

Geometric constraints and variational approaches to image analysis

Daniel Antunes

Abstract

This is the english abstract!

Résumé

Ça c'est le résumé en Français!

Contents

Notation	9
Introduction	11
I Image Processing and Digital Geometry	13
1 Variational methods in Image Processing	15
1.1 Inverse problems in imaging	15
1.2 Bayesian rationale and total variation	19
1.3 Standard techniques	24
1.3.1 Curve evolution	25
1.3.2 Level set	28
1.3.3 Minimum path	29
1.3.4 Convex relaxation	30
2 Discrete methods in Image Processing	35
2.1 Markov Random Fields	36
2.2 Pseudo-boolean functions	42
2.3 Submodular functions	51
2.4 Graph-cut models	53
3 Curvature as regularization	61
3.1 Curvature and the curve-shortening flow	62
3.2 Diffusion and level curves motion	65
3.3 Elastica curve	69
3.4 Discrete methods and squared curvature	74
4 Digital Geometry	85

II Contribution	87
5 A combinatorial model for digital Elastica shape optimization	89
5.1 Continuous and digital Elastica	89
5.2 Local combinatorial scheme	90
5.2.1 Free digital Elastica	93
5.2.2 Constrained digital Elastica	94
5.2.3 Running time	99
5.3 Global optimization	99
5.3.1 Simplified digital Elastica	99
5.3.2 Optimization model for simplified digital Elastica	101
5.3.3 Topological constraints	103
5.3.4 Linear relaxation of P_1	104
5.3.5 Unconstrained version of P_1	105
5.4 Conclusion	105
6 A 2-step evolution model driven by digital Elastica minimization	107
6.1 FlipFlow model	107
6.1.1 Definitions	107
6.1.2 Algorithm	108
6.1.3 Algorithm discussion	111
6.2 Optimization method	113
6.3 Evaluation across m -rings	114
6.4 Data term and image segmentation	116
6.5 Conclusion	122
7 A single step evolution model driven by digital Elastica minimization	123
7.1 BalanceFlow model	123
7.1.1 Definitions	123
7.1.2 Algorithm	126
7.2 Relation with FlipFlow	126
7.3 Towards a graph-based formulation	129
7.4 Conclusion	130
8 Digital Elastica minimization via graph cuts	131
8.1 Standard graph-cut segmentation	131
8.1.1 Definitions	131
8.1.2 Grabcut binary segmentation model	132
8.2 GraphFlow model	134
8.2.1 The candidate graph	134

8.2.2	GraphFlow algorithm	135
8.3	Conclusion	137
9	Results analysis	141
9.1	Free Elastica	141
9.1.1	General experiment	142
9.1.2	Radius choice experiment	145
9.2	Constrained Elastica	145
9.2.1	Discussion	148
9.3	Image segmentation	151
9.3.1	Parameters influence	151
9.3.2	Comparison	153
9.4	Conclusion	158
Appendices		163
A	Curvature and distant disks	165
A.1	Estimating curvature with outer disks	165
B	Pixel incidence matrix	171
B.1	Cellular grid model	171

Notation

Domains

\mathbb{B}	$\{0, 1\}$
\mathbb{U}	$[0, 1]$
\mathbb{Z}	Integers
\mathbb{R}	Reals

Vectors and matrices

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

Functions

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

Sets

A	A set
\mathcal{A}	A family of sets
$\mathbb{1}_A$	Active vector for set A

Digital and continuous shapes

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

LocalSearch

d_D	Signed Euclidean distance transform of shape D
$R_m(D)$	m -ring set of digital set D
$\mathcal{N}_n(D)$	n -neighborhood of shape D
FlipFlow	
$D^{(k)}$	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
$F_r^{(k)}(p), I_r^{(k)}(p), X_r^{(k)}(p)$	Intersection of k -th sets F,I,X with $B_r(p)$
H_F, H_B	Mixed Gaussian distributions
BalanceFlow	
$O^{(k)}$	Incident pixels (inner,outer) of $\partial D^{(k)}$
$u_r(D, p)$	Balance coefficient for digital shape D centered at p
GraphFlow	
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Unordered graph with vertex set \mathcal{V} and edge set \mathcal{E}
I	3-channel colored image
O_n	Optimization band of width n

Introduction

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 A to a lower altitude point B . Which form the track should take if we wish that a ball released at A reaches B 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 into take 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 $v \in \mathbb{R}^3$ whose projection is $P(v) \in \mathbb{R}^2$	Compute the projection $P(v) \in \mathbb{R}^2$ of vector $v \in \mathbb{R}^3$
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 $\tilde{\mathbf{I}}$, compute the original image \mathbf{I} , i.e., the image without noise	Add some random noise to a given image \mathbf{I} to produce noisy image $\tilde{\mathbf{I}}$
Image inpainting: Given image $\tilde{\mathbf{I}}$ with a missing patch, reconstruct the removed patch	Remove a patch from image \mathbf{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 [Kir96], geophysics [Zhd15], natural language processing [SY05], astronomy [Luc94] and the list goes on. The image processing field itself is plenty of them [BB98]. In fact, a great part of real world applications consists into 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.

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

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

$$\mathbb{F} = \left\{ \frac{i}{255} \mid i \in \mathbb{N}, i \leq 255 \right\}. \quad (1.1)$$

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

Image denoising

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

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

$$f_{\hat{\mathbf{I}}} = \arg \min_f E(f, f_{\tilde{\mathbf{I}}})$$

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

Image segmentation

Given an image $f_{\mathbf{I}}$, the *image segmentation* problem consists in finding a partition \mathcal{I} of $f_{\mathbf{I}}$ such that each element of \mathcal{I} is identified with some desired property, usually encoded by the minimum of some functional E

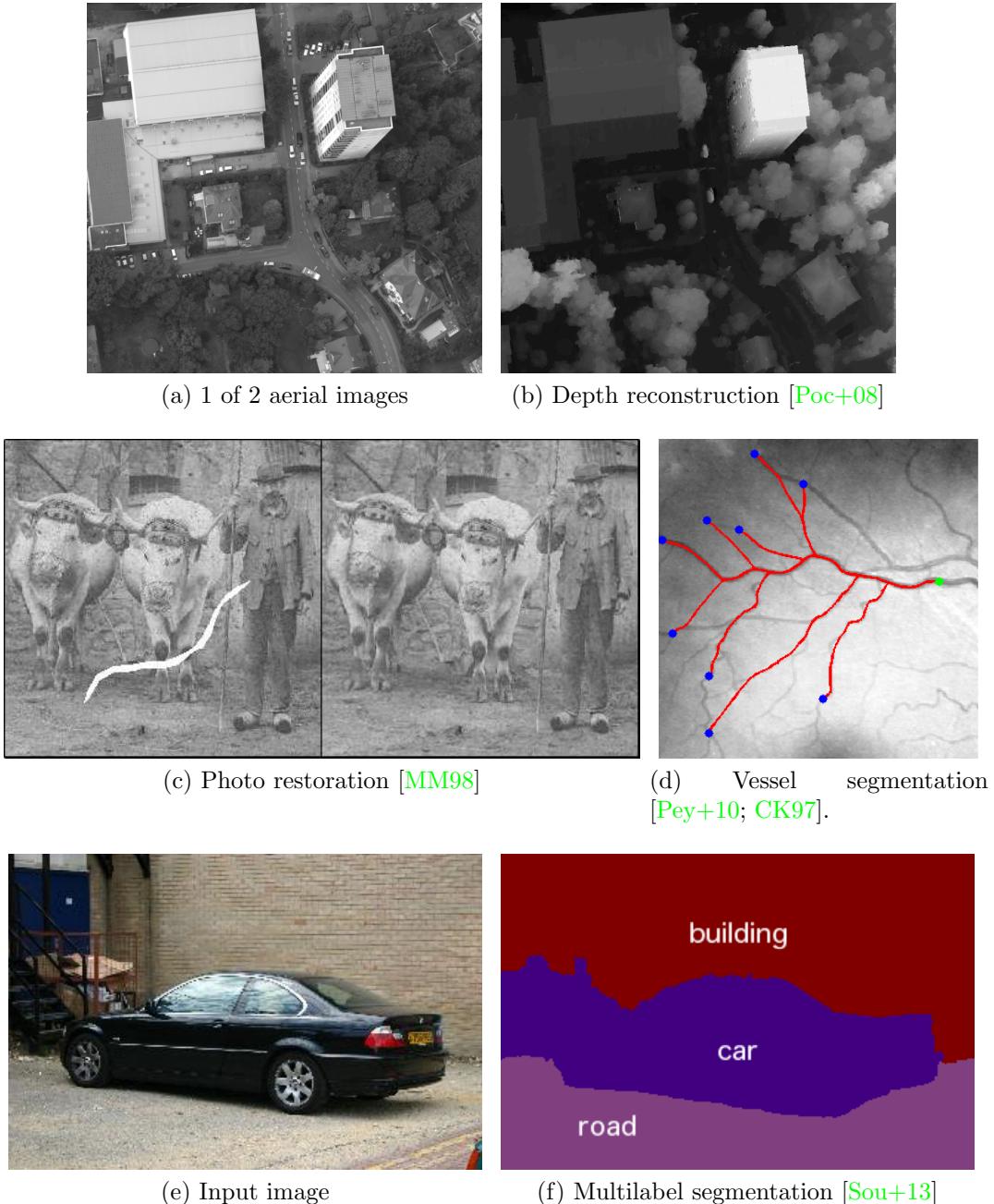


Figure 1.1: Real applications of imaging problems.

Given $f_I : \Omega \rightarrow [0, 1]$ and a positive integer n , find partition $\mathcal{I}^* = \{\Omega_i \subset \Omega \mid i \leq n\}$ such that

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

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

Image inpainting

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

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

$$f_I = \arg \min_f E(f, f_{\tilde{I}}) \quad \text{subject to } f(\Omega \setminus \mathcal{P}) = f_{\tilde{I}}(\Omega \setminus \mathcal{P}).$$

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 [GG84] and we are going to reproduce here the rational for image denoising. We make two assumptions

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

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{N}, \tag{A.1}$$

where \mathbf{N} is a $(m \times n)$ matrix of random variables $N_{i,j}$ and $Pr(N_{i,j} = n) = \frac{1}{\sqrt{2\pi}} \exp(-\lambda \frac{n^2}{2})$.

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

$$Pr(\mathbf{C}) = \exp(-\rho(\mathbf{C})). \tag{A.2}$$



(a) Original image



(b) 10-partition as in [CCP08]



(c) Noisy image



(d) Image denoised with FISTA [BT09]

variation inpainting. Finally we provide a source code for the a
er of a certain energy functional. A method is local if the infor
mation from the known part of the image, usually weighted by
that is, when starting with infinitely many particles, the process i
ng limit in some suitable sense. Our limiting object will be a coale
some disadvantages. Image inpainting methods can be roughly si
antages. Image inpainting methods can be roughly separated in
discovered in Mathematics. Actually there is still a lot to be discov
s a mental picture of what they look like. Some people imagine
No, I do not do a PhD because I want to become a high school
the point that is to be filled in. The latter class of methods is,
ing methods take into account all the information from the k
the image.

(e) Inpainting mask



(f) Inpainted image with [FFA15]

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

We estimate the unknown original image \mathbf{I} as the image $\widehat{\mathbf{I}}$ that is more likely to occur. Applying Bayes' theorem we obtain

$$\widehat{\mathbf{I}} = \arg \max_{\mathbf{C}} Pr(\mathbf{C} | \widetilde{\mathbf{I}}) = \arg \max_{\mathbf{C}} \frac{Pr(\widetilde{\mathbf{I}} | \mathbf{C}) Pr(\mathbf{C})}{Pr(\widetilde{\mathbf{I}})}. \quad (1.4)$$

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

$$Pr(\widetilde{\mathbf{I}} | \mathbf{C}) = Pr(\mathbf{N} = \widetilde{\mathbf{I}} - \mathbf{C}) = \frac{1}{\sqrt{2\pi}} \exp \left(- \frac{\lambda \|\widetilde{\mathbf{I}} - \mathbf{C}\|^2}{2} \right). \quad (1.5)$$

The denominator term is computed as the joint probability

$$Pr(\widetilde{\mathbf{I}}) = \sum_{\mathbf{J} \in \mathbb{F}^{m \times n}} Pr(\widetilde{\mathbf{I}} | \mathbf{J}) Pr(\mathbf{J}) = \frac{1}{\sqrt{2\pi}} \sum_{\mathbf{J} \in \mathbb{F}^{m \times n}} \exp \left(- \frac{\lambda}{2} \|\widetilde{\mathbf{I}} - \mathbf{J}\|^2 - \rho(\mathbf{J}) \right). \quad (1.6)$$

Substituting [Equations \(1.5\)](#) and [\(1.6\)](#) in [Equation \(1.4\)](#) we obtain

$$\widehat{\mathbf{I}} = \arg \max_{\mathbf{C}} \frac{1}{\sqrt{2\pi}} \frac{\exp \left(- \frac{\lambda}{2} \|\widetilde{\mathbf{I}} - \mathbf{C}\|^2 - \rho(\mathbf{C}) \right)}{\sum_{\mathbf{J} \in \mathbb{F}^{m \times n}} \exp \left(- \frac{\lambda}{2} \|\widetilde{\mathbf{I}} - \mathbf{J}\|^2 - \rho(\mathbf{J}) \right)} \quad (1.7)$$

Finnaly, solving [Equation \(1.7\)](#) is equivalent to solve

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

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.

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

$$f_{\tilde{I}} = \arg \min_f F(f) = \frac{\lambda}{2} \int_{\Omega} \|f_{\tilde{I}} - f\|^2 dx + R(f), \quad (1.9)$$

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

$$f_{\tilde{I}} = \arg \min_f F(f) = \frac{\lambda}{2} \int_{\Omega} \|f_{\tilde{I}} - f\|^2 dx + \int_{\Omega} \|\nabla f\|^2 dx. \quad (1.10)$$

Euler-Lagrange equation

We can establish some necessary optimization 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,

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

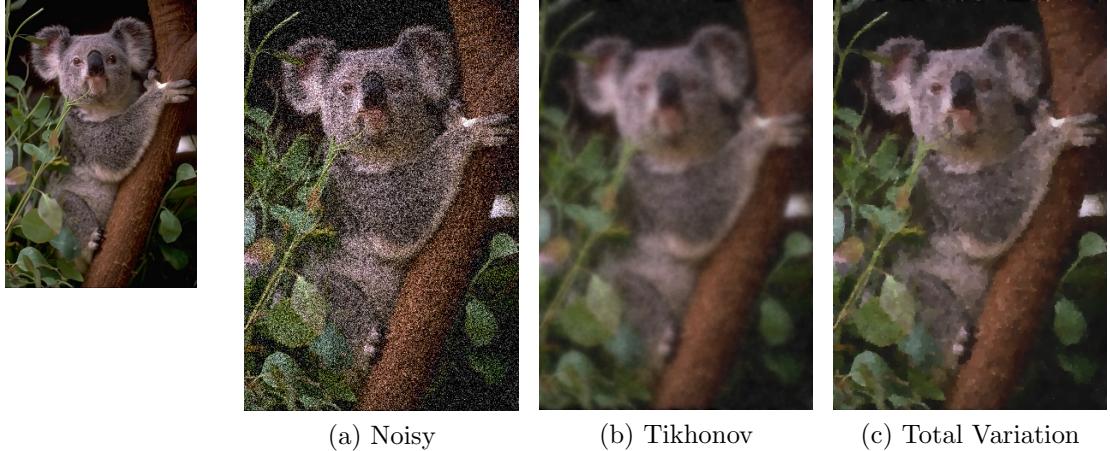


Figure 1.3: **Tikhonov x Total variation denoising.** Total variation is capable to better preserve discontinuities across edges than Tikhonov.

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

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

Since w could be any function, we can write

$$\lambda(f_{\tilde{I}} - g) - \Delta g = 0 \quad (1.11)$$

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 $\hat{\mathbf{I}}, \tilde{\mathbf{I}}$ of $f_{\hat{\mathbf{I}}}, f_{\tilde{\mathbf{I}}}$ 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.

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 $f : \Omega \rightarrow \mathbb{R}^n$, 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

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

The ROF model [ROF92] assumes the image representation is smooth, and from the Euler-Lagrange equation of [Equation \(1.12\)](#) the following gradient flow is derived:

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

The left term is not differentiable, and a small ϵ 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 [Cha04] the total variation definition is exploited to create a convergent algorithm that works by successive projections and that solves model [Equation \(1.12\)](#). In [BT09] 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 is not sufficient to avoid blurred edges. The *Mumford-Shah* functional [MS89] 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., $\Omega \setminus \mathcal{K}$. Moreover, the set \mathcal{K} itself is compelled to be of small length. The Mumford-Shah model consists in minimizing the following functional

$$(f_{\tilde{\mathbf{I}}}, \hat{\mathcal{K}}) = \arg \min_{f, \mathcal{K}} \alpha \int_{\Omega} \|f_{\tilde{\mathbf{I}}} - f\| dx + \beta \int_{\Omega \setminus \mathcal{K}} \|\nabla f\| dx + \lambda Per(\mathcal{K}) \quad (1.13)$$

The functional can be seen as a model for both denoising and segmentation problems. The function $f_{\tilde{\mathbf{I}}}$ being the denoising solution and $\hat{\mathcal{K}}$ the segmentation solution. [Equation \(1.13\)](#) is proven to have a minimizer [DCL89], and in the case \mathcal{K} is fixed, the minimizer is unique (see chapter 25 of [Bar+11]). 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 [AT90]; the finite-differences scheme of [Cha99]; the level-set method of [VC02]; the convex relaxations of [Poc+09; SC14]; and the discrete calculus approach of [FLT17].

1.3.1 Curve evolution

Active contours

Active contours or snakes is a supervised method for doing image segmentation. In the original work [KWT88], an initial parametric curve $C_0(q) \rightarrow (x(q), y(q))$ is evolved towards the local minimum of the snakes energy

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

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 [TD96]. Variations of the original model include extension to 3D-segmentation [MT99] and topologically adaptable snakes [MT95].

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 [Cas+93] the authors propose a model based on the mean curvature motion of the level-sets of a C^2 function u .

Let $u : \Omega \subset \mathbb{R}^2 \rightarrow [0, 1]$ 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 into evolving an extension of u , by including an artificial time parameter t , and compute the steady solution of the flow

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

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.

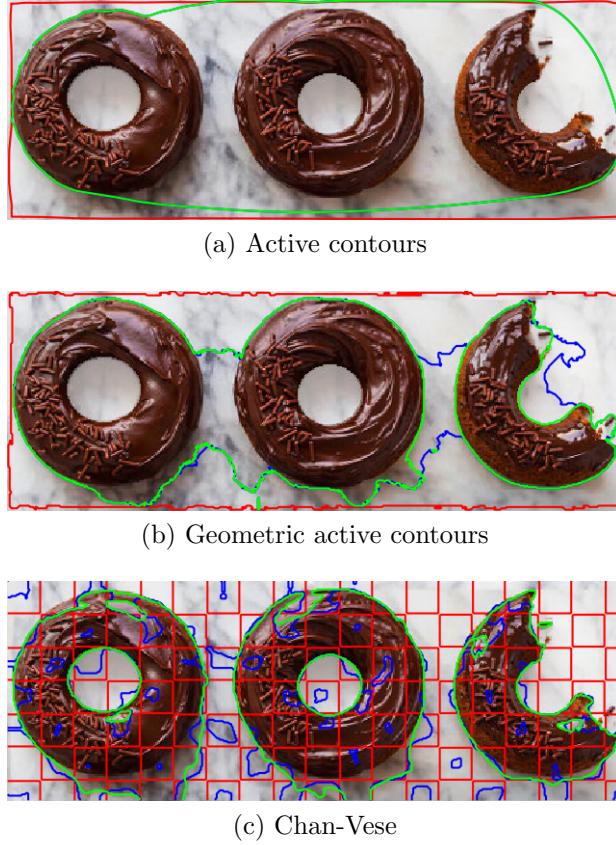


Figure 1.4: **Curve evolution models.** Segmentation results of three curve evolution models. From top row to bottom: active contours, geometric active contours and Chan-Vese. The initial curve (0-level set) is colored in red and the final one is colored in green.

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 $g(\|\nabla f_I\|)$ makes the evolution to stop if an edge is reached. At the steady solution of [Equation \(1.15\)](#) the segmented objects of 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 [[Che+06](#)].

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 $f_I : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{U}^2$ 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 $\phi : \Omega \rightarrow \mathbb{R}$ with bounded first derivative. The image partitions are 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

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

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

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

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

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

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](#). In order to be numerically tractable, the Heaviside and Dirac delta function are regularized as

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

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 [[Get12](#)] to have fast convergence. In [[VC02](#)], 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 [[Cas+93](#)] and the computation of geodesics in a regular surface [[CKS97](#)].

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

$$L(C) = \int W(C) \|C_q\| dq. \quad (1.17)$$

[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_I to match the curves of shortest length, for example, letting $W = g(\|\nabla f_I\|)$ as in [Equation \(1.15\)](#) we obtain

$$L(C) = \int g(\|\nabla f_I(C(q))\|) \|C_q\| dq. \quad (1.18)$$

In [[CKS97](#)] 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

$$C(0, q) = C_0(q) \quad (1.19)$$

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

where \mathbf{n} is the normal vector to the curve C at (t, q) . One can show that, given an initial function $u \in C^2$ 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

$$u(0, x, y) = u_0(x, y) \quad (1.21)$$

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

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 [CK97]. Further development of this work reduced the initialization to just a single point [BC09]. Anisotropic metrics aligned to the image edges are reported to return improved solutions for blood vessels segmentation [Jba+08; BC11]. Ideas from Chan-Vese and Geodesic models are put together in [Che+06]. Finally, an elucidating review of geodesic methods in computer vision can be found in [Pey+10].

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 $f : \Omega \rightarrow \mathbb{R}$ 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 \tilde{f} such that $\tilde{f} \leq f$. In fact, one can compute the envelope of a function f by taking its convex biconjugate.

Definition 1(Convex conjugate): Let $f : \Omega \rightarrow \mathbb{R} \cup \{+\infty, -\infty\}$. 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 [Cha04; BT09].

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 $f_{\tilde{\mathbf{I}}}$ the observed noisy image and consider the total variation model for binary denoising

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

This model is not convex, as the optimization variable $f_{\mathbf{I}}$ 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

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

Whose a local minimum is a steady solution of

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

We reproduce the example given in [CEN06] 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., $f_{\tilde{I}}(x) = \mathbf{1}_{B_R}(x)$. Moreover, we set the initial guess of the level-set function to be precisely the observed image, i.e., $\phi_0(x) = f_{\tilde{I}}(x)$. 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

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

[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 $g(0, \lambda) < g(R, \lambda)$.

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

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 to segment only the ball of largest radius in the picture, but Chan-Vese fails to converge to this solution.

In [CEN06] 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

$$f_I(x) = \int_0^1 \varphi(x, \mu) d\mu,$$

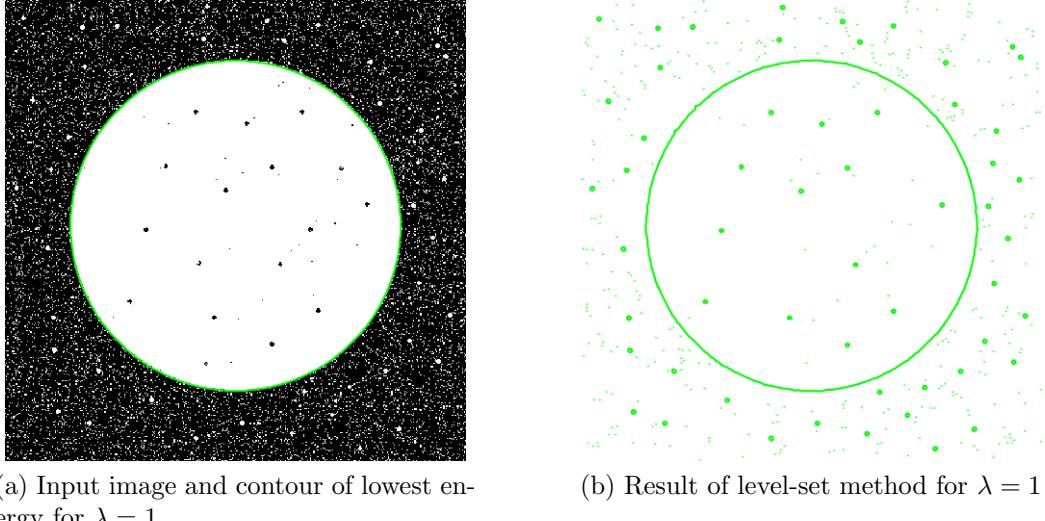
(a) Input image and contour of lowest energy for $\lambda = 1$ (b) Result of level-set method for $\lambda = 1$

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.

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

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

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

$$\int_{\Omega} \|\nabla f_I\| = \int_{\Omega} \int_0^1 \|\nabla \varphi(x, \mu)\| dxd\mu$$

and we can rewrite Equation (1.23) as

$$\min_{\varphi: \Omega \rightarrow \{0,1\}} E^{2-den} = \min_{\varphi} \int_{\Omega} \int_0^1 \|\varphi(x, \mu)\| + (\mu - f_{\tilde{I}}(x))^2 \delta(f_I(x) - \mu) dxd\mu \quad (1.27)$$

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

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

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

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

$$\begin{aligned} E^{2-den} &= \int_{\Sigma} \int_0^1 \|\mathbf{1}_{\{\varphi>\gamma\}}\| + (\mu - f_{\tilde{I}}(x))^2 |\partial_{\mu} \mathbf{1}_{\{\varphi>\gamma\}}| d\Sigma d\gamma \\ E^{2-den} &= \int_0^1 E^{2-den}(\mathbf{1}_{\{\varphi>\gamma\}}) d\gamma \end{aligned}$$

Therefore, if φ^* is the solution of the convex relaxed problem [Equation \(1.29\)](#), $\mathbf{1}_{\{\varphi^*>\gamma\}}$ 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 [\[Poc+08\]](#) this strategy is extended for multilabeling problems and in [\[Poc+09; SCC12\]](#) 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 [\[Cha04\]](#), FISTA [\[BT09\]](#) or the primal-dual [\[CP11\]](#) algorithm. The results are very satisfying, 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 conception 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 on 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 $v \in \mathcal{V}$ 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 $\text{cut}(S, Q)$ 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 \mathbf{X} and we write \mathbf{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 $W_{\mathbf{X}}$ the set of all configurations for the random vector \mathbf{X} . We say that $\mathcal{H} = (\mathcal{G}, \mathbf{X}, \Gamma, P)$ 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

$$\text{Pairwise independencies: } \left\{ X_u \perp X_v \mid \{X_i, \forall i \in \mathcal{V} \setminus \{u, v\}\} \right\} \quad (2.1)$$

$$\text{Local independencies: } \left\{ X_u \perp X_v \mid \{X_i, \forall i \in \mathcal{N}(u)\} \right\} \quad (2.2)$$

$$\text{Global independencies: } \left\{ \mathbf{X}_S \perp \mathbf{X}_Q \mid \mathbf{X}_Z, \text{ where } \mathbf{X}_Z \in \text{cut}(S, Q) \right\}, \quad (2.3)$$

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

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

where we use the shorter notation $\mathbf{X}_{C_{123}} = \mathbf{w}$ 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.

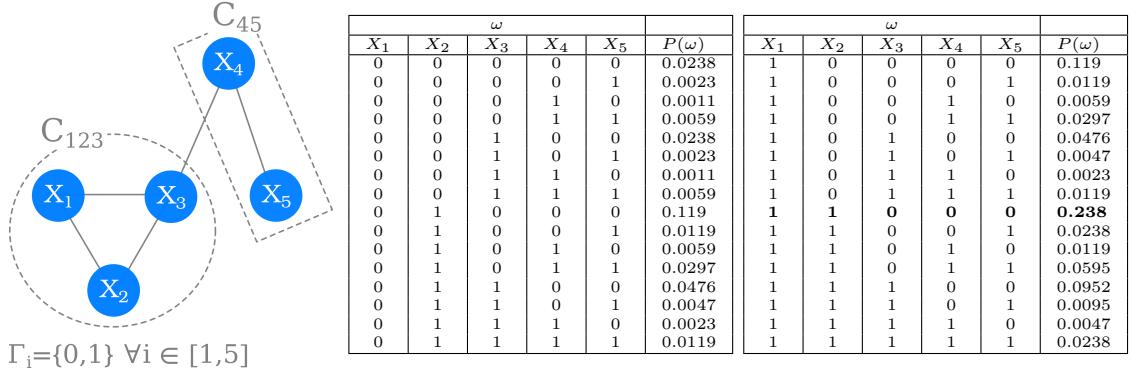


Figure 2.1: **Example of a Markov Random Field.** The nodes X_1, X_2, X_3 forms the 3-clique C_{123} .

Clique factorization and Gibbs energy

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

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

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

For strictly positive distributions ($P(\mathbf{X} = \mathbf{w}) > 0 \quad \forall \mathbf{w} \in W_{\mathbf{X}}$), the *Hammersley-Clifford theorem* [KF09] states that $(\mathcal{G}, \mathbf{X}, \Gamma, P)$ 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

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

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

where \mathcal{C} is the set of all cliques of \mathcal{G} . We define the *order* of such MRF as $\max_{C \in \mathcal{C}} |C| - 1$, 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 [\[KF09\]](#).

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 $\mathcal{H} = (\mathcal{G}, \mathbf{X} \cup \mathbf{Y}, \Gamma_X \cup \Gamma_Y, P)$ such that

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

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

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} = P(\mathbf{X} = \mathbf{w} \mid \mathbf{Y}). \quad (2.6)$$

The vector \mathbf{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 $\Gamma_{\mathbf{X}}$ 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

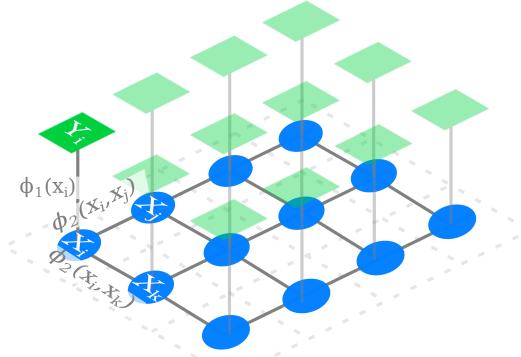


Figure 2.2: **HMM and grid graphs.** A typical HMM in imaging using a 4-neighborhood system.

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.

Grid graph and Tikhonov denoising revisited

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

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

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 $\mathcal{H} = (\mathcal{G}_I, \mathbf{X} \cup \mathbf{Y}, \Gamma_{\mathbf{X}}, \Gamma_{\mathbf{Y}}, P)$ in which the random variables take values over the grayscale levels of the image, grouped in $\Gamma_{\mathbf{X}} = \Gamma_{\mathbf{Y}} = \{i \mid 0 \leq i \leq 255\}$. The Gibbs distribution P is given by Equations (2.4) and (2.5) with clique potentials defined as

$$\begin{aligned}\psi_p(\gamma_p) &= \frac{\lambda}{2} (f_{\tilde{\mathbf{I}}}(p) - \gamma_p)^2, \\ \psi_{pq}(\gamma_p, \gamma_q) &= (\gamma_p - \gamma_q)^2.\end{aligned}$$

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

In ?? we have a representation of this HMM, which encodes the Tikhonov denoising model of [Section 1.2](#). The estimated image $\hat{\mathbf{I}}$ is computed as the maximum likelihood of P , i.e.,

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

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 [[Ish03](#)]. However, MAP inference in general multilabel HMMs is in NP-hard. The scenario is a little bit better for binary HMMs, as we are going to see in the next section.

Potts and Ising model

Let $\mathbf{I} \in \mathbb{F}^{m \times n}$ a grayscale image to which we associated its grid graph $\mathcal{G}_I(\mathcal{V}, \mathcal{E})$. We wish to denoise image \mathbf{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

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

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

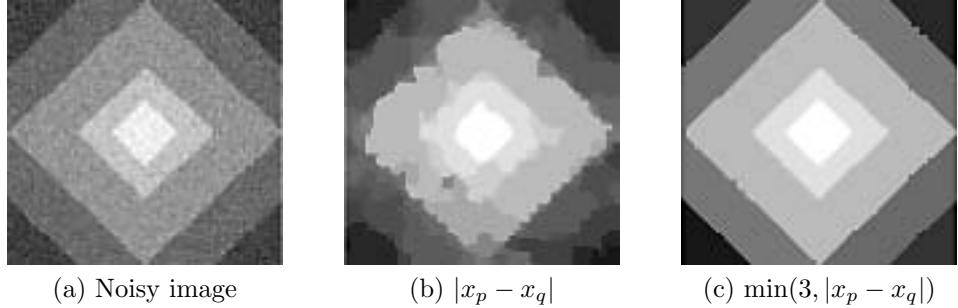


Figure 2.3: **Comparison between convex and truncated potentials** [BKR11](chapter 3). Truncated potentials are discontinuity preserving functions, and are suitable for imaging problems.

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

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

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

Equations (2.9) and (2.10) defines a *Potts model*. Minimizing Equation (2.10) is in NP-hard [BVZ01] 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 $\log_2 |\Gamma_{\mathbf{x}}|$ new variables. The difficulty is that the resulting energy is very likely to lie in a class of binary energies which its minimization is in NP-hard [Ram+08]. Therefore, MAP inference in multilabel HMMs is not likely to be solved exactly in an efficient manner. Instead, approximation algorithms are used (we are going to discuss some of them in Section 2.4).

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

MRF Order	Graph topology	Function class	
		Submodular	Non-submodular
0^{th} order	All topologies	Greedy algorithm	
1^{st} order	1-connected	Viterbi algorithm [Vit67], Belief Propagation [Pea82]	
	≥ 2 -connected	Max-Flow: $O(V ^2 E)$ [KZ04]	
2^{nd} order	All topologies	Max-Flow: $O(V ^2 E)$ [BM85]	NP-
	Higher order	$O(n^5Q + n^6)$ [Orl09]	Hard [NW81]

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 contrast 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 [BH02]

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 auxiliar variables. Naturally, such transformations are not always useful. In the multilabel case the derived energy is almost sure non-submodular and in the case of high-order cliques, the inclusion of auxiliar variables may turn the minimization problem impractical [Ish10].

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

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

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 $f : \{0, 1\}^n \rightarrow \mathbb{R}$ 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 bijec-

tion with the power sets of $V = \{1, 2, \dots, n\}$. Therefore, the PBF $f : \{0, 1\}^n \rightarrow \mathbb{R}$ can be written as

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

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 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.

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

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 polynome 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 ϕ^* . 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.

$$\begin{aligned} f(x_1, x_2) &= a_\emptyset + a_{\bar{1}} + a_{\bar{2}} + a_{\bar{1}\bar{2}} + (a_1 - a_{\bar{1}} + a_{1\bar{2}} - a_{\bar{1}\bar{2}})x_1 \\ &\quad + (a_2 - a_{\bar{2}} + a_{\bar{1}2} - a_{\bar{1}\bar{2}})x_2 \\ &\quad + (a_{12} - a_{\bar{1}2} - a_{1\bar{2}} + a_{\bar{1}\bar{2}})x_1x_2 \end{aligned}$$

$$\begin{aligned} \phi_1(x_1, x_2) &= a_\emptyset + a_1x_1 + a_{\bar{1}}\bar{x}_1 + a_2x_2 + a_{\bar{2}}\bar{x}_2 \\ &\quad + a_{12}x_1x_2 + a_{\bar{1}2}\bar{x}_1x_2 + a_{1\bar{2}}x_1\bar{x}_2 + a_{\bar{1}\bar{2}}x_1\bar{x}_2 \end{aligned}$$

$$\begin{aligned} \phi_2(x_1, x_2) &= (a_\emptyset - c) + (a_1 + c)x_1 + (a_{\bar{1}} + c)\bar{x}_1 + a_2x_2 + a_{\bar{2}}\bar{x}_2 \\ &\quad + a_{12}x_1x_2 + a_{\bar{1}2}\bar{x}_1x_2 + a_{1\bar{2}}x_1\bar{x}_2 + a_{\bar{1}\bar{2}}x_1\bar{x}_2 \end{aligned}$$

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.

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 $\min f = \min g$ [BH02]. We remark, however, that the transformation may add a prohibitive number of auxiliar 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

$$F(\mathbf{x}) = \begin{cases} \min & c_0 + \sum c_i x_i + \sum c_{ij} x_i x_j \\ \text{subject to} & \mathbf{x} \in \{0, 1\}^n. \end{cases} \quad (2.13)$$

[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

$$C(\mathbf{x}, \mathbf{z}) = \left\{ \begin{array}{l} z_{ij} \leq x_i, \\ z_{ij} \leq x_j, \\ z_{ij} \geq x_i + x_j - 1 \end{array} \mid \forall 0 < i < j < n \right\}$$

Therefore, an equivalent linear integer programming formulation is

$$\begin{aligned} \min \quad & c_0 + \sum c_i x_i + \sum c_{ij} z_{ij} \\ \text{subject to} \quad & \mathbf{x}, \mathbf{z} \in \{0, 1\}^n \\ & C(\mathbf{x}, \mathbf{z}) \end{aligned} \tag{2.14}$$

Finally, the relaxation of Equation (2.14) gives

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

Clearly, formulation G is a lower bound of F , i.e., $G(\mathbf{x}, \mathbf{z}) \leq F(\mathbf{x})$. Such lower bound is called the *roof dual* (it was originally defined for a maximization problem) and its shown [HHS84] to be equivalent to

$$G(\mathbf{x}, \mathbf{z}) = C_{\phi^*(f)},$$

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

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

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 $\phi^*(f)$ that results in a maximum number of variable elimination. Such posiform, called the *master posiform*, can be found by finding the maximum flow in some capacitated graph.

Master posiform and max-flow reduction

We present a construction given in [BHS91; BH02] of a capacitated graph $G_\phi(\mathcal{V}, \mathcal{E}, c)$ that encodes some posiform ϕ with constant $C(\phi) = 0$. The authors shown 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 $(\overrightarrow{v_p, v_{\bar{q}}})$ and $(\overrightarrow{v_q, v_{\bar{p}}})$. The full graph is given by

$$\begin{aligned}
\mathcal{V} &= \{v_p \mid p \in L\} \cup \{s, \bar{s}\} \\
\mathcal{E} &= \{(\overrightarrow{s, v_{\bar{p}}}), (\overrightarrow{v_p, t}) \mid \forall a_p > 0\} \cup \{(\overrightarrow{v_p, v_{\bar{q}}}), (\overrightarrow{v_q, v_{\bar{p}}}) \mid \forall a_{pq} > 0\} \\
c((v_p, v_q)) &= c_{pq} = \begin{cases} a_p/2, & \text{if } v_p \notin \{s, \bar{s}\} \text{ or } v_q \notin \{s, \bar{s}\} \\ a_{pq}/2, & \text{if } v_p, v_q \notin \{s, \bar{s}\} \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

A construction of graph G_ϕ is illustrated in [Figure 2.4a](#). Therefore, the posiform ϕ can also be written as $\phi = \sum_{(\overrightarrow{v_p}, \overrightarrow{v_q}) \in \mathcal{E}} c_{pq}$. 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 $\varphi : \mathcal{E} \rightarrow \mathbb{R}_+$ that satisfies

$$\varphi((\overrightarrow{v_p}, \overrightarrow{v_q})) = \varphi(p, q) < c_{pq}, \quad \forall v_p, v_q \in \mathcal{V}$$

$$\sum_{v_p \in \mathcal{V}} \varphi(p, q) = \sum_{v_p \in \mathcal{V}} \varphi(q, p), \quad \forall v_q \in \mathcal{V}.$$

A flow φ^* is said to be maximum if

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

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.,

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

The edges cost is given by the residual function r

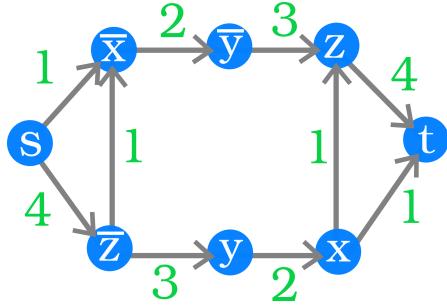
$$r((\overrightarrow{v_p}, \overrightarrow{v_q})) = r_{pq} = \begin{cases} c_{pq} - \varphi(p, q), & (\overrightarrow{v_p}, \overrightarrow{v_q}) \in \mathcal{E} \\ \varphi(p, q), & (\overrightarrow{v_p}, \overrightarrow{v_q}) \in \mathcal{E}^+ \setminus \mathcal{E}. \end{cases}$$

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, $(\overrightarrow{p}, \overrightarrow{s})$ is encoded as $p\bar{s} = 0$.

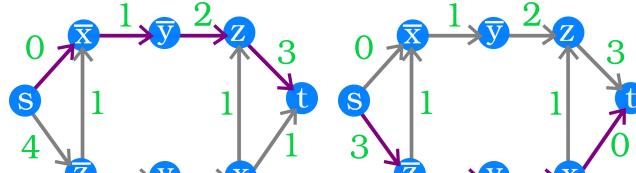
The *Ford-Fulkerson* algorithm computes the maximum flow by incrementing an initial zero flow function $\varphi_0 = 0$ every time an *augmenting path* is found. The k -th augmenting path is a path $\pi_k = (p_0 = s, p_1, p_2, \dots, p_n, p_{n+1} = \bar{s})$ in the residual graph $G_{\phi[\varphi_{k-1}]}$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

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

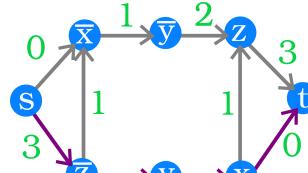
Proposition 1(Residual graph to posiform): Given a posiform



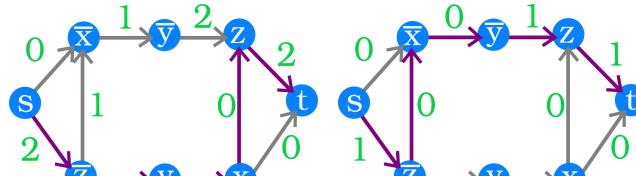
(a) Initial graph G_ϕ for $\phi = 2x + 8z + 2x\bar{z} + 4\bar{x}y + 6\bar{y}\bar{z}$



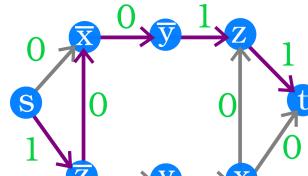
(b) 1-Augmenting path 1



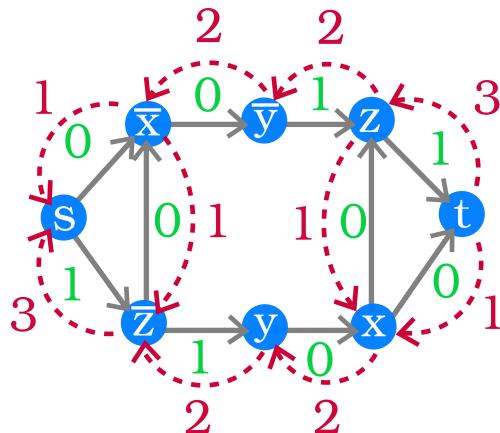
(c) 1-Augmenting path 2



(d) 1-Augmenting path 3



(e) 1-Augmenting path 4



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

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).

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

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 $\pi_k = s, v_{p_1}, v_{p_2}, \dots, v_{p_n}, \bar{s}$ in $G_{\phi[\varphi_k]}$ encodes an alternating sum of literals of the form

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

where

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

By observing that

$$\begin{aligned} \bar{p}_1 + p_1 \bar{p}_2 &= 1 - p_1 p_2 \\ -\bar{p}_{j-1} p_j + p_j \bar{p}_{j+1} &= \bar{p}_{j-1} p_j - p_j p_{j+1} \\ -p_{n-1} p_n + p_n &= \bar{p}_{n-1} p_n, \end{aligned}$$

we can rewrite the alternating sum as

$$\begin{aligned} \phi_\pi &= \epsilon + \epsilon (\bar{p}_1 p_2 + \bar{p}_2 p_3 + \dots + \bar{p}_{n-1} p_n) + \phi' \\ &= \psi + \phi', \end{aligned}$$

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

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

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 $\phi^* = \nu(\varphi^*) + \phi_{G_{\phi[\varphi^*]}} = 4 + 2z + 2\bar{z}\bar{y} + 4zy + 4\bar{y}x + 2z\bar{x}$.

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

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

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 $\mathbf{x} \in \arg \min f$ must agree with $x(U = 1)$ [\[BH02\]](#). Therefore,

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

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 that finds partial solutions of general quadratic PBF. In the most cases, the master posiform allows us to eliminate only few variables, but if the function f is a *submodular* function, the QPBO finds the minimum of f .

2.3 Submodular functions

Definition 1(Submodular set function):

Let V be a set with n elements, e.g., $V = \{1, 2, \dots, n\}$. A set function $f : 2^V \rightarrow \mathbb{R}$ is submodular if

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

An equivalent local definition is given by

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

Proposition 1(Quadratic submodular PBF): Let $f : 2^V \rightarrow \mathbb{R}$ a quadratic PBF written as

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

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 \rightarrow ii): Immediately from [Equation \(2.16\)](#).
- (ii \rightarrow 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 \rightarrow i): We define

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

Therefore

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

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 can minimize submodular and maximize supermodular functions, the opposite problems being in NP -hard.

There are functions which are not submodular neither supermodular and are called *non-submodular*. Optimizing a non-submodular function is in NP -hard but QPBO can be used to find a partial solution. However, it is likely that at the end of the flow computation, several variables would be unlabeled. Some variations includes the QPBO-Probe, which attempts to increase the number of variables labeled by QPBO using a branch-bound technique and the heuristic QPBO-Improve, both proposed in [Rot+07]. The first is guaranteed to return a partial solution, likely with more labels than the one given by QPBO. The latter does not guarantee a partial solution, but the final labeling is guaranteed to have a value as lower as the 0, 1-extended partial solution returned by QPBO.

As discussed in Sections 2.1 and 2.2, several problems in image processing can be modeled in terms of a 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 \mathbf{Y} . In that occasion, we mention that such estimation were dependent of our capacity to optimize a pseudo-boolean function. As we have seen, the optimization of a PBF is closely related with the

computation of maximum flows, or equivalently, minimum cuts in a graph. This point of view is quite enlightening, as we can interpret problems in imaging with the computation of minimum cuts in its grid graph. In the next chapter we present several models inspired in this idea.

2.4 Graph-cut models

Grabcut: Segmentation via graph cuts

In [BJ01] is proposed a model based on the computation of the minimum cut of a graph to segment an image in foreground and background components. It is a binary labeling problem. We present here its formulation as an HMM.

Let $I \in \mathbb{F}^{m \times n}$ a discrete grayscale image. We construct the HMM $\mathcal{H} = (\mathcal{G}_I, \mathbf{X} \cup \mathbf{Y}, \Gamma_{\mathbf{X}} = \{0, 1\}, \Gamma_{\mathbf{Y}} = \{0, \dots, 255\}, P)$ where \mathcal{G}_I is the grid graph of image I . We wish to estimate the hidden states \mathbf{X} among its possible labels $\Gamma_{\mathbf{X}}$, i.e., 0 for background and 1 for foreground, given a set of observed variables \mathbf{Y} , the color intensity of image pixels. We assume that variables in \mathbf{X} and \mathbf{Y} are indexed by image pixels, i.e., \mathbf{X}_p is the random variable corresponding to the pixel p . The probability distribution P is described in terms of clique potentials, described next.

We assume that we are given sets $S_{fg}, S_{bg} \subset I$ representing pixels for which a label is already known. Pixels in $S_{fg}(S_{bg})$ are labeled as foreground (background). We compute two mixed Gaussian distributions H_{fg}, H_{bg} that best fits the color intensities in the sets S_{fg} and S_{bg} , respectively. These distributions are used to define the 1-clique potential

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

$$\phi_p(x_p) = \exp(-\psi_1(x_p)), \quad \forall p \in \Omega \text{ and } p \notin S_{fg} \cup S_{bg}.$$

A higher probability in the foreground (background) distribution reflects in a higher clique potential if $x_p = 1$ ($x_p = 0$). The 2-clique potential encodes the spatial coherence assumption in images, i.e., neighboring pixels tends to have similar intensity values

$$\psi_2(x_p, x_q) = \exp\left(-\frac{\lambda}{d(p, q)} \frac{(I(p) - I(q))^2}{2\sigma^2}\right).$$

$$\phi_{pq}(p, q) = \exp(-\psi_2(x_p, x_q)), \quad \forall p \in \Omega \text{ and } q \in \mathcal{N}(p).$$

where $d(p, q)$ is the Euclidean distance between the pixel coordinates of p, q and λ is a parameter to control the influence between clique potentials ϕ_p and ϕ_{pq} . The Gibbs distribution writes as

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x})$$

The maximum likelihood is computed by minimizing the Gibbs energy E .

$$\begin{aligned} \arg \max_{\mathbf{x}} P(\mathbf{X} = \mathbf{x} \mid \mathbf{Y}) &= \arg \min_{\mathbf{x}} E(\mathbf{x}) \\ &= \arg \min_{\mathbf{x}} \sum_{v_p \in \mathcal{V}} \psi_1(x_p) + \sum_{(v_p, v_q) \in \mathcal{E}} \psi_2(x_p, x_q) \\ &= \arg \min_{\mathbf{x}} \sum_{v_p \in \mathcal{V}} \left(\psi_1(0)\bar{x}_p + \psi_1(1)x_p \right) + \frac{1}{2} \sum_{(v_p, v_q) \in \mathcal{E}} \psi_2(0, 1)\bar{x}_p x_q. \end{aligned} \tag{2.17}$$

Equation (2.17) is a submodular function and it can be minimized using one of the variations of the QPBO algorithm.

We could also take a simpler point of view by working solely with an extended version of the grid graph \mathcal{G}_I . Consider the extended grid graph $\mathcal{G}_{I+}(\mathcal{V}^+, \mathcal{E}^+, c)$ defined as

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

edge e	c(e)	for
$\{v_p, v_q\}$	$\psi_2(x_p, x_q)$	$\forall p \in \mathbf{I}$ and $q \in \mathcal{N}_k(p)$
$\{v_p, s\}$	$\psi_1(x_p = 0)$ M 0	$\forall p \in \mathbf{I}$ and $p \notin S_{fg} \cup S_{bg}$ $p \in S_{fg}$ $p \in S_{bg}$
$\{v_p, t\}$	$\psi_1(x_p = 1)$ 0 M	$\forall p \in \mathbf{I}$ and $p \notin S_{fg} \cup S_{bg}$ $p \in S_{fg}$ $p \in S_{bg}$

The source (target) vertex is associated with foreground (background) label. In the extended graph, a minimum (s, t) -cut of \mathcal{G}_{I+} induces a minimum labeling



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

of Equation (2.17). 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 cuts [DB08] 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 [KT05; JB06]. Although the general complexity of these alternatives may be higher than Ford-Fulkerson algorithm, they are reported [Sze+06] to compute minimum cuts in lower time for grid graphs.

Approximating multilabel energies with α -expansion and (α, β) -swap

For pseudo-boolean functions (binary labels), we know that the submodular class is minimized in polynomial time. For energies of order at most three, the problem is reduced to the computation of a minimum cut in a capacitated graph. For higher orders, the best algorithm has complexity $O(n^5Q + n^6)$ [Orl09], where Q is the time to evaluate the function being minimized. If the function is not binary (multilabeling problems), things get more complicated.

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 $\log |\Gamma_{\mathbf{X}}|$ new variables. The difficulty is that the resulting energy is likely non-submodular [Ram+08], and the minimization of a general non-submodular function is in NP -hard. In this section we describe two algorithms to approximate the MAP inference of multilabel problems based on local-search.

A standard approach to devise an approximation solution is the so called *local-search* paradigm described in [Algorithm 1](#). The local-search paradigm finds a local minimum (maximum) with respect to some given neighborhood \mathcal{N} and an initial solution x_0 . The difference between the global and optimum solutions is denoted the *optimality gap*. Naturally, a larger neighborhood tends to decrease the optimality gap and may lead to an increase in the running time. In the extreme case, in which all configurations are present in the neighborhood \mathcal{N} , the local-search returns the global optimum solution. Therefore, a successful local-search algorithm is the one that has a good compromise between neighborhood size and optimality gap.

input : Function f ; an initial solution $x^{(0)}$; and a neighborhood \mathcal{N} .
output: Local minimum solution of f with respect to neighborhood \mathcal{N} and initial solution $x^{(0)}$.

```

1  $t \leftarrow 0;$ 
2 do
3    $x^{(t+1)} = \arg \min_{x \in \mathcal{N}(x^{(t)})} f(x)$  ; // Local optimization step
4    $t \leftarrow t + 1;$ 
5 while  $f(x^{(t)}) < f(x^{(t-1)})$ ;
```

Algorithm 1: Local search paradigm for minimization.

Approximation algorithms like iterated conditional moves (ICM) [[Bes86](#)] and simulated annealing [[GG84](#)] defines the neighborhood $\mathcal{N}(\mathbf{I}^{(t)})$ as the set of images that differs in a single pixel from $\mathcal{N}(\mathbf{I}^{(t)})$. Consequently, a premature stop in a local optimum of lower quality is likely to happen.

The idea in the α -expansion and (α, β) -swap algorithms consists in to define a large neighborhood $\mathcal{N}(\mathbf{I}^{(t)})$ such that finding the best configuration in such neighborhood can be efficiently computed. In order to illustrate some ideas, let's consider once again the Potts model applied to image denoising of a grayscale image $\mathbf{I} \in \mathbb{F}^{m \times n}$.

$$\phi_{pq}(x_p, x_q) = \begin{cases} K, & \text{if } x_p \neq x_q \\ 0, & \text{otherwise.} \end{cases} \quad (2.18)$$

$$E(\mathbf{x}) = \sum_{p \in \mathcal{V}} \phi_p(x_p) + \sum_{(p,q) \in \mathcal{E}} \phi_{pq}(x_p, x_q). \quad (2.19)$$

As mentioned before, minimizing [Equations \(2.18\)](#) and [\(2.19\)](#) is in NP-hard as long as variables x_p, x_q take values in a set of cardinality greater than 2. Notice that if we limit the problem to only two labels, let's say α, β , [Equations \(2.18\)](#) and [\(2.19\)](#)

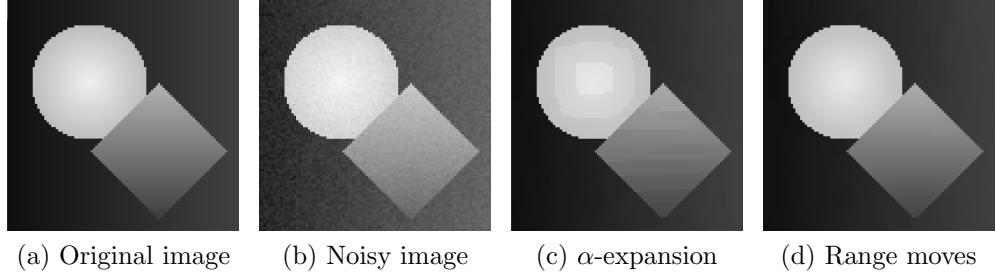


Figure 2.6: **α -expansion x range moves**[Vek07]. Comparison of α -expansion ((α, β)-swap results are similar) and range moves for denoising a piecewise smooth image using the truncated convex potential $\phi_2(x_p, x_q) = \min\{(x_p - x_q)^2, 50\}$. The range move algorithm returns a smoother image than α -expansion.

is a submodular PBF (Ising model) and can be minimized in polynomial time. The key is in to define the transformation function

$$T_{\alpha,\beta}(x_p, c_p) = \begin{cases} \alpha, & \text{if } (x_p = \alpha \text{ or } x_p = \beta) \text{ and } c_p = 0, \\ \beta, & \text{if } (x_p = \alpha \text{ or } x_p = \beta) \text{ and } c_p = 1, \\ x_p, & \text{otherwise.} \end{cases}$$

and the neighborhood

$$\mathcal{N}_{\alpha,\beta}(\mathbf{I}) = \{\mathbf{I}' \mid \mathbf{I}'(p) = T_{\alpha,\beta}(\mathbf{I}(p), \{0, 1\}) \quad \forall p \in \mathbf{I}\}.$$

In the (α, β) -swap algorithm, the local-optimization step (Line 3 in Algorithm 1) is replaced by

```

for  $(\alpha, \beta) \in \{0, 255\}$  do
   $y \leftarrow \arg \min_{y' \in \mathcal{N}_{\alpha,\beta}(x^{(t)})} f(y');$ 
  if  $f(y) < f(x^{(t)})$  then
     $x^{(t)} \leftarrow y;$ 
  end
end
```

The α -expansion works in a similar way, except that the only movement allowed is to change a label to some other label α . In the (α, β) -swap algorithm the local optimization energy is submodular if

$$\phi_{pq}(\alpha, \alpha) + \phi_{pq}(\beta, \beta) \leq \phi_{pq}(\alpha, \beta) + \phi_{pq}(\beta, \alpha) \quad \forall \alpha, \beta \in \Gamma_X. \quad (2.20)$$

In the case of α -expansion, submodularity is ensured if

$$\phi_{pq}(\alpha, \alpha) + \phi_{pq}(\beta, \gamma) \leq \phi_{pq}(\beta, \alpha) + \phi_{pq}(\alpha, \gamma) \quad \forall \alpha, \beta, \gamma \in \Gamma_X. \quad (2.21)$$

The α -expansion and (α, β) -swap algorithms have good overall performance for models based in the Potts potential. In [BVZ01] we can find nice results for stereo. However, they are less appealing if we use a truncated convex 2-clique potential [BKR11](see chapter 11). In this case, the *range moves* algorithm proposed in [Vek07] is reported to deliver better results.

Geodesics computation via graph-cuts

We have seen the natural connection between cuts and binary segmentation describing the Grabcut algorithm. The removal of the minimum cut $C \in \mathcal{E}$ of the Grabcut model partitioned the grid graph in two disjoint connected components, one connected to the source vertex and associated to the foreground and the other connected to the target vertex and associated to the background. The cut C itself can be related with the contour ∂S of the foreground shape (see Figure 2.7). In [BK03] it was shown that one can define a cost function for \mathcal{E} such that the cost of C is arbitrarily close to the length of ∂S .

The key idea is to use the Cauchy-Crofton formula that makes the link between the number of lines intersecting a shape contour and its perimeter. 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 2.7).

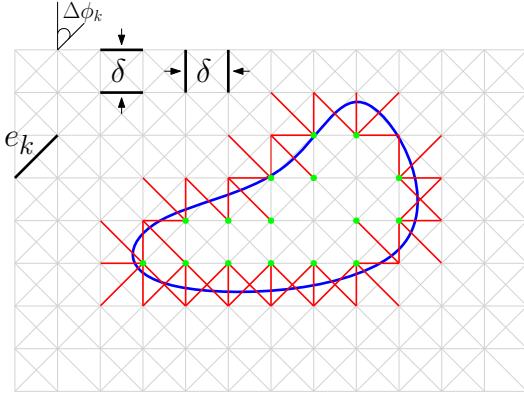


Figure 2.7: **Computing perimeter via graph cuts.** The perimeter of a shape S can be encoded as the value of some cut in the grid graph. In the figure, the contour of shape S intersects a set $C \in \mathcal{E}$ of the grid graph with neighborhood system \mathcal{N}_8 . One can set a cost function on \mathcal{E} such that the cost of C converges to the length of ∂S as the neighborhood system goes to infinity.

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

Another interesting contribution in the attempt to inject geometric information in the graph-cut framework is the connectivity priors of [\[VKR08\]](#). This work provides an additional tool to the Grabcut 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 2.8](#)).

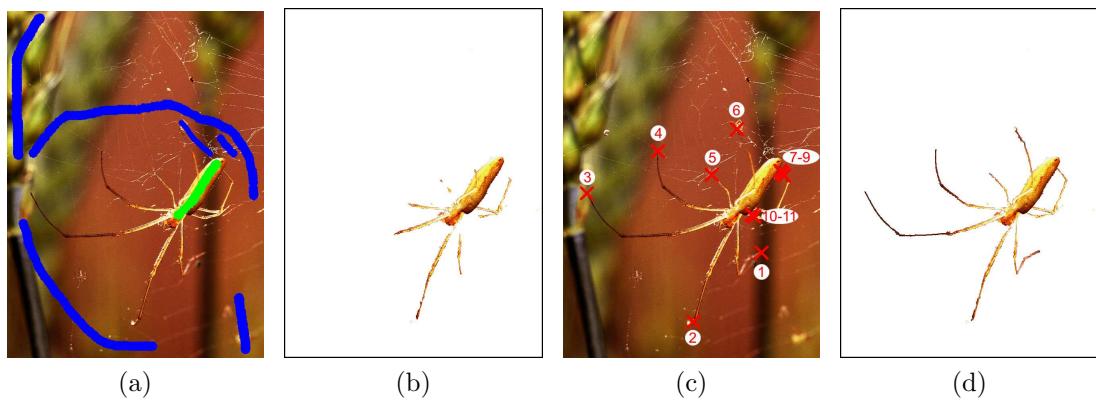


Figure 2.8: **Grabcut with connectivity priors** [VKR08]. In (a) the foreground (green) and background (blue) seeds. In (b) the Grabcut segmentation. In (c) the points selected by the user in which the connectivity constraint is going to be applied. In (d) the final segmentation.

Chapter 3

Curvature as regularization

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. 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. We dedicate Section 3.4 to describe 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

is

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

Definitions

remainder

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

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

such that they are respectively

Let $T(s)$ and $N(s)$ two orthonormal vectors to $C(s)$, i.e., 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).

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

Curve-shortening flow

Let $t \geq 0$ and $u \in [0, U]$, for some $U > 0$. Next, let ~~the~~ $C^{(0)} : [0, U] \rightarrow \mathbb{R}^2$ 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 **at speed** p by an amount v in the normal direction at p . Note that any tangent component is irrelevant to curve deformation. From ?? 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 $dC_t/du = \|C_u\| dC_t/ds$

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

and $N_s = -kT$

Using Equation (3.2) we obtain

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

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

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

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

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

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

Moreover, the curve collapses to a single point in time $t = \frac{R_0^2}{2}$. The curvature flow has many interesting properties [Hui+84; G+86; Eck08]. Among them,

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.
(Note: this is true only in 2D)

Perimeter minimization: The curvature flow is the continuous deformation that decreases the perimeter of a single closed curve the fastest. 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 [OS88]

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

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

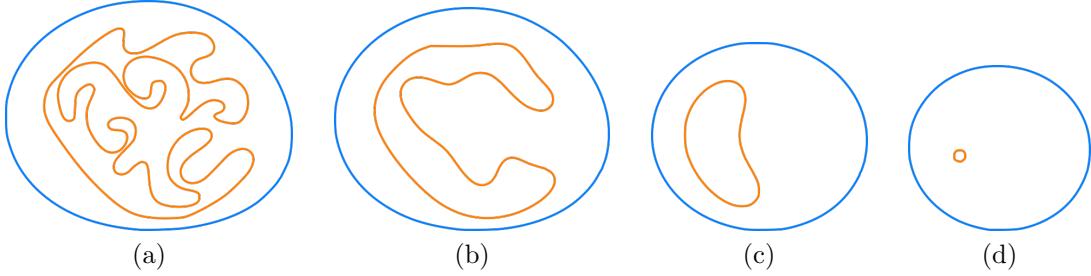


Figure 3.1: **CS flow in action.** CS flow at different times in evolution. In this example we observe the CS flow properties listed in the text.

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 help us to gain extra insight on them.

Curvature in denoising and image segmentation

The necessary optimality condition of the total variation denoising energy is given by
[\(see some ref or see some previous section\)](#)

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

The solution of [Equation \(3.7\)](#) is the steady state of the anisotropic diffusion

$$u(0, x) = f(x) \quad (3.8)$$

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

Note the similarity between [Equation \(3.9\)](#) and the level set formulation of the CS flow in [Equation \(3.6\)](#). If $\lambda \equiv 0$, we denote [Equations \(3.8\)](#) and [\(3.9\)](#) the TV flow[BCN02]. 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 [longer](#) and it decreases the TV by decreasing the surface height, i.e., the color intensity of the pixels(see [Figure 3.2](#)).

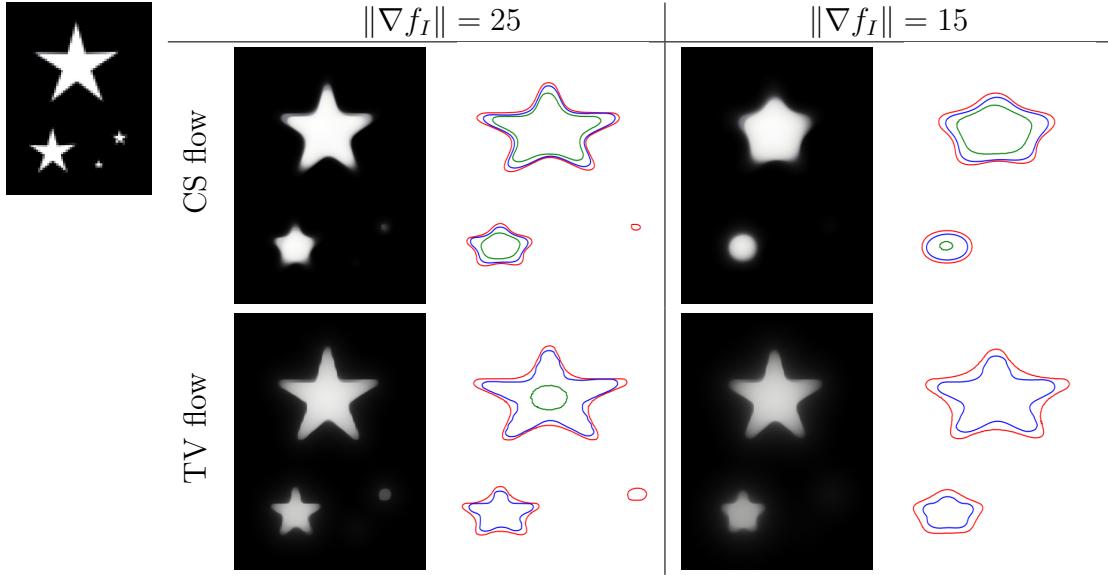


Figure 3.2: **Minimizing TV energy with TV flow and CS flow.** Both flows decrease the image total variation, but CS flow tends to deform object boundaries while doing so. On the other hand, TV flow reduces the image contrast.

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). is it characteristic function or same lambda as above ?

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

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

Curvature and the connectivity principle applied to inpainting

We recall that the inpainting problem consists into reconstructing a collection of missing patches \mathcal{P} of the image. The earlier inpainting model [Ber+00] diffuses image information through the patches boundaries. The particularity of this model is that the Laplacian (heat equation) is diffused along the image level lines. Besides despite

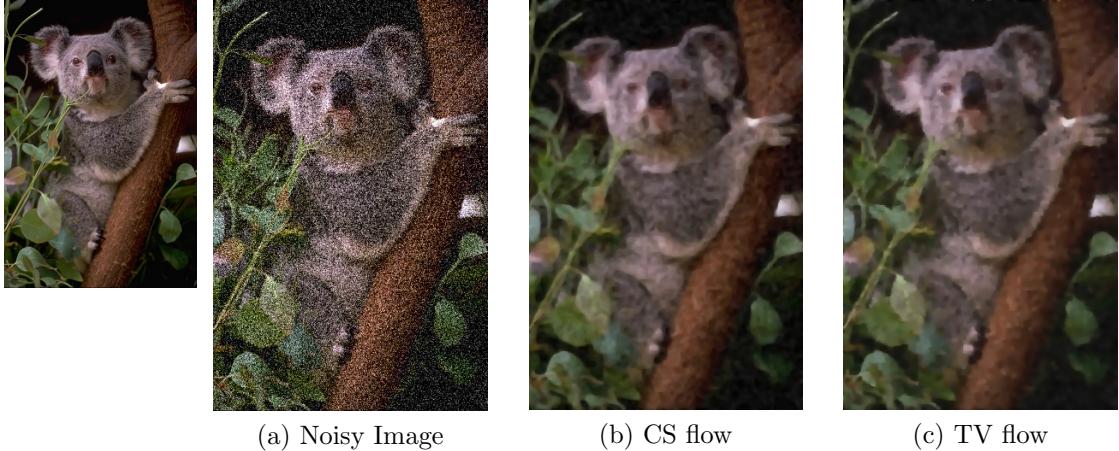


Figure 3.3: TV denoising using CS and TV flow. Results are quite similar, but TV flow image is sharper, although with lower contrast than CS flow. Images (b) and (c) have the same TV value.

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 [Mum94], completes the boundary of partially occluded object with a short and low curvature curve, as illustrated in [Figure 3.4a](#). The model proposed in [CS01] is a TV flow modulated by the curvature of the image level curves.

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

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 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.

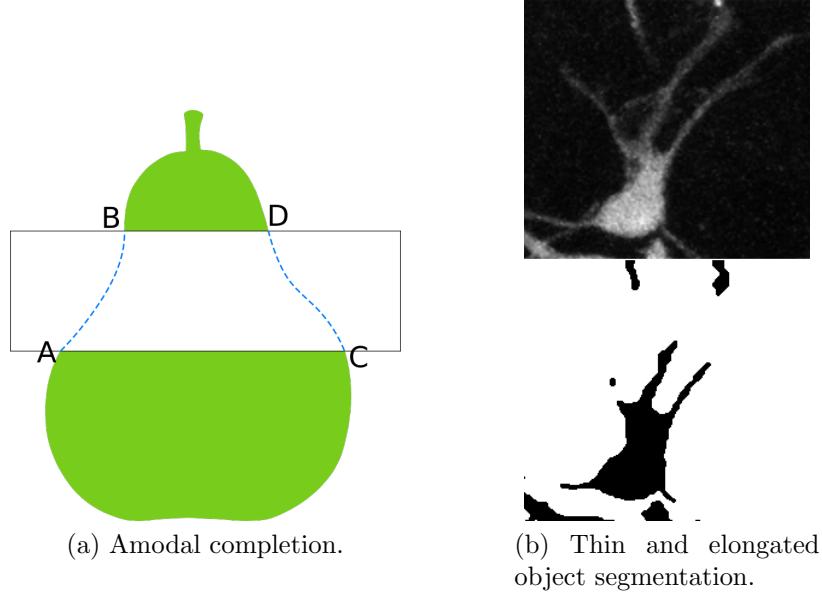


Figure 3.4: **Connectivity principle.** In Figure 3.4a, an illustration of amodal completion, the process followed by the human vision to complete the boundary of occluded object with curves of low curvature [ant](#) that conform the most with the direction of the endpoints. In Figure 3.4b a vessel segmentation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

and



Figure 3.5: **Inpainting by curvature driven diffusions.** [CS01] The diffusion process modeled by [Equation \(3.10\)](#) completes the features disconnected by the inpainting mask.

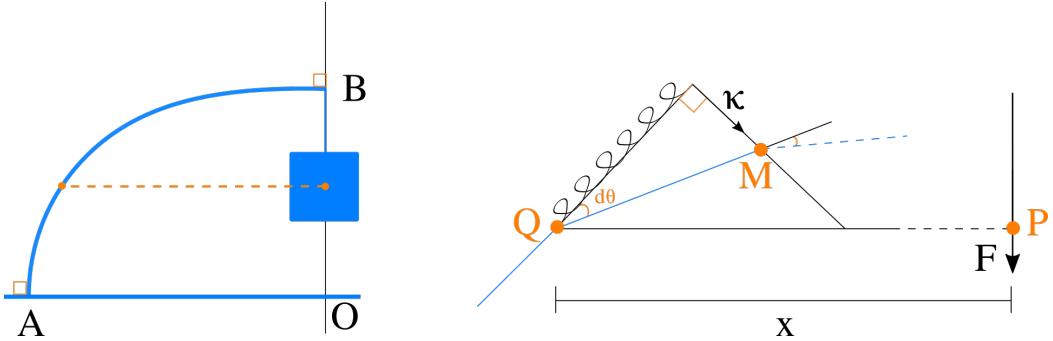


Figure 3.6: **Elastic beam and James Bernoulli's solution.** In equilibrium, the moments at the two ends of the virtual lever PQM should cancel, implying that the curvature is a linear function of x if we assume the spring behaves in accordance with Hooke's law.

3.3 Elastica curve

In 1691 James Bernoulli pose a challenge to **your** colleagues: What is the form taken by an elastic beam whose ~~its~~-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 [Lev08; Tru60],

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

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 $\int_C \kappa^2 ds$.

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

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

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

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 [unconstrained Elastica](#). [Weird ! \(constrained ?\)](#)

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](#). 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

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

In general, a planar curve C minimizing the Elastica energy satisfies the following Euler-Lagrange equation [[Cha+02](#); [Sin08](#)]

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

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.

Imaging models using the Elastica

In [Cha+02] 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 $f_I : \Omega \rightarrow [0, 1]$, let Γ_λ be one of its level curve, i.e.,

$$\Gamma_\lambda = \{x \mid f_I(x) = \lambda\}.$$

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

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

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

$$\frac{d\lambda}{dt} = \|\nabla f_I\| \rightarrow d\lambda = \|\nabla f_I\| dt.$$

Replacing the last expression in [Equation \(3.14\)](#) we obtain

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

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 lessen by the 3rd order gradient flow proposed in [Bal+01]. The idea is to create a new variable θ to

replace the unstable term $\nabla f_I / \|\nabla f_I\|$ 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

$$\int_{\Omega} a |\nabla \cdot \theta|^2 (\alpha + \beta \|k * f_I\|) d\Omega + \int_{\Omega} \|\nabla f_I\| - \theta \cdot \nabla f_I d\Omega, \quad \|\theta\| \leq 1. \quad (3.16)$$

The kernel k (e.g. Gaussian) smooths image f_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 [\[MM98\]](#). The model's first step is to identify pairs of admissible T-junctions, i.e., pixels in the outer 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) . Additionally, 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

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

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 constraint 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 [\[MM98\]](#) 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 [\[MM98\]](#), 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.

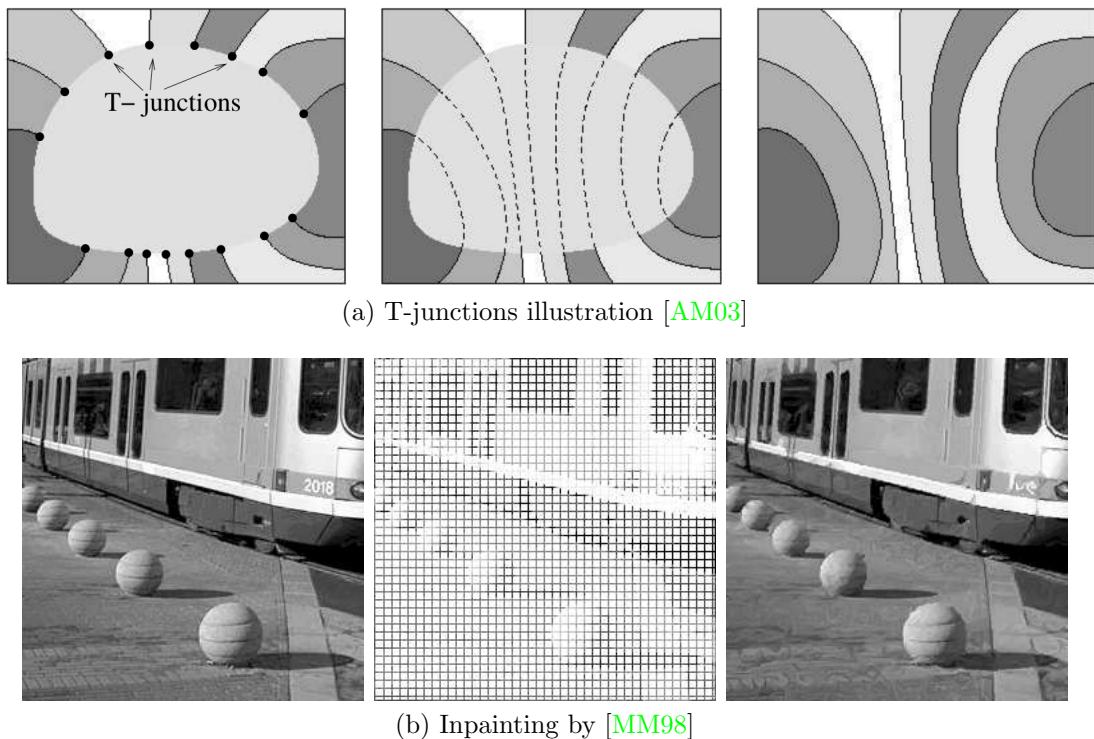


Figure 3.7: **Discrete inpainting.** In Figure 3.7a an illustration of T-junctions and an ideal completion of the level lines; in Figure 3.7b we have the original image in the left, the image to be inpainted in the middle and the result of the discrete inpainting in the right.

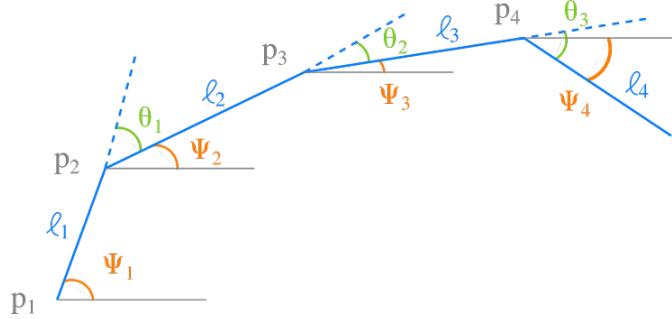


Figure 3.8: **Discrete elastica.** The curvature is approximated by θ_i/l_i .

3.4 Discrete methods and squared curvature

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).

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

Given parameters $\alpha \geq 0, \beta \geq 0$ and a fixed length segment ℓ , the discrete elastica [BHN01] is defined as

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

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

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

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 [BHN01], 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

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

is convergent in the sense of Γ -convergence [BNR01], 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 to be applied 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 elastics along a single contour (not 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;

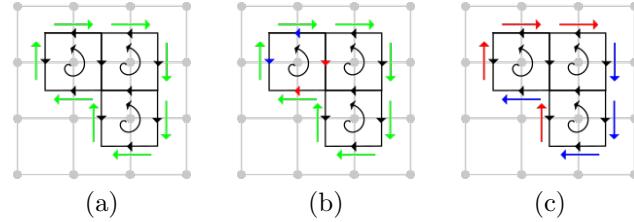


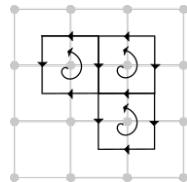
Figure 3.9: **Digital surface and consistency constraints.** In Figure 3.9a, a valid solution made of three cells and a sequence of eight linels. In Figure 3.9b the sum of linel-cell incidence equals to zero (blue are positively incident and red negatively incident). In Figure 3.9c, the sum of linel-edge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

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.

Linear programming model for image segmentation using the discrete elastica

In [SKC09], 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 $\mathbf{x} \in \{0, 1\}^{n+2m}$ 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 \mathbf{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 $\mathbf{A} \in \{-1, 0, 1\}^{m \times (n+2m)}$

this matrix incorporates cell and linel incidence, no ? this is quite non standard

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

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

$$\mathbf{Ax} = 0,$$

i.e., the sum of incidences for each linel must equal to zero (see [Figure 3.9](#)). Finally, we set vector $\mathbf{w} \in \mathbb{R}^{n+2m}$ 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

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

where vector \mathbf{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.

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

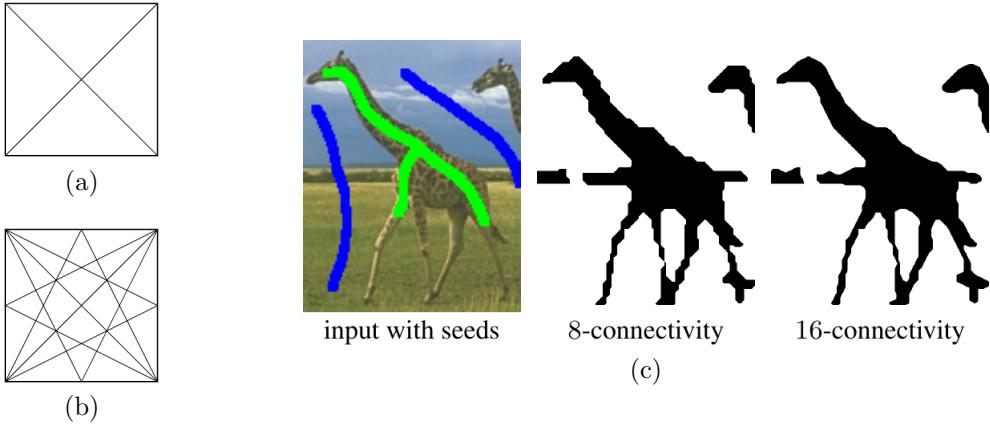


Figure 3.10: **Linear programming model** [SKC09]. In the left, examples of the 8 and 16-connectivity cellular complexes. The authors implementation consists into subdivide a pixel. For example, the 8-connectivity is implemented by rescaling the image twice its size and let the model unit being composed 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.

complexity issue is also covered. Finally, the model is extendable 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. extensible

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.

Unconstrained formulations

The high number of variables and constraints explain the high running time of the linear programming approach. In [EG10], 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 \mathbf{x}_i indicating if the pixel belongs to foreground ($x_i = 1$) or background ($x_i = 0$). For a given segmentation \mathbf{x} , one can identify 90 degrees turns in the 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

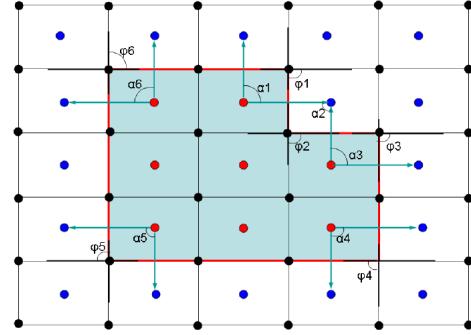


Figure 3.11: **Turn angles detection and triplets** [EG10]. Assuming that red pixels correspond to 1-labeled variables and blue the 0-labeled ones, angle variation is triggered by configurations $(x_i = 1, x_j = 0, x_k = 0)$ and $(x_i = 0, x_j = 1, x_k = 1)$, assuming that x_i is the centered variable.

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

are you sure he is the sole author ?

The coefficients w_{ijk} are defined accordingly with the Brucksteins discretization formula for the elastica, 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

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

where

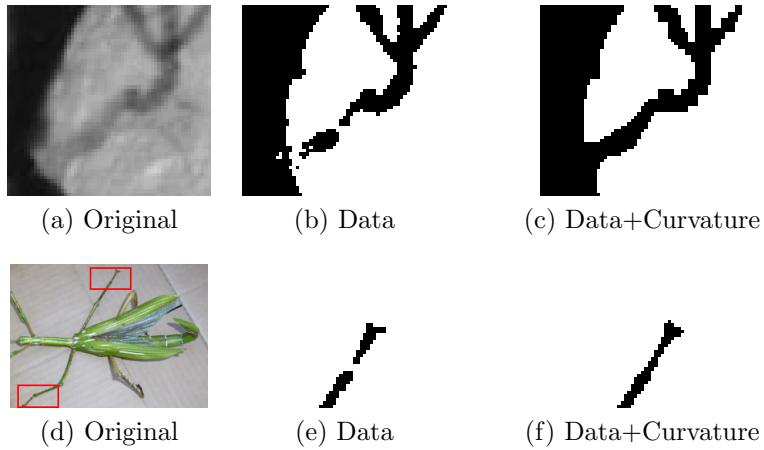


Figure 3.12: **Results for the grid graph model [EG10]**. The original image is displayed in the left; segmentation using only data in the middle; and the segmentation using squared curvature regularization in the right. Connectivity is encouraged, but to a very limited extension.

$$\begin{aligned} w_{ij} &= \frac{1}{2}w_{ijk} \\ w_{ik} &= \frac{1}{2}w_{ijk} \\ w_{jk} &= -\frac{1}{2}w_{ijk}. \end{aligned}$$

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 [Nie+14] is presented an alternative formulation that is extendable to any desirable angle resolution. The idea is in fact quite similar to the one in [EG10] 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

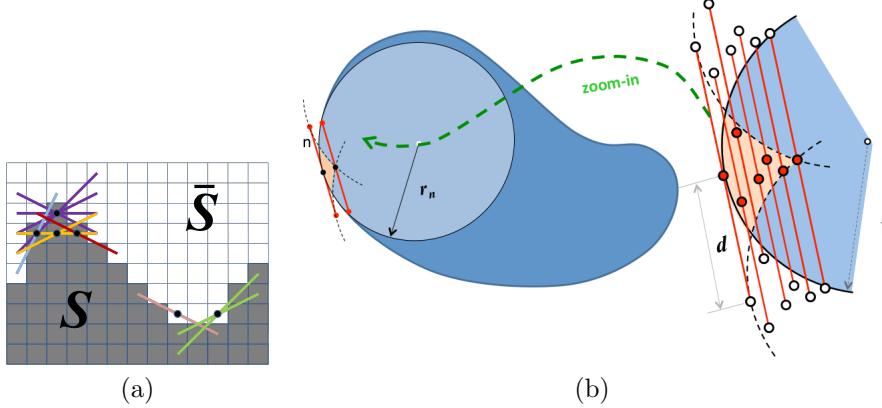


Figure 3.13: **Triplets configurations and integral geometry** [Nie+14]. In Figure 3.13a we intuitively notice that curvature is higher where triplets configurations $(0, 1, 0)$ and $(1, 0, 1)$ are perceived. In Figure 3.13b, an illustration of Theorem 1.

Theorem 1([Nie+14]): 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 tangent at p . The integral squared curvature is approximated by

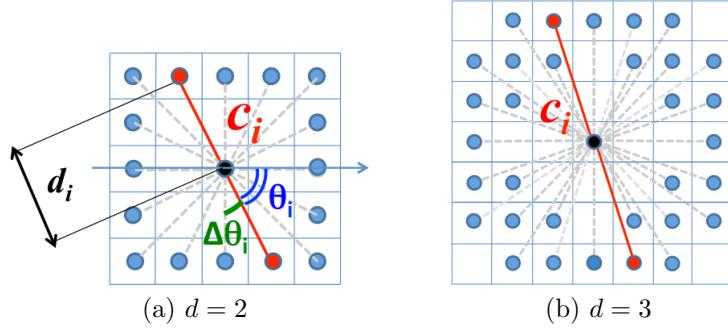


Figure 3.14: **Neighborhood system** [Nie+14]. The higher the value of d , the higher is the angle resolution.

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

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

`\delta`

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

The resulting energy is non-submodular (the positive term in the δ function is non-submodular) and is solved using the LSA-TR method [Gor+14], 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, extendable to arbitrary angle resolution and can be easily modified to include data and length terms. X 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 behaviour of such estimator in shapes of more complex geometry. The fundamental goal of this thesis is to investigate imaging models using geometric

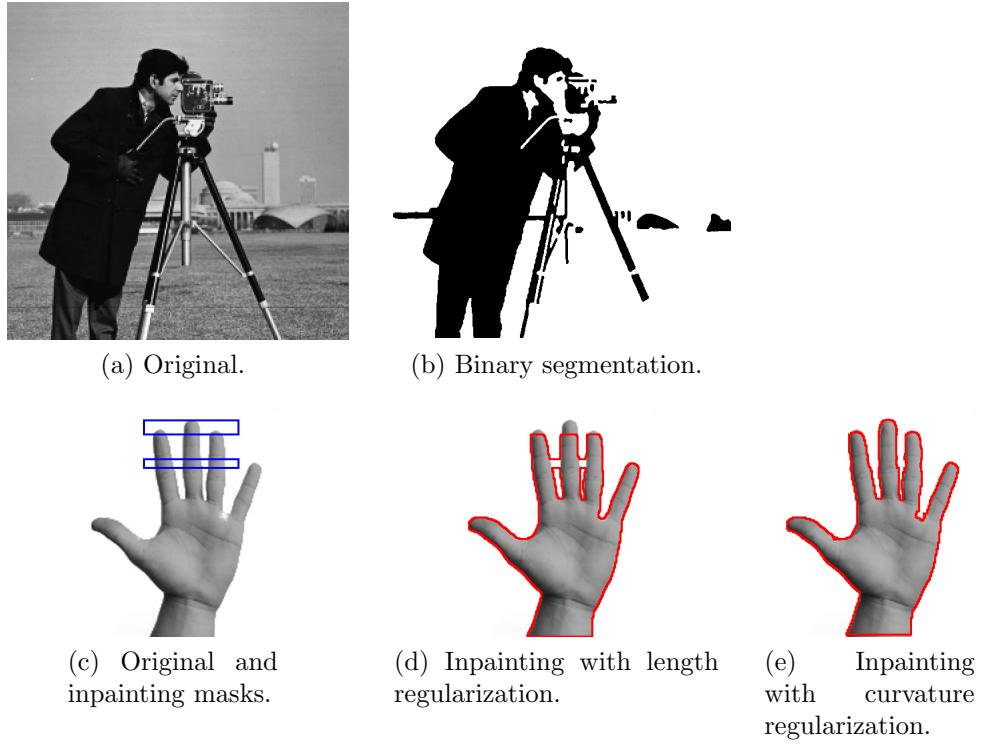


Figure 3.15: **Segmentation and inpainting results** [Nie+14]. The connectivity principle is perfectly illustrated in the inpainting problem but one would expect a completion of the rightmost bar of the camera tripod.

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

Chapter 4

Digital Geometry

Part II

Contribution

Chapter 5

A combinatorial model for digital Elastica shape optimization

In this chapter we review the Elastica energy and some of its properties. Next, we introduce the digital version of the Elastica using multigrid convergent estimators of length and curvature. 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. Finally, 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 Continuous and digital Elastica

To be further developed...

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

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

Similarly, the digital Elastica \hat{E}_Θ of some digitization $D_h(S)$ of S is defined as

$$\hat{E}_\Theta(D_h(S)) = \sum_{\dot{\mathbf{e}} \in \partial D_h(S)} \hat{s}(\dot{\mathbf{e}}) (\alpha + \beta \hat{\kappa}_r^2(D_h(S), \dot{\mathbf{e}}, h)), \quad (5.1)$$

where $\dot{\mathbf{e}}$ denotes the center of the line \mathbf{e} . 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 we omit the grid step h to simplify expressions (or,

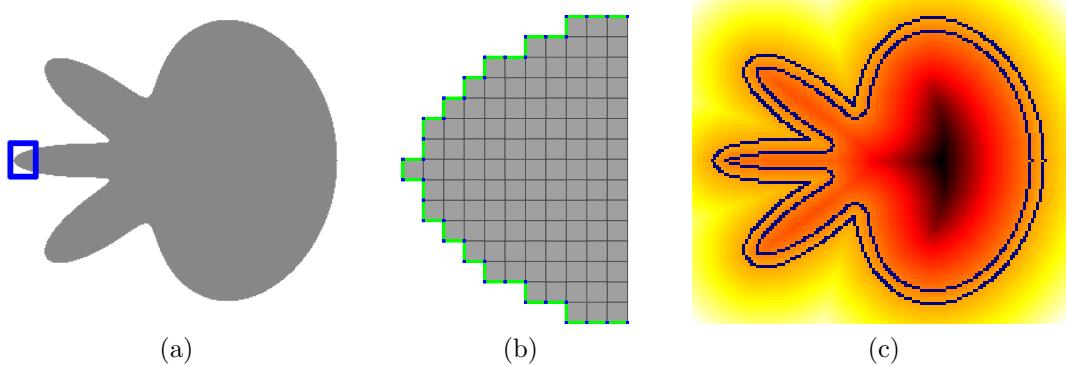


Figure 5.1: The flower shape in figure (a) and the cellular-grid model representation in (b) of the rectangle-bounded region. In figure (b), pixels are colored in gray, linels in green and pointels in blue. In figure (c), the blue pixels denotes a 3-ring set.

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 $\mathcal{N}^{(i)}$ to the shape $D^{(i)}$ and choose the element of $\mathcal{N}^{(i)}$ with lowest energy. The process is suited for the integral invariant estimator but also for other curvature estimators, for example, MDCA [[RL11](#)]. As a matter of fact, our experiments have shown that both estimators induce similar results.

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 5.1](#)). 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 $d_D : \Omega \rightarrow \mathbb{R}$ 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(m-Ring Set): Given a digital shape $D \in \Omega$, its signed distance transformation d_D and natural number $m \neq 0$, the m -ring set of D is defined as

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

where

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

Consider the following set of neighbor candidates to D :

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

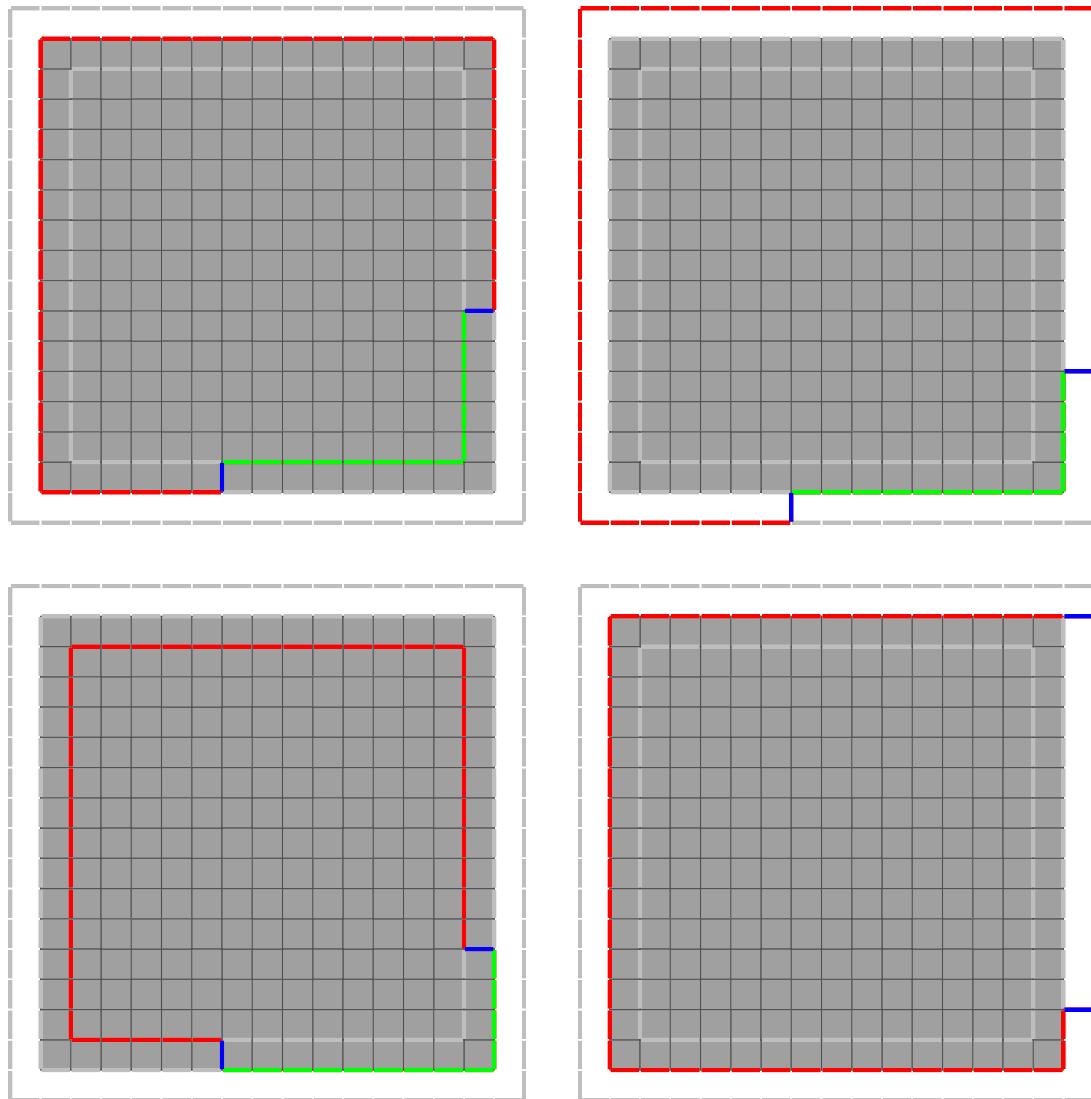
Such set can be extremely large and its complete exhaustion is prohibitively expensive. Instead, we are going to use a subset of it.

Definition 2(n -neighborhood):

Given a digital shape $D \in \Omega$, its n -neighborhood $\mathcal{N}_n(D)$ 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 $R_1(D)$.

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{N}_n is likely very close to the greatest reduction for a member of \mathcal{N}_{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 .

The *LocalSearch Algorithm 2* describes the local combinatorial process and is suitable for any type of digital estimator. To estimate length we use MDSS and to estimate curvature we execute *Algorithm 2* with the MDCA and II- r estimators (r denoting the radius of the estimation ball) to solve the free and constrained Elastica problems.

Figure 5.2: Members of \mathcal{N}_{11} for the square shape.

```

input : A digital set  $D$ ; weight coefficients  $\Theta = (\alpha, \beta)$ ; the number of curve
segments  $nc$ ; the maximum number of iterations maxIt

 $t \leftarrow 1$ ; // multiscale level
 $k \leftarrow 0$ ; // current iteration
 $D^{(0)} \leftarrow D$ ;
while  $k < \text{maxIt}$  and  $t < \log_2 |\partial D^{(k)}|$  do
     $M^{(k,t)} \leftarrow |\partial D^{(k)}|/2^t$ ; // Maximum  $n$ -neighborhood value.
     $J \leftarrow \{j \cdot \frac{M^{(k,t)}}{nc} \mid 1 \leq j \leq nc\}$ ;
     $\mathcal{N}^{(k,t)} \leftarrow \bigcup_{j \in J} \mathcal{N}_j(D^{(k)})$ ;
    //Find neighbor shape with lowest energy.
     $Q^* \leftarrow D^{(k-1)}$ ;
    for  $Q \in \mathcal{N}^{(k,t)}$  do
        if  $\hat{E}_\Theta(Q) < \hat{E}_\Theta(Q^*)$  then
             $Q^* \leftarrow Q$ ;
        end
    end
    delta  $\leftarrow \hat{E}_\Theta(D^{(k-1)}) - \hat{E}_\Theta(D^{(k)})$ ;
    if delta  $\leq 0$  then
        //Better solution not found. Refine the scale.
         $t \leftarrow t + 1$ 
    end
    else
        //Better solution found. Set  $D^{(k)}$  and reset to highest
        //scale.
         $t \leftarrow 1$ ;
         $D^{(k)} \leftarrow Q^*$ ;
         $k \leftarrow k + 1$ ;
    end
end

```

Algorithm 2: LocalSearch algorithm for Elastica minimization.

5.2.1 Free digital Elastica

In the free digital Elastica energy we optimize Equation (5.1) without any constraint. We observe that for $\alpha = 0, \beta > 0$ the Elastica becomes the integration of the squared curvature along the shape contour which has the ball of infinite radius as its minimizer. For $\alpha > 0, \beta = 0$, minimize Elastica becomes minimize perimeter (curvature flow). It is easy to see that for $\alpha, \beta > 0$, the optimal shape for the Elastica is a disk of radius r . We can easily find the value of r .

$$\begin{aligned} \frac{d}{dr} \left(\int_{\partial B(r)} (\alpha + \beta \kappa^2) ds \right) &= 0 \\ \frac{d}{dr} (\alpha 2\pi r + \beta 2\pi/r) &= 0 \\ r &= \left(\frac{\beta}{\alpha} \right)^{1/2}. \end{aligned}$$

Therefore, the optimal shape for the free digital Elastica is a digital disk of finite radius $(\beta/\alpha)^{1/2}$.

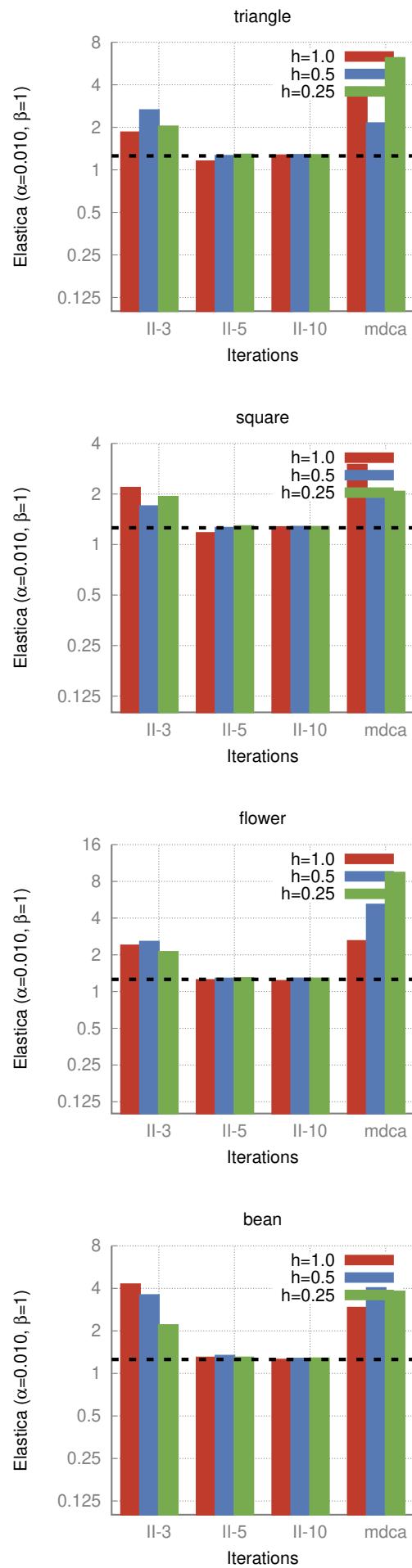
In [Figure 5.3](#) we present the digital Elastica evolution for parameters $\alpha = 0.01$, $\beta = 1$ and three different curvature estimators in three different scales. The shapes evolution using the II-5 estimator are shown in [Figure 5.5](#). We observe that both II-5 and II-10 evolve the shapes 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](#).

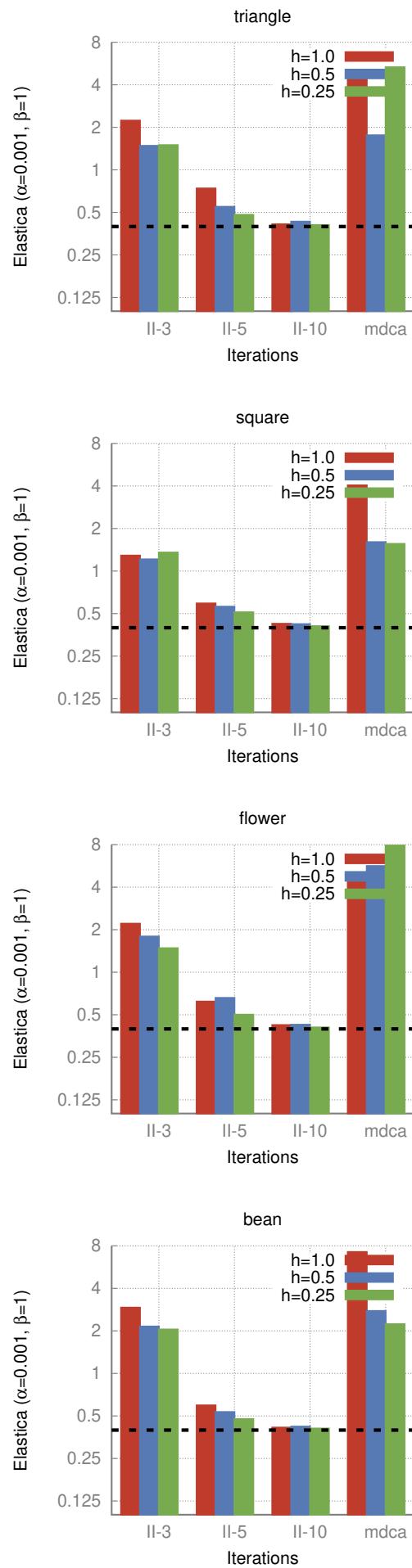
We have executed the same experiments for different parameters α to confirm the effectiveness of our approach. We observe that the plots for $\alpha = 0.001$ in [Figure 5.4](#) follows a pattern similar to those in [Figure 5.3](#) for $\alpha = 0.01$. In particular, the remarks for the II-3 and MDCA 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.

5.2.2 Constrained digital Elastica

An important advantage of [Algorithm 2](#) 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. In [Figure 5.8](#) we compare the flows for different values of α .

We remark that [Algorithm 2](#) is sensitive to the parameter α . For higher values of α , the shapes tends to shrink and the curves are closer to a straight line. For lower values of α , the shapes tends to grow and the curves make more turns.





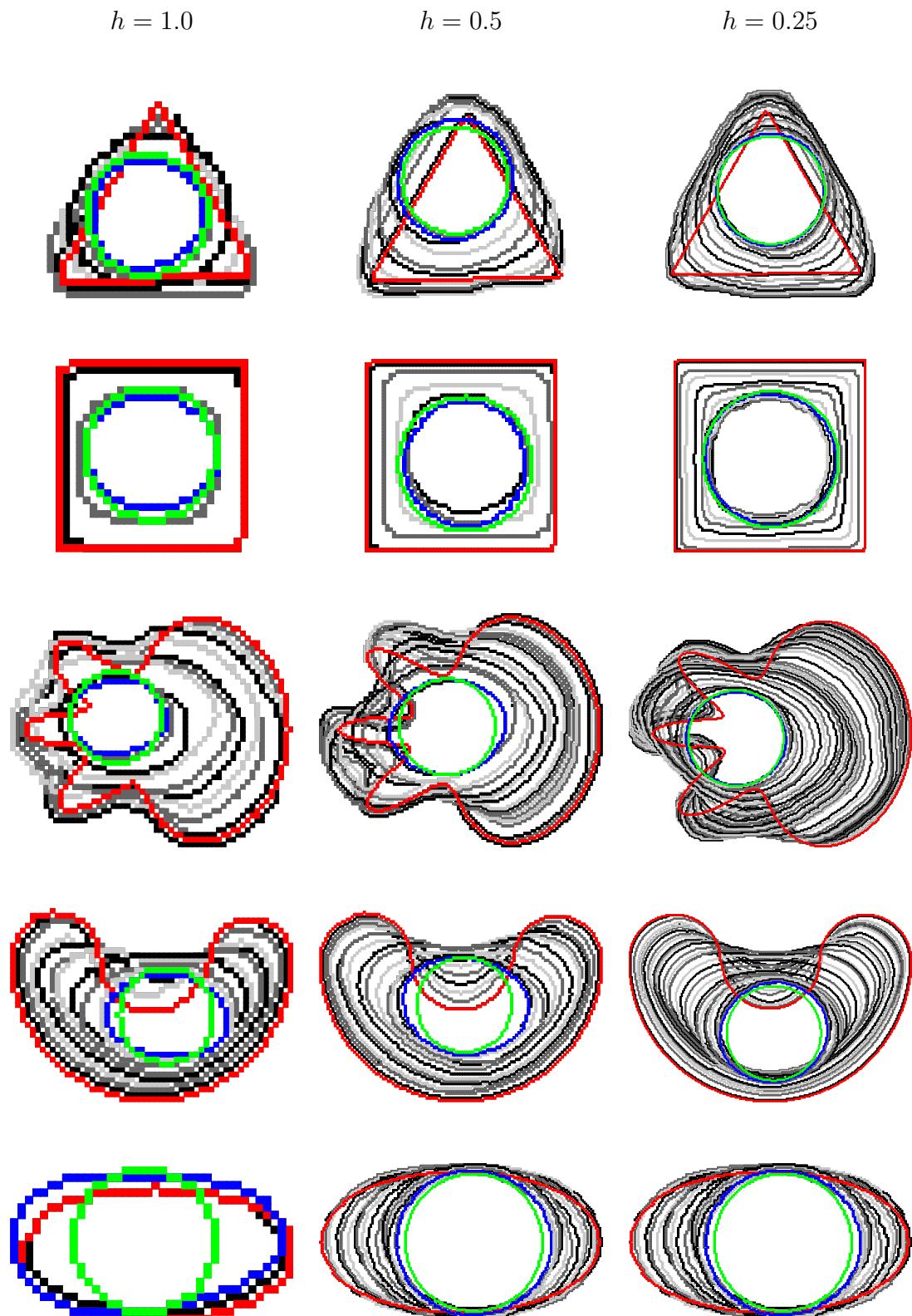


Figure 5.5: LocalSearch algorithm evolutions for several shapes with $\alpha = 0.01$, $\beta = 1$. The initial contour is colored in red; the final contour is colored in blue; and the optimal contour is colored in green.

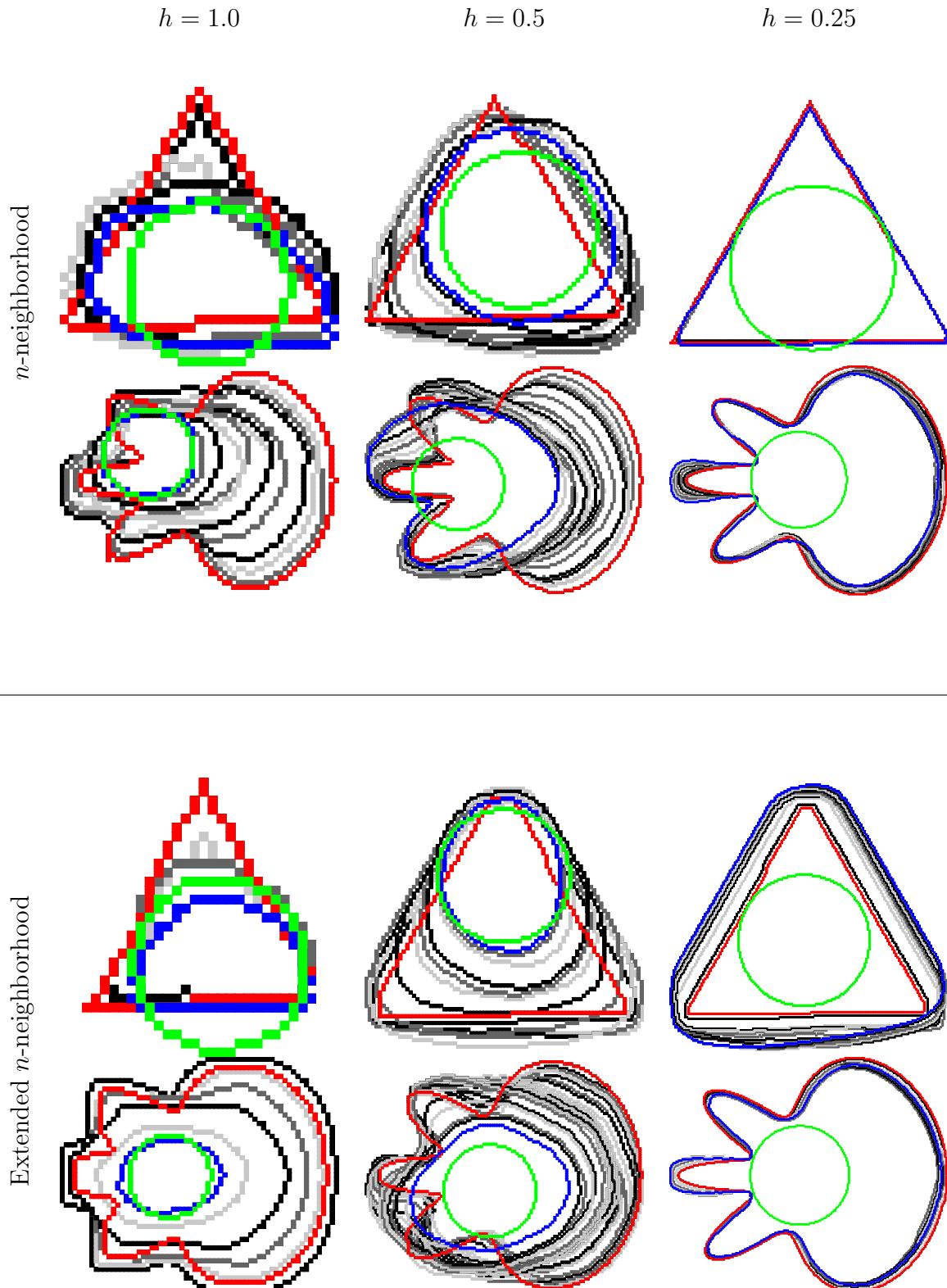


Figure 5.6: In the top row, the MDCA evolution for the neighborhood as presented in [Algorithm 2](#). In the bottom row, the flow using the extended neighborhood. The extended neighborhood additionally includes the n -neighborhood of the dilation and the erosion of the initial shape by a square of side 1.

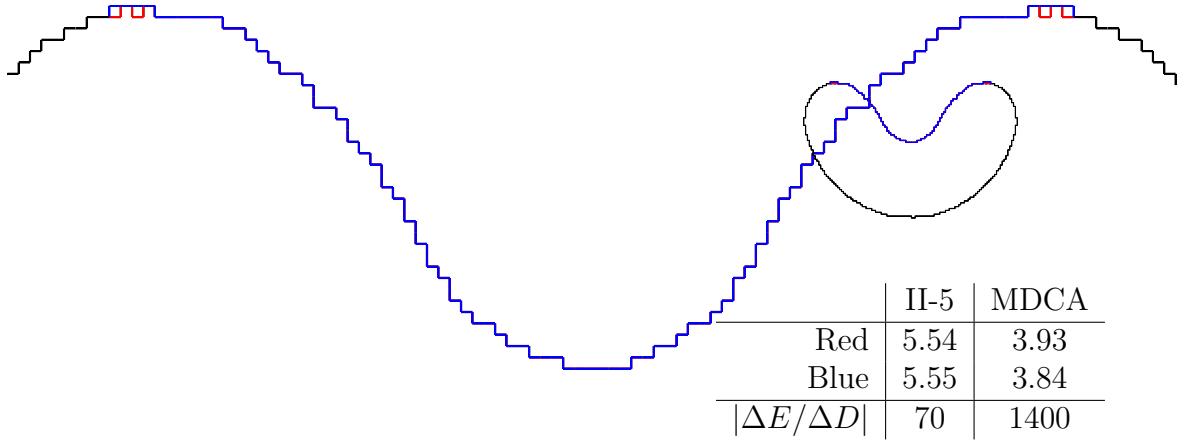


Figure 5.7: A slight variation in the shape boundary (in this example, a 0.07% change or 4 pixels over 5310) inflicts a considerably higher change in the energy value when using MDCA than when using II.

5.2.3 Running time

The running time of [Algorithm 2](#) 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. We observe that it is a complete digital approach, and we do not suffer from discretization and rounding problems, a 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.

5.3 Global optimization

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.

5.3.1 Simplified digital Elastica

The simplified digital Elastica is defined as

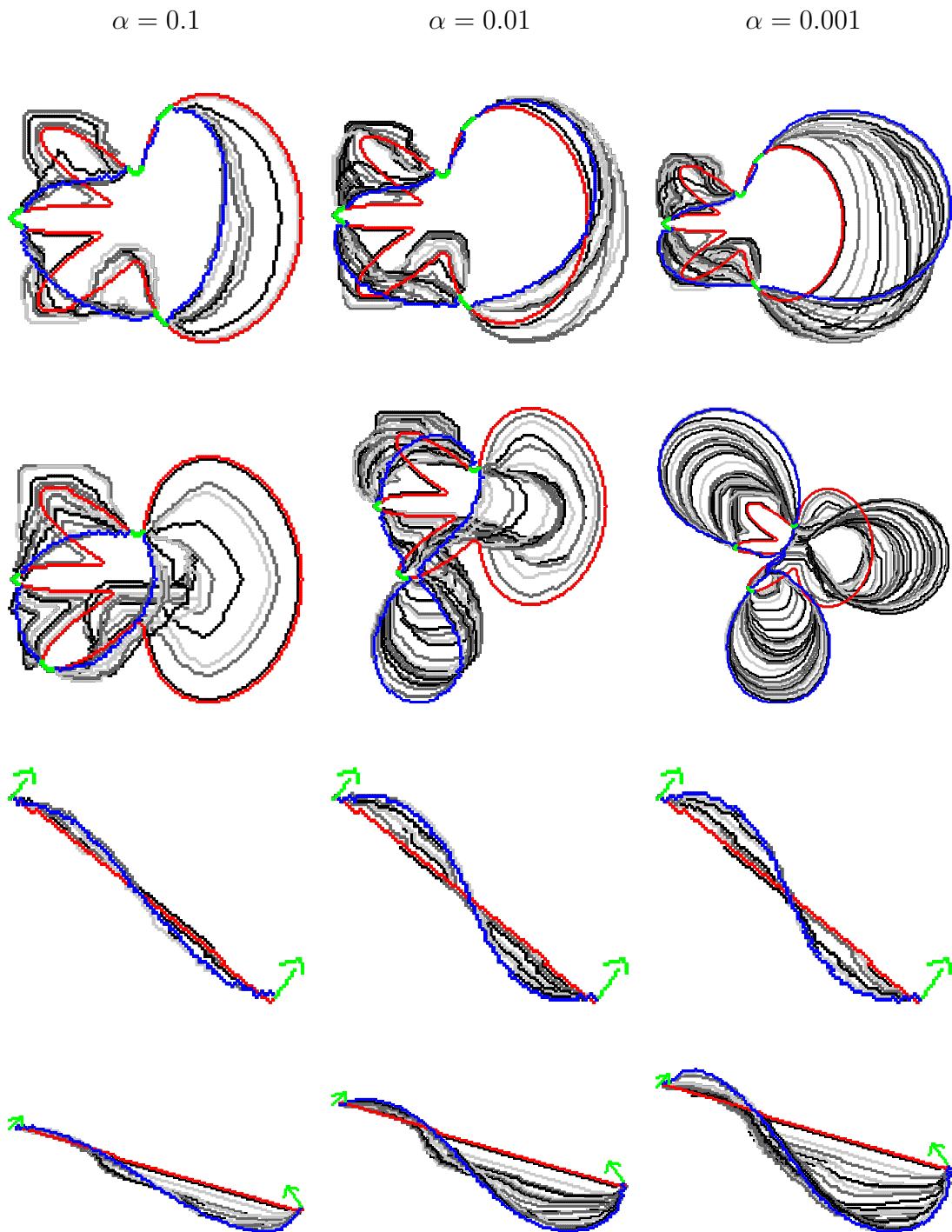


Figure 5.8: In the first and second rows, the flow obtained by forcing the green pixels to be part of the final solution; In the last two rows, the flow obtained by forcing the orientation at the endpoints of the curves.

	$h = 1.0$		$h = 0.5$		$h = 0.25$	
	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 for the free Elastica problem.

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

We argue that Equation (5.2) is a reasonable approximation of Equation (5.1). Indeed, executing Algorithm 2 to minimize this simplified digital Elastica induces very similar results to those for the digital Elastica (see Figure 5.9).

5.3.2 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 $D \in \frac{1}{2}\mathbb{Z}^2$ be the digitization of some shape $S \in \mathbb{R}^2$ using grid step h 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 $\mathbf{x} \in \mathbb{B}^m$, $\mathbf{y} \in \mathbb{B}^n$ and its i -th coefficients are denoted $\mathbf{x}_i, \mathbf{y}_i$. Further, let $\mathbf{A} \in \mathbb{B}^{m \times n}$ the matrix defined as

$$\mathbf{A}_{i,j} = \begin{cases} 1, & x_j \in B_r(y_i) \\ 0, & \text{otherwise.} \end{cases}$$

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

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

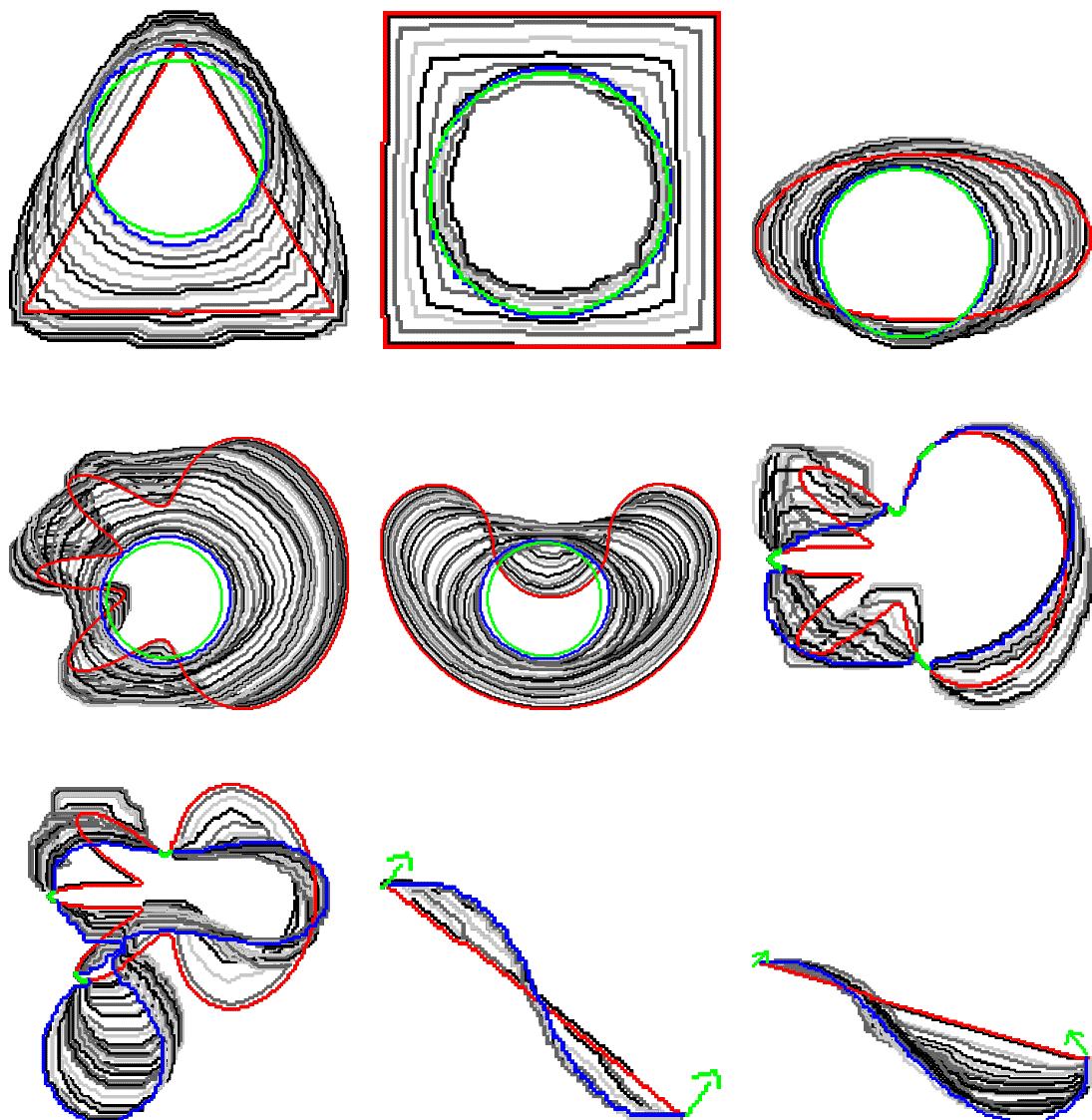


Figure 5.9: Experiments of Section 5.2 for the simplified digital Elastica.

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

$$\min_{\mathbf{x} \in \mathbb{B}^{|X|}, \mathbf{y} \in \mathbb{B}^{|Y|}} E_{\Theta}^{simp}(\mathbf{x}, \mathbf{y}), \quad \text{subject to } T(\mathbf{x}, \mathbf{y}). \quad (P0)$$

Additionally, 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

$$\min_{\mathbf{x} \in \mathbb{B}^{|X|}, \mathbf{y} \in \mathbb{B}^{|Y|}} E_{\Theta}^{simp}(\mathbf{x}, \mathbf{y}), \quad \text{subject to } T(\mathbf{x}, \mathbf{y}), R(\mathbf{x}) \quad (P1)$$

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.

5.3.3 Topological constraints

The estimation ball should be applied in the digital boundary of the shape, which oblige 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.

We create the vector $\mathbf{z} \in \mathbb{B}^{2n}$. We map each linel identified by variable \mathbf{y}_i to components $\mathbf{z}_{2i}, \mathbf{z}_{2i+1}$, 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 $\mathbf{T} \in \mathbb{B}^{n \times m+2n}$ is defined as

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

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

Rewriting formulation (P1)

$$\min \sum_{z_i \in \mathbf{z}} z_i \left(\alpha + \frac{9}{r^6} \beta (c^2 - 2c \mathbf{A}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{A}_i \mathbf{A}_i^T \mathbf{x}) \right)$$

subject to

$$\begin{aligned} \mathbf{T} \times \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} &= 0 \\ R(\mathbf{x}), \\ \mathbf{x} \in \mathbb{B}^m, \mathbf{z} \in \mathbb{B}^{2n}. \end{aligned}$$

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

5.3.4 Linear relaxation of P1

The simplest model we can derive from (P1) consists in the relaxation of the optimization variables, i.e., we impose $\mathbf{x} \in \mathbb{U}^m$ and $\mathbf{z} \in \mathbb{U}^{2n}$, 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 $\mathbf{x}_2^2 \mathbf{x}_4$ is encoded as the sequence $(\mathbf{x}_2, \mathbf{x}_2, \mathbf{x}_4)$. Let \mathcal{T} the collection of opt-terms of order two or higher in (P1) and $\mathcal{T}_i \in \mathcal{T}$ a member of this collection. To linearize (P1), we associate a variable \mathbf{u}_i for each term of \mathcal{T} , i.e., $\mathbf{u} \in \mathbb{U}^{|\mathcal{T}|}$ and we enforce $|\mathcal{T}_i| + 1$ new constraints. In other words, we add the following set of linearization constraints.

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

The linearization of (P1) is written as

$$\min \sum_{z_i \in \mathbf{z}} z_i \left(\alpha + \frac{9}{r^6} \beta (c^2 - 2c \mathbf{A}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{A}_i \mathbf{A}_i^T \mathbf{x}) \right)$$

subject to

$$\begin{aligned} \mathbf{T} \times \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} &= 0 \\ R(\mathbf{x}), \\ L(\mathbf{u}), \\ \mathbf{x} \in \mathbb{U}^m, \mathbf{z} \in \mathbb{U}^{2n}, \mathbf{u} \in \mathbb{U}^{|\mathcal{T}|} \end{aligned}$$

Finally, to obtain a binary vector we round the partial solution vector $\mathbf{x}^\star \in \mathbb{U}^m$. 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, fundamental condition for efficient optimization of the model.

5.3.5 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 $\mathbf{q} \in \mathbb{Z}^m$ for pixels $\mathbf{x} \in \mathbb{B}^m$ is

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

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

$$\mathbf{Q} = \text{diag}(\mathbf{q})\text{diag}(\mathbf{q})$$

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

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

where $\mathbf{1} = (1, 1, \dots, 1)^T \in \mathbb{R}^m$. [Equation \(5.4\)](#) involves the minimization of a fourth order equation and therefore hard to be optimized.

5.4 Conclusion

We gave a historical review of the Elastica and we defined the digital Elastica energy in [Section 5.1](#). The local combinatorial scheme defined in [Section 5.2](#) can evolve different shapes guided by the minimization of digital Elastica energy and it eventually reaches global optimum in the free Elastica problem, justifying the interest for multigrid convergent estimators. The model can also be used to solve the constrained Elastica problem, but its more likely to stop in a local minimum. Finally, we sketch some global optimization models in [Section 5.3](#) for minimizing the simplified Elastica using standard techniques of optimization. The difficulties we pointed out suggest that a practical global optimization model is unlikely to exist. In the next chapter we explore a model that decreases the Elastica energy and that can be used in practice.

Chapter 6

A 2-step evolution model driven by digital Elastica minimization

In the previous chapter we have presented a local combinatorial model using multi-grid 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 2](#) but with fewer guarantees of optimality.

6.1 FlipFlow model

In this section we describe the FlipFlow model that aims to evolve a initial digital shape D into another of lower digital Elastica value. The FlipFlow algorithm consists into 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 $\{D^{(k)} \mid k \geq 0, D^0 = D\}$ intended to decrease the digital Elastica energy of D .

We assume an ordering in Ω , i.e., there exists a bijective function $\omega : \Omega \rightarrow \{1 \dots |\Omega|\}$. Moreover, let $X_\omega : \Omega \rightarrow 2^{\{0,1\}}$ be an operator that transforms digital sets in its corresponding set of binary variables, i.e.,

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

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

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

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 5.3.3](#), we limit the optimization region to a subset of Ω , namely the inner pixel boundary of $D^{(k)}$.

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).

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 $\bar{X}^{(k)}$, i.e., $\bar{X}^{(k)} = X(I(\bar{D}^{(k)}))$.

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

6.1.2 Algorithm

We recall the definition of the II digital curvature estimator:

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

where $c_1 = 3/r^6$ and $c_2 = \pi r^2/2$.

The following sets are important in the expansion of [Equation \(6.1\)](#).

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

Expanding [Equation \(6.1\)](#), we get

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

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 $\theta = (\alpha, \beta) \geq 0$

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

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

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

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 3*. To optimize [Equation \(6.3\)](#) we use the QPBOI algorithm [[Rot+07](#)].

```

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

 $D^{(0)} \leftarrow D;$ 
 $k \leftarrow 1;$ 
while  $k < \text{maxIt}$  do
    //Shrinking mode
    if  $k$  is even then
         $x^{(k-1)} \leftarrow \arg \min_{X^{(k-1)}} E_{(\theta,m)}^{\text{flip}}(D^{(k-1)}, 1 - X^{(k-1)});$ 
         $D^{(k)} \leftarrow F^{(k-1)} + x^{(k-1)};$ 
    end
    //Expansion mode
    else
         $\bar{x}^{(k-1)} \leftarrow \arg \min_{\bar{X}^{(k-1)}} E_{(\theta,m)}^{\text{flip}}(D^{(k-1)}, 1 - \bar{X}^{(k-1)});$ 
         $D^{(k)} \leftarrow \bar{F}^{(k-1)} + \bar{x}^{(k-1)};$ 
    end
     $k \leftarrow k + 1;$ 
end

```

Algorithm 3: FlipFlow algorithm.

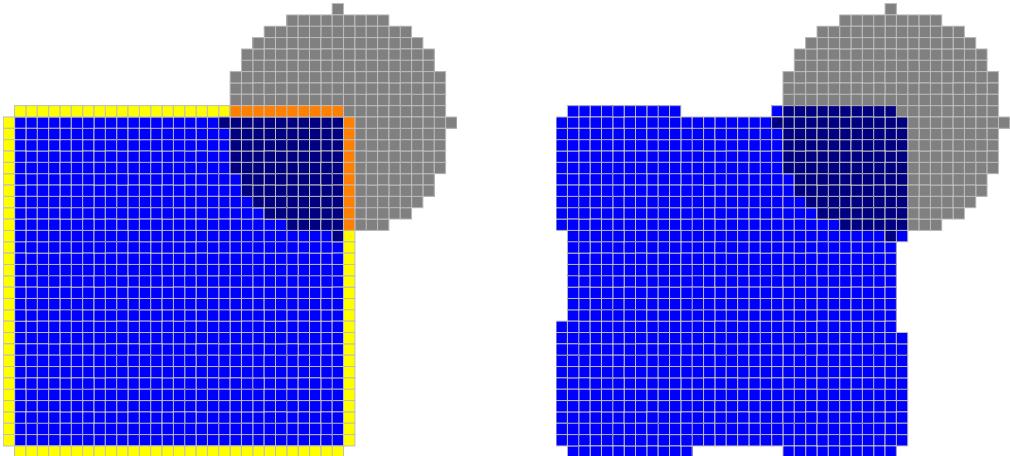


Figure 6.1: Direct use of the optimization result of [Equation \(6.3\)](#) does not decrease squared curvature because contour information is not present in the energy.

6.1.3 Algorithm discussion

As discussed in [Section 5.3](#), 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 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

$$D^{(k+1)} = D^{(k)} - \arg \min E_{(\theta, m)}^{\text{flip}}(D^{(k)}, 1 - X^{(k)}).$$

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 3](#) is made of two modes: shrinking and expansion. The shrinking mode handles convexities and its reasoning is explained in the last paragraph. The expansion mode operates exactly in the same way, but at 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 6.1](#) sums up these arguments.

In [Figure 6.2](#) we show the results of the FlipFlow algorithm for $m = 1, \alpha = 0, \beta = 1$. 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.

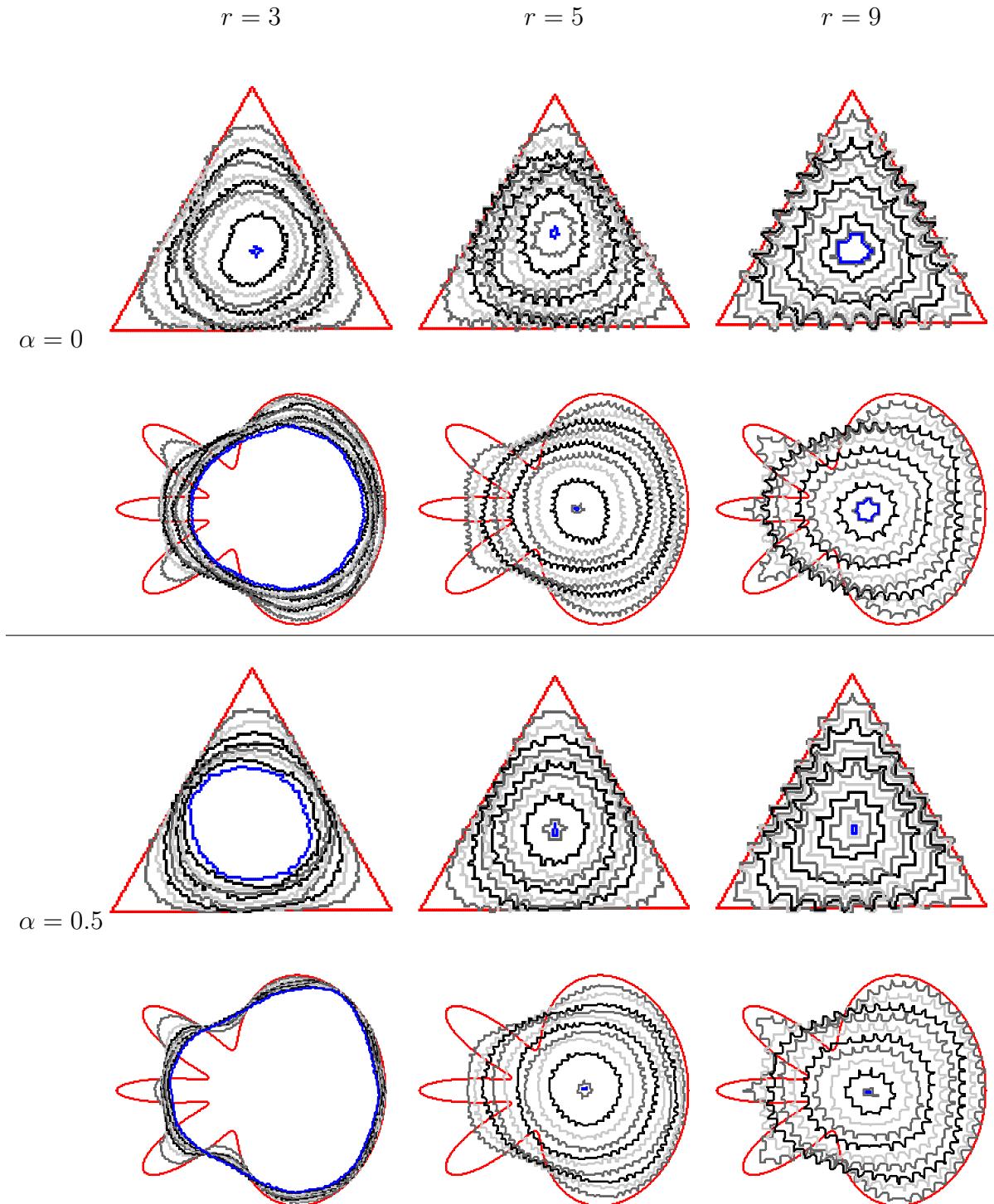


Figure 6.2: The algorithm is very sensitive to the little variations of the estimator, which are particularly important in regions of low squared curvature. Artifacts are somewhat reduced with a length penalization but increases if we use a higher ball radius. For better visualization, curves are displayed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

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

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

6.2 Optimization method

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

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

The function f is submodular if and only if the following inequality holds for each pairwise term $f_{j,k}$ [KZ04]:

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

The energy $E_{(\theta, m)}^{\text{flip}}$ is non-submodular and optimizing it is a difficult problem, which constrains us to use heuristics and approximation algorithms. The QPBO method [HHS84] 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: QPBOP (improve) and QPBOP (probe) [Rot+07]. 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 $E_{(\theta, m)}^{\text{flip}}$ is quite high, but the percentage decreases to zero as we set m to a value closer to r , the estimation ball

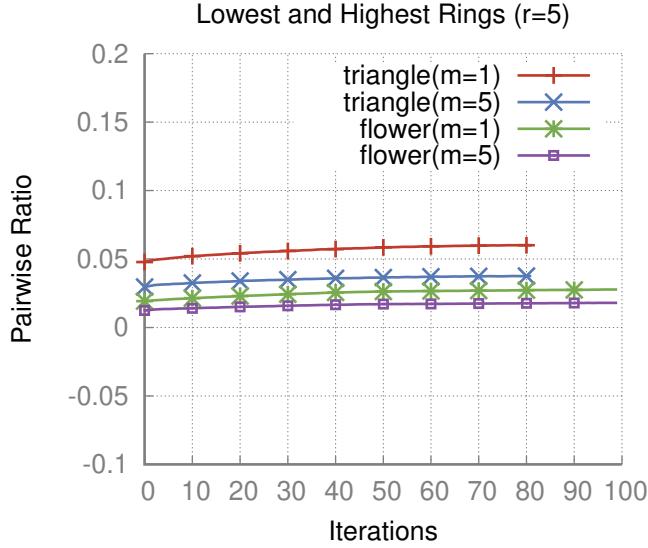


Figure 6.3: We plot the ratio of pairwise terms among all $\binom{|X^{(k)}|}{2}$ combinations. The highest ring has roughly half the number of pairwise terms as the lowest ring.

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 $E_{(\theta,r)-flip}$ is an easy instance of the general non-submodular energy $E_{(\theta,m)}^{flip}$, but this remains to be proved. The number of pairwise terms in $E_{(\theta,r)-flip}$ is roughly half of those in $E_{(\theta,1)-flip}$ (see Figure 6.3), which also explains the higher number of labeled variables.

We have used QPBO to solve $E_{(\theta,m)}^{flip}$. Naturally, in the case where all pixels are labeled by QPBOP, QPBO returns the same labeling as QPBOP. In the next section we show that by evaluating the estimation ball at outer rings, we eliminate the artifacts and we produce smoother flows while preserving a qualitative measure of curvature.

6.3 Evaluation across m -rings

The QPBOP method leaves many pixels unlabeled for $m = 0$, but the ratio of unlabeled pixels tends to decrease as we take higher values of m . 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.

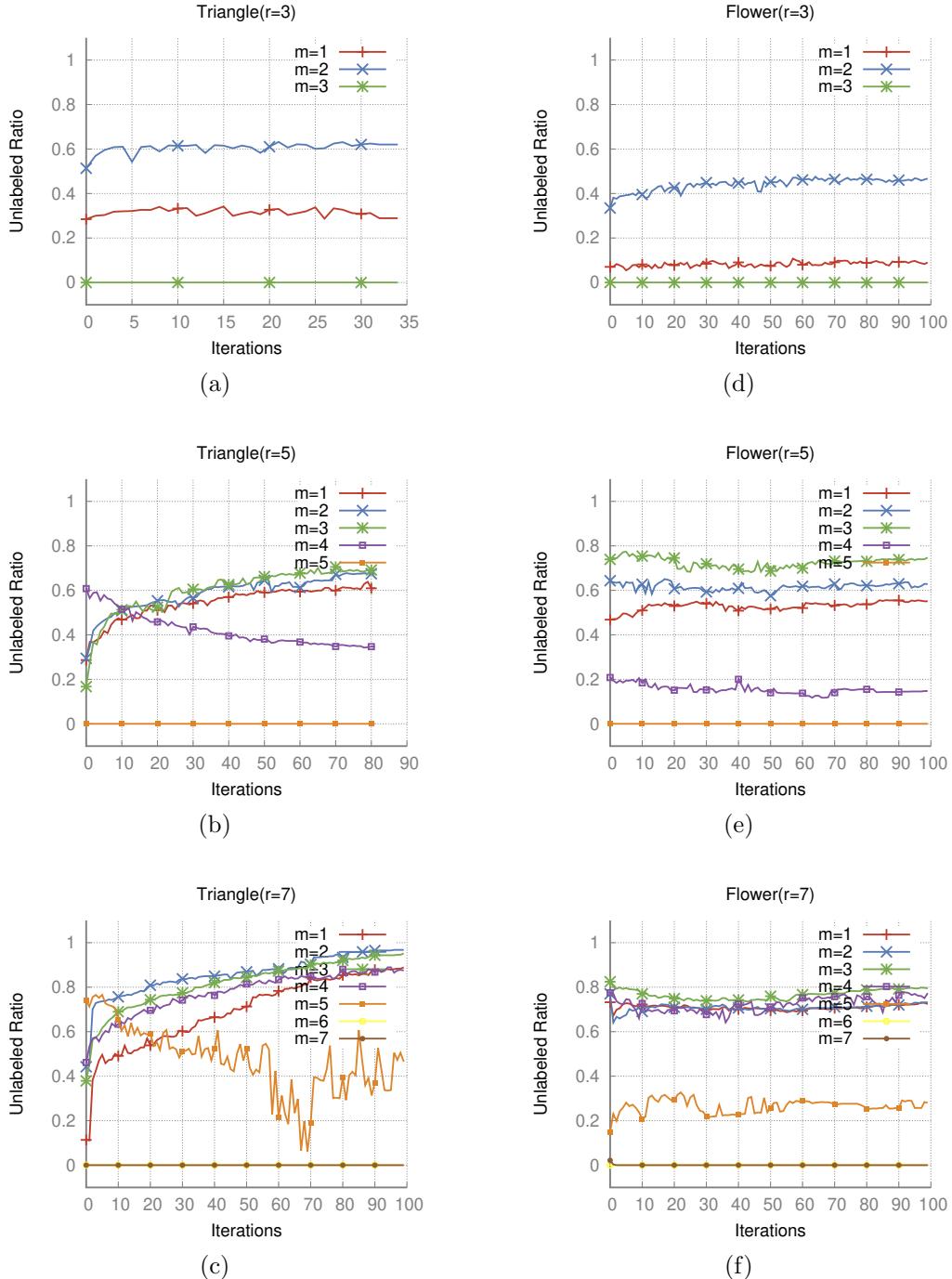


Figure 6.4: For each plot, we first produce the sequence of shapes $\{D^{(k)}\}$ executing FlipFlow with $m = r$. Then, for each shape in $\{D^{(k)}\}$, we execute one iteration of FlipFlow for different values of m and we count the unlabeled pixels. The number of unlabeled pixels by QPBOP remains high for lower values of m , and goes to zero when $m = r$. We observe the same behavior for varying radius values.

In Figures 6.5 and 6.6 we evaluate several flows for different energies $E_{(\theta,m)}^{flip}$. 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 curvature flow.

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 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.

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

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 GrabCut [RKB04]. 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.

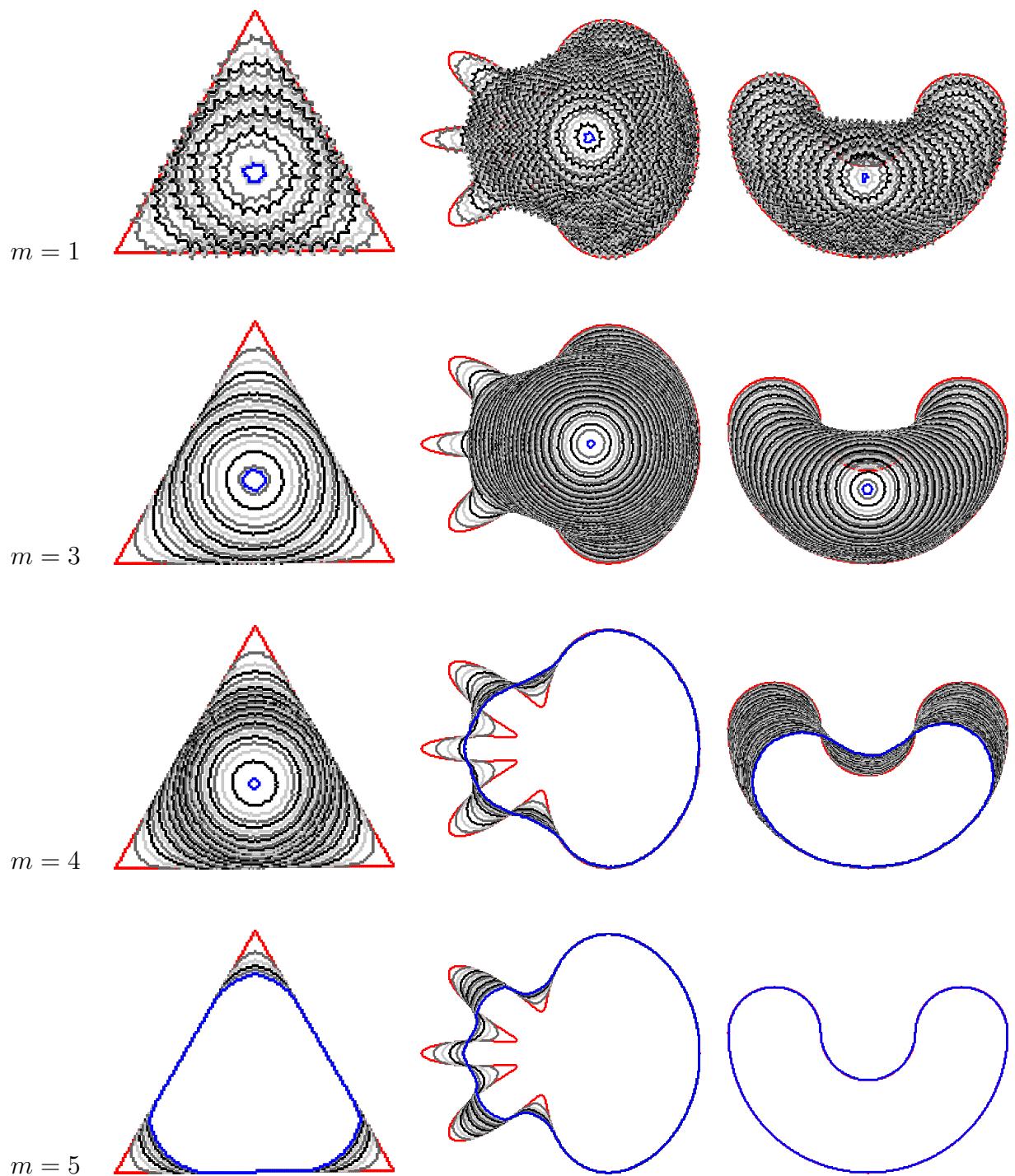


Figure 6.5: By positioning the estimation ball on outer rings, we minimize artifacts creation. The radius of the estimation ball used here equals to 5 and the curves are displayed every 10 iterations.

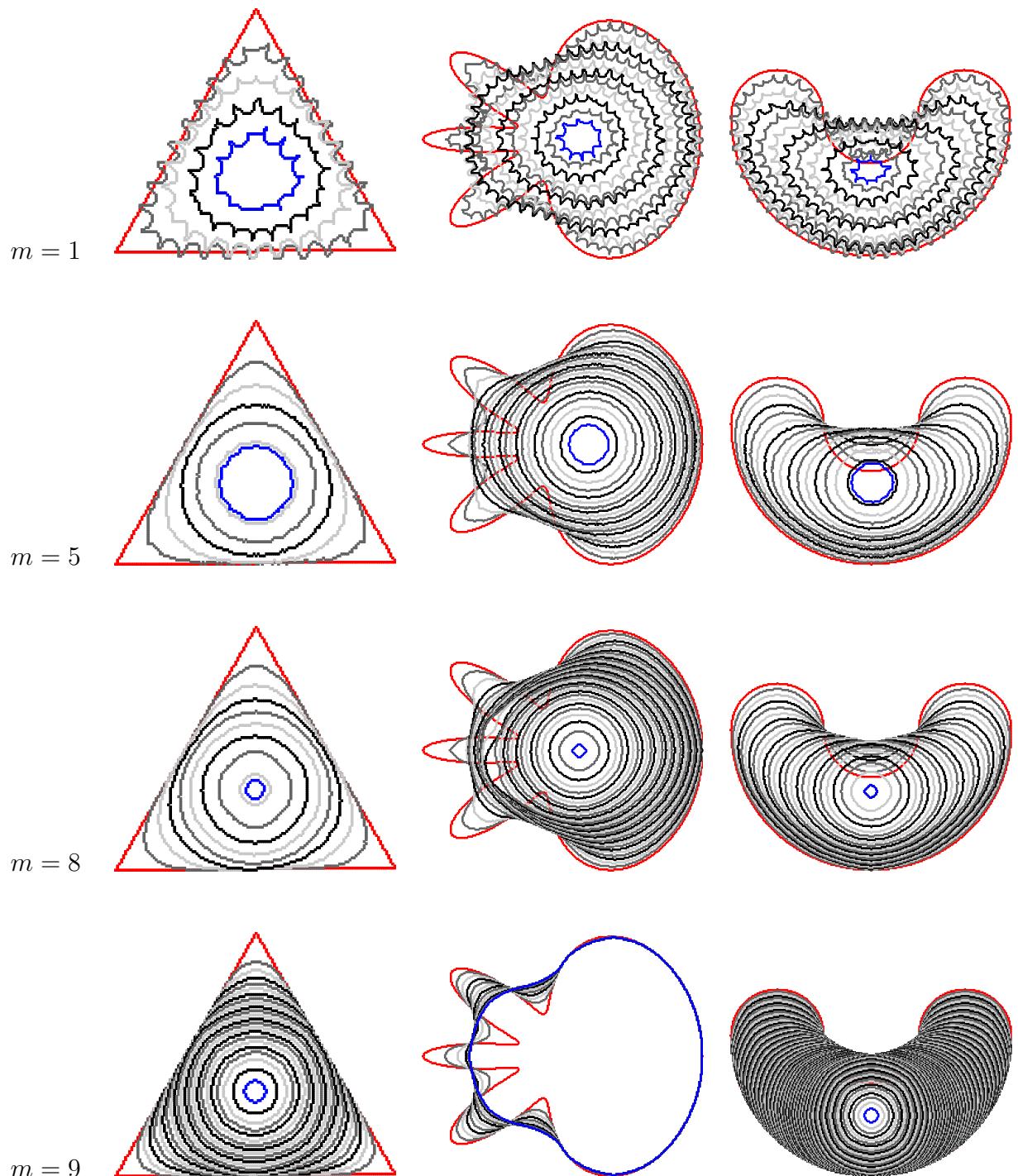


Figure 6.6: By positioning the estimation ball on outer rings, we minimize artifacts creation. The radius of the estimation ball used here equals to 9 and the curves are displayed every 10 iterations.

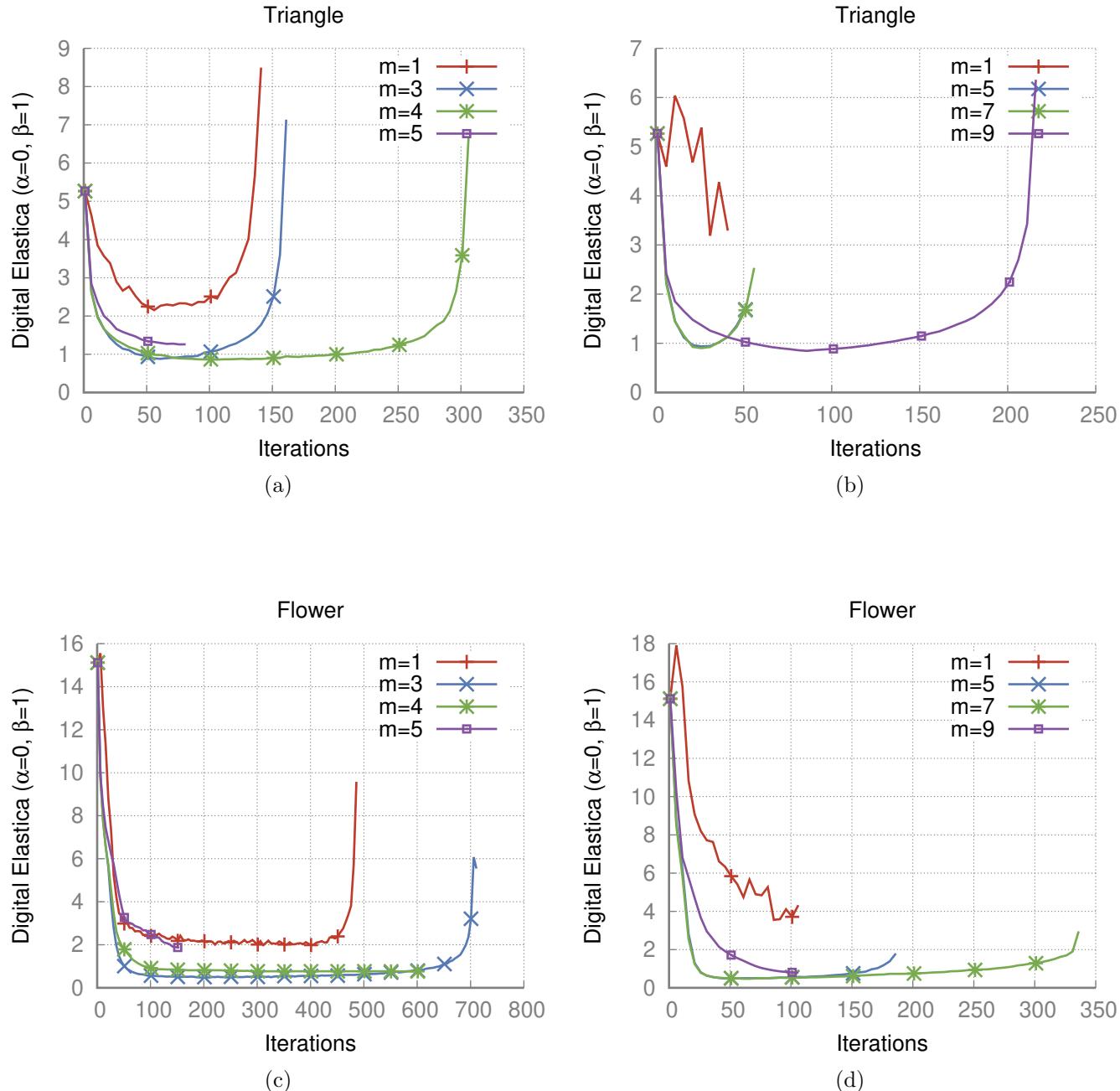


Figure 6.7: Digital Elastica ($\text{II-5}, h = 0.25$) evaluation at each iteration for several m -FlipFlow evolutions. In the left column, we use FlipFlow radius 5 and in the right column we use FlipFlow radius 9.

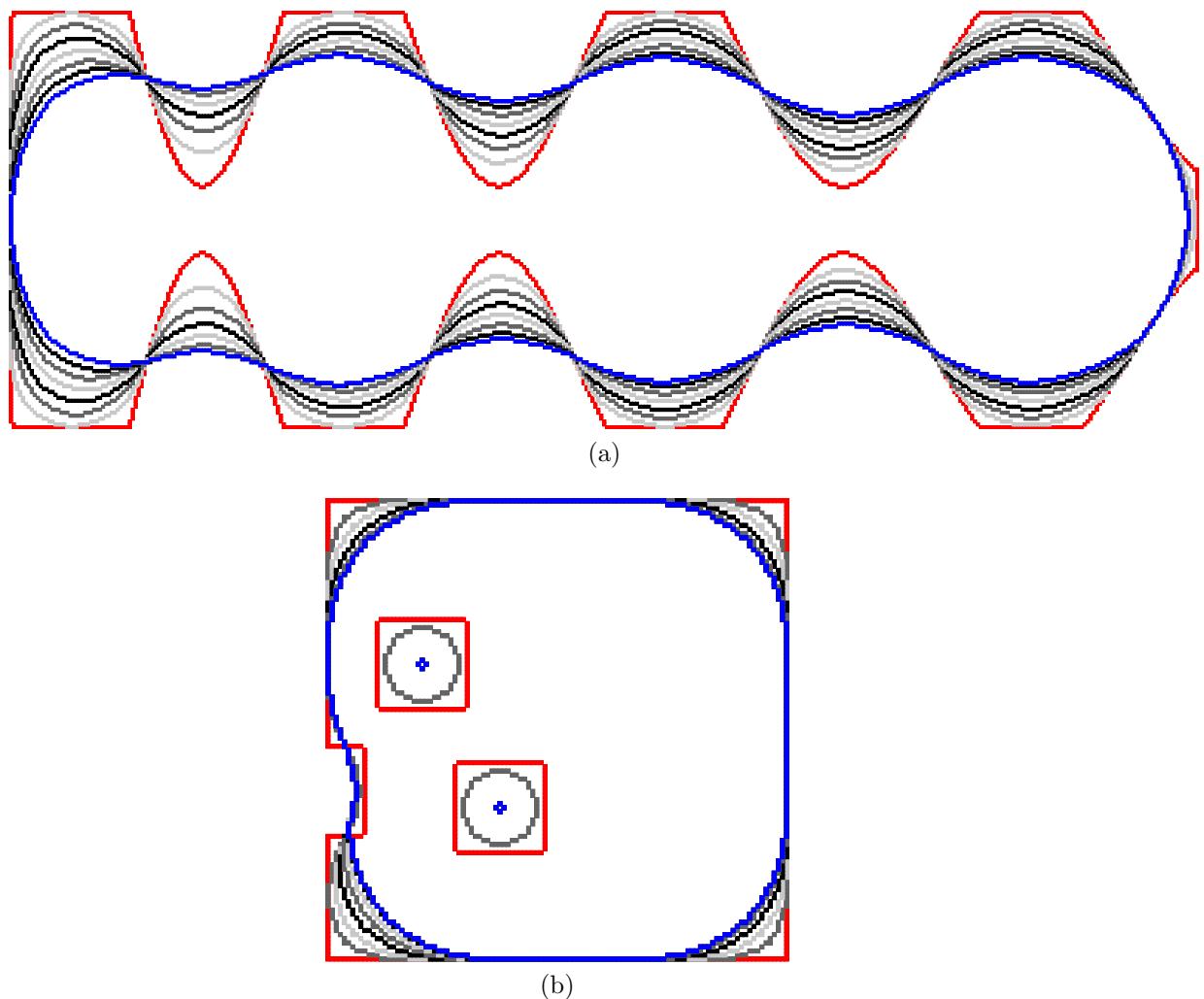


Figure 6.8: The flow evolves regions of high curvature faster, as illustrated in figure (a). Figure (b) illustrates the property of the FlipFlow algorithm to handle changes in topology.

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**;

```

 $D \leftarrow \text{GrabCut}(I, M);$ 
 $D^{(0)} \leftarrow \text{dilate}(D, d);$ 
 $\text{delta} \leftarrow +\infty;$ 
 $k \leftarrow 0;$ 
while  $k < \text{maxIt}$  and  $\text{delta} > \text{tolerance}$  do
    //Shrinking mode
    if  $k$  is even then
         $x^{(k-1)} \leftarrow \arg \min_{X^{(k-1)}} E_{(\boldsymbol{\theta}, m)}^{\text{flip}}(D^{(k-1)}, 1 - X^{(k-1)}) + \gamma g(X^{(k-1)});$ 
         $D^{(k)} \leftarrow F^{(k-1)} + x^{(k-1)};$ 
    end
    //Expansion mode
    else
         $\bar{x}^{(k-1)} \leftarrow \arg \min_{\bar{X}^{(k-1)}} E_{(\boldsymbol{\theta}, m)}^{\text{flip}}(D^{(k-1)}, 1 - \bar{X}^{(k-1)}) + \gamma g(\bar{X}^{(k-1)});$ 
         $D^{(k)} \leftarrow \frac{\bar{F}^{(k-1)} + \bar{x}^{(k-1)}}{\bar{F}^{(k-1)} + \bar{x}^{(k-1)}},$ 
    end
     $\text{delta} \leftarrow |D^{(k)} \setminus D^{(k+1)}|;$ 
     $k \leftarrow k + 1;$ 
end
```

Algorithm 4: FlipFlow algorithm for segmentation.

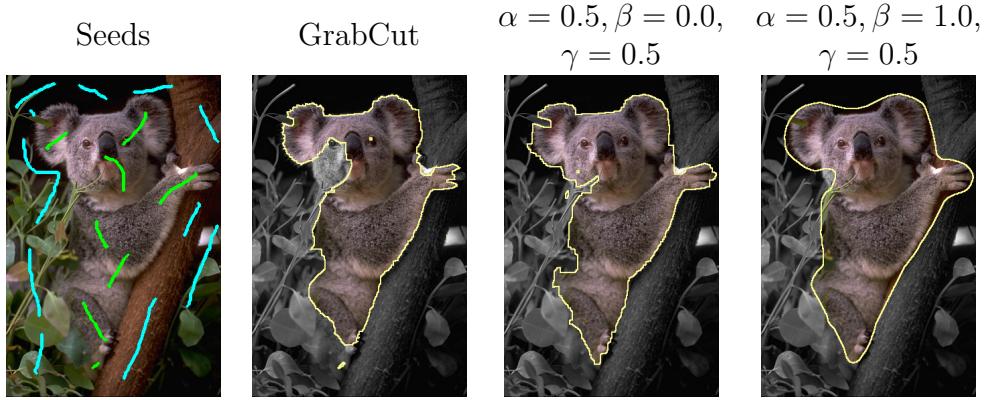


Figure 6.9: Given foreground (green) and background (blue) seeds at picture (a); GrabCut produces picture (b) which is used as input of the Contour Correction algorithm; in pictures (c) and (d) we display the output of Contour Correction algorithm with and without squared curvature regularization.

6.5 Conclusion

We presented the FlipFlow model in [Section 6.1](#), an evolution process that produces shapes of decreasing digital Elastica until a certain inflexion point. [Algorithm 3](#) operates in two distinct modes: the shrinking and the dilation modes. They are responsible for the treatment of convexities and concavities regions, respectively.

[Section 6.2](#) briefly describes some properties of QPBO method and its variants used to solve the quadratic binary problem that appears at every iteration of [Algorithm 3](#). In [Section 6.3](#) we point out the submodularization effect of evaluating the FlipFlow model at outer m -rings. For larger values of m , all pixels are labeled by QPBOP. Finally, we extended [Algorithm 3](#) in order to include a data term regularization and do image segmentation using [Algorithm 4](#).

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 $|B_r(p) \cap D| - |B_r(p) \cap \bar{D}|$ is closer to zero for every pixel $p \in \partial D$. We formalize this idea by defining the balance coefficient.

Definition 1(Balance coefficient): 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 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 keep or remove 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:

$$\begin{aligned} O^{(k)} &:= \{p \in \Omega \mid -1 \leq d_{D^{(k)}}(p) \leq 1\} \\ X^{(k)} &:= X(O^{(k)}) \\ F^{(k)} &:= D^{(k)} \setminus O^{(k)} \\ F_r^{(k)}(p) &:= F^{(k)} \cap B_r(p) \\ O_r^{(k)}(p) &:= O^{(k)} \cap B_r(p) \\ X_r^{(k)}(p) &:= X(O_r^{(k)}(p)). \end{aligned}$$

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

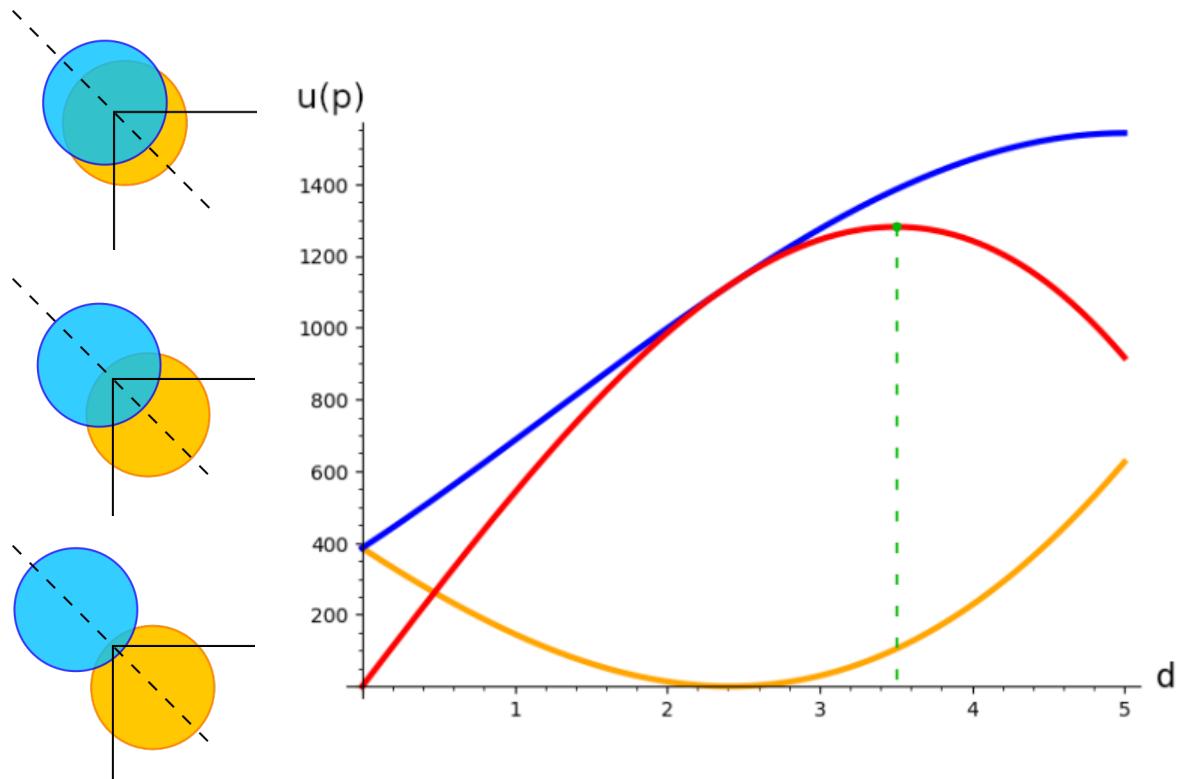


Figure 7.1: The balance coefficient ($r = 5$) of equidistant points from the corner of a square shape contour. The red curve indicates the difference between the blue and the orange curve. The inflexion point in the red curve indicates the point p in which a ball centered at p has balance coefficient equal to zero. The green point marks the point p in which the difference between the balance coefficients is the greatest.

$$u_r(F, X, p) = \left(\frac{\pi r^2}{2} - |F_r(p)| - \sum_{x_j \in X_r(p)} x_j \right)^2.$$

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

$$\begin{aligned} T_m^{bal}(D, X) &= \sum_{p_i, p_o \in \mathcal{R}_m(D)} u_r(F, 1 - X, p_o) - u_r(F, X, p_i) \\ &= \sum_{p_i, p_o \in \mathcal{R}_m(D)} \left(\frac{\pi R^2}{2} - \left(|F_o| + \sum_{x_j \in X_o} 1 - x_j \right) \right)^2 - \left(\frac{\pi R^2}{2} - \left(|F_i| + \sum_{x_j \in X_i} x_j \right) \right)^2, \end{aligned} \quad (7.1)$$

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 \(7.1\)](#) favors configurations of zero difference between the outer and inner ball coefficients, i.e., regions of zero curvature. In regions of positive curvature, the outer (inner) ball has a shortage (excess) of pixels, and minimization of [Equation \(7.1\)](#) tends to label variables in these regions to zero (one).

The m -balance energy family is defined as

$$E_{(\theta, m)}^{bal}(D^{(k)}, X^{(k)}) = \sum_{x_j \in X^{(k)}} \alpha s(x_j) + \beta T_m^{bal}(D^{(k)}, X^{(k)}) + \sum_{x_j \in X^{(k)}} \gamma g(D^{(k)}, x_j). \quad (7.2)$$

We follow the same notation of [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 5](#) 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 proposition

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 ;

```

 $D^{(0)} \leftarrow D;$ 
 $k \leftarrow 1;$ 
while  $k < \text{maxIt}$  do
     $x^{(k-1)} \leftarrow \arg \min_{X^{(k-1)}} E_{(\boldsymbol{\theta}, m)}^{\text{bal}}(D^{(k-1)}, X^{(k-1)}) + \gamma g(X^{(k-1)});$ 
     $D^{(k)} \leftarrow F^{(k-1)} + x^{(k-1)};$ 
     $k \leftarrow k + 1;$ 
end
```

Algorithm 5: BalanceFlow algorithm.

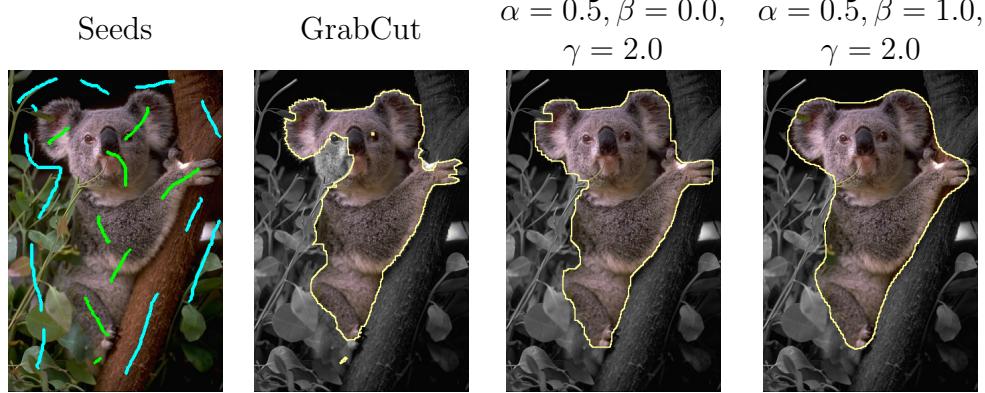


Figure 7.2: Given foreground (green) and background (gray) seeds at picture (a); GrabCut produces picture (b) which is used as input of the BalanceFlow algorithm; in pictures (c) and (d) we display the output of BalanceFlow algorithm with and without squared curvature regularization.

$$T_m^{flip}(D, X) = \sum_{p_i, p_o \in R_m(D)} \left(\frac{\pi R^2}{2} - F_r(p_o) - \sum_{x_j \in X_r(p_o)} x_j \right)^2 + \left(\frac{\pi R^2}{2} - F_r(p_i) - \sum_{x_j \in X_r(p_i)} x_j \right)^2 \quad (7.3)$$

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

$$T_m^{flip}(D, 1 - X) = \sum_{p_i, p_o \in R_m(D)} \left(\frac{\pi R^2}{2} - F_o - \sum_{x_j \in X_o} 1 - x_j \right)^2 + \left(\frac{\pi R^2}{2} - F_i - \sum_{x_j \in X_i} 1 - x_j \right)^2 \quad (7.4)$$

Next, let $A_i = \pi R^2 / 2 - |F_i|$. We rewrite the second term of [Equation \(7.4\)](#) as

$$\begin{aligned} \left(A_i - \sum_{x_j \in X_i} (1 - x_j) \right)^2 &= \left(A_i - |X_i| + \sum_{x_j \in X_i} x_j \right)^2 \\ &= (A_i - |X_i|)^2 + 2(A_i - |X_i|) \sum_{x_j \in X_i} x_j + \left(\sum_{x_j \in X_i} x_j \right)^2 \\ &= A_i^2 - 2A_i|X_i| + |X_i|^2 + 2(A_i - |X_i|) \sum_{x_j \in X_i} x_j + \left(\sum_{x_j \in X_i} x_j \right)^2 \\ &= A_i^2 + 2A_i \sum_{x_j \in X_i} x_j + \left(\sum_{x_j \in X_i} x_j \right)^2 - 2A_i|X_i| + |X_i|^2 - 2|X_i| \sum_{x_j \in X_i} x_j \\ &= 2A_i^2 - \left(A_i - \sum_{x_j \in X_i} x_j \right)^2 + 2 \left(\sum_{x_j \in X_i} x_j \right)^2 - 2A_i|X_i| + |X_i|^2 \\ &\quad - 2|X_i| \sum_{x_j \in X_i} x_j \end{aligned}$$

We group the constants into the constant term $c = 2A_i^2 - 2A_i|X_i|$ to obtain

$$\begin{aligned} \left(A_i - \sum_{x_j \in X_i} (1 - x_j) \right)^2 &= -\left(A_i - \sum x_j \right)^2 + \left(|X_i| - \sum_{x_j \in X_i} x_j \right)^2 + \left(\sum_{x_j \in X_i} x_j \right)^2 + c \\ &= -\left(A_i - \sum_{x_j \in X_i} x_j \right)^2 + P_1(X_i) + c \\ &= -\left(\frac{\pi R^2}{2} - (|F_i| + \sum_{x_j \in X_i} x_j) \right)^2 + P_1(X_i) + c \end{aligned}$$

Finnaly, we replace [Equation \(7.5\)](#) in [Equation \(7.4\)](#) to obtain

$$\begin{aligned} T_m^{flip}(D, 1 - X) &= \left(\frac{\pi R^2}{2} - \left(|F_o| + \sum_{x_j \in X_o} 1 - x_j \right) \right)^2 \\ &\quad - \left(\frac{\pi R^2}{2} - \left(|F_i| + \sum_{x_j \in X_i} x_j \right) \right)^2 + P_1(X_i) + c \\ &= T_m^{bal}(D) + P_1(X_i) + c. \end{aligned} \tag{7.5}$$

■

[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 Towards a graph-based formulation

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.

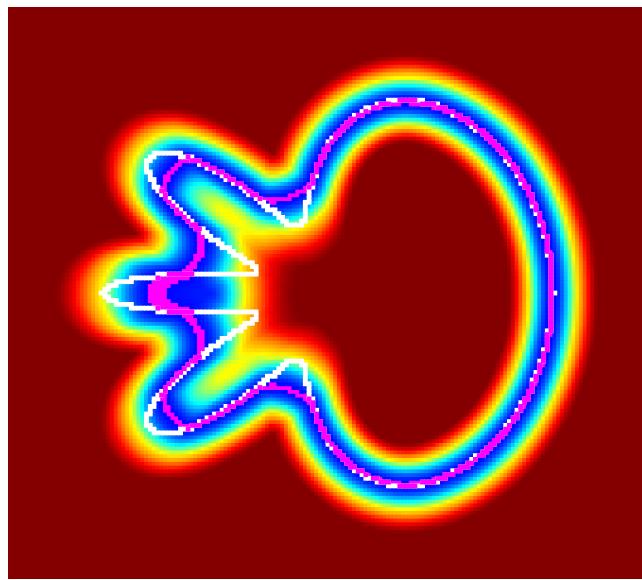


Figure 7.3: Balance coefficient map of radius 12 for the flower shape (contour in white). The pixels in magenta have balance coefficient smaller than 169.

7.4 Conclusion

The BalanceFlow model is closely related to the FlipFlow but it has a simple implementation and interpretation. The balance coefficient is the link that connects both models and the motivation for the graph-cut model proposed in the next chapter.

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-based 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 Standard graph-cut segmentation

The Grabcut model [RKB04] partitions an image in foreground and background components by computing minimum cuts in a graph. The model is very attractive, as minimum cuts are quickly computed for sparse graphs. We start by giving some necessary definitions from graph theory.

8.1.1 Definitions

Definition 1(Unordered graph): The unordered graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a pair of sets \mathcal{V}, \mathcal{E} such that \mathcal{V} is finite and \mathcal{E} is a collection of unordered pairs of \mathcal{V} , i.e.,

$$\mathcal{E} \subseteq \{ \{x, y\} \mid x, y \in \mathcal{V} \}.$$

Definition 2(Connected graph): The graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is connected if for every pair of vertices v, u there exists a path π of length n such that

$$v_1 = v, \quad v_n = u, \quad \pi = \{v_i \mid \{v_{i-1}, v_i\} \subset \mathcal{E}\}, \quad 2 < i < n.$$

A digital set $D \subset \Omega \subset \mathbb{Z}^2$ is naturally represented as a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ by letting each pixel $p \in D$ correspond to a vertex $v_p \in \mathcal{V}$. The set of edges is defined according to the application in mind, but it's encodes the pixels connectivity, i.e., 4, 8-adjacency. Moreover, one can define a cost function $w : \mathcal{E} \rightarrow \mathbb{R}$ on the edge set. In this case, we speak of a weighted graph.

Definition 3(Cut): Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ a connected graph weighted by cost function $w : \mathcal{E} \rightarrow \mathbb{R}$. A cut set is a subset $\mathcal{E}' \subset \mathcal{E}$ such that the graph $\mathcal{G}(\mathcal{V}, \mathcal{E} \setminus \mathcal{E}')$ is disconnected. Moreover, the cut value of \mathcal{E}' is defined as

$$\text{cut}(\mathcal{E}') = \sum_{e \in \mathcal{E}'} w(e).$$

We are interested in (s, t) -cuts, i.e., cuts in which vertices s, t are disconnected after the cut set is removed. We denote s, t as the source and target vertices, respectively.

The Grabcut model consists in a judicious definition of the cost function w in order to segment objects in a digital image according to its color intensity values.

8.1.2 Grabcut binary segmentation model

Let $I : \Omega \rightarrow [0, 1]^3$ be a 3-channel colored image. Consider the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ defined as

$$\begin{aligned} \mathcal{V} &= \{v_p \mid \forall p \in \Omega\} \cup \{s, t\} \\ \mathcal{E} &= \mathcal{E}_{st} \cup \mathcal{E}_{\mathcal{N}}, \end{aligned}$$

where

$$\begin{aligned} \mathcal{E}_{st} &= \{\{s, v_p\}, \{t, v_p\} \mid \forall p \in \mathcal{V}\} \\ \mathcal{E}_{\mathcal{N}} &= \{\{v_p, v_q\} \mid \forall p, q \in \mathcal{V} \text{ and } q \in \mathcal{N}_4(p)\}. \end{aligned}$$

Next, let sets $\mathcal{V}_{\mathcal{F}}, \mathcal{V}_{\mathcal{B}} \subset \mathcal{V}$ to represent foreground and background seeds input by the user. Vertices in $\mathcal{V}_{\mathcal{F}}$ ($\mathcal{V}_{\mathcal{B}}$) are forced to be connected to the source (target) after the removal of the minimum cut set.

Consider the functions

$$B(\{v_p, v_q\}) = \exp\left(-\frac{(I(p) - I(q))^2}{2\sigma^2}\right)$$

$$R(v_p, "bg") = -\ln H_B(p)$$

$$R(v_p, "fg") = -\ln H_F(p),$$

where H_B, H_F are mixed Gaussian distributions derived from foreground and background seeds given by the user; and σ is interpreted as a parameter to configure noise level of the input image.

Finally, define the cost function $w : \mathcal{E} \rightarrow \mathbb{R}$ as

edge e	$w(e)$	for
$\{v_p, v_q\}$	$B(e)$	$e \in \mathcal{E}_N$
$\{v_p, s\}$	$\lambda_r R(v_p, "bg")$	$p \in \mathcal{V}, p \notin \mathcal{V}_F \cup \mathcal{V}_B$
	M	$p \in \mathcal{V}_F$
	0	$p \in \mathcal{V}_B$
$\{v_p, t\}$	$\lambda_r R(v_p, "fg")$	$p \in \mathcal{V}, p \notin \mathcal{V}_F \cup \mathcal{V}_B$
	0	$p \in \mathcal{V}_F$
	M	$p \in \mathcal{V}_B$

$$\text{where, } M = 1 + \max_{p \in \Omega} \sum_{q \in \mathcal{N}_4(p)} B(\{p, q\}).$$

Let \mathcal{E}' be a minimum cut of \mathcal{G} and C the connected component of \mathcal{G} induced by \mathcal{E}' that contains the source vertex. The grabcut energy value is given by

$$grab_\Lambda(C) = \lambda_r \cdot \left(\sum_{p \in C} R(v_p, "fg") + \sum_{p \in \bar{X}} R(v_p, "bg") \right) + \lambda_b \cdot \sum_{e \in \mathcal{E}'} B(e).$$

The Grabcut segmentation is heavily impacted by the quality of the input image. To attenuate the problem, one could extend the Grabcut model by defining a cost function that takes geometric information into account. Indeed, the work [BK03] describes a cost function in which one could do image segmentation with length

regularization. There exist some attempts [Nie+14; EG10] to inject curvature information in a graph-cut framework, but the models present issues on running time and lack precision on the estimation of curvature. In the next section we describe a graph-cut model that uses the balance coefficient to regularize curvature.

8.2 GraphFlow model

Given some energy E , the GraphFlow model is divided into two main steps: candidate selection and validation. At the candidate selection step, a minimum cut is computed for each member of a set of candidate graphs. From the minimum cuts we derive candidate solutions for the minimization of E . The validation step consists in selecting the candidate solution with the lowest value of E .

The GraphFlow model is suitable for the free, constrained Elastica and image segmentation problems. In particular, the GraphFlow model is a simple extension of the Grabcut model for image segmentation that regularizes the squared curvature.

In the next sections, we are going to describe the GraphFlow model that aims to minimize the energy

$$E^{graph} = \hat{E}_{\theta} + grab_{\Lambda}. \quad (8.1)$$

8.2.1 The candidate graph

Let $D \subset \Omega \subset \mathbb{Z}^2$ a digital set. Given $n > 0$, we define the optimization band $O_n(D)$ as

$$O_n(D) := \{p \in \Omega \mid -n \leq d_D(p) \leq n\}.$$

We are going to construct the candidate graph $\mathcal{G}_n(D) = (\mathcal{V}, \mathcal{E})$. The vertex and edge sets are defined as

$$\begin{aligned} \mathcal{V} &= \{v_p \mid p \in D \cup O_n\} \cup \{s, t\} \\ \mathcal{E} &= \mathcal{E}_{st} \cup \mathcal{E}_{\mathcal{N}}. \end{aligned}$$

where s, t are the source and target vertices, respectively. As in the previous section, sets $\mathcal{V}_F, \mathcal{V}_B \subset \mathcal{V}$ correspond to foreground and background seeds input by the user. Additionally, we define the sets

$$\begin{aligned} \mathcal{V}_s &= \{v_p \mid p \in D \setminus O_n(D)\} \\ \mathcal{V}_t &= \{v_p \mid p \in \overline{D} \setminus O_n(D)\}. \end{aligned}$$

The sets $\mathcal{V}_s, \mathcal{V}_t$ are forced to be connected to source and target components, respectively. The cost function $w : \mathcal{E} \rightarrow \mathbb{R}$ is defined as

edge e	$\mathbf{w}(e)$	for
$\{v_p, v_q\}$	$\beta(u(D, p) + u(D, q)) + \lambda_b B(e)$	$p, q \in \mathcal{E}_{\mathcal{N}}$
$\{v_p, s\}$	$\lambda_r R(v_p, \text{"bg"})$ $\lambda_r M_1 \mathbb{1}_{\mathcal{V}_F}(p) + \beta M_2 \mathbb{1}_{\mathcal{V}_s}(p)$	$p \in \mathcal{V}, p \notin \mathcal{V}_F \cup \mathcal{V}_B \cup \mathcal{V}_s \cup \mathcal{V}_t$ otherwise
$\{v_p, t\}$	$\lambda_r R(v_p, \text{"fg"})$ $\lambda_r M_1 \mathbb{1}_{\mathcal{V}_B}(p) + \beta M_2 \mathbb{1}_{\mathcal{V}_t}(p)$	$p \in \mathcal{V}, p \notin \mathcal{V}_F \cup \mathcal{V}_B \cup \mathcal{V}_s \cup \mathcal{V}_t$ otherwise

The constants M_1, M_2 are defined as

$$M_1 = 1 + \max_{p \in \Omega} \sum_{q \in \mathcal{N}_4(p)} B(\{p, q\})$$

$$M_2 = 1 + \max w(p, q).$$

Let \mathcal{E}' the minimum cut of $\mathcal{G}_n(D)$. Moreover, let $C_n(D)$ be the connected component of the graph $\mathcal{G}_n(D)$ that contains the source vertex s and is restricted to $\mathcal{E} \setminus \mathcal{E}'$. The digital set

$$D' = \{p \mid v_p \in C_n(D)\}$$

is said to be a candidate shape to decrease [Equation \(8.1\)](#). We are going to use $C_n(D)$ interchangeably with D' to denote both the set of vertices and the corresponding points of a digital set. In the next section we describe a local-search algorithm whose neighborhood is defined using a set of candidate graphs.

8.2.2 GraphFlow algorithm

The GraphFlow algorithm implements a local-search strategy to minimize [Equation \(8.1\)](#). Its neighborhood is derived from a set of candidate graphs, called the probe-set. The probe-set of D can be defined as any collection of digital sets, but

it makes more sense to define it as a collection of digital shapes that are somehow close to D . For example, we define the a -probe set of shape D as

Definition 1(a -probe set): Let D 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(erotion) by a circle of radius a .

Therefore, we define the local-search neighborhood of digital set D as

$$\mathcal{N}(D) = \{C_n(X) \mid Q \in \mathcal{P}_a(D)\}.$$

Finally, the GraphFlow algorithm is described in [Algorithm 6](#).

```

input : A digital set  $S$ ; the optimization band  $n$ ; the probe set parameter  $a$ ;
        parameter vector  $\boldsymbol{\theta} = (\alpha, \beta)$ ; the maximum number of iterations
        maxIt;
D(0)  $\leftarrow D$ ;
k  $\leftarrow 1$ ;
while k < maxIt do
    //Candidate selection
    N(k)  $\leftarrow \{C_n(Q) \mid Q \in \mathcal{P}_a(D^{(k-1)})\}$ ;
    //Candidate validation
    D(k)  $\leftarrow \arg \min_{X \in \mathcal{N}^{(k)}} \hat{E}_{\boldsymbol{\theta}}(X) + grab_{\Lambda}(X)$ ;
    k  $\leftarrow \text{k} + 1$ ;
end
```

Algorithm 6: GraphFlow algorithm.

We remark that the GraphFlow algorithm possess two fundamental steps: the candidate selection and the candidate validation. The former can be interpreted as an advanced filter that choses the best candidates for digital Elastica minimization from the given a -probe set. Among those choices, we chose the one with lower digital Elastica energy in the candidate validation step.

The GraphFlow algorithm can grow or shrink in accordance with the α coefficient in the digital Elastica (see [Figure 8.1](#)). Using a 0-probe set we recover the convergence to a single point behaviour observed in both FlipFlow and Balance-Flow model. Moreover, its solution for the free Elastica problem is very similar to those given by the enumerative process of [Chapter 5](#), i.e., the shapes converge

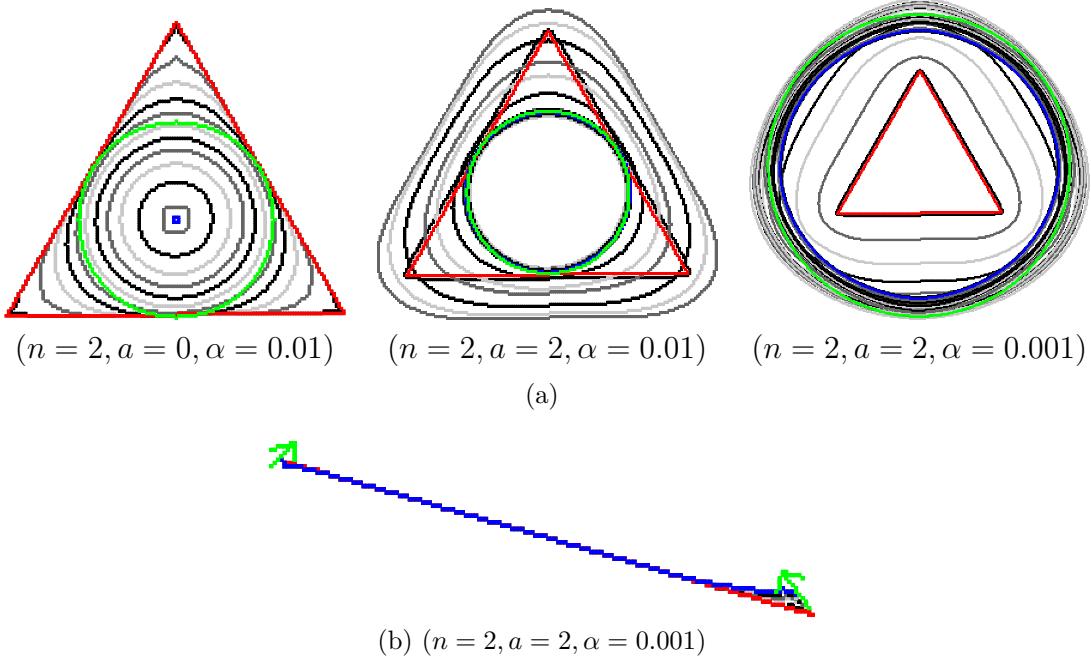


Figure 8.1: The GraphFlow algorithm can shrink and grow in accordance with length penalization and it converges to shape closer to the global optimum (green curve) in the free Elastica problem. In the constrained Elastica, we believe that we can improve the results by using a larger, possibly random, neighborhood. We are using $n = 2, a = 2$ and shapes are displayed at every 10 iterations.

to the expected global optimum. However, for the constrained Elastica problem, the GraphFlow encounters some difficulties to evolve (see Figure 8.1), in particular for the fixed endpoint orientation instance. We believe that a larger neighborhood, possibly random, could solve this issue. The results are explored in more details in Chapter 9.

Some preliminary results of Algorithm 6 applied to image segmentation is shown in Figures 8.2 and 8.3. In particular, we can observe that the GraphFlow presents the completion property, i.e., tends to return segmentation with fewer disconnected components.

8.3 Conclusion

We described a graph-cut model that regularizes the squared curvature, in the free, constrained Elastica and 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 . Moreover, the GraphFlow Al-

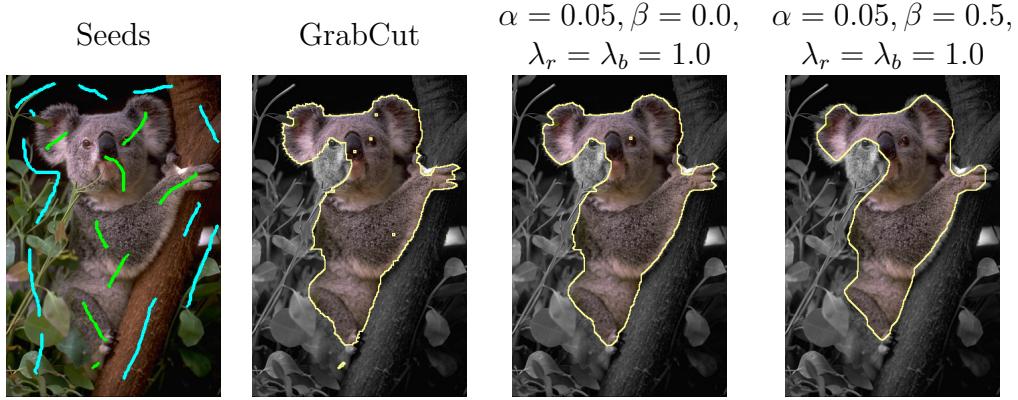


Figure 8.2: Given foreground (green) and background (gray) seeds at picture (a); GrabCut produces picture (b) which is used as input of the GraphFlow algorithm; in pictures (c) and (d) we display the output of GraphFlow algorithm with and without squared curvature regularization.

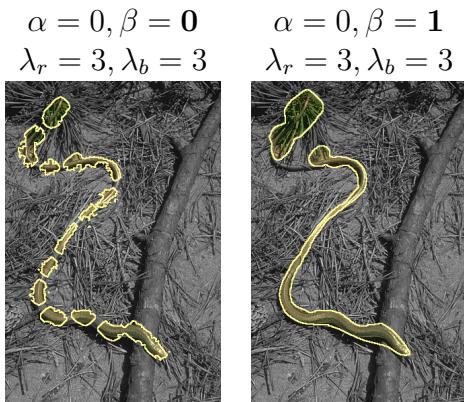


Figure 8.3: Completion property. The oversegmentation picture in the left was obtained with no squared curvature regularization, while the picture in the right was obtained by setting $\beta = 1.0$.

gorithm 6 is faster and simpler to implement than the previous models presented in this thesis.

Chapter 9

Results analysis

In this chapter we analyse the results produced by the four models developed in this thesis. We are going to compare its results 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 Schoenemann.

Model	Implementation	Running time	Free Ela.	Constrained Ela.	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 behaviour 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.

Experiment	<i>maxIt</i>	<i>vRadius</i>	<i>bRadius</i>	<i>h</i>	α	β	<i>nc</i>	LS	FF,BF	GF
								<i>m</i>	<i>a</i>	<i>ob</i>
Exp-General	400	5	7	0.25	0.01	1	4	5	2	3
Exp-Radius	400	5	7 12	0.25	0.001	1	4	5 10	2	3

Table 9.2: Parameter settings for the free Elastica experiments. The headers LS,FF,BF,GF identifies parameters that are exclusive for the LocalSearch, 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: Running time and input size of Exp-General experiment for the free Elastica.

9.1.1 General experiment

The General experiment executes each model using the listed parameters in [Table 9.2](#) for 5 different parametric shapes. The results for the General experiment is 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, i.e., for $\alpha = 0.01$, 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 first than LocalSearch in two occasions, one in the square and the other 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.

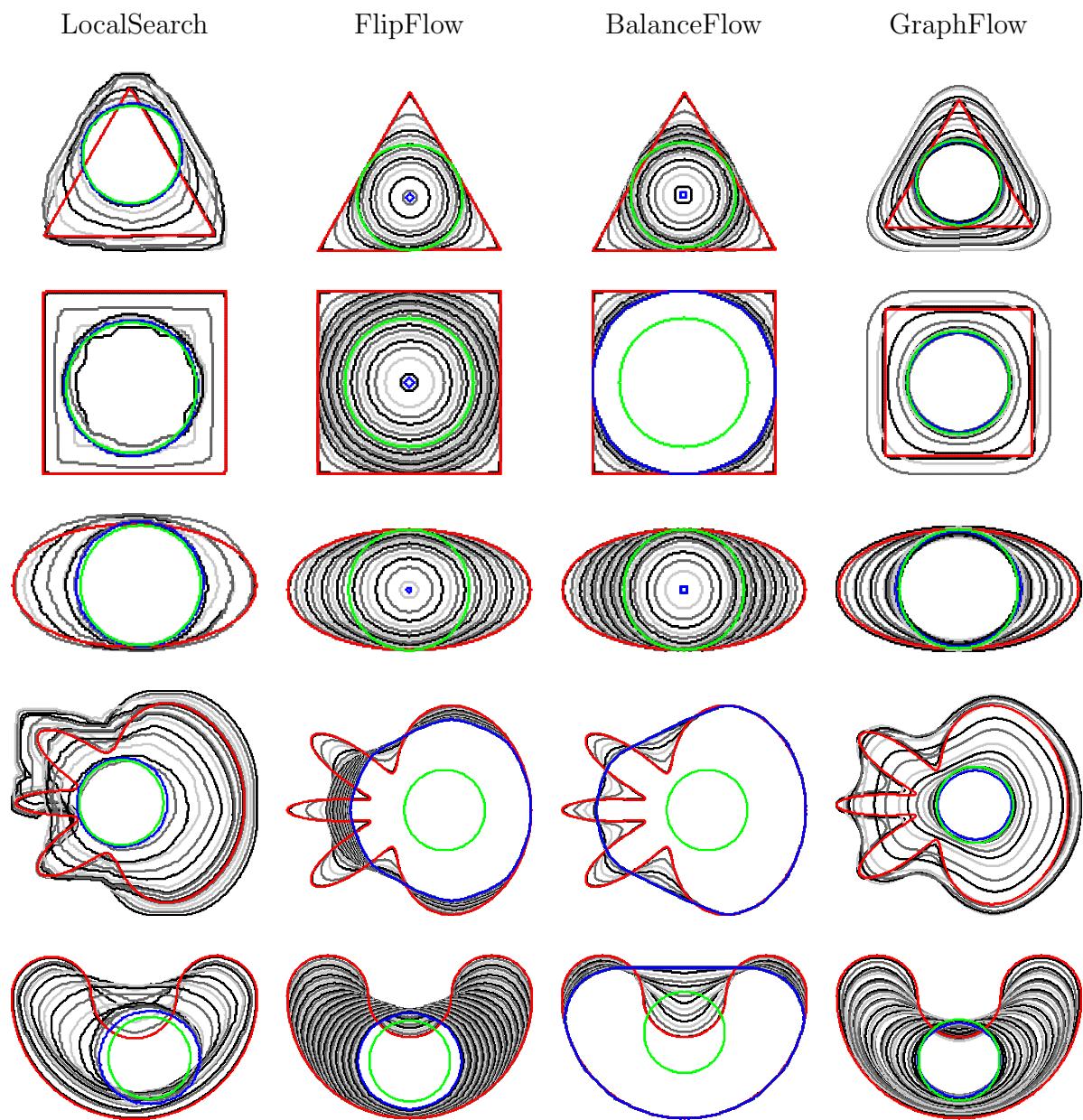


Figure 9.1: Results of Exp-General experiment for the free Elastica. Initial contour is colored in red, final contour is colored in blue and optimal contour is colored in green. Curves are drawn every 10 iterations.

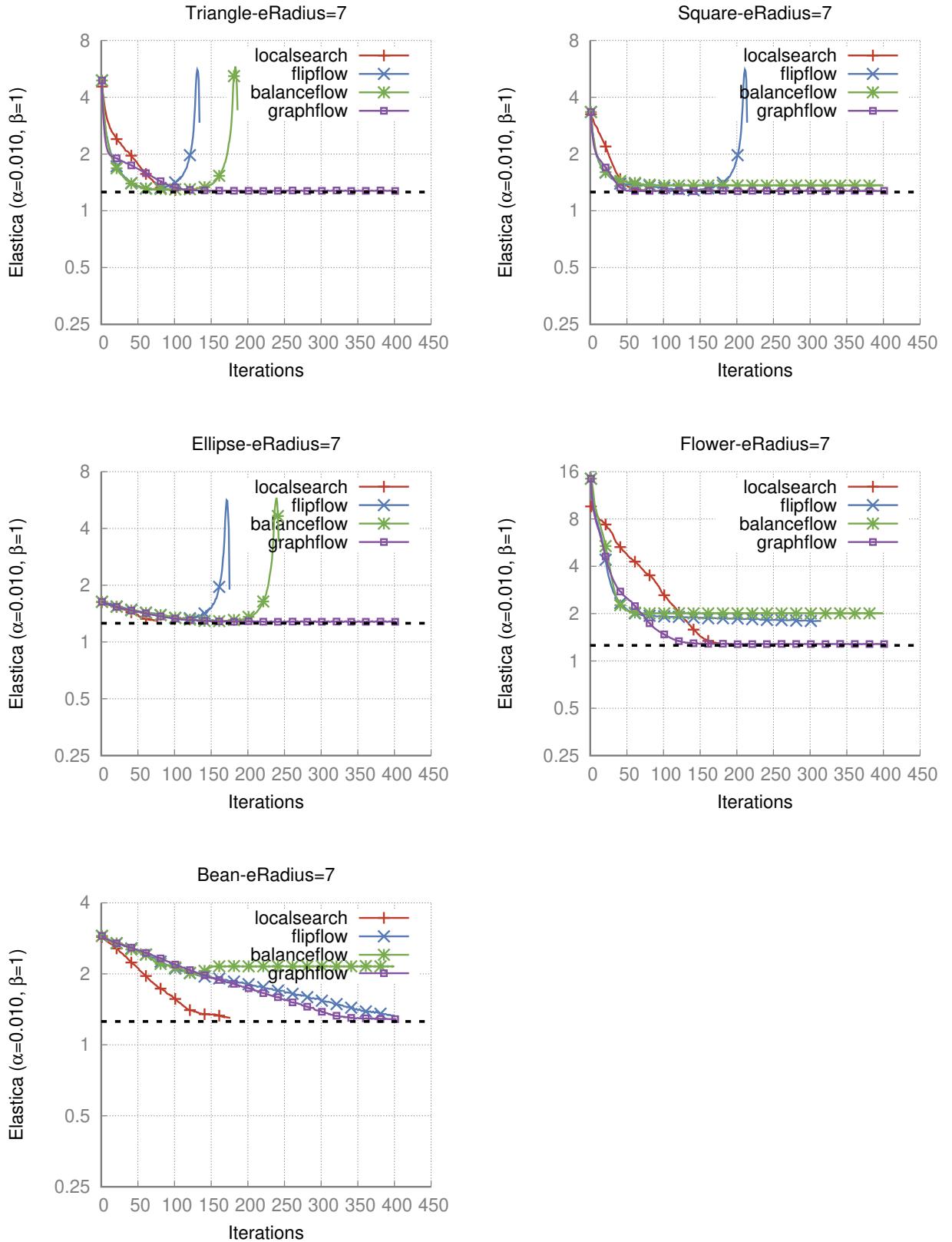


Figure 9.2: Evolution of digital Elastica value per iteration in the Exp-General experiment for the free Elastica. The dashed line marks the optimum value.

9.1.2 Radius choice experiment

In the Exp-Radius experiment, we set the length penalization parameter to $\alpha = 0.001$. Compared to the Exp-General, the expected behaviour is that the shapes will 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 is important to the optimal shape achievement.

In the case of FlipFlow and BalanceFlow, the evolution goes faster with a larger radius, but the shape never grows, it only shrinks. On the other hand, LocalSearch and GraphFlow are sensitive to the value of α and they can grow or shrink the shape accordingly. Moreover, the choice of *bRadius* defines how closer the solution will be from the optimum.

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 use a radius larger than the shape reach at the point of estimation.

However, for the Radius-choice experiment, a value *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 both LocalSearch and GraphFlow return solutions closer to the optimal as we can check in [Figures 9.3](#) and [9.4](#).

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 with two different set 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 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 opposition with the previous section where all the shapes had a closed parametric formula, some of the tested shapes in this section are ad-hoc and a decision about the grid step in this case becomes arbitrary.

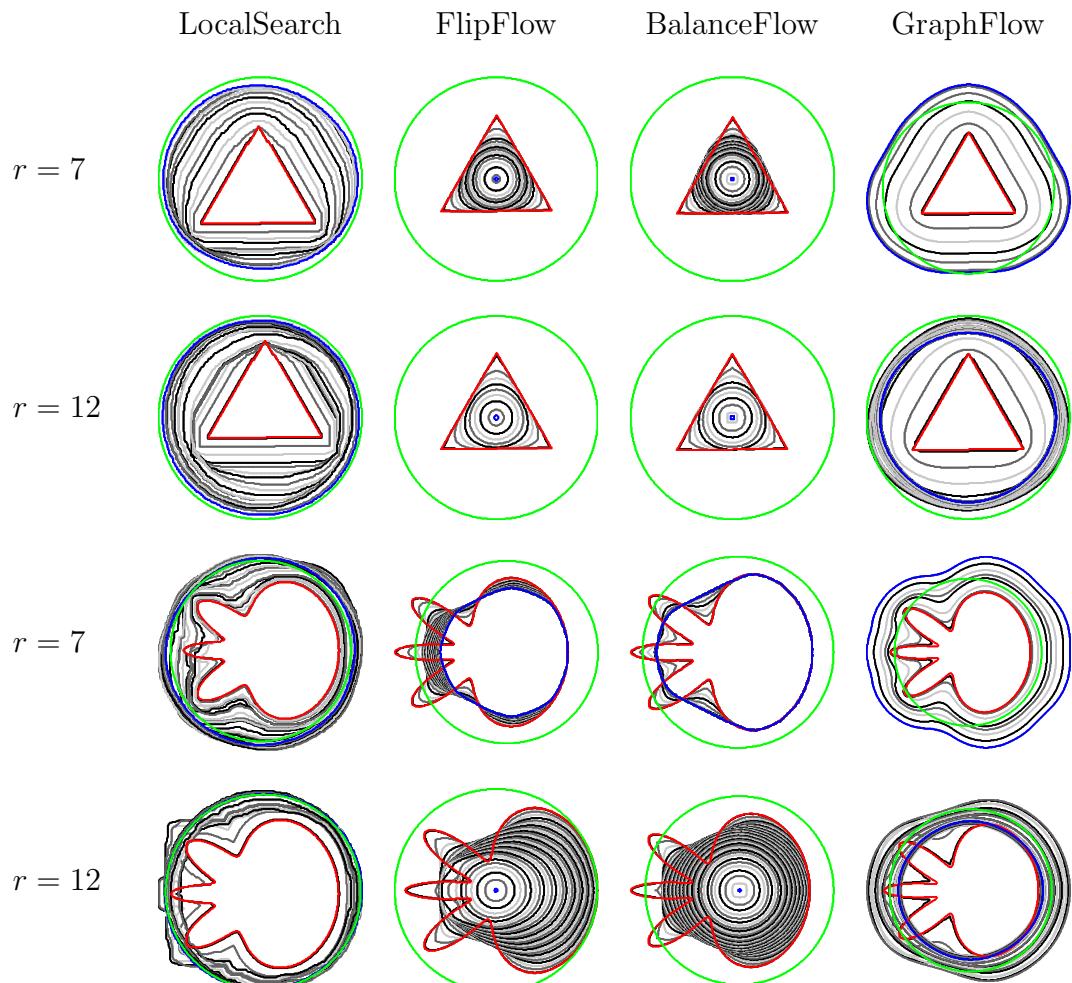


Figure 9.3: Results of Exp-Radius experiment for the free Elastica. Initial contour is colored in red, final contour is colored in blue and optimal contour is colored in green. Curves are drawn every 10 iterations.

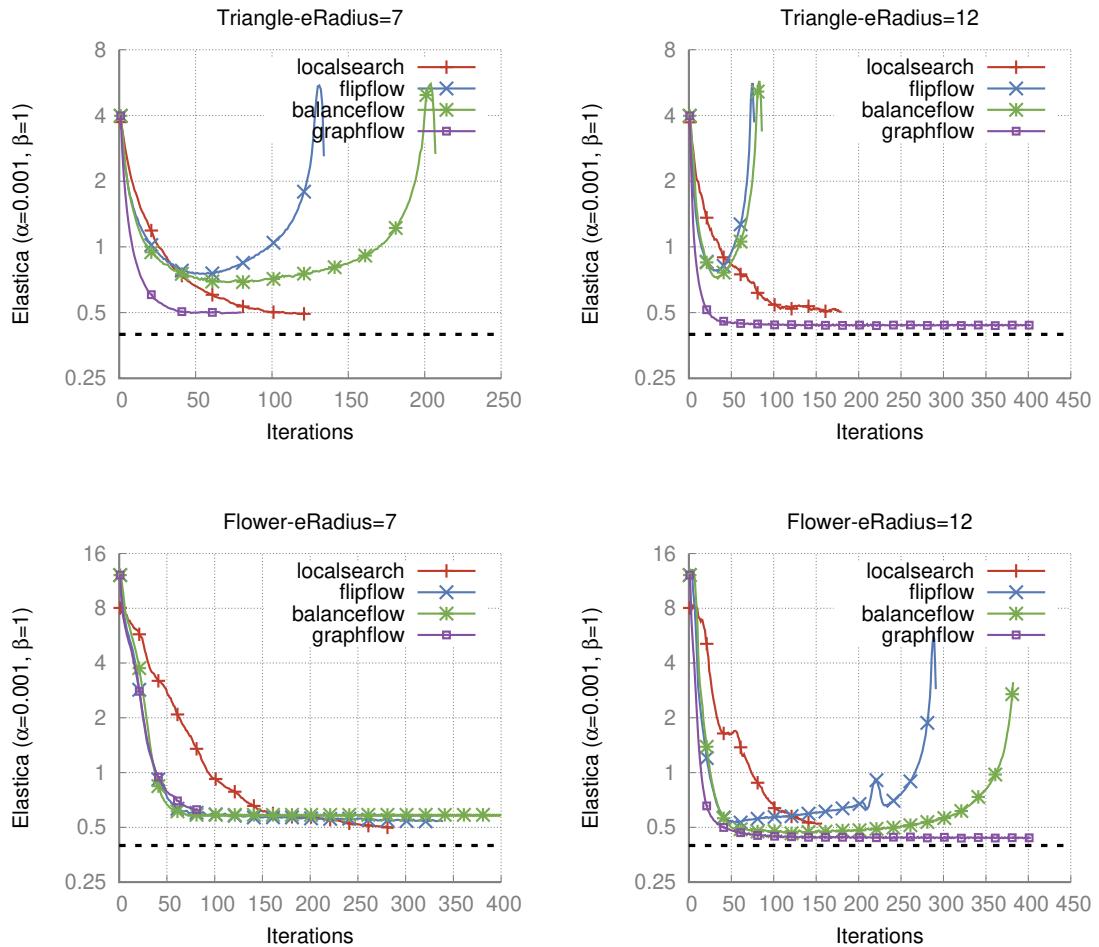


Figure 9.4: Digital Elastica value evolution per iteration of the Radius-choice experiment for the free Elastica.

Experiment	<i>maxIt</i>	<i>vRadius</i>	<i>bRadius</i>	<i>h</i>	α	β	LS	GF	
							<i>nc</i>	<i>a</i>	<i>ob</i>
Exp- α	400	15	15	1.0	0.002 0.0002	1	4	1	2
Exp- <i>bRadius</i>	400	7 50	7 50	1.0	0.0002	1	4	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.

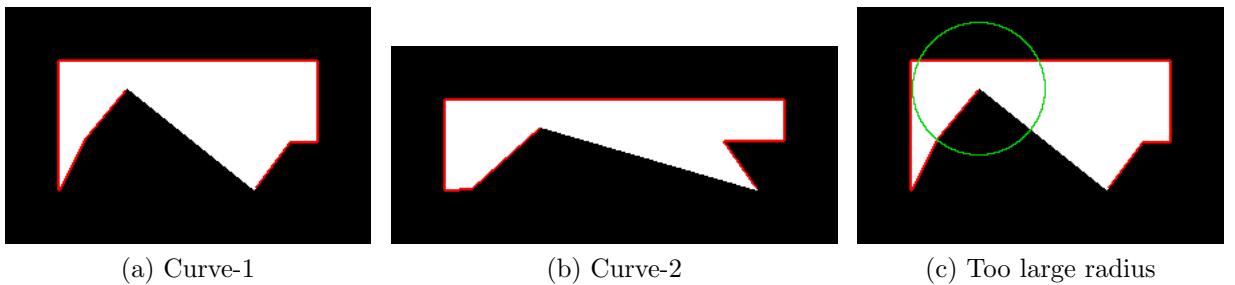


Figure 9.5: The underlying shapes of Curve-1 and Curve-2 for constrained Elastica experiments are shown in figures (a) and (b). Pixels in red are forced to persist in the solution. In figure (c), an example of a too large radius value (50).

9.2.1 Discussion

Differently from LocalSearch, the GraphFlow model encounters some difficulties to evolve both Curve-1 and Curve-2, as illustrated in Figure 9.6. We recall that the LS model employs a more heterogeneous neighborhood than GF. The probe set of GF is created by applying erosions and dilations in the initial shape, which promotes quite rough transformations, i.e., they do not preserve any part of the contour. We believe that by refining the neighborhood, possibly a random one, we could obtain better results for the GraphFlow. Besides that, the GF evolves the Flower instances, though it stops at a shape of higher digital Elastica value than LS.

In Figure 9.7 displays the results of experiment Exp-*bRadius* in which the models are executed with different values for the radius of the estimation disk. As expected, a larger radius can estimate a larger range of curvature values and it is particularly important in the case where $\alpha = 0.0002$. In this case the shape tends to grow and the estimator should be precise enough to identify small variations in curvature. Otherwise, the shape prematurely stops to evolve. Nonetheless, a too large radius (see Figure 9.5) may not be appropriated, and the curves do not evolve as expected.

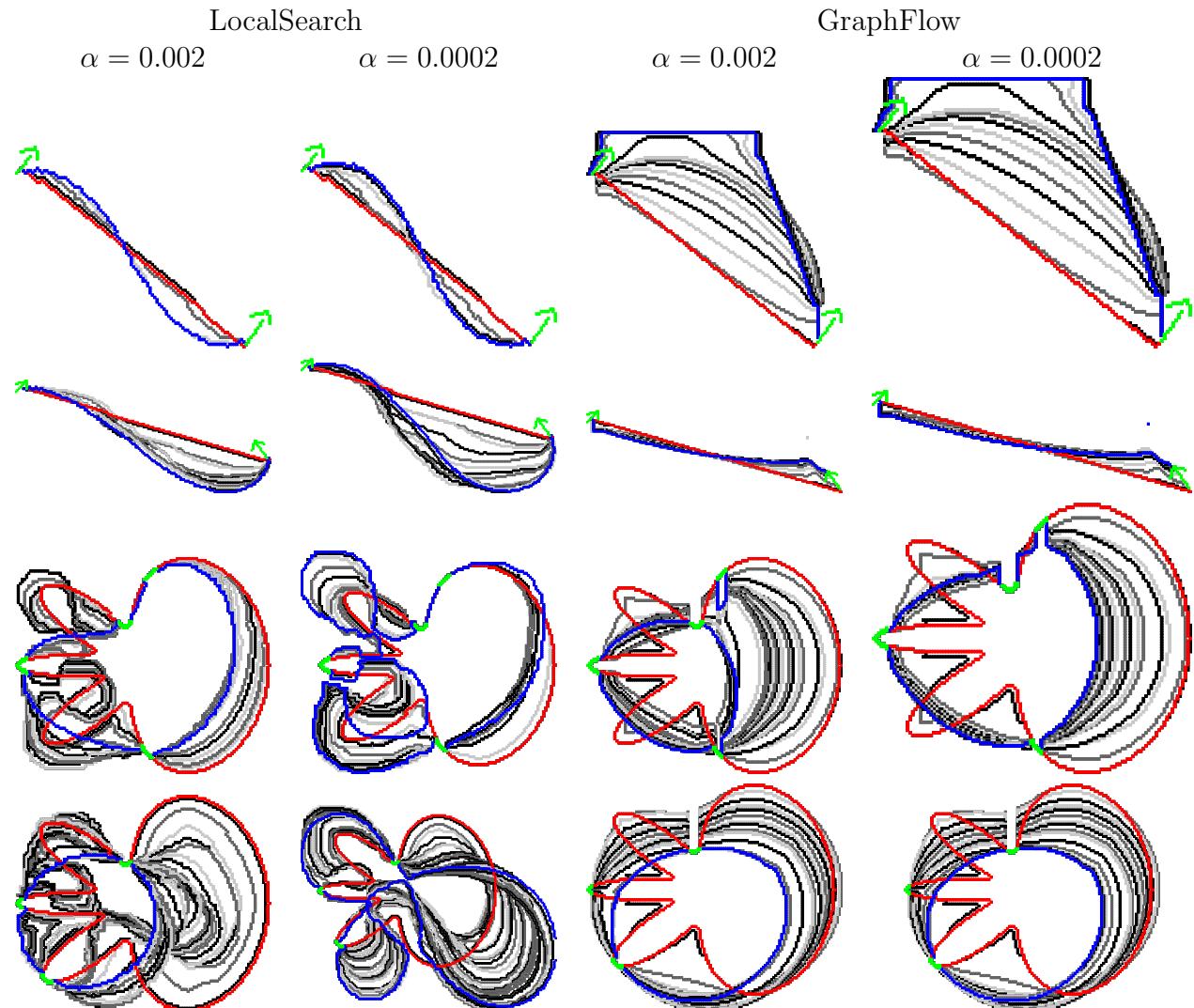


Figure 9.6: Results of $\text{Exp-}\alpha$ for the constrained Elastica. Initial contour is colored in red and final contour is colored in blue. The green pixels indicates pixels forced to persist in the final contour, or a forced orientation. Curves are drawn every 10 iterations.

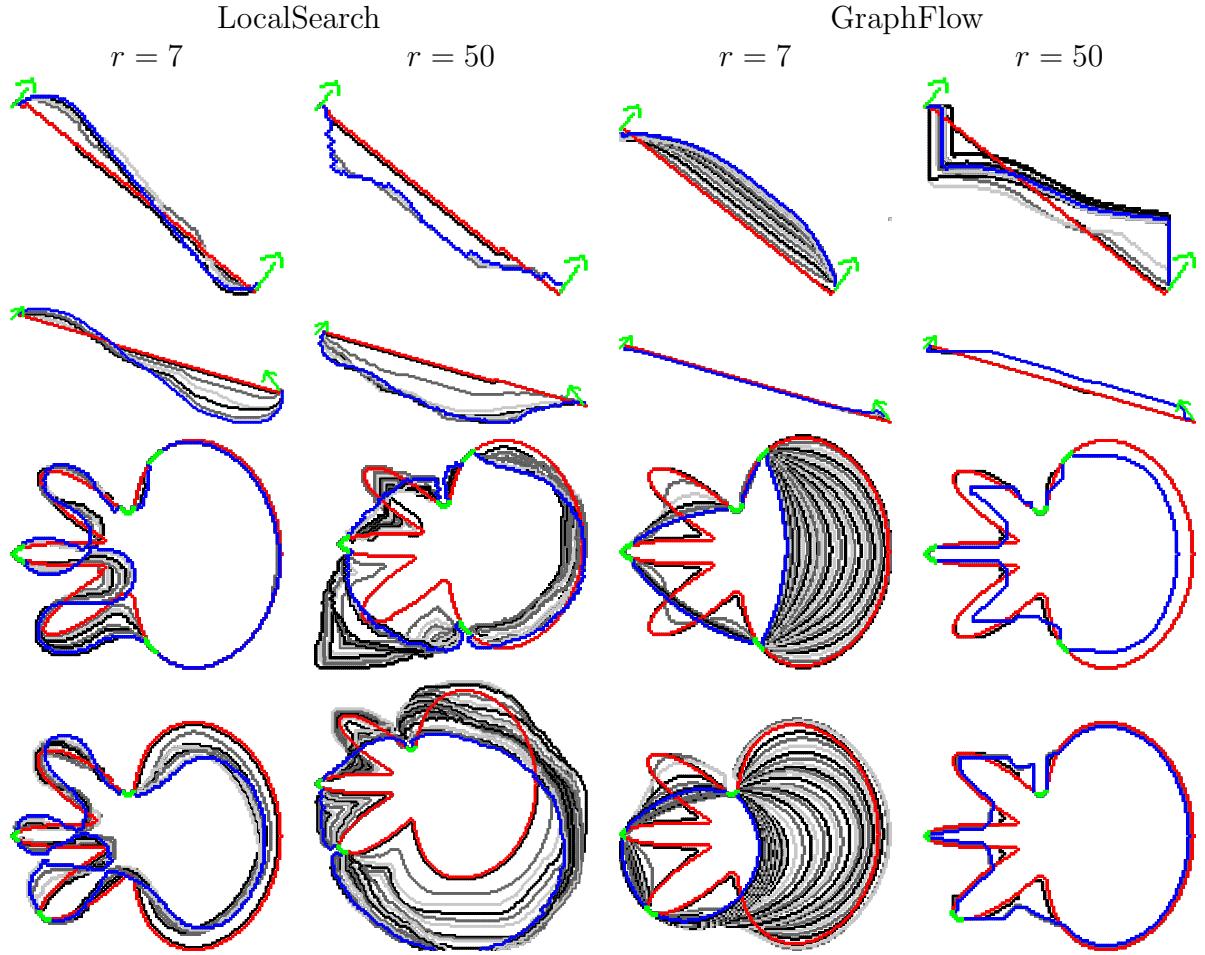


Figure 9.7: Results of *Exp-bRadius* for the constrained Elastica. Initial contour is colored in red and final contour is colored in blue. The green pixels indicates pixels forced to persist in the final contour, or a forced orientation. Curves are drawn every 10 iterations.

	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 column 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

Experiment	<i>maxIt</i>	<i>vRadius</i>	<i>bRadius</i>	<i>h</i>	α	β	FF,BF		GF			
							γ	<i>d</i>	<i>a</i>	<i>ob</i>	λ_r	λ_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	1
Exp- <i>rbRadius</i>	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.

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 the weights parameters (length, curvature, data) and the radius of the estimation ball in the produced segmentation. At the last section we compare our results with the Schoenemann linear curvature regularization (SLCR) [SKC09].

All three models (FF,BF,GF) need a initial segmentation as input. This segmentation is given by a single iteration of the Grabcut 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 $h = 1.0$.

9.3.1 Parameters influence

The models offer parameters to control the relative weight of length (α), curvature (β) and data ($\gamma, \lambda_r, \lambda_b$). We recall that FF and BF accept a single regional parameter γ for data, while GF accepts λ_r to ponderate a regional term and λ_b to weight

Exp-Comparison													
Model	maxIt	vRadius	bRadius	h	α	$\beta(\lambda)$	γ	d	a	ob	λ_r	λ_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 $bRadius = 7$.

a boundary term. The Grabcut 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 $a > 0$ to have length penalization 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 behaviour in the GF is controlled by the value of α . The FF, and BF grow in concavities, but unless a local optimum is found, it tends 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 Grabcut, 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



Figure 9.8: Foreground (green) and background (blue) seeds are shown in the left and the resulting Grabcut segmentation int the right.

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.

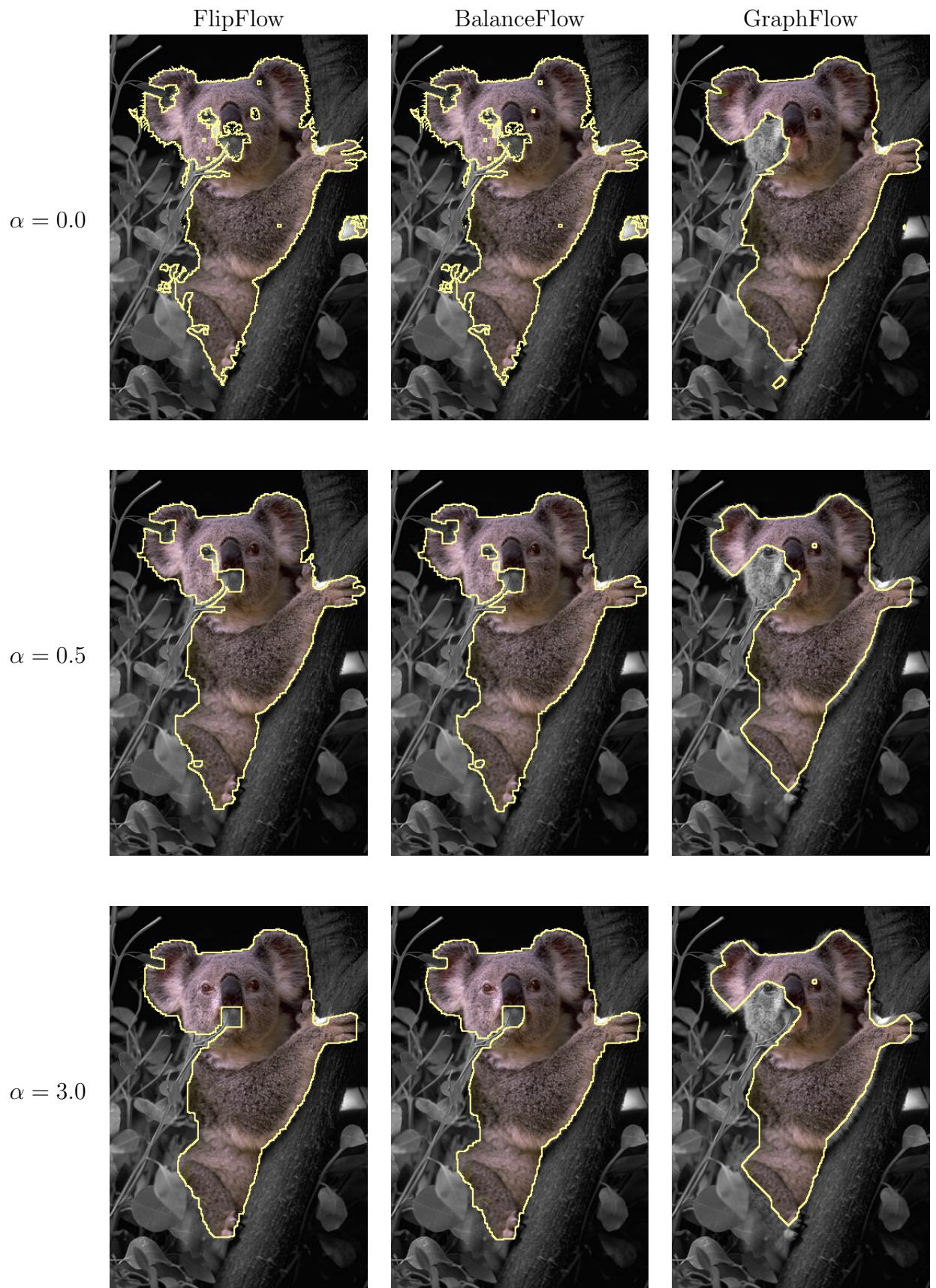
9.3.2 Comparison

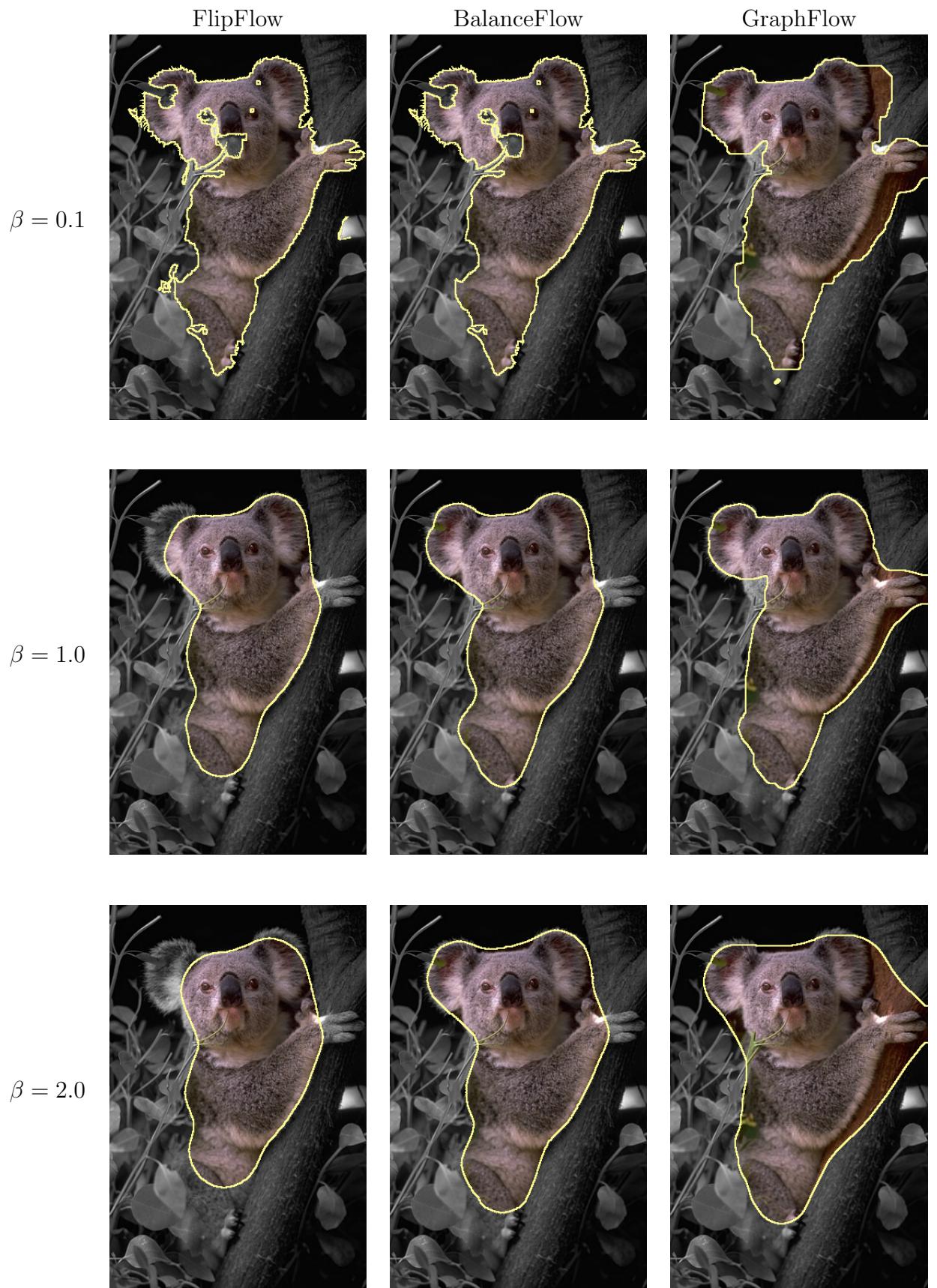
In Figures 9.13 to 9.16 we show the segmentation results of several images for the three models developed in this thesis and the SLCR model. We opt to set the same parameters in all instances, although a better segmentation could be obtained by setting them separately.

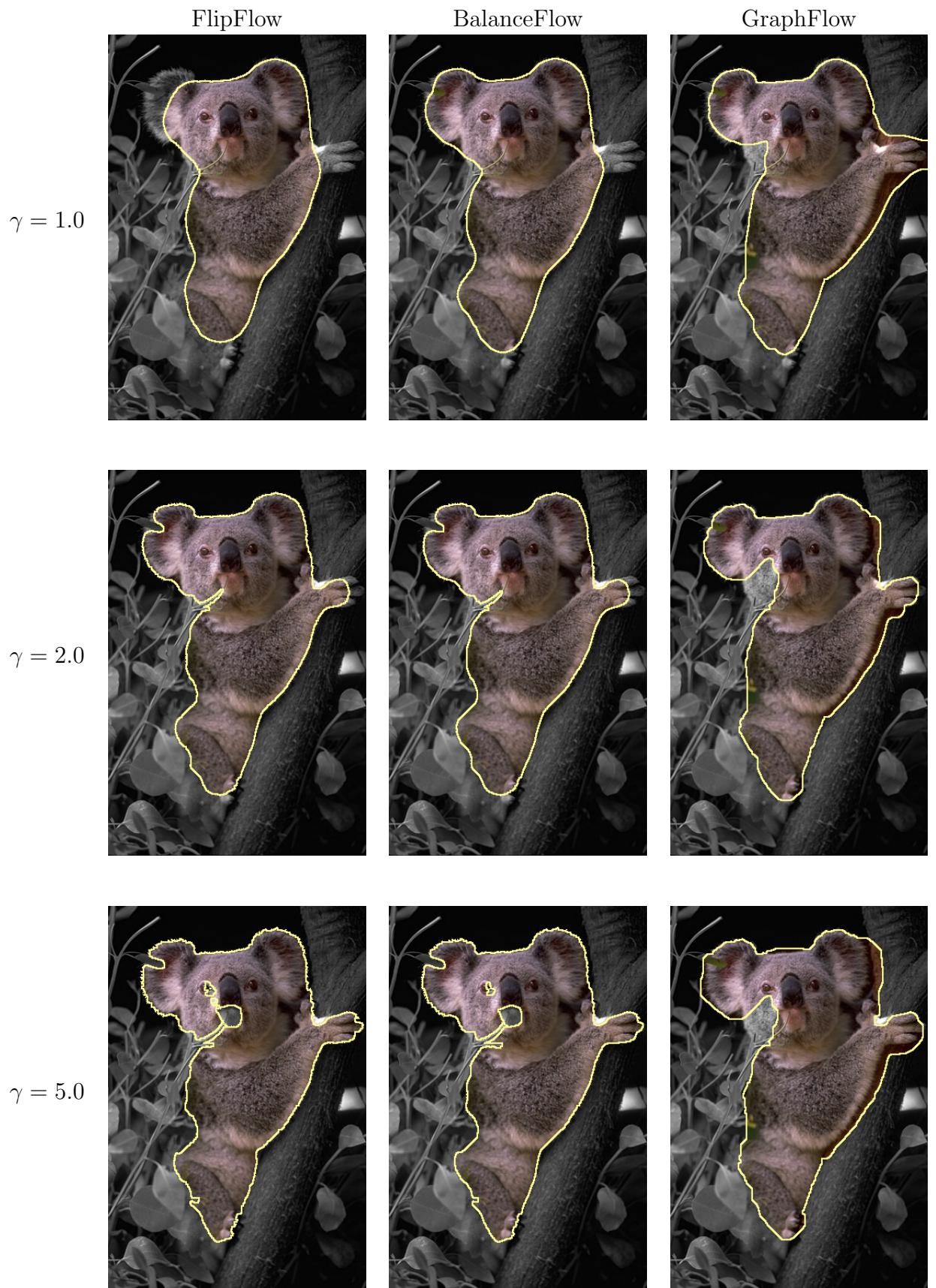
The curvature regularization in the FF,BF models are well perceived in the airplane, camel and man-in-white pictures, but those models are not able to correctly segment the brown-snake in Figure 9.16. They also have a hard time to segment the birds in Figure 9.14 due to the object large curvature range. Nonetheless, they correctly segmented the green-snake in Figure 9.15 while the Grabcut return three disconnected components.

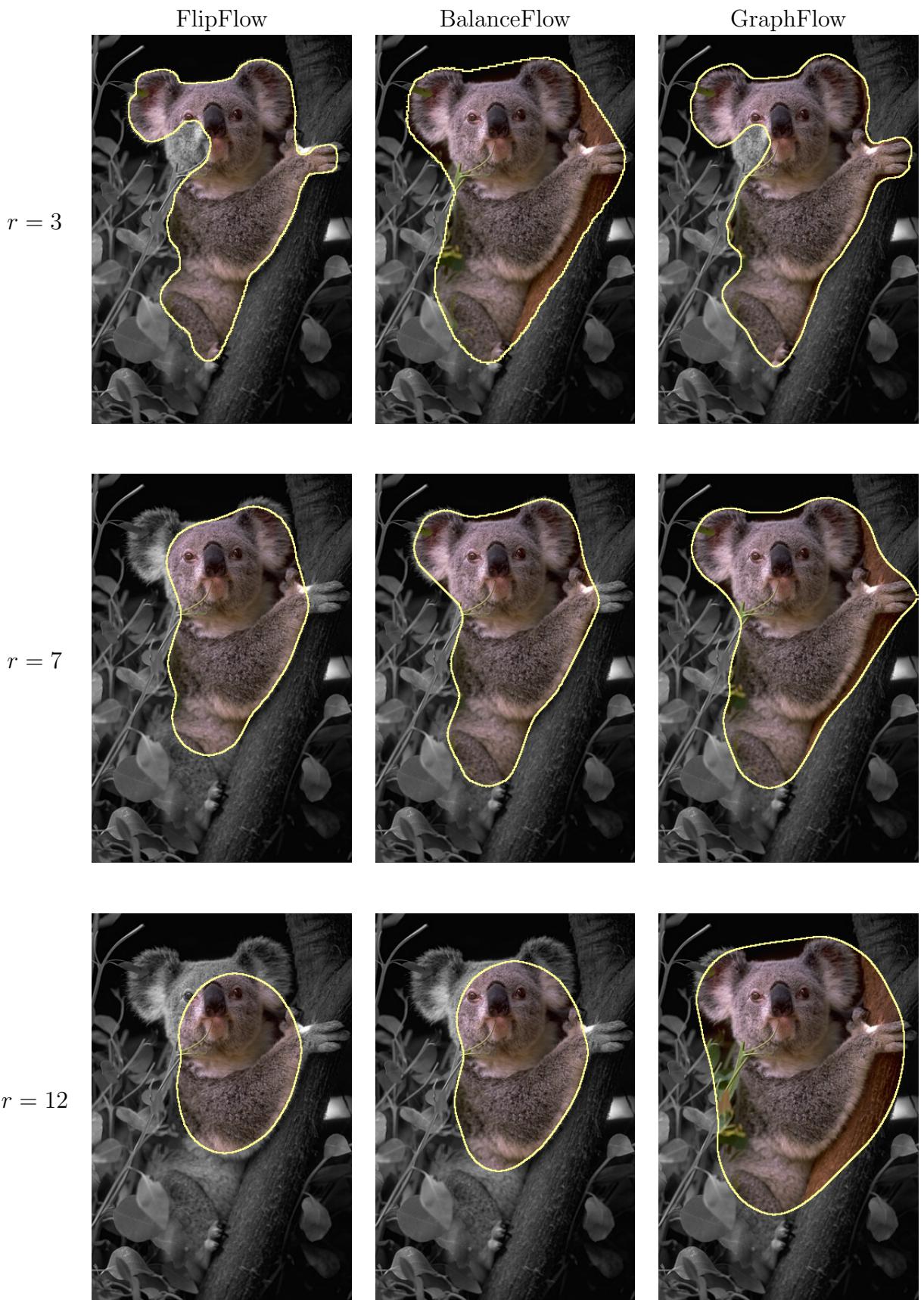
For the chosen parameters, the GraphFlow did not evolve the initial Grabcut segmentation to much, except for the brown-snake in Figure 9.16, for which GF and SLCR presented the best segmentation. However, the GF has the lowest running time among the four models.

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

Figure 9.9: Exp- α

Figure 9.10: Exp- β

Figure 9.11: Exp- γ

Figure 9.12: Exp- $bRadius$

connectivity (set to 8), but is likely to follow an increase in running time, which is already the highest between the models tested.

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 Grabcut model, which originally does not implement geometric regularization, and presents lower running times than FF and BF. Finally, our models have shown to be competitive with the SLCR segmentation algorithm that employs curvature regularization.

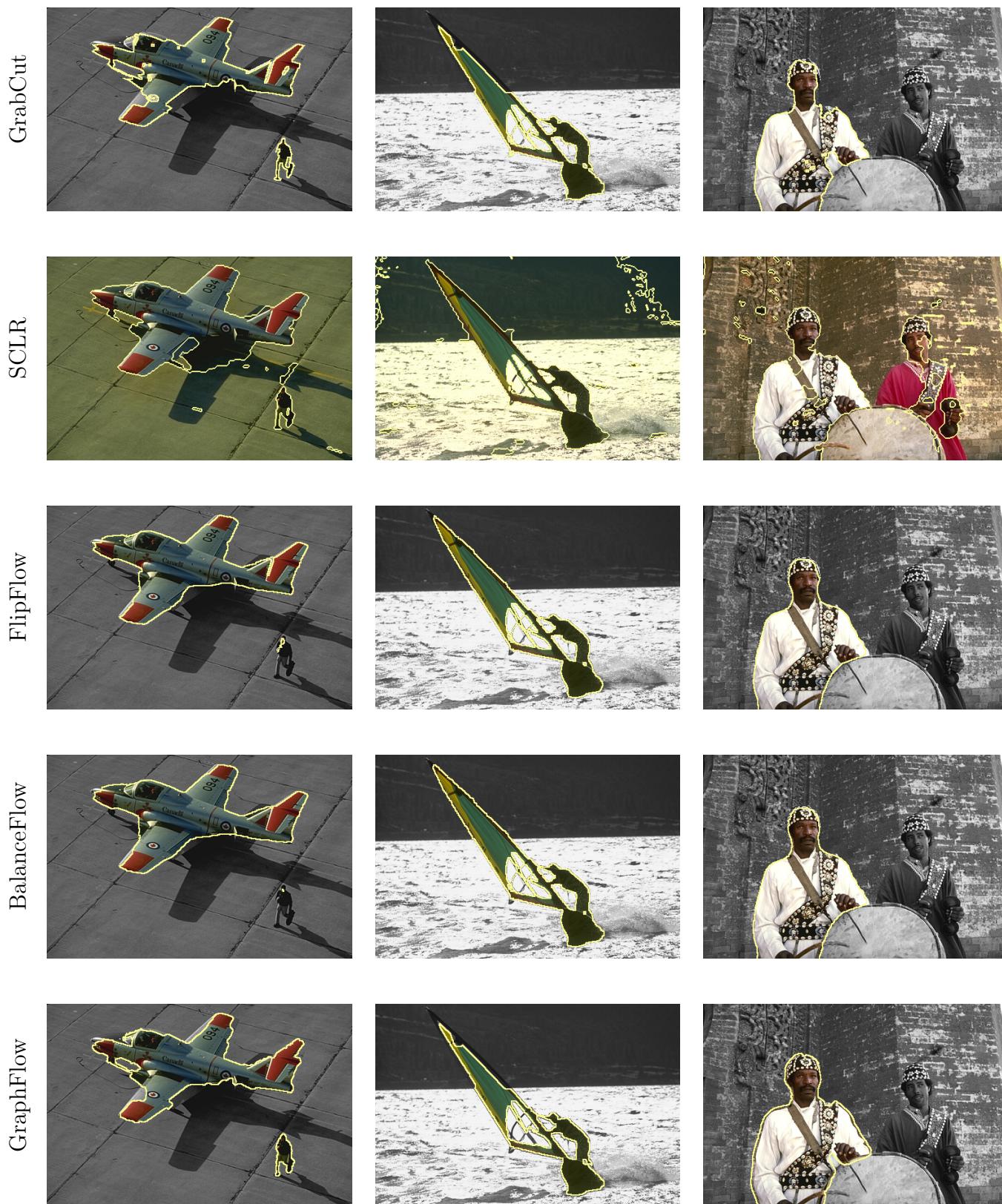


Figure 9.13: Comparison 1

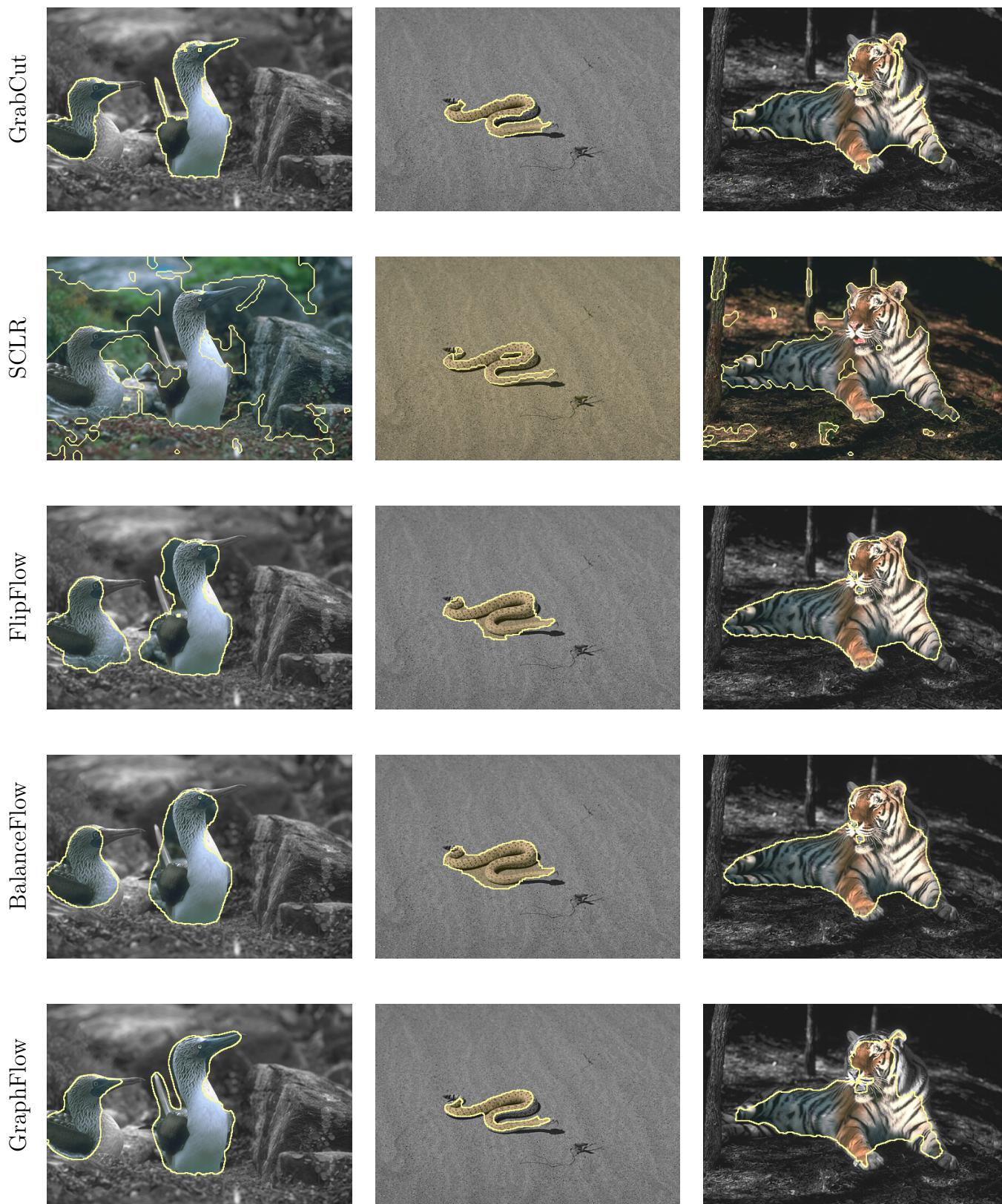


Figure 9.14: Comparison 2



Figure 9.15: Comparison 3

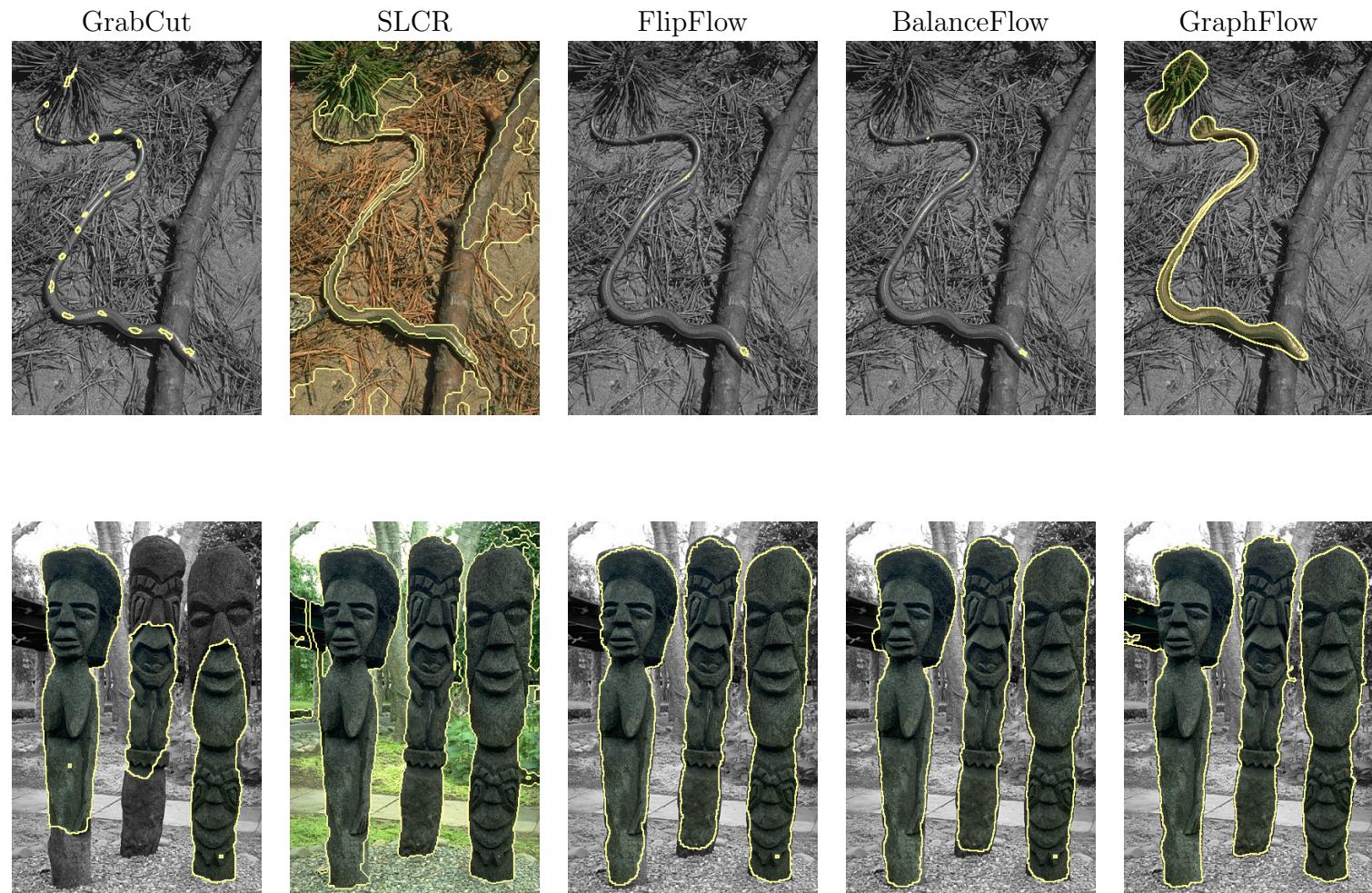


Figure 9.16: Comparison 4

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.

A.1 Estimating curvature with outer disks

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 $g_R : \mathcal{R} \times \mathcal{C} \rightarrow \mathbb{R}$ as

$$g_R(p) = \left(\Theta_i(p) - \Theta_o(p) \right)^2.$$

Claim 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

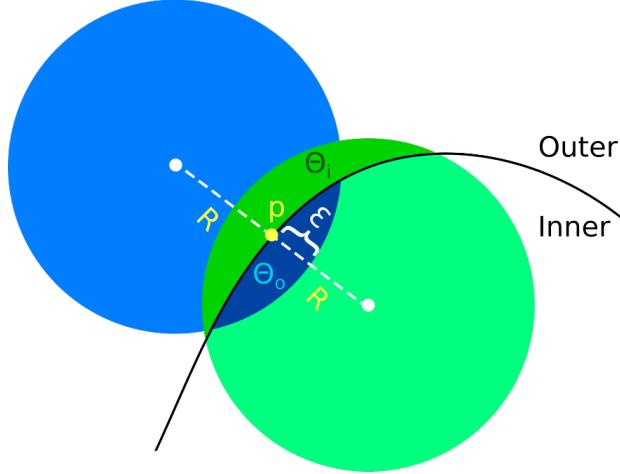


Figure A.1: disks of radius $R + \epsilon$ distant R units from $p \in \mathcal{C}$ in the normal direction.

$$\begin{aligned} C(x) &= C(0) + \frac{dC}{dx}x + \frac{1}{2}\frac{d^2C}{dx^2}x^2 + O(x^3) \\ &= \frac{\kappa}{2}x^2 + O(x^3). \end{aligned}$$

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 .

$$\begin{aligned} h(x) &= R + \epsilon - \sqrt{(R + \epsilon)^2 - x^2} \\ \Theta_o &= 2 \int_0^{x_o} f(x) + \epsilon - h(x) \end{aligned}$$

To compute the intersection point x_o of the parabola with the disk, we use again Pythagoras' theorem.

$$\begin{aligned} (R + \epsilon)^2 &= (R - \frac{\kappa}{2}x_o^2)^2 + x_o^2 \\ 0 &= \frac{\kappa^2}{4}x_o^4 + (1 - R\kappa)x_o^2 + R^2 - (R + \epsilon)^2 \end{aligned}$$

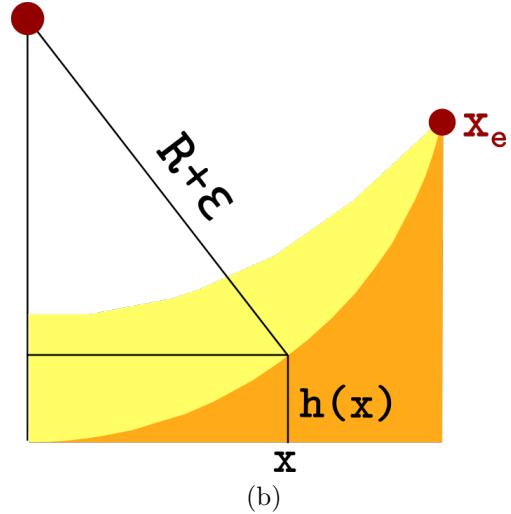
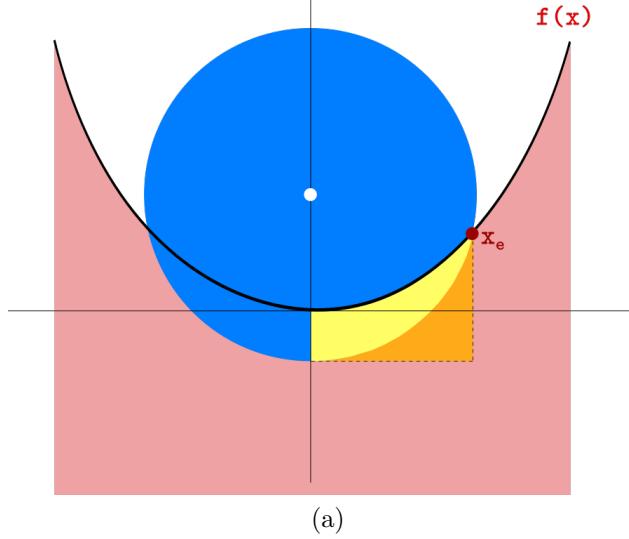


Figure A.2: The yellow area equals $\Theta_o/2$, the same value of the area under the parabola from $x = 0$ until $x = x_o$ minus the orange area $h(x)$.

By setting $z_o = x_o^2$

$$\begin{aligned}\Delta_o &= (1 - R\kappa)^2 + \kappa^2(2R\epsilon + \epsilon^2) \\ z_o &= \frac{2}{\kappa^2}(R\kappa - 1 + \sqrt{\Delta_o}) \\ x_o &= \frac{\sqrt{2}}{\kappa} \sqrt{R\kappa - 1 + \sqrt{\Delta_o}}\end{aligned}$$

We proceed similarly for the inner disk.

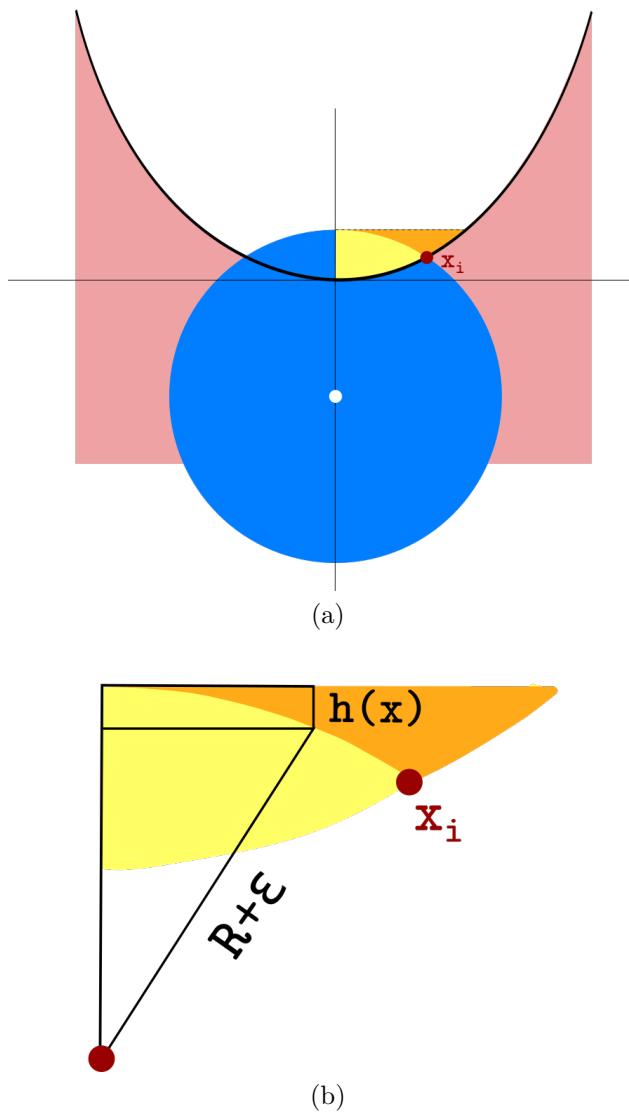


Figure A.3: The yellow area corresponds to Θ_i and it equals the area between the parabola and the disk from $x = 0$ until $x = x_i$.

$$\Theta_i = 2 \int_0^{x_i} \epsilon - f(x) - h(x)$$

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 $\kappa = 0, R = 0$. \blacksquare

Therefore, the squared curvature at point p can be estimated as

$$\hat{\kappa}_{R-sep}^2 = \frac{144}{125R^6}g_R(p).$$

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.

B.1 Cellular grid model

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.

$$\mathbf{P}_{i,j} = \begin{cases} 1, & \text{face } j \text{ is positive incident to edge } i \\ -1, & \text{face } j \text{ is negative incident to edge } i \\ 0, & \text{otherwise} \end{cases}$$

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 $\mathbf{y} \in$

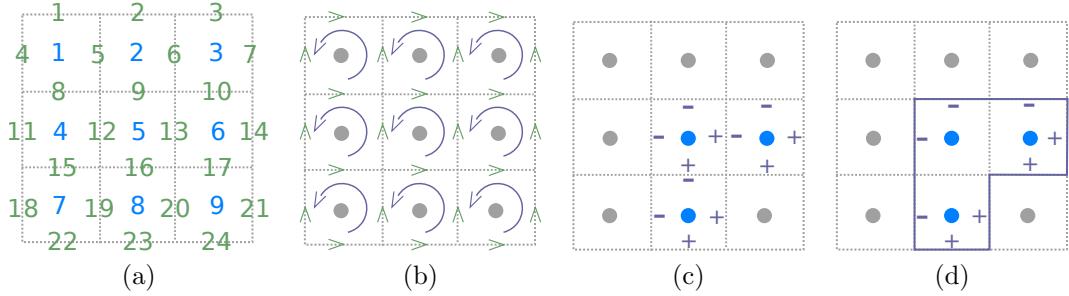


Figure B.1: Figures (a) and (b) illustrates indices assignments and orientation; figure (d) shows the result of applying the pixel incidence matrix for the active pixels in figure (c).

\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.

$$\mathbf{P} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{P} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ 0 \\ -1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Bibliography

- [AM03] Luigi Ambrosio and Simon Masnou. “A direct variational approach to a problem arising in image reconstruction”. In: *Interfaces and Free Boundaries* 5.1 (2003), pp. 63–81 (cit. on p. 73).
- [AT90] Luigi Ambrosio and Vincenzo Maria Tortorelli. “Approximation of functional depending on jumps by elliptic functional via t-convergence”. In: *Communications on Pure and Applied Mathematics* 43.8 (1990), pp. 999–1036 (cit. on p. 25).
- [Bal+01] Coloma Ballester, Marcelo Bertalmio, Vicent Caselles, Guillermo Sapiro, and Joan Verdera. “Filling-in by joint interpolation of vector fields and gray levels”. In: *IEEE transactions on image processing* 10.8 (2001), pp. 1200–1211 (cit. on p. 71).
- [Bar+11] Leah Bar, Tony F. Chan, Ginmo Chung, Miyoun Jung, Nahum Kiryati, Rami Mohieddine, Nir Sochen, and Luminita A. Vese. “Mumford and Shah Model and its Applications to Image Segmentation and Image Restoration”. In: *Handbook of Mathematical Methods in Imaging*. New York, NY: Springer New York, 2011, pp. 1095–1157 (cit. on p. 25).
- [BB98] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. CRC Press, 1998. ISBN: 9781439822067 (cit. on p. 16).
- [BC09] Fethallah Benmansour and Laurent D Cohen. “Fast object segmentation by growing minimal paths from a single point on 2D or 3D images”. In: *Journal of Mathematical Imaging and Vision* 33.2 (2009), pp. 209–221 (cit. on p. 30).
- [BC11] Fethallah Benmansour and Laurent D Cohen. “Tubular structure segmentation based on minimal path method and anisotropic enhancement”. In: *International Journal of Computer Vision* 92.2 (2011), pp. 192–210 (cit. on p. 30).
- [BCN02] Giovanni Bellettini, Vicent Caselles, and Matteo Novaga. “The total variation flow in RN”. In: *Journal of Differential Equations* 184.2 (2002), pp. 475–525 (cit. on p. 65).

- [Ber+00] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. “Image inpainting”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 417–424 (cit. on p. 66).
- [Bes86] Julian Besag. “On the statistical analysis of dirty pictures”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 48.3 (1986), pp. 259–279 (cit. on p. 56).
- [BH02] Endre Boros and Peter L. Hammer. “Pseudo-Boolean optimization”. In: *Discrete Applied Mathematics* 123.1 (2002), pp. 155–225 (cit. on pp. 42, 44, 46, 50).
- [BHN01] Alfred M Bruckstein, Robert J Holt, and Arun N Netravali. “Discrete elastica”. In: *Applicable Analysis* 78.3-4 (2001), pp. 453–485 (cit. on pp. 74, 75).
- [BHS91] Endre Boros, Peter L Hammer, and Xiaorong Sun. *Network flows and minimization of quadratic pseudo-Boolean functions*. Tech. rep. Technical Report RRR 17-1991, RUTCOR, 1991 (cit. on p. 46).
- [BJ01] Y. Y. Boykov and M. P. Jolly. “Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images”. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1. 2001, 105–112 vol.1 (cit. on p. 53).
- [BK03] Y. Boykov and V. Kolmogorov. “Computing geodesics and minimal surfaces via graph cuts”. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. Oct. 2003, 26–33 vol.1 (cit. on pp. 58, 133).
- [BKR11] Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov random fields for vision and image processing*. Mit Press, 2011 (cit. on pp. 41, 58).
- [BM85] Alain Billionnet and Michel Minoux. “Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions”. In: *Discrete Applied Mathematics* 12.1 (1985), pp. 1–11 (cit. on p. 42).
- [BNR01] Alfred M Bruckstein, Arun N Netravali, and Tom J Richardson. “Epi-convergence of discrete elastica”. In: *Applicable Analysis* 79.1-2 (2001), pp. 137–171 (cit. on p. 75).
- [BT09] Amir Beck and Marc Teboulle. “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”. In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202 (cit. on pp. 20, 24, 31, 34).

- [BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. “Fast approximate energy minimization via graph cuts”. In: *IEEE Transactions on pattern analysis and machine intelligence* 23.11 (2001), pp. 1222–1239 (cit. on pp. 41, 58).
- [Cas+93] Vicent Caselles, Francine Catté, Tomeu Coll, and Françoise Dibos. “A geometric model for active contours in image processing”. In: *Numerische mathematik* 66.1 (1993), pp. 1–31 (cit. on pp. 26, 29).
- [CCP08] Antonin Chambolle, Daniel Cremers, and Thomas Pock. *A convex approach for computing minimal partitions*. Tech. rep. 649. CMAP, Ecole Polytechnique, France, 2008 (cit. on p. 20).
- [CEN06] Tony F Chan, Selim Esedoglu, and Mila Nikolova. “Algorithms for finding global minimizers of image segmentation and denoising models”. In: *SIAM journal on applied mathematics* 66.5 (2006), pp. 1632–1648 (cit. on p. 32).
- [Cha+02] Tony F. Chan, Sung Ha Kang, Kang, and Jianhong Shen. “Euler’s Elastica And Curvature Based Inpaintings”. In: *SIAM J. Appl. Math* 63 (2002), pp. 564–592 (cit. on pp. 70, 71).
- [Cha04] Antonin Chambolle. “An Algorithm for Total Variation Minimization and Applications”. In: *Journal of Mathematical Imaging and Vision* 20.1 (Jan. 2004), pp. 89–97. ISSN: 1573-7683 (cit. on pp. 24, 31, 34).
- [Cha99] Antonin Chambolle. “Finite-differences discretizations of the Mumford-Shah functional”. In: *Mathematical Modelling and Numerical Analysis* 33.2 (1999), pp. 261–288 (cit. on p. 25).
- [Che+06] Li Chen, Yue Zhou, Yonggang Wang, and Jie Yang. “GACV: geodesic-aided C–V method”. In: *Pattern Recognition* 39.7 (2006), pp. 1391–1395 (cit. on pp. 27, 30).
- [CK97] Laurent D Cohen and Ron Kimmel. “Global minimum for active contour models: A minimal path approach”. In: *International journal of computer vision* 24.1 (1997), pp. 57–78 (cit. on pp. 18, 30).
- [CKS97] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. “Geodesic Active Contours”. In: *International Journal of Computer Vision* 22.1 (Feb. 1997), pp. 61–79 (cit. on p. 29).
- [CP11] Antonin Chambolle and Thomas Pock. “A first-order primal-dual algorithm for convex problems with applications to imaging”. In: *Journal of mathematical imaging and vision* 40.1 (2011), pp. 120–145 (cit. on p. 34).

- [CS01] Tony F Chan and Jianhong Shen. “Nontexture inpainting by curvature-driven diffusions”. In: *Journal of Visual Communication and Image Representation* 12.4 (2001), pp. 436–449 (cit. on pp. 67, 68).
- [DB08] Andrew Delong and Yuri Boykov. “A scalable graph-cut algorithm for ND grids”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8 (cit. on p. 55).
- [DCL89] Ennio De Giorgi, Michelle Carriero, and Antion Leaci. “Existence theorem for a minimum problem with free discontinuity set”. In: *Archive for Rational Mechanics and Analysis* 108.4 (1989), pp. 195–218 (cit. on p. 25).
- [Eck08] Klaus Ecker. “Heat equations in geometry and topology”. In: *Jahresber. Deutsch. Math.-Verein* 110.3 (2008), pp. 117–141 (cit. on p. 64).
- [EG10] N. Y. El-Zehiry and L. Grady. “Fast global optimization of curvature”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 2010, pp. 3257–3264 (cit. on pp. 78–80, 134).
- [FFA15] Vadim Fedorov, Gabriele Facciolo, and Pablo Arias. “Variational Framework for Non-Local Inpainting”. In: *Image Processing On Line* 5 (2015), pp. 362–386 (cit. on p. 20).
- [FLT17] Marion Foare, Jacques-Olivier Lachaud, and Hugues Talbot. “Numerical Implementation of the Ambrosio-Tortorelli Functional Using Discrete Calculus and Application to Image Restoration and Inpainting”. In: *Reproducible Research in Pattern Recognition*. Ed. by Bertrand Kerautret, Miguel Colom, and Pascal Monasse. Cham: Springer International Publishing, 2017, pp. 91–103. ISBN: 978-3-319-56414-2 (cit. on p. 25).
- [G+86] Michael Gage, Richard S Hamilton, et al. “The heat equation shrinking convex plane curves”. In: *Journal of Differential Geometry* 23.1 (1986), pp. 69–96 (cit. on p. 64).
- [Get12] Pascal Getreuer. “Chan-Vese Segmentation”. In: *Image Processing On Line* 2 (2012), pp. 214–224 (cit. on p. 29).
- [GG84] S. Geman and D. Geman. “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (Nov. 1984), pp. 721–741 (cit. on pp. 19, 56).
- [Gor+14] L Gorelick, O Veksler, Y Boykov, I Ben Ayed, and A Delong. “Local submodular approximations for binary pairwise energies”. In: *Computer Vision and Pattern Recognition*. Vol. 1. 2014, p. 4 (cit. on p. 82).

- [HHS84] Peter L Hammer, Pierre Hansen, and Bruno Simeone. “Roof duality, complementation and persistency in quadratic 0–1 optimization”. In: *Mathematical programming* 28.2 (1984), pp. 121–155 (cit. on pp. 45, 113).
- [Hui+84] Gerhard Huisken et al. “Flow by mean curvature of convex surfaces into spheres”. In: *Journal of Differential Geometry* 20.1 (1984), pp. 237–266 (cit. on p. 64).
- [Ish03] Hiroshi Ishikawa. “Exact optimization for Markov random fields with convex priors”. In: *IEEE transactions on pattern analysis and machine intelligence* 25.10 (2003), pp. 1333–1336 (cit. on p. 40).
- [Ish10] Hiroshi Ishikawa. “Transformation of general binary MRF minimization to the first-order case”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.6 (2010), pp. 1234–1249 (cit. on p. 42).
- [JB06] Olivier Juan and Yuri Boykov. “Active graph cuts”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 1. IEEE. 2006, pp. 1023–1029 (cit. on p. 55).
- [Jba+08] Saâd Jbabdi, Pierre Bellec, Roberto Toro, Jean Daunizeau, Mélanie Péligrini-Issac, and Habib Benali. “Accurate anisotropic fast marching for diffusion-based geodesic tractography”. In: *International journal of biomedical imaging* 2008 (2008) (cit. on p. 30).
- [KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press, 2009. ISBN: 9780262013192 (cit. on pp. 37, 38).
- [Kir96] A. Kirsch. *An Introduction to the Mathematical Theory of Inverse Problems*. Applied Mathematical Sciences. Springer New York, 1996. ISBN: 9780387945309 (cit. on p. 16).
- [KT05] Pushmeet Kohli and Philip HS Torr. “Efficiently solving dynamic markov random fields using graph cuts”. In: *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*. Vol. 2. IEEE. 2005, pp. 922–929 (cit. on p. 55).
- [KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. “Snakes: Active contour models”. In: *International Journal of Computer Vision* 1 (4 1988), pp. 321–331. ISSN: 1573-1405 (cit. on p. 25).
- [KZ04] V. Kolmogorov and R. Zabin. “What energy functions can be minimized via graph cuts?” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (Feb. 2004), pp. 147–159 (cit. on pp. 42, 113).

- [Lev08] Raph Levien. “The elastica: a mathematical history”. In: *Electrical Engineering and Computer Sciences University of California at Berkeley* (2008) (cit. on p. 69).
- [Luc94] L. B. Lucy. “Astronomical inverse problems”. In: *Reviews in Modern Astronomy* 7 (1994), pp. 31–50 (cit. on p. 16).
- [MM98] S. Masnou and J. M. Morel. “Level lines based disocclusion”. In: *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*. Oct. 1998, 259–263 vol.3 (cit. on pp. 18, 72, 73).
- [MS89] David Mumford and Jayant Shah. “Optimal approximation by piecewise smooth functions and associated variational problems”. In: *Communications on pure and applied mathematics* 42.5 (1989), pp. 577–685 (cit. on p. 25).
- [MT95] T. McInerney and D. Terzopoulos. “Topologically adaptable snakes”. In: *Proceedings of IEEE International Conference on Computer Vision*. 1995, pp. 840–845 (cit. on p. 26).
- [MT99] T. McInemey and D. Terzopoulos. “Topology adaptive deformable surfaces for medical image volume segmentation”. In: *IEEE Transactions on Medical Imaging* 18.10 (1999), pp. 840–850 (cit. on p. 26).
- [Mum94] David Mumford. “Elastica and computer vision”. In: *Algebraic geometry and its applications*. Springer, 1994, pp. 491–506 (cit. on p. 67).
- [Nie+14] C. Nieuwenhuis, E. Toeppe, L. Gorelick, O. Veksler, and Y. Boykov. “Efficient Squared Curvature”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 4098–4105 (cit. on pp. 80–83, 134).
- [NW81] George L Nemhauser and Laurence A Wolsey. “Maximizing submodular set functions: formulations and analysis of algorithms”. In: *North-Holland Mathematics Studies*. Vol. 59. Elsevier, 1981, pp. 279–301 (cit. on p. 42).
- [Orl09] James B Orlin. “A faster strongly polynomial time algorithm for submodular function minimization”. In: *Mathematical Programming* 118.2 (2009), pp. 237–251 (cit. on pp. 42, 55).
- [OS88] Stanley Osher and James A Sethian. “Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations”. In: *Journal of computational physics* 79.1 (1988), pp. 12–49 (cit. on p. 64).

- [Pea82] Judea Pearl. “Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach”. In: *Proceedings of the Second AAAI Conference on Artificial Intelligence*. AAAI’82. Pittsburgh, Pennsylvania: AAAI Press, 1982, pp. 133–136 (cit. on p. 42).
- [Pey+10] Gabriel Peyré, Mickael Péchaud, Renaud Keriven, Laurent D Cohen, et al. “Geodesic methods in computer vision and graphics”. In: *Foundations and Trends® in Computer Graphics and Vision* 5.3–4 (2010), pp. 197–397 (cit. on pp. 18, 30).
- [Poc+08] Thomas Pock, Thomas Schoenemann, Gottfried Graber, Horst Bischof, and Daniel Cremers. “A convex formulation of continuous multi-label problems”. In: *European conference on computer vision*. Springer. 2008, pp. 792–805 (cit. on pp. 18, 34).
- [Poc+09] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. “An algorithm for minimizing the Mumford-Shah functional”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 1133–1140 (cit. on pp. 25, 34).
- [Ram+08] Srikumar Ramalingam, Pushmeet Kohli, Kartik Alahari, and Philip HS Torr. “Exact inference in multi-label CRFs with higher order cliques”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8 (cit. on pp. 41, 55).
- [RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. “GrabCut”: Interactive Foreground Extraction Using Iterated Graph Cuts”. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 309–314. ISSN: 0730-0301 (cit. on pp. 116, 131).
- [RL11] Tristan Roussillon and Jacques-Olivier Lachaud. “Accurate Curvature Estimation along Digital Contours with Maximal Digital Circular Arcs”. In: *Combinatorial Image Analysis*. Ed. by Jake K. Aggarwal, Reneta P. Barneva, Valentin E. Brimkov, Kostadin N. Koroutchev, and Elka R. Korutcheva. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 43–55 (cit. on p. 90).
- [ROF92] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear Total Variation Based Noise Removal Algorithms”. In: *Phys. D* 60.1-4 (Nov. 1992), pp. 259–268. ISSN: 0167-2789 (cit. on p. 24).
- [Rot+07] Carsten Rother, Vladimir Kolmogorov, Victor S. Lempitsky, and Martin Szummer. “Optimizing Binary MRFs via Extended Roof Duality”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1–8 (cit. on pp. 52, 109, 113).

- [SC14] Evgeny Strekalovskiy and Daniel Cremers. “Real-Time Minimization of the Piecewise Smooth Mumford-Shah Functional”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer International Publishing, 2014, pp. 127–141. ISBN: 978-3-319-10605-2 (cit. on p. 25).
- [SCC12] Evgeny Strekalovskiy, Antonin Chambolle, and Daniel Cremers. “A convex representation for the vectorial Mumford-Shah functional”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1712–1719 (cit. on p. 34).
- [Sin08] David A Singer. “Lectures on elastic curves and rods”. In: *AIP Conference Proceedings*. Vol. 1002. 1. American Institute of Physics. 2008, pp. 3–32 (cit. on p. 70).
- [SKC09] T. Schoenemann, F. Kahl, and D. Cremers. “Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation”. In: *2009 IEEE 12th International Conference on Computer Vision*. Sept. 2009, pp. 17–23 (cit. on pp. 76, 78, 151).
- [Sou+13] Mohamed Souiai, Evgeny Strekalovskiy, Claudia Nieuwenhuis, and Daniel Cremers. “A co-occurrence prior for continuous multi-label optimization”. In: *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer. 2013, pp. 209–222 (cit. on p. 18).
- [SY05] Nicolas Stroppa and François Yvon. “An Analogical Learner for Morphological Analysis”. In: *Proceedings of the Ninth Conference on Computational Natural Language Learning*. CONLL ’05. Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 120–127 (cit. on p. 16).
- [Sze+06] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. “A comparative study of energy minimization methods for markov random fields”. In: *European conference on computer vision*. Springer. 2006, pp. 16–29 (cit. on p. 55).
- [TD96] McInerney T. and Terzopoulos D. “Deformable Models in Medical Image Analysis: A Survey”. In: *Medical Image Analysis* 1.2 (1996) (cit. on p. 26).
- [Tru60] Clifford Ambrose Truesdell. *The rational mechanics of flexible or elastic bodies*. 1960 (cit. on p. 69).

- [VC02] Luminita A. Vese and Tony F. Chan. “A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model”. In: *International Journal of Computer Vision* 50 (3 Dec. 2002), pp. 271–293. ISSN: 1573-1405 (cit. on pp. 25, 29).
- [Vek07] Olga Veksler. “Graph cut based optimization for MRFs with truncated convex priors”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8 (cit. on pp. 57, 58).
- [Vit67] Andrew Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE transactions on Information Theory* 13.2 (1967), pp. 260–269 (cit. on p. 42).
- [VKR08] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. “Graph cut based image segmentation with connectivity priors”. In: *2008 IEEE conference on computer vision and pattern recognition*. IEEE. 2008, pp. 1–8 (cit. on pp. 59, 60).
- [Zhd15] M.S. Zhdanov. *Inverse Theory and Applications in Geophysics*. Methods in Geochemistry and Geophysics. Elsevier Science, 2015. ISBN: 9780444627124 (cit. on p. 16).