Christopher V. Alvino and Anthony Yezzi, "Fast Mumford-Shah segmentation using image scale space bases", Proc. SPIE 6498, Computational Imaging V, 64980F (2007).

http://dx.doi.org/10.1117/12.715201

Fast Mumford-Shah Segmentation using Image Scale Space Bases

Christopher V. Alvino and Anthony J. Yezzi

Georgia Institute of Technology, School of Electrical and Computer Engineering, Atlanta, GA, USA

ABSTRACT

Image segmentation using the piecewise smooth variational model proposed by Mumford and Shah is both robust and computationally expensive. Fortunately, both the intermediate segmentations computed in the process of the evolution, and the final segmentation itself have a common structure. They typically resemble a linear combination of blurred versions of the original image. In this paper, we present methods for fast approximations to Mumford-Shah segmentation using reduced image bases. We show that the majority of the robustness of Mumford-Shah segmentation can be obtained without allowing each pixel to vary independently in the implementation. We illustrate segmentations of real images that show how the proposed segmentation method is both computationally inexpensive, and has comparable performance to Mumford-Shah segmentations where each pixel is allowed to vary freely.

Keywords: Mumford-Shah functional, efficient image segmentation, image scale spaces, linear heat equation, non-linear diffusion

1.INTRODUCTION

Image segmentation involves extracting the boundaries of one or more objects in an image. Several active contour methods for image segmentation have been proposed beginning with the seminal work of Kass et al. [1] Many works have appeared based on geometric models for active contours [2–5] including edge-based models [1,4,6] and region-based models.

Mumford and Shah presented the variational problem for piecewise smooth segmentation. [7,8] Several implementations of Mumford-Shah segmentation have been presented. [9,10] Tsai et al. report applications of solving the Mumford-Shah functional for image denoising, interpolation and magnification. [10] The robustness of the Mumford-Shah model is well-known. However, so is its high computational cost.

In this paper we present a fast method for approximate Mumford-Shah segmentation that relies on representing the image regions via a reduced image basis. The idea of using a reduced image basis is that it can capture most of the accuracy of a full Mumford-Shah segmentation, at a fraction of the computational cost. In the implementation we compare between two different bases that both come from scale spaces of smoothed version of the image. One scale space is the space of images smoothed by the linear heat equation. One scale space is the space of anisotropic diffusions presented by Perona and Malik. [11]

This method can also be used as a very effective method of finding the initial contours for full basis Mumford-Shah segmentation, thus yielding an algorithm that converges quickly and very accurately obtains the piecewise smooth image functions.

2.THEORY

In this section, we will first discuss the image segmentation model proposed by Mumford and Shah. [8] We will discuss how this model has typically been implemented assuming a basis whose dimension is equal to the number of pixels in the image. We will then discuss the proposed fast approximation to computing Mumford-Shah segmentation using a reduced basis. Finally we will discuss the generalization of the proposed method to arbitrary multiple regions.

2.1. Mumford-Shah Model

The variational segmentation model proposed by Mumford and Shah assumes that the image is a piecewise smooth function, i.e., a function that is smooth within regions but not necessarily across the boundaries of these regions. The proposed cost functional simultaneously penalizes the function for deviating from the image, penalizes the function for deviating from smoothness, and penalizes the length of the boundary contour. The appropriate energy functional is,

$$\begin{aligned} E &= \sum_{i=1}^{N} \int_{R_i} \left(f_{R_i} - I\right)^2 d\bar{x} \\ &\quad + \alpha \sum_{i=1}^{N} \int_{R_i} \|\nabla f_{R_i}\|^2 d\bar{x} \\ &\quad + \beta L \,, \end{aligned} \tag{1}$$

Computational Imaging V, edited by Charles A. Bouman, Eric L. Miller, Ilya Pollak
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6498, 64980F, © 2007 SPIE-IS&T · 0277-786X/07/$18

where I is the image, R_i is the ith open subset of the image domain, N is the total number of such regions, f_{R_i} is the restriction of the model function, f, to the region R_i, $d\bar{x}$ is the image domain area element, and $\|\nabla g\|$ is the Euclidean norm of the gradient of the function g. Note that the opens regions, R_i, are always such that the R_i's are disjoint from one another, i.e., $R_i \cap R_j = \emptyset$ for $i \neq j$, and the set formed by the boundaries of these regions is a contour, C with total length, L. Finally, α and β are weighting parameters designed to control the penalty tradeoff between the three terms.

Note that the first two terms in this functional involve both the function, f, and, implicitly, the contour, C. Proper derivation of the first variation for these two terms produces terms that depend on the evolution of the contour since the regions, and hence the boundary contour, are allowed to deform. The third term does not depend on the function, f, but only the contour, C.

Evolution in the direction opposite to the gradient ensures that the cost functional is decreasing as quickly as possible. The evolution that does this involves the image functions and the contour as follows,

$$\frac{\partial f_{R_i}}{\partial t} = -2\left(f_{R_i} - I\right) + \Delta f_{R_i} \tag{2}$$

$$\begin{aligned}\frac{\partial C}{\partial t} &= \left((I - f_{R_a})^2 - (I - f_{R_b})^2\right)\vec{\mathcal{N}} \\ &\quad + \alpha\left(\|\nabla f_{R_a}\|^2 - \|\nabla f_{R_b}\|^2\right)\vec{\mathcal{N}} \\ &\quad + \beta\kappa\vec{\mathcal{N}}\end{aligned} \tag{3}$$

where t is an artificial evolution time parameter, R_a is locally the region on one side of the contour and R_b the local region on the other side of the contour, $\vec{\mathcal{N}}$ is the local unit normal vector to the contour, and κ is the local curvature of the contour.

In theory, evolution in this manner that is continuous both spatially and in evolution time, would reach a local minimum that depends on the initialization of the contour. In practice however, some finite basis must be chosen to represent the image functions, f_{R_1} through f_{R_N}. Most implementations of this technique allow each image pixel to assume its own value, i.e., they assume that the function basis has dimension equal to the number of pixels in the image and that the basis itself consists of functions,

$$\delta_{(u,v)}(x,y) = \begin{cases} 1 & \text{if } x = u \text{ and } y = v \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

for all image points, (x, y). In this manner, the function f can been seen as,

$$f_{R_i}(x,y) = \sum_{(u,v)\in R_i} c_{(u,v)}\delta_{(u,v)}(x,y)\,, \tag{5}$$

and evolution of the function f_{R_i} corresponds to evolution of each of the basis weights, $c_(u, v)$, for each region.

Unfortunately, implementation of these evolution functions is very computationally expensive, even when the functions, f_{R_i} for $1 \leq i \leq N$ are only evolved partially before each contour evolution step. This technique is similar to the implementation technique used in.[10] However, it is typically superfluous to allow each pixel to vary independently since the image regions typically have a certain smooth structure.

2.2. Proposed Function Model

The proposed technique is to choose a basis for the functions, f_{R_i}, that is significantly smaller than the pixel-by-pixel basis mentioned in the previous section but also contains enough resemblance to the functions f_{R_i} that are obtained when the pixel-by-pixel basis is used and the evolution of the function reaches its final solution. The intention is that the evolution of the contour will precede much in the same fashion as when the pixel-by-pixel basis is used but the problem of finding a local minimizer of the cost functional when the contour remains fixed will become much less computationally expensive. The choice of which basis to use is not obvious since the image regions are changing during the evolution. We will defer discussion of the specific choice of basis and will now discuss computation of the basis weights for arbitrary bases.

In general, let the image basis be Φ_b for integer $1 \leq b \leq B$. Then the function f has a restriction to region R_i given by,

$$f_{R_i} = \sum_{b=1}^{B} c_{b,i}\Phi_b\,, \tag{6}$$

where $c_{b,i} \in \mathbb{R}$ is the weight for basis vector Φ_b and for region R_i. Thus, the portion of the cost functional that involves the function f_{R_i} is,

$$E(c_{1,i},\ldots,c_{B,i}) = \int_{R_i} \left(\sum_{b=1}^{B} c_{b,i}\Phi_b - I \right)^2 + \alpha \left\| \nabla \left(\sum_{b=1}^{B} c_{b,i}\Phi_b \right) \right\|^2 d\bar{x}\,. \tag{7}$$

For each region, R_i, a necessary condition for the $c_{j,i}$'s to be a minimizer of E is that for all j,

$$\frac{\partial E}{\partial c_{j,i}} = \int_{R_i} 2\left(\sum_{b=1}^{B} c_{b,i}\Phi_b - I \right)\Phi_j + 2\alpha \left\langle \nabla\Phi_j, \sum_{b=1}^{B} c_{b,i}\nabla\Phi_b \right\rangle d\bar{x} = 0\,, \tag{8}$$

which can be rearranged into the linear system,

$$\{\Gamma_i + \alpha\Lambda_i\}\,\mathbf{c}_i = \Xi_i\,, \tag{9}$$

where,

$$\Gamma_i = \begin{bmatrix} \int_{R_i}\Phi_1\Phi_1 & \cdots & \int_{R_i}\Phi_B\Phi_1 \\ \int_{R_i}\Phi_1\Phi_2 & \ldots & \int_{R_i}\Phi_B\Phi_2 \\ \vdots & & \vdots \\ \int_{R_i}\Phi_1\Phi_B & \ldots & \int_{R_i}\Phi_B\Phi_B \end{bmatrix}, \tag{10}$$

and,

$$\Lambda_i = \begin{bmatrix} \int_{R_i}\langle\nabla\Phi_1,\nabla\Phi_1\rangle & \cdots & \int_{R_i}\langle\nabla\Phi_B,\nabla\Phi_1\rangle \\ \int_{R_i}\langle\nabla\Phi_1,\nabla\Phi_2\rangle & \ldots & \int_{R_i}\langle\nabla\Phi_B,\nabla\Phi_2\rangle \\ \vdots & & \vdots \\ \int_{R_i}\langle\nabla\Phi_1,\nabla\Phi_B\rangle & \ldots & \int_{R_i}\langle\nabla\Phi_B,\nabla\Phi_B\rangle \end{bmatrix}, \tag{11}$$

are each $B \times B$ matrices that must be computed for each region. Also,

$$\mathbf{c}_i = \begin{bmatrix} c_{1,i} \\ c_{2,i} \\ \vdots \\ c_{B,i} \end{bmatrix}, \text{ and } \Xi_i = \begin{bmatrix} \int_{R_i}\Phi_1 I \\ \vdots \\ \int_{R_i}\Phi_B I \end{bmatrix}, \tag{12}$$

are each $B \times 1$ vectors. Note that Ξ_i must be computed for each region. Note that all integrals in Eqs. (10)-(12) are with respect to $d\bar{x}$.

Thus, by solving the linear system in Eq. (9), we obtain, for each region, R_i, a vector of coefficients, $\mathbf{c}_i$, that yields the function f_{R_i} that minimizes Eq. (7) for a fixed region.

2.3. Choice of Basis

It is important to choose a basis that yields a desirable tradeoff between computational cost and segmentation performance. For this reason, we note that when during evolution via the Mumford-Shah evolution using the pixel-by-pixel basis, that the image regions resemble linear heat equation blurred version of the original image. For this reason, it makes sense to choose a basis that consists of various levels of linear heat equation (LHE) blurring. However, when the contour reaches the edges of the objects within an image, the regions, f_{R_i}, resemble regions that are smoothed via the geometric heat equation, since the geometric heat equation (GHE) smooths more within the edges of an object than across the edges. [11]

Thus, in this paper, we will illustrate examples of this proposed method using two different bases. The first is a basis that consists of a various scales from the LHE-based scale space on the image. This is the well known Gaussian scale space. The second is a basis that consists of various scales from the GHE-based scale space on the image. Using too many scale levels in the basis has the effect of making the matrix to be inverted, $\Gamma_i + \alpha\Lambda_i$, ill-conditioned. It also increases computational cost of the algorithm.

3.IMPLEMENTATION

In this section, we discuss various implementations of Mumford-Shah segmentation that we will compare against the proposed method for computational performance. In each of these implementations, Mumford-Shah evolution is set up as a two-step iterative algorithm. The first step is one in which the function evolution is performed while holding the contour, and thus the regions R_i, fixed. This evolution is performed by some predetermined amount until some stopping criterion is met. Typically, it is not iterated until full convergence since this would be extremely time consuming. The second step is where the contour evolution is performed. The more accurate the function evolution that occurs during the first step, the faster the evolution of the contour will be during the second step.[10] It is in this step that the regions R_i change, thereby rendering the information obtained during the function evolution step inaccurate. This necessitates then going back to the first step again and repeating again until the contour ceases to move.

There are various choices of how to evolve for the functions, f_{R_i} during the first step of the algorithm. We will describe some of the methods below. Note that the first two, Jacobi Mumford-Shah and Multigrid Mumford-Shah both use the full pixel-by-pixel basis shown in Eq. (4).

3.1. Jacobi Mumford-Shah

We will denote solving for the functions f_{R_i} by way of Jacobi relaxation iterations as *Jacobi Mumford-Shah*. Jacobi iterations are a relaxation technique for solving discrete boundary value problems. More detail about the Jacobi relaxation methods can be found in.[12]

The two most natural stopping criteria for Jacobi iterations are either to iterate some fixed amount of iterations, K, or to iterate until convergence, i.e., iterate until the error is below some predetermined threshold. As we will see, iterating until convergence is too computationally expensive to be practical. Thus, we shall report Jacobi Mumford-Shah for various fixed iteration amounts, denoted by K.

3.2. Multigrid Mumford-Shah

We will denote solving for the function f_{R_i} by way of Multigrid relaxation technique as *Multigrid Mumford-Shah*. Multigrid techniques solve for the solution of boundary value problems by computing the residual error on the original grid, coarsifying this residual error and solving for the coarse grid correction to the original function on the coarse grid where the computational cost is low, and then upsampling the correction to the original grid and correcting the function. Multigrid methods are known to be particularly efficient at solving elliptic or elliptic-like partial differential equations. For more detail about Multigrid relaxation methods, we refer the reader to.[12,13]

We specifically use a single Multigrid V-cycle for each function evolution, solving exactly for the coarsest scale correction. In addition, and use zeros steps of Jacobi pre-relaxation and one step of Jacobi post relaxation per grid level. According to the notation in[12] this corresponds to the parameters, $\nu_1 = 0$ and $\nu_2 = 1$.

3.3. Reduced Basis Implementation

We will denote representing the functions f_{R_i} by way of the reduced basis shown in Eq. (6) as *Proposed LHE* or *Proposed GHE* depending on whether the basis is obtained from the linear heat equation or the geometric heat equation.

The specific basis used for each evolution method always included the image itself, I and the unit image 1 that has value 1 at each pixel. In addition, some levels of smoothing in between I and 1 are also included. Will we denote the number of basis images used in this method as $B \geq 2$ which will always imply $\Phi_1 = I$, $\Phi_B = 1$ and Φ_2 through Φ_{B-1} will be images within the scale space for increasing levels of smoothing.

3.4. Contour Evolution

In order to present a fair comparison, the same contour evolution was used for every method in this paper. Therefore, the speed of the contour evolution was determined primarily by the function accuracy. Specifically, we use level set methods in order to allow for topological changes.[14] We compute the level set function only on a narrow band of pixels around the contour for speed. The contour evolves by,

$$\begin{aligned} \frac{\partial C}{\partial t} &= \left((I - f_{R_a})^2 - (I - f_{R_b})^2\right)\vec{\mathcal{N}} \\ &\quad + \alpha\left(\|\nabla f_{R_a}\|^2 - \|\nabla f_{R_b}\|^2\right)\vec{\mathcal{N}} \\ &\quad + \beta\kappa\vec{\mathcal{N}}\,, \end{aligned} \tag{13}$$

which corresponds to evolving the level set function, Ψ, by

$$\Psi_t = -\frac{\partial C}{\partial t} \cdot \nabla\Psi \tag{14}$$

We used the fastest possible stable time step for contour evolution that ensures stability of the level set function when updating it with an explicit Forward Euler update scheme.

Note that level set methods naturally define a foreground and background region by the sign of the level set function. That is, the foreground region can be considered all regions in which the level set function, Ψ is positive and the background region is that in which Ψ is negative. Thus, this method naturally lends itself to a two-region method where the foreground function is defined by one coefficient vector, $\mathbf{c}_1$ and the background is defined by another coefficient vector, $\mathbf{c}_2$. We will call this method the *two region* method. Unfortunately, disconnected foreground regions should not necessarily have the same coefficient vector, $\mathbf{c}_1$. In the next section, we generalize this method by labeling disconnected foreground and background regions separately.

3.5. Multiregion Labeling

It is important to note that since the first two methods use the pixel-by-pixel basis, that it is not necessary to keep track of which other pixels are within the same region at a given time. This is because the pixel itself is its own basis function and therefore the pixel's intensity value is the value of the coefficient corresponding to that basis function. Therefore, it is possible to determine the value of a given pixel by only using information of the pixel itself and neighboring pixels that are within the same region. It is trivial to determine whether a neighboring pixel is in the same region when using signed distance level set functions for the representation of the contour. If the sign of the level set function is the same at the point of interest as it is at the neighboring point, then the two points are in the same region.

However, this is where the proposed method differs. Since each basis function involves all of the pixels within the image and the computation of Γ_i, Λ_i, and Ξ_i require knowledge of all of the pixels that are in the specific region and not just the neighboring pixels, it is necessary to label each pixel as belonging to a certain region. Unfortunately, two different non-neighboring pixels on the level set function having the same sign does not guarantee that the pixels are in the same region.

Thus, it is necessary to compute which pixels are in which regions. While this may seem prohibitive at first, this is handled by a very efficient two queue flood-fill labeling scheme that we will summarize in the Appendix. The output of this algorithm is a labeling that tells us which pixels are in which regions and allows us to compute Γ_i, Λ_i, and Ξ_i for each region. In this manner, we are able to label each pixel as belonging to a certain region, and thus, assign a different coefficient vector, $\mathbf{c}_i$ to region i. We will call this method the *multiregion* method.

4.EXPERIMENTS

In this section, we present experimental results for the proposed method as compared with various other segmentation techniques. In addition we stress the computational improvement of this technique over Mumford-Shah segmentations using Jacobi Mumford-Shah and Multigrid Mumford-Shah.

4.1. LHE vs. GHE Comparison

Figure 1 shows an example segmentation of a peregrine falcon image. While this image is nearly trivial to segment, we include it for the purpose of reporting computational performance, and to display a specific property of the Proposed LHE method as compared with the Proposed GHE method.

The top row in this image shows the evolution of Multigrid Mumford-Shah segmentation. The leftmost image is the original image and the initialization of the contour. The three images to the right show the evolution of the algorithm to convergence. The middle row shows the segmentation result of the Proposed LHE method using a basis of dimension $B = 3$, where $\Phi_1 = I$, $\Phi_2 = l(I, 10.0)$, and $\Phi_3 = 1$. Note that $l(I, \tau)$ denotes τ time units of Gaussian blurring on image I. Note the similar looking function regions when compared with Multigrid Mumford-Shah, especially around the boundary of the peregrine falcon. The exact evolution of the contour is slightly different but produces a nearly identical resulting segmentation. The bottom row illustrates the segmentation result of the Proposed GHE method using a basis of dimension $B = 3$, where $\Phi_1 = I$, $\Phi_2 = g(I, 10.0)$, and $\Phi_3 = 1$. Again, $g(I, \tau)$ denotes τ time units of geometric heat equation blurring on image I. Note that there is more accurate edge information on the wing of the peregrine falcon in the second image from the left.

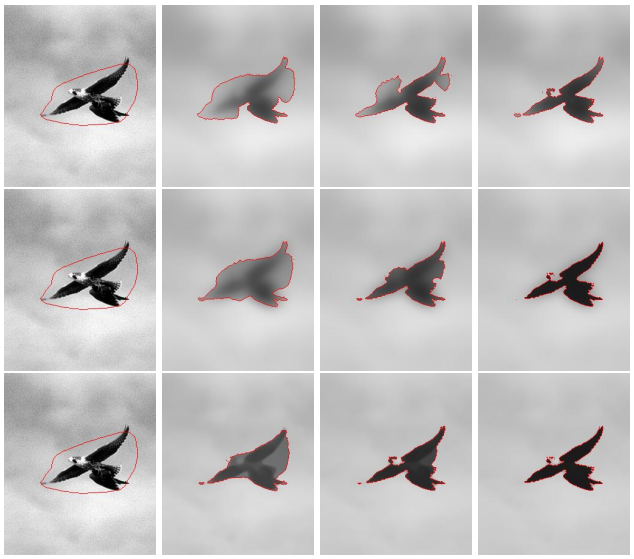

Figure 1. Peregrine falcon segmentation: For each row, progression from left to right shows segmentation evolution from initial contour to final segmentation. (Top row) Multigrid Mumford-Shah segmentation with pixel-by-pixel basis. (Middle row) Proposed LHE has very similar function regions when using drastically reduced basis. (Bottom row) Proposed GHE produces similar segmentation but function regions preserve image edges better than in top two rows.

4.2. Two Region vs. Multi-Region Results

Figure 2 compares Multigrid Mumford-Shah with the Proposed GHE method assuming two regions and the Proposed GHE method allowing for multiple regions. The top row shows the evolution of Multigrid Mumford-Shah on an image containing various types of fruit having different intensity levels. Since the Multigrid Method uses the pixel-by-pixel basis, each piece of fruit assumes its own intensity. The middle row shows the evolution of the two region Proposed GHE method. The two region Proposed GHE method has a limitation in that it only uses two sets of coefficients to represent the image, i.e., one for the foreground and one for the background. Therefore, this method is not able to represent the different fruit intensities. The bottom row shows the evolution of the multiregion Proposed GHE method. Note that in the third image from the left, when the strawberry on the bottom right of the image becomes disconnected from the remaining foreground region, it becomes darker since this region then obtains its own coefficient vector, $\mathbf{c}_i$. The final segmentation image on the right has seven distinct regions, each with its own coefficient vector. The result matches very closely with the pixel-by-pixel basis Mumford-Shah segmentation technique but at a fraction of the computational cost.

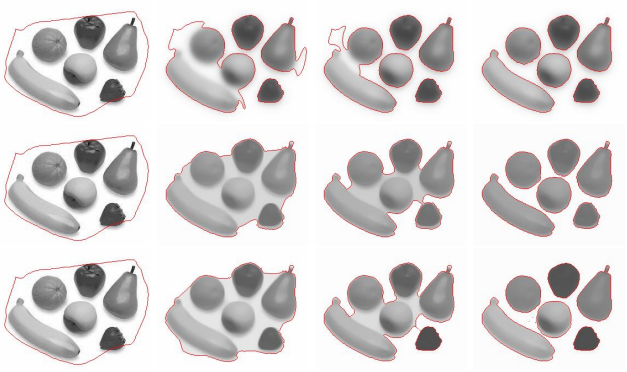

Figure 2. Fruits image: (Top row) Segmented using full pixel-by-pixel basis Mumford-Shah segmentation. (Middle row) Segmented using Proposed GHE method using two distinct regions, foreground and background. Note the lack of accuracy in mean intensity for each function region. (Bottom row) Segmented using Proposed GHE method using multiple regions. Note the ability to represent each different fruit as a separate region with distinct basis weight coefficients.

4.3. Segmentation in Noise

Figure 3 shows an example of a segmentation of the fruit image in noise. The top row shows Multigrid Mumford-Shah segmentation, the left of which is the initialization and the right of which is the final segmentation. The middle row shows the initial contour and final segmentation using the two-region proposed GHE method. The bottom row shows the initial contour and final segmentation using the multiregion proposed GHE method.

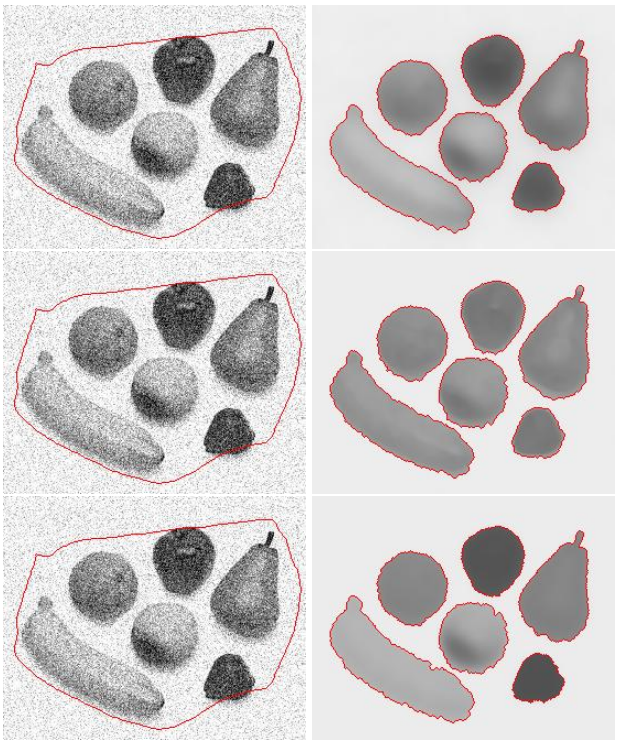

Figure 3. Noisy fruit image: (Top row) Initial contour overlayed on noisy image along with final Multigrid Mumford-Shah segmentation of image. (Middle row) Initial contour and final segmentation using two-region proposed GHE method. (Bottom row) Initial contour and final segmentation using multiregion proposed GHE method.

4.4. Computational Improvement

Table 1 shows the computational speed improvement of the proposed algorithms. In this table, "Proposed 2R LHE" represents the two region reduced basis technique proposed in this paper where there are only two different coefficient vectors, one for the foreground and one for the background. As stated before, LHE represents linear heat equation basis. "Proposed MR LHE" represents the multiregion version of the proposed method. The parameter K/B shows the number of Jacobi iterations, K for the Jacobi Mumford-Shah method or then number of basis vectors, B for the proposed techniques.

The peregrine image is of size 250×300 pixels and the fruits image is of size 275×221 pixels. The main reason for the much increased computational cost of the fruits image is that initial contour was chosen further from the final segmentation in the fruits image than in the peregrine image.

Note that the proposed method is substantially faster than Multigrid Mumford-Shah. Among the variants of the proposed segmentation method, the multiregion (MR) methods are the slowest because they are required to label each pixel as being in a specific region every iteration. While this algorithm is not slow, it bring with it some computational cost. The Jacobi Mumford-Shah method is the slowest of all.

image	method	K/B	iterations	time
peregrine	MG Mum-Shah		450	59s
peregrine	Jacobi Mum-Shah	1	3400	243s
peregrine	Jacobi Mum-Shah	2	2000	232s
peregrine	Jacobi Mum-Shah	5	1200	300s
peregrine	Proposed 2R LHE	3	200	6s
peregrine	Proposed 2R GHE	3	250	5s
peregrine	Proposed MR LHE	3	250	9s
peregrine	Proposed MR GHE	3	250	8s
fruits	MG Mum-Shah		3350	385s
fruits	Jacobi Mum-Shah	1	10800	670s
fruits	Jacobi Mum-Shah	2	6400	641s
fruits	Jacobi Mum-Shah	5	3800	810s
fruits	Proposed 2R LHE	3	600	15s
fruits	Proposed 2R GHE	3	400	9s
fruits	Proposed MR LHE	3	1100	47s
fruits	Proposed MR GHE	3	550	23s

Table 1. Computational performance of proposed method compared with Multigrid Mumford Shah (MG Mum-Shah), Jacobi Mumford Shah with various number of function evolution iterations, K, per contour evolution, and proposed technique.

4.5. Brain CT Segmentation

Figure 4 shows a multiregion segmentation of a 145×145 pixel brain CT image using the Proposed GHE method with a three-image basis. Note that the method is able to segment a majority of the important features in this image including the white objects in the center of the brain as well as the ears.

5.CONCLUSION

In this paper, we have presented a reduced basis method for fast robust approximation to Mumford-Shah segmentation. We explain how the specific choice of basis allows for accurate approximation to the full, standard "pixel-by-pixel" basis Mumford-Shah evolution, while reducing computational cost significantly. We presented a method for labeling like regions, thus enabling the use of regions with independent coefficient vectors. We showed the performance of this method for real images that were both noiseless and noisy. Future work includes determining optimal bases for Mumford-Shah segmentation.

6.APPENDIX

Here we explain the algorithm for two queue flood-fill labeling. This algorithm gives each pixel in the region a label to identify it only with points that are within the same region. Even though this algorithm is run every iteration, it is a manageable part of the computational cost of the proposed technique. The proposed multigrid technique still outperforms other comparable methods with respect to computational cost.

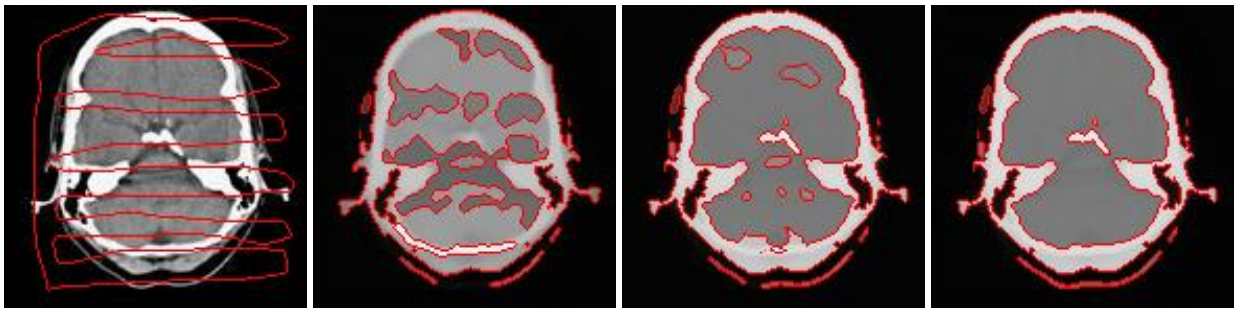

Figure 4. Segmentation of brain CT image using Multiregion Proposed GHE method.

1. $L \leftarrow 1$
2. Enqueue upper leftmost pixel into queue #1
3. While queue #1 is not empty:
   (a) Dequeue pixel p from queue #1 and label p to be in region L
   (b) For all neighbors, q, of p that have not been labeled
      i. If q and p are in the same region and q has never been in queue #1, enqueue q into queue #1
      ii. If q and p are not in the same region and q has never been in queue #2, enqueue q into queue #2
4. If queue #2 is not empty, continue to dequeue from queue #2 until an unlabeled pixel is found or until queue #2 is empty. If an unlabeled pixel, r, is found, enqueue it into queue #1.
5. Increment L
6. If queue #1 is empty terminate algorithm, otherwise go to step 3

REFERENCES

1. M. Kass, A. Witkin, and D. Terzopolous, "Snakes: Active contour models," *Int. Journal of Computer Vision* **1**, pp. 321–331, 1987.
2. V. Caselles, F. Catte, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numerische Mathematik* **66**, pp. 1–31, 1993.
3. V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *Int. J. Comput. Vis.* **22**(1), pp. 61–79, 1997.
4. R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propogation: A level set approach," *IEEE Trans. Pattern Anal. Machine Intell.* **17**, pp. 158–175, 1995.
5. A. Yezzi, S. Kichenassamy, A. Kumar, P. J. Olver, and A. Tannenbaum, "A geometric snake model for segmentation of medical imagery," *IEEE Trans. Med. Imaging* **16**(2), pp. 199–209, 1997.
6. S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Conformal curvature flows: From phase transitions to active vision," *Arch. Retion. Mech. Anal.* **134**, pp. 275–301, 1996.
7. D. Mumford and J. Shah, "Boundary detection by minimizing functionals," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 22–26, (San Francisco, CA), June 1985.
8. D. Mumford and J. Shah, "Optimal approximation by piecewise smooth functions and associated variational problems," *Comm. Pure Appl. Math.* **17**, pp. 577–685, 1989.
9. T. F. Chan and L. A. Vese, "A level set algorithm for minimizing the Mumford-Shah functional in image processing," Tech. Rep. CAM 00-13, UCLA, Department of Mathematics, 2000.
10. A. Tsai, A. J. Yezzi, and A. S. Willsky, "Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification," *IEEE Trans. Image Proc.* **10**, pp. 1169–1186, August 2001.
11. P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Machine Intell.* **12**, pp. 629–639, July 1990.

12. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, second ed., 1992.
13. W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, second ed., 2000.
14. S. Osher and J. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on the Hamilton-Jacobi equations," *Journal of Computational Physics* **79**, pp. 12–49, 1988.