not impose any constraint in Equation (3.12), its minimization is interesting only if $\alpha > 0$ and $\beta > 0$ and we call this problem the *free Elastica*. On the other hand, if one imposes constraints to the minimization of Equation (3.12)(e.g., fixed tangent endpoints), we refer to this problem as the *constrained Elastica*.

One can easily derive the solution of the free Elastica for a closed curve. Setting $\beta = 0$, the minimization of Equation (3.12) behaves as the curve-shortening flow of section Section 3.1.2. On the other hand, if we set $\alpha = 0$, the minimum is a disk of infinite radius. For the intermediate case, with both parameters greater than zero, it is easy to see that the solution is a circle C_r of some radius r, therefore

$$\begin{aligned} C_r = \arg\min_C E(C) \rightarrow \frac{\partial}{\partial r}\int_{C_r} \alpha + \beta\kappa^2 ds = 0 \\ \frac{\partial}{\partial r}\left(\alpha 2\pi r + 2\beta\pi/r\right) = 0 \\ r = \left(\frac{\beta}{\alpha}\right)^{1/2}. \end{aligned}$$

In general, a planar curve C minimizing the Elastica energy satisfies the following Euler-Lagrange equation [**chan02elasticainpainting**; **singer08lectures**]

$$2\kappa_{ss} + \kappa^3 = \frac{\alpha}{\beta}\kappa \Leftrightarrow \frac{\partial^4 C}{\partial s^4} + \left(\frac{\partial^2 C}{\partial s^2}\right)^3 = \frac{\alpha}{\beta}\frac{\partial^2 C}{\partial s^2}. \tag{3.13}$$

This fourth order expression is the main difficult in minimizing the Elastica. In the following we are going to describe some models that attempt to minimize the Elastica and the strategies employed.

3.3.1 Imaging models using the Elastica

In [**chan02elasticainpainting**] the authors propose an inpainting model in which the image level lines interrupted by the inpainting domain are completed by Elastica curves. The key is to define an energy that minimize each of the level curves in terms of the Elastica simultaneously. Given an image $f_{\boldsymbol{I}} : \Omega \rightarrow [0, 1]$, let Γ_λ be one of its level curve, i.e.,

$$\Gamma_\lambda = \{x \mid f_{\boldsymbol{I}}(x) = \lambda\}.$$

Therefore, we minimize each level curve in terms of the Elastica energy by minimizing the functional

$$\int_0^1 \int_{\Gamma_\lambda} \left(\alpha + \beta\kappa^2\right) ds d\lambda = \int_0^1 \int_{\Gamma_\lambda} \left(\alpha + \beta \nabla \cdot \left(\frac{\nabla f_I}{\|\nabla f_I\|}\right)^2\right) ds d\lambda \tag{3.14}$$

Next, let dt be the length element in the normal direction of the level curves. Therefore,

$$\frac{d\lambda}{dt} = \|\nabla f_{\boldsymbol{I}}\| \rightarrow d\lambda = \|\nabla f_{\boldsymbol{I}}\| dt.$$

Replacing the last expression in Equation (3.14) we obtain

$$\int_0^1 \int_{\Gamma_\lambda} \left(\alpha + \beta \nabla \cdot \left(\frac{\nabla f_{\boldsymbol{I}}}{\|\nabla f_{\boldsymbol{I}}\|}\right)^2\right) \|\nabla f_I\| ds dt = \int_\Omega \left(\alpha + \beta \nabla \cdot \left(\frac{\nabla f_{\boldsymbol{I}}}{\|\nabla f_{\boldsymbol{I}}\|}\right)^2\right) \|\nabla f_{\boldsymbol{I}}\| d\Omega. \tag{3.15}$$

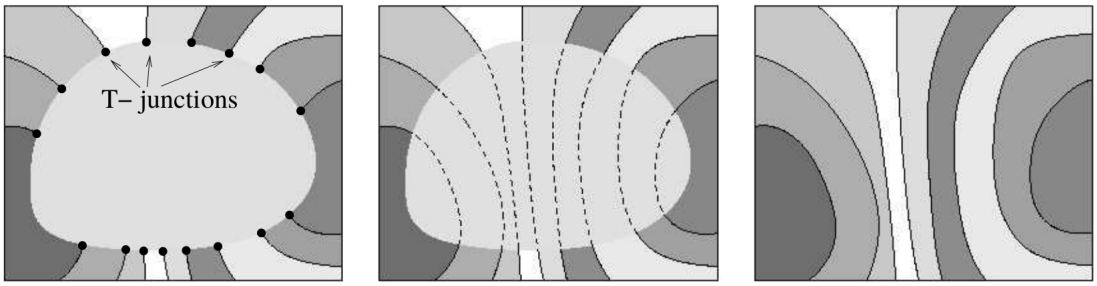
The last expression is derived from the fact that dt and ds are orthogonal length elements. The authors propose a finite difference scheme for the gradient flow derived from the Euler-Lagrange of Equation (3.15), which happens to be of $4th$ order. The high order poses some issues with numerical instability, namely the definition of the time step and convergence analysis, which is skipped. The running time is also quite high even for small inpainting domains. Moreover, one can expect a local solution at most, and giving the highly non-convex character of the equation, an acceptable local solution might be very difficult to be found without a conveniently chosen initial solution. Finally, the method tends to produce blurry edges even for small inpainting domains.

The numerical instability arising from Equation (3.15) is lessened by the $3rd$ order gradient flow proposed in [**ballester01filling**]. The idea is to create a new variable θ to replace the unstable term $\nabla f_{\boldsymbol{I}}/\|\nabla f_{\boldsymbol{I}}\|$ and include a penalization term that forces θ to assume the value of its original interpretation. The energy to be minimized is closely related to Equation (3.15) and is written as

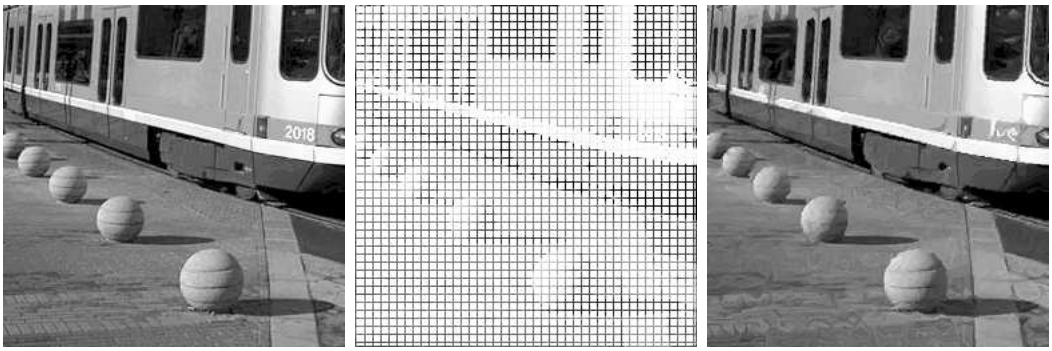
$$\int_\Omega a|\nabla \cdot \theta|^2 \left(\alpha + \beta \|k * f_{\boldsymbol{I}}\|\right) d\Omega + \int_\Omega \|\nabla f_{\boldsymbol{I}}\| - \theta \cdot \nabla f_{\boldsymbol{I}} d_\Omega, \quad \|\theta\| \leq 1. \tag{3.16}$$

The kernel k (e.g. Gaussian) smooths image $f_{\boldsymbol{I}}$ and is included due to numerical reasons. Nonetheless, Equation (3.16) suffers from similar issues to those of Equation (3.15). Generally, PDE-based methods encounter difficulties in producing solutions with discontinuities, one of the main features of images.

A discrete approach to inpainting is proposed in [**masnou98inpainting**]. The model's first step is to identify pairs of admissible T-junctions, i.e., pixels in the outer



(a) T-junctions illustration [**ambrosio03direct**]



(b) Inpainting by [**masnou98inpainting**]

Figure 3.7: **Discrete inpainting.** In Figure 3.7a an illustration of T-junctions and an ideal completion of the level lines; in Figure 3.7b we have the original image in the left, the image to be inpainted in the middle and the result of the discrete inpainting in the right.

boundary of the inpainting domain that have the same color value and orientation (see Figure 3.7a). Due to a property of level curves, for every value color value λ there is an even number of T-junctions. Next, let J be the set of T-junctions and Γ_j^t a curve connecting an admissible pair (j,t). Additionaly, we denote θ_j, θ_t the respective angles the curve Γ_j^t makes with the associated level set at j and t. The authors propose to find a matching $\mathcal{M} = \{(j,t) \mid j,t \text{ admissible}\}$ and its respective curves such that it minimizes the energy

$$\sum_{(j,t)\in\mathcal{M}} \int_{\Gamma_j^t} \alpha + \beta|\kappa| ds + \theta_j + \theta_t. \tag{3.17}$$

Assuming that one knows the optimal matching $\mathcal{M}$, the curves connecting the T-junctions pairs are geodesics. In the plane, that means that the T-junctions are connected by polygonal curves. An ingenious dynamic programming algorithm is conceived by exploiting the causality relation imposed by the non-crossing con-

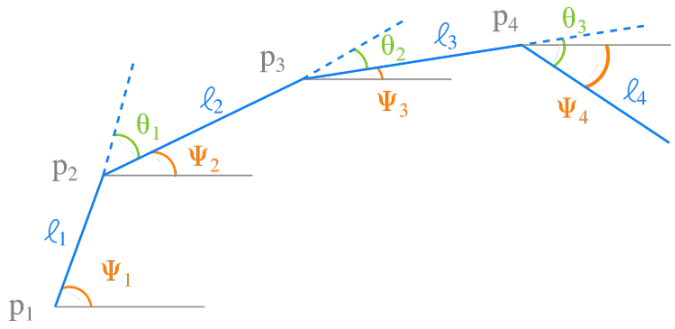


Figure 3.8: **Discrete elastica.** The curvature is approximated by θ_i/l_i.

straint of the level curves and excellent results are obtained. In particular, the algorithm can reconstruct large inpainting domains and does not produce blurry edges (see Figure 3.7b). However, Equation (3.17) cannot produce curvy level lines.

The work of [**masnou98inpainting**] illustrates an advantage of discrete approaches to image models: discontinuities are naturally implemented. To find a discrete model is not a trivial task, and sometimes a compromise needs to be done in order to achieve efficiency, as it was the case in [**masnou98inpainting**], in which the absolute value of the curvature is used instead of its square. In the next section we are going to describe some discrete models for Elastica in its most known form, i.e., using the square of the curvature.

3.4 Discrete methods and squared curvature

3.4.1 Discrete elastica

Let $\mathcal{P}_n = \{p_i \mid 1 \leq i \leq n\}$ a sequence of n points describing a polygonal line. We define the following elements (an illustration is given in Figure 3.8).

$$\begin{aligned} \text{Segment length:} \quad & \ell_i, \quad && 1 \leq i \leq n \\ \text{Segment angle:} \quad & \psi_i, \quad && 1 \leq i \leq n-1 \\ \text{Angle deviation:} \quad & \theta_i, \quad && 1 \leq i \leq n-2. \end{aligned}$$

Given parameters $\alpha \geq 0, \beta \geq 0$ and a fixed length segment ℓ, the discrete elastica [**bruckstein01discrete**] is defined as

$$E(\boldsymbol{\theta}, \ell) = \ell \cdot \sum_{1 \leq i \leq n-2} \alpha + \beta \left(\frac{\theta_i}{\ell}\right)^2 = \sum_{1 \leq i \leq n-2} \alpha\ell + \beta \frac{\theta_i^2}{\ell}. \tag{3.18}$$

Without loss of generality, we assume that the curve starts at the origin with tangent $\psi_1 = t_1$ and ends at point $(L, 0)$ with tangent $\psi_{n-1} = t_{n-1}$, L being the curve length. Then, the general discrete elastica problem is defined as

$$\min_{\boldsymbol{\psi},\ell} \sum_{1\leq i\leq n-2} \alpha\ell + \beta\frac{(\psi_{i+1}-\psi_i)^2}{\ell}$$

subject to

$$\begin{aligned} \sum_{1\leq i\leq n-1} \ell\cos\psi_i &= L, \\ \sum_{1\leq i\leq n-1} \ell\sin\psi_i &= 0, \\ \psi_1 &= t_1, \\ \psi_{n-1} &= t_{n-1}. \end{aligned}$$

The constraints can be included in the objective function with its respective Lagrange multipliers coefficients and we end up with a nonlinear system. In [**bruckstein01discrete**], the authors reported that standard methods as Newton-Raphson return nice approximations of the continuous elastica. In a companion paper, the authors proved that a similar formulation of the discrete elastica, namely

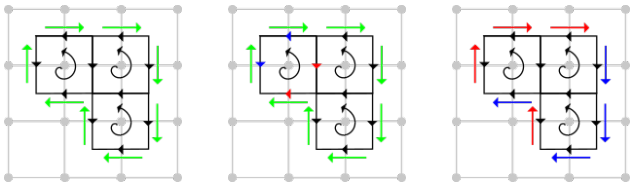
$$E(\boldsymbol{\theta}, l) = \sum_{1\leq i\leq n-2} \alpha\ell + \beta\left(\frac{\theta_i^2}{\min\{\ell_i, \ell_{i+1}\}}\right) \tag{3.19}$$

is convergent in the sense of Γ-convergence [**bruckstein01epi**], i.e., the minimizer of the discrete elastica approaches the minimizer of the elastica as the number of discretization points goes to infinity.

The discrete elastica was proposed in the context of nonlinear splines with applications in shape design. In imaging, we have to consider additional constraints. For example, digital images are defined in an uniform grid, while the discrete elastica imposes no restriction on the location of the interpolation points in the plane. In fact, this is a fundamental difference between discrete and digital settings. In the list that follows, we point out some of the issues one has to handle in order to adapt the discrete elastica in image segmentation

1. **Data term:** The interpolation points should lie close to the contour of objects and by doing that we lose the implicit description of the curve as a sequence of lengths and angle deviations, being forced to also consider the points coordinates;

2. **Contour complexity:** We are handling closed contours, meaning that we likely have to minimize not one but many elasticas along a single contour (not



(a) (b) (c)

Figure 3.9: **Digital surface and consistency constraints.** In Figure 3.9a, a valid solution made of three cells and a sequence of eight linels. In Figure 3.9b the sum of linel-cell incidence equals to zero (blue are positively incident and red negatively incident). In Figure 3.9c, the sum of linel-edge incidence equals to zero.

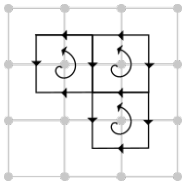
to mention multisegmentation). In practice, that means that we have to chose feature points in the image to force the curve to pass through them;

3. **Point sampling:** Finally, one has to decide how many points to use in the interpolation and where to position them (a uniform sampling is unlikely a good strategy), bearing in mind the compromise between number of points and computation time.

3.4.2 Linear programming model for image segmentation using the discrete elastica

In [**schoenemann09linear**], the authors handle the previous list of issues by considering only the discrete elasticas lying on a 2D cellular complex. The 2D cellular complex is composed by non-overlapping surfaces called *cells* and its lower dimension components: *linels* and *pointels*. The cellular complex forms a tessellation of the plane and the shape of the cells is arbitrary. The chosen cellular complex has a direct influence in the quality of the discrete elastica, as a finer tessellation covers a larger range of angles (see Figures 3.10a and 3.10b). To simplify exposition, we are going to describe the model for a standard cellular grid complex, in which each cell represents a single pixel in the image.

Let n, m the number of cells and edges, respectively. We establish a convention on cells and linels orientation. We define the positive orientation of a cell as being counter-clockwise and the positive orientation of linels as being left (horizontal) and down (vertical), as indicated in the figure below.



One binary variable is defined for each cell and two other binary variables are defined for each linel, one for each possible orientation the linel may assume. We refer to the variables associated to linels as edges. The variables are grouped in vector $\boldsymbol{x} \in \{0,1\}^{n+2m}$ and, hence, a solution is made of active cells and active edges. However, we have to make sure that solutions are consistent, i.e., a solution $\boldsymbol{x}$ must encode a digital surface.

A digital surface is composed of a set of cells and a closed path (loop) of edges forming its boundary. To enforce digital surfaces, we define an incidence relation between linels and cells and between linels and edges. This relation is encoded by matrix $\boldsymbol{A} \in \{-1,0,1\}^{m\times(n+2m)}$

$$\boldsymbol{A}_{i,j} = \begin{cases} 1, & \text{if positive incident} \\ -1, & \text{if negative incident} \\ 0, & \text{otherwise.} \end{cases}$$

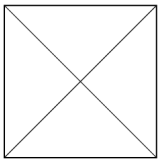
We say that a linel is positive incident to a cell if their orientations agree and negative incident if they disagree. Similarly, a linel is positive (negative) incident to an edge if they agree (disagree) in orientation. The space of solution is correctly restricted to digital surfaces by imposing

$$\boldsymbol{A}\boldsymbol{x} = 0,$$

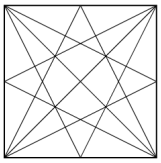
i.e., the sum of incidences for each linel must equal to zero (see Figure 3.9). Finally, we set vector $\boldsymbol{w} \in \mathbb{R}^{n+2m}$ that is going to hold the data, length and squared curvature penalization terms. For example, data term coefficients are associated to cells variables while length and curvature terms are associated to edges. The complete formulation is written as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{w}^t\boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = 0 \\ & \boldsymbol{x} \in [0,1]^{n+2m}, \end{aligned}$$

where vector $\boldsymbol{x}$ was relaxed. The quality of the model naturally depends of the plane tessellation defined by the chosen cellular complex. A simple grid complex is not that interesting, as the discrete elastica can turn only on multiples of $\pi/2$. The authors present results for two different cellular complexes. They are equivalent to 8-connectivity or 16-connectivity in a graph-cut framework.



(a)



(b)

input with seeds

8-connectivity

16-connectivity

(c)

Figure 3.10: **Linear programming model [schoenemann09linear].** In the left, examples of the 8 and 16-connectivity cellular complexes. The authors implementation consists in subdividing a pixel. For example, the 8-connectivity is implemented by rescaling the image twice its size and let the model unit being composed of 4 pixels of the rescaled image. In the right, a result with different connectivities using a data term derived from given foreground (green) and background (blue) seeds.

The data term issue is handled by separating the roles of cells and edges. Cells variables holds data terms and edge variables hold curvature and length information. The refinement of the cellular complex is done via pixel subdivision, therefore, every cell lying in the interior of a pixel p carries data term relative to pixel p. The consistency constraints guarantees well defined boundaries and the contour complexity issue is also covered. Finally, the model is extensible to different angular resolutions while keeping a digital setting, and a discrete elastica with an arbitrary sequence of interpolation points can be well approximated given a good choice of the cellular complex.

Some results are displayed in Figure 3.10c. However, the computation time is quite high and the segmentation present sharp angles even when employing a 16-connectivity cellular complex.

3.4.3 Unconstrained formulations

The high number of variables and constraints explain the high running time of the linear programming approach. In [**zehiry10fast**], the authors encode the squared curvature value in an unconstrained quadratic non-submodular PBF and propose a model for binary image segmentation.

Let I be a digital image with n pixels. We associate to each pixel the binary variable $\boldsymbol{x}_i$ indicating if the pixel belongs to foreground ($x_i = 1$) or background ($x_i = 0$). For a given segmentation $\boldsymbol{x}$, one can identify 90 degrees turns in the

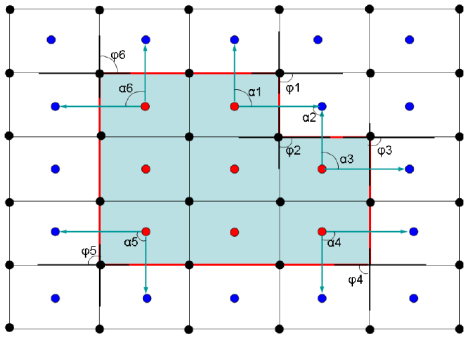


Figure 3.11: **Turn angles detection and triplets [zehiry10fast].** Assuming that red pixels correspond to 1-labeled variables and blue the 0-labeled ones, angle variation is triggered by configurations $(x_i = 1, x_j = 0, x_k = 0)$ and $(x_i = 0, x_j = 1, x_k = 1)$, assuming that x_i is the centered variable.

segmentation contour by inspecting sequential pairs of vertices neighbors (given an order on the neighbors), as illustrated in figure Figure 3.11. The turns can expressed as a 3-clique potential, as shown in the table below

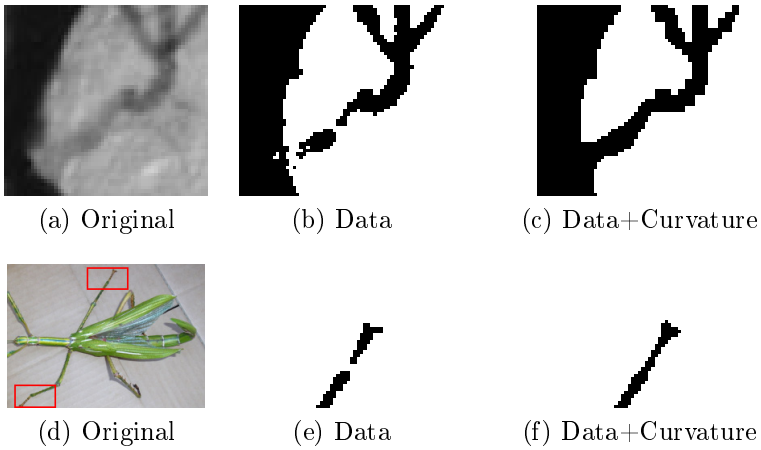
x_i	x_j	x_k	w
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	w_{ijk}
1	0	0	w_{ijk}
1	0	1	0
1	1	0	0
1	1	1	0

The coefficients w_{ijk} are defined accordingly with the curvature discretization given by Equation (3.19), i.e.,

$$w_{ijk} = \frac{\alpha_i^2}{\min\{|e_{ij}|, |e_{jk}|\}}.$$

Remarkably, the 3-clique potential can be decomposed in three 2-cliques potentials

$$E(x_i, x_j, x_k) = w_{ij}(x_i - x_j)^2 + w_{ik}(x_i - x_k)^2 + w_{jk}(x_j - x_k)^2, \tag{3.20}$$



(a) Original (b) Data (c) Data+Curvature

(d) Original (e) Data (f) Data+Curvature

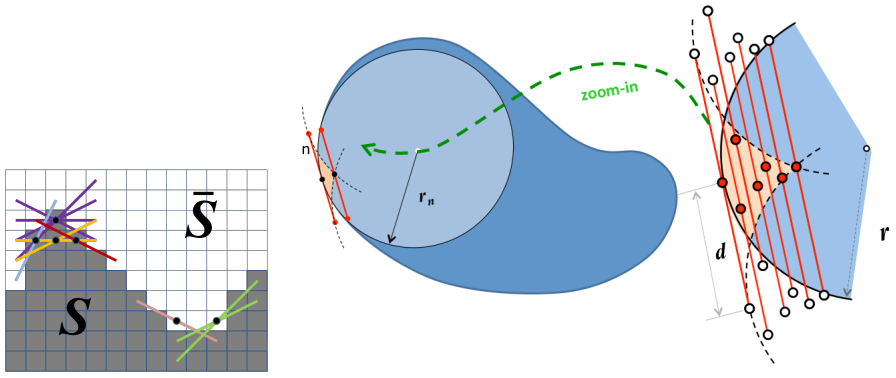
Figure 3.12: **Results for the grid graph model [zehiry10fast].** The original image is displayed in the left; segmentation using only data in the middle; and the segmentation using squared curvature regularization in the right. Connectivity is encouraged, but to a very limited extension.

where

$$w_{ij} = \frac{1}{2} w_{ijk}$$
$$w_{ik} = \frac{1}{2} w_{ijk}$$
$$w_{jk} = -\frac{1}{2} w_{ijk}.$$

Equation (3.20) is a non-submodular PBF (the negative w_{jk} coefficient creates a non-submodular term) and it is optimized using QPBO or one of its variations. The computation is much faster than the linear programming formulation and the model favors the connectivity principle, although the completion effect seems limited to very small portions (see Figure 3.12). A clear drawback is the limitation in the angle resolution, which tends to produce block artifacts. The authors argue that the model can be extended to any desired connectivity, but it is not clear how this is done.

In [**nieuwenhuis14efficient**] is presented an alternative formulation that is extensible to any desirable angle resolution. The idea is in fact quite similar to the one in [**zehiry10fast**] in the sense that curvature is measured by counting the number of configurations $(0, 1, 0)$ and $(1, 0, 1)$ of triplets of vertices. Intuitively, as illustrated in Figure 3.13a, the curvature is higher at regions in which these configurations are more frequent. The authors prove the following theorem



(a) (b)

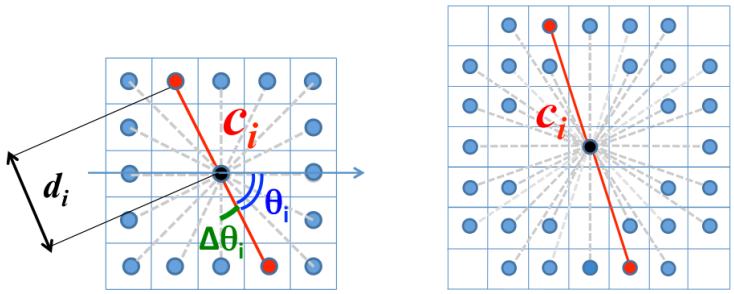
Figure 3.13: **Triplets configurations and integral geometry [nieuwenhuis14efficient].** In Figure 3.13a we intuitively notice that curvature is higher where triplets configurations $(0, 1, 0)$ and $(1, 0, 1)$ are perceived. In Figure 3.13b, an illustration of Theorem 1.

Theorem 1([nieuwenhuis14efficient]): Let contour point n have tangent orientation t and osculating ball B_r of radius $r = 1/|k|$. Then, the set of all points $p \in B_r$ such that $\|p - n\| \leq r$ and $(p \pm d \cdot t) \notin B_r$ for given distance $d < r$ has area

$$A(\kappa, d) = \frac{|\kappa| d^3}{4} + O(d^4).$$

Notice that the theorem is valid for cliques aligned with the tangent at the point of curvature computation. The set of available cliques orientations is defined according to the chosen neighborhood. The neighborhood system of size $(2d+1) \times (2d+1)$ for pixel p is similar to a digital circle of radius d centered at p (although not the same), in the sense that the neighbors of p are chosen in order to have approximately the same distance d from p (see Figure 3.14). The larger the value of d, the greater the chances to have a triplet orientation that matches the tangents along the object contour.

Let d be the chosen neighborhood system and c_i the corresponding triplets in this neighborhood. Moreover, assume that the triplets are ordered with respect the angle made with the horizontal axis. Let $\Delta\theta_i$ be the angle difference between two consecutive triplets and $i(p)$ the triplet index with the closest orientation to the



(a) $d = 2$ (b) $d = 3$

Figure 3.14: **Neighborhood system [nieuwenhuis14efficient].** The higher the value of d, the higher is the angle resolution.

tangent at p. The integral squared curvature is approximated by

$$\begin{aligned}\int_C \kappa^2 ds &\approx \sum_p |\kappa_p| \Delta\theta_{i(p)} \\ &= \sum_p \frac{4\Delta\theta_{i(p)}}{d^3_{i(p)}} \cdot A(\kappa_p, d_{i(p)}) \\ &\approx \sum_i \sum_p \frac{4\Delta\theta_i}{d_i^3} \cdot \delta(c_i(p)),\end{aligned}$$

where the function δ is one for triplets configurations $(0,1,0)$ or $(1,0,1)$ and zero otherwise. The function δ can be expressed as

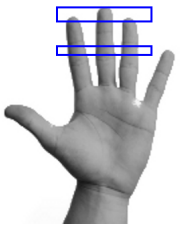
$$\begin{aligned}\delta(x_a, x_b, x_c) &= x_i(1-x_j)x_k + (1-x_i)x_j(1-x_k) \\ &= x_j + x_i x_k - x_i x_j - x_j x_k.\end{aligned}$$

The resulting energy is non-submodular (the positive term in the δ function is non-submodular) and is solved using the LSA-TR method [**gorelick14local**], reported to produce better results than QPBO for some non-submodular instances. Some results for inpainting and segmentation are shown in Figure 3.15. The model is unconstrained, extensible to arbitrary angle resolution and can be easily modified to include data and length terms.

Although an argument is provided to justify the curvature approximation, it is difficult to do a multigrid analysis, i.e., to analyse its convergence over different resolutions. The experiments provided by the authors are limited to the computation of the elastica along a disk, and we are not sure about the behavior of such estimator in shapes of more complex geometry. The fundamental goal of this thesis is to investigate imaging models using geometric penalization estimated by proven multigrid

(a) Original.

(b) Binary segmentation.



(c) Original and inpainting masks.

(d) Inpainting with length regularization.

(e) Inpainting with curvature regularization.

Figure 3.15: **Segmentation and inpainting results [nieuwenhuis14efficient].** The connectivity principle is perfectly illustrated in the inpainting problem but one would expect a completion of the rightmost bar of the camera tripod.

convergent estimators. In the next chapter we explore the multigrid convergence property and some other concepts of digital geometry.

Chapter 4

Digital Geometry

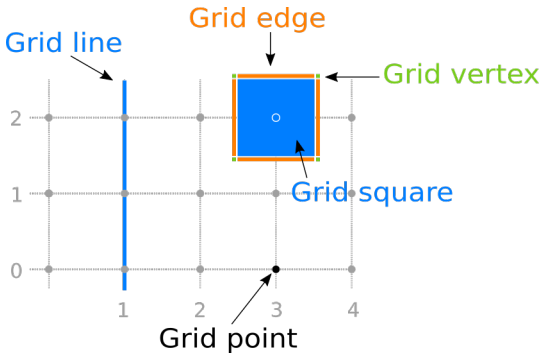
The primitives of Euclidean geometry, such as lines and points, are idealized objects. A line segment is infinitely thin and a point is dimensionless. When referring to such entities, we eventually make use of visual representations and the line is incarnated as a trace and the point as a dot in the black board, for example. As long as we have the mathematical description of some shape S, we do not need any visual representation of it to compute its tangents, curvatures or perimeter.

In imaging, however, it is almost always the case that we do not have a mathematical description of the objects in the scene. We need to identify the primitives from its visual representations. We are not dealing with idealized objects anymore, but with finite descriptions of them. One of the subjects of study of digital geometry, and the one focused in this chapter, is how to correctly measure geometric properties in digitized objects.

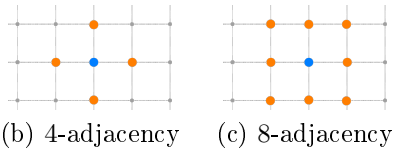
We introduce basic concepts of digital geometry in Section 4.1 and we point out the difference between exact sampling and digitization. In Section 4.2, we introduce the multigrid convergence property and we give examples of several multigrid convergent estimators for local and global geometric measurements.

4.1 Ground concepts

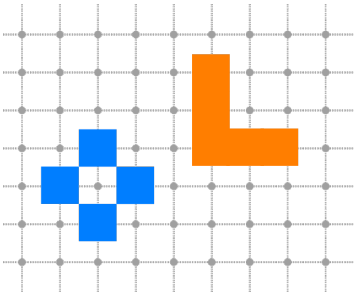
In this section we define the grid point model and its adjacency relations, which allow us to define the digital line primitive. Next, we describe the grid intersection and Gauss digitization mappings, which link continuous objects to its digital representations in the grid point model. The concepts defined in this section can be explored in much more depth in the book [**klette04digital**].



(a) Grid point model



(b) 4-adjacency (c) 8-adjacency



(d) Connected components

Figure 4.1: **Digital grid and adjacency relations.** The grid point model and its components in (a); The 4 and 8 adjacency relations in (b) and (c); A 4-connected set (orange) and a 8-connected set (blue) in (d).

4.1.1 Digital grid, digitization and digital line

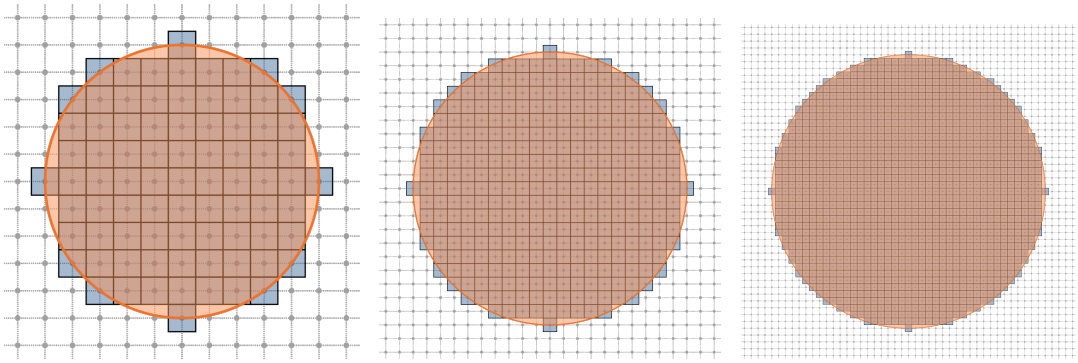
Definition 1(*2D* **Digital grid**)**:** The *2D* digital grid $h\mathbb{Z}^2 = \{p = (h \cdot i, h \cdot j) \mid i, j \in \mathbb{Z}\}$ with grid step (resolution) $h \in \mathbb{R}^+ \setminus \{0\}$ is a regular sampling of the plane. Its members are called *grid points*.

We can think of the digital grid as a regular tessellation of the plane. The grid points are the center of the *grid squares* (pixels). The other components, *grid edges* and *grid vertices* are illustrated in Figure 4.1a. They form the *grid point model* of the plane. Next we define the two commonest adjacency relations in the grid point model.

Definition 2(4**-adjacency relation**)**:** Two grid points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ are 4-adjacent iff $p_1 \neq p_2$ and $|x_1 - x_2| + |y_1 - y_2| = 1$. We denote relation membership as $p_1 A_4 p_2$.

Definition 3(8**-adjacency relation**)**:** Two grid points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ are 8-adjacent iff $p_1 \neq p_2$ and $\max\{|x_1 - x_2|, |y_1 - y_2|\} = 1$. We denote relation membership as $p_1 A_8 p_2$.

The adjacency relations are illustrated in Figures 4.1b and 4.1c. Armed with an adjacency relation, we can define the notions of path and connectivity. A 4-*connected path* is a sequence $\{p_1, p_2, \cdots, p_n\}$ of grid points such that $p_i A_4 p_{i+1}$ for



(a) $h = 1.0$ (b) $h = 0.5$ (c) $h = 0.25$

Figure 4.2: **Gauss digitization.** Digitization of a ball of radius 5 in three different resolutions.

all $1 \leq i < n$. A set P is 4-*connected* if for every $p, q \in P$ there exists a 4-connected path starting at p and ending at q. Analogous definitions holds for 8-adjacency relation (see Figure 4.1d).

The adjacency relations are also used to define the $4, 8$ neighborhood sets of a grid point.

$$\mathcal{N}_4(p) = \{q \mid pA_4q\}.$$
$$\mathcal{N}_8(p) = \{q \mid pA_8q\}.$$

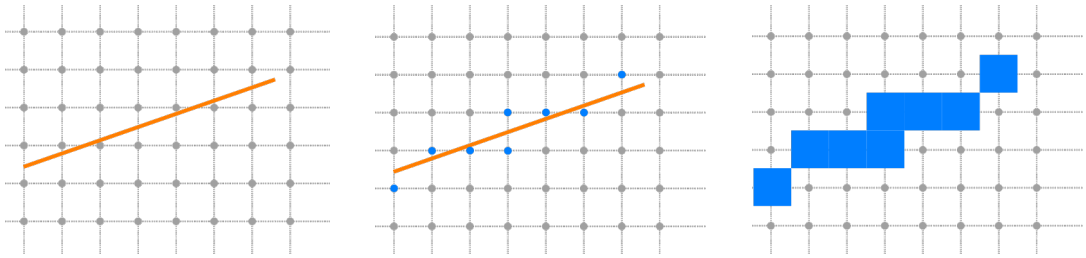
Next, we are going to define mappings that will give us the grid point representation of continuous objects.

Definition 4(Gauss digitization): Let $S \subset \mathbb{R}^2$ a shape in the plane. Its Gauss digitization $D_h(S)$ in a digital grid of resolution $h \in \mathbb{R}^+ \setminus \{0\}$ is defined as the set of grid points contained in S.

In Figure 4.2 we show the Gauss digitization of an Euclidean ball of radius 5. As one could expect, the Gauss digitization is not adequate for the digitization of lines. In this case, the grid intersection digitization is preferred.

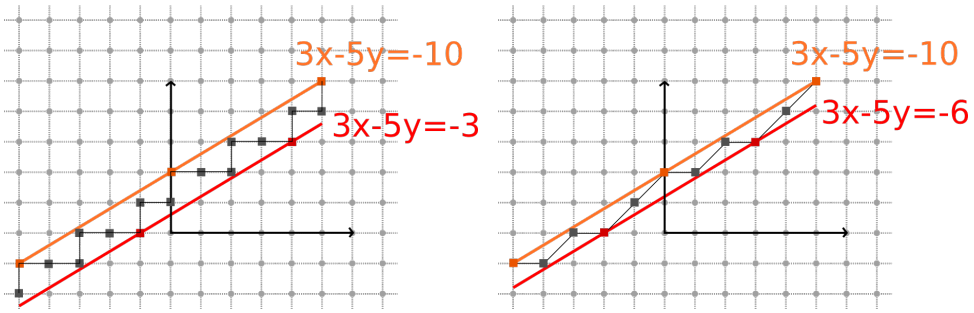
Definition 5(Grid intersection digitization): The grid intersection digitization of a planar curve γ is the set of all grid points in the digital grid that are closest (Euclidean distance) to the intersection points of γ with the grid lines.

An illustration of grid intersection digitization is given in Figure 4.3. The digitization gives us a mapping from continuous objects to its digital grid representation,



(a) (b) (c)

Figure 4.3: **Grid intersection digitization.** The Euclidean line is superposed in the digital grid in a); the closest grid points from the intersection with the grid lines are highlighted in blue in b); the grid intersection digitization of the line segment. The digitized segment is 8-connected.



(a) $\varepsilon = |a| + |b|$ (b) $\varepsilon = \max\{|a|, |b|\}$

Figure 4.4: **Digital straight segment.** In a) we have a 4-connected DSS and in b) a 8-connected DSS.

but an operation in the opposite direction is necessary if we wish to identify geometric primitives. Alternatively, one could define digital geometry primitives, for example, the digital counterpart of a line segment.

Definition 6(Digital straight segment (DSS)): Let $a, b, \mu \in \mathbb{Z}$ such that $\gcd(a, b) = 1$. The digital straight segment of tangent a/b shifted of μ from the origin and of width ε is any subset of $\mathbb{Z}^2$ satisfying

$$\{(x, y) \mid \mu \leq ax - by \leq \mu + \varepsilon - 1\}.$$

The value of ε defines the connectivity of the DSS, as illustrated in Figure 4.7a. In applications we are going to be interested in the recognition of maximal DSS, for which linear time algorithms are available (some of them described in [**klette04digital**], chapter 9). A maximal DSS is a DSS that is not contained in any other DSS. The

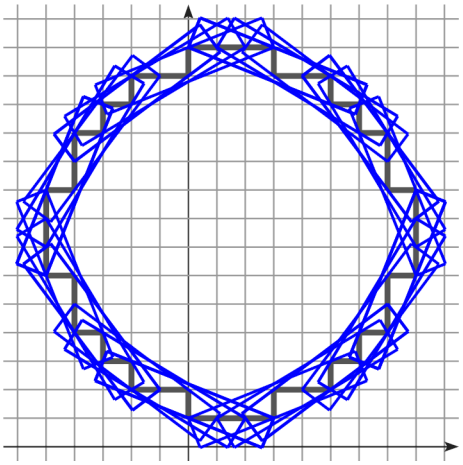


Figure 4.5: **Tangential conver.** The tangential cover is the collection of all maximal DSS in the digital contour of some digital shape. (Extracted from [**lachaud13multigrid**])

collection of maximal DSS's in the boundary of a digital set form its *tangential cover* (see Figure 4.5). The tangential cover computation is a fundamental step in the computation of convergent estimators of tangent, as we are going to see in the next section.

Finally, we refer to the *digital contour* $\partial_h S$ of $D_h(S)$ as the collection of grid vertices on the boundary of the axis-aligned polygonal shape formed by the grid squares of $D_h(S)$.

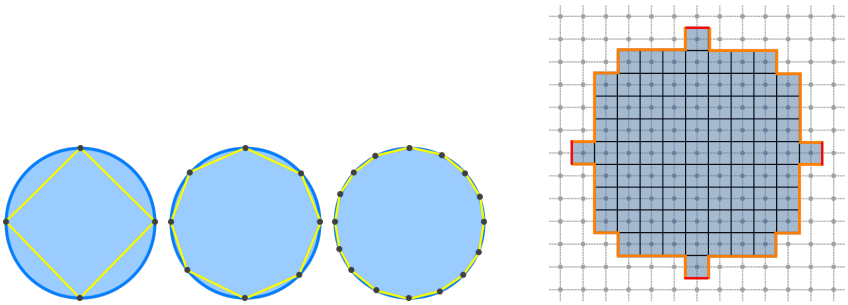
4.1.2 Exact sampling versus digitization

Let S some regular shape in the plane, for example, a disk of radius r. Assume that we need to measure the perimeter of S but we do not know that S is a disk. Additionally, let us assume that we can ask to an oracle for a collection of n points on the boundary of S. To make things simple, let us assume that the oracle gives us an ordered sequence of n uniformly spaced points on the boundary of S for a given orientation of ∂S.

We could estimate the perimeter of S by simply computing the length of the polygon connecting the n points, i.e.,

$$L(S) \approx \sum_{i=1}^{n-1} \|\overline{p_i p_{i+1}}\|.$$

This estimation converges to $2\pi r$ as the number of sampling points increase. Now, let us consider the scenario in which we have a digitization of S. The oracle gives



(a) (b)

Figure 4.6: **Exact sampling x digitization.** The perimeter is approximated by the Euclidean polygon formed by n points in the boundary of S in (a). The perimeter is estimated by counting the number of grid edges in (b).

us a digitization of S in any resolution h. Can we measure the length of S using the same strategy employed in the previous scenario?

Let us say that we engage into estimating the length of S by computing the length of the axis-aligned polygon derived from the digitization of S, as illustrated in Figure 4.6. We assume that the disk is centered at a grid point. It is easy to see that for each quarter of the disk, we have $r/h + 2$ horizontal steps and $r/h + 2$ vertical steps. Therefore,

$$\hat{L}(S) = 4h(r/h + r/h + 1) = 8r + 4h.$$

The estimator converges to $8r$ as the resolution increases. Several estimators based on the assignment of weights to local configurations were proposed. The BLUE (best linear unbiased) estimator [**dorst87length**] estimates perimeter as

$$\hat{L}(S) = h(0.948n_i + 1.343n_d),$$

where n_i are the number of axis-aligned steps and n_d the number of diagonal steps. Other estimators propose to use an extended set of configurations to cover a higher number of directions, but none of these approaches cannot achieve multigrid convergence, whatever the finite number of configurations employed [**tajine03local**].

This simple example points out that standard discretization strategies of continuous measures or energies, as the one proposed for the discrete elastica in Chapter 3, do not necessarily extend well in the digital world. The fundamental issue with digital objects is that we have to handle *digitization errors*. We do not have an exact sampling of the continuous object.

4.2 Geometric measurements in digital objects

4.2.1 Multigrid convergence and perimeter estimation

As exemplified in the previous section, geometric measurements in digital objects can be tricky. Intuitively, a good estimator should converge to its continuous counterpart value as the grid resolution is refined. The criteria that formalizes this intuition is the multigrid convergence property.

Definition 1(Multigrid convergence):
Let $\mathcal{F}$ a family of shapes in the plane and Q a global measurement (e.g., perimeter, area) on members of $\mathcal{F}$. Additionally, denote $D_h(S)$ a digitization of shape S in a digital grid of resolution h. The estimator $\hat{Q}$ of Q is multigrid convergent for the family $\mathcal{F}$ if and only if for every shape $S \in \mathcal{F}$, there exists $h_S > 0$ such that

$$\forall h \leq h_S, \quad |\hat{Q}(D_h(S), h) - Q(S)| \leq \tau_S(h),$$

where $\tau_S : \mathbb{R}^+ \setminus \{0\} \to \mathbb{R}^+$ is the speed of convergence of $\hat{Q}$ towards Q for S.

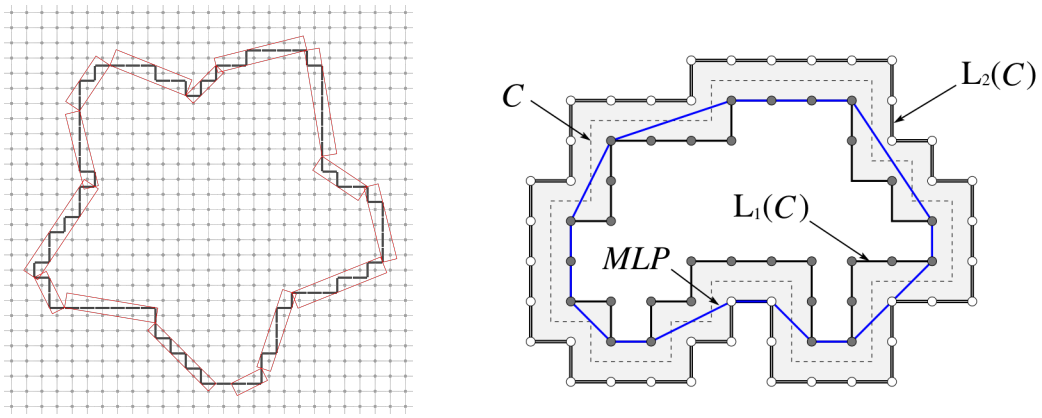
In the following, we give some examples of multigrid convergent estimators for perimeter.

DSS estimator [kovalevsky92theoretical]: It estimates the perimeter by partitioning the digital contours in a sequence of longest DSS's. Starting from any point p, it finds the longest DSS starting from p and it repeats the process until all the digital contour is covered. This set of DSS's defines a polygon whose perimeter is the estimated value (see Figure 4.7a). It is multigrid convergent for the family of piecewise 3-smooth convex shapes and also convex polygons.

MLP estimator [sloboda98approximation]: It estimates the perimeter of the digital contour as the perimeter of the minimum length polygon that separates interior grid points from exterior grid points (see Figure 4.7b). It is multigrid convergent for all finite convex shapes.

The DSS estimator can be implement in linear-time by using a linear-time DSS recognition algorithm. A linear-time algorithm for the MLP computation is also available [**provenccal09two**].

Perimeter and area are examples of global properties of shape S, i.e., they are not defined at points in S but for the whole shape. A multigrid convergent estimator for area consists in simply counting the number of grid points in its digitization and scale it by h^2 [**klette00multigrid**]. A detailed comparison of several multigrid convergent estimators can be found in [**coeurjolly12multigrid**].



(a) DSS estimator (b) MLP estimator (extracted from [**provenccal09two**])

Figure 4.7: **Multigrid convergent estimators for perimeter.** The perimeter is estimated by sum of the lengths of the recognized DSS in (a). The perimeter is estimated as the length of the minimum length polygon separating inner and outer pixels in (b).

4.2.2 Tangent and multigrid convergence of local quantities

Tangent and curvature are examples of local properties computed along the boundary of some shape S in the plane. We need a slight different definition of multigrid convergence in order to map points of the Euclidean boundary to those in the digital contour.

Definition 2(Multigrid convergence for local geometric quantities): Let $\mathcal{F}$ a family of shapes in the plane and Q a local measurement along the boundary ∂S of $S \in \mathcal{F}$. Additionally, denote $D_h(S)$ a digitization of S in a digital grid of resolution h and $\partial_h S$ its digital contour. The estimator $\hat{Q}$ of Q is multigrid convergent for the family $\mathcal{F}$ if and only if for every shape $S \in \mathcal{F}$, there exists $h_S > 0$ such that the estimate $\hat{Q}(D_h(S), p, h)$ is defined for all $p \in \partial_h S$ with $0 < h < h_S$, and for any $x \in \partial S$,

$$\forall p \in \partial_h S \text{ with } \|p - x\|_\infty \leq h, \quad |\hat{Q}(D_h(S), p, h) - Q(S, x)| \leq \tau_S(h),$$

where $\tau_S : \mathbb{R}^+ \setminus \{0\} \to \mathbb{R}^+$ has null limit at 0. This function defines the speed of convergence of $\hat{Q}$ towards Q for S.

The λ-MST tangent estimator [**lachaud07tangent**] computes the tangential cover of $D_h(S)$ and then estimates the tangent direction at point $p \in \partial_h S$ as a weighted combination of the tangents directions $\tan^{-1}(a/b)$ of maximal DSS

(a, b, μ, ε) passing through p. We recall that we defined digital contours as a collection of grid vertices, but we can always interpret them as grid points. It is enough to translate every grid vertex by $h(0.5, 0.5)$.

For a given order in the points of $\partial_h S$, a maximal DSS starting at p_i and ending at p_j is denoted C_{ij}. For each $p_k \in \partial_h S$, let $\mathcal{C}(p_k)$ be the set of maximal DSS's passing through p_k. The eccentricity of a point p_k is defined as

$$e(p_k) = \begin{cases} \frac{|k-j|}{|i-j|}, & \text{if } C_{ij} \in \mathcal{C}(p_k) \\ 0, & \text{otherwise.} \end{cases}$$

The tangent direction at p_k is estimated as

$$\hat{\theta}(p_k) = \frac{\sum_{C_{ij} \in \mathcal{C}(p_k)} \lambda(e(p_k)) \tan^{-1}(a_{ij}/b_{ij})}{\sum_{C_{ij} \in \mathcal{C}(p_k)} \lambda(e(p_k))},$$

where λ is a mapping from $[0, 1]$ to $\mathbb{R}^+$ with $\lambda(0) = \lambda(1) = 0$ and $\lambda > 0$ elsewhere. The λ-MST estimator is multigrid convergent for the family of convex shapes that are twice differentiable with continuous curvature. The convergence speed is of $O(h^{1/3})$ [**lachaud07tangent**].

The λ-MST estimator can be used to estimate the contribution of each grid edge to the perimeter of the shape, the so-called *elementary length*. Let $\{\boldsymbol{e}_i\}$ be the collection of grid edges (vectors) in the digital contour of $D_h(S)$. We compute the λ-MST estimator for the sequence of points formed by the center points $\dot{\boldsymbol{e}}_i$ of each $\boldsymbol{e}_i$. The elementary length is defined as

$$\hat{\ell}(\boldsymbol{e}_i) = \big(\sin(\hat{\theta}(\dot{\boldsymbol{e}}_i)), \cos(\hat{\theta}(\dot{\boldsymbol{e}}_i))\big) \cdot \boldsymbol{e_i}.$$

One can integrate the elementary length to obtain a multigrid convergent estimator for the perimeter of S. More generally, given a measurement g in ∂S and a multigrid convergent estimator $\hat{g}$ of g in $\partial_h S$, the expression

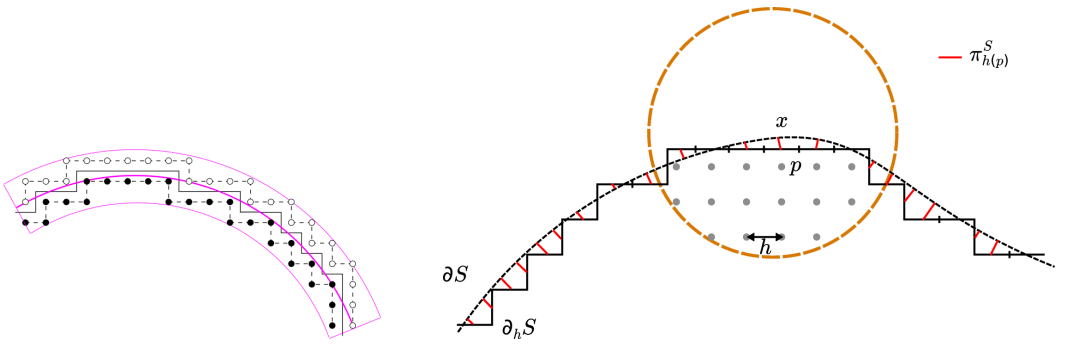
$$\sum_{\boldsymbol{e}_i \in \partial_h S} \hat{g}(\boldsymbol{e}_i)\hat{l}(\boldsymbol{e}_i),$$

is a multigrid convergent estimator for the energy [**lachaud06hdr**]

$$\int_{\partial S} g ds.$$

4.2.3 Multigrid convergent estimators of curvature

Analogously to digital straight segments, the *digital circular arc* is the digital counterpart of a circular arc. For any $C \in \partial_h S$, a grid point p is an interior (exterior)



(a) Digital circular arc (extracted from [**roussillon11mdca**])

(b) II estimator (extracted from [**coeurjolly13integral**])

Figure 4.8: **Curvature estimation.** The digital circular arc separates the interior grid points (black) from the exterior grid points (white) in (a). The back-projection operator maps digital grid points to points in ∂S in (b).

point of C if $p \in D_h S$ ($p \notin D_h S$) and there exists a grid edge in C that is incident to the grid square corresponding to p.

Definition 3(Digital circular arc): A segment $C \in \partial_h S$ is a digital circular arc if and only if the interior and exterior grid points of C are circularly separable, i.e., there exists an Euclidean circle that either encloses the interior points without enclosing any exterior points or that encloses the exterior points without enclosing any interior point.

An illustration is given in Figure 4.8a.The Maximal Digital Circular Arcs estimator (MDCA) [**roussillon11mdca**] estimates the curvature at point $p \in \partial_h S$ as the inverse of the radius of the most centered digital circular arc that contains p. The MDCA estimator is multigrid convergent for the family of convex shapes in the plane with continuous, strictly positive and bounded curvature. An alternative is the λ-MDCA estimator [**schindele17mdca**], which follows the same rational of the λ-MST. The λ-MDCA has been proven multigrid convergent for the same family of convex shapes with continuous curvature. It converges with speed $O(h^{1/3})$.

The next curvature estimator is based on the concept of integral invariants. Generally speaking, an invariant is a function whose value is unaffected by the action of some group on the elements of its domain. The curvature, for example, is an invariant for shapes in $\mathbb{R}^2$ with respect to the Euclidean group of rigid transformations. However, curvature is a second order differential invariant and its computation is very sensitive to noise. In the digital grid, the issues appearing in tangent estimation are amplified for curvature estimation.

An integral invariant, on the other hand, is a value computed via integration and one can expect to be much more robust to noise. In the context of digital estimation, an integral invariant is attractive because we have already a very simple multigrid convergent estimator to estimate the area. In [**manay04intinvariant**], the authors define the integral area invariant and link it with the measurement of curvature

Definition 4(Integral area invariant): Let $S \subset \mathbb{R}^2$ and $B_r(x)$ the Euclidean ball of radius r centered at point x. Further, let $\mathbb{1}_S(\cdot)$ be the characteristic function of S. The integral area invariant $\sigma_{S,r}(\cdot)$ is defined as

$$\forall x \in \partial S, \quad \sigma_{S,r}(x) = \int_{B_r(x)} \mathbb{1}_S(y)dy.$$

The value $\sigma_{S,r}(x)$ is the area of the intersection of the ball $B_r(x)$ with shape S. By approaching the shape at point $x \in S$, one can rewrite the intersection area $\sigma_{S,r}(x)$ in the form of the Taylor expansion [**pottman09intinvariant**]:

$$\sigma_{S,r}(p) = \frac{\pi}{2}r^2 - \frac{\kappa(S,x)}{3}r^3 + O(r^4),$$

where $\kappa(S,x)$ is the curvature of S at point x. By isolating κ we can define a curvature estimator

$$\tilde{\kappa}(x) \coloneqq \frac{3}{r^3}\left(\frac{\pi r^2}{2} - \sigma_{S,r}(x)\right). \tag{4.1}$$

In [**coeurjolly13integral**], the authors combine the approximation Equation (4.1) and the estimator of area to define a multigrid convergent estimator for the curvature (see Figure 4.8b).

Definition 5(Integral Invariant Curvature Estimator): Let $D_h(S)$ a digitization of $S \subset \mathbb{R}^2$. The integral invariant curvature estimator is defined for every point $p \in \partial_h S$ as

$$\hat{\kappa}_r(D_h(S), p, h) \coloneqq \frac{3}{r^3}\left(\frac{\pi r^2}{2} - \widehat{\text{Area}}\left(D_h\big(B_r(p)\big) \cap D_h(S), h\right)\right).$$

where $\widehat{\text{Area}}(D,h)$ estimated the area of D by counting its grid points and then scaling them by h^2. This estimator is multigrid convergent for the family of compact shapes in the plane with 3-smooth boundary. It converges with speed $O(h^{\frac{1}{3}})$ for radii chosen as $r = \Theta(h^{\frac{1}{3}})$ [**lachaud17robust**].

4.3 Conclusion

In imaging problems it is very rare to have a mathematical description of the objects in the scene. If we wish to measure geometric properties on these objects, we have to deal with its digitization error. In contrast with exact sampling, whose points can be located everywhere in $\mathbb{R}^2$, the digital grid restricts the sampling to a subset of $h\mathbb{Z}^2$. In this chapter, we have shown that we cannot extend standard discretization strategies to do geometric measurements of digital objects. Instead, we should study the problem from the point of view of digital geometry and the multigrid convergence property.

Part II

Contributions

Chapter 5

A combinatorial model for digital elastica shape optimization

The goal of this chapter is to experimentally validate a model for the elastica energy using multigrid convergent estimators of length and curvature. In Section 5.1 we introduce the digital elastica and in Section 5.2 we present a combinatorial optimization model capable to evolve a shape to another of lower digital elastica energy. In several occasions, the final shape is indeed the optimal one, which confirms the pertinence of using multigrid convergent estimators to optimize geometric-related energies in digital sets. In Section 5.4, we present several attempts to derive a global model to minimize a simplification of the digital elastica and we discuss why they fail.

5.1 Digital elastica

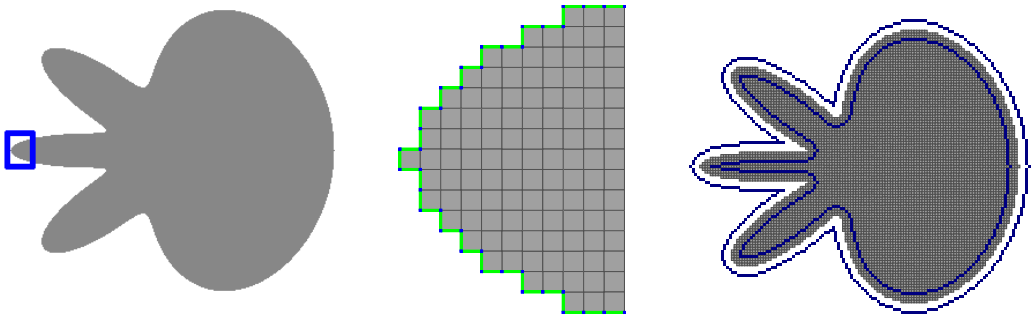
The elastica energy of parameters $\boldsymbol{\theta} = (\alpha \geq 0, \beta \geq 0)$ for some Euclidean shape $S \subset \mathbb{R}^2$ is defined as

$$E_{\boldsymbol{\theta}}(S) = \int_{\partial S} \alpha + \beta \kappa(s)^2 ds.$$

Similarly, the digital elastica $\hat{E}_{\boldsymbol{\theta}}$ of some digitization $D_h(S)$ of S is defined as

$$\hat{E}_{\boldsymbol{\theta}}(D_h(S)) = \sum_{\dot{\boldsymbol{e}} \in \partial_h S} \hat{s}(\dot{\boldsymbol{e}}) \left(\alpha + \beta \hat{\kappa}^2(D_h(S), \dot{\boldsymbol{e}}, h) \right), \tag{5.1}$$

where $\dot{\boldsymbol{e}}$ denotes the center of the linel $\boldsymbol{e}$ and the estimators of length $\hat{s}$ and curvature $\hat{\kappa}$ are multigrid convergent. In the expression above, we will substitute an arbitrary subset D of $\mathbb{Z}^2$ to $D_h(S)$ since the continuous shape S is unknown. In the following



(a) (b) (c)

Figure 5.1: **Cellular grid model and m-ring set.** The flower shape in figure (a) and the cellular-grid model representation in (b) of the blue bounded rectangle region in (a). In figure (b), pixels are colored in gray, linels in green and pointels in blue. In figure (c), the blue pixels denotes a 3-ring set.

we omit the grid step h to simplify expressions (or, putting it differently, we assume that the shape of interest is rescaled by $1/h$ and we set $h = 1$).

In the next section, we describe a combinatorial scheme whose aim is to optimize the digital elastica energy Equation (5.1).

5.2 Local combinatorial scheme

Given a digital shape $D^{(0)}$ we describe a process that generates a sequence $D^{(i)}$ of shapes with non-increasing elastica energy. The idea is to define a neighborhood of shapes $\mathcal{W}^{(i)}$ to the shape $D^{(i)}$ and choose the element of $\mathcal{W}^{(i)}$ with lowest energy.

Let D be a 2-dimensional digital shape embedded in a domain $\Omega \subset \mathbb{Z}^2$. We adopt the cellular-grid model to represent D, i.e., pixels and its lower dimensional counterparts, linels and pointels, are part of D (see Figure 5.1b). In particular, we denote by ∂D the topological boundary of D, i.e., the connected sequence of linels such that for each linel we have one of its incident pixels in D and the other not in D.

Let $d_D : \Omega \to \mathbb{R}$ be the signed Euclidean distance transformation with respect to shape D. The value $d_D(p)$ gives the Euclidean distance between $p \notin D$ and the closest pixel in D. For points $p \in D$, $d_D(p)$ gives the negative of the distance between p and the closest pixel not in D.

Definition 1(Inner pixel boundary):
Given a digital shape D embedded in a domain Ω, we define its inner pixel boundary set $I(D)$ as

$$I(D) := \{\, p \mid p \in D, |\mathcal{N}_4(x) \cap D| < 4 \,\},$$

where $\mathcal{N}_4(p)$ denotes the 4-adjacent neighbor set of p (without p).

Definition 2(m-Ring Set): Given a digital shape $D \in \Omega$, its signed distance transformation d_D and natural number $m \geq 0$ the *m-ring set of D* is defined as

$$R_m(D) := L_m \cup L_{-m},$$

where

$$L_m(D) := \begin{cases} I(D), & m = 0 \\ \{p \in \Omega \mid m - 1 < d_D(p) \leq m\}, & m > 0 \\ \{p \in \Omega \mid m + 1 > d_D(p) \geq m\}, & m < 0 \end{cases}$$

Thus, the m-ring is composed of two sets of pixels with positive and negative distances to the shape D if $m > 0$ (see Figure 5.1c), and equal to the inner pixel boundary in the case $m = 0$. Consider the following set of neighbor candidates to D:

$$\{Q \mid Q \subset R_1(D) \cup D \text{ and } Q \text{ is connected}\}.$$

Such set can be extremely large and its complete exhaustion is prohibitively expensive. Instead, we are going to use a subset of it.

Definition 3(n-neighborhood):
Given a digital shape $D \subset \Omega$, its n-neighborhood $\mathcal{W}_n(D)$ is defined as the set of digital shapes that can be built from D by adding or removing a sequence of $k \in [0, n]$ connected pixels in $R_1(D)$.

In Figure 5.2 it is shown two members of the $\mathcal{W}_{12}$ neighborhood. At first glance, we may be tempted to set the local-search neighborhood at the k-th iteration as the union of all n-neighborhood for $1 < n < |\partial D^{(k)}|$. However, that is often unnecessary and time consuming, as the greatest reduction in digital elastica for a member of $\mathcal{W}_n$ is likely very close to the greatest reduction for a member of $\mathcal{W}_{n-1}$. Moreover, we can improve running time by implementing a multiscale approach, i.e., we look for reductions in digital elastica for larger values of n first, and in case of a negative answer we refine our search by choosing a smaller n.

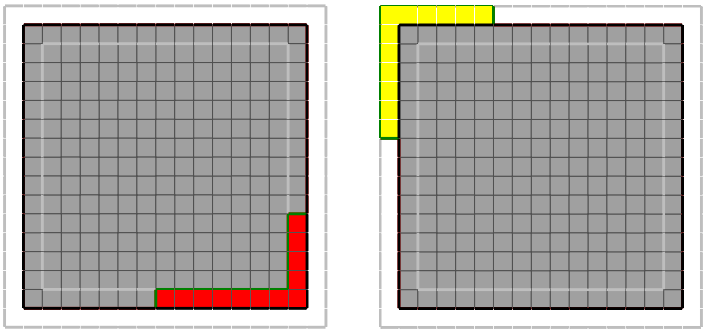


Figure 5.2: **12-Neighborhood of the square shape.** The black linels mark the digital contour of the original shape and the gray linels mark the 1-ring. In the left, a member of $\mathcal{W}_{12}$ in which the red pixels were removed; and in the right a member of $\mathcal{W}_{12}$ in which the yellow pixels were added.

The *LocalSearch* Algorithm 1 describes the local combinatorial process and is suitable for any combination of multigrid convergent estimators of perimeter and curvature. In our experiments, we set the λ-MST [**lachaud07tangent**] to estimate elementary length and perimeter. We compare our results for two curvature estimators: the MDCA [**roussillon11mdca**] and the integral invariant (II-r) estimator [**coeurjolly13integral**] (r denoting the radius of the estimation ball).

5.3 Experimental results

We study the behavior of Algorithm 1 in two problem configurations.

Free elastica. The digital elastica Equation (5.1) is optimized without any constraint.

Constrained elastica. We force pixels to be present in the final shape or we impose an orientation in the endpoints of a segment of the digital contour.

5.3.1 Free elastica

As observed in Section 3.3, the closed planar curve that minimizes the free elastica is the circumference of a disk of radius $(\beta/\alpha)^{1/2}$

$$\partial B_{(\beta/\alpha)^{1/2}} = \arg\min_{C} \int_{C} (\alpha + \beta\kappa^2) ds.$$

input : A digital set D; weight coefficients $\boldsymbol{\theta} = (\alpha, \beta)$; the maximum number of iterations maxIt

$t \longleftarrow 1$; `// multiscale level`
$k \longleftarrow 0$; `// current iteration`
$D^{(0)} \longleftarrow D$;
while k $<$ maxIt **and** $t < \log_2 |\partial D^{(k)}|$ **do**
  $n \longleftarrow |\partial D^{(k)}|/2^t$; `// Maximum` n`-neighborhood value.`
  $\mathcal{W}^{(k,t)} \longleftarrow \mathcal{W}_n(D^{(k)})$;
  `//Find neighbor shape with lowest energy.`
  $Q^\star \longleftarrow D^{(k-1)}$;
  for $Q \in \mathcal{W}^{(k,t)}$ **do**
    if $\hat{E}_{\boldsymbol{\theta}}(Q) < \hat{E}_{\boldsymbol{\theta}}(Q^\star)$ **then**
      $Q^\star \longleftarrow Q$;
    end
  end
  delta $\longleftarrow \hat{E}_{\boldsymbol{\theta}}(D^{(k-1)}) - \hat{E}_{\boldsymbol{\theta}}(D^{(k)})$;
  if delta ≤ 0 **then**
    `//Better solution not found. Refine the scale.`
    $t \longleftarrow t + 1$
  end
  else
    `//Better solution found. Set` $D^{(k)}$ `and reset to highest scale.`
    $t \longleftarrow 1$;
    $D^{(k)} \longleftarrow Q^\star$;
    k $\longleftarrow$ k $+1$;
  end
end

Algorithm 1: LocalSearch algorithm for elastica minimization.

In Figure 5.3 we present the digital elastica evolution for parameters $\alpha = 0.01, \beta = 1$ and four different curvature estimators in three different scales. The shapes evolution using the II-5 estimator are shown in Figure 5.5. We observe that both II-5 and II-10 evolve the shapes to disks of radius close to the optimum value of 10. The II-3 estimator stops prematurely at a local optimum due its limited sensibility compared to II-5 or II-10, while MDCA encounters some difficulties to evolve in a high resolution setting and it also stops at some local minimum. In fact, the MDCA estimator, although with higher convergence speed, is more sensitive to noise than II, as illustrated in Figure 5.7. Nonetheless, the results can be improved by using a larger neighborhood, as illustrates Figure 5.6.

We have executed the same experiments for different parameters α to confirm the effectiveness of our approach. We observe that the plots for $\alpha = 0.001$ in Figure 5.4 follows a pattern similar to those in Figure 5.3 for $\alpha = 0.01$. In particular, the remarks for the II-3 and MDCA estimator are the same. Further, we point out that II-5 values are slightly farther from the optimum for $\alpha = 0.001$. The reason being that the shapes evolve to a ball of higher radius compared to the case $\alpha = 0.01$. At some point of the evolution for $\alpha = 0.001$, the sensibility of II-5 is not sufficient to escape from local minimum. We remark that the adoption of an automatic selection of the estimation ball radius may attenuate this problem.

5.3.2 Constrained elastica

An important advantage of Algorithm 1 is that constraints can be imposed with minimum effort. We present results for two types of constraints. In the first type, we force some pixels to be part of the final solution and in the second we impose orientations at the endpoints of a curve, as in the general elastica problem. In Figure 5.8 we compare the flows for different values of α.

We clearly observe that lower values of α produce longer curves with smoother turns, as expected. However, the local nature of the method may lead to sub-optimal solutions. A global optimization method would not only handle these issues, but would naturally possess the completion property associated with the squared curvature, which may be damped by local approaches.

5.3.3 Running time

The running time of Algorithm 1 is summarized in table Table 5.1. All the experiments in this thesis were executed on a 32-core $2.4Ghz$ CPU. Although its use in practical applications is limited, we demonstrated that digital estimators are effective in their measurements and the flows evolve as expected, reaching the global optimum for some shapes in the free elastica problem. We observe that this approach is fully digital, and we do not suffer from discretization and rounding problems, a

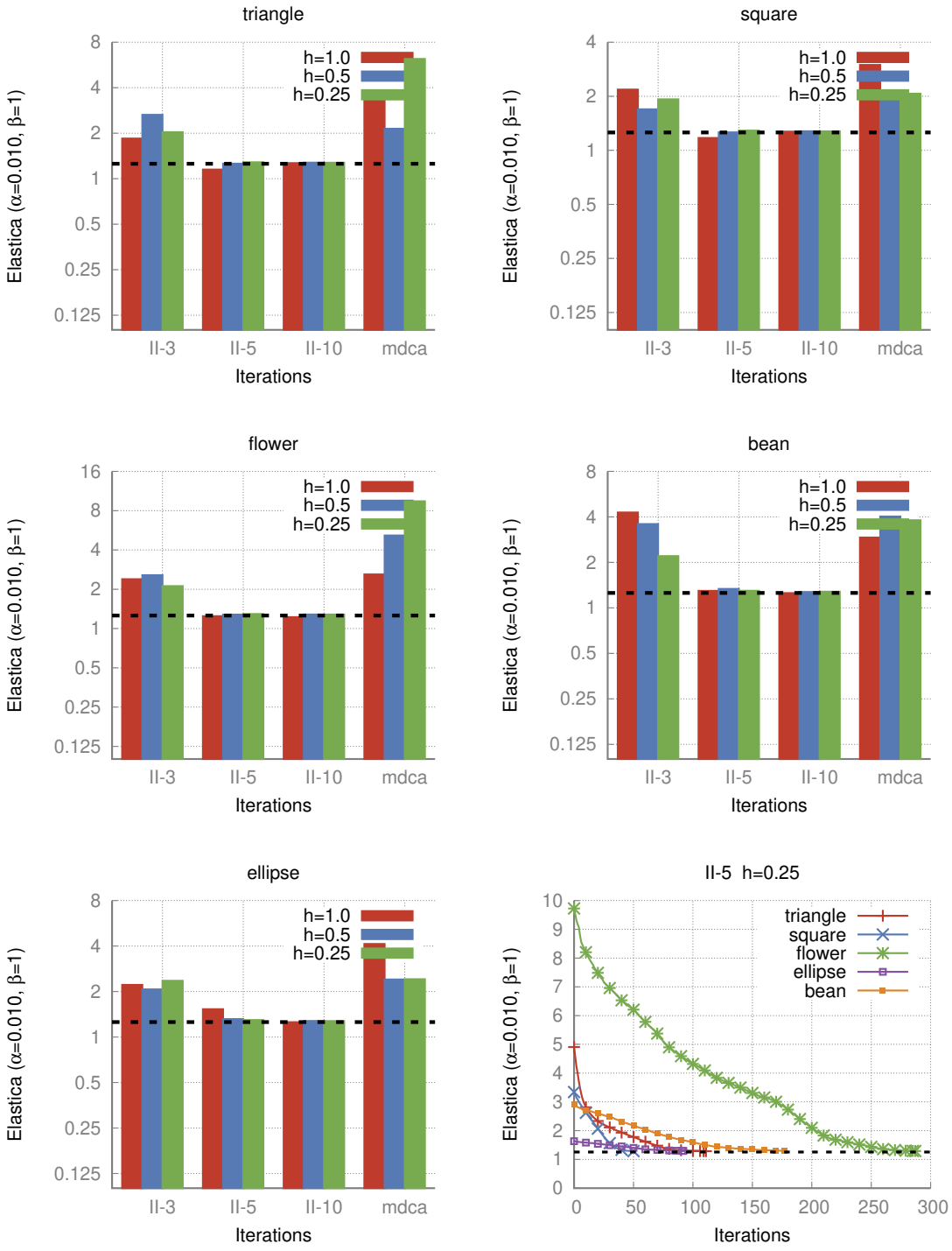


Figure 5.3: **Free elastica evolution plots for $(\alpha = 0.01, \beta = 1)$.** Minimum value attained for the digial elastica in comparison with the global optimum (dashed line) for different curvature estimators and in different scales. The last figure summarizes the digital elastica evolution value for all shapes using the II-5 estimator and grid step $h = 0.25$.

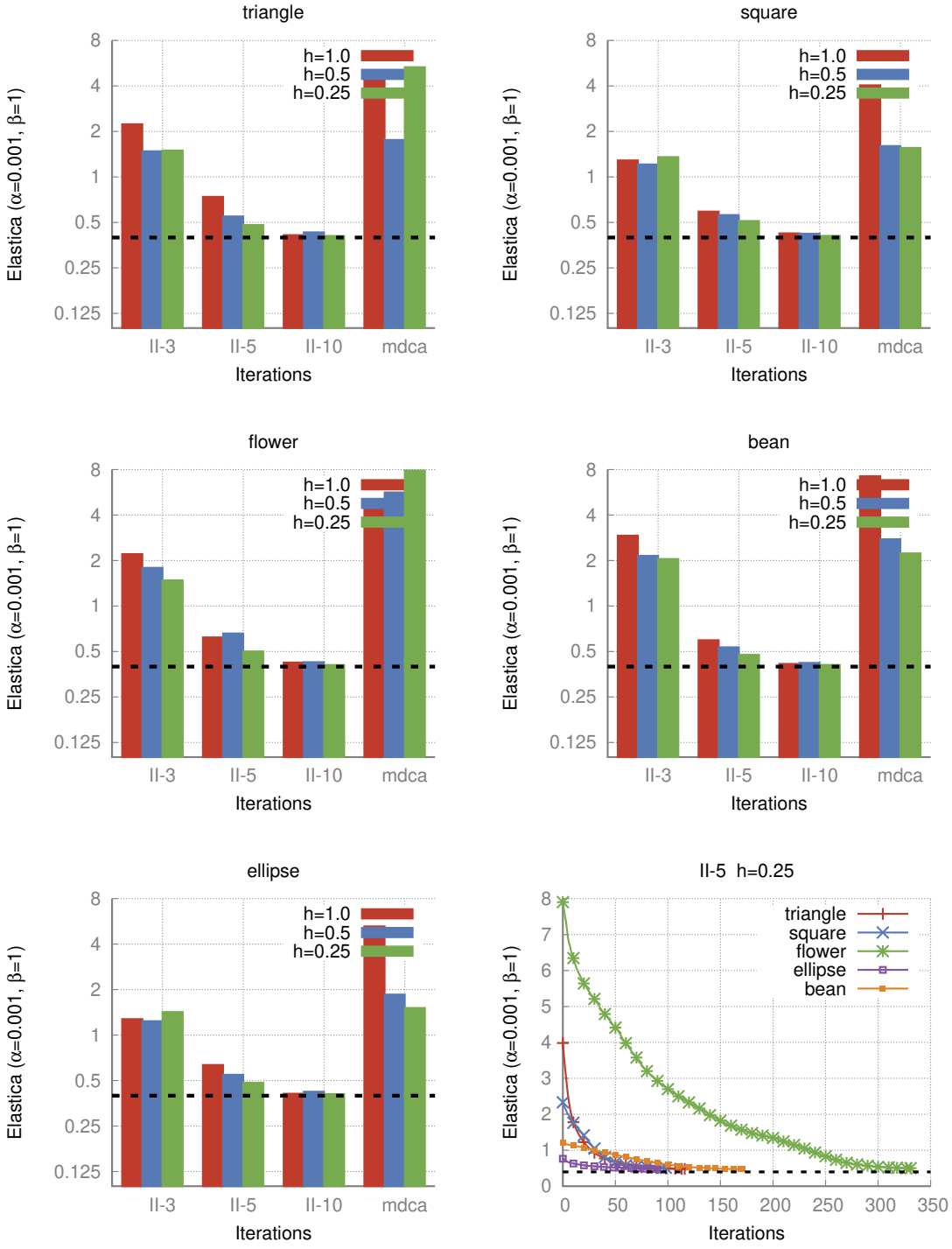


Figure 5.4: **Free elastica evolution plots for $(\alpha = 0.001, \beta = 1)$.** Minimum value attained for the digial elastica in comparison with the global optimum (dashed line) for different curvature estimators and in different scales. The last figure summarizes the digital elastica evolution value for all shapes using the II-5 estimator and grid step $h = 0.25$.

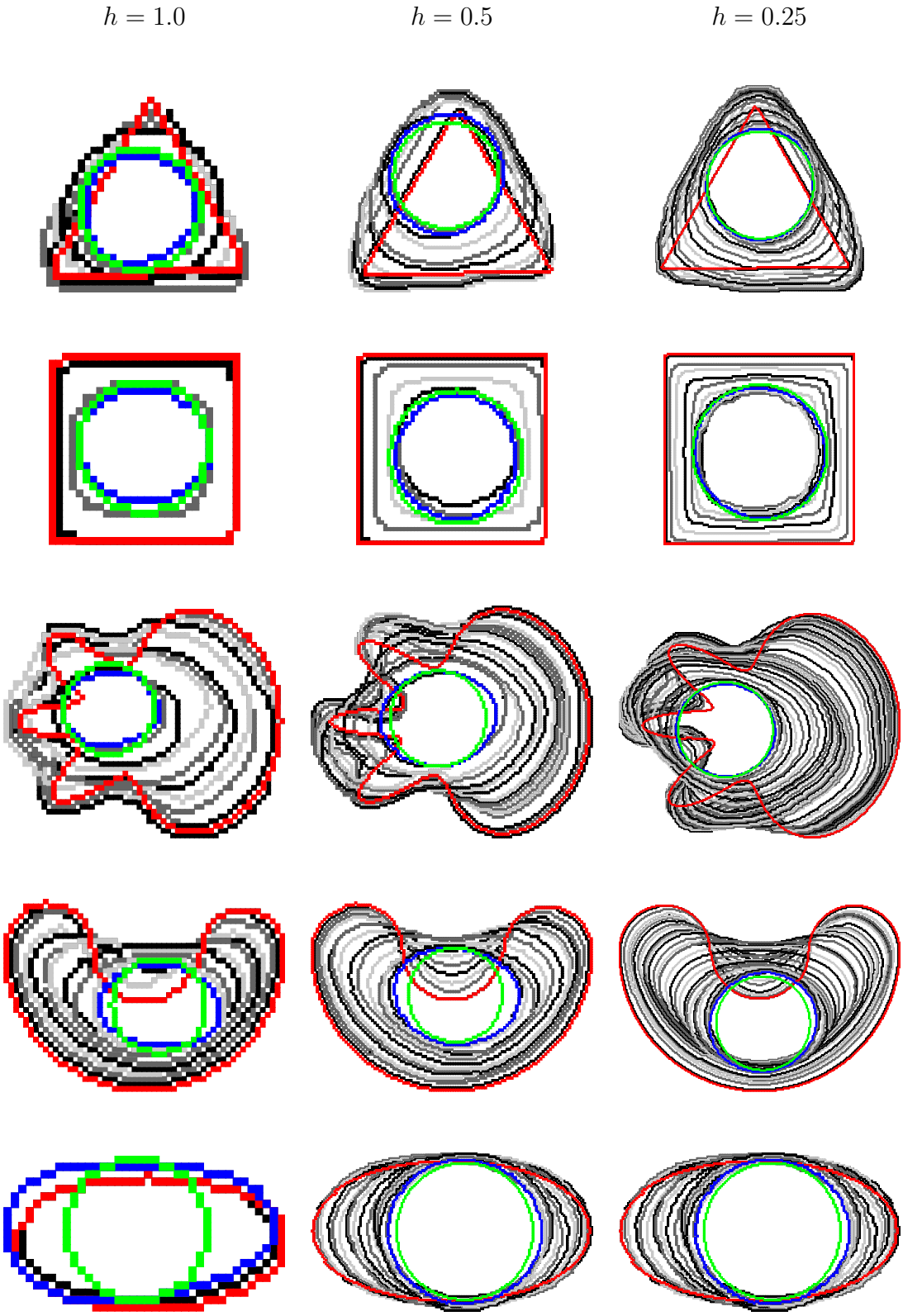


Figure 5.5: **Free elastica results for** $(\alpha = 0.01, \beta = 1)$**.**LocalSearch algorithm evolutions for several shapes. The initial contour is colored in red; the final contour is colored in blue; and the optimal contour is colored in green.

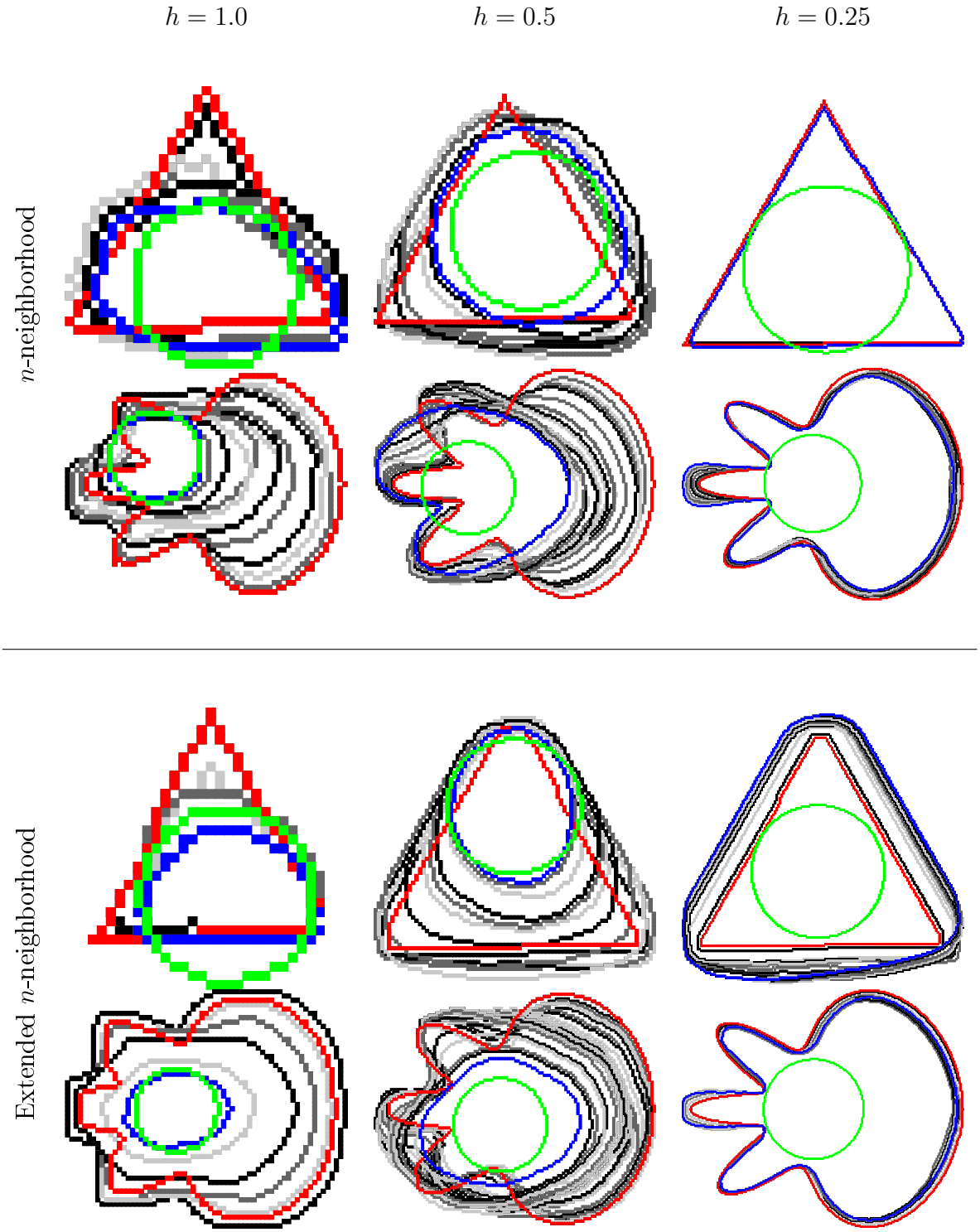
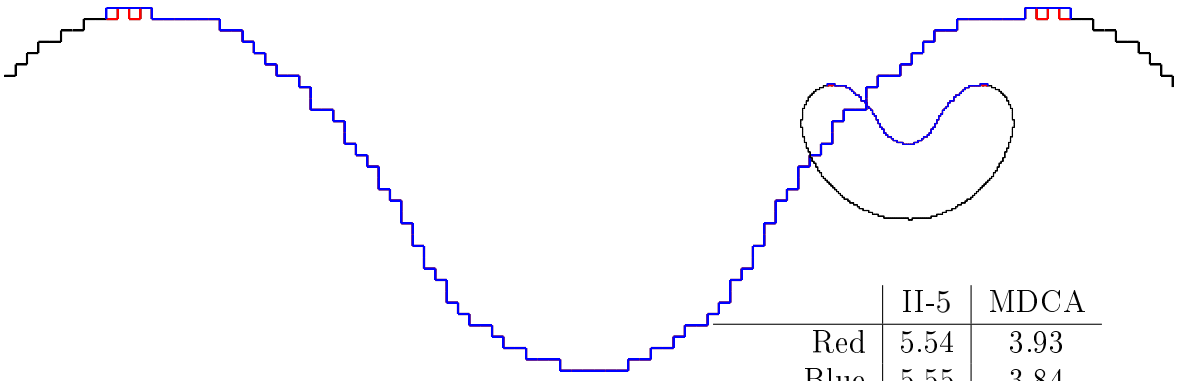


Figure 5.6: **Effects of an extended neighborhood in the MDCA evolution.** In the top row, the MDCA evolution for the neighborhood as presented in Algorithm 1. In the bottom row, the flow using the extended neighborhood. The extended neighborhood additionally includes the n-neighborhood of the dilation and the erosion of the initial shape by a square of side 1.



	II-5	MDCA
Red	5.54	3.93
Blue	5.55	3.84
$\lvert\Delta E/\Delta D\rvert$	70	1400

Figure 5.7: **MDCA sensitivity to noise.** The ratio of change in energy value (ΔE) by change of the number of pixels (ΔD) is much higher for the MDCA than for the II estimator.

common issue in continuous models. Furthermore we have checked that this approach works indifferently with Integral Invariant curvature estimator and Maximal Digital Circular Arc curvature estimator, given an appropriate neighborhood. So the convergence of the digital curvature estimator seems to be the cornerstone to get a digital curve behaving like a continuous elastica.

5.4 Global optimization

As observed in previous section, a global optimization method is important to recover the completion property of curvature, which is useful in inpainting and segmentation of thin and elongated objects. In this section we turn to a global optimization approach. However, instead of minimizing Equation (5.1) we focus on a simplified version of it in which we do not compute the local length estimator. This simplification reduces the order of the energy and will likely lead to a practical model.

5.4.1 Simplified digital elastica

The simplified digital elastica is defined as

$$\hat{E}^{simp}_{\boldsymbol{\theta}}(D_h(S)) = \sum_{\dot{\boldsymbol{e}}\in\partial_h S} \alpha + \beta\hat{\kappa}_r^2(D_h(S), \dot{\boldsymbol{e}}, h). \tag{5.2}$$

$\alpha = 0.1$ $\alpha = 0.01$ $\alpha = 0.001$

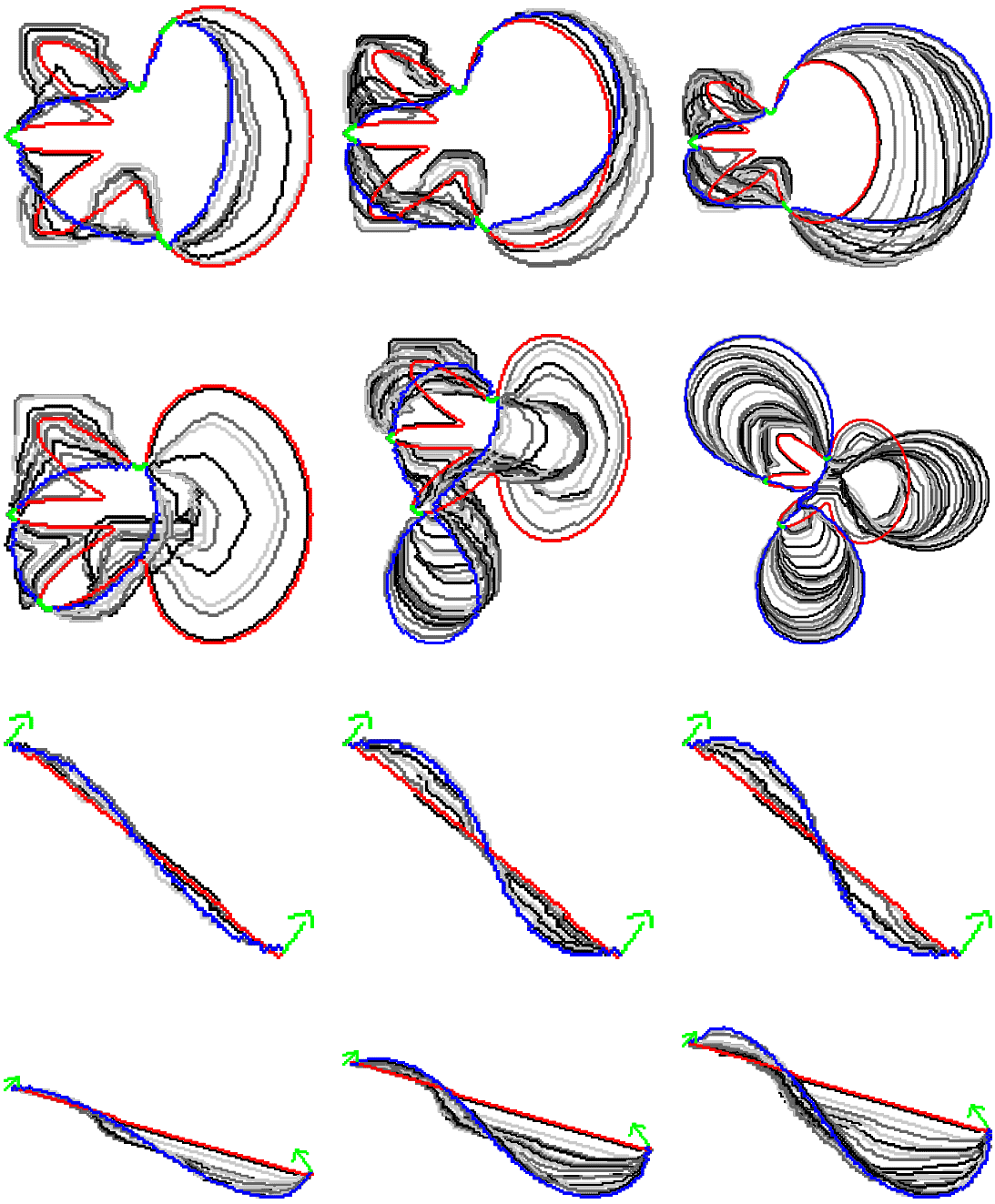


Figure 5.8: **Constrained elastica results.** In the first and second rows, the flow obtained by forcing the green pixels to be part of the final solution; In the last two rows, the flow obtained by forcing the orientation at the endpoints of the curves.

	$h = 1.0$		$h = 0.5$		$h = 0.25$	
	Pixels	Time	Pixels	Time	Pixels	Time
Triangle	521	2s (0.07s/it)	2080	43s (0.81s/it)	8315	532s(4.8s/it)
Square	841	0.9s (0.09s/it)	3249	8s (0.3s/it)	12769	102s (2s/it)
Flower	1641	13s (0.24s/it)	6577	209s (1.68s/it)	26321	3534s (12.3s/it)
Bean	1574	7s (0.16s/it)	6278	88s (1.08s/it)	25130	1131s (6.4s/it)
Ellipse	626	1s (0.14s/it)	2506	16s (0.44s/it)	10038	286s (3.1s/it)

Table 5.1: **Running time of LocalSearch.** The running times for the free elastica problem are displayed. Notice that even having a similar number of pixels, the square (bean) shape evolves much faster than the triangle (flower).

We argue that Equation (5.2) is a reasonable approximation of Equation (5.1). Indeed, executing Algorithm 1 to minimize this simplified digital elastica induces very similar results to those for the digital elastica (see Figure 5.9).

5.4.2 Optimization model for simplified digital elastica

In contrast with the previous section, the model described here is designed for the integral invariant estimator only. Let $D \subset \left(\mathbb{Z} + \frac{1}{2}\right)^2$ be the digitization of some shape $S \subset \mathbb{R}^2$ in half-integer coordinates space. We assume that D has m pixels (located at integer coordinates) and n linels (one and only one of its coordinates is $\frac{1}{2}$). Optimization variables are represented as column vectors $\boldsymbol{x} \in \{0,1\}^m$, $\boldsymbol{y} \in \{0,1\}^n$ and its i-th coefficients are denoted $\boldsymbol{x}_i, \boldsymbol{y}_i$. Further, let $\boldsymbol{A} \in \{0,1\}^{m \times n}$ the matrix defined as

$$\boldsymbol{A}_{i,j} = \begin{cases} 1, & \boldsymbol{x}_j \in B_r(\boldsymbol{y}_i) \\ 0, & \text{otherwise.} \end{cases}$$

In other words, the column vector $\boldsymbol{A}_i$ represents the pixels that are in the interior of the disk $B_r(y_i)$ of radius r centered at $\boldsymbol{y}_i$.

$$\begin{aligned} E_{\boldsymbol{\theta}}^{simp}(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{\boldsymbol{y}_i \in \boldsymbol{y}} \boldsymbol{y}_i \left(\alpha + \beta \hat{\kappa}_r^2(D, \boldsymbol{y}_i) \right) \\ &= \sum_{\boldsymbol{y}_i \in \boldsymbol{y}} \boldsymbol{y}_i \left(\alpha + \beta \big(\frac{3}{r^3} \big(\frac{\pi}{r^2} - |B_r(\boldsymbol{y}_i)| \big) \big)^2 \right) \\ &= \sum_{\boldsymbol{y}_i \in \boldsymbol{y}} \boldsymbol{y}_i \left(\alpha + \frac{9}{r^6} \beta \big(c^2 - 2c \boldsymbol{A}_i^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{A}_i \boldsymbol{A}_i^T \boldsymbol{x} \big) \right), \end{aligned} \tag{5.3}$$

where $c = \pi r^2/2$. We remark that linels and pixels in the solution must be topologically consistent, i.e., linels must form connected closed curves and the pixels must

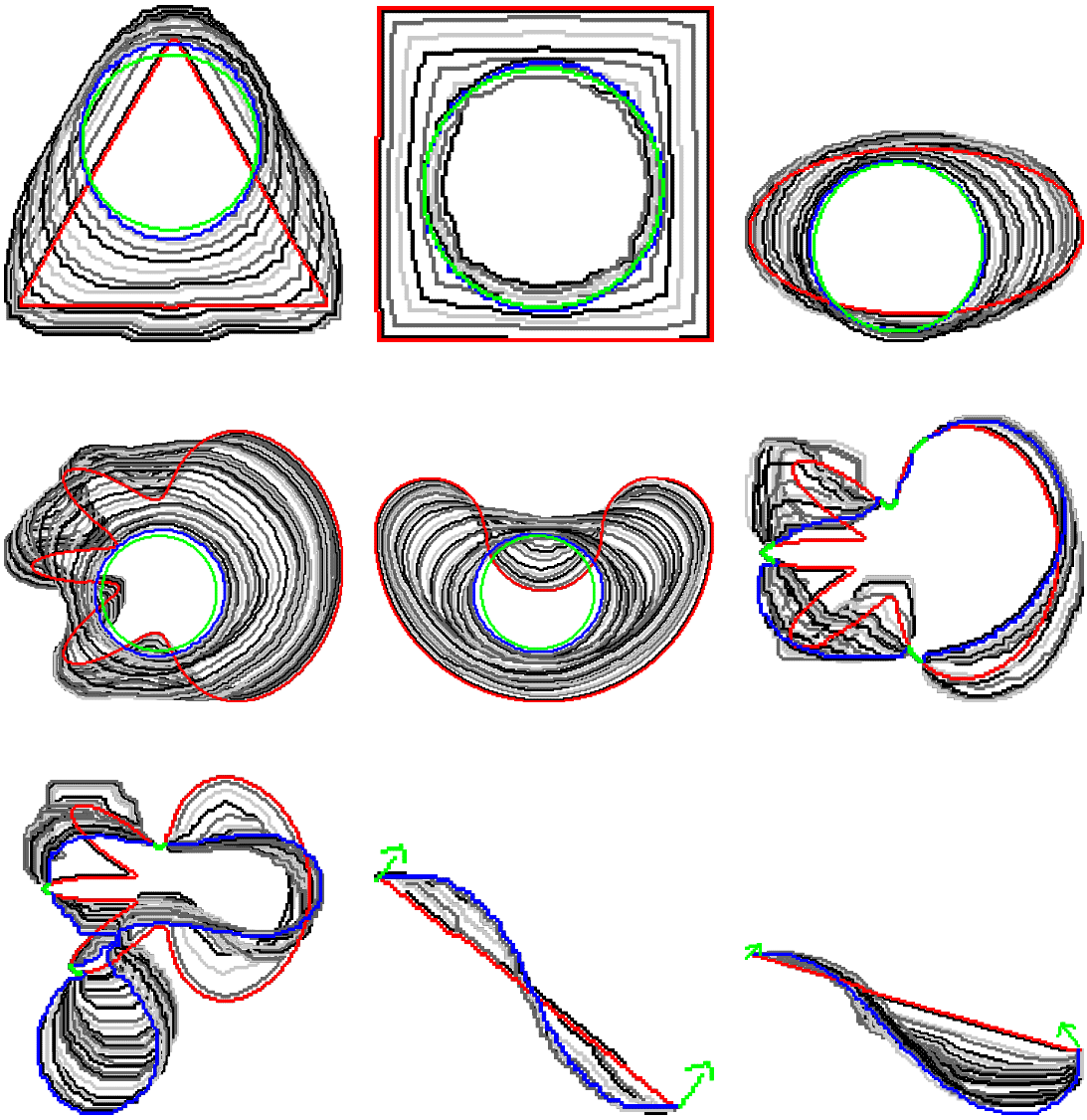


Figure 5.9: **Simplified elastica results for** $(\alpha = 0.01, \beta = 1)$. Experiments of Section 5.2 for the simplified digital elastica.

lie in the interior of those curves. This restriction is encoded in a set of topological constraints $T(\boldsymbol{x}, \boldsymbol{y})$ detailed later. So far we have

$$\min_{\boldsymbol{x}\in\{0,1\}^{|X|},\boldsymbol{y}\in\{0,1\}^{|Y|}} E_{\boldsymbol{\theta}}^{simp}(\boldsymbol{x}, \boldsymbol{y}), \quad \text{subject to } T(\boldsymbol{x}, \boldsymbol{y}). \quad (P0)$$

Additionaly, in real applications involving the minimization of elastica, we have a set of constraints R that plays the role of regularization. For example, we may force some of the pixels in the original shape to be part of the solution; for imaging problems, we may add a data attachment term, and so on. Finally, we can write the general optimization problem as

$$\min_{\boldsymbol{x}\in\{0,1\}^{|X|},\boldsymbol{y}\in\{0,1\}^{|Y|}} E_{\boldsymbol{\theta}}^{simp}(\boldsymbol{x}, \boldsymbol{y}), \quad \text{subject to } T(\boldsymbol{x}, \boldsymbol{y}), R(\boldsymbol{x}) \quad (P1)$$

Formulation $P1$ is a constrained binary non-convex third order problem and likely difficult to be solved optimally. Nonetheless, we can use standard optimization techniques to acquire some intuition on the model.

5.4.3 Topological constraints

The estimation ball should be applied in the digital contour of the shape, which obliges us to impose topological constraints in the model to avoid inconsistent solutions. In order to accomplish that, we set an arbitrary orientation for the faces and another for the edges. We choose counter-clockwise for faces; left-to-right for horizontal edges; and bottom-to-up for vertical edges. The topological constraints are essentially the same as those implemented in the linear programming model of [**schoenemann09linear**] and described in Section 3.4.

We create the vector $\boldsymbol{z} \in \{0, 1\}^{2n}$. We map each linel identified by variable $\boldsymbol{y}_i$ to components $\boldsymbol{z}_{2i}, \boldsymbol{z}_{2i+1}$, one for each possible orientation the linel may assume. Next, we extend the linel incidence matrix defined in Appendix B to hold incidence with respect to oriented edges. The new matrix $\boldsymbol{T} \in \{0, 1\}^{n\times m+2n}$ is defined as

$$0 \leq j < m, \quad \boldsymbol{T}_{i,j} = \begin{cases} 1, & \text{Pixel } j \text{ is positively incident to linel } i \\ -1, & \text{Pixel } j \text{ is negatively incident to linel } i \\ 0, & \text{otherwise,} \end{cases}$$

$$m \leq j < m + 2n, \quad \boldsymbol{T}_{i,j} = \begin{cases} 1, & \text{Edge } j \text{ is positively incident to linel } i \\ -1, & \text{Edge } j \text{ is negatively incident to linel } i \\ 0, & \text{otherwise.} \end{cases}$$

Rewriting formulation (P1)

$$\begin{aligned} & \min \sum_{z_i \in \boldsymbol{z}} \boldsymbol{z}_i \left(\alpha + \frac{9}{r^6}\beta\big(c^2 - 2c\boldsymbol{A}_i^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{A}_i^T\boldsymbol{x}\big) \right) \\ \text{subject to} \quad & \boldsymbol{T} \times \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = 0 \\ & R(\boldsymbol{x}), \\ & \boldsymbol{x} \in \{0,1\}^m, \boldsymbol{z} \in \{0,1\}^{2n}. \end{aligned}$$

We observe that for a linel identified by variable $\boldsymbol{y}_i$, constraints $\boldsymbol{T}$ forces at most one of the variables $\boldsymbol{z}_{2i}, \boldsymbol{z}_{2i+1}$ to be evaluated to one.

5.4.4 Linear relaxation of $P1$

The simplest model we can derive from (P1) consists in the relaxation of the optimization variables, i.e., we impose $\boldsymbol{x} \in [0,1]^m$ and $\boldsymbol{z} \in [0,1]^{2n}$, and we linearize all second and third order terms.

Consider the summation in (P1). An opt-term is an ordered sequence of optimization variables, e.g., the opt-term $\boldsymbol{z}_1\boldsymbol{x}_2\boldsymbol{x}_4$ is encoded as the sequence $(\boldsymbol{z}_1, \boldsymbol{x}_2, \boldsymbol{x}_4)$. Let $\mathcal{T}$ the collection of opt-terms of order two or higher in (P1). To linearize (P1), we associate a variable $\boldsymbol{u_i} \in [0,1]$ for each $\mathcal{T}_i \in \mathcal{T}$ and we enforce $|\mathcal{T}_i| + 1$ new constraints. In other words, we add the following set of linearization constraints.

$$L(\boldsymbol{u}) = \left\{ \left\{ \boldsymbol{u}_i \leq t, \quad \forall t \in \mathcal{T}_i \right\} \cup \left\{ \boldsymbol{u}_i \geq \sum_{t \in \mathcal{T}_i} t - |\mathcal{T}_i| + 1 \right\} \;\middle|\; \forall \mathcal{T}_i \in \mathcal{T} \right\}$$

For example, let $\boldsymbol{u}_i$ be the auxiliar variable associated with the opt-term $\boldsymbol{z}_1\boldsymbol{x}_2\boldsymbol{x}_4$. Then, we are going to add the following constraints

$$\begin{aligned} & \boldsymbol{u}_i \leq \boldsymbol{z}_1 \\ & \boldsymbol{u}_i \leq \boldsymbol{x}_2 \\ & \boldsymbol{u}_i \leq \boldsymbol{x}_4 \\ & \boldsymbol{u}_i \geq \boldsymbol{z}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_4 - 2. \end{aligned}$$

The linearization of (P1) is written as

$$\begin{aligned} & \min \sum_{z_i \in \boldsymbol{z}} \boldsymbol{z}_i \left(\alpha + \frac{9}{r^6} \beta \big(c^2 - 2c \boldsymbol{A}_i^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{A}_i \boldsymbol{A}_i^T \boldsymbol{x} \big) \right) \\ \text{subject to} \quad & \\ & \boldsymbol{T} \times \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = 0 \\ & R(\boldsymbol{x}), \\ & L(\boldsymbol{u}), \\ & \boldsymbol{x} \in [0,1]^m, \boldsymbol{z} \in [0,1]^{2n}, \boldsymbol{u} \in [0,1]^{|\mathcal{T}|} \end{aligned}$$

Finally, to obtain a binary vector we round the partial solution vector $\boldsymbol{x}^\star \in \mathbb{U}^m$. For an instance with m pixels we have about $2m$ linels. After linearization, we can expect to have up to $O(m^3)$ variables, dampening our attempts to solve it globally even for low resolution images. One can also try quadratic formulations by linearizing only the third order terms. Unfortunately, the matrix of quadratic terms is not semi-definite positive, a fundamental condition for efficient optimization of the model.

5.4.5 Unconstrained version of P1

We can use the pixel incidence matrix defined in Appendix B to define an unconstrained version of P1. The pixel incidence vector $\boldsymbol{q} \in \mathbb{Z}^m$ for pixels $\boldsymbol{x} \in \mathbb{B}^m$ is

$$\boldsymbol{q} = \boldsymbol{P}^T \boldsymbol{P} \boldsymbol{x}$$

In order to suppress the sign, we define diagonal matrix $\boldsymbol{Q} \in \mathbb{R}^{m \times m}$ as

$$\boldsymbol{Q} = diag(\boldsymbol{q}) diag(\boldsymbol{q})$$

Let $\boldsymbol{B} \in \mathbb{B}^{m \times m}$ such that column vector $\boldsymbol{B}_j$ represents the pixels in the interior of a disk of radius R centered at pixel i. Finally, we search for solutions of

$$\min_{\boldsymbol{x}} \frac{9}{R^6} \sum_{j}^{m} \left(\frac{\pi R^2}{2} - \frac{1}{2} \mathbf{1}^T \boldsymbol{Q} \boldsymbol{B}_j \right)^2, \tag{5.4}$$

where $\mathbf{1} = (1, 1, \cdots, 1)^T \in \mathbb{R}^m$. Equation (5.4) involves the minimization of a fourth order equation and is therefore hard to optimize.

5.5 Conclusion

We have defined a local combinatorial scheme and provided an algorithm to minimize the digital elastica. We have shown experimental results of the LocalSearch algorithm for shapes with different geometries and we observed that the method converges to the expected global optimum in the free elastica problem, justifying the interest for multigrid convergent estimators. The same model can also be used to solve the constrained elastica problem, but is more likely to stop in a local minimum.

We observed that a global optimization method is important to recover the completion property of curvature. However, the high order of the elastica energy makes it a challenging energy to be globally optimized. We sketched some global optimization models for minimizing the simplified elastica (of lower order) using standard techniques of optimization. The difficulties we pointed out suggest that a practical global optimization model using the II estimator, if it exists, should make use of ingenious techniques.

Nonetheless, we believe it is still possible to partially recover the completion effect based on local approaches, more likely to have practical implementations. In the next chapter, we describe a second local model faster than the one proposed in this chapter.

Chapter 6

A 2-step evolution model driven by digital elastica minimization

In the previous chapter we have presented a local combinatorial model using multigrid convergent estimator that proved to be very successful in optimizing the digital elastica but too slow to be used in practice. We have also attempted to derive a global optimization model, but unfortunately such model is unlikely to be solved in the current state of art of binary optimization techniques. In this chapter we present a second local optimization model that is much faster than Algorithm 1 but with fewer guarantees of optimality.

We describe the binary non-submodular FlipFlow energy and discuss some curious aspects of it in Section 6.1. Optimization strategies are presented in Sections 6.2 and 6.3. Finally, we describe an application of the FlipFlow model to image segmentation in Section 6.4.

6.1 FlipFlow model

In this section we describe the FlipFlow model that aims to evolve an initial digital shape D into another of lower digital elastica value. The FlipFlow algorithm consists in deciding, at each iteration k, which pixels in the *inner boundary* of $D^{(k)}$ are to be removed and which are to be kept.

6.1.1 Definitions

Let D be a digital shape with domain $\Omega \subset \mathbb{Z}^2$. We describe a flow $\left\{D^{(k)} \mid k \geq 0, D^0 = D\right\}$ intended to decrease the digital elastica energy of D.

We assume an ordering in Ω, i.e., there exists a bijective function $\omega : \Omega \to \{1 \cdots |\Omega|\}$. Moreover, let $X_\omega : \Omega \to 2^{\{0,1\}}$ be an operator that transforms digital

sets in its corresponding set of binary variables, i.e.,

$$X_\omega(\Omega) := \left\{ x_{\omega(p)} \in \{0,1\} \mid p \in \Omega \right\}.$$

We will simply write $X(\Omega)$, assuming that exists an underlying ordering function ω.

A $\{0,1\}$ assignment of the variables in $X(\Omega)$ is denoted $x(\Omega)$. We define the sum of a digital set D and an assignment $x(\Omega)$ as

$$D + x(\Omega) = D \cup \left\{ p \mid p \in P, x_{\omega(p)} = 1 \right\}.$$

Next, we define the set of optimization variables. In order to guarantee connectivity and thus avoid the enforcement of the topological constraints discussed in Section 5.4.3, we limit the optimization region to a subset of Ω, namely the inner pixel boundary of $D^{(k)}$. We recall the definition given in Chapter 5.

$$I(D) := \{\, p \mid p \in D, |\mathcal{N}_4(x) \cap D| < 4 \,\},$$

To simplify notation, the inner pixel boundary of $D^{(k)}$ is simply denoted $I^{(k)}$. At each iteration, the set $X^{(k)}$ of optimization variables is defined as

$$X^{(k)} := X(I^{(k)}).$$

In the case we optimize the complement of D, we write $\overline{X}^{(k)}$, i.e., $\overline{X}^{(k)} = X(I(\overline{D}^{(k)}))$.

An assignment of $X^{(k)}$ is simply denoted $x^{(k)}$.

6.1.2 Model and algorithm

We recall the definition of the II digital curvature estimator (see Section 4.2 for details):

$$\hat{\kappa}^2(p) = c_1 \Big(c_2 - |B_r(p) \cap D^{(k)}| \Big)^2, \tag{6.1}$$

where $c_1 = 3/r^6$ and $c_2 = \pi r^2/2$. The following sets are important in the expansion of *Equation* (6.1).

$$\begin{aligned} F^{(k)} &:= D^{(k)} \setminus I^{(k)} && \text{(Invariant foreground)} \\ F_r^{(k)}(p) &:= F^{(k)} \cap B_r(p) \\ I_r^{(k)}(p) &:= I^{(k)} \cap B_r(p) \\ X_r^{(k)}(p) &:= X\big(I_r^{(k)}(p)\big). \end{aligned}$$

Expanding Equation (6.1), we get

$$\begin{aligned}\hat{\kappa}^2(p) &= c_1\Big(c_2 - |F_r^{(k)}(p)| - \sum_{x_j \in X_r^{(k)}(p)} x_j\Big)^2 \\ &= c_1\Big(C + 2\left(|F_r^{(k)}(p)| - c_2\right) \sum_{x_j \in X_r^{(k)}(p)} x_j + \sum_{x_j \in X_r^{(k)}(p)} x_j^2 + \sum_{\substack{x_j, x_l \in X_r^{(k)}(p) \\ j<l}} 2x_j x_l\Big),\end{aligned} \tag{6.2}$$

where $C = c_2^2 - 2c_2 \cdot |F_r^{(k)}(p)| + |F_r^{(k)}(p)|^2$ is a constant. As Equation (6.2) is a term to be optimized, we can ignore constants and multiplication factors. Moreover, as we are in a binary optimization setting, we can further simplify Equation (6.2) by exploiting the binary character of variables and eliminating monomials of second order. We define the following family of energies for given parameters $\boldsymbol{\theta} = (\alpha, \beta) \geq 0$ and $m \geq 0$

$$\begin{aligned}E^{flip}_{(\boldsymbol{\theta},m)}(D^{(k)}, X^{(k)}) = &\sum_{x_j \in X^{(k)}} \alpha s(x_j) + \\ &\sum_{\substack{p \in \\ R_m(D^{(k)})}} 2c_1\beta\Big((1/2 + |F_r^{(k)}(p)| - c_2) \cdot \sum_{\substack{x_j \in \\ X_r^{(k)}(p)}} x_j + \sum_{\substack{j<l, \\ x_j, x_l \in \\ X_r^{(k)}(p)}} x_j x_l\Big),\end{aligned} \tag{6.3}$$

where $s(\cdot)$ denotes the length penalization term, written as

$$s(x_{w(p)}) = \sum_{q \in \mathcal{N}_4(p)} t(q), \quad \text{where } t(q) = \begin{cases} (x_{w(p)} - x_{w(q)})^2, & \text{if } q \in I^{(k)} \\ (x_{w(p)} - 1)^2, & \text{if } q \in F^{(k)} \\ (x_{w(p)} - 0)^2, & \text{otherwise.} \end{cases} \tag{6.4}$$

We recall that R_m refers to the m-ring defined in Section 5.2. Each choice of m generates a different flow, which is generally described in the *FlipFlow* Algorithm 2. To optimize Equation (6.3) we use the QPBOI algorithm [**rother07qpbo**].

6.1.3 Algorithm discussion

We are going to justify Algorithm 2 considering the case in which $m = 0$, i.e., for energy $E^{flip}_{\theta,0}$. As discussed in Section 5.4, the topological constraints are a fundamental part in a global optimization model for the digital elastica but the complexity added to it dampens any hope of optimizing it efficiently. In the proposed FlipFlow model, we exclude topological constraints and we end up with the tractable binary second order Equation (6.3). However, due the lack of contour information, the minimization of Equation (6.3) for $D^{(k)}$ results in undesirable shapes of even higher

input : A digital set D; The ring number m; Length(α), curvature(β) grouped in parameter vector $\boldsymbol{\theta}$; the maximum number of iterations maxIt;

$D^{(0)} \longleftarrow D$;
$k \longleftarrow 1$;
while k $<$ maxIt **do**
  //Shrink mode
  if k *is even* **then**
    $x^{(k-1)} \longleftarrow \underset{X^{(k-1)}}{\arg\min}\, E^{flip}_{(\boldsymbol{\theta},m)}(D^{(k-1)}, 1 - X^{(k-1)})$;
    $D^{(k)} \longleftarrow F^{(k-1)} + x^{(k-1)}$;
  end
  //Expansion mode
  else
    $\overline{x}^{(k-1)} \longleftarrow \underset{\overline{X}^{(k-1)}}{\arg\min}\, E^{flip}_{(\boldsymbol{\theta},m)}(D^{(k-1)}, 1 - \overline{X}^{(k-1)})$;
    $D^{(k)} \longleftarrow \overline{\overline{F}^{(k-1)} + \overline{x}^{(k-1)}}$;
  end
  k $\longleftarrow$ k +1;
end

Algorithm 2: FlipFlow algorithm.

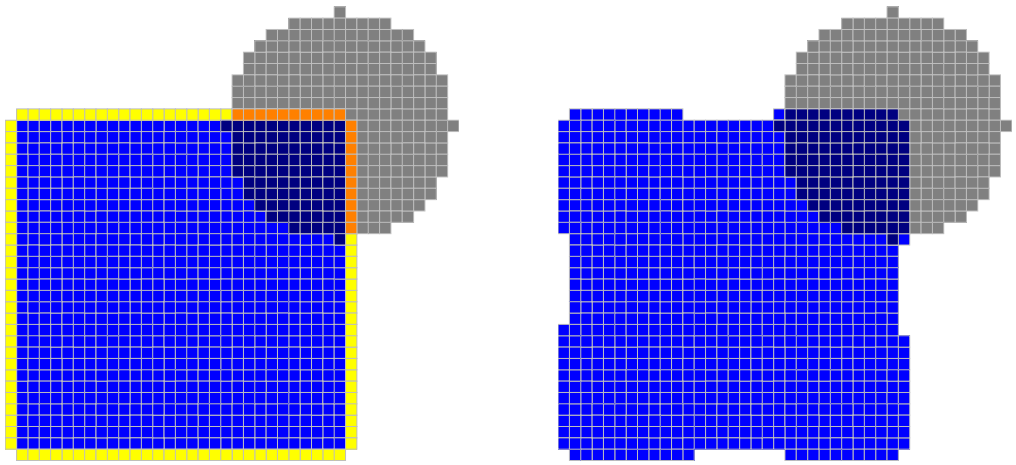


Figure 6.1: **Fixed boundary evaluation.** The proposed model does not take contour information into account, thus the II curvature estimator is evaluated and optimized with respect the initial boundary. In the left, the initial shape and the optimization variables colored in yellow. In the right, the final shape without taking the complementary solution.

shrink mode	$\kappa \gg 0$	$\kappa \geq 0$	$\kappa < 0$
$x^{(k)}$	$x_j = 1$	$x_j \in \{0, 1\}$	$x_j = 0$
$D^{(k+1)} \leftarrow F^{(k)} + x^{(k)}$	eroded	prob. eroded	unchanged
expansion mode	$\overline{\kappa} \gg 0$	$\overline{\kappa} \geq 0$	$\overline{\kappa} < 0$
$\overline{x}^{(k)}$	$\overline{x}_j = 1$	$\overline{x}_j \in \{0, 1\}$	$\overline{x}_j = 0$
$D^{(k+1)} \leftarrow \overline{\overline{F}^{(k)} + \overline{x}^{(k)}}$	dilated	prob. dilated	unchanged

Table 6.1: **Shrink and expansion modes.** Since the curvature is negated when reversing the curve (i.e. $\overline{\kappa} = -\kappa$), this process can only shrink convex parts in shrink mode and expand concave parts in expansion mode.

digital elastica energy values (see Figure 6.1). Interestingly, by using the inverse of the optimal assignement, we can derive a shape of lower digital elastica energy. Therefore, the next shape is given by

$$D^{(k+1)} = F^{(k)} + \arg\min E^{flip}_{(\boldsymbol{\theta},m)}\big(D^{(k)}, 1 - X^{(k)}\big).$$

Recall that the integral invariant estimator approaches curvature by computing the difference between half of the area of a chosen ball and the area of the intersection of this ball with the shape. In particular, regions of positive curvature have fewer pixels in their intersection set than on its complement w.r.t the estimation ball. This implies that variables in such regions are labeled with 1, as the unbalance grows otherwise. We attenuate curvature if we shift the center of the estimation ball towards the interior of the shape, which means to remove the 1-labeled pixels. That is why we take the complement of the optimization solution.

The explanation above covers the treatment of convex parts, but the way to treat concavities is not much different. Indeed, concave regions are convex in the shape complement. The FlipFlow Algorithm 2 is made of two modes: shrink and expansion. The shrink mode handles convexities and its reasoning is explained in the previous paragraph. The expansion mode operates exactly in the same way, but on the image complement, and by doing this we are able to handle concavities. It is called expansion mode because the optimization region, in this case, is the outer pixel boundary of the original shape. Table Table 6.1 sums up these arguments.

In Figure 6.2 we show the results of the FlipFlow algorithm for $(m = 0, \alpha = \{0, 0.5\}, \beta = 1)$. We observe a global evolution towards rounder shapes, but several artifacts are formed along the boundary. An estimation ball of higher radius evolves the shapes faster, but the contours become noisier. Setting $\alpha > 0$ attenuates the

problem for lower radius but the produced shapes does not match with our intuition of what a flow driven by the squared curvature must be like. In the next section we investigate how the energy properties and optimization method used can explain this behavior.

6.2 Optimization method

Let f be a function of n binary variables with unary and pairwise terms, i.e.

$$f(y_1, \cdots, y_n) = \sum_j f_j(y_j) + \sum_{j<k} f_{j,k}(y_j, y_k).$$

As observed in Section 2.2, function f is submodular if and only if the following inequality holds for each pairwise term $f_{j,k}$:

$$f_{j,k}(0,0) + f_{j,k}(1,1) \leq f_{j,k}(0,1) + f_{j,k}(1,0).$$

The energy $E^{flip}_{(\boldsymbol{\theta},m)}$ is non-submodular and optimizing it is a difficult problem, which constrains us to use heuristics and approximation algorithms. The QPBO method [**hammer84**] transforms the original problem in a max-flow/min-cut formulation and yields a full optimal labeling for submodular energies. For non-submodular energies the method is guaranteed to return a partial labeling with the property that the set of labeled variables is part of an optimal solution. That property is called partial optimality.

In practice, QPBO can leave many pixels unlabeled. There exist two extensions to QPBO that alleviate this limitation: QPBOI (improve) and QPBOP (probe) [**rother07qpbo**]. The first is an approximation method that is guaranteed to not increase the energy, but loses the property of partial optimality. The second is an exact method which is reported to label more variables than QPBO.

The percentage of unlabeled pixels by QPBOP for $E^{flip}_{(\boldsymbol{\theta},m)}$ is quite high, but the percentage decreases to zero as we set m to a value closer to r, the estimation ball radius. Therefore, we are more confident in taking the solution for values of m close to r. However, the way it varies across values of m differs from shape to shape, as is illustrated in Figure 6.4. We also noticed that, for $m = r$, all the pixels were labeled, which indicates that $E^{flip}_{(\boldsymbol{\theta},r)}$ is an easy instance of the general non-submodular energy $E^{flip}_{(\boldsymbol{\theta},m)}$, but this remains to be proved. The number of pairwise terms in $E^{flip}_{(\boldsymbol{\theta},r)}$ is roughly half of those in $E^{flip}_{(\boldsymbol{\theta},1)}$ (see Figure 6.3), which also explains the higher number of labeled variables.

We have used QPBOI to solve $E^{flip}_{(\boldsymbol{\theta},m)}$. Naturally, in the case where all pixels are labeled by QPBOP, QPBOI returns the same labeling as QPBOP. In the next section we show that by evaluating the estimation ball at outer rings, we eliminate

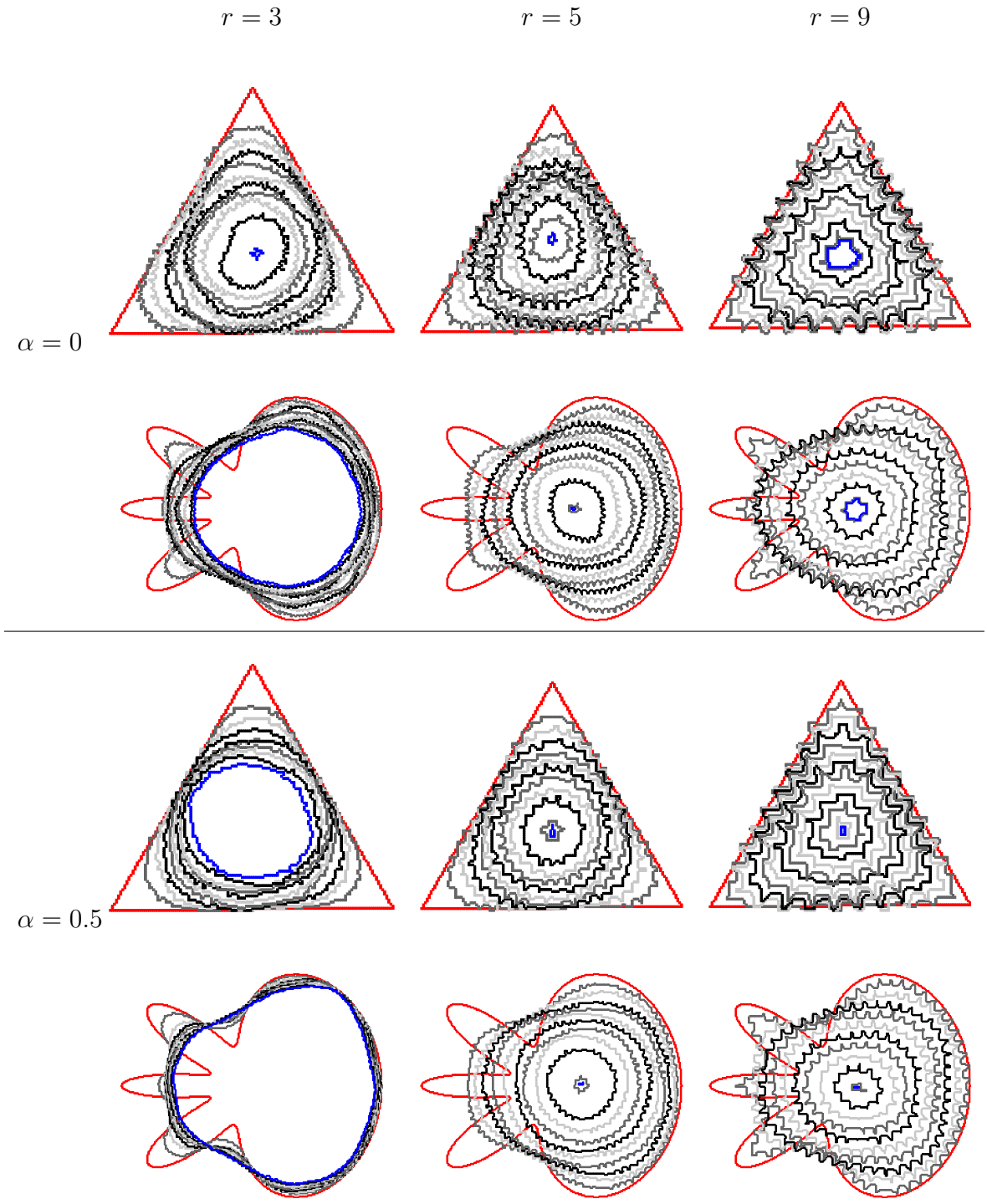


Figure 6.2: **FlipFlow results for $m = 0, \beta = 1$.** The algorithm is very sensitive to the little variations of the estimator, which are particularly important in regions of low squared curvature. Artifacts are somewhat reduced with a length penalization but increases if we use a higher ball radius. For better visualization, curves are displayed every 1/10 of the number of iterations.

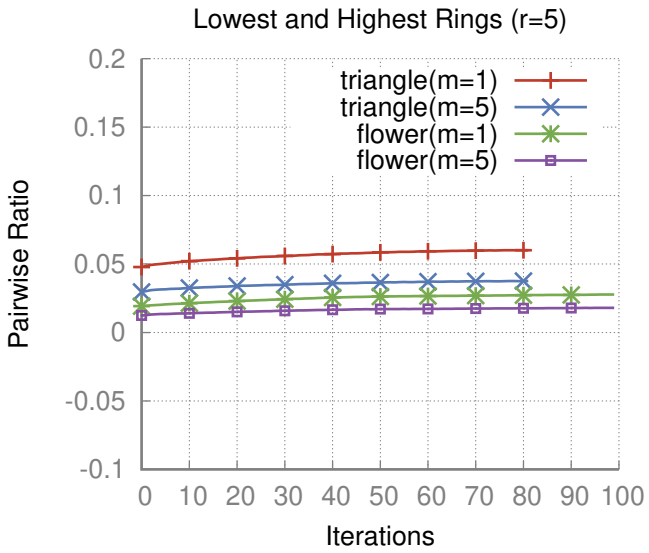


Figure 6.3: **Pairwise terms ratio.** We plot the ratio of pairwise terms among all $\binom{|X^{(k)}|}{2}$ combinations. The highest ring has roughly half the number of pairwise terms as the lowest ring.

the artifacts and we produce smoother flows while preserving a qualitative measure of curvature.

We postpone the comparison of the proposed methods in this thesis to Chapter 9. At this point, we observe that the FlipFlow performs up to 10x faster than the LocalSearch algorithm.

6.3 Evaluation across m-rings

The QPBOP method leaves many pixels unlabeled for $m = 0$, but in some occasions, the ratio of unlabeled pixels decreases as we take values of m closer to r. In this section, we argue that by evaluating the estimation ball along outer rings we obtain smoother evolutions by focusing the optimization process only on regions of high squared curvature value.

In Figures 6.5 and 6.6 we evaluate several flows for different energies $E^{flip}_{(\boldsymbol{\theta},m)}$. As expected, the number of artifacts decrease as the value of m increases, while the process still tends to shrink the shape to a single point, resembling the curve shortening flow discussed in Section 3.1.

We confirm the stability of the model by looking at the plots of the digital elastica energy values for the produced shapes. Moreover, the produced flow has

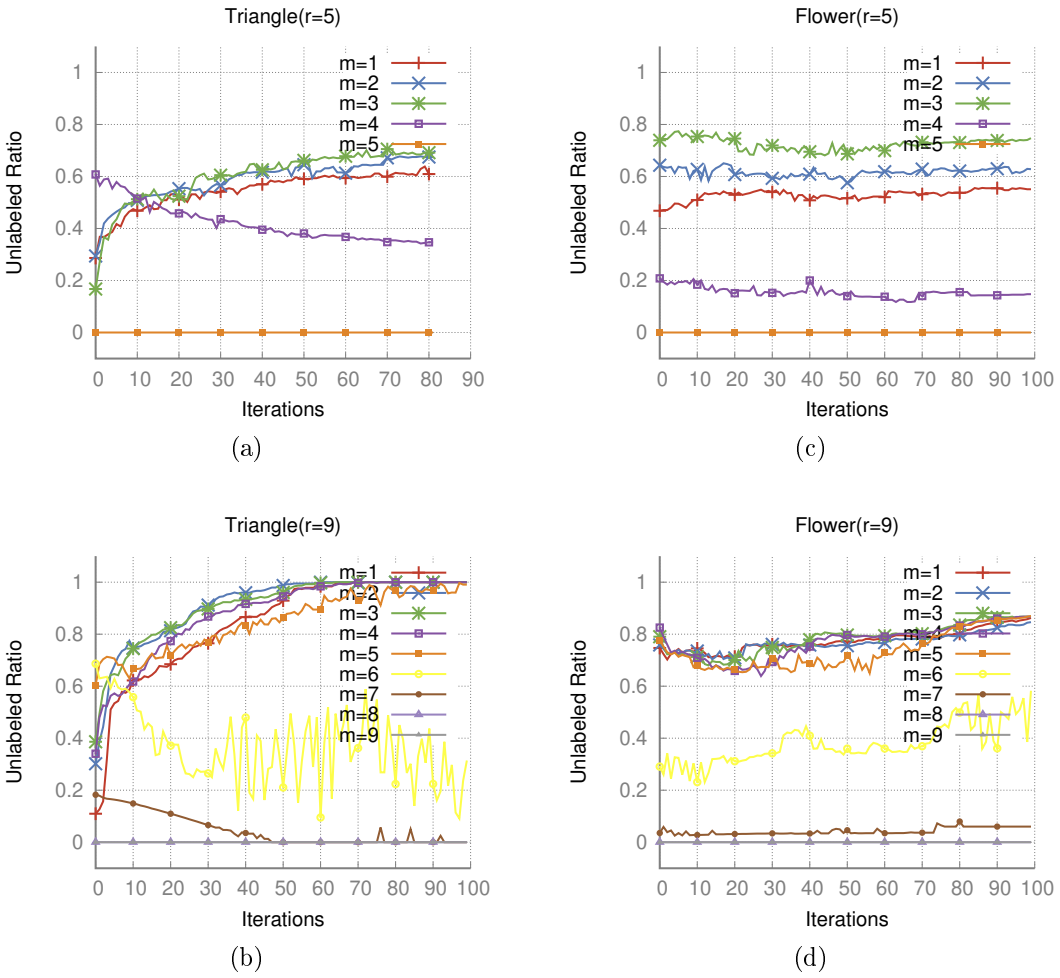


Figure 6.4: **Unlabeled variables ratio across m-rings.** For each plot, we first produce the sequence of shapes $\{D^{(k)}\}$ executing FlipFlow with $m = r$. Then, for each shape in $\{D^{(k)}\}$, we execute one iteration of FlipFlow for different values of m and we count the unlabeled pixels. The number of unlabeled pixels by QPBOP remains high for small values of m, and goes to zero when $m = r$.

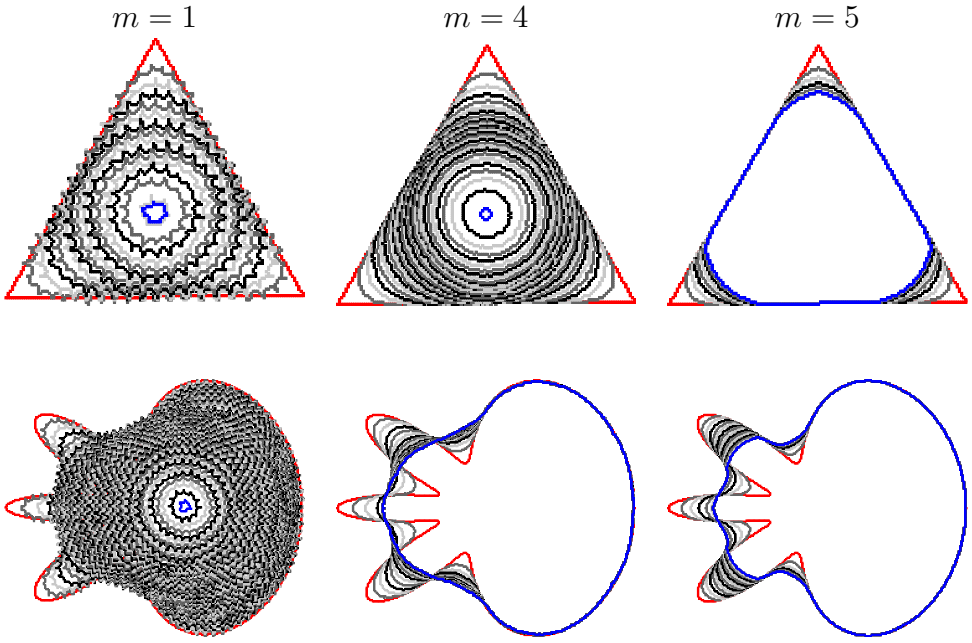


Figure 6.5: **FlipFlow results for r = 5.** By positioning the estimation ball on outer rings, we minimize the apparition of sharp artifacts.

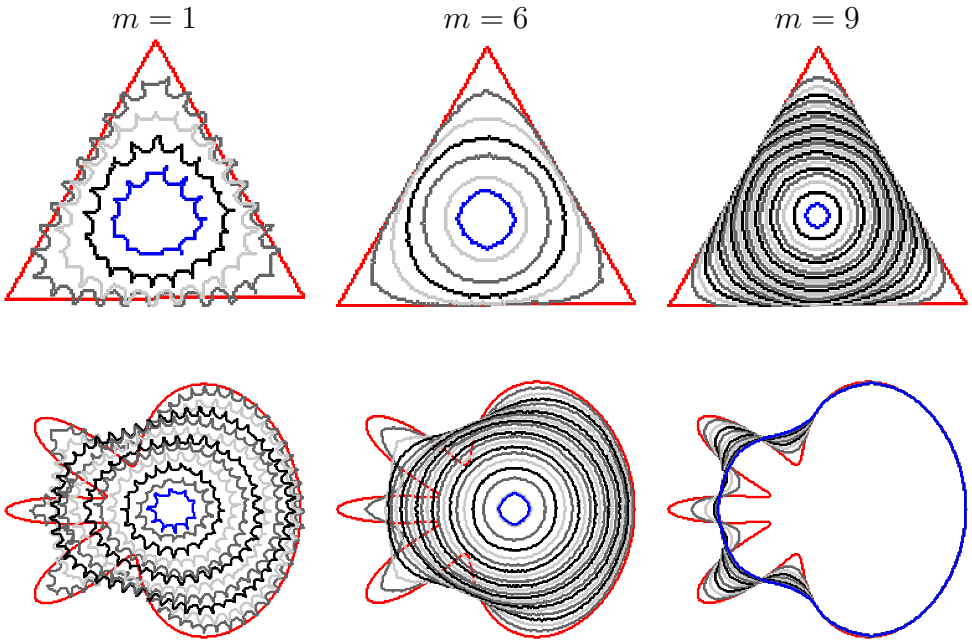


Figure 6.6: **FlipFlow results for r = 9.** By positioning the estimation ball on outer rings, we minimize the apparition of sharp artifacts.

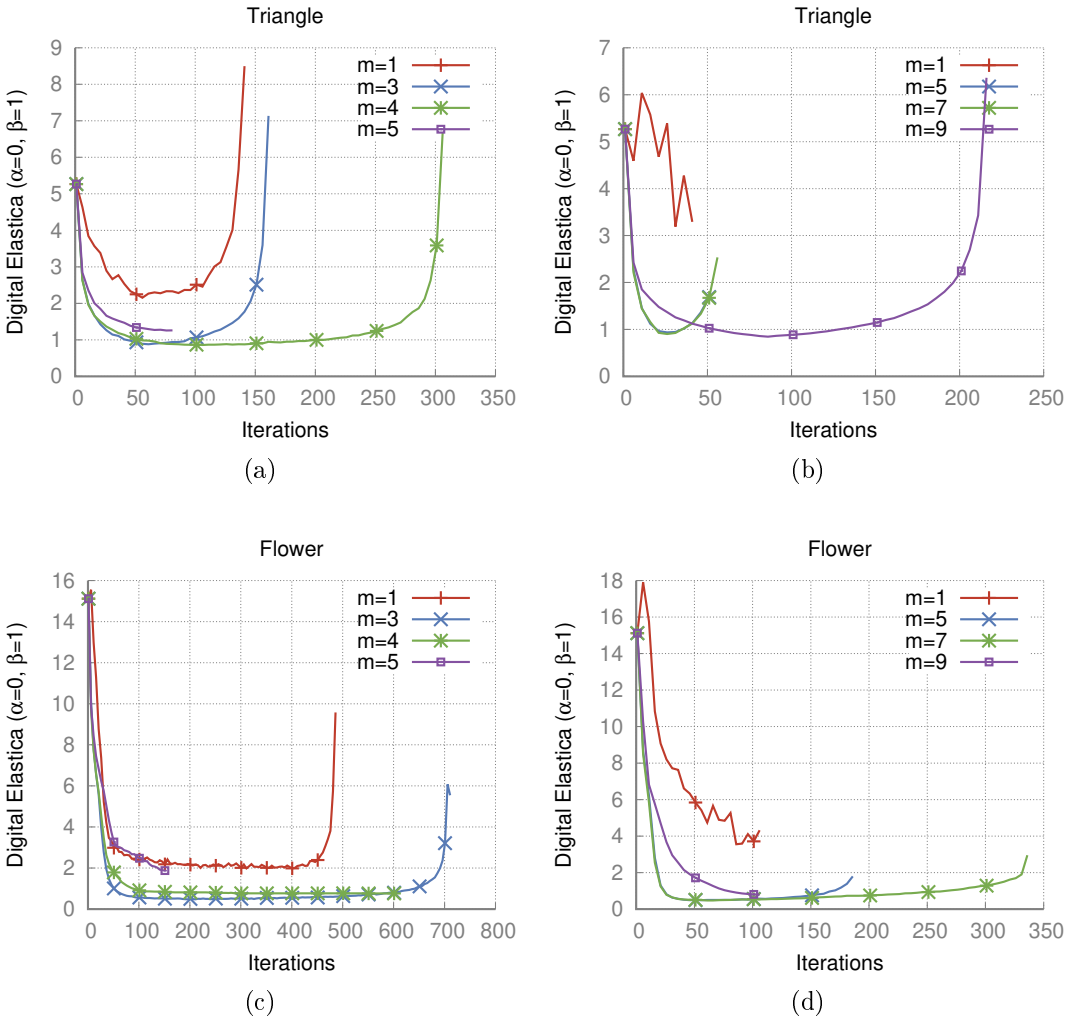
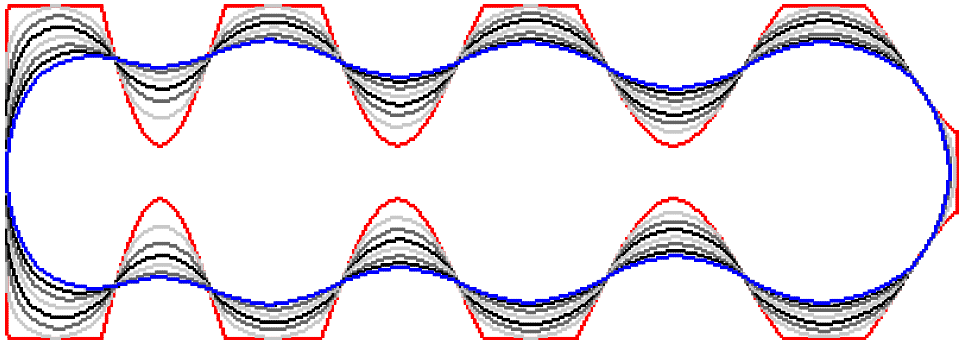
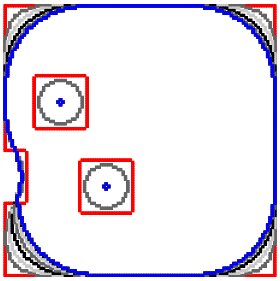


Figure 6.7: **Digital elastica evaluation.** We present the results of the FlipFlow model for different values of m. In the left column, we use FlipFlow radius 5 and in the right column we use FlipFlow radius 9.



(a)



(b)

Figure 6.8: **Evolution speed and topology changes.** The flow evolves regions of high curvature faster, as illustrated in figure (a). Figure (b) illustrates the property of the FlipFlow algorithm to handle changes in topology.

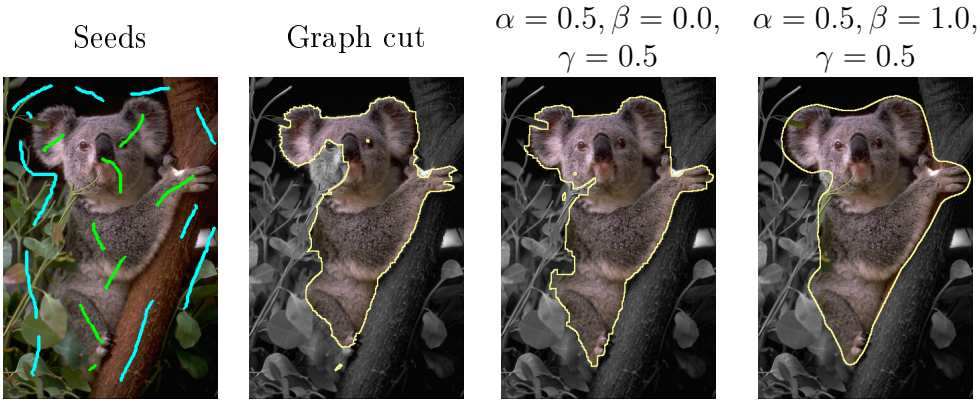


Figure 6.9: **FlipFlow results for segmentation.**Given foreground (green) and background (blue) seeds at picture (a); Graph cut produces picture (b) which is used as input of the Contour Correction algorithm; in pictures (c) and (d) we display the output of Contour Correction algorithm with and without squared curvature regularization.

no difficulties in handling changes on topology, and it presents different speeds for regions with low and high curvature values, as illustrated in Figure 6.8.

The computation of estimation balls at outer rings raises the question about its validity as a curvature estimator. In fact, one can estimate the curvature using outer balls (see Appendix A), but we were not able to prove its multigrid convergence. However, this relation suggests that curvature information is at least qualitatively present in the outer balls computation, and this computation is preferred, as it is easier to optimize accordingly to the experiments illustrated in Figures 6.4 to 6.7.

6.4 Data term and image segmentation

We present an application of the FlipFlow algorithm to supervised image segmentation. The FlipFlow acts as a contour correction method. Here we use a data fidelity term in order to characterize the object of interest. Given foreground and background seeds selected by the user, we derive mixed Gaussian distributions of color intensities H_f,H_b, and we define the data fidelity term as the cross-entropy, i.e.

$$g(x_{w(p)}) = -x_{w(p)} \log H_f(p) - (1 - x_{w(p)}) \log H_b(p) \tag{6.5}$$

We use the FlipFlow algorithm to regularize an initial contour output by some segmentation algorithm or delineated by the user. In this application, the data term of the FlipFlow is set to the data fidelity term Equation (6.5).

The algorithm can be initialized by a collection of compact sets, or with the result of a third-party segmentation algorithm, as Graph cut [**boykov01graphcut**]. We include an additional parameter d that dilates the initial sets using a square of side one before executing the flow.

An illustration of the application of the FlipFlow model in image segmentation is presented in Figure 6.9. We present a more exhaustive list of experiments and comparisons with other methods in Chapter 9.

6.5 Conclusion

We proposed the FlipFlow algorithm, an evolution process based on the minimization of a second-order non-submodular energy that produces shapes of decreasing digital elastica until a certain inflexion point. We point out the submodularization effect of evaluating the FlipFlow model at outer m-rings and we observed that the model evolves the shapes in a similar fashion to the curve-shortening flow for an appropriate choice of parameters.

input : An image I; seeds mask M; the estimation ball radius r; parameter vector $\boldsymbol{\theta} = (\alpha, \beta)$; data term weight (γ) ; initial dilation d; stop condition value tolerance; the maximum number of iterations maxIt;

$D \longleftarrow$ Graph cut(I, M);
$D^{(0)} \longleftarrow$ dilate(D,d);
delta $\longleftarrow +\infty$;
$k \longleftarrow 0$;
while k $<$ maxIt **and** delta $>$ tolerance **do**
  //Shrink mode
  if k *is even* **then**
    $x^{(k-1)} \longleftarrow \underset{X^{(k-1)}}{\arg\min}\, E^{flip}_{(\boldsymbol{\theta},m)}(D^{(k-1)}, 1 - X^{(k-1)}) + \gamma g(X^{(k-1)})$;
    $D^{(k)} \longleftarrow F^{(k-1)} + x^{(k-1)}$;
  end
  //Expansion mode
  else
    $\overline{x}^{(k-1)} \longleftarrow \underset{\overline{X}^{(k-1)}}{\arg\min}\, E^{flip}_{(\boldsymbol{\theta},m)}(D^{(k-1)}, 1 - \overline{X}^{(k-1)}) + \gamma g(\overline{X}^{(k-1)})$;
    $D^{(k)} \longleftarrow \overline{\overline{F}^{(k-1)} + \overline{x}^{(k-1)}}$;
  end
  delta $\longleftarrow |D^{(k)} \setminus D^{(k+1)}|$;
  k $\longleftarrow$ k $+1$;
end

Algorithm 3: FlipFlow algorithm for segmentation.

The FlipFlow algorithm operates in two distinct modes: the shrink and the expansion modes. They are responsible for the treatment of convex and concave regions, respectively. The model handles changes in topology and can be extended to include extra terms as data regularization term. In particular, we described an application of the FlipFlow model to image segmentation.

It is remarkable that curvature regularization is achieved by evaluating the estimation disks at outer rings. Another surprising fact is that we are using a quite non-standard operation of taking the solution complement. We have developed an heuristic reasoning to explain the latter and we argue in Appendix A that curvature can still be measured evaluating disks at outer rings. Nonetheless, we may have taken an unnecessary tortuous path and missed the fundamental concept behind it. In the next chapter we identify the key concept behind the FlipFlow model to define a variant of this algorithm which is easier to implement.

Chapter 7

A single step evolution model driven by digital elastica minimization

In this chapter we introduce the balance coefficient, a value that indicates how far the intersection of some ball and a digital shape is from half of the ball area. This value motivates the definition of a new family of energies to regularize the squared curvature: the m-BalanceFlow. We are going to show that the BalanceFlow is closely related to the FlipFlow energy, but the former has an easier interpretation and leads to a simpler algorithm to implement.

7.1 BalanceFlow model

In Section 6.1.3, we have argued that inverting the solution was necessary to reduce the squared curvature estimation in the next shape of the flow. In this section, we investigate further the reasons involved in this unusual inversion step.

7.1.1 Definitions

In the FlipFlow model no contour information is given, and the strategy there were to label the pixels such that the next shape has a boundary in which the estimation balls were more balanced than in the previous shape, i.e., the difference $|B_r(p) \cap D| - |B_r(p) \cap \overline{D}|$ is closer to zero for every pixel $p \in \partial D$. We formalize this idea by defining the balance coefficient.

Definition 1(Balance coefficent): Given digital shape $D \in \Omega$, positive number r and point $p \in \Omega$, the *balance coefficient* of D at p is defined as

$$u_r(D, p) = \left(\frac{\pi r^2}{2} - |B_r(p) \cap D| \right)^2 .$$

The balance coefficient definition is very similar to the II squared curvature estimator. Nonetheless, we have decided to make a new definition since we do not have the scaling factor $\frac{9}{R^6}$ and that the balance coefficient can be computed everywhere and not only in the digital boundary of the shape. Let $F = B_r(p) \cap D$ and $G = B_r(p) \setminus F$. The balance coefficient is symmetric with respect to F and G.

$$u_r(D,p) = \left(\frac{\pi r^2}{2} - |F|\right)^2 = \left(-\frac{\pi r^2}{2} + |G|\right)^2 = \left(\frac{\pi r^2}{2} - |G|\right)^2.$$

The balance coefficient is used to estimate the effect of moving the estimation ball center towards the interior or the exterior of the shape. For example, consider Figure 7.1 in which we plot the balance coefficients of points along the diagonal of a square shape.

The balance coefficient grows if the estimation ball is moved towards the exterior and decreases if the estimation ball is moved towards the interior of the shape. That is an indication that we should remove point p from D to decrease the squared curvature of the shape at point $p \in \partial D$. The same point p may be touched by several balls, each of them contributing with some value for the ultimate decision of keeping or removing point p of D.

We are going to consider the inner and outer pixel boundary simultaneously as the optimization variables. We recall that $D^{(k)}$ corresponds to the digital shape D after the k-th iteration of the flow and that $d_{D^{(k)}}$ is its signed distance transform. We list the sets needed to define the new family of energies:

$$\begin{aligned}
O^{(k)} &:= \{p \in \Omega \mid -1 \leq d_{D^{(k)}}(p) \leq 1\} \\
X^{(k)} &:= X(O^{(k)}) \\
F^{(k)} &:= D^{(k)} \setminus O^{(k)} \\
F_r^{(k)}(p) &:= F^{(k)} \cap B_r(p) \\
O_r^{(k)}(p) &:= O^{(k)} \cap B_r(p) \\
X_r^{(k)}(p) &:= X\big(O_r^{(k)}(p)\big).
\end{aligned}$$

We recall that an assignment of variables $X(\Omega)$ is denoted $x(\Omega)$. In case the shape is described in terms of a set of variables $X(\Omega)$, i.e., $D = F \cup x(\Omega)$, the balance coefficient is written as

$$u_r(F,X,p) = \Big(\frac{\pi r^2}{2} - |F_r(p)| - \sum_{x_j \in X_r(p)} x_j\Big)^2.$$

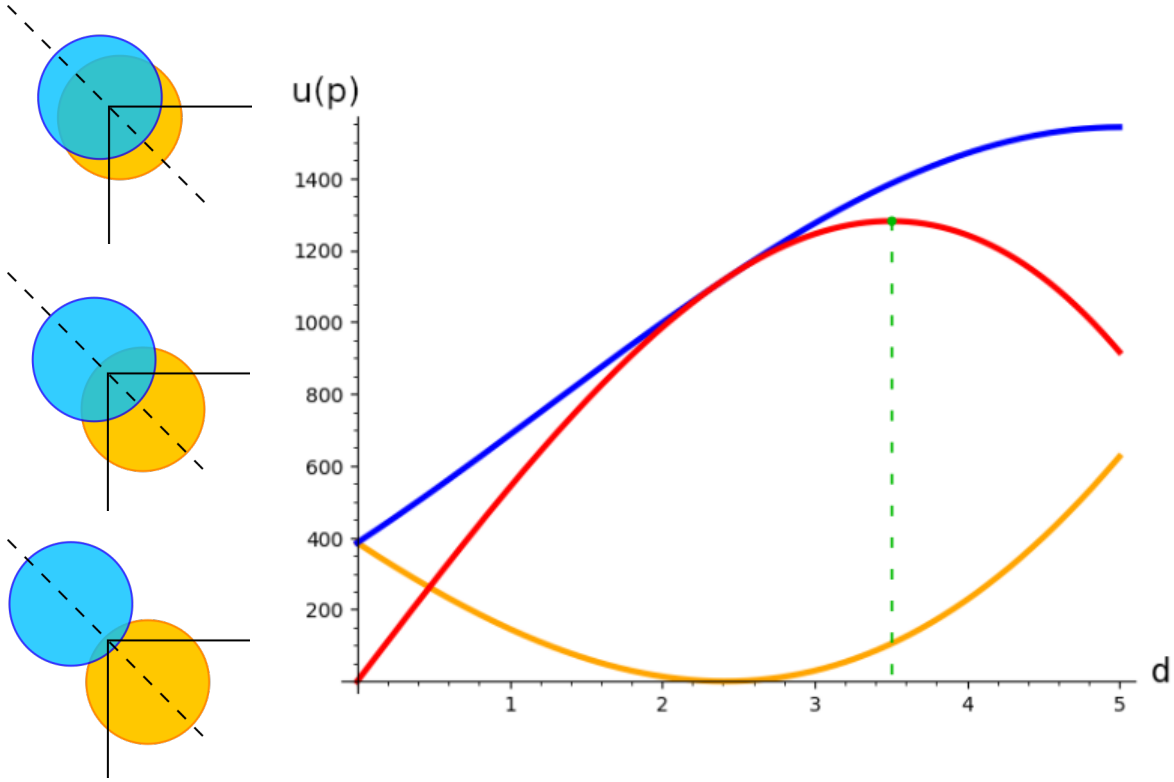


Figure 7.1: **The balance coefficient ($r = 5$) of equidistant points from the corner of a square shape contour.** The balance coefficient for the outer (inner) disk is colored in blue (orange). The red curve indicates the difference between the blue and the orange curve. The inflexion point in the red curve indicates the point p in which a ball centered at p has balance coefficient equal to zero. The green point marks the point p in which the difference between the balance coefficients is the greatest.

7.1.2 Algorithm

Let $p_i, p_o \in \mathcal{R}_m(D)$ the inner and outer ball centers in the m-ring of D, respectively. We assume $k = 0$ if omitted, i.e., $D = D^{(0)}$. We define the term $T_m^{bal}(D)$ as

$$\begin{aligned} T_m^{bal}(D, X) &= \sum_{p_i, p_o \in \mathcal{R}_m(D)} u_r(F, 1 - X, p_o) - u_r(F, X, p_i) \\ &= \sum_{p_i, p_o \in \mathcal{R}_m(D)} \left(\frac{\pi R^2}{2} - \Big(|F_o| + \sum_{x_j \in X_o} 1 - x_j \Big) \right)^2 - \\ &\quad \sum_{p_i, p_o \in \mathcal{R}_m(D)} \left(\frac{\pi R^2}{2} - \Big(|F_i| + \sum_{x_j \in X_i} x_j \Big) \right)^2, \end{aligned} \tag{7.1}$$

where $X_o = X_r(p_o), X_i = X_r(p_i), F_o = F_r(p_o), F_i = F_r(p_i)$.

Equation (7.1) is conceived to implement the reasoning described in Section 6.1.3. In convex regions, the outer ball has a higher balance coefficient than the inner ball, and minimization of Equation (7.1) tends to set $x_j = 0$, i.e., the shape shrinks. On the other hand, if we have a concave region, the inner ball has a higher balance coefficient, and minimization of Equation (7.1) tends to set $x_j = 1$ (the shape dilates).

The m-balance energy family is defined as

$$E_{(\boldsymbol{\theta}, m)}^{bal}(D^{(k)}, X^{(k)}) = \sum_{x_j \in X^{(k)}} \alpha s(x_j) + \beta T_m^{bal}(D^{(k)}, X^{(k)}). \tag{7.2}$$

We follow the same notation as in Chapter 6 to denote the data term $g(D, x)$ and the length penalization term $s(x)$ that are defined as in Equations (6.4) and (6.5), respectively. The BalanceFlow algorithm is summarized in Algorithm 4 and an illustration is shown in Figure 7.2.

7.2 Relation with FlipFlow

The BalanceFlow returns similar solutions to the FlipFlow algorithm, as the experiments in Chapter 9 illustrates. Indeed, they are closely related. We recall the curvature regularization term of the FlipFlow Equation (6.3) for some digital shape D before proving the proposition that establishes this link.

$$T_m^{flip}(D, X) = \sum_{p_i, p_o \in R_m(D)} \Big(\frac{\pi R^2}{2} - |F_o| - \sum_{x_j \in X_o} x_j \Big)^2 + \Big(\frac{\pi R^2}{2} - |F_i| - \sum_{x_j \in X_i} x_j \Big)^2 \tag{7.3}$$

input : A digital set D; The ring number m; parameter vector $\boldsymbol{\theta} = \alpha, \beta)$; data term coefficient γ; the maximum number of iterations maxIt;

$D^{(0)} \longleftarrow D$;
$k \longleftarrow 1$;
while k $<$ maxIt **do**
  $x^{(k-1)} \longleftarrow \arg\min_{X^{(k-1)}} E^{bal}_{(\boldsymbol{\theta},m)}(D^{(k-1)}, X^{(k-1)}) + \gamma g(X^{(k-1)})$;
  $D^{(k)} \longleftarrow F^{(k-1)} + x^{(k-1)}$;
  k $\longleftarrow$ k +1;
end

Algorithm 4: BalanceFlow algorithm.

Figure 7.2: **Segmentation using the BalanceFlow model.** Given foreground (green) and background (gray) seeds at picture (a); Graph cut produces picture (b) which is used as input of the BalanceFlow algorithm; in pictures (c) and (d) we display the output of BalanceFlow algorithm with and without squared curvature regularization.

Proposition 1(FlipFlow and BalanceFlow relation): The curvature terms of FlipFlow and BalanceFlow are related by the equality

$$T_m^{flip}(D, 1-X) = T_m^{bal}(D) + P_1(X_i) + c,$$

where $P_1(X_i) = (\sum_{X_i} 1 - x_j)^2 + (\sum_{X_i} x_j)^2$ and c is a constant.

Proof:

We replace x_j by $(1-x_j)$ in Equation (7.3), which corresponds to the complement of the solution in the FlipFlow model. To simplify notation, we are going to omit the radius r and replace points p_i, p_o by underscored i, o, i.e., $F_i := F_r(p_i)$. We rewrite Equation (7.3) as

$$T_m^{flip}(D, 1-X) = \sum_{p_i, p_o \in R_m(D)} \Big(\frac{\pi R^2}{2} - |F_o| - \sum_{x_j \in X_o} 1 - x_j\Big)^2 + \Big(\frac{\pi R^2}{2} - |F_i| - \sum_{x_j \in X_i} 1 - x_j\Big)^2 \tag{7.4}$$

Next, let $A_i = \pi R^2/2 - |F_i|$. We rewrite the second term of Equation (7.4) as

$$\begin{aligned}
\Big(A_i - \sum_{x_j \in X_i} (1 - x_j)\Big)^2 &= \Big(A_i - |X_i| + \sum_{x_j \in X_i} x_j\Big)^2 \\
&= (A_i - |X_i|)^2 + 2(A_i - |X_i|) \sum_{x_j \in X_i} x_j + \Big(\sum_{x_j \in X_i} x_j\Big)^2 \\
&= A_i^2 - 2A_i|X_i| + |X_i|^2 + 2(A_i - |X_i|) \sum_{x_j \in X_i} x_j + \Big(\sum_{x_j \in X_i} x_j\Big)^2 \\
&= A_i^2 + 2A_i \sum_{x_j \in X_i} x_j + \Big(\sum_{x_j \in X_i} x_j\Big)^2 - 2A_i|X_i| + |X_i|^2 - 2|X_i| \sum_{x_j \in X_i} x_j \\
&= 2A_i^2 - \Big(A_i - \sum_{x_j \in X_i} x_j\Big)^2 + 2\Big(\sum_{x_j \in X_i} x_j\Big)^2 - 2A_i|X_i| + |X_i|^2 \\
&\quad - 2|X_i| \sum_{x_j \in X_i} x_j
\end{aligned}$$

We group the constants into the constant term $c = 2A_i^2 - 2A_i|X_i|$ to obtain

$$\begin{aligned}\Big(A_i - \sum_{x_j \in X_i} (1 - x_j)\Big)^2 &= -\Big(A_i - \sum x_j\Big)^2 + \Big(|X_i| - \sum_{x_j \in X_i} x_j\Big)^2 + \Big(\sum_{x_j \in X_i} x_j\Big)^2 + c \\ &= -\Big(A_i - \sum_{x_j \in X_i} x_j\Big)^2 + P_1(X_i) + c \\ &= -\Big(\frac{\pi R^2}{2} - (|F_i| + \sum_{x_j \in X_i} x_j)\Big)^2 + P_1(X_i) + c\end{aligned}$$

Finnaly, we replace Equation (7.5) in Equation (7.4) to obtain

$$\begin{aligned}T_m^{flip}(D, 1 - X) &= \left(\frac{\pi R^2}{2} - \Big(\, |F_o| + \sum_{x_j \in X_o} 1 - x_j \,\Big)\right)^2 \\ &\quad - \left(\frac{\pi R^2}{2} - \Big(\, |F_i| + \sum_{x_j \in X_i} x_j \,\Big)\right)^2 + P_1(X_i) + c \\ &= T_m^{bal}(D) + P_1(X_i) + c. \qquad (7.5)\end{aligned}$$

■

Proposition 1 tell us that, apart from a constant, the two energies differ in the penalty term $P_1(X_i)$. Locally, such term favors solutions in which half of the variables touched by the inner ball are labeled one. Indeed, the FlipFlow and BalanceFlow behave similarly. In the next chapter we make extensive comparisons between them.

7.3 Conclusion

We proposed the BalanceFlow model for digital elastica minimization, which was inspired by the definition of the balance coefficient, and we proved a link between the BalanceFlow and FlipFlow algorithm. The models present similar results, but the BalanceFlow has an easier implementation.

Looking at Figure 7.1, it would be interesting to find the point in which the inner or outer ball reaches the zero balance. In Figure 7.3 each color represents the balance coefficient u_{12} of the closed shape with contour colored in white. We highlighted in magenta the pixels p for which $u_{12}(D, p) \leq 169$. Figure 7.3 suggests that an evolution based on the zero level set of the balance coefficient may decrease the digital Elastica energy. In the next chapter we describe a graph-cut model in which the cost function is given by a function of the balance coefficients.

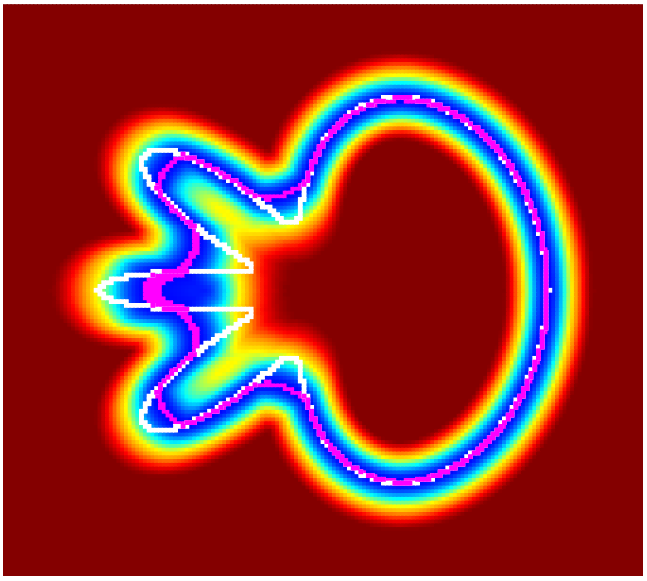


Figure 7.3: **Balance coefficient map**. We display the balance coefficients of radius 12 for the flower shape (contour in white) in every point of the given domain. The pixels in magenta have balance coefficient smaller than 169, and indicates the region in which points with zero balance coefficient are located.

Chapter 8

Digital elastica minimization via graph cuts

In the previous chapter we have defined the concept of balance coefficient that motivates us to introduce the BalanceFlow model. In fact, the balance coefficient is also present in the FlipFlow energy and it seems that its computation is in the core of the evolution processes described so far. We confirm this hypothesis once more in this chapter. We present a graph cut model that converges to the optimum digital shape for the free digital elastica problem. Moreover, the model is easily adapted to image segmentation tasks.

8.1 GraphFlow model

The model presented in this chapter is highly influenced by the graph cut model described in Section 2.3 [**boykov01graphcut**]. We recall that this model constructs a cost function on the edges of the image grid graph such that the minimum cut of the graph minimizes the segmentation energy. The model is very attractive as minimum cuts are quickly computed for sparse graphs.

Let $\boldsymbol{I} \in \mathbb{F}^{m \times n}$ a discrete image and its capacitated grid graph $\mathcal{G}_{\boldsymbol{I}+}(\mathcal{V}^+, \mathcal{E}^+, c)$ as defined in Section 2.3. Given a cut $\mathcal{E}'$ of $\mathcal{G}_{\boldsymbol{I}+}$ that partitions the graph in disjoint sets S and T, we recall that the energy minimized by the graph cut model is written as

$$E_{\boldsymbol{\gamma}}^{gcut}(\mathcal{G}_{\boldsymbol{I}+}, \mathcal{E}') = \gamma_r \left(\sum_{v_p \in S} \psi_1(1) + \sum_{v_p \in T} \psi_1(0) \right) + \gamma_b \sum_{(v_p, v_q) \in \mathcal{E}'} \psi_2(0,1),$$

where $\gamma_r \geq 0$ and $\gamma_b \geq 0$ are parameters controlling the influence of the data and space coherence terms, respectively. Given a neighborhood cardinality k (e.g. 8),

data and space coherence terms are defined as

$$\psi_1(x_p) = \begin{cases} -\ln H_{bg}\big(I(p)\big), & \text{if } x_p = 0 \\ -\ln H_{fg}\big(I(p)\big), & \text{if } x_p = 1, \end{cases}$$

$$\psi_2(x_p, x_q) = \begin{cases} \exp\left(-\dfrac{1}{d_E(p,q)}\dfrac{(I(p)-I(q))^2}{2\sigma^2}\right), & q \in \mathcal{N}_k(p) \\ 0, & \text{otherwise.} \end{cases}$$

Let $D^{(0)}$ the digital set induced from the foreground component returned by the standard graph cut algorithm. The GraphFlow model produces a sequence of digital shapes $D^{(k)}$ and is composed of two steps

Candidate selection: We associate to $D^{(k)}$ a set of neighbor shapes $\mathcal{P}(D^{(k)})$. For each $D' \in \mathcal{P}(D^{(k)})$ we construct its *candidate graph* $\mathcal{G}_{D'}$ and we compute its minimum cut Q according to energy

$$E_{\boldsymbol{\gamma}}^{gflow}(\mathcal{G}_{D'}, Q, D^{(k)}) = \sum_{(v_p,v_q)\in Q} \big(u(D^{(k)}, p) + u(D^{(k)}, q)\big) + E_{\boldsymbol{\gamma}}^{gcut}(\mathcal{G}_{D'}, Q). \tag{8.1}$$

Validation: Each minimum cut Q computed in the previous step induces a solution candidate D_Q. We group minimum cuts and solution candidates in the *solution candidates set* $sol(D^{(k)})$. We choose among the solution candidates the one that minimizes

$$E_{(\boldsymbol{\theta},\boldsymbol{\gamma})}^{val}(\mathcal{G}_{\boldsymbol{I}}, Q, D_Q) = \hat{E}_{\boldsymbol{\theta}}(D_Q) + E_{\boldsymbol{\gamma}}^{gcut}(\mathcal{G}_{\boldsymbol{I}}, Q). \tag{8.2}$$

We recall that $\hat{E}_\theta$ stands for the digital elastica energy Equation (5.1). We observe that, in the validation step, we consider the image grid graph and not the candidate graph. The validation step plays an important role in the emulation of the completion property associated with the squared curvature term.

The GraphFlow model is suitable for the free, constrained elastica and image segmentation problems and can be seen as a simple extension of the graph cut segmentation model with a regularization term based on squared curvature.

8.1.1 Candidate graphs and solution candidates set

The set of candidate graphs of D are derived from some neighborhood of shapes with respect to D. We define the a-probe set as an example of such neighborhood.

Definition 1(a-probe set): Let $D \subset \Omega \subset \mathbb{Z}^2$ a digital set and a a natural number. The a-probe set of D is defined as

$$\mathcal{P}_a(D) = D \cup \bigcup_{a'<a} D^{+a'} \cup D^{-a'},$$

where $D^{+a}(D^{-a})$ denotes a dilation(erosion) by a circle of radius a.

We are going to construct a candidate graph for each member of $\mathcal{P}_a(D)$, but we are going to consider only the pixels of D contained in a band around its contour.

Definition 2(Optimization band): Let $D \subset \Omega \subset \mathbb{Z}^2$ a digital set and $n > 0$. The optimization band $O_n(D)$ is defined as

$$O_n(D) := \{p \in \Omega \mid -n \leq d_D(p) \leq n\}.$$

For each $D' \in \mathcal{P}_a(D)$ we construct the capacited graph $\mathcal{G}_{D'}(\mathcal{V}, \mathcal{E}, c)$ with vertex and edge sets defined as

$$\begin{aligned} \mathcal{V} &= \{v_p \mid p \in O_n(D')\} \cup \{s, t\} \\ \mathcal{E} &= \mathcal{E}_{st} \cup \mathcal{E}_{\mathcal{N}}, \end{aligned}$$

where s, t are the source and target vertices, respectively, and

$$\begin{aligned} \mathcal{E}_{st} &= \{(s, v_p), (v_p, t) \mid p \in O_n(D')\} \\ \mathcal{E}_{\mathcal{N}_k} &= \{\{v_p, v_q\} \mid p \in O_n(D') \text{ and } q \in \mathcal{N}_k(p)\}. \end{aligned}$$

In case of image segmentation, we assume that there exist sets $\mathcal{V}_{fg}, \mathcal{V}_{bg} \subset \mathcal{V}$ corresponding to foreground and background seeds furnished by the user. The cost function $c : \mathcal{E} \to \mathbb{R}$ is defined as

edge e	$\mathbf{c(e)}$	**for**
$\{v_p, v_q\}$	$\beta \cdot \big(u(D', p) + u(D', q)\big) + \gamma_b \cdot \psi_2(0, 1)$	$\{v_p, v_q\} \in \mathcal{E}_{\mathcal{N}}$
$\{v_p, s\}$	$\gamma_r \cdot \psi_1(0)$	$p \in O_n(D'), v_p \notin \mathcal{V}_{fg} \cup \mathcal{V}_{bg}$
	M	$v_p \in \mathcal{V}_{fg}$
	0	$v_p \in \mathcal{V}_{bg}$
$\{v_p, t\}$	$\gamma_r \cdot \psi_1(1)$	$p \in O_n(D'), v_p \notin \mathcal{V}_{fg} \cup \mathcal{V}_{bg}$
	0	$v_p \in \mathcal{V}_{fg}$
	M	$v_p \in \mathcal{V}_{bg}$

where the constant M is given by

$$M = 1 + \max_{p \in O_n(D')} \beta \cdot \big(u(D', p) + u(D', q)\big) + \gamma_b \cdot \psi_2(0, 1).$$

Let $mincut(Q, \mathcal{G})$ a predicate indicating that Q is a minimum cut set of some capacited graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, c)$. We define the solution candidates set of digital set D as

$$sol(D) = \bigcup_{D' \in \mathcal{P}_a(D)} \Big\{ \big(Q, D(Q)\big) \mid mincut(Q, \mathcal{G}_{D'}) \Big\}.$$

8.1.2 GraphFlow algorithm

The GraphFlow algorithm implements a local-search strategy to minimize Equation (8.2) with search space given by the solution candidates set defined in the previous section. We opted to not stop the method in the case that shape $D^{(k+1)}$ has higher energy than $D^{(k)}$ as a strategy to escape local minimum. In the implementation presented here, the unique stop condition is the number of iterations. Clearly, this strategy can be reviewed depending on the application.

We remark that the GraphFlow algorithm has two fundamental steps. In the candidate selection, we build the solution candidates set from the minimum cuts of the candidate graphs. Next, in the validation step, we choose the digital set with minimum value for Equation (8.2). If we interpret the balance coefficient minimization as the best move one can make towards digital elastica minimization, the solution candidates set can be seen as the neighboring shapes with highest potential to minimize the elastica energy for the given a-probe set.

input : An image $\boldsymbol{I}$ or a digital set D; the optimization band n; the probe set parameter a; parameter vector $\boldsymbol{\theta} = (\alpha, \beta)$; parameter vector $\boldsymbol{\gamma} = (\gamma_r, \gamma_b)$; the maximum number of iterations maxIt;

if *Image* $\boldsymbol{I}$ *is given* **then**
  $D^{(0)} \longleftarrow graphcut(\boldsymbol{I})$;
end
else
  $D^{(0)} \longleftarrow D$;
  $(\gamma_r, \gamma_b) \longleftarrow (0, 0)$;
end

$k \longleftarrow 1$;
while k $<$ maxIt **do**
  `//Candidate selection`
  $sol(D^{(k)}) \longleftarrow \bigcup_{D' \in \mathcal{P}_a(D^{(k)})} \left\{ \left(Q, D(Q)\right) \mid mincut(Q, \mathcal{G}_{D'}) \right\}$;
  `//Candidate validation`
  $(Q^{(k)}, D^{(k)}) \longleftarrow \underset{(Q,S) \in sol(D^{(k)})}{\arg\min} \hat{E}_{\boldsymbol{\theta}}(S) + E^{gcut}_{\boldsymbol{\gamma}}(\mathcal{G}_{\boldsymbol{I}}, Q)$;
  k $\longleftarrow$ k $+1$;
end

Algorithm 5: GraphFlow algorithm.

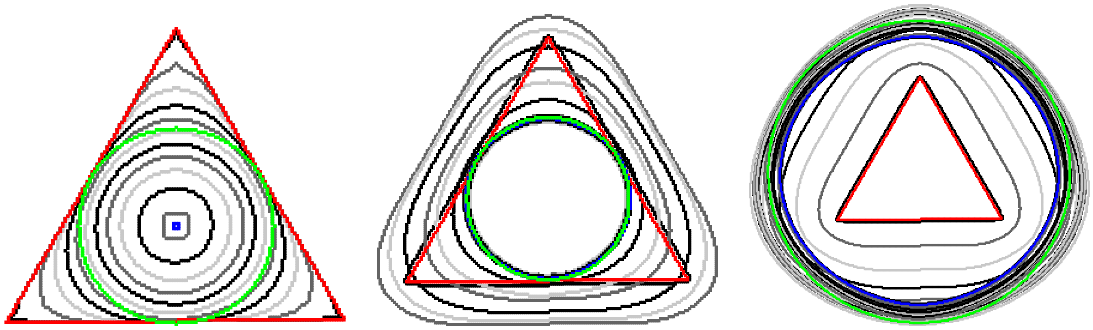
The GraphFlow algorithm produces a flow that is much more in accordance with our expectations for a flow guided by the elastica energy than the previous models. We recall that both FlipFlow and BalanceFlow have a shrinking bias that let them behave in a similar fashion to the curvature flow. On the other hand, the GraphFlow grows and shrinks in accordance with the α coefficient in the digital elastica (see Figure 8.1). If we use a 0-probe set, we recover the convergence to a single point behavior, confirming the linking between the previous models.

In fact, the solution for the free elastica problem is very similar to those given by the enumerative process of Chapter 5, i.e., the shapes converge to the expected global optimum, but with the advantage of producing smoother flows and much faster than the LocalSearch algorithm (up to 100× faster). However, for the constrained elastica problem, the GraphFlow encounters some difficulties to evolve (see Figure 8.1), in particular for the fixed endpoints' orientation instances. We believe that a larger neighborhood, possibly random, could solve this issue. The results are explored in more details in Chapter 9.

Some preliminar results of Algorithm 5 applied to image segmentation are shown in Figures 8.2 and 8.3. In particular, we can observe that the GraphFlow presents the completion property, i.e., it tends to return a segmentation with fewer disconnected components.

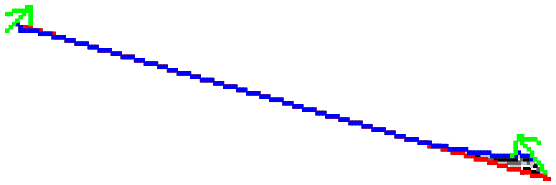
8.2 Conclusion

We described a graph cut model that regularizes the squared curvature and it is suitable for image segmentation. The evolution produced by the GraphFlow responds to the length penalization term α, i.e., the shape tends to grow (shrink) for lower (higher) values of α and we observe a convergence to a shape closer to the global optimum in the free elastica problem. In the constrained elastica, we believe that a larger neighborhood is necessary to produce better results. Finally, the GraphFlow Algorithm 5 is faster and simpler to implement than the previous models presented in this thesis.



$(n = 2, a = 0, \alpha = 0.01)$ $(n = 2, a = 2, \alpha = 0.01)$ $(n = 2, a = 2, \alpha = 0.001)$

(a)



(b) $(n = 2, a = 2, \alpha = 0.001)$

Figure 8.1: **GraphFlow results.** The GraphFlow algorithm can shrink and grow in accordance with length penalization and it converges to a shape closer to the global optimum (green curve) in the free elastica problem. In the constrained elastica, we believe that we can improve the results by using a larger, possibly random, neighborhood. We are using $n = 2, a = 2$ and shapes are displayed at every 10 iterations.

Figure 8.2: **GraphFlow segmentation.** Given foreground (green) and background (gray) seeds at picture (a); Graph cut produces picture (b) which is used as input of the GraphFlow algorithm; in pictures (c) and (d) we display the output of GraphFlow algorithm with and without squared curvature regularization.

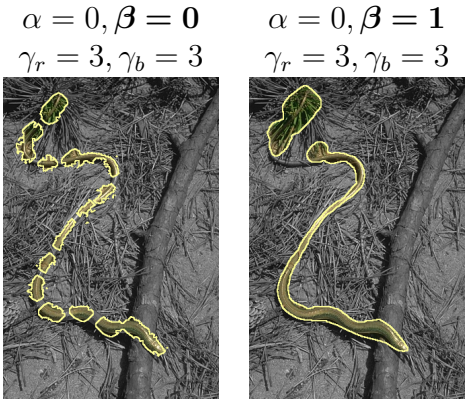


Figure 8.3: **GraphFlow and completion property**. The oversegmented picture in the left was obtained with no squared curvature regularization, while the picture in the right was obtained by setting $\beta = 1.0$.

Chapter 9

Experimental analysis

In this chapter we analyse the results produced by the four models developed in this thesis. We are going to compare their outputs for each of the three problems considered: free elastica, constrained elastica and image segmentation. Table 9.1 summarizes the models properties.

In the image segmentation section, we compare our results with the linear model for curvature regularization of [**schoenemann09linear**].

Model	Implementation	RT	Free Elas.	Constrained Elas.	Image Seg.
LocalSearch (LS)	medium	slow	yes(opt)	yes	no
FlipFlow (FF)	hard	acceptable	yes	no	yes
BalanceFlow (BF)	medium	acceptable	yes	no	yes
GraphFlow (GF)	easy	fast	yes(opt)	yes	yes

Table 9.1: **Models summary.** The qualitative attributes are relative, e.g., the GraphFlow presents the lowest running time while LocalSearch presents the highest.

9.1 Free elastica

The free elastica problem consists in finding a shape with the lowest digital elastica. The approach to solve this problem, as well the two others that follow, is to iteratively evolve an initial shape to another with lower digital elastica value. We have ran two experiments, summarized in Table 9.2, to illustrate the evolution process behavior for each of the models described in this thesis.

We make a distinction between the radius used to compute the balance coefficient (*bRadius*) and the one used to estimate curvature using the II-estimator (*vRadius*) in the validation function of GraphFlow and LocalSearch. In particular, the *vRadius* is the one used to plot the graphs in Figures 9.2 and 9.4. Moreover, the *vRadius* is always scaled by the grid step, while the *bRadius* is never scaled.

							FF,BF	GF	
Experiment	$maxIt$	$vRadius$	$bRadius$	h	α	β	m	a	n
Exp-General	400	5	7	0.25	0.01	1	5	2	3
Exp-Radius	400	5	7 12	0.25	0.001	1	5 10	2	3

Table 9.2: **Parameter settings for the free elastica experiments.** The headers FF,BF,GF identifies specific parameters for the FlipFlow, BalanceFlow and GraphFlow models, respectively.

	Pixels (initial shape)	LocalSearch	FlipFlow	BalanceFlow	GraphFlow
Triangle	8315	4.8s/it	0.4s/it	0.38s/it	0.14s/it
Square	12769	2s/it	0.51s/it	0.47s/it	0.12s/it
Ellipse	10038	3.1s/it	0.64s/it	0.57s/it	0.1s/it
Flower	26321	12.3s/it	1.23s/it	0.94s/it	0.14s/it
Bean	25130	6.4s/it	1.2s/it	1.17s/it	0.16s/it

Table 9.3: **Exp-General summary.** Running time and input size of Exp-General experiment for the free elastica.

9.1.1 Exp-General

The Exp-General experiment executes each model using the listed parameters in Table 9.2 for 5 different parametric shapes. The results for the Exp-General experiment are shown in Figure 9.1. One can check in the plots of Figure 9.2 how the digital elastica value evolves at each iteration. For this experiment we also provide Table 9.3 with the model's respective running times.

We observe that both LocalSearch and GraphFlow evolves the initial shape to another shape closer to the optimal one, e.g., for $\alpha = 0.01$, the model evolves to the disk of radius 10. However, the GraphFlow model is simpler to implement and much faster than LocalSearch (see Table 9.3). Even with a smaller neighborhood, the GraphFlow achieves its convergence before LocalSearch in two occasions, one in the square and another in the flower evolution.

At the first iterations, FlipFlow and BalanceFlow produce shapes with lower digital elastica energy. However, the models do not stop to evolve even if a shape of smaller perimeter and lower digital elastica ceases to exist, and starting from this point, the digital elastica value increases.

9.1.2 Exp-Radius

In the Exp-Radius experiment, we set the length penalization parameter to $\alpha = 0.001$. Compared to the Exp-General, the expected behavior is that the shapes will

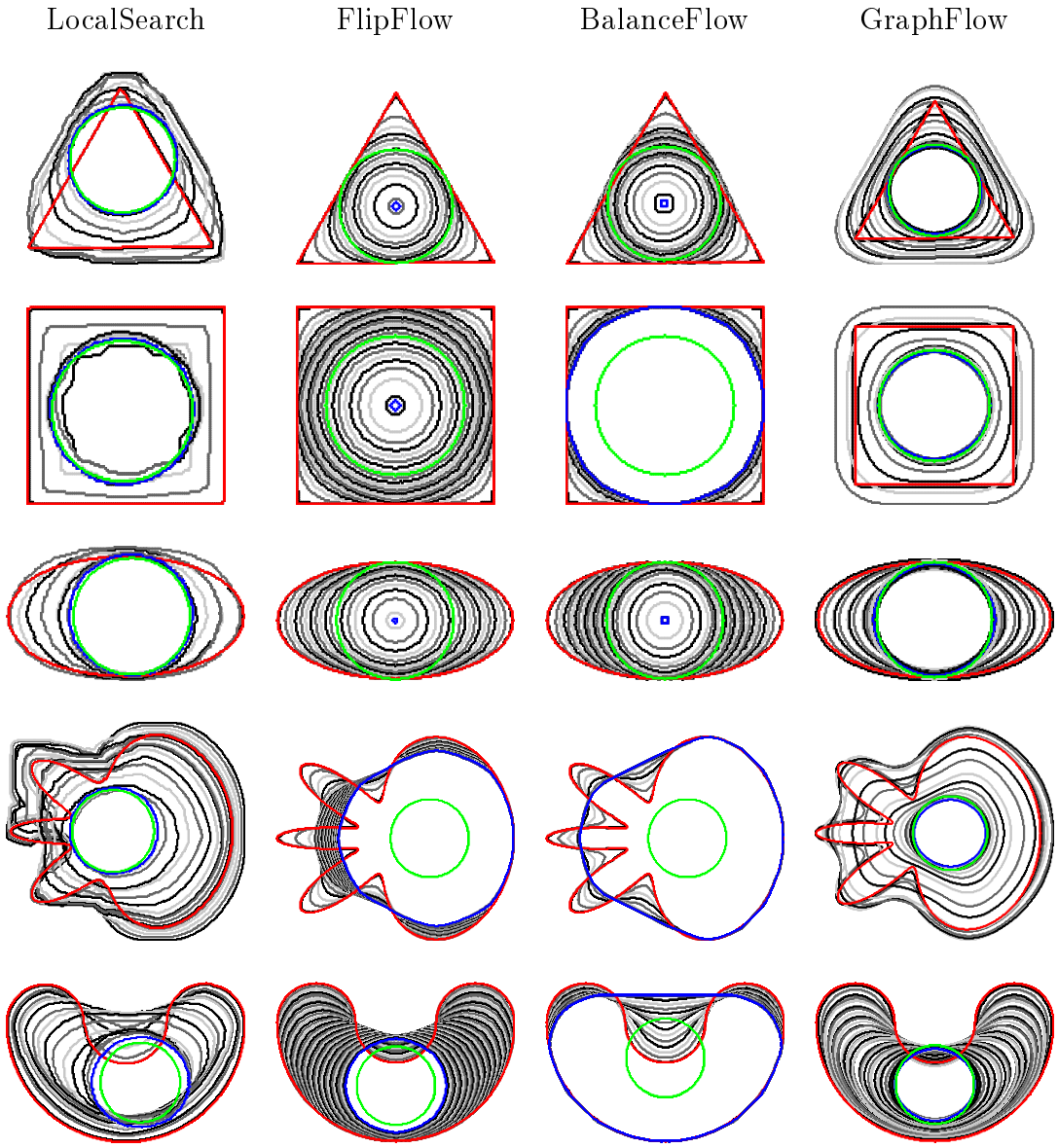


Figure 9.1: **Exp-General results for the free elastica.**. Initial contour is colored in red, final contour is colored in blue and optimal contour is colored in green. Curves are drawn every 10 iterations.

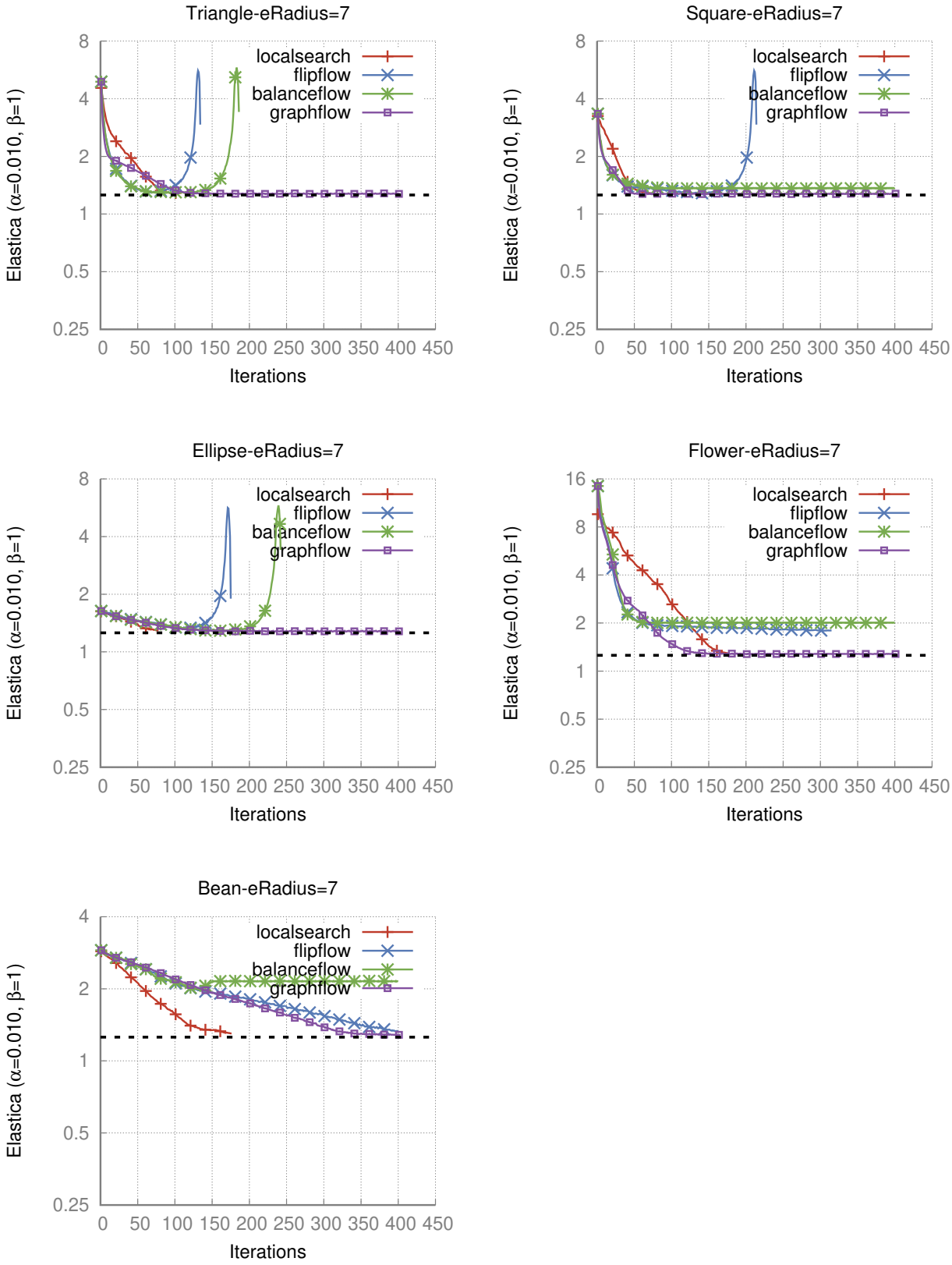


Figure 9.2: **Exp-General plots for free elastica**. LocalSearch and GraphFlow converges to values closer to the global optimum. The dashed line marks the optimum value.

grow till reach the optimal disk of radius $1/0.001^{0.5} \approx 31$. This experiment confirms the natural observation that the choice of the *bRadius* parameter influences the produced flows. The experiment was not executed for the LocalSearch model as this model is not sensitive to the *bRadius* parameter.

In the case of FlipFlow and BalanceFlow, the evolution goes faster with a larger radius, and, as mentioned before, the shape never grows, it only shrinks. On the other hand, GraphFlow is sensitive to the value of α and the shapes can grow or shrink acoordingly. Moreover, the choice of *bRadius* defines how closer the solution will be from the optimum in the case of the GraphFlow.

We recall that the II estimator measures curvature by using a disk of a given radius. The radius parameter defines the range of values estimated by the estimator. At first glance, a larger radius returns a more precise estimation, but we should be careful in not using a radius larger than the reach of the shape at the point of estimation (see Figure 9.5). A value of $bRadius = 7$ is too small to identify the small variations that a shape growing to a disk of radius 31 suffers. Therefore, when we set $bRadius = 12$, the GraphFlow returns solutions closer to the optimal, as we can check in Figures 9.3 and 9.4. The results suggests that the *bRadius* should be dynamically set in order to escape local optimum solutions.

9.2 Constrained elastica

The constrained elastica problem consists in finding the shape of minimum digital elastica that respects some set of constraints. We ran experiments for two sets of constraints: in the first, we impose that a set of pixels in the digital boundary of the initial shape must persist in the final shape; in the second, we evolve a curve whose endpoints' orientations are fixed.

For the constrained elastica, only LocalSearch and GraphFlow were evaluated. We believe that both FlipFlow and BalanceFlow can be modified to evolve the constrained elastica, but such modifications were not implemented in this thesis. Table 9.4 lists the parameters used in the experiments and Table 9.5 the running time of Exp-α experiment.

We remark that for every experiment in this section the grid step is set to $h = 1.0$, i.e., the Euclidean and digital radius are the same. In contrast with the previous section, where all the shapes had a closed parametric formula, some of the tested shapes in this section are created ad-hoc and a decision about the grid step in this case becomes arbitrary.

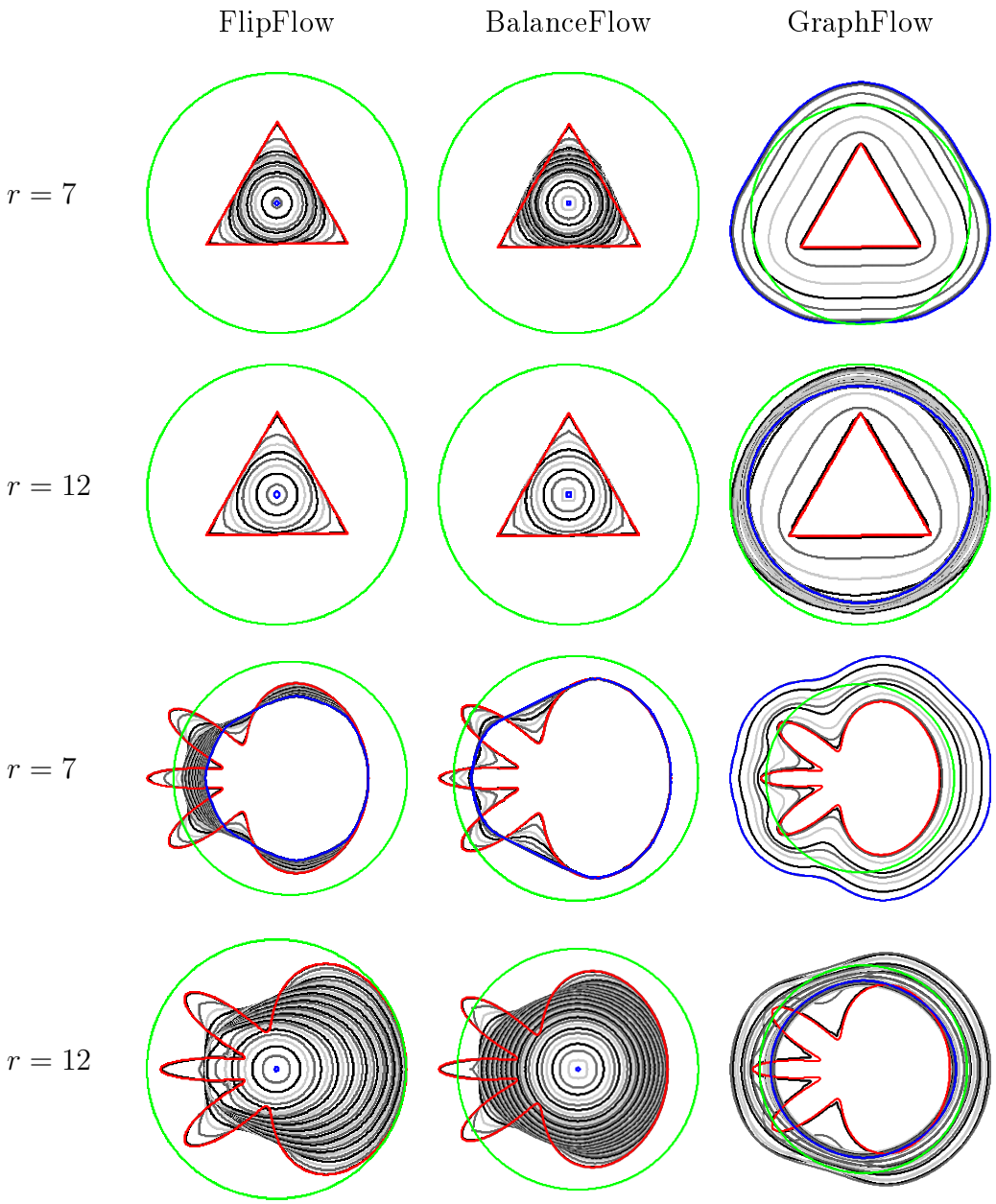


Figure 9.3: **Exp-Radius results for the free elastica.** A small value of *bRadius* may not be sufficient to identify small variations in smooth shapes, resulting in a premature on a local minimum. We clearly observe this behavior in the GraphFlow model. Initial contour is colored in red, final contour is colored in blue and optimal contour is colored in green. Curves are drawn every 10 iterations.

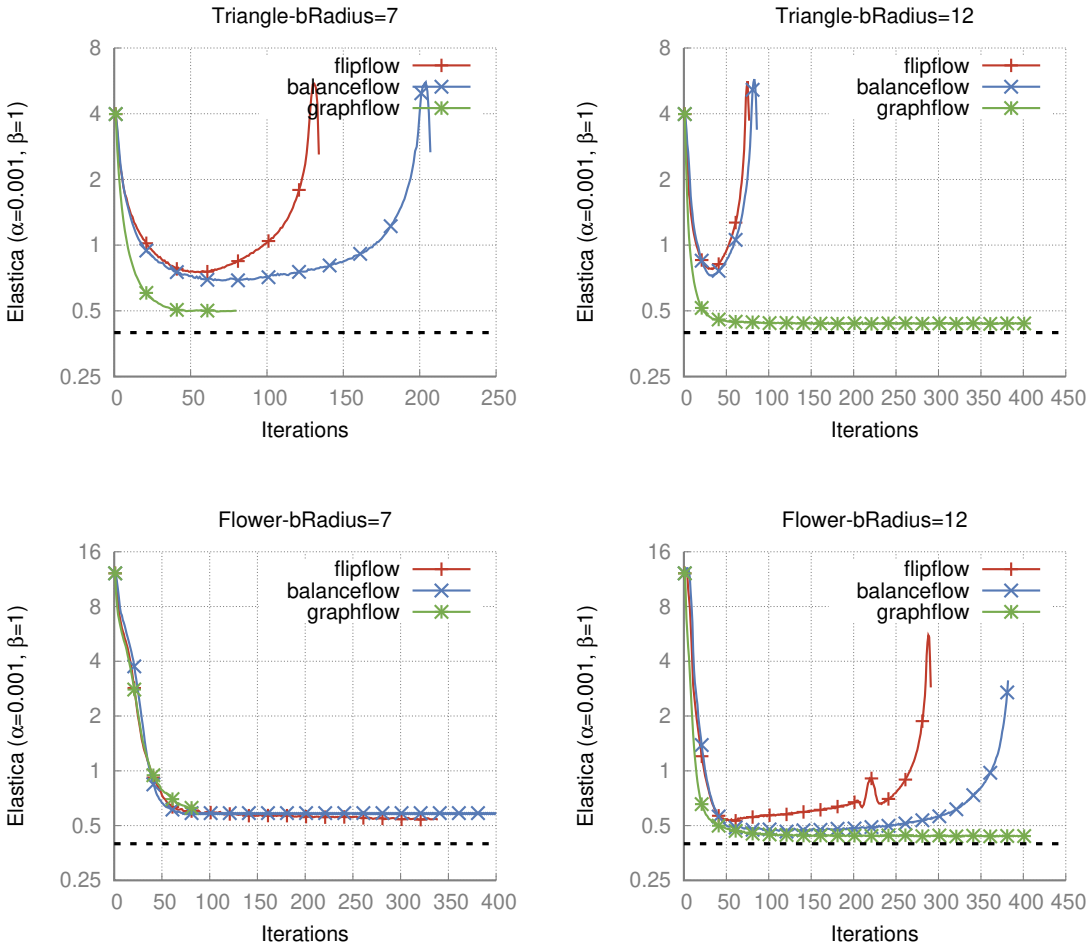


Figure 9.4: **Exp-Radius plots for the free elastica.** The GraphFlow approaches values closer to the global optimum when choosing a higher value for *bRadius* in the case $\alpha = 0.001$.

							GF	
Experiment	*maxIt*	*vRadius*	*bRadius*	h	α	β	a	n
Exp-α	400	7	7	1.0	0.002 0.0002	1	1	2
Exp-Radius	400	15 50	15 50	1.0	0.002	1	1	2

Table 9.4: **Parameter settings for the constrained elastica experiments**. The headers LS,GF identifies parameters that are exclusive for the LocalSearch and GraphFlow models, respectively.

Note about implementation

In the LocalSearch model, we simply ignore the solutions which do not respect the constraints. In the GraphFlow model, we judiciously set the cost function to guarantee that pixels marked fixed will belong to the source component of the cut and will remain a pixel of the digital contour.

Let $A \subset D$ be the set of fixed pixels of digital set D and $A' \not\subset D$ their neighbors not contained in D. Next, we set the following edge capacities

$$c(\{s, v_a\}) = M, \quad \forall a \in A$$
$$c(\{v_{a'}, t\}) = M, \quad \forall a' \in A',$$

where M is set as the highest capacity of the graph.

9.2.1 Discussion

The results show that the GF model is not appropriate to solve the constrained elastica problem and we identify two reasons for that.

Poor shape neighborhood. The a-probe set promotes quite rough and simple transformations to capture the fine information around the fixed pixels, and we confirm this hypothesis by comparing the GF results with those of the LS model, much more suitable for this kind of task because of its heterogeneous neighborhood.

Unresponsiveness of balance coefficient to fixed pixels. The balance coefficient computation ignores that some of the pixels will stay fixed and encourage movements that increases the elastica instead of decreasing. That is particularly clear in Figure 9.6c of Curve-1. The region around the right endpoint is convex, and the balance coefficient encourages a contraction movement on the non-fixed pixels. Because of fixed pixel constraint, an expansion movement should be expected.

Exp-α

We present results for two different values of length coefficient of the digital elastica. The LS model behaves as expected, producing longer and smoother shapes for higher values of α. That is particularly clear in Figures 9.6a, 9.6b, 9.6e and 9.6f. Nonetheless, we observe that the proposed neighborhood is no more sufficient to escape bad local optimum solutions (Figures 9.6i, 9.6j, 9.6m and 9.6n).

The GF model, on the other hand, fails to decrease the digital elastica for the curves examples of Figures 9.6c, 9.6d, 9.6g and 9.6h and it has shown to be insensitive to the length coefficient for the tested cases.

(a) Curve-1 (b) Curve-2 (c) Too large radius

Figure 9.5: **Constrained elastica curves and radius value**. The underlying shapes of Curve-1 and Curve-2 for constrained elastica experiments are shown in figures (a) and (b). Pixels in red are forced to persist in the solution. In figure (c), an example of a too large radius value (50), larger than the shape reach.

Exp-Radius

In this experiment we evaluate the results for different values of *bRadius*. We observe that a larger radius may be beneficial because a larger disk is sensitive to a higher range of curvature values, and consequently, it produces smoother shapes (see Figures 9.7a, 9.7e, 9.7i and 9.7m). However, one should not set the radius to a value larger than the reach of the shape, as the curvature estimation becomes compromised Figures 9.7b and 9.7f. We observe, once again, that a dynamic setting of the radius of the estimation ball is a more appropriate strategy.

The GF model repeats the bad results of the last experiment, and even create disconnected components Figures 9.7k and 9.7o.

9.3 Image segmentation

The FlipFlow,BalanceFlow and GraphFlow can be extended to do image segmentation. In this section we show the results of several experiments that illustrates the influence of each of the weight parameters (length,curvature,data) and the radius of the estimation ball in the produced segmentation. In the last section we compare our results with the linear programming model for squared curvature regularization SLCR [**schoenemann09linear**].

All three models (FF,BF,GF) need an initial segmentation as input. This segmentation is given by the Graph cut algorithm. Table 9.6 lists the parameters configuration for each experiment and Table 9.8 summarizes their running times.

The experiments are divided in two sections. In the first, we study the influence of each parameter in the produced segmentation and in the second we compare our results with those produced by SLCR. For the same reason described in the previous section, the grid step in all experiments is set to $h = 1.0$.

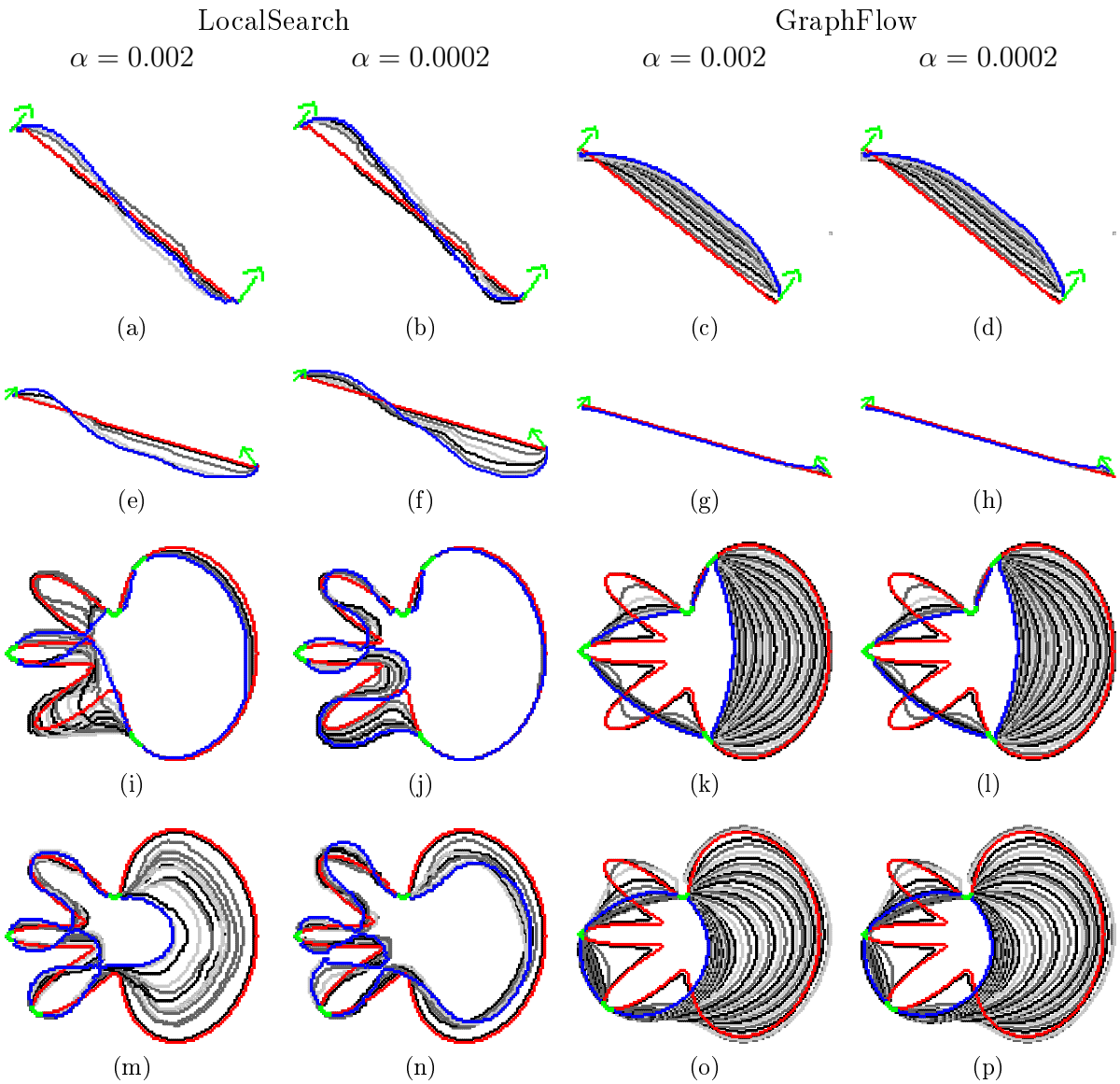


Figure 9.6: **Results of Exp-α for the constrained elastica.** Initial contour is colored in red and final contour is colored in blue. The green pixels indicates pixels forced to persist in the final contour, or a forced orientation. Curves are drawn every 10 iterations.

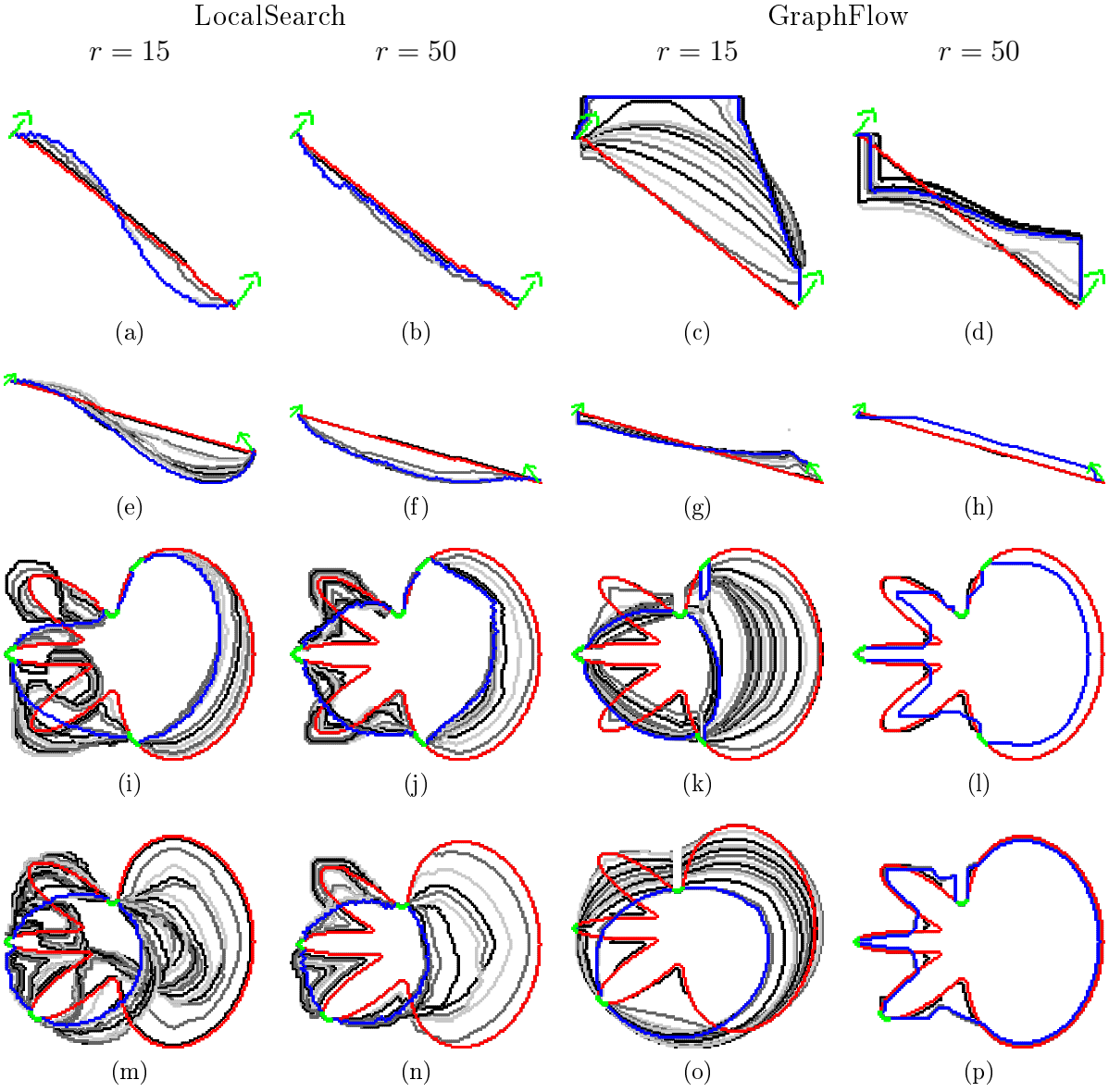


Figure 9.7: **Results of Exp-Radius for the constrained elastica.** Initial contour is colored in red and final contour is colored in blue. The green pixels indicates pixels forced to persist in the final contour, or a forced orientation. Curves are drawn every 10 iterations.

	Pixels (initial shape)	LocalSearch	GraphFlow
Curve-1	12306	4.7s/it	1s/it
Curve-2	11527	6.2s/it	1s/it
Flower-1	7481	4.5s/it	0.3s/it
Flower-2	7481	2.5s/it	0.21s/it

Table 9.5: **Running time and input size of the Exp-α experiment for the constrained elastica.** The pixels columns is with respect the number of pixels in the shape. In the case of the curves, it is with respect to the underlying shapes of Figure 9.5

							FF,BF		GF			
Experiment	*maxIt*	*vRadius*	*bRadius*	h	α	β	γ	d	a	n	γ_r	γ_b
Exp-α	200	7	7	1.0	0 0.5 3.0	0	1	0	2	2	1	0
Exp-β	200	7	7	1.0	0	0.1 1 2	1	0	2	2	1	1
Exp-γ	200	7	7	1.0	0	1	1 2 5	0	2	2	1 2 5	1 2 5
Exp-*rbRadius*	200	3 7 12	3 7 12	1.0	0	3	1	0	2	2	1	1

Table 9.6: **Parameter settings for the image segmentation experiments.** The headers FF,BF,GF identifies parameters that are exclusive of FlipFlow, BalanceFlow and GraphFlow models, respectively.

Exp-Comparison												
Model	*maxIt*	*vRadius*	*bRadius*	h	α	$\beta(\lambda)$	γ	d	a	*ob*	γ_r	γ_b
FF,BF	200	7	7	1.0	0.5	1.0	1.0	0	-	-	-	-
GF	200	7	7	1.0	0.0002	1.0	-	-	2	2	3	3
SCLR	-	-	-	-	-	2.0	1.0	-	-	-	-	-

Table 9.7: **Parameter settings for the comparison experiments.** The β parameter in FF,BF,GF corresponds to the λ parameter in SCLR.

Exp-Comparison Running time			
Model	Minimum	Maximum	Average
FlipFlow	60s	297s	156s
BalanceFlow	37s	184s	93.7s
GraphFlow	11s	150s	75s
SLCR	2.87min	52.24min	18.4min

Table 9.8: Running time and input size of Exp-*bRadius* for the image segmentation problem and $bRadius = 7$.

9.3.1 Influence of parameters

The models offer parameters to control the relative weight of length (α), curvature (β) and data ($\gamma, \gamma_r, \gamma_b$). We recall that FF and BF accept a single regional parameter γ for data, while GF accepts γ_r to ponderate a regional term and γ_b to weight a boundary term. The Graph cut input and its result are shown in Figure 9.8. The experiment results are displayed in Figures 9.9 to 9.12.

The FlipFlow and BalanceFlow present similar results for all experiments, as expected. The Exp-α experiment regularizes only length, and we can observe that the segmentations produced by all three models tend to be staircased. We remark that in the GF model, length penalization is not present in the cost function of the candidate graph, but only in the validation function. In particular, we need a a-probe set with $a > 0$ so that length penalization has an influence in the produced segmentation.

In experiment Exp-β we vary the squared curvature weight coefficient. We observe in Figure 9.10 that the produced segmentation smooths out for increasing values of β. We recall that the shrink/growing behavior in the GF is controlled by the value of α. The FF and BF grow in concavities, but unless a local optimum is found, they tend to shrink.

The models' response for the variation of data term is shown in Figure 9.11. A higher value of γ tends to produce results similar to the initial segmentation given by Graph cut, i.e., with almost no length or curvature regularization.

Finally, Exp-*bRadius* illustrates how the choice of the estimation ball radius influences the segmentation. In Figure 9.12 we observe that a small radius results in contours with sharp changes of angle (first row), a consequence of the limited number of different estimations that can be given by a ball of small radius. As the radius increase, a richer variation of estimations is possible, and we have smoother contours (second-row). However, a big radius may omit important details of the object, as the coala's ears in the last row of Figure 9.12. That suggests that a multiradius approach may deliver improved segmentations.

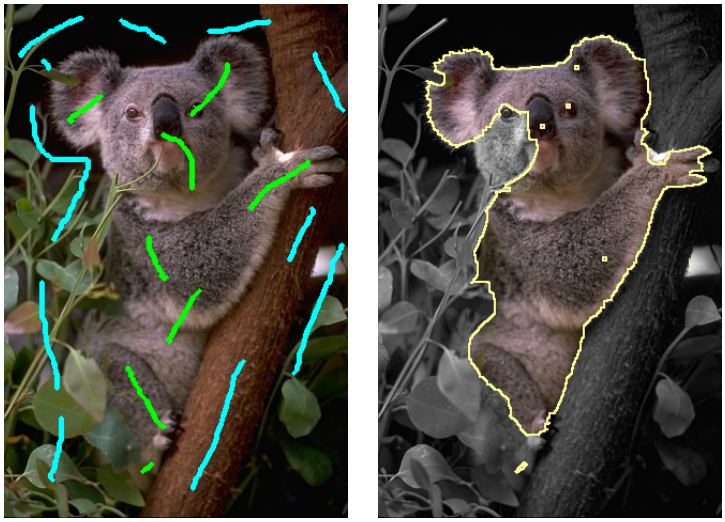


Figure 9.8: Foreground (green) and background (blue) seeds are shown in the left and the resulting Graph cut segmentation in the right.

9.3.2 Comparison

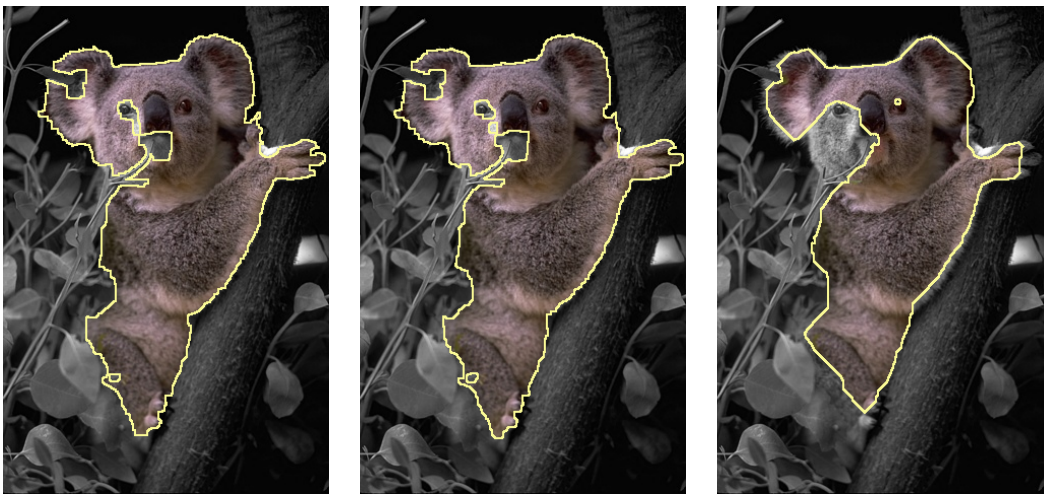
In Figures 9.13 to 9.16 we show the segmentation results of several images for the three models developed in this thesis, the SLCR and Graph cut models. We opt to set the same parameters in all instances, although a better segmentation could be obtained by setting them separately.

The curvature regularization in the FF,BF models are well perceived in the airplane, camel and man-in-white pictures, but those models are not able to correctly segment the brown-snake in Figure 9.16. They also have a hard time to segment the birds in Figure 9.14 due to the object large curvature range. Nonetheless, they correctly segmented the green-snake in Figure 9.15 while the Graph cut return three disconnected components.

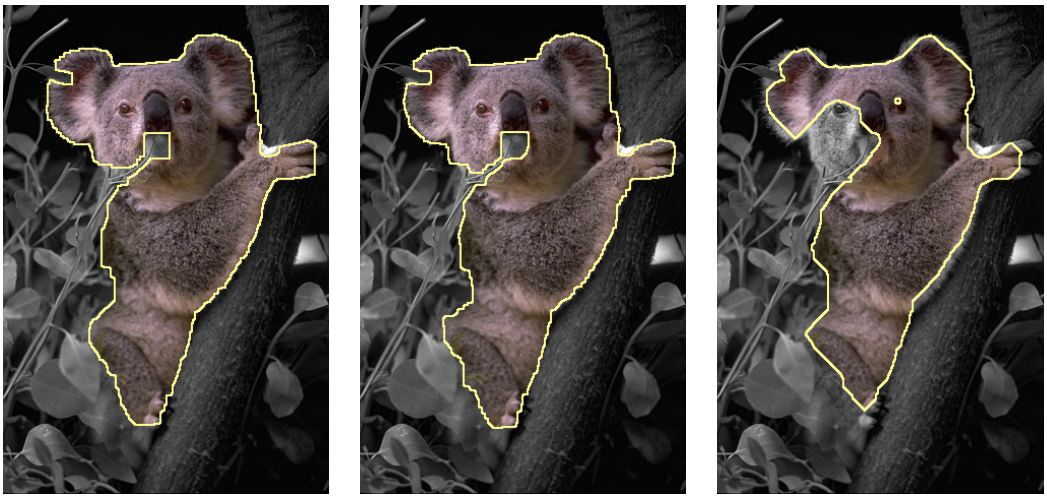
For the chosen parameters, the GraphFlow did not evolve the initial Graph cut segmentation to much, except for the brown-snake in Figure 9.16, for which GF and SLCR presented the best segmentation. However, the GF has the second lowest running time among the five models.

The SLCR tends to oversegment, notably in the man-in-white and the camel pictures. We observe that the contours present sharp turns due to the low precision of the curvature estimator. The precision could be improved by increasing the pixels connectivity (set to 8), but is likely to follow an increase in running time, which is already the highest between the models tested.

(a) $\alpha = 0.0$



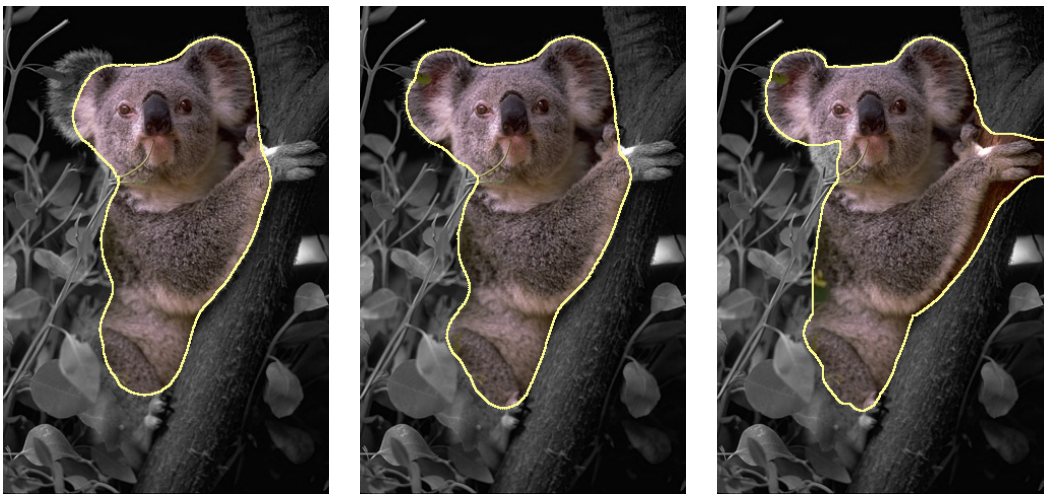
(b) $\alpha = 0.5$



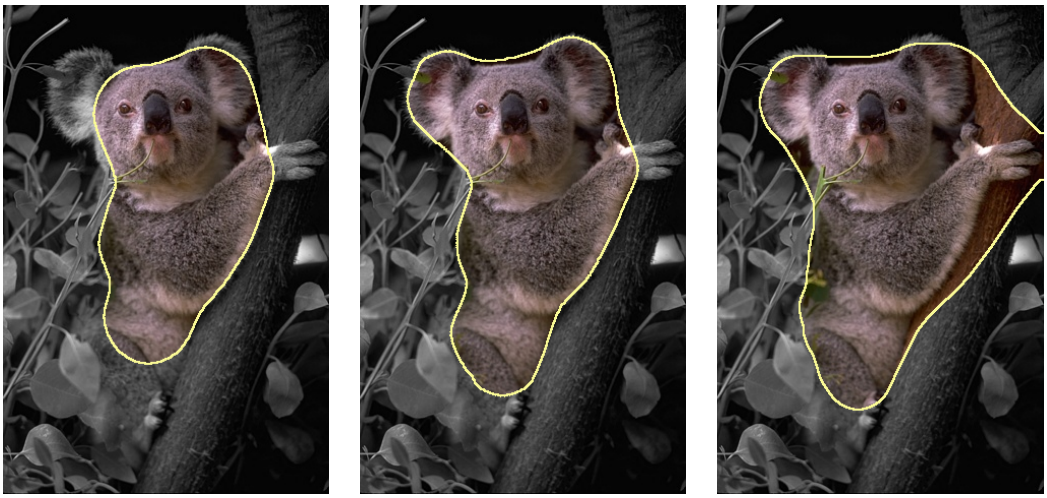
(c) $\alpha = 3.0$

Figure 9.9: **Results of Exp-α for segmentation.**

(a) $\beta = 0.1$

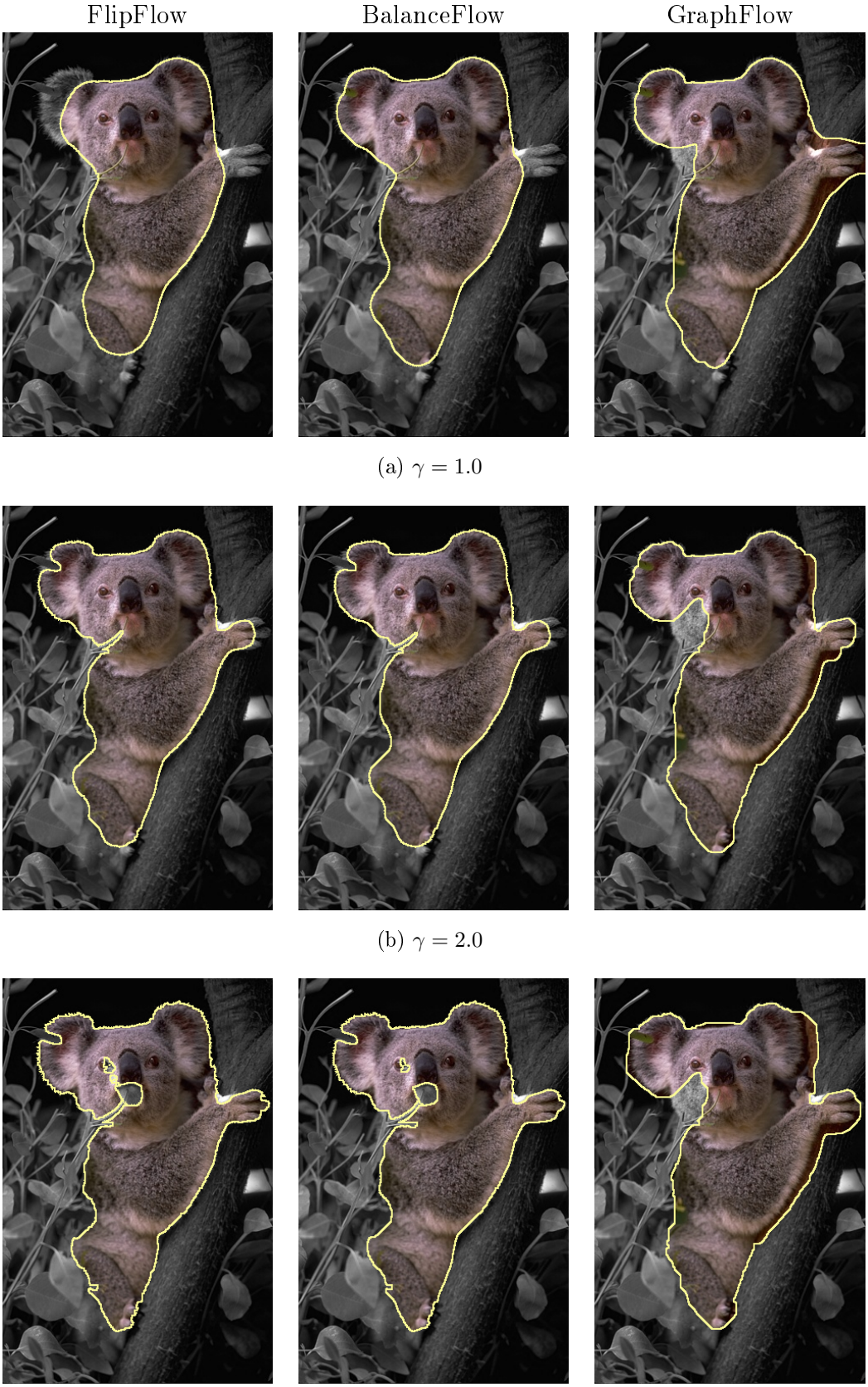


(b) $\beta = 1.0$



(c) $\beta = 2.0$

Figure 9.10: **Results of Exp-β for segmentation.**

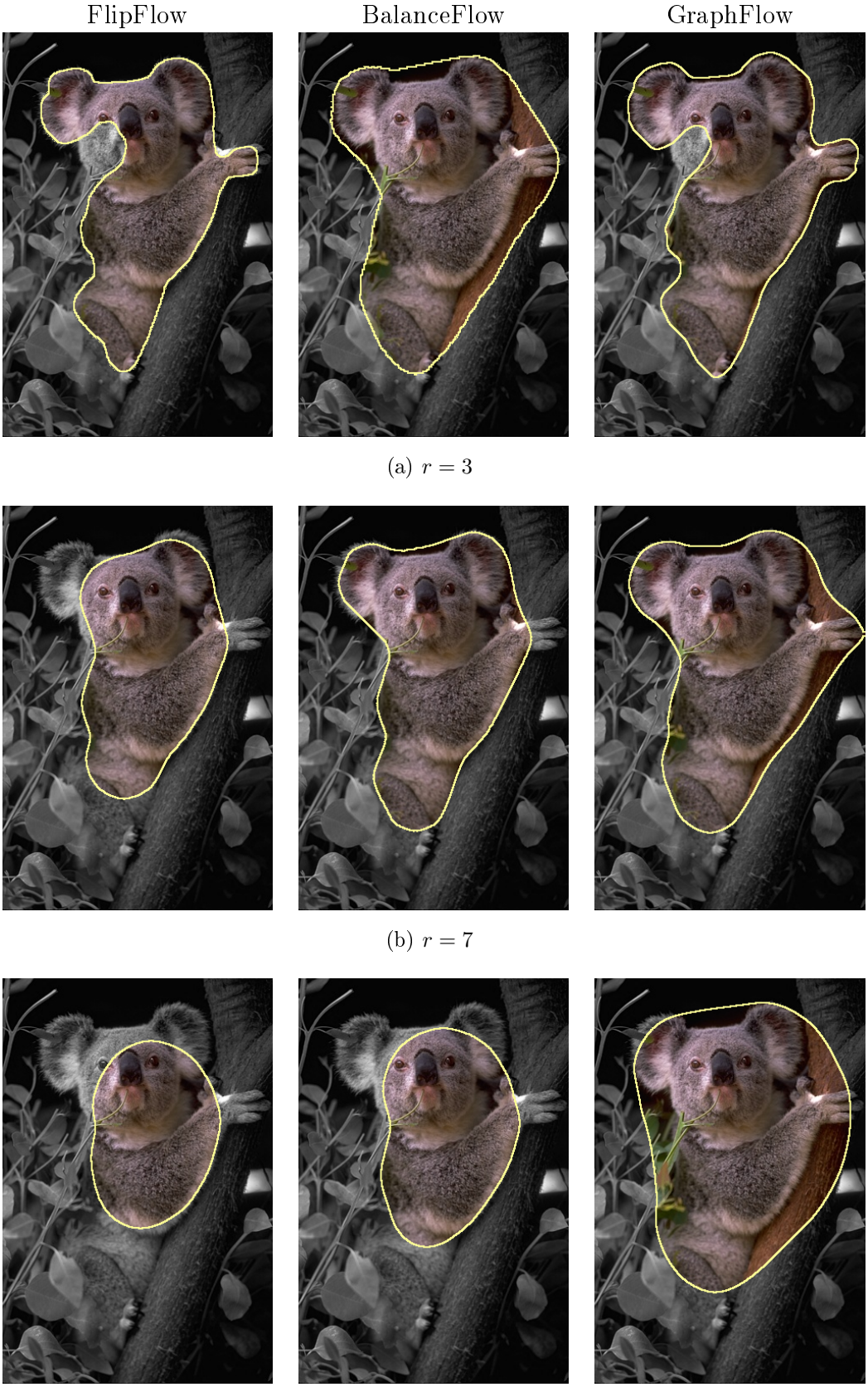


(a) $\gamma = 1.0$

(b) $\gamma = 2.0$

(c) $\gamma = 5.0$

Figure 9.11: **Results of Exp-γ for segmentation.**



(a) $r = 3$

(b) $r = 7$

(c) $r = 12$

Figure 9.12: **Results of Exp-*Radius* for segmentation.**

Figure 9.13: **Segmentation results comparison set 1**

Figure 9.14: **Segmentation results comparison set 2.**

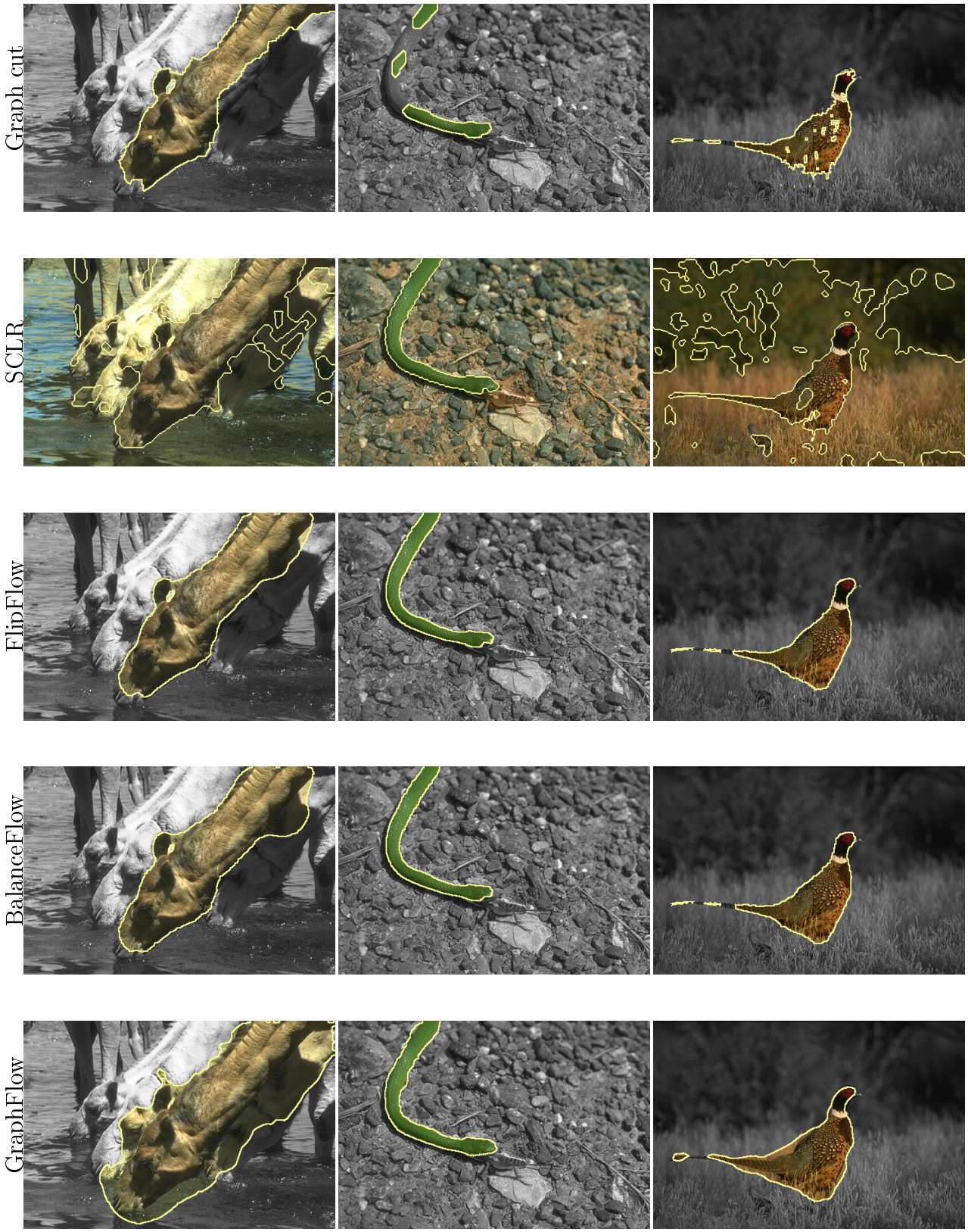


Figure 9.15: **Segmentation results comparison set 3.**

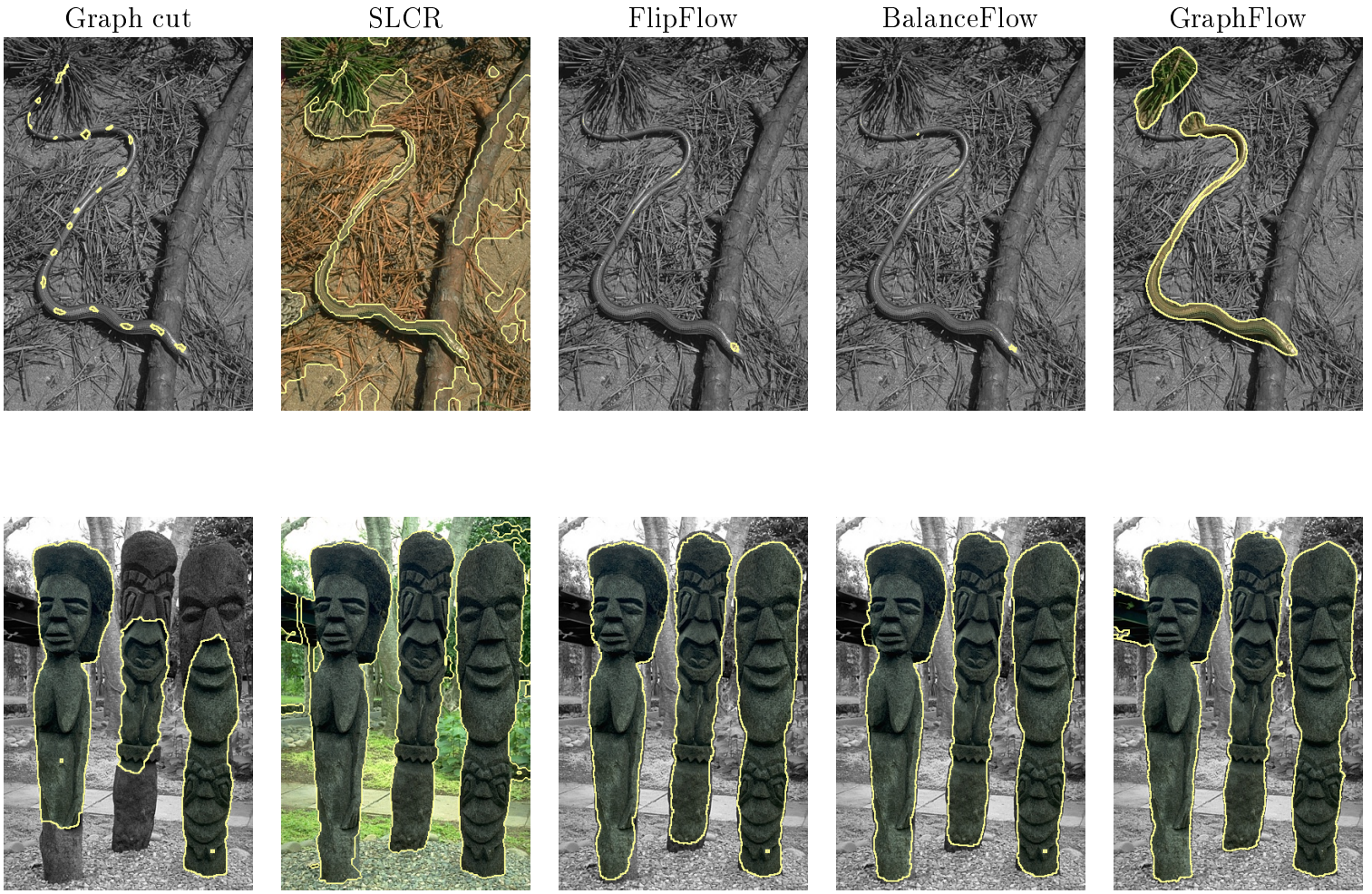


Figure 9.16: **Segmentation results comparison set 4**

9.4 Conclusion

All the four models described in this thesis can be used to produce shapes of lower digital elastica. Moreover, the FlipFlow,BalanceFlow and GraphFlow can be extended to do image segmentation with curvature regularization. In particular, the GraphFlow is a straightforward extension of the standard Graph cut model, which originally does not implement geometric regularization, and presents lower running times than FF and BF. Additionally, we observe that the GraphFlow can recover the completion property associated to the squared curvature regularization in some occasions. Finally, our models are competitive with the SLCR segmentation algorithm that employs curvature regularization.

Chapter 10

Conclusion and perspectives

The goal of this thesis was to study and propose models for image processing problems using multigrid convergent estimators of geometric properties, in particular the curvature and the minimization of the elastica energy. We argued that classical discretization schemes are based on the assumption that an exact sampling is available, which is not the case for digital images. The idea that convergence to the estimated value is achieved as the number of considered points increases is no longer valid in the digital world. A trivial extension of the linear discretization of Section 3.4 to the digital space would consider all the digital pixels, but in this case, only angle variations of 90 degrees would be measured.

Our choice for the minimization of elastica was motivated by the completion property and its applications to both inpainting and segmentation. It is a challenging problem and the current models are limited to the completion of small regions and/or have strong metrication errors. One reason for that is that the completion property is quite difficult to be recovered by local approaches. Nonetheless, we have insisted on those since our attempts of global optimization happened to be impractical.

Another difficulty is to inject the multigrid convergent estimators in an optimization framework. Estimators such as MDCA and MDSS are based on the computation of digital geometry primitives whose expression as an optimizable energy is not easy. The most suitable to this task is indeed the II estimator, for which we tested global optimization models, a constrained and an unconstrained one, but without success.

The first model we studied, the LocalSearch model, had the goal to validate the use of multigrid convergent estimators in an optimization framework. For this purpose, we were successful. In the LS model, there is no restriction with respect to the estimators used. We achieved the global optimum in several occasions for the free elastica problem, and in the constrained elastica, the final curves resembled

those minimizing the elastica. This model finds fewer applications due its high running time, but it was used as a reference for the others that followed.

The BalanceFlow was developed for the II estimator and it is based on the concept of balance coefficient. It builds up on the FlipFlow model and it can be considered as an improvement of the latter, though the models are not exactly the same. The running times are much faster and we have proposed an application to image segmentation. However, the FlipFlow algorithm presents a shrinking bias that makes it resemble more to the curve-shortening flow than the elastica flow. Consequently, the elastica is decreased until a certain inflection point.

Finally, the GraphFlow model exploits strategies present in all the previous models to produce our fastest algorithm. The balance coefficient is used to ponder the edges of the candidate graphs which are derived from a neighborhood of shapes: the a-probe set. The flows produced in the free elastica problem behaves as expected and, like the LocalSearch model, it achieves the global optimum in several occasions. We demonstrated its use in image segmentation and we were able to recover the completion property in many instances.

Perspectives

GraphFlow and perimeter. In the candidate selection step, the candidate graphs are pondered with their balance coefficients and data term(s) from image input in the case of segmentation, but we have not tested the addition of a perimeter cost, for example, the costs proposed in [**boykov03geodesics**].

GraphFlow and random neighborhood of shapes. The a-probe set is a very simple neighborhood and may not be appropriate to some scenarios, as demonstrated in the contrained elastica problem. We could take advantage of the fact that the candidate graphs are relatively small and that the candidate selection step could be implemented in parallel to devise a stronger algorithm without considerable increase in running time.

Dynamic radius. We have seen that the radius of the estimation disk should not be set to a value higher than the reach of the shape, but we also observe that a too small radius is not sensitive enough to subtle variations in a region of low curvature. A dynamic approach could be implemented with the help of the MDCA estimator, for example, which is a parameter-free estimator.

Multiresolution. We can improve the running time of segmentation in large images by employing a multiresolution approach. Moreover, the multigrid convergence property give us more guarantees of estimation quality in different resolutions and it could be very useful in strategies of this type.

Global optimization and multigrid convergent estimators. A global approach is crucial to recover the completion property in its totality, and a discrete global model using multigrid convergent estimator of curvature is still unknown.

Appendices

Appendix A

Curvature and distant disks

In Chapter 6 we present the FlipFlow algorithm and we remark that the choice of the ring number is of fundamental importance in order to derive a smooth flow. This observation lead us to ask if there is any relation between curvature and outer rings. In this appendix we give a positive answer for this question.

Let $\mathcal{C}$ an oriented curve in the plane. We center disks B_i and B_o of radius $R+\epsilon$ and its centers aligned with the normal direction of the curve at some point $p \in \mathcal{C}$. Moreover, the distance from disks center to p equals to R.

Let $\Theta_o(\Theta_i)$ to denote the intersection of $B_o(B_i)$ with the inner(outer) region of the curve. We define the function $g_R : \mathcal{R} \times \mathcal{C} \to \mathbb{R}$ as

$$g_R(p) = \Big(\Theta_i(p) - \Theta_o(p)\Big)^2.$$

Proposition 1(R-separated disks curvature): Let $\mathcal{C} \in \mathbb{R}^2$ be a curve such that for a point $p \in \mathcal{C}$ its curvature equals to κ. For $\epsilon = R/2$ and for sufficiently small values of R and κ, we can approximate g_R by

$$g_R(p) \approx \frac{125}{144} R^6 k^2 + O(R^8\kappa^4)$$

Proof: For every point p in C, consider its Frenet frame formed by the tangent vector at p, $T(p)$ and the normal vector at p, $N(p)$. We assume the origin of the frame is at point p. Let x be a variable in the axis defined by $T(p)$. Expanding $C(x)$ around the origin we obtain

$$\begin{aligned} C(x) &= C(0) + \frac{dC}{dx}x + \frac{1}{2}\frac{d^2C}{dx^2}x^2 + O(x^3) \\ &= \frac{\kappa}{2}x^2 + O(x^3). \end{aligned}$$

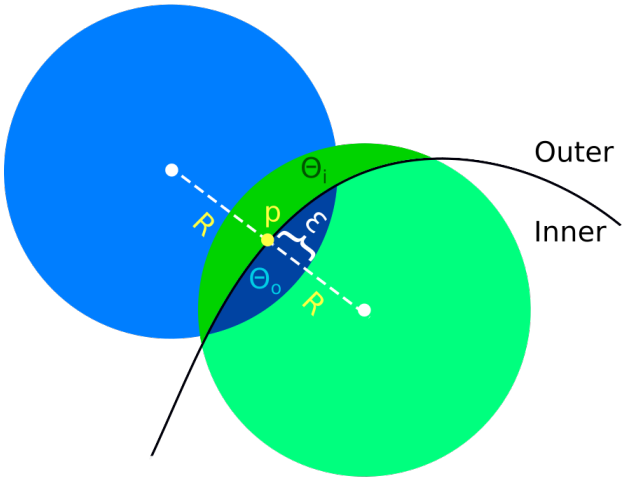


Figure A.1: disks of radius $R+\epsilon$ distant R units from $p \in \mathcal{C}$ in the normal direction.

In other words, the second order approximation for the curve C in the Frenet frame is the parabola $f(x) = \kappa/2x^2$ passing at the origin. We are going to use this parabola to estimate Θ_o and Θ_i.

We proceed by computing the intersection area Θ_o.

$$h(x) = R + \epsilon - \sqrt{(R+\epsilon)^2 - x^2}$$
$$\Theta_o = 2\int_0^{x_o} f(x) + \epsilon - h(x)$$

To compute the intersection point x_o of the parabola with the disk, we use again Pythagoras' theorem.

$$(R+\epsilon)^2 = (R - \frac{\kappa}{2}x_o^2)^2 + x_o^2$$
$$0 = \frac{\kappa^2}{4}x_o^4 + (1 - R\kappa)x_o^2 + R^2 - (R+\epsilon)^2$$

By setting $z_o = x_o^2$

$$\Delta_o = (1 - R\kappa)^2 + \kappa^2(2R\epsilon + \epsilon^2)$$
$$z_o = \frac{2}{\kappa^2}(R\kappa - 1 + \sqrt{\Delta_o})$$
$$x_o = \frac{\sqrt{2}}{\kappa}\sqrt{R\kappa - 1 + \sqrt{\Delta_o}}$$

We proceed similarly for the inner disk.

$$\Theta_i = 2\int_0^{x_i} \epsilon - f(x) - h(x)$$

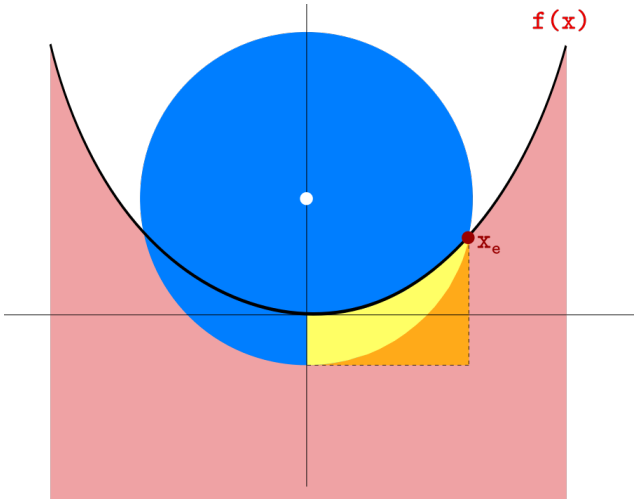
The intersection point x_i between the parabola and the inner disk is given by

$$\begin{aligned}(R+\epsilon)^2 &= (R+\frac{\kappa}{2}x_o^2)^2 + x_i^2 \\ 0 &= \frac{\kappa^2}{4}x_i^4 + (1+R\kappa)x_i^2 + R^2 - (R+\epsilon)^2 \\ \Delta_i &= (1+R\kappa)^2 + \kappa^2(2R\epsilon+\epsilon^2) \\ x_i &= \frac{\sqrt{2}}{\kappa}\sqrt{-R\kappa - 1 + \sqrt{\Delta_i}}.\end{aligned}$$

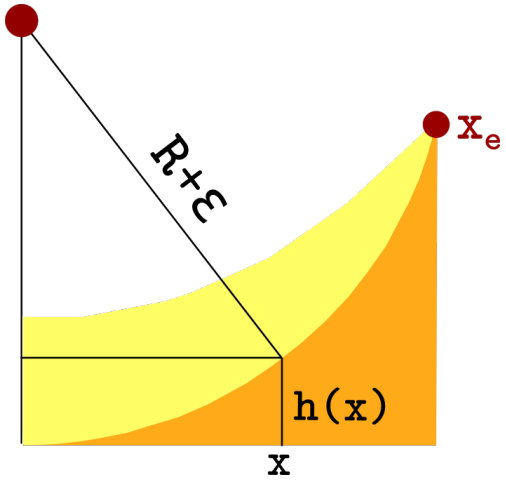
The claimed approximation is obtained by expanding g_R with its 6th order Taylor series around $\kappa = 0, R = 0$. ■

Therefore, the squared curvature at point p can be estimated as

$$\hat{\kappa}^2_{R-sep} = \frac{144}{125R^6}g_R(p).$$

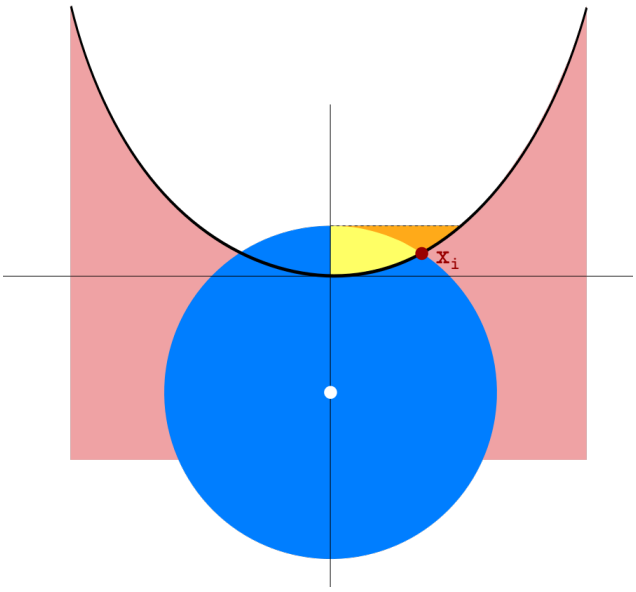


(a)

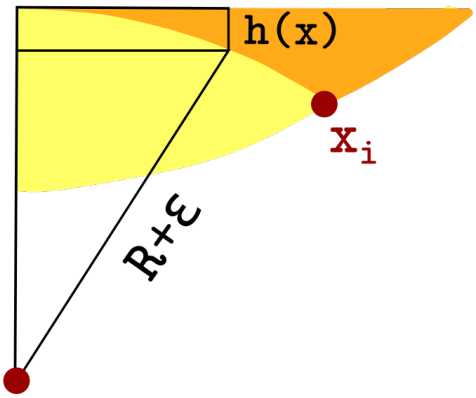


(b)

Figure A.2: The yellow area equals $\Theta_o/2$, the same value of the area under the parabola from $x = 0$ until $x = x_o$ minus the orange area $h(x)$.



(a)



(b)

Figure A.3: The yellow area corresponds to Θ_i and it equals the area between the parabola and the disk from $x = 0$ until $x = x_i$.

Appendix B

Pixel incidence matrix

In this chapter we present the pixel incidence matrix defined in the theory of discrete calculus and we propose a curvature-based model using this concept.

We restrict our analysis to the integer plan because it's where theory and application is developped in this thesis, but it can be extended to higher dimensions. In the plan $\mathbb{Z}^2$, we identify faces (pixels), edges (linels) and vertices (pointels). An arbitrary (however coherent) orientation is set for faces and edges. For example, we set faces counter-clockwise, vertical edges to point up and horizontal edges to point right (see figure).

Definition 1(Pixel incidence matrix): Let $\Omega \in \mathbb{Z}^2$ be a connected portion of the integer plan with m faces and n edges. The pixel incidence matrix $\mathbf{P} \in \mathbb{Z}^{n \times m}$ is a matrix in which each column $\mathbf{P}_j$ represents the incidence relations between face j and the edges of the domain.

A face is incident to an edge if the edge itself is part of the face's boundary and not incident otherwise. We say it is positive incident to an edge if besides incident, the face orientation agrees with the orientation the edge, and negative incident in the case it doesn't agree.

$$\mathbf{P}_{i,j} = \begin{cases} 1, & \text{face } j \text{ is positive incident to edge } i \\ -1, & \text{face } j \text{ is negative incident to edge } i \\ 0, & \text{otherwise} \end{cases}$$

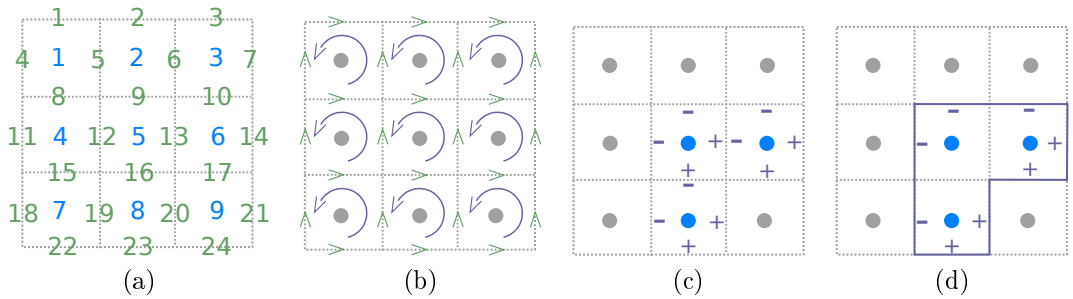


Figure B.1: Figures (a) and (b) illustrates indices assignments and orientation; figure (d) shows the result of applying the pixel incidence matrix for the active pixels in figure (c).

Example 1: Consider a portion Ω of $\mathbb{Z}^2$ with 9 faces ,24 edges and orientation as depicted in figures B.1a and B.1b. The pixel incidence matrix applied for vector $\mathbf{x} \in \mathbb{Z}^9$, representing the active pixels depicted in figure B.1c, results in vector $\mathbf{y} \in \mathbb{Z}^{24}$ representing the active boundary depicted in figure B.1d. The linel incidence matrix is simply defined as the transpose of the pixel incidence matrix. Applying the former to a vector of active linels returns its set of incident pixels. For example, $\mathbf{P}^T\mathbf{P}$ applied to vector x above results in $[0, -1, -1, -1, 2, 3, -1, 3, -2]^T$. Positive coefficients represents inner incident pixels, and negative coefficients outer incident pixels. Moreover, the absolute value of each coefficient represents the number of linels incident to the corresponding pixel. For example, pixel 6 is incident to three linels, namely the linels $10, 14, 17$.

$$\mathbf{P} = \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}, \qquad \mathbf{P} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ 0 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0
\end{bmatrix}$$