An Integral Solution to Surface Evolution PDEs via Geo-Cuts

Yuri Boykov[1], Vladimir Kolmogorov[2], Daniel Cremers[3], and Andrew Delong[1]

[1] University of Western Ontario, Canada, {yuri,adelong3}@csd.uwo.ca
[2] University College London, UK, vnk@adastral.ucl.ac.uk
[3] University of Bonn, Germany, cremers@cs.ucla.edu

Abstract. We introduce a new approach to modelling gradient flows of contours and surfaces. While standard variational methods (e.g. level sets) compute local interface motion in a *differential* fashion by estimating local contour velocity via energy derivatives, we propose to solve surface evolution PDEs by explicitly estimating *integral* motion of the whole surface. We formulate an optimization problem directly based on an integral characterization of gradient flow as an infinitesimal move of the (whole) surface giving the largest energy decrease among all moves of equal size. We show that this problem can be efficiently solved using recent advances in algorithms for global hypersurface optimization [4, 2, 11]. In particular, we employ the *geo-cuts* method [4] that uses ideas from integral geometry to represent continuous surfaces as cuts on discrete graphs. The resulting interface evolution algorithm is validated on some 2D and 3D examples similar to typical demonstrations of level-set methods. Our method can compute gradient flows of hypersurfaces with respect to a fairly general class of continuous functionals and it is flexible with respect to distance metrics on the space of contours/surfaces. Preliminary tests for standard L_2 distance metric demonstrate numerical stability, topological changes and an absence of any oscillatory motion.

1 Introduction

As detailed in [4, 11, 12], discrete minimum cut/maximum flow algorithms on graphs can be effectively used for optimization of a fairly wide class of functionals defined on contours and surfaces in continuous metric spaces. So far, graph based methods were presented as a global optimization alternative to local variational optimization methods such as the level set method. Efficient algorithms for finding global optima have a number of advantages over local optimization methods. However, in some cases it is necessary to observe gradual changes of a contour/surface that they display under gradient flow (or gradient descent). For example, gradient flow models dynamics of many natural phenomena in physics. In computer vision, ability to track gradual changes in segmentation allowed variational methods to successfully incorporate shape priors [8, 14].

In this paper we propose a new integral approach to computing gradient flow of interfaces with respect to a fairly general class of energy functionals. The proposed algorithm generates a timely sequence of cuts corresponding to gradient flow of a given contour. Note that the proposed method is not a new implementation of level set methods but rather an alternative numerical method for evolving interfaces. Our method does not use any level set function to represent contours/surfaces. Instead, it uses an implicit contour/surface representation via geo-cuts [4]. As the level set method, our approach handles topological changes of the evolving interface.

2 Variational Methods and PDEs in Computer Vision

Numerous computer vision problems can be addressed by variational methods. Based on certain assumptions regarding the image formation process, one formulates an appropriate cost functional which is subsequently minimized by implementing the Euler-Lagrange equations in a gradient flow partial differential equation (PDE). This technique has become standard in various fields of computer vision ranging from motion estimation [9, 3, 17, 13], over image enhancement [15, 16] to segmentation [10, 6]. While not all PDEs are derived from a variational approach, in this work we will focus on the class of PDEs which correspond to the gradient flow to an underlying variational principle.

Despite their enormous success in the local optimization of a large class of cost functionals, PDEs suffer from certain drawbacks. In particular, they are inherently differential approaches, they rely on the notion of an energy gradient which – in many cases — requires intense numerical computations. The numerical discretization of PDEs requires a careful choice of appropriate time step sizes. Extensive research went into determining conditions which guarantee stability of the respective implementations. In practice, meaningful time step sizes are often chosen based on various heuristics.

In contrast to this differential approach, we develop in this paper an integral approach to solving a certain class of gradient flow PDEs. To this end we revert to efficient combinatorial optimization methods acting on a discrete space. In a number or recent papers [4, 2, 11], it was shown that various optimization problems defined on surfaces in continuous spaces can be efficiently solved by discrete combinatorial optimization methods. In contrast to these works, the present paper is not focussed on determining the global optima of respective cost functions, but rather on actually modelling the local gradient descent evolution of the corresponding variational approaches. In this sense, we hope to further bridge the gap between continuous variational approaches and discrete combinatorial approaches to optimization.

3 From Differential to Integral Approach

Our main goal is an algorithm for computing gradient flows for hypersurfaces based on novel optimization techniques [4, 2, 11] which are fundamentally different from standard variational methodology. Gradient flow for contours and surfaces amounts to evolving an initial boundary under Euler-Lagrange equation of a given energy functional. Such propagation of surfaces corresponds to many natural phenomena and it can be derived from the laws of physics. Standard variational calculus justifies the corresponding PDE in the context of (local) energy optimization and provides numerical methods (including level-sets) for solving it directly via finite difference or finite element schemes. In contrast, our new approach solves the corresponding PDE indirectly.

Our approach to gradient flow was motivated by the numerical stability of global optimization methods in [4, 2, 11]. In this paper we show how to turn them into robust surface evolution methods that may overcome some of the numerical limitations of standard variational techniques. Note that variational methods rely on estimates of energy derivatives in order to compute each point's local *differential* motion (velocity). In contrast, our main idea is to compute *integral* motion of the surface as a whole. In particular, geo-cuts [4] allow to compute such motion by means of integral geometry without estimating derivatives.

When trying to model surface evolutions by global optimization approaches, we are faced with the following discrepancy between local and global optimization methods: while existing global optimization methods [4, 2, 11] are merely focussed on finding the boundary with the lowest energy, the gradient approaches make use of the energy *gradient*, i.e. they focus on the maximal energy reduction per change in the boundary. Therefore, any algorithm for computing gradient flow needs to have means to incorporate a measure of the boundary change.

3.1 Distances Between Contours

There are numerous metrics to measure change between boundaries. In fact, the question of which metric on the space of contours should be used has been largely ignored in the context of calculus of variation. Most so-called gradient descent evolutions implicitly assume an L_2 inner product. Several recent advances were made regarding the derivation of Euler-Lagrange equations with respect to more sophisticated contour metrics, focussed either on using the correct metric [19], or on designing novel gradient flows with certain desirable properties [7]. A very similar freedom in the choice of metric on the space of contours will also arise in our novel integral formulation of boundary evolution.

Note that motion of a contour C in a differential framework is described by a (normal) vector field $v = \frac{dC}{dt}$ where velocity vector v_s is given for every contour point s. Then, standard L_2 measure for boundary change is defined as $||\frac{dC}{dt}||^2 = \int_C |v_s|^2 ds = \langle \frac{dC}{dt}, \frac{dC}{dt} \rangle$ using Euclidean inner product $\langle , \rangle$. As mentioned above, employing other inner products on the space of contours will entail different kinds of gradient flows of a contour.

In order to avoid local differential computations, we will represent the motion of a contour C by integral measures of boundary change: a distance metric on the space of contours that for any two contours C and C_0 assigns a (nonnegative) distance value $dist(C, C_0)$. Such distance metric could be consistent with ideas for measuring boundary change in the differential framework if for $C \to C_0$

$$dist(C, C_0) = \langle dC, dC \rangle + o(||dC||^2) \tag{1}$$

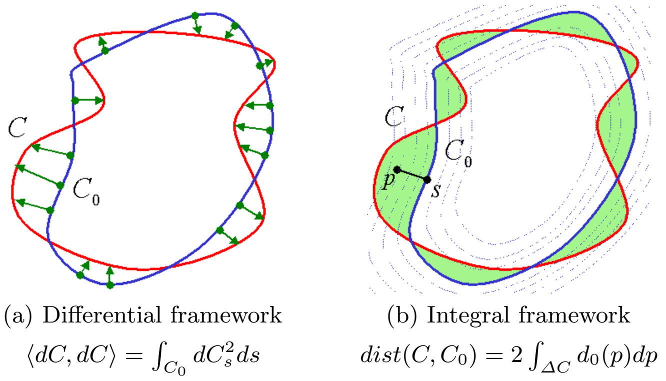

C
C_0
(a) Differential framework
$\langle dC, dC\rangle = \int_{C_0} dC_s^2 ds$
C
C_0
p
s
(b) Integral framework
$dist(C, C_0) = 2\int_{\Delta C} d_0(p)dp$

Fig. 1. **L_2 distance between two near-by contours. In the integral framework $dist(C, C_0)$ is equal to a weighted area of the highlighted region. The weight of each point p is given by a distance $d_0(p)$ to the nearest point on C_0.**

where $dC = C - C_0$ is a field of (normal) vectors defined on points in C_0 and connecting them with points on C as shown in Figure 1(a). In other words: The integral distance metric is consistent with the differential approach if the two metrics are identical up to higher order terms.

Example 1. (L_2 *metrics*) Figure 1 illustrates relationship between differential and integral approaches to measuring boundary change between two contours for the most standard case of L_2 inner product $\langle,\rangle$. The corresponding distance metric on the space of contours in $\mathcal{R}^2$ is

$$dist(C, C_0) = 2 \cdot \int_{\Delta C} d_0(p)dp$$

where p are points in $\mathcal{R}^2$, function $d_0(p)$ is a distance from p to the nearest point on C_0 (*distance map*), and ΔC is a region between two contours. Then, (1) holds because integrating the distance function $2d_0(p)$ along a (normal) direction connecting some point $s \in C_0$ and a point $q \in C$ gives $||q - s||^2 = dC_s^2$. ∎

Our general integral approach to front propagation is described in the next subsection. The method is well defined for any distance metric on the space of contours. However, if one wants to model the gradient flow corresponding to a differential formulation with a given inner product $\langle,\rangle$ then the consistent distance metric (1) should be used.

3.2 Integral Formulation of Gradient Flow

Our method for solving PDEs for contours or surfaces evolution is motivated as follows. Gradient flow (descent) of an interface C under any given energy functional $F(C)$ can be intuitively viewed as a temporal sequence of infinitesimal steps where each step gives the largest decrease of the contour energy among all steps of the same size. This almost banal interpretation of the gradient descent suggests that the contour C_{t+dt} corresponding to an infinitesimal step in the gradient descent from a contour C_t can be obtained by solving the following constrained optimization problem:

$$\min_{C\,:\,dist(C,C_t)=\epsilon} F(C)$$

for some (arbitrarily) small value $\epsilon > 0$ fixing the distance $dist(C, C_t)$ from the contour C_t. Equivalently, the method of Lagrangian multipliers shows that C_{t+dt} should also solve unconstrained optimization problem

$$\min_{C} \quad F(C) + \lambda \cdot dist(C, C_t)$$

for some (arbitrarily) large value of parameter λ. These formulations for C_{t+dt} do not establish an explicit relationship between the temporal step size dt and the values of ϵ or λ. We just know that for each small dt there is some corresponding small $\epsilon = \epsilon(dt)$ or some corresponding large $\lambda = \lambda(dt)$.

In fact, it is not difficult to establish an exact relationship between λ and dt using a well known PDE for evolution of an interface C under a gradient flow (descent) with respect to a given energy functional $F(C)$. Our general approach to computing gradient flows will be based on optimization of energy (2).

Theorem 1. *Consider a family of contours C_t minimizing energy*

$$E_t(C) = F(C) + \frac{1}{2(t-t_0)} \cdot dist(C, C_0) \tag{2}$$

where metric $dist(C, C_0)$ is consistent with some inner product $\langle , \rangle$ according to equation (1). Then, as $t \to t_0$

$$C_t = C_0 + v \cdot (t - t_0) + o(\Delta t)$$

where vector field $v = -\frac{dF}{dC}$ is a gradient of F with respect to inner product $\langle , \rangle$. That is, as $t \to t_0$ the contour C_t solves the standard gradient flow PDE

$$\frac{\partial C}{\partial t} = -\frac{dF}{dC} \tag{3}$$

Proof. Since $E_t(C) = F(C) + \frac{1}{2(t-t_0)}\langle C - C_0, C - C_0\rangle$ then any contour C optimizing E_t should satisfy

$$0 = \frac{dE_t}{dC} = \frac{dF}{dC} + \frac{1}{(t-t_0)} \cdot (C - C_0).$$

Thus, optimality of C_t for E_t implies $C_t - C_0 = -(t - t_0) \cdot \frac{dF}{dC}$ which is equivalent to the standard gradient flow equation (3) as $t \to t_0$. ∎

Standard variational (differential) methods for computing contour evolution under gradient flow, including level sets, explicitly estimate the derivative $\frac{dF}{dC}$ and use finite differences or finite elements to approximate the PDE (3). Theorem 1 suggests an alternative integral approach to computing gradient flows. Assuming that C_0 is a current state of the contour, we can obtain an optimal contour C_t minimizing (2) for some small time step $\varDelta t = (t - t_0)$. The gradient flow problem is solved by a sequence of optimal steps $\{C_0 \to C_t\}$ where at each new iteration C_0 is reset to the optimal contour computed in the previous step.

Optimization of E_t may look like a difficult task. However, our integral approach is practical because a wide class of continuous functionals $F(C)$ and metrics $dist(C, C_0)$ in energy (2) can be efficiently optimized by recent global methods [4, 2, 11]. In particular, Section 4 describes details of a discrete approximation algorithm for gradient flows based on geo-cuts [4]. This algorithm is based on implicit representation of continuous contours as cuts on discrete graphs. Optimization of continuous contour/surface energy (2) via geo-cuts avoids explicit differentiation of E_t and relies on efficient combinatorial algorithms.

3.3 Discussion and Relation to Previous Approaches

Note that energy E_t in (2) can be globally minimized using geo-cuts [4] when the first term, hypersurface functional $F(C)$, includes anisotropic Riemannian length/area and any regional bias. It is important, however, that energy (2) contains another term $dist(C, C_0)$. This second term is critical for implementing gradient flow instead of global minimization of functional $F(C)$. Note that $dist(C, C_0)$ enforces shape stabilization and slows down the contour generating gradual motion instead of a jump into a global minima. As shown in Example 1 from Section 3.1, standard gradient flow with respect to L_2 inner product corresponds to shape constraint $dist(C, C_0)$ penalizing deviation of C from C_0 according to the area between the contours weighted by the distance from C_0. In fact, this is a simple regional bias that can be easily incorporated into geo-cuts. Additional details are given in Section 4.

Our approach can also compute gradient flow with respect to inner products different from L_2. One has to determine the distance metric $dist(C, C_0)$ consistent with the corresponding inner product as in equation (1).

Generally speaking, one can use our general framework for propagating hypersurfaces by optimizing energy (2) with an arbitrary distance metric $dist(C, C_0)$. In this case the method may not really correspond to any true gradient flow but it may still generate some gradual motion of an interface. That may be sufficient for many practical applications in computer vision that do not need exact gradient flow. The results in [19, 7] suggest that using specialized distance metrics could be beneficial in applications.

Interestingly, some existing discrete algorithms for active contours are special cases of our general approach for some specific distance metric $dist(C, C_0)$.

Example 2. (*DP snakes*) A well-known dynamic programming approach to 2D snakes [1] uses control points to represent discrete snake C. Then, a typical snake energy functional $F(C)$ is iteratively minimized by dynamic programming over positions of control points. In order to simulate gradient-descent-like motion, the algorithm in [1] allows each point to move only in a small box around their current position. Intuitively speaking, this idea does capture the spirit of gradient flow motion. However, as easily follows from our theories in Section 3.2, the motion generated in [1] corresponds to energy (2) with a $0-1$ boxy distance metric on a space of snakes $dist(C, C_0) = \prod_p \delta_{|dC_p|>\epsilon}$ where ϵ is a given box size and $dC_p = C(p) - C_0(p)$ is a shift of snake's control point p. This distance metric is not consistent with any inner product (bilinear form). Therefore, the corresponding algorithm does not generate a true gradient flow[4]. At the same time, our theories suggest a simple correction to the problem; In order to get a true L_2 gradient flow, DP-snakes algorithm [1] could amend box-based move constraints with a quadratic motion penalty $dist(C, C_0) = \sum_p dC_p^2$ which can be easily handled by dynamic programming.

Example 3. (*Fixed Band Graph-Cuts*) An approach to segmentation in [18] is very similar to DP-snakes algorithm in [1]. Dynamic programming in DP-snakes is replaced by graph cuts but [18] still uses the idea of a fixed size "box". Since graph cuts do not use control points to represent contours and instead rely on implicit binary graph-partitioning representation, the boxes around control points are replaces by a small band around a current cut. Otherwise, the active contour algorithm presented in [18] can be described through energy (2) with the same $0-1$ boxy distance metric $dist(C, C_0) = \delta_{|dC|_h>\epsilon}$ where $|dC|_h$ is the Hausdorff distance between C and C_0. It follows that the method in [18] does not correspond to gradient flow for any reasonable inner product. In fact, fixed-band approach in [18] is likely to generate a jerky non-smooth motion. In contrast, our theoretical framework allows a principled approach to contour evolution via graph cuts. Using proper distance $dist(C, C_0)$ consistent with some (continuous) inner product $\langle , \rangle$ allows to control geometric artifacts that may arise in front propagation using discrete optimization techniques.

4 Computing Gradient Flow via Geo-Cuts

As discussed before, there are a number of algorithms that can (globally) minimize continuous functional (2) and practically implement our approach to solving gradient flow PDEs. This paper concentrates on a solution based on geo-cuts [4].

4.1 Review of Geo-Cuts

Geo-cuts is a graph based approach to minimizing continuous functionals $E(C)$ based on representing contours as cuts on a discrete graph (Fig. 2). Nodes in

[4] A standard DP snake [1] under Euclidean length functional $F(C)$ will not generate a true mean curvature motion. In particular, this snake may not converge to a circle before collapsing into a point.

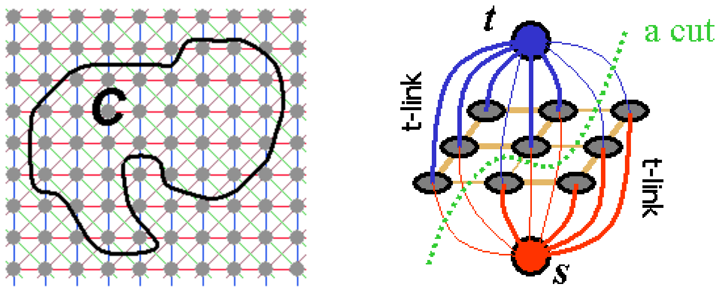


Fig. 2. Geo-cuts: Any continuous contour C corresponds to a cut on a graph. Edge weights define discrete *cut metric* assigning length to C based on the cost of the corresponding cut. With appropriately chosen weights, cut metric approximates any continuous anisotropic Riemannian metric.

this graph correspond to sampled points in space. A cut is a binary labeling of these nodes. In the context of continuous contour representation, binary labels $\{1, 0\}$ say if the point (node) is inside or outside of the contour. Note that this implicit representation of continuous contours does not say precisely where the boundary is between the two neighboring graph nodes (points in space) with different labels. In fact, the lack of sub-pixel accuracy does not cause problems for geo-cuts because they do not use estimates of local gradients/derivatives, e.g. curvature, and rely on methods of integral geometry instead.

There are also two special nodes, *terminals* s and t (source and sink). Graph edges are *n-links* and *t-links*, as in Fig. 2. Typically, n-links encode regularization term in the energy (length or area) while t-links encode regional bias.

The first step in the geo-cuts approach is to construct a graph whose cut metric approximates that of functional $E(C)$. Such construction exists for a fairly large class of continuous functionals E, as shown in [4, 12]. In general, $E(C)$ can be any submodular (graph-representable) functional over contours/surfaces. In particular, it can include the following terms:

– Geometric length/area under a fairly wide class of continuous metrics (including any anisotropic Riemannian metric);
– Flux with respect to any continuous vector field;
– Regional term integrating arbitrary potential function over the interior of C.

After constructing an appropriate graph, the cut with the smallest cost can be computed very efficiently via min cut/max flow algorithms.

We now apply this framework to the problem of computing gradient flow at time t given current contour C_0. In order to minimize functional $E_t(C)$ in eq. (2), we need to approximate terms $F(C)$ and $dist(C, C_0)$ with a discrete cut metric. According to the characterization above, the first term $F(C)$ can be any submodular functional over contours/surfaces. This covers a widely used special case when F is a geometric length/area in a Riemannian metric induced by the image. Let us consider the second term.

As discussed in section 3.1, we have some freedom in choosing function $dist(C, C_0)$. There are many distance measures corresponding to different inner products $\langle , \rangle$. For example, in order to incorporate standard L_2 inner product we can use the distance function described in example 1. It can be rewritten as

$$dist(C, C_0) = -2 \cdot \int_{int(C_0)} d_0(p)dp + 2 \cdot \int_{int(C)} d_0(p)dp \tag{4}$$

where $int(C)$ is the interior of C and $d_0(p)$ is now the *signed* distance map; it is negative inside C_0 and positive outside. The first term above is a constant independent of C. The second term is a regional bias that can be incorporated into geo-cuts using t-links. Note that we can use non-Euclidean signed distance maps to implement metrics on the space of contours different from L_2.

4.2 Minimizing Energy E_t with Geo-Cuts

Our approach to gradient flows amounts to finding small moves $C(t)$ from $C(0) = C_0$ for $(t - t_0) < \delta t$ and then resetting time and energy (2) for $C'_0 = C(t)$. Section 4.3 describes this *move-reset* algorithm. In this section, however, we assume that C_0 is fixed and discuss properties of a timely sequence of cuts $\{C(t)|t \geq 0\}$ where each particular cut $C(t)$ is a global minima of $E_t(C)$ for a given t. For simplicity of notation we will assume that $t_0 = 0$.

It can be shown that the time axis can be split into a finite number of intervals $[t_i, t_{i+1}]$ such that there is cut C_i which is optimal for all $t \in [t_i, t_{i+1}]$. Our goal is therefore to find a sequence of critical time instances $t_1, t_2, t_3, \ldots, t_n$ when an optimal cut changes. We will also need to find the corresponding sequence of optimal cuts $C_1, C_2, C_3, \ldots, C_n$. Note that the initial contour C_0 will be an optimal cut for any $t \in [0, t_1]$. Also, "final cut" C_n is a global minimizer of functional $F(C)$. It may happen that C_0 is already a global minimum, in which case $n = 0$. Below we list a number of useful facts about this sequence of cuts.

Remark 1. It is possible to prove that

$$F(C_0) > F(C_1) > F(C_2) > ... > F(C_n)$$

Therefore, the energy will never increase during the algorithm. This will prevent any oscillatory behaviour present in some implementations of level sets.

Remark 2. Similar to the continuous case, the gradient flow (gradient descent) algorithm above is related to the following constrained optimization problem (where C is now a discrete cut and $F(\cdot)$ encodes cut metric):

$$\min_{C \,:\, dist(C_0, C) = \epsilon} F(C) \tag{5}$$

More precisely, an optimal solution C_t for energy (2) at any given time $t > 0$ solves the constrained minimization problem above for

$$\epsilon_t = dist(C_0, C) \tag{6}$$

Indeed, suppose that there is some other cut C such that $dist(C_0, C) = \epsilon_t$ and $F(C) < F(C_t)$. Then, $F(C) + \frac{1}{2t} dist(C_0, C) < F(C_t) + \frac{1}{2t} dist(C_0, C)$ and we have a contradiction to the fact that C_t is optimal for energy (2).

Equation (6) explicitly determines the "size" of each gradient descent step $\epsilon_i = dist(C_0, C_i)$ generated by our algorithm. It is easy to show that

$$0 < \epsilon_1 < \epsilon_2 < \ldots < \epsilon_n < \infty$$

Remark 3. An interesting observation is that C_1 (the first cut different from initial solution C_0) is a solution for

$$\min_{C \neq C_0} \frac{F(C) - F(C_0)}{dist(C_0, C)}$$

The proof is based on the fact that at time t_1 energy (2) has two distinct optimal cuts C_1 and C_0. Thus, $F(C_0) = F(C_1) + \frac{1}{2t_1} dist(C_0, C_1)$ and the proof follows from a standard "binary search" algorithm for ratio optimization. It also follows that the corresponding optimal value of the ratio is equal to $-\frac{1}{2t_1}$. Note that the optimal (minimal) ratio value has to be negative (non-positive). Indeed, unless C_0 is a global minimizer of $F(\cdot)$ we have at least one cut (e.g. C_1) where the value of the ratio is negative (since $F(C_1) < F(C_0)$). Note that optimization of the ratio above is meaningful in discrete formulation only. It can be shown that in the continuous case the ratio above converges to $-\infty$ as $C \to C_0$.

Remark 4. The first cut C_1 is the most accurate gradient descent step from C_0 as it corresponds to the smallest step size ϵ_1. Ideally, we want to compute the optimal solution of (5) for the smallest value of ϵ while ϵ_1 is the smallest step size where our graph cut algorithm can detect an optimal move. The size of that smallest step ϵ_1 is possibly due to approximation errors in our discrete graph-cut formulation and may depend on graph "resolution".

4.3 Summary of Gradient Flow Algorithm

It follows that the gradient flow is approximated the best when we reset to initial cut $C'_0 = C_1$ and update the energy (2) after each small move C_1. In practice we may not need to be so conservative but that needs to be checked experimentally.

It is possible to show that

$$\epsilon_1 \approx (2t_1 \cdot ||\nabla F||)^2 = (2t_1 \cdot ||\frac{dF}{dC}(C_0)||)^2 \qquad (7)$$

Indeed, using remark 3 and expression $\epsilon_1 = dist(C_0, C_1) \approx ||C_1 - C_0||^2$ we get

$$-\frac{1}{2t_1} = \frac{F(C_1) - F(C_0)}{\epsilon_1} \approx \frac{\langle \nabla F, C_1 - C_0 \rangle}{\epsilon_1} \approx -\frac{||\nabla F|| \cdot ||C_1 - C_0||}{\epsilon_1} \approx -\frac{||\nabla F||}{\sqrt{\epsilon_1}}$$

Equation (7) allows to determine a stopping criteria for our algorithm when we converged to a local minima where $\nabla F = 0$. In practice we may stop if the gradient of F is smaller than some predefined threshold, $||\nabla F|| < \delta$, which corresponds to the stopping condition

$$dist(C_0, C_1) < (2t_1 \cdot \delta)^2$$

5 Experimental Validation

Although the focus of our paper is mainly theoretical, we have generated preliminary results to show that gradient flow can indeed be approximated with an integral representation of length and hypersurfaces. The image sequences that follow were generated by combining the geo-cuts method for computing $F(C)$ and a signed distance map for computing $dist(C, C_0)$ as in (4). The distance map is computed in such a way that for pixels p at the boundary of cut C_0 there holds $d_0(p) = \pm 0.5$. (Pixel p is said to be at the boundary if it is 4-connected to some pixel q in the other segment). In our tests we compute the first cut C_1 and reset $t'_0 = t_1, C'_0 = C_1$. We use an implementation of maxflow graph-cuts [5] as a tool for optimizing the energy. Note that in the figures below we show only selected time frames of the gradient flow motion computed by out method.

In Figures 3, 6 and 7 we have intentionally used low-resolution grids to illustrate that even extremely coarse integral representations can yield accurate gradient flow motion, despite a lack of sub-pixel accuracy. Our test results on length/area minimization with a Euclidian metric demonstrate that our algorithm *first* converges to a circle/sphere and *then* converges about the center to a point–exactly the progression expected of a correct gradient flow simulation, and is a critical test of any such algorithm.

A plot in Figure 4 presents empirical evidence confirming that gradient flow generated by our method has accurate temporal dynamics. This plot shows the radius of a circle evolving under (Euclidean) curvature flow computed by our method. Our algorithm directly generates time Δt for each step allowing us to show actual temporal dynamics of the flow. The plot demonstrates accurate temporal behaviour of the moving circle consistent with the theory. The same plot also demonstrates that our algorithm can compute gradient flow with a high temporal resolution so that the generated motion is very gradual.

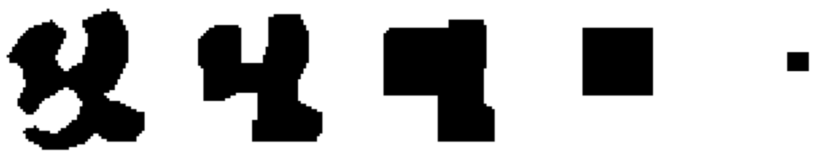
Figure 5 provides additional evidence of our method's accuracy. We compute (Euclidean) curvature flow of a "sausage" which gradually moves from the ends where the curvature is non-zero while the straight sides do not move until the sausage turns into a circle. This result would be impossible to obtain with a DP-snake [1] or fixed-band graph cuts [18]. For example, each step of the algorithm in [18] will uniformly erode the "sausage" from all sides. The "sausage" will collapse into a line interval (not into a point) in jumpy moves of equal size (band width).

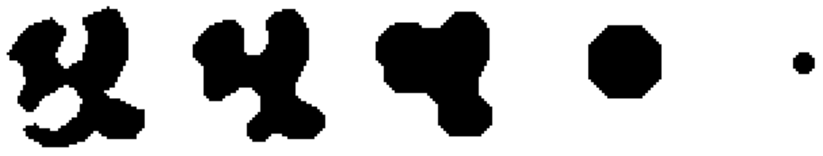
Our tests with image-based Riemannian metrics (e.g. see Figure 6) have confirmed that topological changes in contours occur in a similar manner to level-set methods, and that contours do not exhibit oscillatory motion but instead remain fixed at local minima.

As of this writing, we have yet to experiment with more justified ways of both controlling the time steps and the manner in which the distance map(s) are used. One potential source of inaccuracy lies in the fact that the distance map can be determined only with precision 0.5, since we use discrete representation of contours via geo-cuts. The influence of this effect is most significant near the boundary of contour C_0. This suggests that using the first cut C_1 is not necessarily the most accurate method. The problem, however, may be solved by using cuts C_k for bigger time step $t_k > t_1$. This idea can be combined with supersampling the grid graph. Despite this potential difficulty, the experiments indicate that even our preliminary implementation gives very encouraging results, which show that geo-cuts approach may provide a numerically stable method for solving gradient flow PDEs.

References

1. Amir A. Amini, Terry E. Weymouth, and Ramesh C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, September 1990.
2. Ben Appleton and Hugues Talbot. Globally minimal surfaces by continuous maximal flows. *IEEE transactions on Pattern Analysis and Pattern Recognition (PAMI)*, 28(1):106–118, January 2006.
3. M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise–smooth flow fields. *cvgip-iu*, 63(1):75–104, 1996.
4. Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Int. Conf. on Computer Vision*, volume I, pages 26–33, 2003.
5. Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.
6. T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Trans. Image Processing*, 10(2):266–277, 2001.
7. G. Charpiat, O. Faugeras, and R. Keriven. Approximations of shape metrics and application to shape warping and empirical shape statistics. *Journal of Foundations of Computational Mathematics*, 5(1):1–58, 2005.
8. D. Cremers, F. Tischhäuser, J. Weickert, and C. Schnörr. Diffusion Snakes: Introducing statistical shape knowledge into the Mumford–Shah functional. *IJCV*, 50(3):295–313, 2002.
9. B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
10. M. Kass, A. Witkin, and D. Terzolpoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
11. D. Kirsanov and S.-J. Gortler. A discrete global minimization algorithm for continuous variational problems. *Harvard CS. Tech. Rep.*, TR-14-04, July 2004.
12. V. Kolmogorov and Y. Boykov. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *ICCV*, October 2005.
13. P. Kornprobst, R. Deriche, and G. Aubert. Image sequence analysis via partial differential equations. *Journal of Math. Imaging and Vision*, 11(1):5–26, 1999.
14. Nikos Paragios. Shape-based segmentation and tracking in cardiac image analysis. *IEEE Transactions on Medical Image Analysis*, pages 402–407, 2003.
15. P. Perona and J. Malik. Scale-space and edge-detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.
16. J. Weickert. *Anisotropic diffusion in image processing.* Teubner, Stuttgart, 1998.
17. J. Weickert and C. Schnörr. A theoretical framework for convex regularizers in PDE–based computation of image motion. *IJCV*, 45(3):245–264, 2001.
18. N. Xu, R. Bansal, and N. Ahuja. Object segmentation using graph cuts based active contours. In *CVPR*, volume II, pages 46–53, 2003.
19. Anthony Yezzi and Andrea Mennucci. Conformal metrics and true “gradient flows” for curves. In *IEEE Intl. Conf. on Comp. Vis.*, 2005.


(a) Curvature flow in “Manhattan” L_1 metric (4-neighborhood)


(b) Curvature flow in “Octagonal” metric (8-neighborhood)

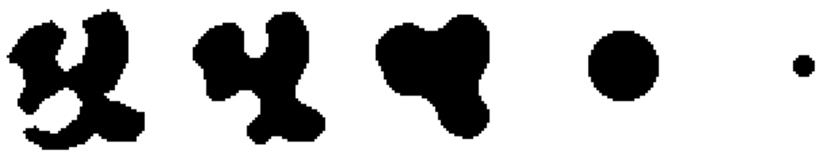

(c) Curvature flow in Euclidean (L_2) metric (16-neighborhood)

Fig. 3. Length minimizing (curvature) flow of a 2D contour under different homogeneous metrics. Pixalization reflects the actual resolution used in the experiments and demonstrates that our discrete geo-cuts representation of contours generates accurate gradient flow without explicitly tracking the contour with sub-pixel accuracy as in level-sets.

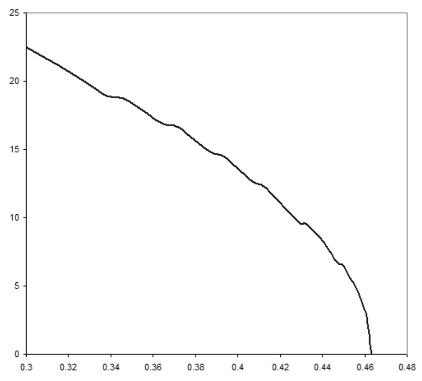


Fig. 4. Empirical plot of a radius of a circle under curvature flow. Theoretically, this function is $r(t) = \sqrt{const - 2t}$.

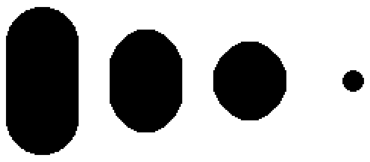


Fig. 5. Euclidean length minimizing flow of a 2D "sausage" (16-neighb.) Note that the straight sides have zero curvature and they do not move until the top and the bottom sides (with positive curvature) collapse the "sausage" to a circle.

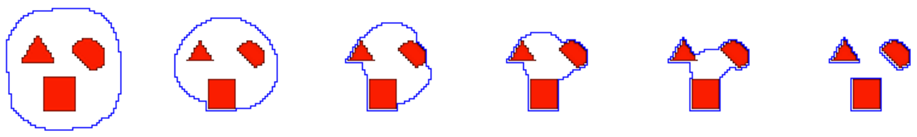
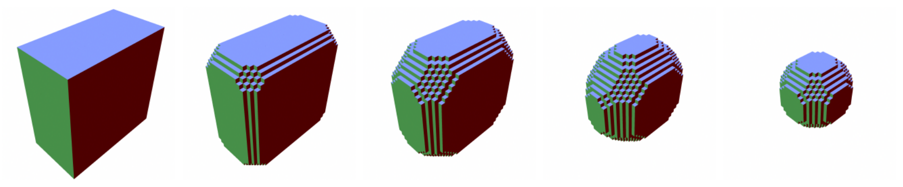
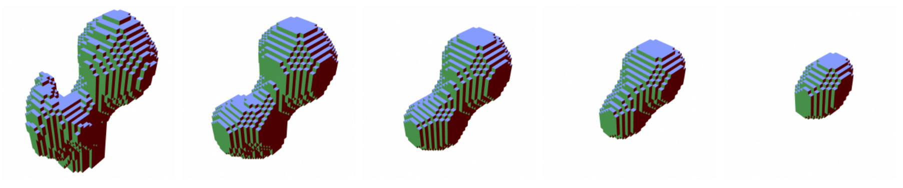


Fig. 6. Length minimizing flow of a 2D contour (blue) under image-based anisotropic Riemannian metric (16-neighborhood)



(a) Gradient flow for a cube (26-neighborhood)



(b) Gradient flow for a blob (26-neighborhood)

Fig. 7. Euclidean area minimizing flow of a surface in 3D. Voxalization reflects the actual resolution.