A Geometric Flow Approach for Region-based Image Segmentation

Juntao Ye and Guoliang Xu

Abstract—Geometric flows have been successfully used for surface modeling and designing, largely because they are inherently good at controlling geometric shape evolution. Variational image segmentation approaches, on the other hand, detect objects of interest by deforming certain given shapes. This motivates us to revisit the minimal partition problem for segmentation of images, and propose a new geometric flow based formulation and solution to it. Our model intends to derive a mapping that will evolve given contours or piecewise-constant regions towards objects in the image. The mapping is approximated by B-spline basis functions and the positions of the control points are to be determined. Starting with the energy functional based on intensity averaging, we derive an Euler-Lagrange equation and then a geometric evolution equation. The linearized system of equations is efficiently solved via a special matrix-vector multiplication technique. Furthermore, we extend the piecewise-constant model to a piecewise-smooth model which effectively handles images with intensity inhomogeneity.

Index Terms—Image segmentation, geometric flow, partial differential equations, minimization, variational method.

I. INTRODUCTION

VARIATIONAL approaches have been proven to be effective for segmentation and detection of objects from an image. These methods share a common feature that they define an energy functional minimizer which generally depends on the given image and the characteristics used to identify different segmented regions. The Euler-Lagrange equation of these models can often be described using a partial differential equation (PDE), which is iterated until it reaches a steady state. This way, the shapes of given regions or contours are deformed towards objects of interest in an image.

Among those variational approaches, the most successful family are the level set based methods. Level set methods formulate the problem of manifold deformation by embedding it into one higher dimensional space (e.g. building a curve into a surface thus evolving the curve turns into moving the surface). Yet the added cost of increased dimension is usually worth doing, because parameterizing the boundary of the shape and following its evolution, which is also considered to be a hard part of the problem, is no longer needed. The level set function expands, splits, merges, and does all the work. If, there exists a technique that does not increase the dimension of the problem, and at the same time is capable of parameterizing objects of interest without much pain, would it be a compelling alternative to the level set methods for image segmentation? Geometric partial differential equations (a.k.a. geometric flows) is a technique capable of parameterizing and modeling time-varying objects. In fields such as computer-aided geometric design, it has been successfully used for surface modeling and shape designing in the past few years (see Section I-C for a brief review). We are curious to know if image segmentation could also benefit from the recent advances of this technique. This motivated us to revisit the minimal partition problem of Mumford and Shah [24], and proposes a new geometric flow based approach for segmenting gray-scale images.

Our model, by no exception, minimize the Mumford-Shah functional and segment an image into a number of piecewise-constant regions. Yet it is essentially different from popular level-set based implementations for this problem. We intend to derive a mapping that deforms given contours or regions towards objects in the image. The mapping is constructed as a B-spline approximation, and the positions of the control points are to be determined. The energy functional is defined on intensity averaging, and the Euler-Lagrange equation and then a geometric evolution equation are derived. The discretization of the equation involves a finite element scheme in the spatial domain and a forward Euler scheme in the temporal domain. Solving the linearized system of equations is accomplished by a tricky matrix-vector multiplication technique. This model is not limited to piecewise-constant partitioning. With minus modification, it can be extended to a piecewise-smooth model so that images with intensity inhomogeneity can be handled.

The outline of the paper is as follows: we first give the background on the Mumford-Shah problem and related work on segmentation, and the overview of geometric flow methods. In Section II, we present our reformulation of the problem and the derived geometric flow equation. Section III details the numerical solution to the equation. Section IV describes some implementation details and experimental results. In Section V the extension to the piecewise-smooth model is introduced. Discussions and the conclusion are drawn in the end.
A. The Mumford and Shah problem

Let $\Omega = [0, 1]^2 \subset \mathbb{R}^2$ be a 2D domain, and $I_0(\mathbf{u}) : \Omega \rightarrow \mathbb{R}$ be an image-function over the domain, where $\mathbf{u} = (x, y)^T$. Suppose that a partition $P$ of the domain $\Omega$ is defined as

$$P = \{\Omega_r\}_{r=1}^n, \text{ such that } \Omega = \bigcup_{r=1}^n \Omega_r, \text{ (1)}$$

where $\bar{\Omega}_r$ is the closure of the open sub-domain $\Omega_r$. Mumford and Shah [24] formulated the segmentation problem in computer vision as follows:

Given an observed image $I_0$, find a decomposition $\{\Omega_r\}$ of $\Omega$ and an optimal piecewise smooth approximation $I$ of $I_0$, such that $I$ varies smoothly within each $\Omega_r$, and rapidly or discontinuously across the boundaries of $\Omega_r$.

A reduced case of the above Mumford-Shah model, often called the minimal partition problem, is obtained by restricting the segmented image $I$ to be piecewise-constant functions, i.e. $I = \text{constant } c_r$ inside each connected region $\Omega_r$. Mumford and Shah proposed to solve this problem by minimizing the following functional:

$$E^{MS}(I, \Gamma) = \sum_r \int_{\Omega_r} (I_0 - c_r)^2dxdy + \beta |\Gamma|, \text{ (2)}$$

where $\Gamma$ is a closed subset in $\Omega_r$, made up of the boundary curves of all regions, and the length of the curve $|\Gamma|$ acts as the regularizer, and $\beta \geq 0$ is a given tuning parameter. It is not easy to find a minimizer for functional of Equ.(2) as it is non-convex and has an unknown $\Gamma$. Some researchers attempted to solve the weak formulation or an approximation to the problem. Those attempts include the region growing approach, which minimize the Mumford-Shah functional using greedy algorithms [17], and the elliptical approximations, which embed the contour in a 2D phase-field function [1]. The Mumford-Shah functional has also been calculated using a statistical framework [39]. In Section IV, we put forward our approximation solution.

B. Related work for variational segmentation

Some variational segmentation models depend on the gradient of the original image, often called an edge-detector, to stop the evolving region boundary at the boundary of the desired object. The classical snake (or active contour) model [16] is the most important and influential edge-based method. Other variant models rely on level-set formulation to define the edge-function, including Caselles et al.’s mean curvature based geometric active contour model [3], Malladi et al.’s level-set based active contour model [22], and Caselles et al.’s geodesic model [4]. Dependence on the image gradient to stop the evolution only works for objects with edges defined by gradient. This is because the discrete gradients are bounded, and the stopping function is never zero on the edges and the curve may pass through the boundary. Moreover, the edge-function contains a Gaussian function for the purpose of smoothing. In case of the image being very noisy, the smoothing term has to be strong, which will blur the edge features as well.

Another type of models, which does not rely on the edge-detector, can detect objects with smooth or discontinuous boundaries. These models attempt to minimize the Mumford-Shah functional for segmentations, and result in a solution image consisting of a number of piecewise-constant regions. The very popular Chan-Vese model [9] is a level-set implementation of the special case of Mumford-Shah model. This model was further extended and generalized to segmentation of multi-channel images [8], and segmentation of an image into arbitrary regions [33]. The computational efficiency of these models has also been improved later-on [14], [15], [29]. Lie et al. made it possible that only one level-set function is needed to represent 2$^n$ unique regions [21]. More recent work includes Sumengen and Manjunath’s graph partitioning active contour [30] and Li et al.’s work that eliminates the need of level-set re-initialization [18].

C. Overview of the geometric flow method

Geometric flows, as a class of important geometric partial differential equations, have been highlighted in many areas, with Computer Aided Geometric Design is probably the field that benefited most from geometric flow methods. The frequently used geometric flows include mean curvature flow (MCF), weighted MCF, surface diffusion flow and Willmore flow etc. Different flows exhibit different geometric properties that could meet the requirement of various applications. The biharmonic equation, which is linear, has been used for interactive surface design [13], [32]. MCF and its variants, which are second order equations and also the most important and effective flows, have been intensively used for fairing and denoising surface meshes [2], [11], [12]. MCF cannot achieve the $C^1$ continuity at the boundary, thus for applications demanding high level of smoothness, higher order equations have to be used, e.g. the surface diffusion flow for surface fairing [27], and the Willmore flow for surface restoration [10], [36].

The construction of geometric flows is not a trivial task. In early days many geometric flows were manually manufactured by combining several geometric entities and differential operators, thus were lack of physical or geometric meaning. This drawback can be overcome by the construction of gradient descent flow. Gradient descent flow method can transform an optimization problem into an initial value (initial-boundary value) problem of an ordinary differential equation and thus is widely used in variational calculus. Constructing gradient descent flow needs to address two main issues, the definition of gradient and suitable choice of inner products. For a generic nonlinear energy functional, the gradient can be defined by Gateaux derivative. For the same energy functional, different inner products will generate different geometric flows, some of which have been mentioned above.

II. DESCRIPTION OF OUR MODEL

A. Reformulation of the minimal partition problem

It can be easily seen from Equ.(2) that for fixed partitions, the energy is minimized by setting $c_r = \text{mean}(I_0)$ in $\Omega_r$. 
Thus the minimization problem can be solved by tackling two sub-problems inter-changeably:
1) for fixed $c_r$, change the shape of $\Omega_r$;
2) for fixed $\Omega_r$, compute the new constant

$$c_r = \frac{\int_{\Omega_r} I_0(u)du}{\text{Area}(\Omega_r)} = \frac{\int_{\Omega_r} I_0(u)du}{\int_{\Omega_r} du}. \quad (3)$$

Since the second one is trivial to solve, how to solve sub-problem 1 is the focus of this paper.

With Equ.(1), we define a piecewise-constant function space over partition $P$ as:

$$S(P) = \{ f(u) = \sum_{r=1}^{n} c_r \psi_r(u) : c_r \in \mathbb{R} \}, \quad (4)$$

where $\psi_r(u)$ is the characteristic function of $\Omega_r$:

$$\psi_r(u) = \begin{cases} 1 & u \in \Omega_r \\ 0 & \text{otherwise} \end{cases}. \quad (5)$$

According to [40], we have

$$|\Gamma| = \sum_{r=1}^{n} |\Gamma_{r}| = \sum_{r=1}^{n} \int_{\Omega} \| \nabla \psi_r(u) \| du \quad = \sum_{r=1}^{n} \| \nabla \psi_r(u) \| du. \quad (6)$$

Sub-problem 1 desires that the sub-domains $\Omega_r$ are reshaped but the function values $c_r$ on $\Omega_r$ unchanged. The objective of reshaping $\Omega_r$ is to find a best fit to the boundary of objects in image $I_0$ with specified function values on the regions. To achieve this, our idea is to construct a mapping that deforms $\Omega_r$, and then the piecewise-constant image-function defined on it will be deformed subsequently. The minimal partition problem is then reformulated to be

**Problem.** For a given partition $P_a = \{ \Omega_r \}_{r=1}^{n}$ of the domain $\Omega$ and the corresponding piecewise-constant image $I_a \in S(P_a)$, find a smooth and one-to-one mapping $x(u) : \Omega \rightarrow \Omega$ such that the energy functional

$$\delta(x) = \int_{\Omega} \left[ |I_0(u) - I_a(x(u))|^2 + \beta \sum_{r} \| \nabla \psi_r(x(u)) \| \right] du \quad (7)$$

is minimized. Once the mapping $x(u)$ is determined, re-sample the image by $I_b(u) := I_a(x(u))$, so that $I_b \in S(P_b)$ is the segmented image we need.

The mapping $x(u)$ maps partition $P_a$ to a new partition $P_b = \{ \Omega_r \}_{r=1}^{n}$, possibly with varied number of regions. Fig.3 illustrates one such example.

**B. The B-Spline approximation for $x(u)$**

We construct $x(u)$ as a vector-valued bi-cubic B-spline function defined on $[0,1]^2$,

$$x(u) = \sum_{i=0}^{M+2} \sum_{j=0}^{N+2} p_{ij} \phi_{ij}(u) = \sum_{i=0}^{M+2} \sum_{j=0}^{N+2} p_{ij} N_i(x) N_j(y),$$

where $N_i(x)$ is the cubic B-spline basis function defined on the equi-spaced knots

$$[0, 0, 0, 0, 1/M, 2/M, \cdots, M-1/M, 1, 1, 1, 1],$$

and $M$ is a positive integer. A closed form expression of $N_i(x)$ could be found in [25]. $p_{ij} \in \mathbb{R}^2$ is the control point of the B-spline. This way, the total number of control points is $(M+3) \times (N+3)$. Proper values for $p_{ij}$ are to be determined so that $x(u)$, which minimizes Equ.(7), is uniquely defined.

Although it is intuitive to index all control points in manner of a 2D grid, they are reordered into a 1D array for ease of description and implementation. Thus we can assume, for example, that the point $p_{ij}$ in the 2D grid is indexed as $p_k$ in a linear array, and $\phi_k$ becomes $\phi_k$. We use vector $p = (p_0, p_1, \cdots, p_{n_1})^T$ to denote all control points.

**C. The $L^2$-gradient Flow**

Now we construct an $L^2$-gradient flow to minimize the energy functional $\delta(x)$ by a total variation. Let

$$x(u, \epsilon) = x(u) + \epsilon \Phi(u), \quad \Phi \in C_0^1(\Omega). \quad (10)$$

Then we have

$$\delta(x(u), \Phi) = \frac{d}{d\epsilon} \delta(x(u, \epsilon)) \big|_{\epsilon=0}, \quad (11)$$

where $x = x(u)$ and

$$\delta(x(u), \Phi) = \int_{\Omega} \left[ 2[I_0(u) - I_a(x(u))] (\nabla I_a(x(u))^T \delta(x) + \beta \sum_{r=1}^{N} \left( \nabla \psi_r(x) \right)^T \psi_r(x) \delta(x) \right] du. \quad (12)$$

Let $\Phi = I_2 \phi$, where $I_2$ is the identity matrix and $\phi \in C_0^1(\Omega)$, and together with Equ.(10), there is $\delta(x) = \Phi = I_2 \phi$. To deduce the associated Euler-Lagrange equation for $x(u)$, parameterizing the descent direction by an artificial time $t \geq 0$ yields the following weak form $L^2$-gradient flow that defines the motion of $x(u)$:

$$\int_{\Omega} \frac{\partial x}{\partial t} \phi \ du = - \int_{\Omega} \left[ 2[I_0(u) - I_a(x(u))] \nabla I_a(x(u)) \phi + \beta \sum_{r=1}^{N} \left( \nabla \psi_r(x) \right)^T \psi_r(x) \right] \phi \ du. \quad (13)$$

So far we have converted a minimization problem to a time-dependent PDE, which relies on the pixel intensity of the
III. NUMERICAL SOLUTION

The PDE (13) is solved numerically, thus the desired segmentation image \( I_0 \) is approached via a series of intermediate images \( I^{(0)} = I_a, I^{(1)}, I^{(2)}, \ldots, I^{(\alpha)} = I_b \). We will detail how this is achieved in this section.

A. The spatial and temporal discretization

The PDE (13) can be discretized, using finite element method in the spatial domain, into a system of ordinary differential equations (ODE):

\[
M \frac{dp}{dt} = -q,
\]

where \( M = [m_{k\ell}]_{k,\ell=1}^{n_1 n_2}, q = [q_k]_{k=0}^{n_1} \),

\[
m_{k\ell} = \int_{\Omega} \phi_k \phi_{\ell} I_2 \ du,
\]

\[
q_k = \int_{\Omega} \left[ 2I_0(u) - I_a(x) \right] \frac{\nabla I_a(x)}{\|
abla I_a(x)\|} \phi_k \ du + \beta \sum_{r=1}^{n} \frac{\nabla^2 \psi_r(x) \cdot \nabla \psi_r(x)}{\|
abla \psi_r(x)\|} \phi_k \ du.
\]

To preserve the overall shape of \( \Omega \), the control points along the boundary of \( \Omega \) are not allowed to alter, so only the inner ones change. When reordering the 2D array of control points into a linear array, it is desirable that all inner control points are numbered “earlier” than the boundary ones, as

\[
P_0, \ldots, P_{n_0}, P_{n_0+1}, \ldots, P_{n_1}
\]

and the B-spline bases are correspondingly reordered. With the new order, vector \( p \) can be written into block form as \( p = (p_c, p_b)^T \), where \( p_c \) is the vector for all inner control points and \( p_b \) for all boundary control points. Correspondingly there is

\[
M = \begin{pmatrix} M_{cc} & M_{cb} \\ M_{bc} & M_{bb} \end{pmatrix}, \quad q = \begin{pmatrix} q_c \\ q_b \end{pmatrix}.
\]

Since only the inner control points are unknowns in (14), \( M_{bc} \) and \( q_b \) can be dropped off the system. Moreover, for the stationary boundary control points, there is \( dp_b/dt = 0 \). Therefore, we end up with

\[
M_{cc} \frac{dp_c}{dt} = -q_c.
\]

This ODE can be solved using the forward Euler scheme if an initial condition is specified. More specifically, the right-hand-side terms in the \( s \)-th iteration step are treated as known quantities which are computed from the previous step. Then the final linear system is

\[
M_{cc} p_c^{(s+1)} = M_{cc} p_c^{(s)} - \tau q_c^{(s)}.
\]

B. Image resampling

Since the solution to Eq.(13) is approached numerically via a series of intermediate mappings \( x^{(0)}(u), x^{(1)}(u), \ldots \), the piecewise-constant image needs to be updated accordingly. Once the mapping \( x(u) \) at the \( s \)-th step is determined, we update the image via \( I^{(s)}(u) := I^{(0)}(x(u)) \), so that \( I^{(s)} \) is a better approximation to \( I_0 \) than \( I^{(s-1)} \).

The discrete version of \( I^{(s)}(u) \) is obtained by resampling at the pixel grid points \( u_{pq} = (x_p, y_q)^T \) with

\[
x_p = \frac{pW}{H}, \quad y_q = \frac{qH}{W}, \quad p = 0, \ldots, W, \quad q = 0, \ldots, H,
\]

where \( W \) is the width and \( H \) is the height of the image. The equation \( I^{(s)}(u) := I^{(0)}(x(u)) \) manifests that re-sampling is much like a backward lookup process: for each pixel \( u_{pq} \in \text{domain}(I^{(s)}) \), its intensity assumes the intensity of “pixel” \( x(u_{pq}) \) of image \( I^{(0)} \). Since \( x(u_{pq}) \) has a very little chance to be a grid point, we round it off to the closest pixel \( u_{a,b}^{(0)} \), then set \( I^{(s)}(u_{pq}) := I^{(0)}(u_{a,b}) \). This way, for each pixel \( u_{pq}^{(s)} \) of the new image the mapping tells from which pixel in the old image it comes.

We choose such a backward lookup because \( x(u) \) is a actually a backward mapping, as can be inferred from \( I^{(s)}(u) := I^{(0)}(x(u)) \). A forward lookup, though more intuitive, is not feasible, as a forward mapping is not handy. If we were to do it this way, we need to start with each pixel \( u_{pq}^{(0)} \in \text{domain}(I^{(0)}) \) and try to find its correspondence in \( I^{(s)} \), using \( I^{(s)}(x^{-1}(u)) := I^{(0)}(u) \) (transformed from the preceding equation). But the forward mapping \( x^{-1}(u) \) is not known yet. Even if this forward mapping can be constructed, \( x^{-1}(u_{pq}^{(0)}) \) is unlikely to be a grid point either, thus the intensity of each grid point has to be interpolated from its surrounding intensity values.

Note that during the resampling, two adjacent pixels that previously belong to one region could be mapped to two separate regions, resulting an automatic topology change of partitions.

C. The initial control points setup

To start the evolution, we need to set up an initial mapping \( x^{(0)}(u) \) and an initial piecewise-constant image \( I_0 \). The mapping \( x^{(0)}(u) \) is chosen to be a self-mapping: \( x^{(0)}(u) = u \), so that \( I^{(0)}(u) = I_a(x^{(0)}(u)) = I_a(u) \). Control point coordinates that satisfy this condition can be found in [35]: for a \((M+3) \times (N+3) \) control grid, the point coordinates are the Cartesian product between two sets \( \{i_1 \}_{i=0}^{M} \) and \( \{j_2 \}_{j=0}^{N} \), where

\[
\alpha_i = \begin{cases} 0 & i = 0, \\
\frac{1}{3} (M + 1) & i = 1, \\
\frac{i}{M+1} & i = 2, \ldots, M, \\
1 - \frac{i}{M+1} & i = M + 1, \\
1 & i = M + 2,
\end{cases}
\]

and \( \beta_j \) is defined in a similar way.
sufficient and necessary condition for successful resampling. This implies that \( x(u) \) defines a regular surface. However, this could be violated during the evolution. As shown in Fig.1c, the mapping is soon becoming irregular in the next few iterations. An irregular surface in our system can be imagined as a planar surface folding into a self-intersection.

Our solution is to reset the control grid to its initial position and start a new evolution over again. However, the segmented image obtained so far must be carried over into the new evolution. Therefore, if control points need to be reset at the \( s \)-th step, we set \( I(0) := I(s) \) for the new evolution. The sufficient and necessary condition for \( x(u) \) to be regular is that the normal vector is non-zero everywhere on the surface, i.e. \( x_x \times x_y \neq 0 \), where the partial derivatives \( x_x \) and \( x_y \) are the two tangential vectors.

### IV. EXTENSION TO PIECEWISE-SMOOTH MODEL

So far we have presented a new edge-detector free model that is comparable to the Chan-Vese model. However, the Chan-Vese model is often incapable of handling images with intensity inhomogeneity. Particularly for medical images, intensity inhomogeneity often occurs, usually due to technical limitations or artifacts introduced by the object being imaged. Segmentation of such images usually requires intensity inhomogeneity correction as a preprocessing step or more sophisticated models than piecewise-constant models. Vese et al. [31] independently proposed two similar region-based models for more general images. Aiming at minimizing the Mumford-Shah functional, both models cast image segmentation as a problem of finding an optimal approximation of the original image by a piecewise-smooth function. Michaelovich et al. [23] proposed an active contour model using the Bhattacharyya difference between the intensity distributions inside and outside a contour. Li et al. [19] proposed a new, local-homogeneous method, in a variational level set formulation. They first define a region-scalable fitting (RSF) energy functional in terms of a contour and two fitting functions that locally approximate the image intensities on the two sides of the contour. The optimal fitting functions are shown to be the averages of local intensities on the two sides of the contour. Since this model is not global-homogeneous, it suffers a serious problem that the result largely depends on the initial contour, which has been addressed in more recent work [34].

Motivated by Li et al.’s work, we extended our piecewise-constant model to a piecewise-smooth model. In the minimal partition problem, the constant \( c_r \) in Eqn.2 can be replaced with a continuous function \( f_r(u) \) for each region:

\[
E^{MS}(I, \Gamma) = \sum_r \int_{\Omega_r} [(I_0 - f_r(u))^2 + \beta |\Gamma|].
\]

To minimize the above functional, in the second sub-problem Eqn.3, instead of computing one single constant for the whole region \( \Omega_r \), we compute a continuous function

\[
f_r(u) = \frac{\int_{\Omega_r} K(u - y)I_0(y)\,dy}{\int_{\Omega_r} K(u - y)\,dy}, \quad u, y \in \Omega_r,
\]

where \( K \) is a nonnegative function and same as in [19] it is chosen as a Gaussian kernel

\[
K_o(u) = \frac{1}{2\pi\sigma^2} e^{-|u|^2/2\sigma^2}.
\]

Thus for each pixel \( u_{pq} \in \Omega_r \), \( f_r(u_{pq}) \) is a weighted average of its local vicinity within domain \( \Omega_r \). So far with minus modification to sub-problem 2 in the piecewise-constant formulation, we have obtained a piecewise-smooth model that shares the same framework with the piecewise-constant model.

We maintain two data structures for each region in our implementation, one for pixels that constitute the region...
boundary and the other for all interior pixels. This facilitates the computing of vector \( q_k \) of Equ.(15), as in the piecewise-
constant model, \( \nabla I_a(x) \), \( \nabla \psi_1(x) \) and \( \nabla^2 \psi_1(x) \) are all zeros for interior pixels. For the piecewise-smooth model, evaluating vector \( q_k \) of Equ.(15) becomes more costly, as \( \nabla I_a(x) \) is no
longer zero for interior pixels.

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

A. Solving the linear system

In Equ.(19) \( M_{cc} \) is a constant sparse matrix and does not change from step to step, since it only depends on the basis
functions. It is natural to construct the matrix at the pre-
processing stage and use it throughout. Since the \( x \)- and the \( y \)-component are independent, Equ.(19) can be split to two half-
sized sub-systems and solved individually. For such a linear system, our first option was using the PARDISO package [26], which is among the fastest sparse system solvers. Yet our
experiments show that it still consumes a fairly amount of CPU resources. Another option is to compute \( M_{cc}^{-1} \) at the pre-
processing, then solving the linear system in each step turns into a matrix-vector multiplication. However, when the control grid is of fair size, the inversion is expensive and even the cost of multiplication between vector \( q_k \) and the non-sparse \( M_{cc}^{-1} \) is not negligible. Fortunately, this inversion-
multiplication strategy could be accelerated in our system.

Since matrix elements are computed according to a specific
pattern, \( i.e. \) each one being an integral of two basis functions, \( M_{cc} \) can be written as a Kronecker product of two smaller-sized matrices, so is \( M_{cc}^{-1} \). This property makes it possible that both the inversion and the multiplication are done in a much
more efficient way. As we can see from Table I, the cost for solving the linear system often accounts for less than 1% of
total execution time. More details are given in the Appendix.

B. Region management

In level-set active contour models, the evolving curve \( C \)
is embedded as the zero level set of a Lipschitz continuous function \( \phi \), \( i.e. \), \( C(\phi) = \{ u \in \Omega : \phi(u) = 0 \} \), with \( \phi \) having opposite signs on each side of \( C \). The segmented object of interest is implicitly represented by pixels \( \{ u_{pq} : \phi(u_{pq}) > 0 \} \)
and the background is represented by \( \{ u_{pq} : \phi(u_{pq}) < 0 \} \), or vice versa. Two groups of pixels, each forming a connected
region but not inter-connected, are considered as one integrated
regions as long as they both satisfy \( \phi(u_{pq}) > 0 \). There is no signed distance function in our geometric flow based
model, therefore each individual region has to be stored and
maintained explicitly. This extra region management work
increases the complexity of the implementation, yet it offers
more flexibility in controlling the total number of regions and
the size of each region.

The resampling process (Section III-B) yields a new image,
intensity of each pixel belonging to \( \{ c_r \} \), the piecewise-
constants of the image in the previous iteration. To solve sub-
problem Equ.(3), pixels are first regrouped to form a set of new
regions, and then the averaged constants are re-evaluated.
Each new region is collected by a flood-fill scanline algorithm
in our implementation. Two disjoint regions are regarded as
two individual regions, even if they share the identical constant

<table>
<thead>
<tr>
<th>Image</th>
<th>#iter</th>
<th>time (sec)</th>
<th>RHS</th>
<th>SOL</th>
<th>EVAL</th>
<th>BLUR</th>
<th>FILL</th>
<th>MRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>fig6(a)</td>
<td>200</td>
<td>5.0</td>
<td>40%</td>
<td>0%</td>
<td>20%</td>
<td>40%</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>fig6(b)</td>
<td>350</td>
<td>112.0</td>
<td>40%</td>
<td>0%</td>
<td>24%</td>
<td>30%</td>
<td>2%</td>
<td>1%</td>
</tr>
<tr>
<td>fig6(c)</td>
<td>350</td>
<td>101.4</td>
<td>43%</td>
<td>1%</td>
<td>17%</td>
<td>3%</td>
<td>3%</td>
<td>3%</td>
</tr>
<tr>
<td>fig6(d)</td>
<td>150</td>
<td>10.5</td>
<td>45%</td>
<td>1%</td>
<td>22%</td>
<td>27%</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>fig6(e)</td>
<td>200</td>
<td>22.5</td>
<td>41%</td>
<td>0%</td>
<td>13%</td>
<td>40%</td>
<td>4%</td>
<td>2%</td>
</tr>
<tr>
<td>fig6(f)</td>
<td>300</td>
<td>10.7</td>
<td>42%</td>
<td>1%</td>
<td>16%</td>
<td>37%</td>
<td>3%</td>
<td>1%</td>
</tr>
</tbody>
</table>

Table I: System performance for segmenting six images in Fig.6. Percentage of total running time are given for six tasks: RHS—forming the right-hand-side vector of the linear system of equations; SOL—solving equation with matrix-vector multiplication; EVAL—evaluating Equ.8; BLUR—Gaussian filtering; FILL—scanline flood-fill algorithm to detect new regions; MRG—mergeing regions if needed.
value.

During the evolution of a region, it is possible that a region in the old image turns into more regions in the new image. With our region management strategy, it is not difficult to monitor such topological change. Recall there is $I^{(s)}(u_{pq}^{(s)}) := I^{(0)}(u_{pq}^{(0)})$ for re-sampling, each pixel in the new image knows from which region it comes, and thereby each new region knows where it is from. Yet certain region splitting is caused by noise in the image or the round-off error in re-sampling. Fig.2 illustrates one such example, where the small region in the right-most figure is either a noise or caused by round-off errors. In this case, this small region is merged with the background region. Such a region, surrounded by one and only one other region, is called an island region. Merging occurs if an island region contains fewer pixels than a given threshold. Region merging also alleviates the over-segmentation issue.

C. Results

Our algorithm is implemented in C++ and all the experiments ran on a laptop with Intel Core 2 Duo 2.80GHz CPU, yet no multi-core programming was involved. How do we determine the number of control points? Normally more control points mean that more basis functions are used to approximate the mapping, making it easier to catch small details such as high curvature an object. However, if the number of control points is so large as to be above a certain level, adding more control points only increases the size of the linear system, bringing no quality improvement to the segmentation result. In our system, the dimension of the control grid is about $1/10$ to $1/5$ of the image grid, i.e. using a $M \times N$ control grid for an $5M \times 5N$ image. The temporal step-size $\tau$ is chosen to be a fraction of $1/(MN+1)$ so that $\tau$ is related to the total number of control points.

Fig.3 to 5 are the results obtained by our piecewise-constant model. Fig.3 contains three different geometric shape, one of which has a blurred boundary. The given initial partition breaks down as it evolves, reaching seven regions as the stable state. In Fig.4 we show how we can detect lines and curves in a image. In Fig.5, we show how our model works on a synthetic highly noisy image. Similarly, the bear and the trees can hardly be segmented with edge-based methods as it is not easy to detect a boundary from scattered spots. Also note that the experiment shows the bear is detected earlier than the two trees, as the shapes of the trees have larger curvatures.

Our piecewise-smooth model has been tested on several images with intensity inhomogeneity. Fig.6 shows the segmentation results for synthetic and real images. Li et al. [9] has reported that the mean shift algorithm does not perform well on these challenging images. Fig.9 and 10 show more examples for segmenting ultrasound and MR images. Table I shows the execution time for the results in Fig.6 and the time percentage of each task with our proposed method.

VI. DISCUSSIONS AND CONCLUSION

A. Comparison with level set methods

We have presented a geometric flow approach for the minimal partition problem of image segmentation. This approach is essentially a minimization of energy based on shape evolution. Given that there have been so many publications in recent years on image segmentation using energy minimization, we would like to make comparison with the existing method, especially the popular level set methods.

We applied the classical Chan-Vese model [9] and Li et al.’s model to the eight images in Fig.6, 9 and 10. The results of the C-V model are shown in Fig.7 and 11, and the results of Li’s model are shown in Fig.8 and 12. The classical C-V model is less capable of handling intensity inhomogeneity, while Li’s model performs very well on these images in terms of both quality and execution time (time is given in the captions). For the six images in Fig.6, Li’s model beats ours by 2x - 15x times faster. For Fig.9, the piecewise-constant version of our model is capable to get a decent result, thus the execution time beats Li’s model. For Fig.10, the performance of our model is quite close to Li’s model. Li et al. [19], [20] has reported a 25x - 60x times performance gain over Chan-Vese’s piecewise-smooth model [33] for images in Fig.6. If it is true, our method is faster than Chan-Vese’s PS model. The above experimental data shows that although our method is often not as fast as the fully developed state-of-the-art level set techniques, it still shows potentials.

Contours obtained by our methods sometimes show some irregularity compared to those obtained by the level-set methods. Level-set contours are derived from algebraic representations, while our contours are formed via repeated image resampling and discretization (§III-B). People tend to assume that image objects have smooth boundaries thus smooth contours are preferred, yet for some highly noised or blurred images, it is...
often hard to tell where exactly the boundaries are. Although the soundness of the above assumption is questionable, we have to admit that smooth contours are more desirable in many applications. For example, sufficiently smooth 3D surface models are often to be constructed following the segmentations of slices of medical images. To tune the smoothness of the segmentation contours, a length parameter $\beta$ is used in our model (see Equ. 7, 13 and 15). This parameter usually works well in lessening irregularity, but we also found that for some noisy and high intensity-varying images a large $\beta$ value slows down the computation convergence. We consider sacrificing efficiency for regularity is not encouraged, as smoothing or fairing irregular curves and surfaces nowadays can be done very efficiently with mature techniques (including those geometric flow based approaches [2], [11], [12], [27]). Thus we prefer to adopt small $\beta$ for the contour evolution, and leave the regularization to a post-processing step. On the other hand, post-processing is usually unavoidable, even for the level-set based methods. In Chan-Vese model [9], there is a length parameter $\mu$ that intends to control the total number and the sizes of the objects. Yet this parameter does not perform very well in controlling the number of objects, thus often leads to many small unwanted regions (see Fig.12). These regions have to be properly treated at the post-processing. In our model, the region management mechanism is more powerful in controlling the total number and the sizes.

**B. Limitations**

As the Chan-Vese model, our model is not based on an edge-function to define the energy functional, objects whose boundaries are not defined by gradient or with very smooth boundaries can be easily detected. Even for very noisy images, a denoising pre-process is not necessary.

The variational formulation in the Chan-Vese model is non-convex and a typical gradient-descent implementation is not guaranteed to converge to the global minimum and can get stuck in local minima. A typical case is where the contour is stuck at the outer boundary of an object with an interior hole. It has been proposed that various tricks can be devised to improve the global convergence. One technique, which is used in the original paper [9], is to modify the delta function in the Euler-Lagrange equation so that it is nonzero everywhere. This corresponds to allowing contours to be initiated everywhere in the image, enhancing the chance of capturing the global minimum. Another idea is to initialize the optimization with a large number of small close contours uniformly distributed in the image, which has a similar effect. A more novel, and fundamentally different, approach has been proposed more recently in [6], [7]. The basic idea is to convexify the objective function by taking advantage of the implicit geometric properties of the variational models. Using an auxiliary variable, the Chan-Vese model can be recast in a convex minimization problem.
Fig. 13. The initial contour matters: only objects overlapping with the given contour/region are detected. This manifests that our model behaves as a local minimizer.

Our geometric flow approach suffers the similar local minima problem, as the Euler-Lagrange equation for \( x(u) \) in our model acts only as a local minimizer. Thus the evolution result depends on the initial segmentation image \( f(0) \). If the initial contour intersects the object of interest, this model works just fine. If there is no overlapping between two, the evolving contour may not “flow” to that object. Fig.13 is an example showing that different initial partitions leading to different results. Hopefully, if one intends to automatically detect all objects of an image, certain techniques can be used to find more meaningful initial contours, e.g. the locus of the zero-crossing of the Laplacian of the smoothed images [28] or the seed points algorithm with a light-weighted merging [37].

C. Further Extensions

As the scalar Chan-Vese model can be easily extended [8], it is also straightforward to extend our piecewise-constant model for segmenting vector-valued images (such as RGB or multispectral). For a \( N \)-channel image, Eq.2 becomes

\[
E^{MS}(I, \Gamma) = \sum_{r} \int_{\Omega_{r}} \frac{1}{N} \sum_{k=1}^{N} \lambda_{k}(I_{0} - c_{k}^{r})^{2} dx dy + \beta |\Gamma|, \tag{24}
\]

where \( c_{k}^{r} \) is the \( k \)-th component of the constant vector \( c_{r} = (c_{r}^{1}, \cdots, c_{r}^{N}) \) over region \( \Omega_{r} \), and \( \lambda_{k} \) is the weight for each channel.

With the above extension to vector-valued model, texture segmentation is possible. We can simply follow the framework of [5]; first convolve Gabor functions with the original textured image to obtain different channels and then some of these channels will be the input to the vector-valued segmentation algorithm.

As the level set technique, our model extends trivially to high dimensions. To segment 3d images, the formulation can be simply modified by adding a third dimension to \( u \), and turning Eq.8 into

\[
x(u) = \sum_{i=0}^{M+2} \sum_{j=0}^{N+2} \sum_{k=0}^{L+2} \mathbf{p}_{ijk} N_{i}(x) N_{j}(y) N_{k}(z).
\]

APPENDIX A

EFFECTIVE INVERSION OF MATRIX \( \mathbf{M}_{cc} \)

Suppose two sets of basis functions

\[
N_{i}(t), \quad i = 1, \cdots, m, \text{ and } \tilde{N}_{j}(t), \quad j = 1, \cdots, n,
\]

are defined on \( \mathbb{R} \). Let

\[
\phi_{(i-1)n+j}(x, y) = N_{i}(x) \tilde{N}_{j}(y),
\]

be a set of 2D basis functions defined on the \( xy \)-plane. Define

\[
d_{\alpha\beta} = \int_{\mathbb{R}^{2}} \phi_{\alpha}(x, y) \phi_{\beta}(x, y) dx dy,
\]

and the matrix \( D = [d_{\alpha\beta}]_{\alpha,\beta=1}^{m,n} \). Now we describe how to compute \( D^{-1} \) efficiently.

Suppose \( i, j \in \{1, \cdots, m\} \) and \( \bar{i}, \bar{j} \in \{1, \cdots, n\} \), and also

\[
\alpha = (i-1)m + j, \quad \beta = (\bar{i}-1)n + \bar{j},
\]

by introducing two notations \( c_{ij} \) and \( \bar{c}_{\bar{i}\bar{j}} \) we have

\[
d_{\alpha\beta} = \int_{\mathbb{R}^{2}} N_{i}(x) \tilde{N}_{j}(y) N_{\bar{i}}(x) \tilde{N}_{\bar{j}}(y) dx dy
\]

\[
= \int_{\mathbb{R}^{2}} N_{i}(x) N_{\bar{i}}(x) dx \int_{\mathbb{R}^{2}} \tilde{N}_{j}(y) \tilde{N}_{\bar{j}}(y) dy
\]

\[
= c_{ij} \bar{c}_{\bar{i}\bar{j}}, \tag{25}
\]

\( D \) can be written as

\[
D = \begin{bmatrix}
c_{11} \bar{C} & c_{12} \bar{C} & \cdots & c_{1m} \bar{C} \\
c_{21} \bar{C} & c_{22} \bar{C} & \cdots & c_{2m} \bar{C} \\
\cdots & \cdots & \cdots & \cdots \\
c_{m1} \bar{C} & c_{m2} \bar{C} & \cdots & c_{mm} \bar{C}
\end{bmatrix}
\]

\[
= (C \otimes \bar{C})
\]

\[
= (C \otimes \mathbf{I}_{n}) \text{diag}\{\bar{C}, \bar{C}, \cdots, \bar{C}\}, \tag{26}
\]

where \( \otimes \) denotes the Kronecker product of two matrices, \( \mathbf{I}_{n} \) is the identity matrix, and

\[
C = \begin{bmatrix}
c_{11} & \cdots & c_{1m} \\
\cdots & \cdots & \cdots \\
c_{m1} & \cdots & c_{mm}
\end{bmatrix}, \quad \bar{C} = \begin{bmatrix}
\bar{c}_{11} & \cdots & \bar{c}_{1n} \\
\cdots & \cdots & \cdots \\
\bar{c}_{n1} & \cdots & \bar{c}_{nn}
\end{bmatrix}.
\]

Using (26), \( D^{-1} \) can be computed as

\[
D^{-1} = C^{-1} \otimes \bar{C}^{-1}
\]

\[
= (C^{-1} \otimes \mathbf{I}_{n}) \text{diag}\{\bar{C}^{-1}, \bar{C}^{-1}, \cdots, \bar{C}^{-1}\}. \tag{27}
\]

Therefore, we only need to compute the inverse of two small-sized matrices, with the computational complexity being \( O(m^{3}) + O(n^{3}) \). Actually for solving Eq.(19), there is no need to assemble \( D^{-1} \) explicitly. With Eq.(27), vector \( q_{c} \) can be left-multiplied first by \( \bar{C}^{-1} \) and then by \( C^{-1} \), reducing the complexity from \( O(m^{2}n^{2}) \) to \( O(mn^{2} + m^{2}n) \). Moreover, both \( C^{-1} \) and \( \bar{C}^{-1} \) are numerically close to banded, i.e. the majority of its off-diagonal elements are small and can be treated as zeros, thus further speed-up can be achieved (the complexity is one order lower) by sacrificing a little precision.

The above derivation is valid for B-spline defined on uniform knots with spacing 1 and with multiple end knots. However, the B-spline in our model are defined on knots with spacing \( \frac{1}{M} \) as Eq.(9). Then \( c_{ij} \) needs to be scaled by a factor \( \frac{1}{M} \).

Instead of evaluating \( c_{ij} \) numerically, their exact values can be obtained using the closed form representation of the B-spline basis. We put all non-zeros in the upper triangular of

\[
\begin{bmatrix}
\mathbf{M}_{cc}
\end{bmatrix}
\]

Copyright (c) 2011 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.
and other elements can be deduced according to relations 
\[ c_{m+2-i,m+2-j} = c_{ij} \quad \text{and} \quad c_{ij} = c_{ij}. \]

**ACKNOWLEDGMENT**

Images of Fig.6 and 10 are courtesy of University of Graz and Li et al.\[19\]. We want to thank Dr. Shiming Xiang, Lingfeng Wang and Ying Wang from Institute of Automation, Chinese Academy of Sciences, for their valuable discussions.

**REFERENCES**


Guoliang Xu is a professor at Academy of Mathematics and System Sciences, Chinese Academy of Sciences. He is also an faculty member of The State Key Laboratory of Scientific and Engineering Computing; the Institute of Computational Mathematics and Scientific/Engineering Computing of the Chinese Academy of Sciences. He is on the editorial boards for the Journal of Numerical Methods and the Computer Application and the International Journal of Information and Computational Science. He is a member of The Chinese Computational Mathematical Society and a council member of The Beijing Computational Mathematical Society.