Chapter 1

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 multigrind convergent estimators of length and curvature. We present a combinatorial optimization model capable to evolve a shape to another of lower digital Elastica value. 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.

energy

1.1 Continuous and digital Elastica

To be further developed...

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

alpha >= 0, beta >= 0

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

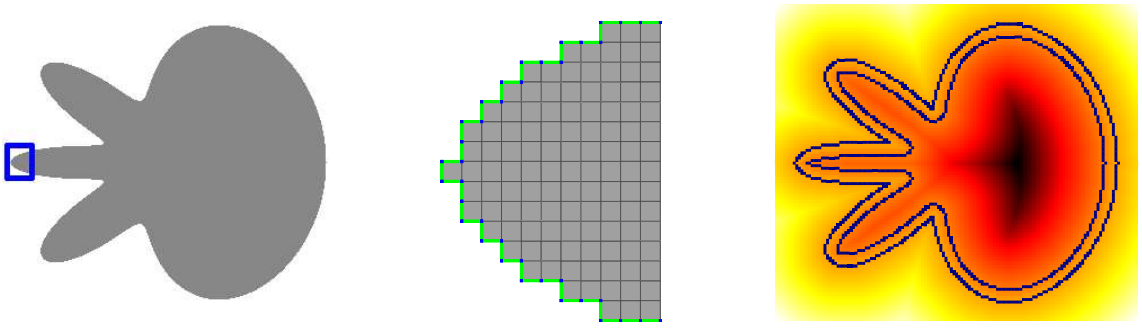
Similarly ? On the other hand, the digital Elastica $\hat{E}_\Theta$ of some digitization $D_h(S)$ of S is defined as

$$\hat{E}_\Theta(D_h(S)) = \sum_{\dot{\boldsymbol{e}} \in \partial D_h(S)} \hat{s}(\dot{\boldsymbol{e}}) \left(\alpha + \beta \hat{\kappa}_r^2(D_h(S), \dot{\boldsymbol{e}}, h) \right), \tag{1.1}$$

linel ? where $\dot{\boldsymbol{e}}$ denotes the center of the edge $\boldsymbol{e}$. 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, 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 that permit us to find the minimum digital shape with respect the digital Elastica energy for some neighborhood of shapes of D.

whose aim is to optimize the digital elastica energy (1.1).



(a) (b) (c)

Figure 1.1: 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.

1.2 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 either estimators induce similar results.

both

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

negative of the distance

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 unecessary and timing 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 multiscaling 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.

time

multiscale

Algorithm 1 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 1 with the MDCA and II-r estimators (r denoting the radius of the estimation ball) to solve the free and constrained Elastica problems.

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

$t \longleftarrow 1$; // level of multiscaling

highest multiscale level ?

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

to highest sale

    $t \longleftarrow 1$;
    $D^{(k)} \longleftarrow Q^\star$;
    k $\longleftarrow$ k $+1$;
  end
end

Algorithm 1: LocalSearch algorithm for Elastica minimization.

emphasize “LocalSearch” somewhere

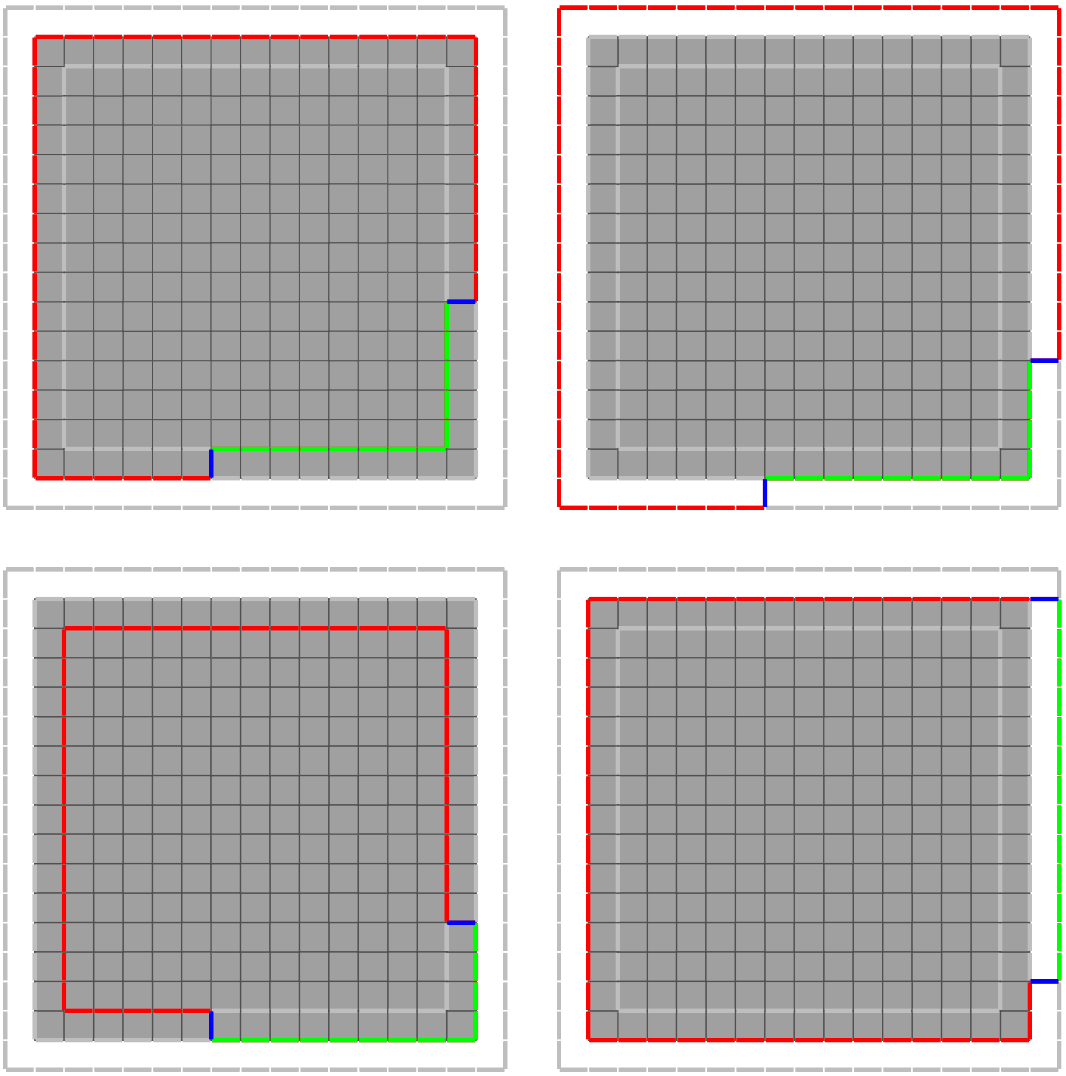


Figure 1.2: Members of $\mathcal{N}_{11}$ for the square shape.

1.2.1 Free digital Elastica

In the free digital Elastica energy we optimize Equation (1.1) 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.

$$\frac{d}{dr}\Big(\int_{\partial B(r)} (\alpha + \beta\kappa^2)ds\Big) = 0$$
$$\frac{d}{dr}\big(\alpha 2\pi r + \beta\pi/r\big) = 0$$
$$r = \alpha^{-1/2}.$$

where is beta ?

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

In Figure 1.3 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 1.5. 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 difficults to evolve in a high resolution setting and it also stops at some local optimum. In fact, the MDCA estimator, although with higher convergence speed, is more sensitive to noise than II, as illustrated in Figure 1.7. Nonetheless, the results can be improved by using a larger neighborhood, as illustrates Figure 1.6.

difficulties

minimum

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 1.4 follows a pattern similar to those in Figure 1.3 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 optimum. We remark that the adoption of an automatic selection of the estimation ball radius may attenuate this problem.

minimum

1.2.2 Constrained digital Elastica

An important advantage of Algorithm 1 is that constraints can be imposed with minimum effort. We present results for two types of constraints. In the first type, we force some pixels to be part of the final solution and in the second we impose orientations at the endpoints of a curve. In Figure 1.8 we compare the flows for different values of α.

We remark that Algorithm 1 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.

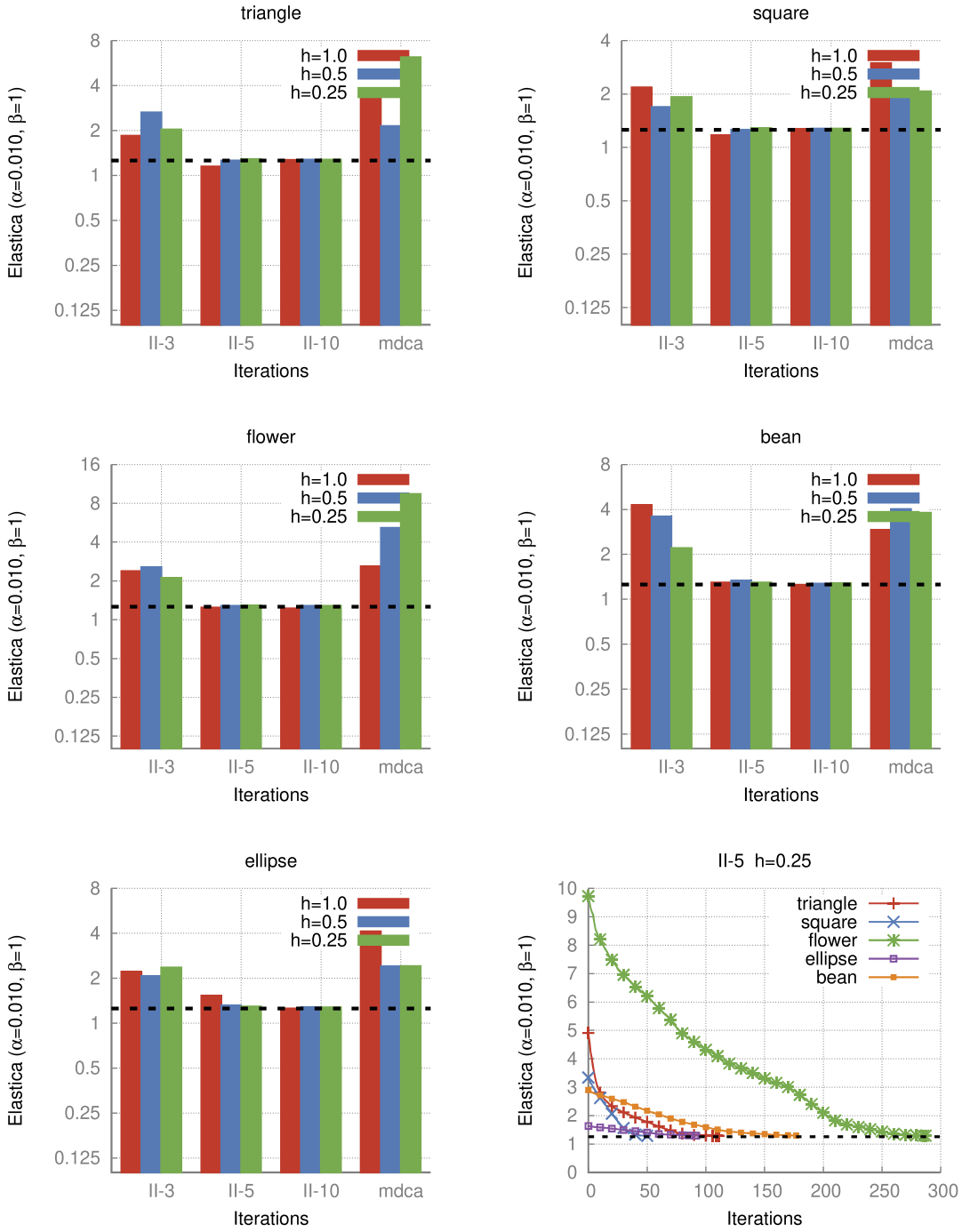


Figure 1.3: Minimum value attained for the digial Elastica ($\alpha = 0.01, \beta = 1$) in comparisson comparison with the global optimium (dashed line) for different curvature estimators and in different scales. The last figure summarizes the digital Elastica evolution value for all shapes using grid step $h = 0.25$.

Very clear and convincing

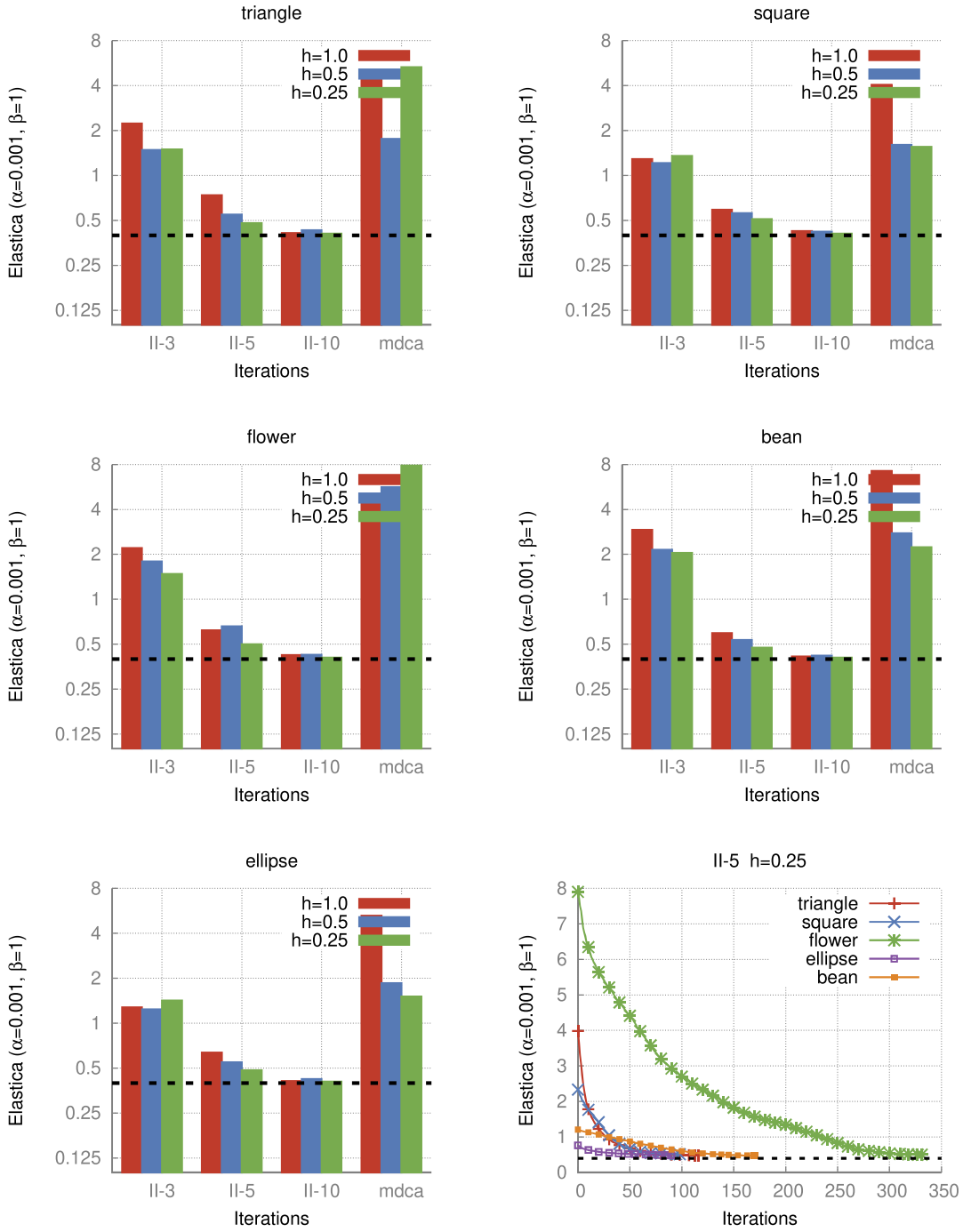


Figure 1.4: Minimum value attained for the digial Elastica ($\alpha = 0.001, \beta = 1$) in comparisonparisson with the global optimium (dashed line) for different curvature estimators and in different scales. The last figure summarizes the digital Elastica evolution value for all shapes using grid step $h = 0.25$.

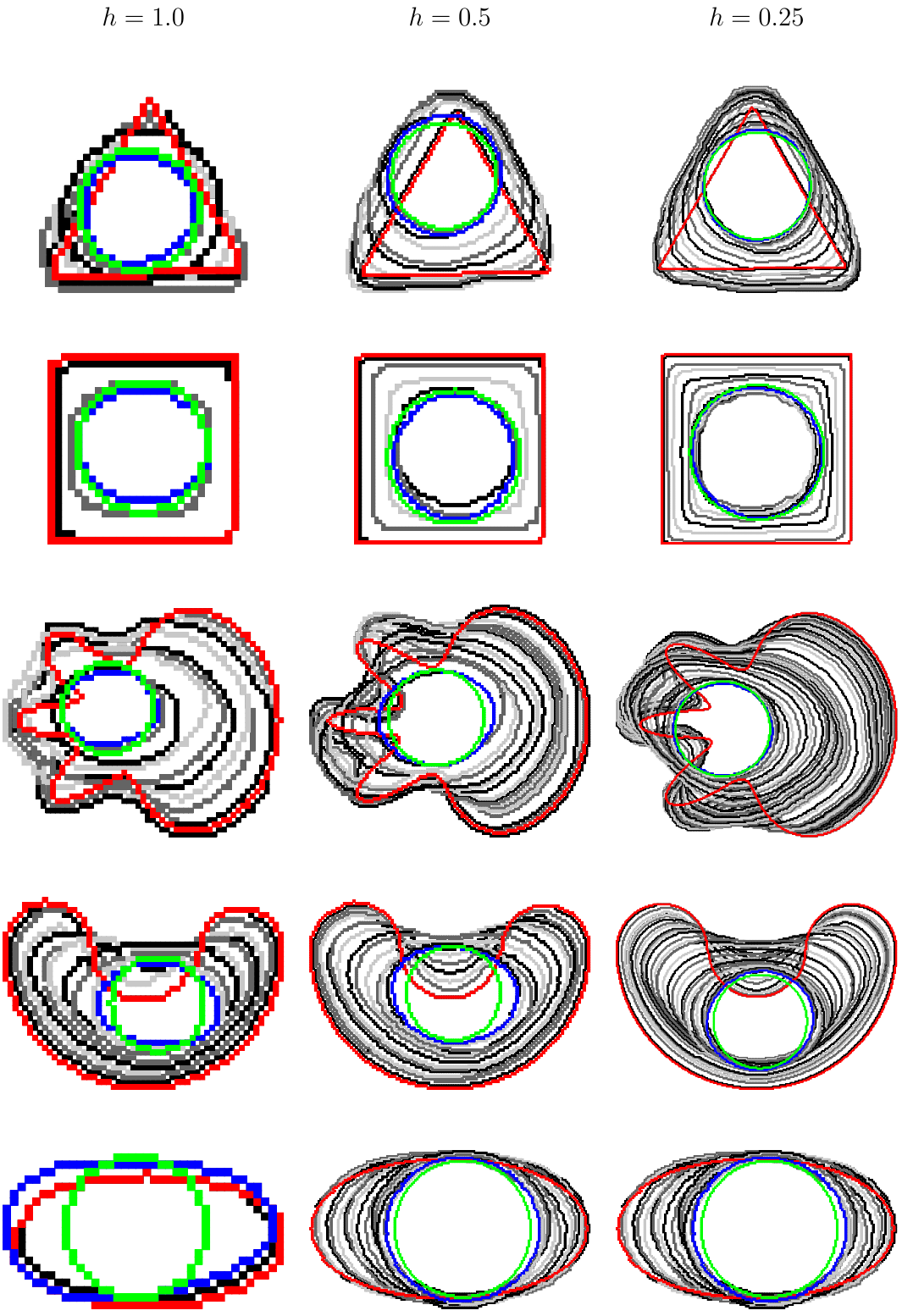


Figure 1.5: LocalSearch algorithm evolutions for several shapes with $\alpha = 0.01, \beta = 1$. The initial contour is colored in red; the final contour is colored in blue; and the optimal contour is colored in green.

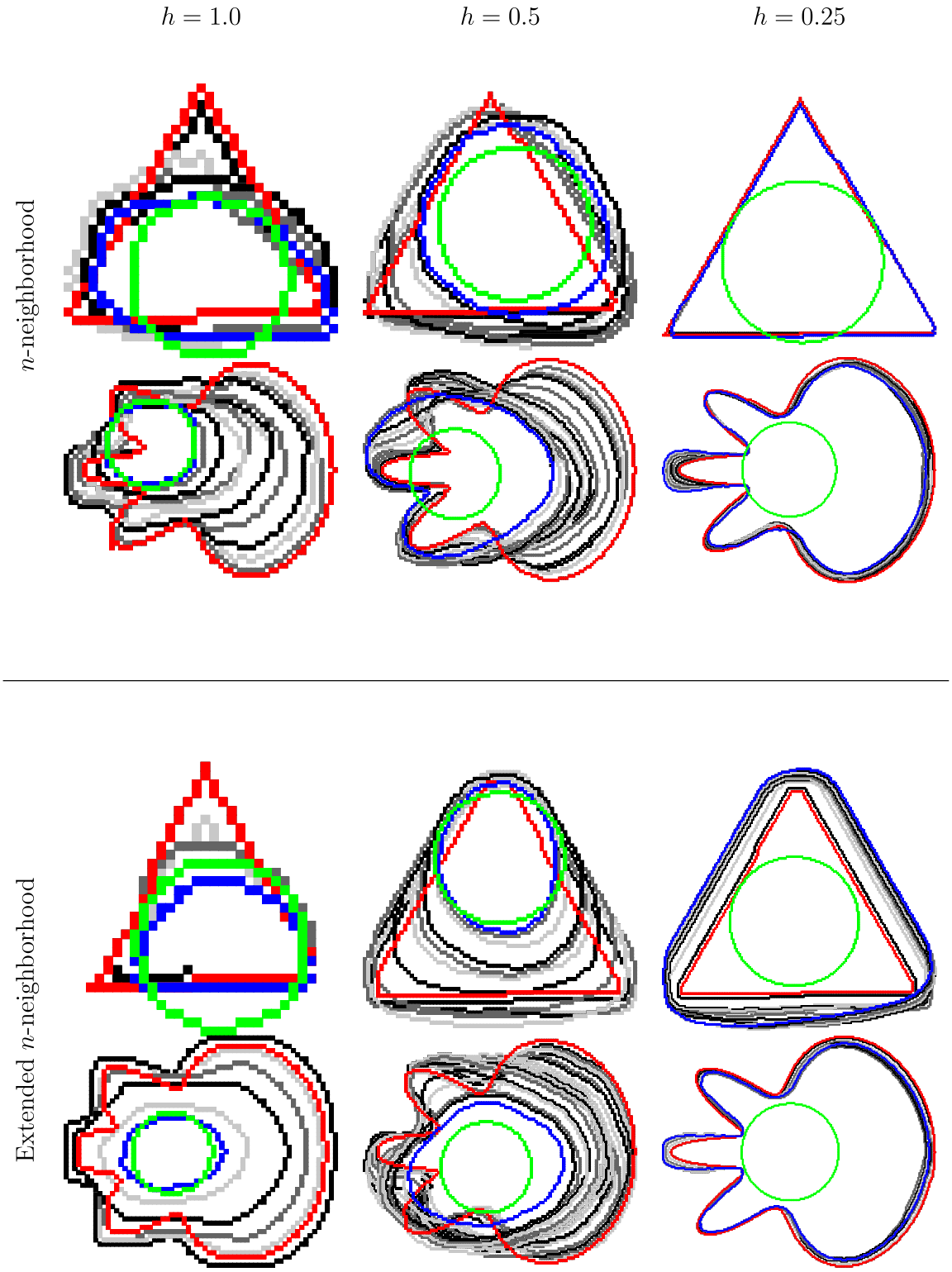
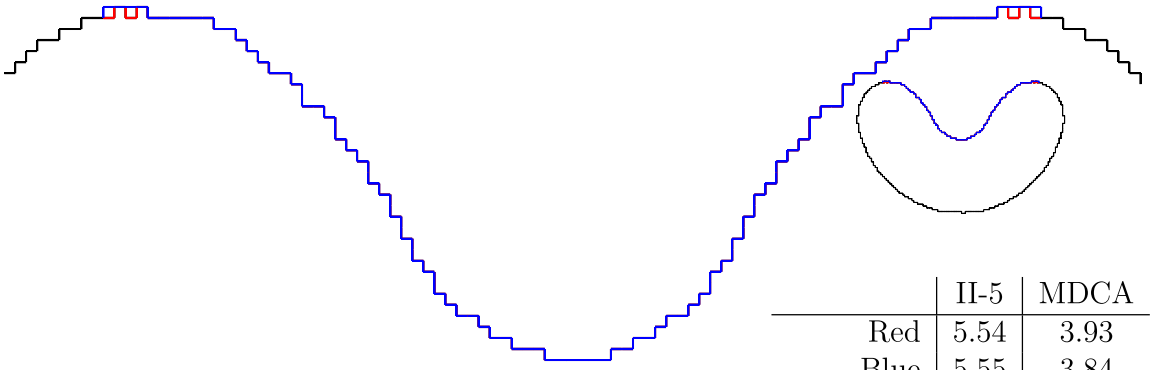


Figure 1.6: In the top row, the MDCA evolution for the neighborhood as presented in Algorithm 1. In the bottom row, the flow using the extended neighborhood. The extended neighborhood additionaly includes the n-neighborhood of the dilation and the erosion of the initial shape by a square of side 1.



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

Figure 1.7: 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.

1.2.3 Running time

The running time of Algorithm 1 is summarized in table Table 1.1. 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.

1.3 Global optimization

In this section we turn to a global optimization approach. However, instead of minimizing Equation (1.1) we are going to optimize a simplified version of it in which we don't need to compute the local length estimator. do not This simplification is necessary to get optimizable models ?

1.3.1 Simplified digital Elastica

The simplified digital Elastica is defined as

$$\hat{E}^{simp}_{\Theta}(D_h(S)) = \sum_{\dot{\boldsymbol{e}}\in\partial D_h(S)} \alpha + \beta\hat{\kappa}^2_r(D_h(S), \dot{\boldsymbol{e}}, h). \tag{1.2}$$

We argue that Equation (1.2) is a reasonable approximation of Equation (1.1). Indeed, executing Algorithm 1 for minimize the simplified digital Elastica obtains very similar results to those for the digital Elastica (see Figure 1.9).

to minimize this

induces

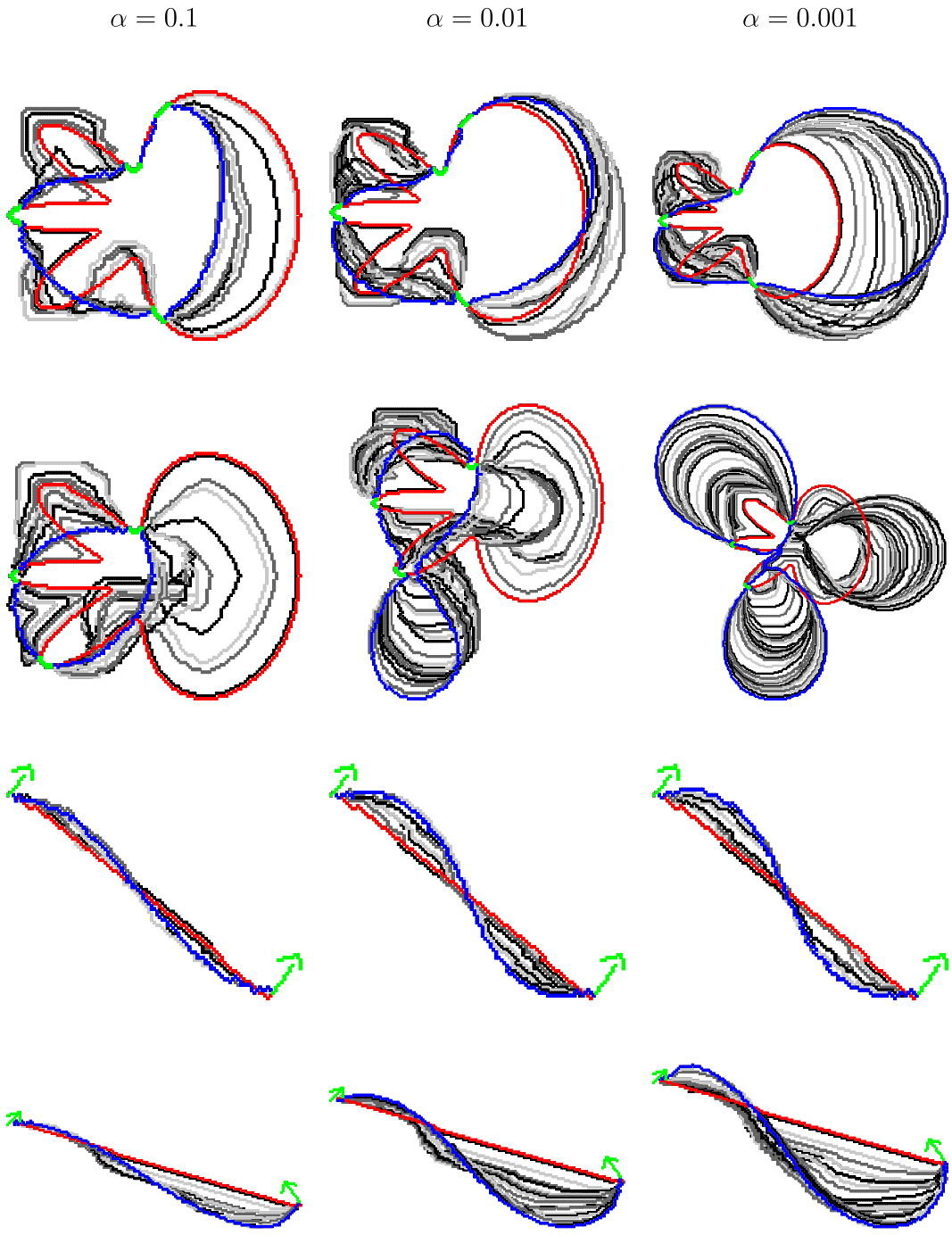


Figure 1.8: In the first and second rows, the flow obtained by forcing the green pixels to be part of the final solution; In the last two rows, the flow obtained by forcing the orientation at the endpoints of the curves.

	$h = 1.0$		$h = 0.5$		$h = 0.25$	
	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 1.1: Running time of LocalSearch for the free Elastica problem.

1.3.2 Optimization model for simplified digital Elastica

in contrast with

Differently from 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 $\boldsymbol{x} \in \mathbb{B}^m$, $\boldsymbol{y} \in \mathbb{B}^n$ and its i-th coefficients are denoted $\boldsymbol{x}_i, \boldsymbol{y}_i$. Further, let $\boldsymbol{A} \in \mathbb{B}^{m\times n}$ the matrix defined as

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

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

$$\begin{aligned} E^{simp}_{\Theta}(\boldsymbol{x},\boldsymbol{y}) &= \sum_{\boldsymbol{y}_i\in\boldsymbol{y}} \boldsymbol{y}_i \left(\alpha + \beta\hat{\kappa}^2_r(D,\boldsymbol{y}_i) \right) \\ &= \sum_{\boldsymbol{y}_i\in\boldsymbol{y}} \boldsymbol{y}_i \left(\alpha + \beta\big(\frac{3}{r^3}(\frac{\pi}{r^2} - |B_r(\boldsymbol{y}_i)|)\big)^2 \right) \\ &= \sum_{\boldsymbol{y}_i\in\boldsymbol{y}} \boldsymbol{y}_i \left(\alpha + \frac{9}{r^6}\beta\big(c^2 - 2c\boldsymbol{A}_i^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{A}_i\boldsymbol{A}_i^T\boldsymbol{x}\big) \right), \end{aligned} \tag{1.3}$$

topologically

where $c = \pi r^2/2$. We remark that linels and pixels in the solution must be topologicaly 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 $T(x, y)$ detailed later. So far we have

(in bold)

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

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

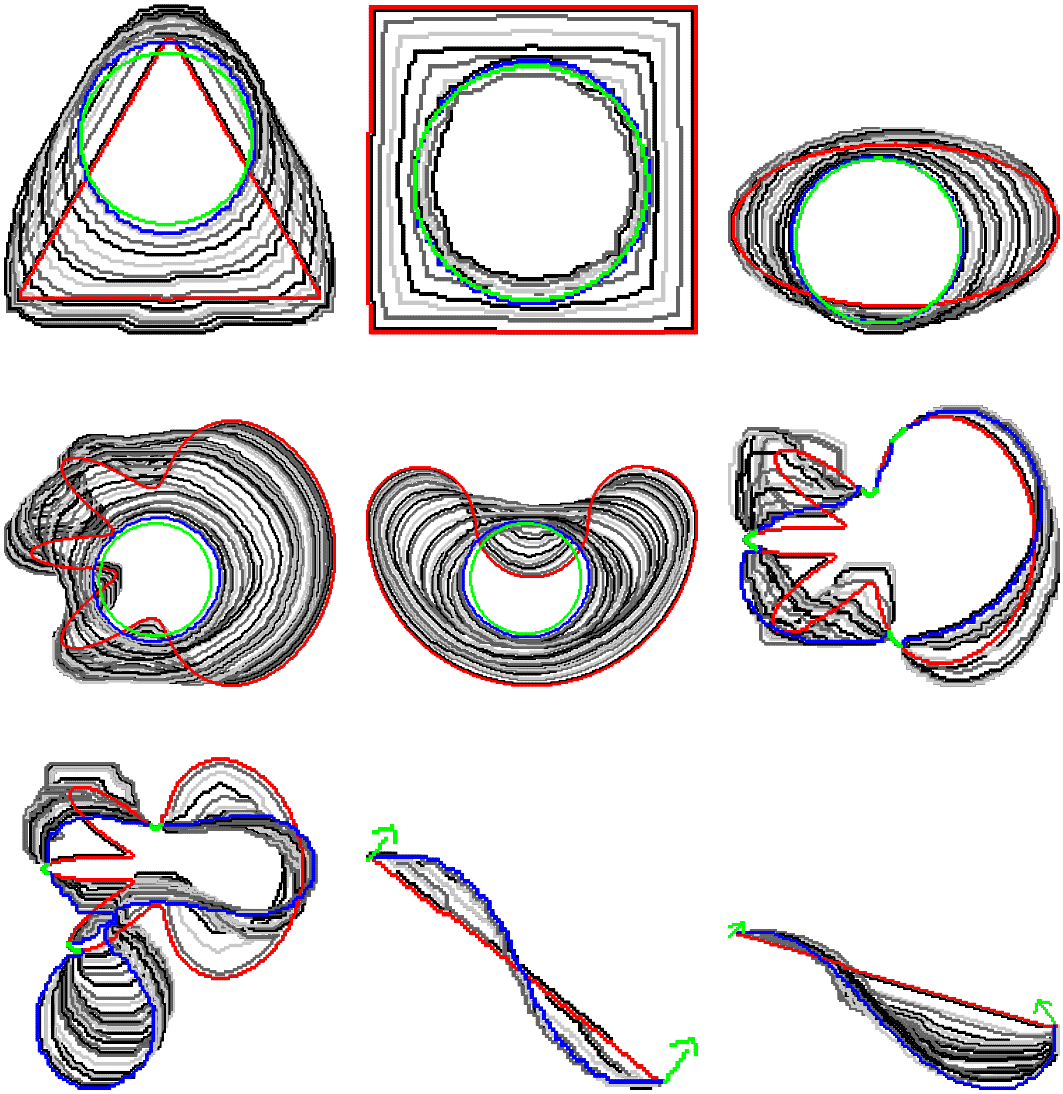


Figure 1.9: Experiments of Section 1.2 for the simplified digital Elastica.

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

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.

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

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

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

Rewriting formulation (P1)

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

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

1.3.4 Linear relaxation of $P1$

the relaxation of

The simplest model we can derive from (P1) consists in to relax the optimization variables, i.e., we impose $\boldsymbol{x} \in \mathbb{U}^m$ and $\boldsymbol{z} \in \mathbb{U}^{2n}$, 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 $\boldsymbol{x}_2^2\boldsymbol{x}_4$ is encoded as the sequence $(\boldsymbol{x}_2, \boldsymbol{x}_2, \boldsymbol{x}_4)$. 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 $\boldsymbol{u_i}$ for each term of $\mathcal{T}$, i.e., $\boldsymbol{u} \in \mathbb{U}^{|\mathcal{T}|}$ and we enforce $|\mathcal{T}_i|+1$ new constraints. In other words, we add the following set of linearization constraints.

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

The linearization of (P1) is written as

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

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

1.3.5 Unconstrained version of P1

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

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

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

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

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

$$\min_{\boldsymbol{x}} \frac{9}{R^6} \sum_{j}^{m} \left(\frac{\pi R^2}{2} - \frac{1}{2} \mathbf{1}^T \boldsymbol{Q} \boldsymbol{B}_j \right)^2, \tag{1.4}$$

where $\mathbf{1} = (1, 1, \cdots, 1)^T \in \mathbb{R}^m$. Equation (1.4) involves the minimization of a fourth order equation and therefore hard to be optimized.

1.4 Conclusion

We gave a historical review of the Elastica and we defined the digital Elastica energy in Section 1.1. The local combinatorial scheme defined in Section 1.2 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 optimum. Finnaly, we sketch some global optimization models in Section 1.3 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.

minimum

Chapter 2

A 2-step evolution model driven by digital Elastica minimization

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

2.1 FlipFlow model

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

deciding

2.1.1 Definitions

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

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

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

We will

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

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

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

Faire un titre court aussi

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 1.3.3, 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 $\overline{X}^{(k)}$, i.e., $\overline{X}^{(k)} = X(I(\overline{D}^{(k)}))$.
An assignment of $X^{(k)}$ is simply denoted $x^{(k)}$.

2.1.2 Algorithm

We recall the definition of the II digital curvature estimator:

$$\hat{\kappa}^2(p) = c_1\Big(c_2 - |B_r(p) \cap D^{(k)}|\Big)^2, \tag{2.1}$$

where $c_1 = 3/r^6$ and $c_2 = \pi r^2/2$.
The following sets are important in the expansion of *Equation* (2.1).

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

Expanding Equation (2.1), we get

$$\begin{aligned} \hat{\kappa}^2(p) &= c_1\Big(c_2 - |F_r^{(k)}(p)| - \sum_{x_j \in X_r^{(k)}(p)} x_j\Big)^2 \\ &= c_1\Big(C + 2\left(|F_r^{(k)}(p)| - c_2\right) \sum_{x_j \in X_r^{(k)}(p)} x_j + \sum_{x_j \in X_r^{(k)}(p)} x_j^2 + \sum_{\substack{x_j, x_l \in X_r^{(k)}(p) \\ j<l}} 2x_j x_l\Big), \end{aligned} \tag{2.2}$$

where $C = c_2^2 - 2c_2 \cdot |F_r^{(k)}(p)| + |F_r^{(k)}(p)|^2$ is a constant. As Equation (2.2) 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 (2.2) by exploiting the binary character of variables and eliminating monomials of second order. We define the following family of energies for given parameters $\boldsymbol{\theta} = (\alpha, \beta) \geq 0$

$$E^{flip}_{(\boldsymbol{\theta},m)}(D^{(k)}, X^{(k)}) = \sum_{x_j \in X^{(k)}} \alpha s(x_j) + \sum_{\substack{p \in \\ R_m(D^{(k)})}} 2c_1\beta\Big((1/2 + |F_r^{(k)}(p)| - c_2) \cdot \sum_{\substack{x_j \in \\ X_r^{(k)}(p)}} x_j + \sum_{\substack{j<l, \\ x_j, x_l \in \\ X_r^{(k)}(p)}} x_j x_l\Big), \quad (2.3)$$

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

$$s(x_{w(p)}) = \sum_{q \in \mathcal{N}_4(p)} t(q), \quad \text{where } t(q) = \begin{cases} (x_{w(p)} - x_{w(q)})^2, & \text{if } q \in I^{(k)} \\ (x_{w(p)} - 1)^2, & \text{if } q \in F^{(k)} \\ (x_{w(p)} - 0)^2, & \text{otherwise} \end{cases} \quad (2.4)$$

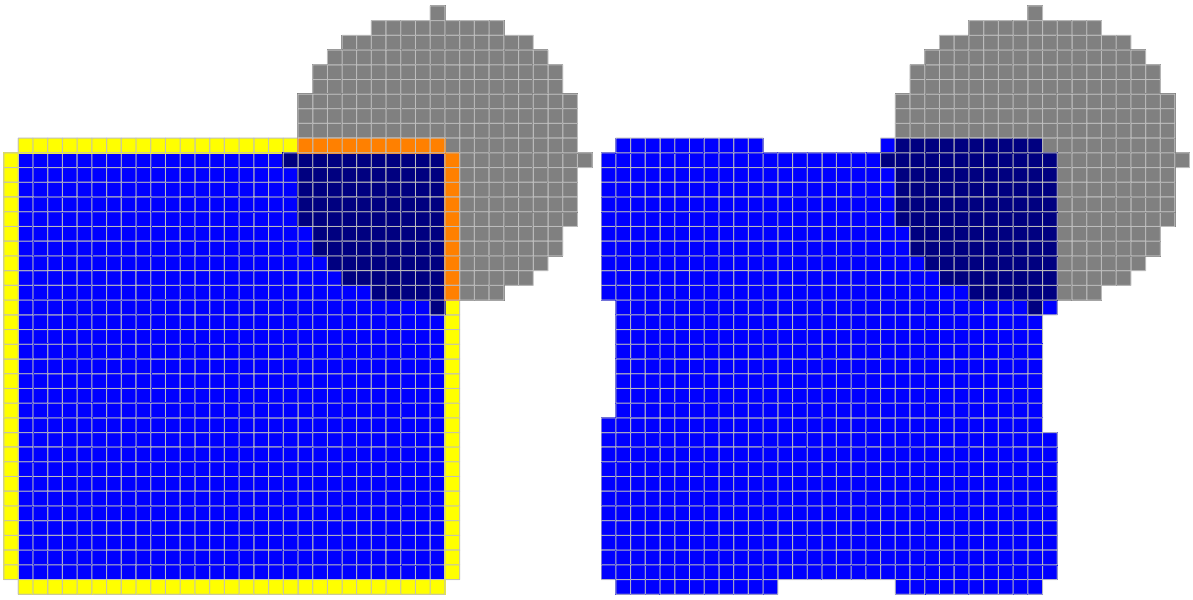
We recall that R_m refers to the $m - ring$ defined in Section 1.2. Each choice of m generates a different flow, which is generally described in the FlipFlow Algorithm 2. To optimize Equation (2.3) we use the QPBOI algorithm [RKLS07].

emphasize

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

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

Algorithm 2: FlipFlow algorithm.



does not

Diret use of Figure 2.1: Directly using the optimization result of Equation (2.3) doesn't decrease squared curvature because contour information is not present in the energy.

2.1.3 Algorithm discussion

As discussed in Section 1.3, 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 (2.3). However, due the lack of contour information, the minimization of Equation (2.3) for $D^{(k)}$ results in undesirable shapes of even higher digital Elastica energy values (see Figure 2.1). 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

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

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

The explanation above covers the treatment of convex parts, but the way to treat is concavities it's not much different. Indeed, concave regions are convex in the shape complement. The FlipFlow Algorithm 2 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,

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

Table 2.1: Since the curvature is negated when reversing the curve (i.e. $\overline{\kappa} = -\kappa$), this process can only shrink convex parts in shrink mode and expand concave parts in expansion mode.

and by doing this we are able to handle concavities. It is called expansion mode because the optimization region, in this case, is the outer pixel boundary of the original shape. Table Table 2.1 sums up these arguments.

In Figure 2.2 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 doesn't 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.

2.2 Optimization method

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

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

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 $E^{flip}_{(\boldsymbol{\theta},m)}$ is non-submodular and optimizing it is a difficult problem, which constrains us to use heuristics and approximation algorithms. The QPBO method [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: QPBOI (improve) and QPBOP (probe) [RKLS07].

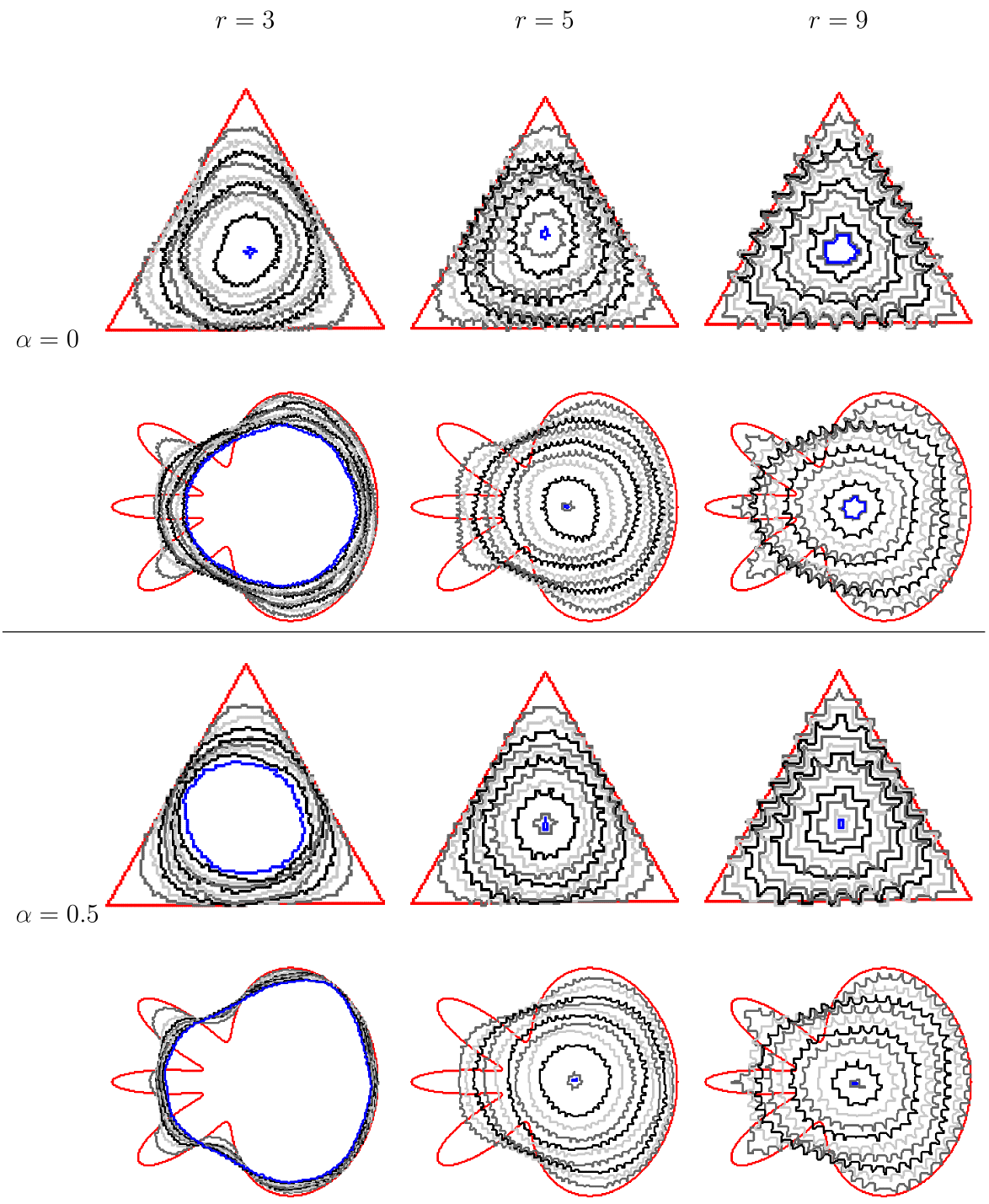


Figure 2.2: The algorithm is very sensitive to the little variations of the estimator, which are particularly important in regions of low squared curvature. Artifacts are somewhat reduced with a length penalization but increases if we use a higher ball radius. For better visualization, curves are displayed every 1/10 of the number of iterations.

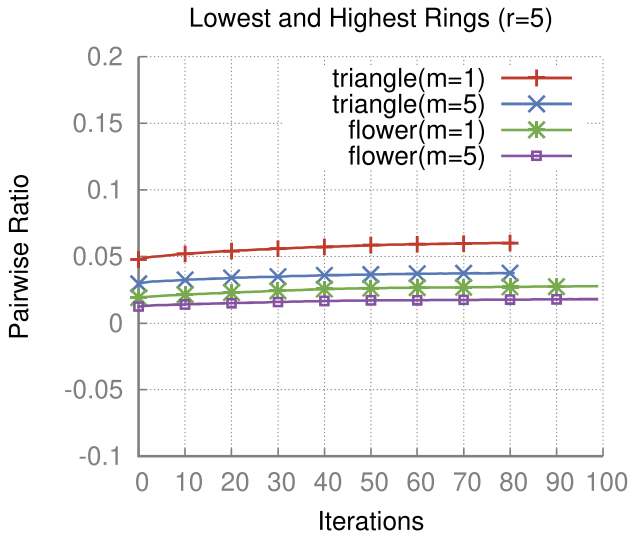


Figure 2.3: We plot the ratio of pairwise terms among all $\binom{|X^{(k)}|}{2}$ combinations. The highest ring has roughly half the number of pairwise terms as the lowest ring.

The first is an approximation method that is guaranteed to not increase the energy, but loses the property of partial optimality. The second is an exact method which is reported to label more variables than QPBO.

The percentage of unlabeled pixels by QPBOP for $E^{flip}_{(\boldsymbol{\theta},m)}$ is quite high, but the percentage decreases to zero as we set m to a value closer to r, the estimation ball radius. Therefore, we are more confident in taking the solution for values of m close to r. However, the way it varies across values of m differs from shape to shape, as is illustrated in Figure 2.4. We also noticed that, for $m = r$, all the pixels were labeled, which indicates that $E_{(\boldsymbol{\theta},r)-flip}$ is an easy instance of the general non-submodular energy $E^{flip}_{(\boldsymbol{\theta},m)}$, but this remains to be proved. The number of pairwise terms in $E_{(\boldsymbol{\theta},r)-flip}$ is roughly half of those in $E_{(\boldsymbol{\theta},1)-flip}$ (see Figure 2.3), which also explains the higher number of labeled variables.

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

2.3 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.

In Figures 2.5 and 2.6 we evaluate several flows for different energies $E^{flip}_{(\boldsymbol{\theta},m)}$. As expected, the number of artifacts decrease as the value of m increases, but the process still tends to shrink the shape to a single point, resembling the curvature flow. while ?

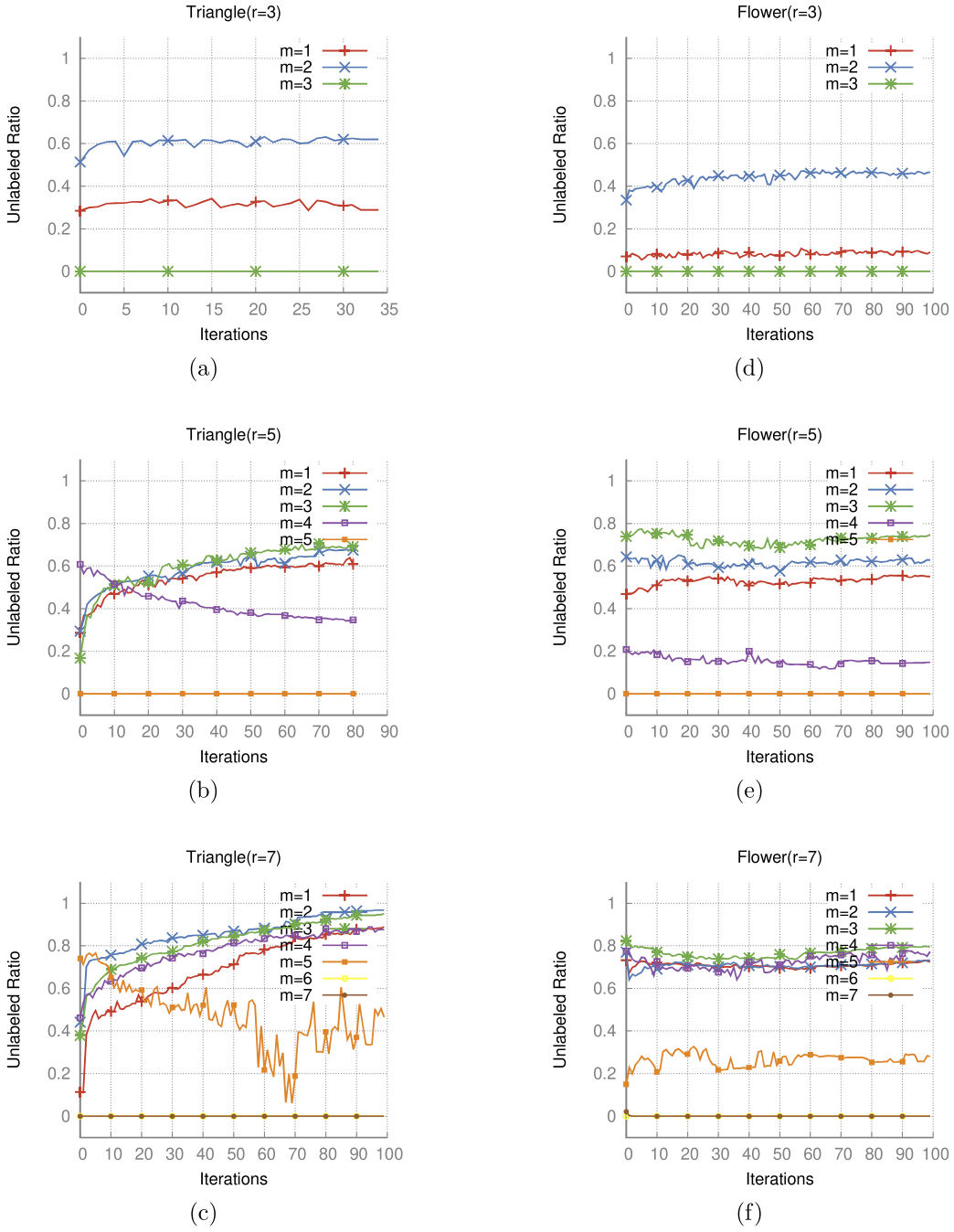


Figure 2.4: For each plot, we first produce the sequence of shapes $\{D^{(k)}\}$ executing FlipFlow with $m = r$. Then, for each shape in $\{D^{(k)}\}$, we execute one iteration of FlipFlow for different values of m and we count the unlabeled pixels. The number of unlabeled pixels by QPBOP remains high for lower values of m, and goes to zero when $m = r$. We observe the same behavior for varying radius values.

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 2.8.

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 **??**), but we were not able to prove its multigrid convergence. However, this relation suggests that curvature information is at least qualitatevely present in the outer balls computation, and this computation is preferred, as it is easier to optimize acordingly to the experiments illustrated in Figures 2.4 to 2.7.

2.4 Data term and image segmentation

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

$$g(x_{w(p)}) = -x_{w(p)} \log H_f(p) - (1 - x_{w(p)}) \log H_b(p) \tag{2.5}$$

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 (2.5).

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 2.9. We present a more exhaustive list of experiments and comparisons with other methods in Chapter 5.

2.5 Conclusion

We presented the FlipFlow model in Section 2.1, an evolution process that produces shapes of decreasing digital Elastica until a certain inflexion point. Algorithm 2 operates in two distint modes: the shrinking and the dilation modes. They are responsible for the treatment of convexities and concavities regions, respectively.

Section 2.2 briefly describes some properties of QPBO method and its variants used to solve the quadratic binary problem that appears at every iteration of Algorithm 2. In Section 2.3 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. Finnaly, we extended Algorithm 2 in order to include a data term regularization and do image segmentation using Algorithm 3.

In the next chapter we identify the key concept behind the FlipFlow model to define an easier to implement version of it. a variant of this algorithm which is easier to implement.

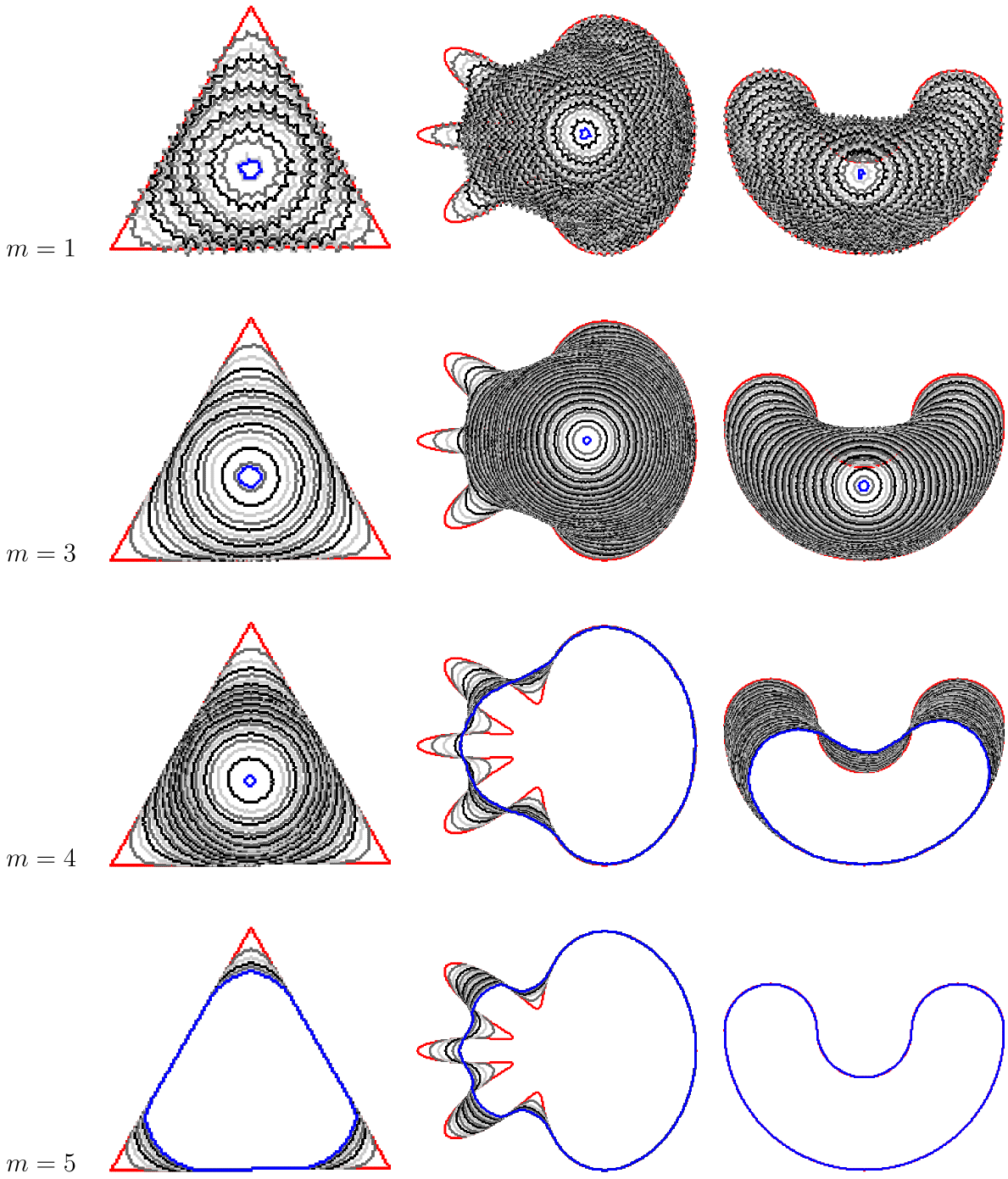


Figure 2.5: By positioning the estimation ball on outer rings, we minimize artifacts creation. The radius of the estimation ball used here equals ~~to~~ 5 and the curves are displayed every 10 iterations.

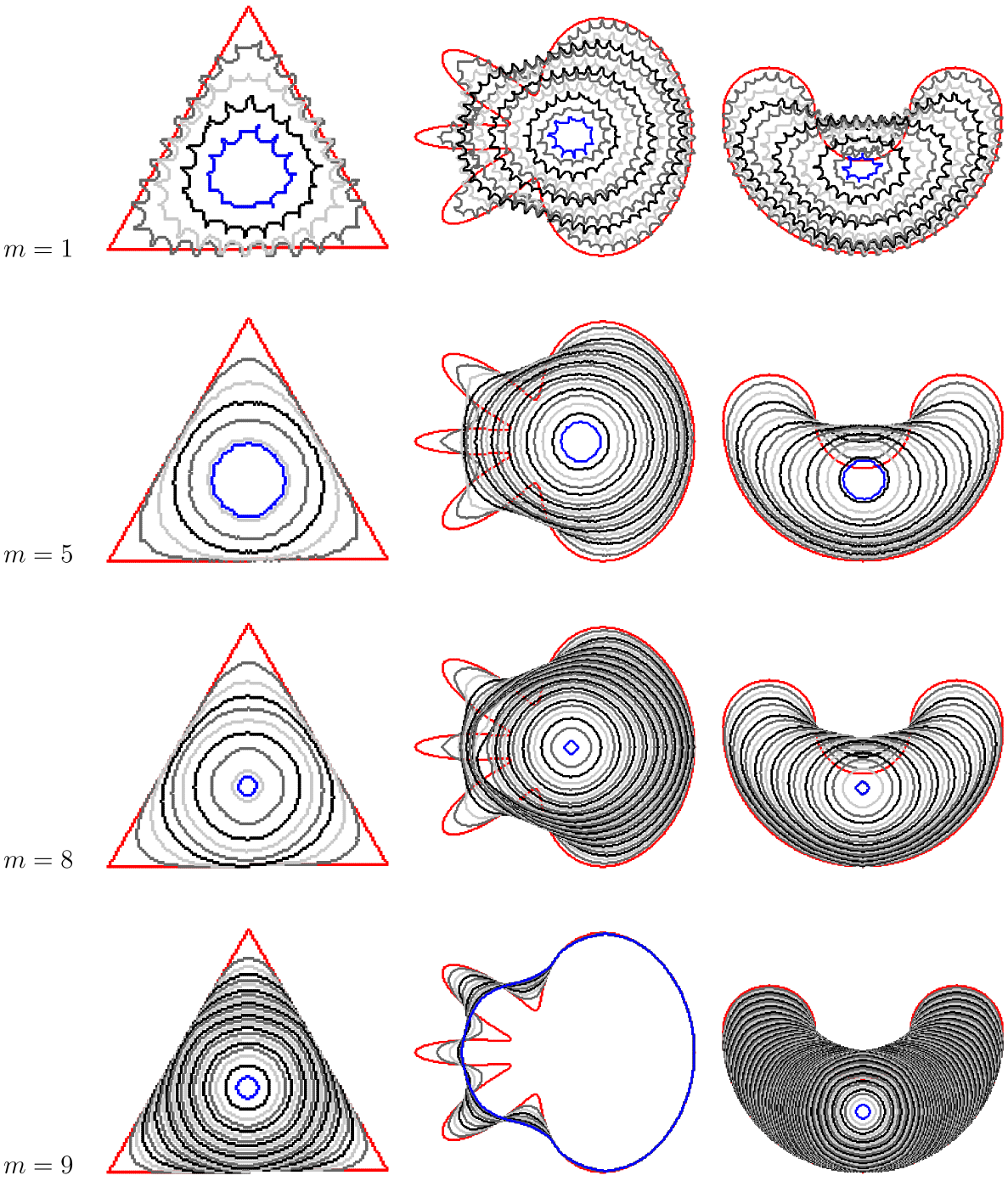
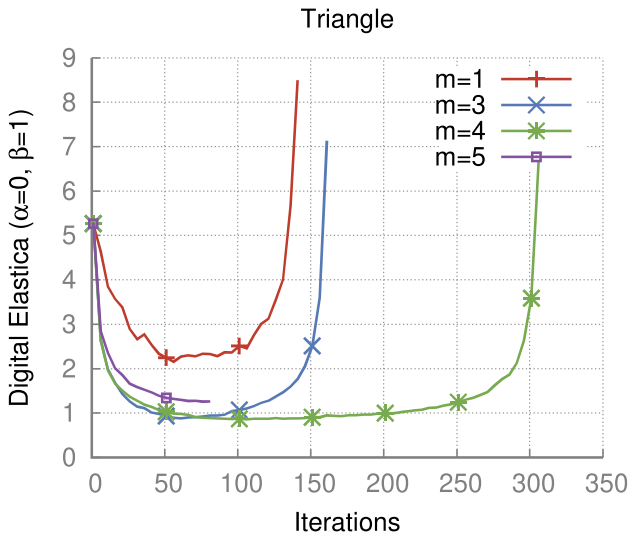
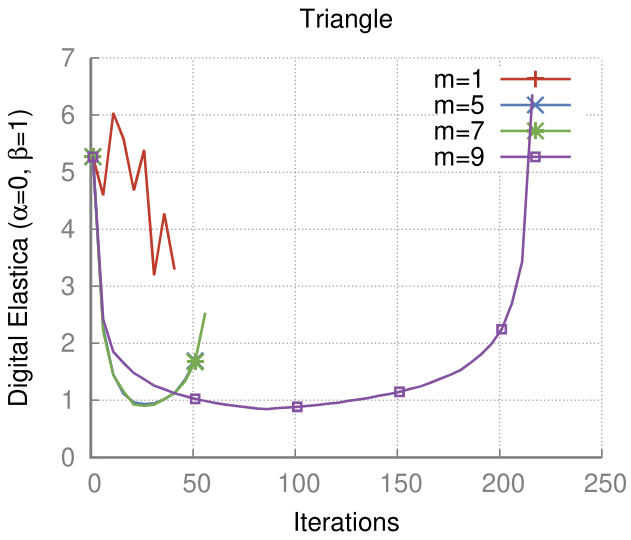


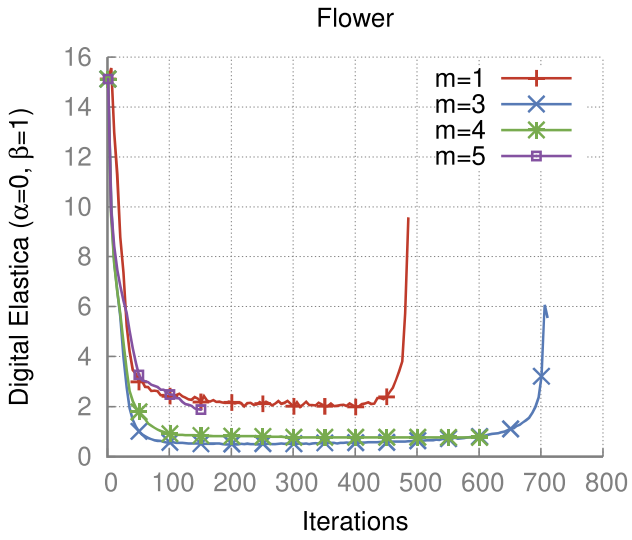
Figure 2.6: By positioning the estimation ball on outer rings, we minimize artifacts creation. The radius of the estimation ball used here equals ~~to~~ 9 and the curves are displayed every 10 iterations.



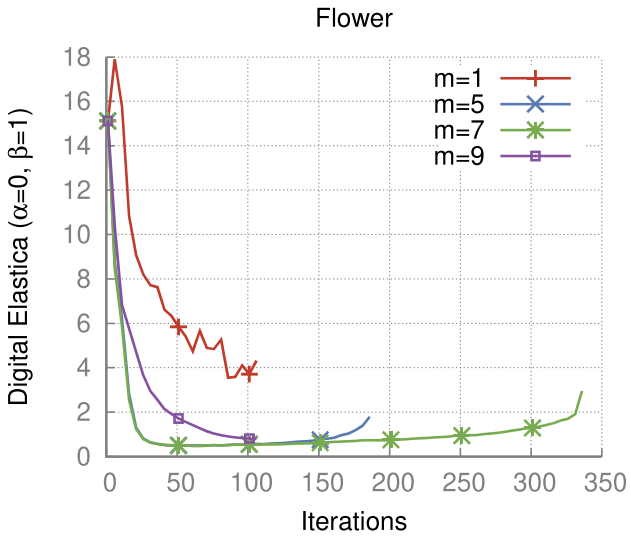
(a)



(b)

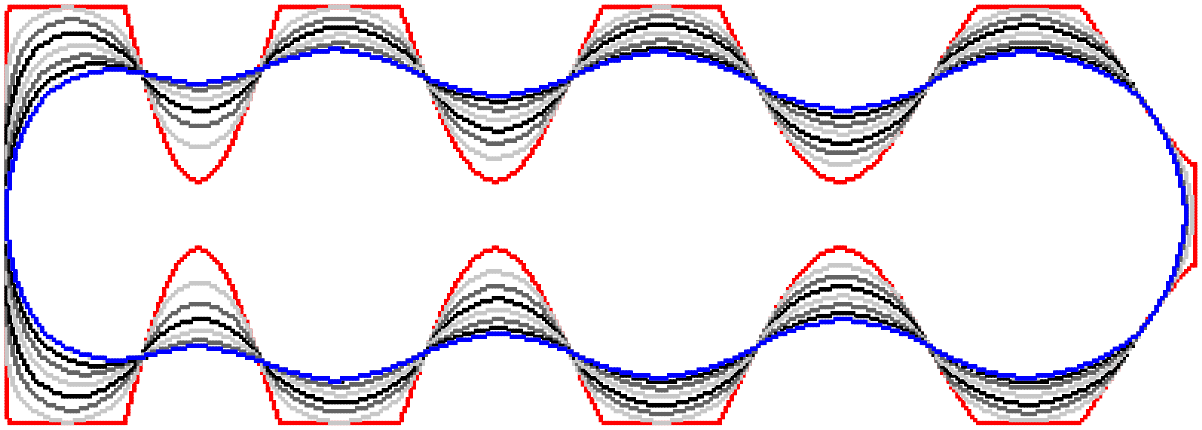


(c)

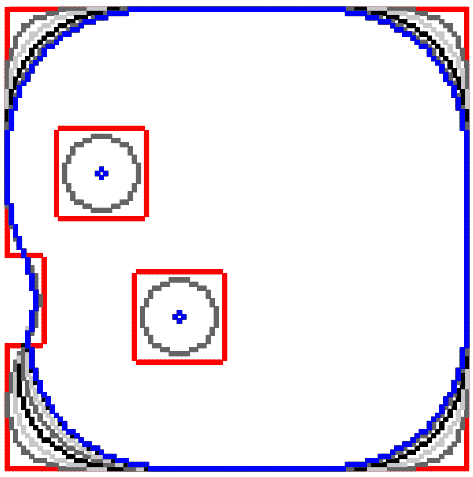


(d)

Figure 2.7: Digital Elastica (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.



(a)



(b)

Figure 2.8: 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.

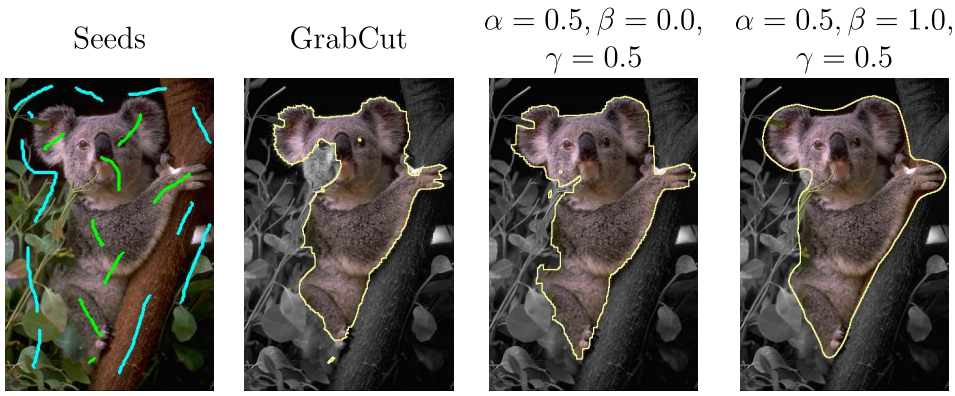


Figure 2.9: Given foreground (green) and background (blue) seeds at picture (a); GrabCut produces picture (b) which is used as input of the Contour Correction algorithm; in pictures (c) and (d) we display the output of Contour Correction algorithm with and without squared curvature regularization.

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

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

Algorithm 3: FlipFlow algorithm for segmentation.

Chapter 3

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 curvatute: 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 a simpler algorithm to implement.

leads to

3.1 BalanceFlow model

In Section 2.1.3, 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.

3.1.1 Definitions

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

by defining

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

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

The balance coefficient definition is very similar to the II squared curvature estimator. Nonetheless, we have decided to make a new definition since we don't 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.

to

Make a short title also

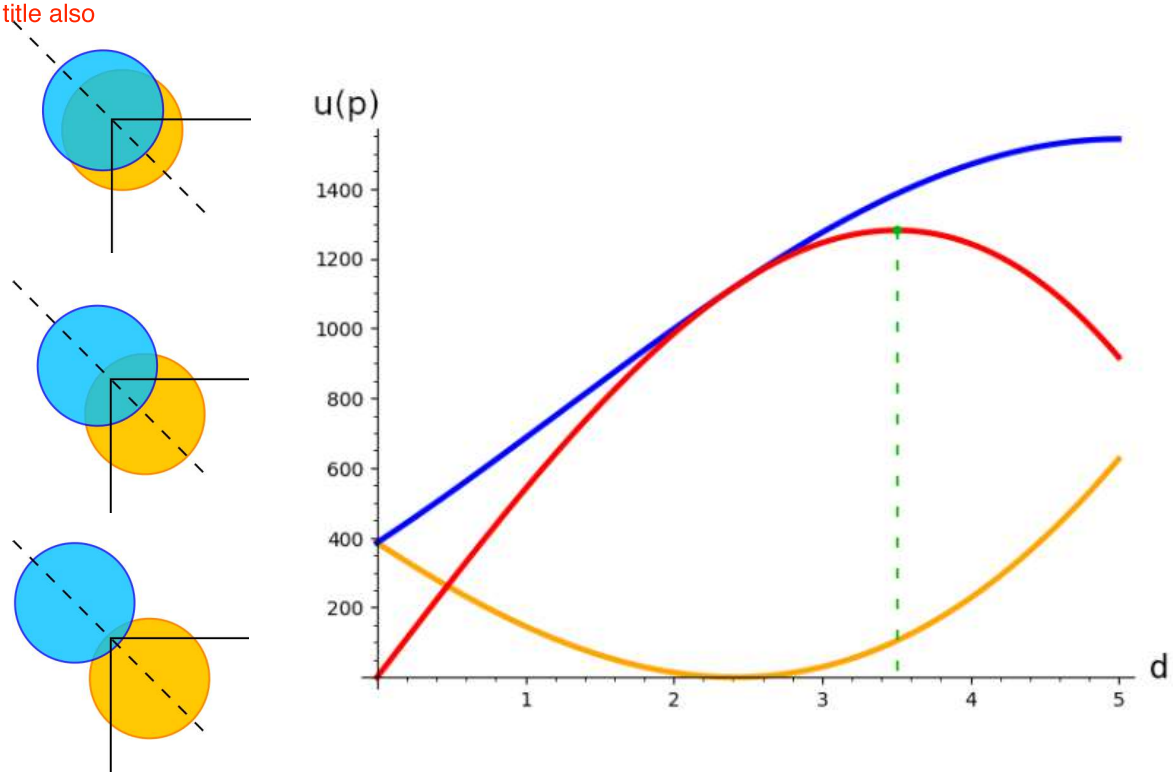


Figure 3.1: The balance coefficient ($r = 5$) of equidistant points from the corner os a square shape contour. The red curve indicates the difference between the blue and the orange curve. The inflexion point in the red curve indicates the point p in which a ball centered at p has balance coefficient equal to zero. The green point marks the point p in which the difference between the balance coefficients is the greatest.

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

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 3.1 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:

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

We recall that an assignment of variables $X(\Omega)$ is denoted $x(\Omega)$. In case the shape is described in terms of a set of variables $X(\Omega)$, i.e., $D = F \cup x(\Omega)$, the ballance coefficient is written as

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

3.1.2 Algorithm

Let $p_i, p_o \in \mathcal{R}_m(D)$ the inner and outer ball centers in the m-ring of D, respectively. We assume $k = 0$ if omitted, i.e., $D = D^{(0)}$. We define the term $T_m^{bal}(D)$ as

$$
\begin{aligned}
T_m^{bal}(D, X) &= \sum_{p_i, p_o \in \mathcal{R}_m(D)} u_r(F, 1 - X, p_o) - u_r(F, X, p_i) \\
&= \sum_{p_i, p_o \in \mathcal{R}_m(D)} \left(\frac{\pi R^2}{2} - \Big(|F_o| + \sum_{x_j \in X_o} 1 - x_j \Big)\right)^2 - \left(\frac{\pi R^2}{2} - \Big(|F_i| + \sum_{x_j \in X_i} x_j \Big)\right)^2,
\end{aligned} \tag{3.1}
$$

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 (3.1) favors configurations of zero difference between the outer and inner ball coeficients, 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 (3.1) tends to label variables in these regions to zero (one).

The m-balance energy family is defined as

$$
E_{(\boldsymbol{\theta}, m)}^{bal}(D^{(k)}, X^{(k)}) = \sum_{x_j \in X^{(k)}} \alpha s(x_j) + \beta T_m^{bal}(D^{(k)}, X^{(k)}) + \sum_{x_j \in X^{(k)}} \gamma g(D^{(k)}, x_j). \tag{3.2}
$$

We follow the same notation of Chapter 2 to denote the data term $g(D, x)$ and the length penalization term $s(x)$ that are defined as in Equations (2.4) and (2.5), respectively. The BalanceFlow algorithm is summarized in Algorithm 4 and an illustration is shown in Figure 3.2.

input : A digital set D; The ring number m; parameter vector $\boldsymbol{\theta} = \alpha, \beta)$; data term coefficient γ; the maximum number of iterations maxIt;

$D^{(0)} \longleftarrow D$;
$k \longleftarrow 1$;
while k $<$ maxIt **do**
  $x^{(k-1)} \longleftarrow \arg\min_{X^{(k-1)}} E^{bal}_{(\boldsymbol{\theta},m)}(D^{(k-1)}, X^{(k-1)}) + \gamma g(X^{(k-1)})$;
  $D^{(k)} \longleftarrow F^{(k-1)} + x^{(k-1)}$;
  k $\longleftarrow$ k +1;
end

Algorithm 4: BalanceFlow algorithm.

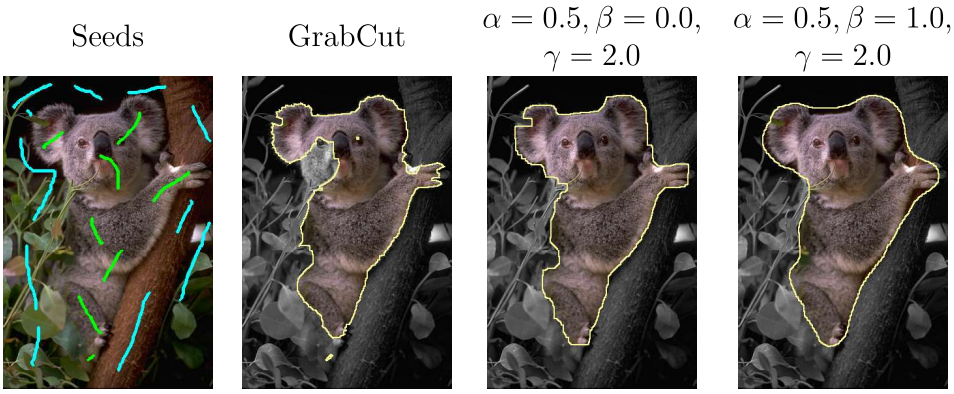


Figure 3.2: Given foreground (green) and background (gray) seeds at picture (a); GrabCut produces picture (b) which is used as input of the BalanceFlow algorithm; in pictures (c) and (d) we display the output of BalanceFlow algorithm with and without squared curvature regularization.

3.2 Relation with FlipFlow

The BalanceFlow returns similar solutions to the FlipFlow algorithm, as the experiments in Chapter 5 illustrates. Indeed, they are closely related. We recall the curvature regularization term of the FlipFlow Equation (2.3) for some digital shape D before proving proposition

$$T_m^{flip}(D,X) = \sum_{p_i,p_o \in R_m(D)} \Big(\frac{\pi R^2}{2} - F_r(p_o) - \sum_{x_j \in X_r(p_o)} x_j\Big)^2 + \Big(\frac{\pi R^2}{2} - F_r(p_i) - \sum_{x_j \in X_r(p_i)} x_j\Big)^2 \tag{3.3}$$

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 $P_1(X_i) = (\sum_{X_i} 1 - x_j)^2 + (\sum_{X_i} x_j)^2$.

Proof:

by We start by replacing x_j to $(1-x_j)$ in Equation (3.3), which corresponds to ~~take~~ 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 (3.3) as

$$T_m^{flip}(D,1-X) = \sum_{p_i,p_o \in R_m(D)} \Big(\frac{\pi R^2}{2} - F_o - \sum_{x_j \in X_o} 1 - x_j\Big)^2 + \Big(\frac{\pi R^2}{2} - F_i - \sum_{x_j \in X_i} 1 - x_j\Big)^2 \tag{3.4}$$

Next, let $A_i = \pi R^2/2 - |F_i|$. We rewrite the second term of Equation (3.4) as

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

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

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

Finnaly, we replace Equation (3.5) in Equation (3.4) to obtain

$$\begin{aligned}T_m^{flip}(D, 1 - X) &= \left(\frac{\pi R^2}{2} - \Big(\, |F_o| + \sum_{x_j \in X_o} 1 - x_j \,\Big)\right)^2 - \left(\frac{\pi R^2}{2} - \Big(\, |F_i| + \sum_{x_j \in X_i} x_j \,\Big)\right)^2 + P_1(X_i) + c \\ &= T_m^{bal}(D) + P_1(X_i) + c. \qquad (3.5)\end{aligned}$$

■

Proposition 1 tell us that, apart from a constant, the two energies differ in the penalty term $P_1(X_i)$. Locally, such term favors solutions in which half of the variables touched by the inner ball are labeled one. Indeed, the FlipFlow and BalanceFlow behave similarly. In the next chapter we make extensive comparisons between them.

3.3 Towards a graph-based formulation

Looking at Figure 3.1 it would be interesting to find the point in which the inner or outer ball reaches the zero balance. In Figure 3.3 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 which $u_{12}(D, p) \leq 169$. Figure 3.3 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.

3.4 Conclusion

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

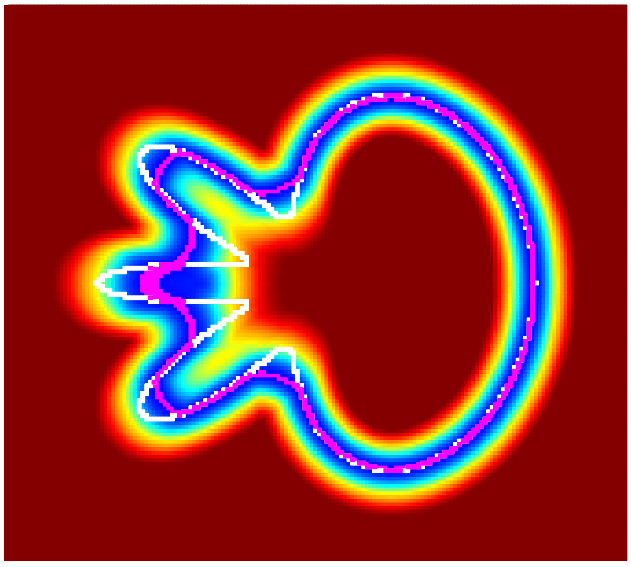


Figure 3.3: Balance coefficient map of radius 12 for the flower shape (contour in white). The pixels in magenta have balance coefficient smaller than 169.

Chapter 4

Digital Elastica minimization via graph cuts

In the previous chapter we have defined the concept of balance coefficient that motivate 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 in the free digital Elastica problem. Moreover, the model is easily adapted for image segmentation tasks.

motivates

for

to

4.1 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 fastly computed for sparse graphs. We start by giving some necessary definitions from graph theory.

quickly

4.1.1 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 $p \in D$ ~~to~~ correspond to a vertex $v_p \in \mathcal{V}$. The set of edges is defined accordingly

according to

with the application in mind, but it~~'s~~ encodes the pixels connectivity, i.e., $4, 8$-adjacent. adjacency Moreover, one can define a cost function $w : \mathcal{E} \to \mathbb{R}$ 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 $w : \mathcal{E} \to \mathbb{R}$. 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

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

by (or in) We are interested on (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 accordingly with its color intensity values. according to

4.1.2 Grabcut binary segmentation model

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

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

where

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

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

$$B(\{v_p, v_q\}) = exp\left(-\frac{\big(I(p) - I(q)\big)^2}{2\sigma^2}\right)$$
$$R(v_p, \text{“}bg\text{”}) = -ln H_B(p) \qquad \$\backslash ln \ldots\$$$
$$R(v_p, \text{“}fg\text{”}) = -ln H_F(p), \qquad \$\backslash ln \ldots\$$$

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.

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

edge e	**w(e)**	**for**
$\{v_p, v_q\}$	$B(e)$	$e \in \mathcal{E}_{\mathcal{N}}$
$\{v_p, s\}$	$\lambda_r R(v_p, \text{“bg”})$	$p \in \mathcal{V}, p \notin \mathcal{V}_F \cup \mathcal{V}_B$
	M	$p \in \mathcal{V}_F$
	0	$p \in \mathcal{V}_B$
$\{v_p, t\}$	$\lambda_r R(v_p, \text{“fg”})$	$p \in \mathcal{V}, p \notin \mathcal{V}_F \cup \mathcal{V}_B$
	0	$p \in \mathcal{V}_F$
	M	$p \in \mathcal{V}_B$

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

be a Let $\mathcal{E}'$ the 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

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

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 [NTG+14, EZG10] 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 coeficient to regularize curvature.

4.2 GraphFlow model

Given some energy E, the GraphFlow model is divided in 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 select the candidate solution with the lowest value of E.

into

selecting

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

$$E^{graph} = \hat{E}_{\boldsymbol{\theta}} + grab_{\Lambda}. \tag{4.1}$$

4.2.1 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

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

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. Additionaly, we define the sets

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

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

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

The constants M_1, M_2 are defined as

$$\begin{aligned}M_1 &= 1 + \max_{p \in \Omega} \sum_{q \in \mathcal{N}_4(p)} B(\{p, q\}) \\ M_2 &= 1 + \max w(p, q).\end{aligned}$$

Let $\mathcal{E}'$ the minimum cut of $\mathcal{G}_n(D)$. Moreover, let $C_n(D)$ 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 (4.1). 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 which neighborhood is defined using a set of candidate graphs.

4.2.2 GraphFlow algorithm

The GraphFlow algorithm implements a local-search strategy to minimize Equation (4.1). Its neighborhood is derived from a set of candidate graphs, called the probe-set. The probe-set of D can be defined as any colection of digital sets, but it makes more sense to define it as a colection 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 os D is defined as

$$\mathcal{P}_a(D) = D \cup \bigcup_{a'<a} D^{+a} \cup D^{-a},$$

where $D^{+a}(D^{-a})$ denotes a dilation(erosion) by a circle of radius a.

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

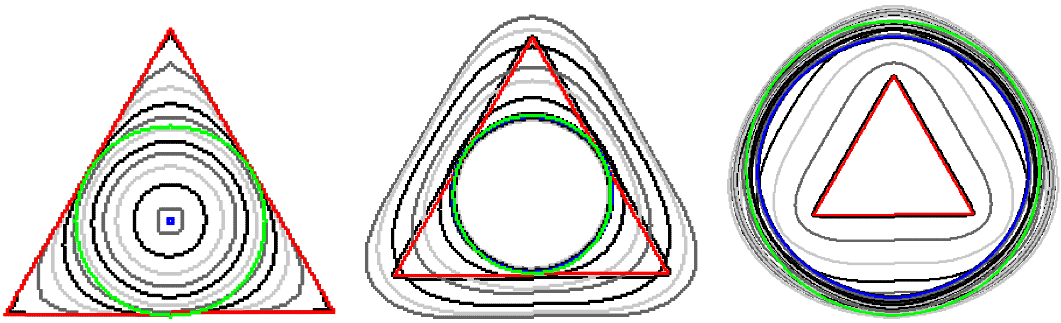
$$\mathcal{N}(D) = \{C_n(X) \mid Q \in \mathcal{P}_a(D)\}.$$

Finnaly, the GraphFlow algorithm is described in Algorithm 5.

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;

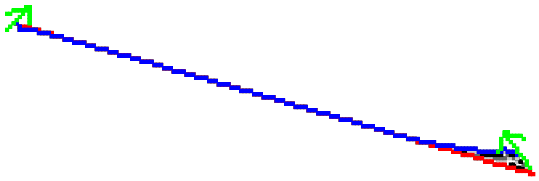
$D^{(0)} \longleftarrow D$;
$k \longleftarrow 1$;
while k < maxIt **do**
  //Candidate selection
  $\mathcal{N}^{(k)} \longleftarrow \left\{C_n(Q) \mid Q \in \mathcal{P}_a(D^{(k-1)})\right\}$;
  //Candidate validation
  $D^{(k)} \longleftarrow \arg\min_{X \in \mathcal{N}^{(k)}} \hat{E}_{\boldsymbol{\theta}}(X) + grab_{\Lambda}(X)$;
  k $\longleftarrow$ k +1;
end

Algorithm 5: GraphFlow algorithm.



$(n = 2, a = 0, \alpha = 0.01)$ $(n = 2, a = 2, \alpha = 0.01)$ $(n = 2, a = 2, \alpha = 0.001)$

(a)



(b) $(n = 2, a = 2, \alpha = 0.001)$

Figure 4.1: The GraphFlow algorithm can shrink and grow accordingly 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.

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 accordingly with the α coefficient in the digital Elastica (see Figure 4.1). For $a = 0$, we recover the convergence to a single point behaviour observed in both FlipFlow and BalanceFlow model. Moreover, its solution for the free Elastica problem is very similar to those given by the enumerative process of Chapter 1, i.e., the shapes converges to the expected global optimum. However, for the constrained Elastica problem, the GraphFlow encounters some difficulties to evolve (see Figure 4.1), 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 5.

Some preliminar results of Algorithm 5 applied in image segmentation is shown in Figures 4.2 and 4.3. In particular, we can observe that the GraphFlow presents the completion property, i.e., tends to return segmentation with fewer disconnected components.

4.3 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

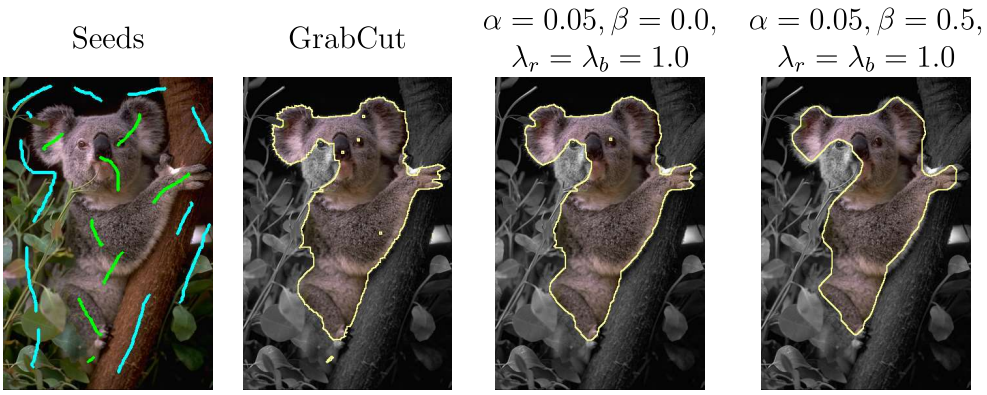


Figure 4.2: Given foreground (green) and background (gray) seeds at picture (a); GrabCut produces picture (b) which is used as input of the GraphFlow algorithm; in pictures (c) and (d) we display the output of GraphFlow algorithm with and without squared curvature regularization.

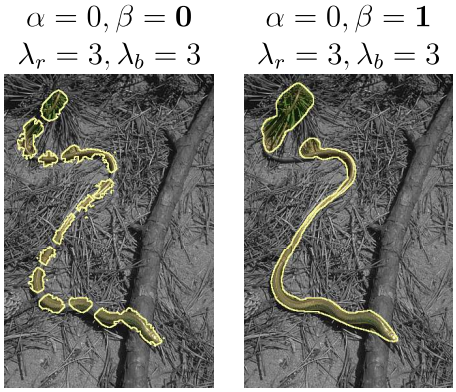


Figure 4.3: Completion property. The oversegmentation picture in the left was obtained with no squared curvature regularization, while the picture in the right was obtained by setting $\beta = 1.0$.

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 Algorithm 5 is faster and simpler to implement than the previous models presented in this thesis.

Chapter 5

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 5.1 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 5.1: Models summary. The qualitative attributes are relative, e.g., the GraphFlow presents the lowest running time while LocalSearch presents the highest.

5.1 Free Elastica

The free Elastica problem consists in to find a shape with the lowest digital Elastica. The approach to solve this problem as well the two others that follow, is to iteractively evolve a initial shape to another with lower digital Elastica value. We have ran two experiments, summarized in Table 5.2, 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 5.2 and 5.4. Moreover, the $vRadius$ is always scaled by the grid step, while the $bRadius$ is never scaled.

5.1.1 General experiment

The General experiment executes each model using the listed parameters in Table 5.2 for 5 different parametric shapes. The results for the General experiment is shown in Figure 5.1. One can check in the plots of Figure 5.2 how the digital Elastica value evolves at each

							LS	FF,BF	GF	
Experiment	$maxIt$	$vRadius$	$bRadius$	h	α	β	nc	m	a	ob
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 5.2: 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 5.3: Running time and input size of Exp-General experiment for the free Elastica.

iteration. For this experiment we also provide Table 5.3 with the model's respective running times.

We observe that both LocalSearch and GraphFlow evolves the initial shape to another 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, as Table 5.3 evidentiates. Even with a smaller neighborhood, the GraphFlow achieves its convergence first than LocalSearch in two ocassions, 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.

5.1.2 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 evidentiates 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 acoordingly. 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.

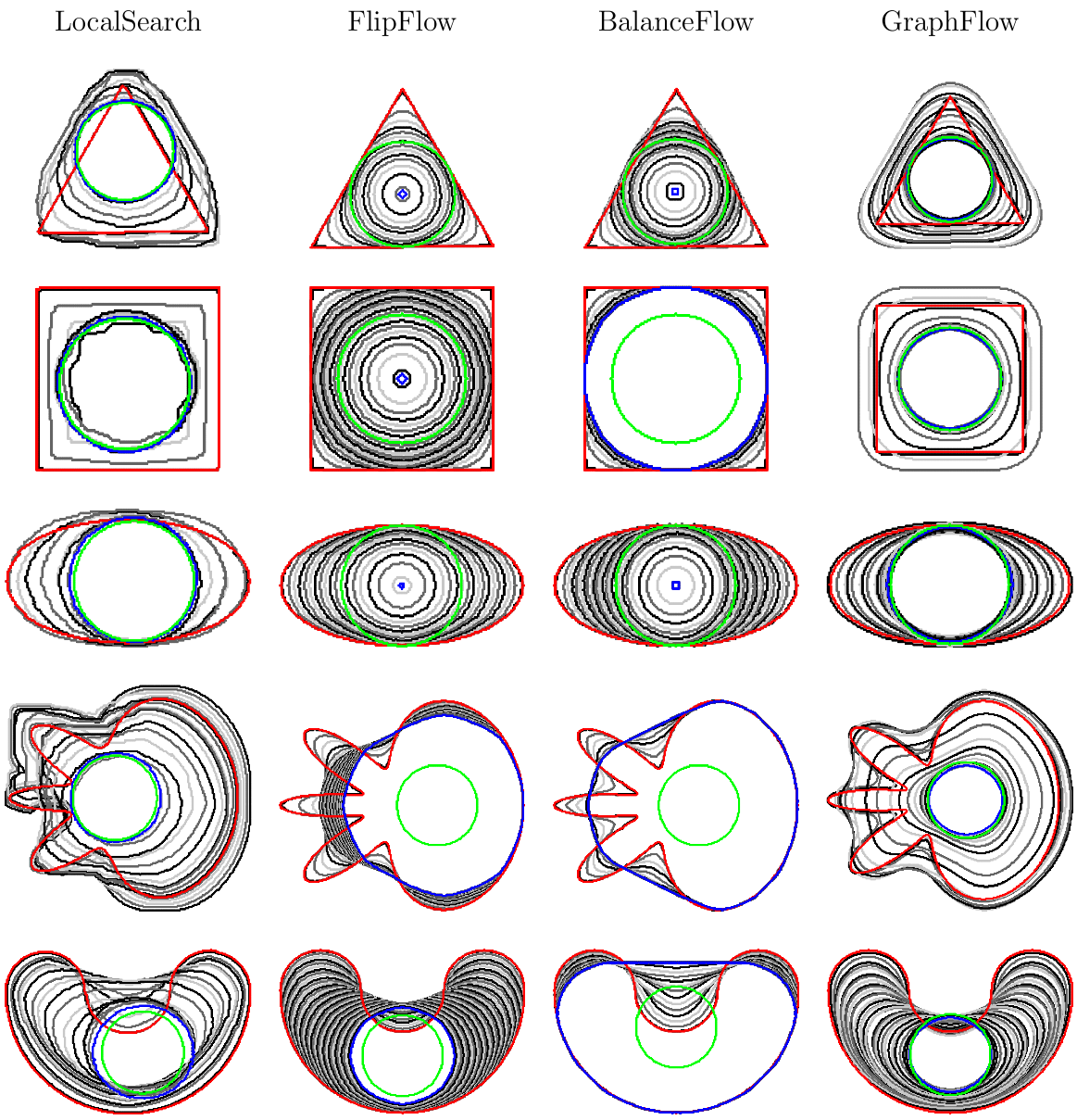


Figure 5.1: Results of Exp-General experiment for the free Elastica. Initial contour is colored in red, final contour is colored in blue and optimal contour is colored in green. Curves are drawn every 10 iterations.

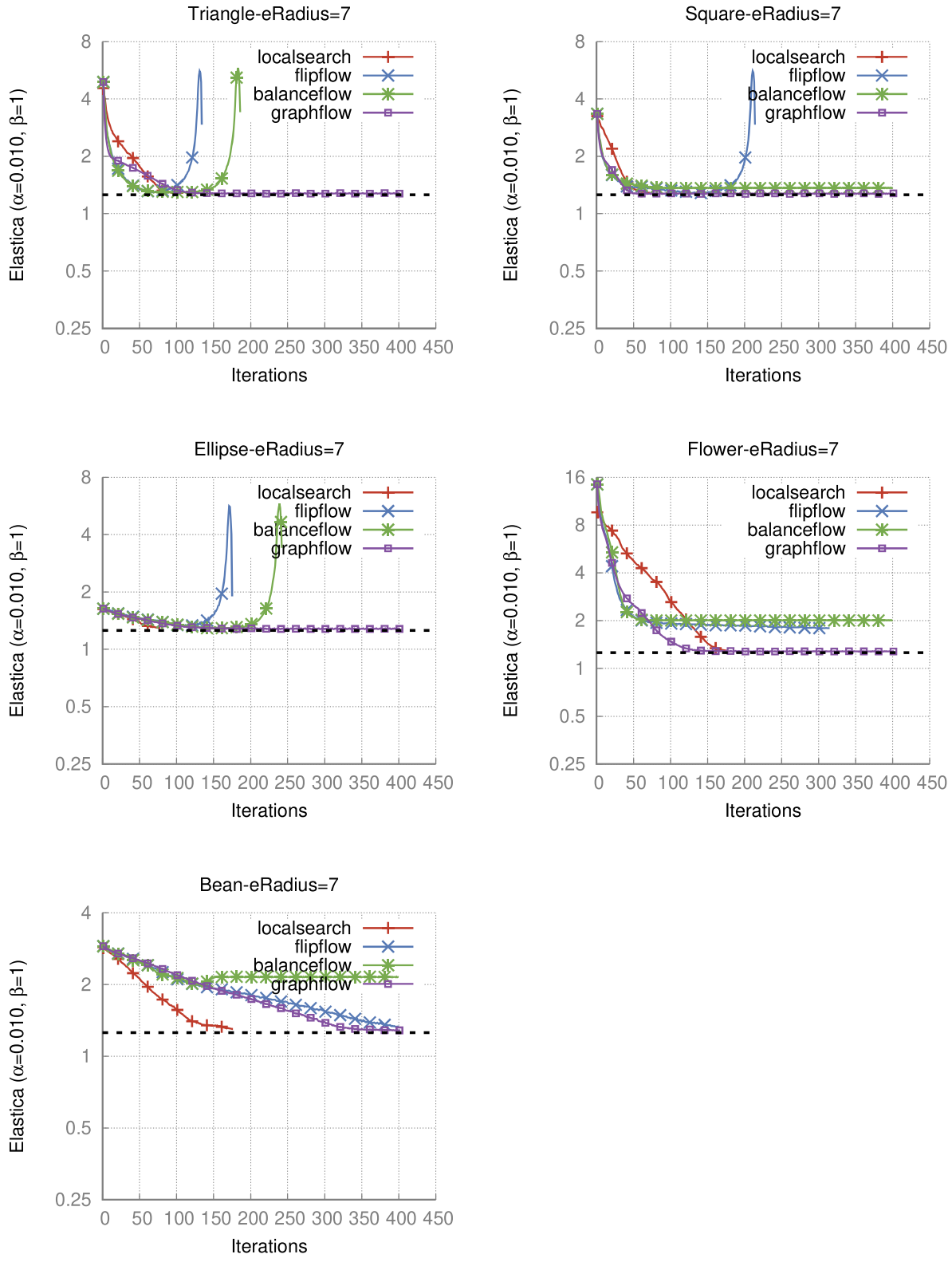


Figure 5.2: Evolution of digital Elastica value per iteration in the Exp-General experiment for the free Elastica. The dashed line marks the optimum value.

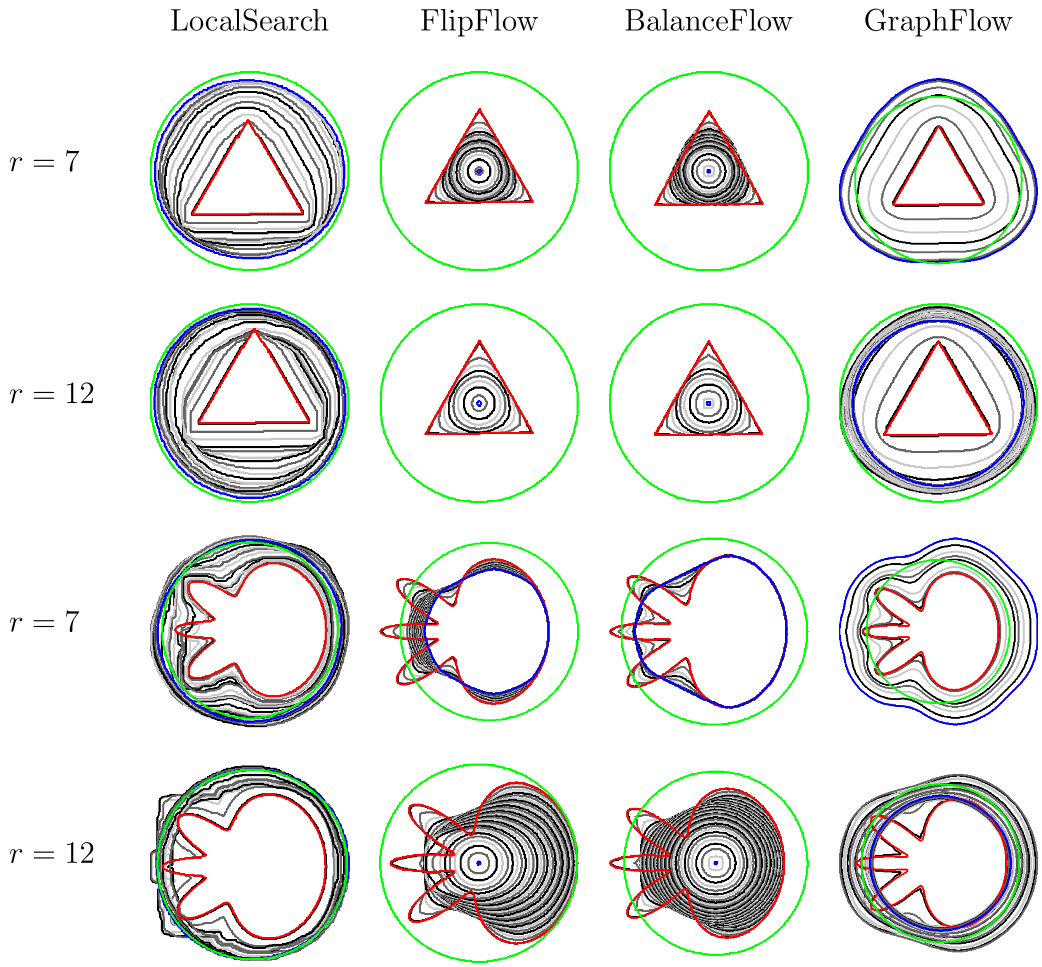


Figure 5.3: Results of Exp-Radius experiment for the free Elastica. Initial contour is colored in red, final contour is colored in blue and optimal contour is colored in green. Curves are drawn every 10 iterations.

However, for the Radius-choice experiment, a value $bRadius = 7$ is too small to identify the small variations that a shape that grows till become 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 5.3 and 5.4.

growing to

5.2 Constrained Elastica

The constrained Elastica problem consists in to find the shape of minimum digital Elastica that respects some set of constraints. We realized 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 which ~~the~~ endpoints are fixed.

finding

ran

whose

For the constrained Elastica, only LocalSearch and GraphFlow were evaluated. We believe that both FlipFlow and BalanceFlow can be modified to evolve the constrained

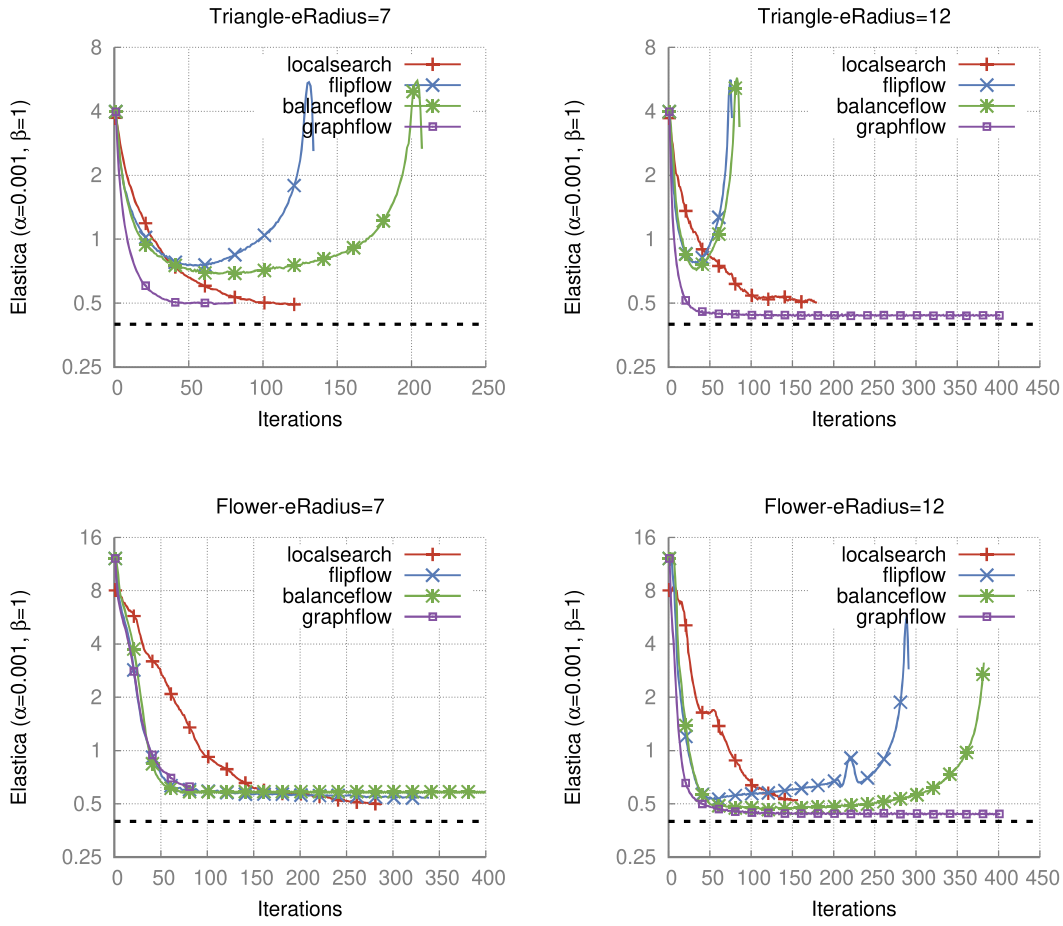


Figure 5.4: Digital Elastica value evolution per iteration of the Radius-choice experiment for the free Elastica.

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

Table 5.4: Parameter settings for the constrained Elastica experiments. The headers LS,GF identifies parameters that are exclusive for the LocalSearch and GraphFlow models, respectively.

Elastica, but such modifications were not implemented in this thesis. Table 5.4 lists the parameters' used in the experiments and Table 5.5 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. Differently from 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.

In opposition with

5.2.1 Discussion

Differently from LocalSearch, the GraphFlow model encounters some difficult to evolve both Curve-1 and Curve-2, as illustrated in Figure 5.6. 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.

difficulties

In Figure 5.7 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 large range of curvature values and it is particular 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 radius that is too large (see Figure 5.5) may not be appropriated, and the curves do not evolve as expected.

larger

particularly

5.3 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 5.6 lists the parameters configuration for each experiment and Table 5.8 summarizes their running times.

(a) Curve-1 (b) Curve-2 (c) Too large radius

Figure 5.5: 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).

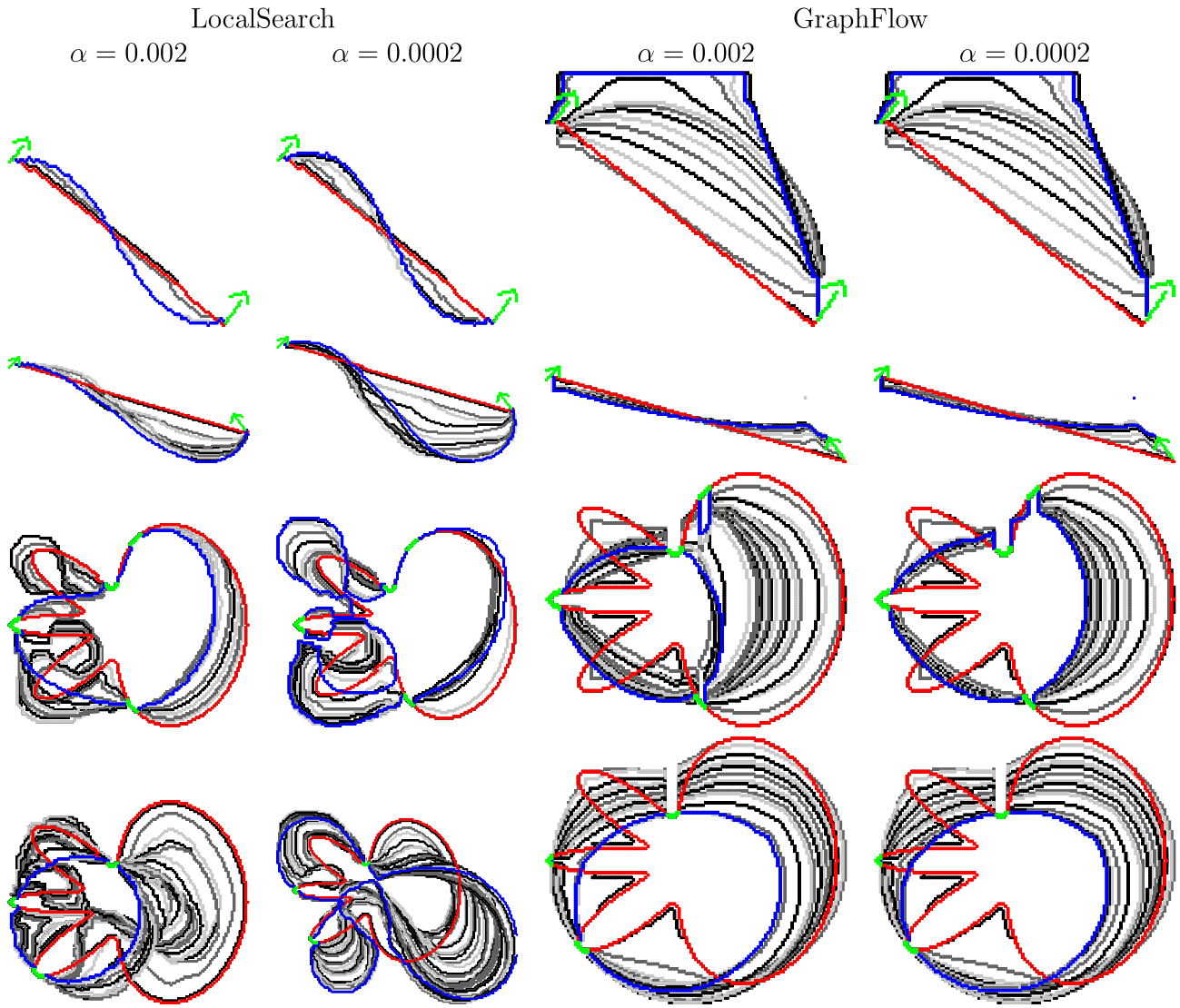


Figure 5.6: Results of Exp-α for the constrained Elastica. Initial contour is colored in red and final contour is colored in blue. The green pixels indicates pixels forced to persist in the final contour, or a forced orientation. Curves are drawn every 10 iterations.

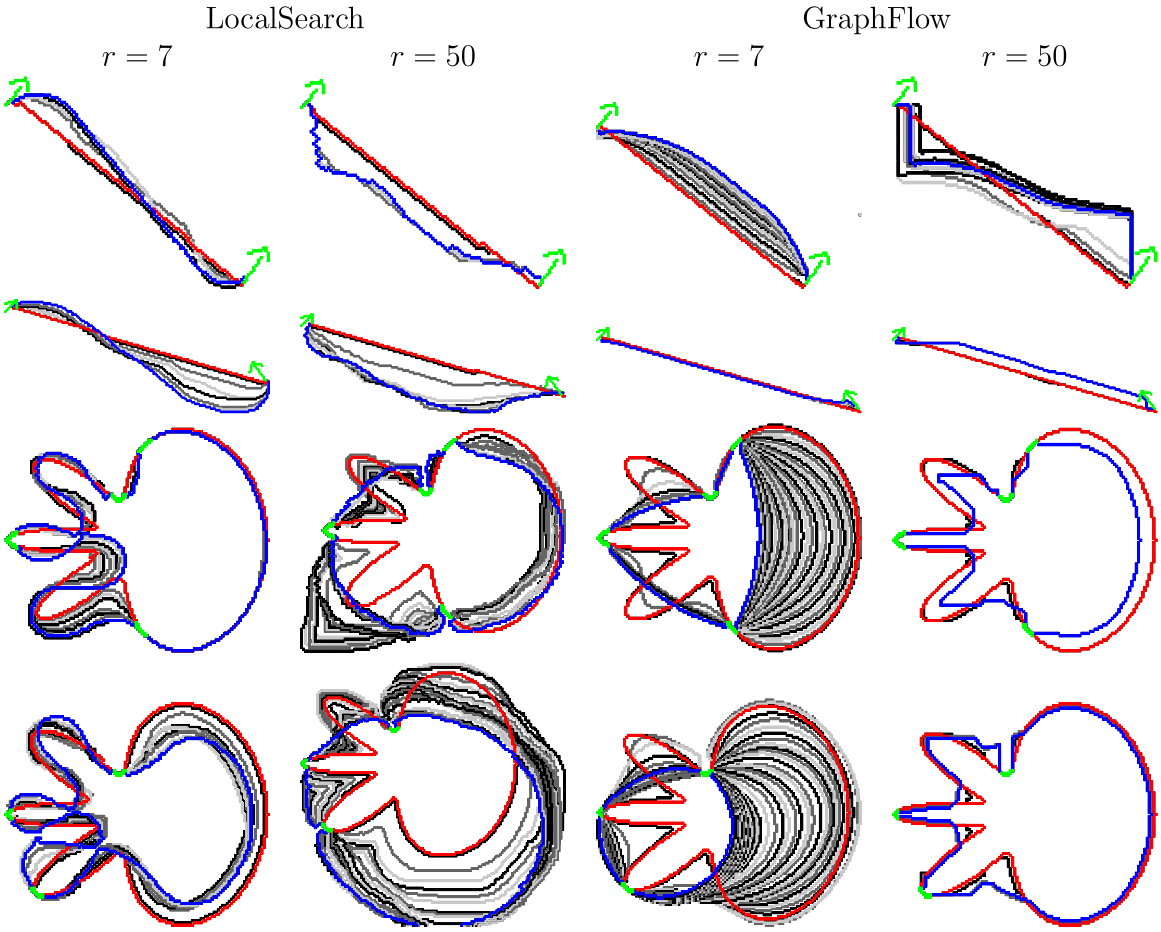


Figure 5.7: Results of Exp-*bRadius* for the constrained Elastica. Initial contour is colored in red and final contour is colored in blue. The green pixels indicates pixels forced to persist in the final contour, or a forced orientation. Curves are drawn every 10 iterations.

	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 5.5: Running time and input size of the Exp-α experiment for the constrained Elastica. The pixels columns is with respect the number of pixels in the shape. In the case of the curves, it is with respect the underlying shapes of Figure 5.5

respect to

							FF,BF		GF			
Experiment	*maxIt*	*vRadius*	*bRadius*	h	α	β	γ	d	a	ob	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 2 5	1 2 5
Exp-*rbRadius*	200	3 7 12	3 7 12	1.0	0	3	1	0	2	2	1	1

Table 5.6: Parameter settings for the image segmentation experiments. The headers FF,BF,GF identifies parameters that are exclusive of FlipFlow, BalanceFlow and GraphFlow models, respectively.

Exp-Comparison												
Model	*maxIt*	*vRadius*	*bRadius*	h	α	$\beta(\lambda)$	γ	d	a	ob	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 5.7: Parameter settings for the comparison experiments. The β parameter in FF,BF,GF corresponds to the λ parameter in SCLR.

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 $h = 1.0$.

5.3.1 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 a boundary term. The Grabcut input and its result are shown in Figure 5.8. The experiment results are displayed in Figures 5.9 to 5.12.

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.

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 5.8: Running time and input size of Exp-*bRadius* for the image segmentation problem and $bRadius = 7$.

Figure 5.8: Foreground (green) and background (blue) seeds are shown in the left and the resulting Grabcut segmentation int the right.

In experiment Exp-β we vary the squared curvature weigth coefficient. We observe in Figure 5.10 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 5.11. 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 5.12 we observe that a small radius results in contours with sharp changes of angle (first row), a consequence of the limited number of different estimations that can be given by a ball of small radius. As the radius increase, a richer variation of estimations is possible, and we have smoother contours (second-row). However, a big radius may omit important details of the object, as the coala's ears in the last row of Figure 5.12. That suggests that a multiradius approach may deliver improved segmentations.

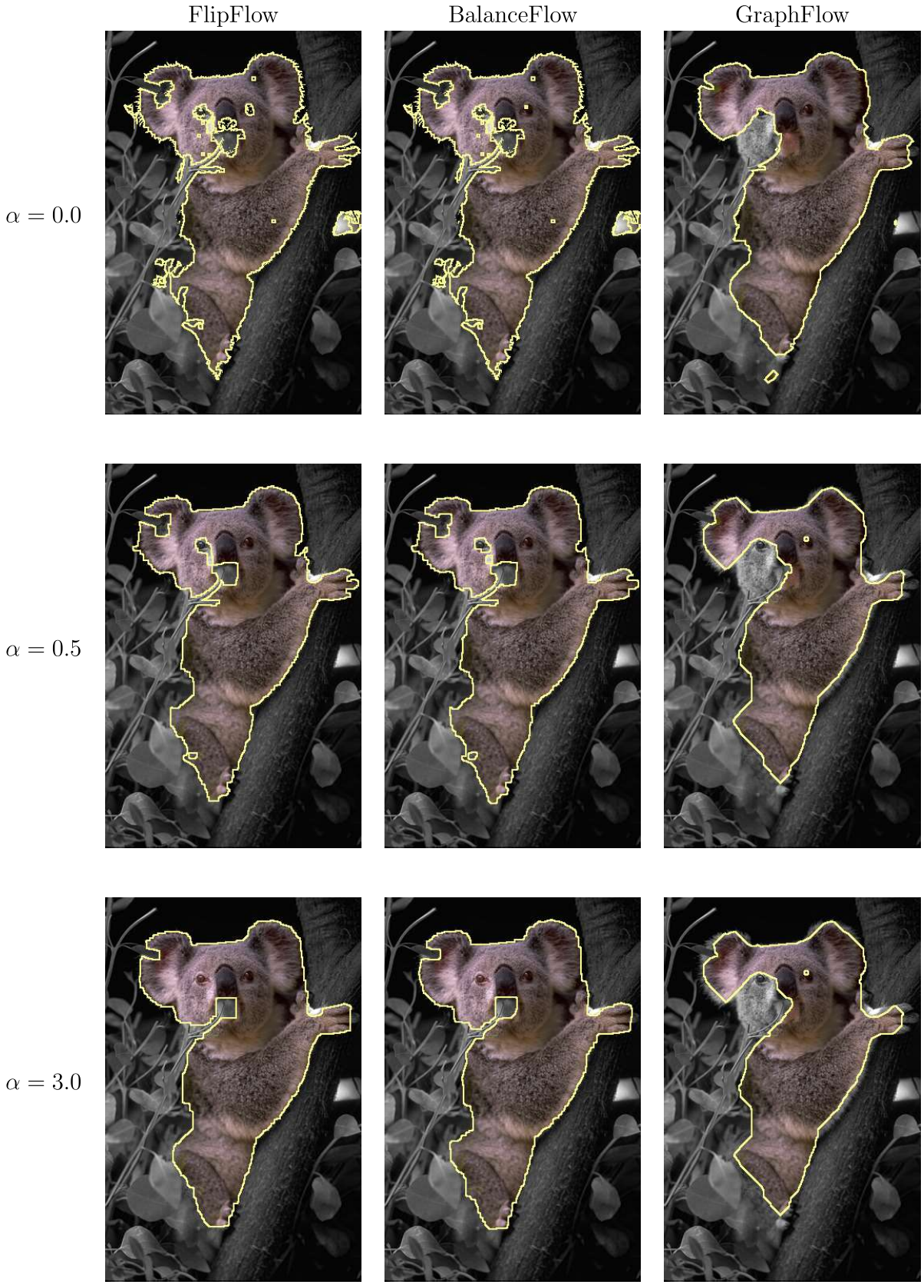


Figure 5.9: Exp-α

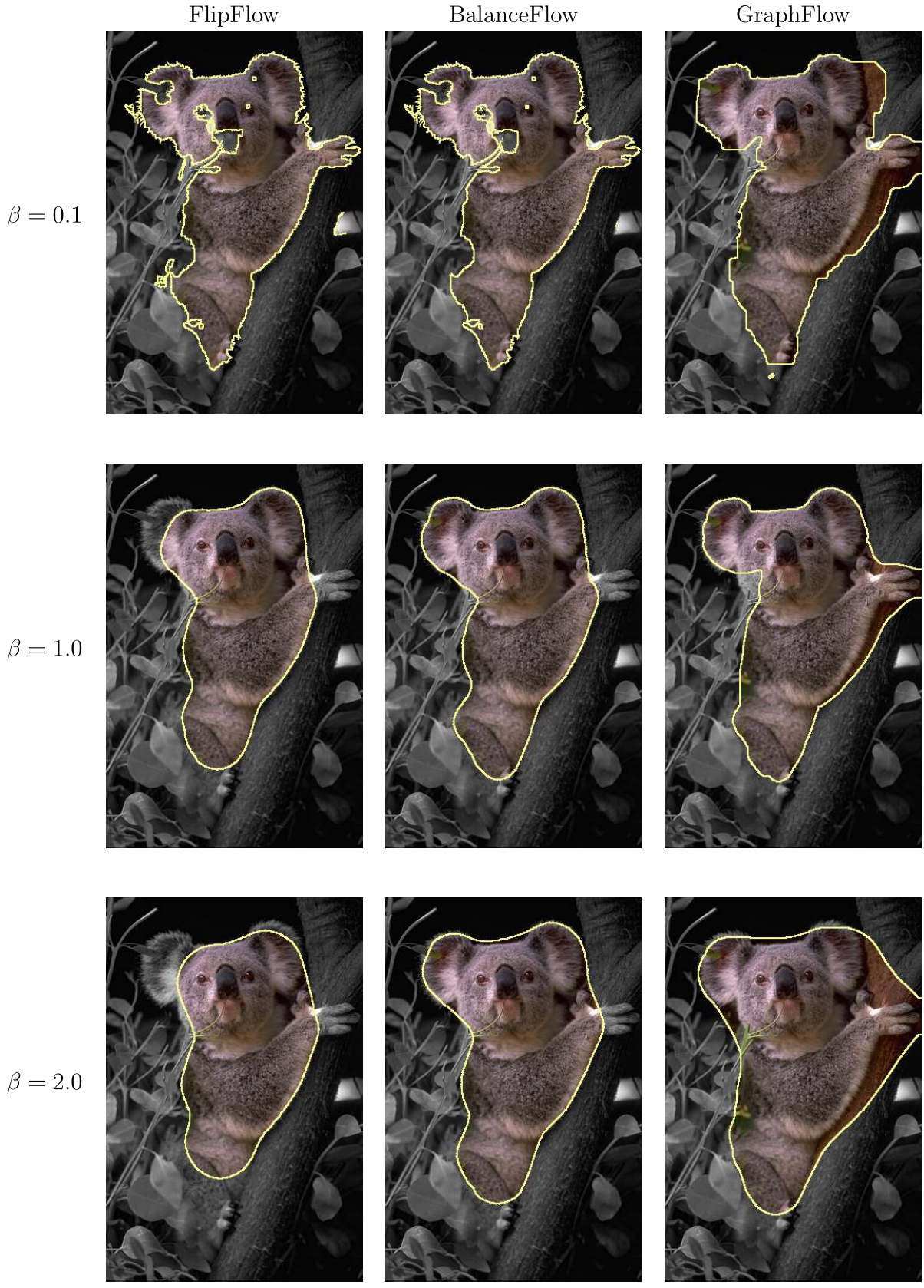


Figure 5.10: Exp-β

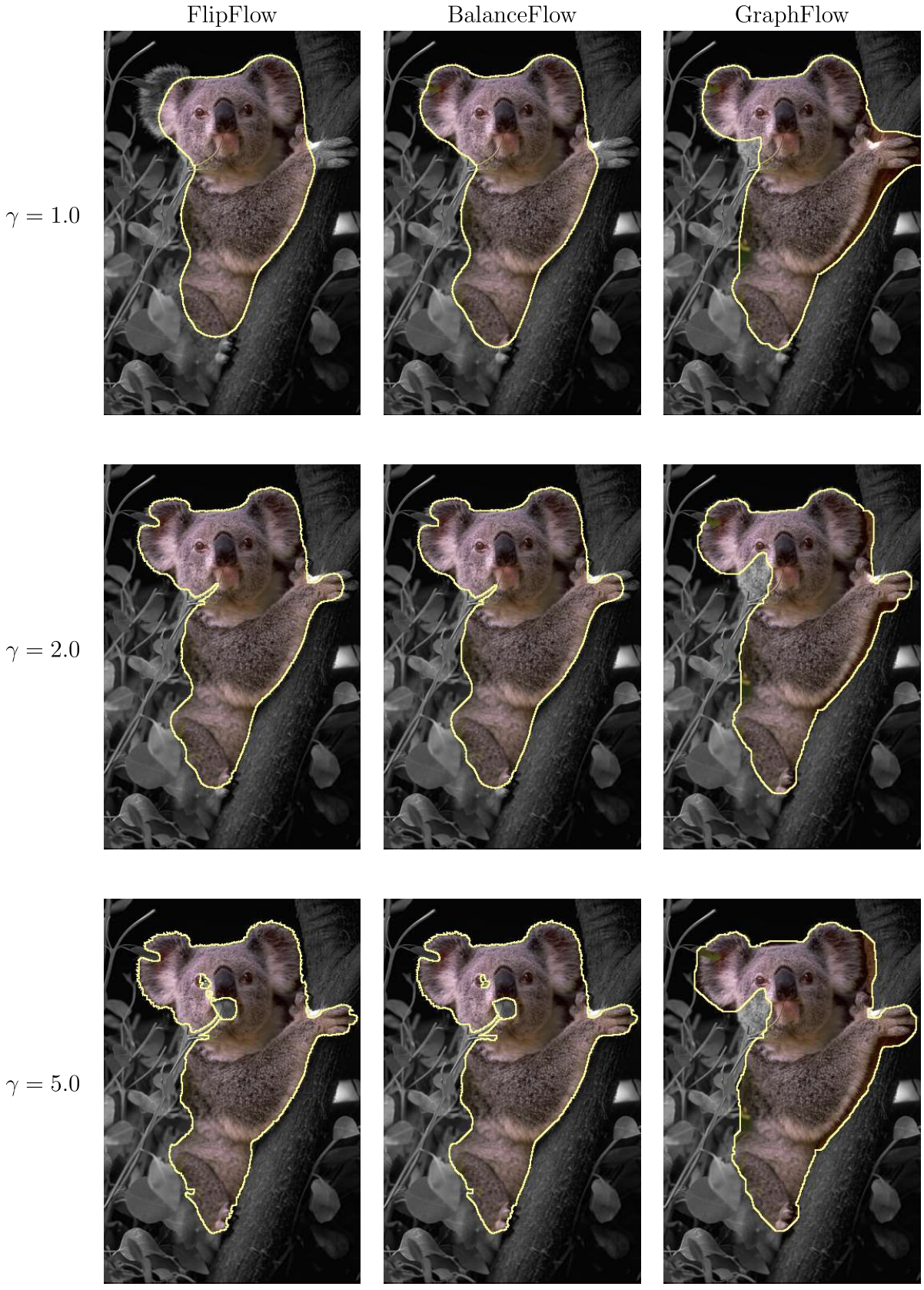


Figure 5.11: Exp-γ

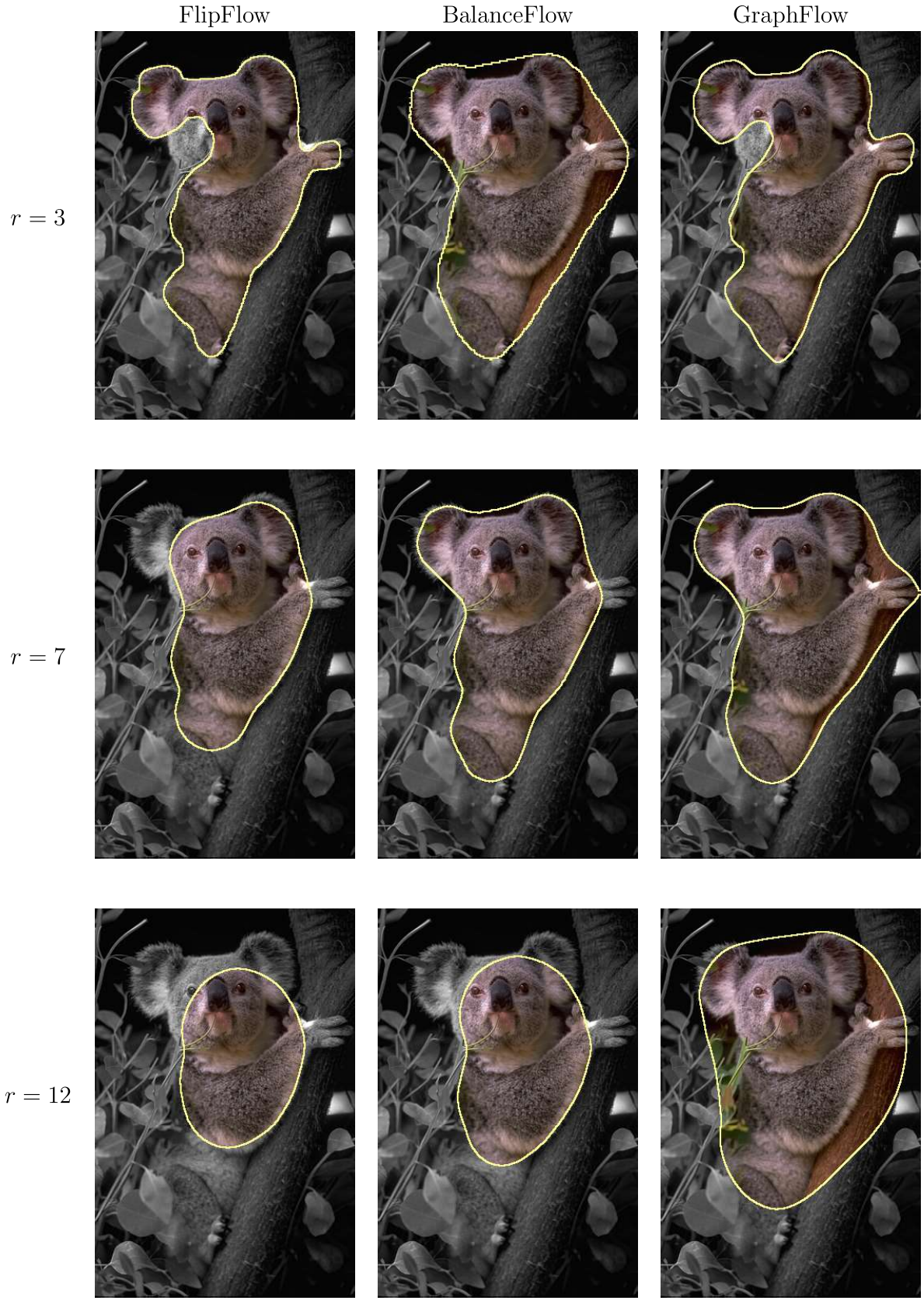


Figure 5.12: Exp-*bRadius*

5.3.2 Comparison

In Figures 5.13 to 5.16 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 5.16. They also have a hard time to segment the birds in Figure 5.14 due to the object large curvature interval jointly with the single radius approach adopted in this experiments. Nonetheless, they correctly segmented the green-snake in Figure 5.15 while the Grabcut return three disconnected components.

phrase ?

For the chosen parameters the GraphFlow did not evolved the Grabcut segmentationn to much, except for the brown-snake in Figure 5.16, which GF and SLCR presented the best segmentation. However, the GF has the ~~best~~ lowest running time among the four models.

for which

The SLCR tends to oversegment, notably in the man-in-white and the camel pictures. We observe that the contours present sharp turns due to the low precision of the curvature estimator. The precision could be improved by increasing the pixels connectivity (set to 8), but is likely to follow a increase in running time, which is already the highest between the models tested.

5.4 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.

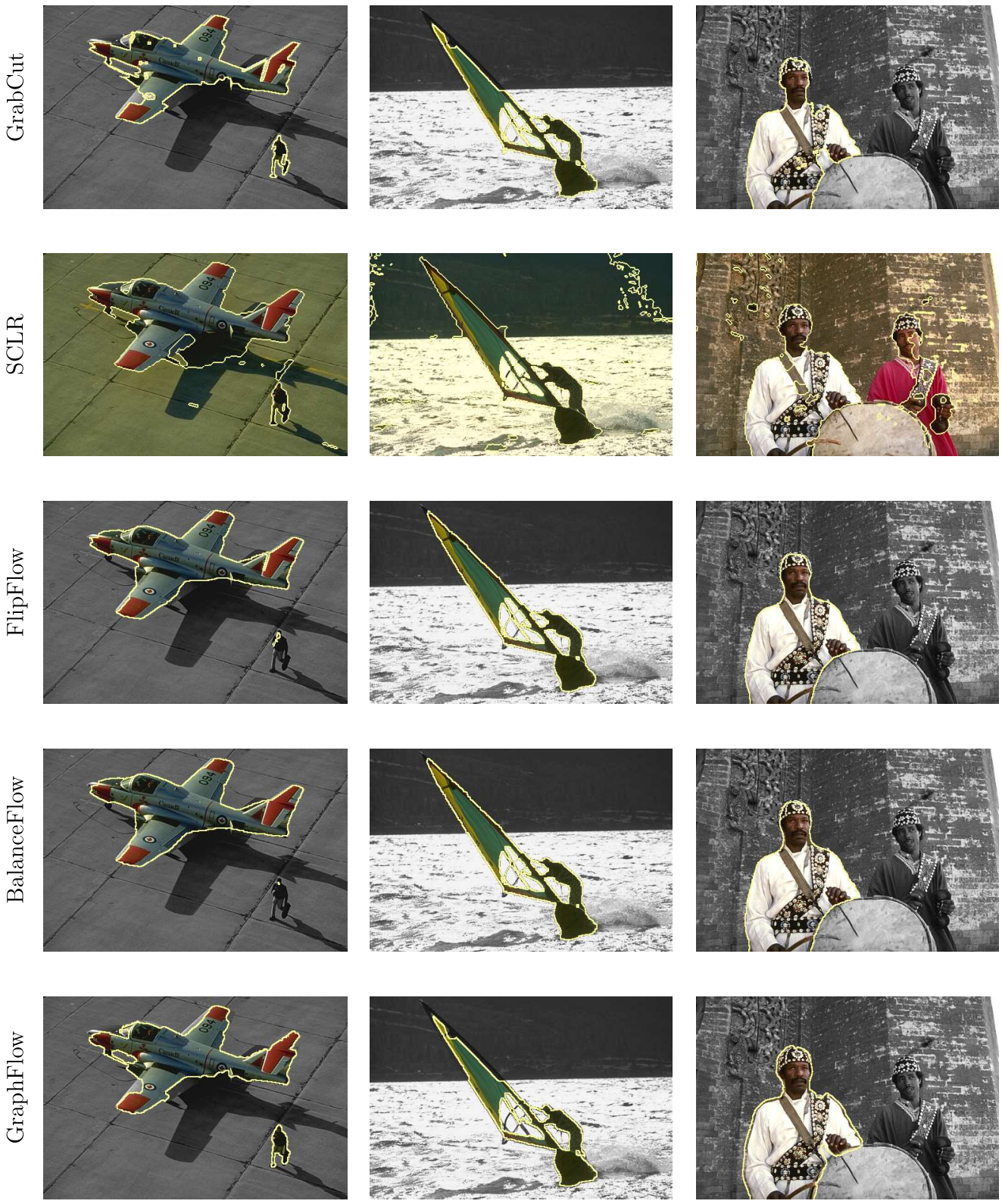


Figure 5.13: Comparison 1

Figure 5.14: Comparison 2

Figure 5.15: Comparison 3

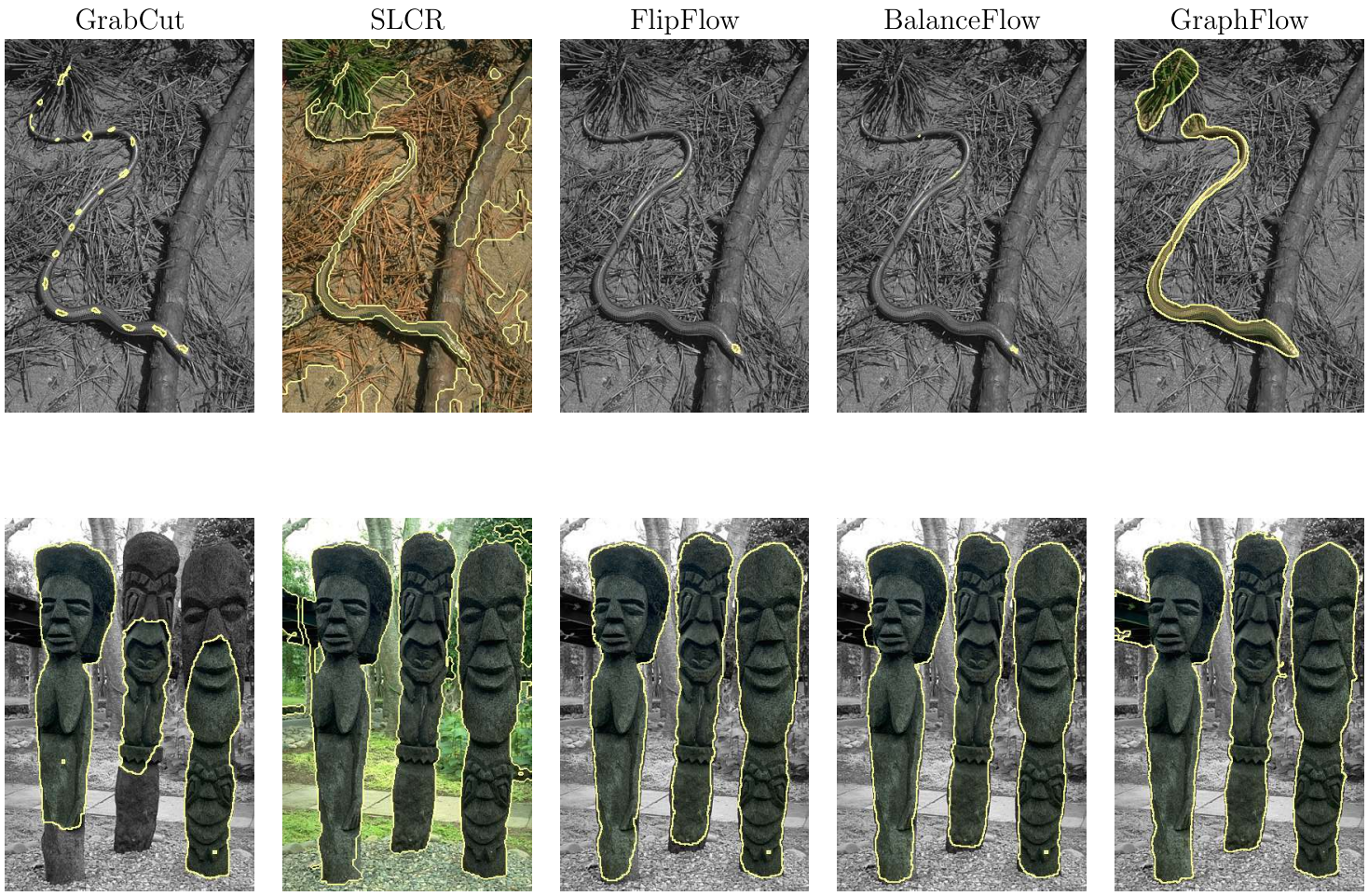


Figure 5.16: Comparison 4