A Multiscale Algorithm for Mumford-Shah Image Segmentation [1]

Tony F. Chan and Selim Esedoglu
Mathematics Department, UCLA

December 17, 2003

Abstract

We propose a new multiscale procedure for image segmentation that is based on the algorithm introduced by Song and Chan in [11]. After pointing out the presence of spurious stationary states for the original algorithm on certain types of images, we explain how the proposed procedure abates this issue and leads to a faster segmentation method.

Keywords: segmentation, variational methods, Mumford-Shah model
AMS Subject Classification: 65K10

1 Introduction

Image segmentation is a fundamental procedure in computer vision. Its goal is to automatically partition a given image into regions that contain distinct objects. A common approach is to pose segmentation as a best approximation problem: an approximation to the given, possibly very complicated, image is sought among "simple" images in which distinct regions are self evident.

The Mumford-Shah segmentation model [5] is a variational approach that adapts this point of view by exhibiting the simplified image as the minimizer of an energy. In one of its most common versions, the class of simplified images consists of functions that take at most two values, thereby partitioning the image domain into two *phases*. The energy involved in this case is the least squares distance to the given image (which enforces *fidelity* to the original image), plus a penalty for the length of the interface that separates the two phases of the approximate image. The problem is then to determine the optimal choices for the two values that the approximate image takes, and the interface that defines the regions where these values are taken.

Variational approaches to segmentation thus invariably involve minimizing over curves in the plane, which is a challenging task from a numerical

[1]This work was supported in part by NSF contract DMS-9973341, NSF contract ACI-0072112, ONR contract N00014-03-1-0888, and NIH contract P20 MH65166.

perspective. As such, there has been a great deal of research activity on how to best numerically minimize the Mumford-Shah functional or one of its simplified versions. For example, the work of Chan and Vese [1] employs partial differential equations and the level set technology of Osher and Sethian [6, 7] to implement a curve evolution on the plane that guides the curve along the path of steepest descent for the segmentation energy, thus approximating the solution by an iterative procedure that can be thought of as the numerical solution at large times of the PDE involved.

Steepest descent based techniques for minimizing Mumford-Shah like segmentation energies are often implemented by explicit time stepping for the time dependent PDE's involved, and therefore result in slow algorithms whose computational complexity scale poorly with respect to the image size. One natural way to speed up these algorithms would be to implement implicit schemes. Recently, however, new techniques have been proposed that dispense with the steepest descent approach altogether.

Gibou and Fedkiw [2] propose ignoring the interface length term in the segmentation energy, which slows down gradient descent based approaches; this allows rapid solution of the remaining terms. The regularizing role played by the length term in the original model is then taken over by preprocessing and post-processing steps that smooth out by some diffusion process the original image to be segmented and the segmentation boundary (interface), respectively.

In [11], Song and Chan take a related but different approach. The Song-Chan (SC) algorithm consists in starting with an initial guess for the solution, and then refining that initial guess by visiting each pixel in the image in a given order, and updating the region membership of the pixel in question by computing the energy for each possible value that can be taken at that pixel and selecting the one that gives the minimum value. This can be interpreted as taking very large time steps in the gradient flow, after checking explicitly that the energy is decreased by each. Compared with more traditional implementations, in many cases this algorithm can bring an arbitrary initial guess to within a small neighborhood of the solution with significantly fewer iterations. As such, it can be either used on its own, or as a preliminary step – that does most of the job very rapidly – for more accurate but slower segmentation algorithms. Moreover, unlike the approach of Gibou and Fedkiw, the SC algorithm remains faithful to the segmentation model that inspired it: at every step it is guaranteed, by definition, to decrease a version of the Mumford-Shah segmentation energy.

In this paper we propose a multiscale method that follows up on the SC algorithm. It is based on the observation that the original algorithm operates at only one scale: that of a single pixel. By generalizing the SC algorithm to operate on more scales, we arrive at a multilevel procedure that can eliminate certain spurious stationary states of the original algorithm and thus attain lower energy values. We show how the generalized algorithm can be implemented efficiently and made to reach reasonable segmentations faster than the original SC algorithm.

We should point out that there are other multiscale approaches to the segmentation problem. The one proposed in this paper is closest in spirit to the work of Koepfler et. al. [3]. There, the authors propose a technique for minimizing the piecewise constant Mumford-Shah energy with no bound on the number of phases in the segmentation. Their algorithm is based on region merging: starting from a very fine segmentation, it selectively merges neighboring distinct regions in order of greatest energy decrease. The multiscale nature of their algorithm is achieved by gradually varying the balance between the fidelity and penalty terms in the energy, and requires the selection of a "schedule" for the parameter that represents this balance in the segmentation model. Another, and more recent, multiscale procedure for segmentation has been proposed in [10].

After introducing notation and basic definitions, we recall in Section 2 the two-phase, piecewise constant Mumford-Shah model. Then in Section 3 we describe the Gauss-Seidel version of the original SC algorithm as introduced in [11]. We compare and contrast the SC algorithm with the K-Means procedure in Section 4. Section 5 is devoted to exhibiting stationary states of the original SC algorithm and to explaining the underlying reason for their presence. Then in Section 6, we propose the multiscale procedure that addresses these underlying reasons. Finally in Section 7 we present numerical results to verify that indeed in certain difficult segmentation scenarios, the proposed multiscale procedure leads to cleaner segmentations while maintaining efficiency.

2 Notation and definitions

In this section we introduce some notation that will be used in the description of the algorithms. We also recall the two phase piecewise constant version of the Mumford-Shah variational model, as it was used in [11].

The computational grid is assumed to be regular, rectangular, and of size $N \times N$. It will be represented by the set of indices

$$I = \{1, 2, \dots, N\} \times \{1, 2, \dots, N\}.$$

The original image presented for segmentation will be denoted $\{f_{i,j}\}$, where $(i,j) \in I$. A two-phase segmentation of an image defined on the grid can be described by a triplet of the form (P, c_0, c_1) where $P \subset I$, and $c_0, c_1 \in \mathbf{R}$, such that the corresponding two valued approximation to the image (i.e. the "segmented" version of the image) is

$$u_{i,j} = \begin{cases} c_1 & \text{if } (i,j) \in P, \\ c_0 & \text{if } (i,j) \in I \setminus P. \end{cases}$$

We will write $\mathbf{1}_P$ to denote the discrete indicator function of a subset P of the grid; in other words

$$(\mathbf{1}_P)_{i,j} := \begin{cases} 1 & \text{if } (i,j) \in P, \\ 0 & \text{if } (i,j) \in P^c. \end{cases}$$

Given the original image $\{f_{i,j}\}$ and a *regularization parameter* μ, our discrete version of the two-phase, piecewise constant Mumford-Shah segmentation energy [1] is

$$\begin{aligned} MS_\mu(c_0, c_1, P) = & \sum_{(i,j)\in P^c} \big(c_0 - f_{i,j}\big)^2 + \sum_{(i,j)\in P} \big(c_1 - f_{i,j}\big)^2 \\ & + \mu \sum_{(i,j)} \sqrt{\big((\mathbf{1}_P)_{i,j} - (\mathbf{1}_P)_{i+1,j}\big)^2 + \big((\mathbf{1}_P)_{i,j} - (\mathbf{1}_P)_{i,j+1}\big)^2} \quad (1) \end{aligned}$$

This type of energy was used by Chan and Vese in [1]. The first two terms in the right hand side are the *fidelity* terms: they require that the two valued minimizer of the energy should be a good approximation, in the least squares sense, to the original image. The third term on the right hand side imposes a penalty on the (anisotropic) length of the interface that separates the two domains on which the approximating function makes a transition from one of its values, c_0, to the other, c_1. By choosing the parameter μ that multiplies this term, it is thus possible to control the level of detail desired in the two-valued approximation to the original image that is the result of this approach.

This form of the two-phase, piecewise constant Mumford-Shah functional agrees with the version considered in the original paper [11]. Notice that describing the interface between the two phases by a *binary* function defined on the grid makes the regularization term (which measures the length of the interface) very anisotropic.

Given a partition P, the optimal constants c_0 and c_1 are determined in terms of P. They are given by the appropriate averages of the original image

$f_{i,j}$:

$$\begin{aligned} c_0 = c_0(P) &:= \left(\sum_{(i,j)\in I} f_{i,j} (\mathbf{1}_{P^c})_{i,j} \right) \left(\sum_{(i,j)\in I} (\mathbf{1}_{P^c})_{i,j} \right)^{-1}, \\ c_1 = c_1(P) &:= \left(\sum_{(i,j)\in I} f_{i,j} (\mathbf{1}_{P})_{i,j} \right) \left(\sum_{(i,j)\in I} (\mathbf{1}_{P})_{i,j} \right)^{-1} \end{aligned} \tag{2}$$

Hence we will sometimes write $MS_\mu(P)$ to indicate succinctly that the constants c_0 and c_1 in the evaluation of MS_μ have been chosen as such.

Let us define rectangular subgrids $R(i,j;a,b)$ of the full grid I as follows:

$$R(i,j;a,b) := \{(k,l) \in I : i \leq k \leq i+a-1 \text{ and } j \leq l \leq j+b-1\}$$

So, R corresponds to an $a \times b$ rectangular subgrid of I whose upper left hand corner is located at the (i,j)-th position of I.

3 The Song-Chan algorithm [11]

In this section we briefly recall the two phase version of the original algorithm with Gauss-Seidel style updates.

As explained in the previous section, two phase segmentations can be described by the partition P, which is a subset of the grid; the set P then determines the two constants c_0, c_1 that the two valued approximation takes inside and outside P, respectively.

The main idea of the SC algorithm is as follows: It visits each pixel in the grid in a given order, and compares the current energy value with what it would be if the P-membership of the pixel is reversed. If the reversal improves the energy, it is affected by updating P and also the values of the two constants c_0, c_1 to reflect the change in P. We now give a more precise and systematic description of this procedure below:

Algorithm: For each $(i,j) \in I$, compare the two energies $MS_\mu(P \cup \{(i,j)\})$ and $MS_\mu(P \cap \{(i,j)\}^c)$. There are two possibilities:

Case 1: $(i,j) \in P^c$. In this case, compute $MS_\mu(P \cup \{(i,j)\})$. If $MS_\mu(P \cup \{(i,j)\}) < MS_\mu(P)$, then update P as follows:

$$P \longrightarrow P \cup \{(i,j)\}.$$

Calculate the new values of c_0, c_1, and MS_μ.

Case 2: $(i,j) \in P$. In this case, compute $MS_\mu(P \cap \{(i,j)\}^c)$. If $MS_\mu(P \cap \{(i,j)\}^c) < MS_\mu(P)$, then update P as follows:

$$P \longrightarrow P \cap \{(i,j)\}^c.$$

Calculate the new values of c_0, c_1, and MS_μ.

In a single *sweep* of the algorithm through the pixels in the image, each pixel is visited once, in some order, and the procedure explained above is applied. The algorithm repeats itself until it gets to a sweep that fails to update any of the pixels, or until one of the phases (either P or $I \setminus P$) becomes empty.

An important feature of the algorithm is that if $MS_\mu(P)$, $c_0(P)$, and $c_1(P)$ are known, then the competing energies $MS_\mu(P \cup \{(i,j)\})$, $MS_\mu(P \cap \{(i,j)\})$, and their corresponding optimal constants $c_0(P \cup \{(i,j)\})$, $c_1(P \cup \{(i,j)\})$, $c_0(P \cap \{(i,j)\}^c)$, and $c_1(P \cap \{(i,j)\}^c)$ can be calculated *locally*, i.e. by a small number of operations that is independent of the size of the grid I. This is very important for computational efficiency, and hence it is important to preserve this feature in improvements to the original algorithm.

It is easy to see that the algorithm has to terminate in a finite number of steps. This follows from two facts: Every update strictly decreases the segmentation energy, and the algorithm has only a finite number of states. Moreover, in [11] it is shown that in the $\mu = 0$ case the original algorithm can converge in a single sweep for certain (simple) kinds of images.

4 Comparison with K-Means

In the absence of the regularization term (i.e when $\mu = 0$), the variational model (1) reduces to a *quantization* or *clustering* problem: it is the problem of finding two constants c_0 and c_1 that best "explain" the data in the least squares sense.

The K-Means algorithm is a technique for solving this quantization problem (and its generalization to the case with K-phases and vectorial data). We explain how it works in the two phase situation: First, as an initial guess, the data points are randomly assigned to one of the two phases. Then, given the initial partition, the optimal choice for the constant c_0 and c_1 is made; clearly, the prudent choice is, just like in formula (2), to set c_0 and c_1 to be the mean of the data points in their respective partitions. Having determined the constants, one then goes back to readjust the partitions by visiting each data point and reassigning it to one of the two phases, depending on whether it is closer to c_0 or c_1. These two stages of the algorithm, namely updating the means and the assignment of the data points to the phases, is now repeated.

There are many versions of the K-Means procedure outlined above; one basic variation on this theme concerns whether the means should be updated every time a data point gets reassigned to a different phase (in which case one predicts the new value of the two means *before* the reassignment), or only at the end of a complete sweep through all the data points. The former strategy, sometimes called *sequential* K-Means, is completely equivalent to what we have called the "Gauss-Seidel" version of SC algorithm *when the regularization term* $\mu = 0$ (while the latter strategy would be the "Jacobi" version). The close relation between the K-Means algorithm and the model of [1] was noted and exploited previously in [8]. It is also central to the algorithm of [2].

SC algorithm can thus be understood as an adaptation of the sequential K-Means procedure to the special case of image data and to the presence of the geometric regularization term that measures the length of the boundary between the various phases. When $\mu > 0$, because of the geometric term the reassignment of pixels to various phases depends not only on the number of pixels in each phase and the values of the corresponding means, but also on the membership of the neighboring pixels. This is an essential difference between the SC algorithm and K-Means. Results presented in [11] show that for many types of images, such as those well approximated by piecewise constant functions, the SC algorithm is a very fast segmentation technique. There, it is also shown that for simple images and in the absence of the perimeter term, the algorithm in fact converges in a single sweep. The K-Means algorithm has been previously studied for its convergence properties; for example in [9] the authors describe some general results. It would be very interesting to explore the implications of these convergence results for the specific segmentation setting that is the focus of [11] and this paper.

The K-Means algorithm is often used as a preprocessing step for more elaborate and costly methods. It is possible to use it in this capacity also in the two-phase segmentation setting; this is essentially what is done in [2]. However, the standard K-Means algorithm has no notion of geometry built into it. This can be seen in the numerical examples shown in Figures 5 and 12. Hence using the K-Means algorithm does not address the main difficulty with segmentation energy (1): presence of the length term. It is the length term that slows down the standard ways of minimizing (1). The motivation behind the SC algorithm is to come up with a K-Means type fast procedure that incorporates the length term and which is thus compatible with (1).

5 Stationary states of the SC algorithm

When the regularization parameter μ is nonzero, it is well known that the piecewise constant Mumford-Shah segmentation model has local minima, for instance with respect to the L^2 distance, except for very special types of images. These minima are usually few and far apart; their presence does not necessarily constitute a drawback of the model since in segmentation applications it makes little sense to expect a unique solution. Typically, a minimization technique would be considered quite successful if it finds one such local minimum.

In this section we will show that when the regularization parameter is non-zero, the SC algorithm can introduce many spurious stationary states in addition to the local minima inherent to the segmentation model. Our examples exploit a particular feature of the algorithm: it modifies the segmentation only one pixel at a time. Therefore, it can have difficulty segmenting noisy images that include noise at larger scales.

A simple example can be constructed out of binary images, where the "noise" is in the geometry. Consider the indicator function of a region with smooth but undulatory boundary as the original image to be segmented. Assume that the undulations have moderate wavelength. Then, for moderate values of the regularization parameter, we would expect the Mumford-Shah model to "straighten out" the undulations: the interface between the two phases of the segmentation should be a smoother, simpler version of the original boundary.

If such an image is sampled on a fine enough grid so that the undulations are well resolved, say at the level of several pixels, and if the original image thus sampled is taken as the initial guess for one of the phases of the segmentation, then the original SC algorithm would be unable to modify it at all. Indeed, under such conditions switching any pixel of the initial segmentation from one phase to another would at best leave the length term unaltered, and for certain increase the fidelity term.

Figure 1 illustrates our point. It shows an original binary image that consists of the indicator function of a half-space with thin and long "hairs" sticking out. These "hairs" of course make the length of the boundary (i.e. the "edge") in the image extremely and unnecessarily long. However the hairs are four pixels wide, and their tips are carefully designed. It can be easily verified that if this original image itself is taken as the initial guess for one of the phases of the segmentation, then in fact none of the pixels can be updated: *removing any of the pixels that make up the boundary of the region shown, including the hairs, would eliminate at most two edges* *and introduce at least two new edges.* Moreover, such an operation would increase the fidelity term of the energy, which is at its lowest when the segmentation agrees with the original image. Hence, the algorithm is stuck in its initial state, for any non-zero choice of the regularization parameter. Meanwhile, for any large enough (but moderate) choices of that parameter the indicator function of the half-space without the hairs can be shown to have less energy.

Other, more suggestive examples are possible. Given any original binary image that has a large enough neighborhood contained entirely in one of its phases, say the zero phase, we can place an "island" of ones, whose shape approximates that of a circle, eight pixels in diameter, in the middle of that neighborhood. Taking this modified original image as the initial guess for one of the phases, we can verify that the SC algorithm would be unable to remove the island. In a high resolution image, many such islands can be placed, imitating presence of a particular type of noise. See *Figure 2* for this type of example.

In [3] the authors describe a technique for eliminating local minima from a region growing technique for minimizing the piecewise constant Mumford-Shah functional (with no restriction on the number of regions). Their idea is to *gradually* increase the parameter μ: they start with 0, and then follow a *schedule* of increasing values, each time minimizing the energy involved and using the resulting segmentation as initial guess for the next μ value. Let us conclude this section by pointing out that such an approach will not avoid the stationary states considered above. Indeed, the examples constructed above are stationary for *all* values of $\mu \geq 0$.

6 Multiscale versions of the SC algorithm

We can begin to address the issues explained in the previous section in a straightforward way. As our discussions in that section indicate, the cure is to allow the SC algorithm to operate at more scales than just that of a single pixel. Intuitively, we thus empower the algorithm to remove noise at different scales in the geometry of the interface.

This can be accomplished by allowing the algorithm to update entire "neighborhoods" of pixels at once. For instance, at each pixel visit we can ask whether updating all the pixels in an $a \times b$ neighborhood of that pixel to either the zero phase or the one phase decreases the energy. If it does, then all pixels in the neighborhood can be set to belong to that phase. This procedure is repeated for every pixel in the image. Let us describe the

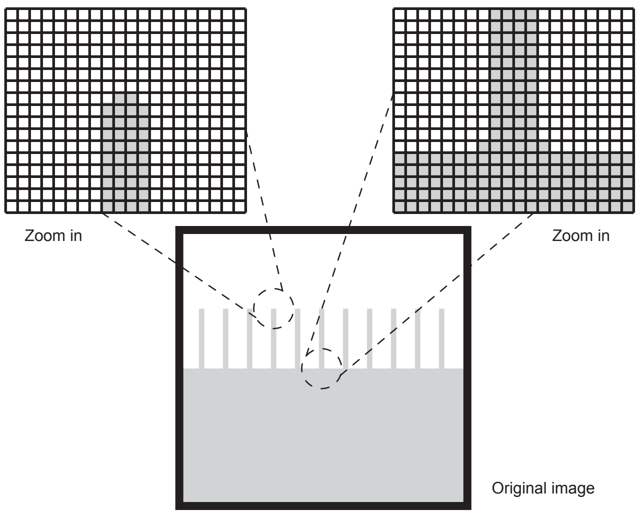

Figure 1: A steady state of the original SC algorithm. The original binary image, which is shown, is also taken as the initial segmentation. Even though the boundary of the region depicted is very large due to the presence of “hairs”, the algorithm in its original form is unable to update a single pixel. This is true for all values of the regularization parameter $\mu \geq 0$. The reason is that removing any single pixel from the region does not decrease the length term, and would increase the fidelity term in the energy.

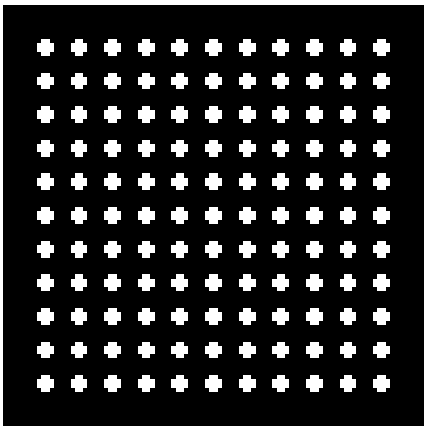

Figure 2: Another spurious steady state of the original algorithm, in the presence of regularization. Once again the original image, which is binary as shown, is taken also as the initial guess for 1-phase of the the segmentation. In this example white pixels correspond to the 1-phase. The “islands” of 1's have the shape of plus signs, and are made up of 12 pixels. As with the previous example (*Figure 1*), the state is stationary for all $\mu \geq 0$: removing any single pixel cannot decrease the edge length.

algorithm more precisely:

Algorithm: Fix integers $a, b \geq 1$. For each $(i,j) \in I$, listed in some order, compare the following three energies:

$$MS_\mu(P), MS_\mu\big(P \cup R(i,j;a,b)\big), \text{ and } MS_\mu\big(P \cap R^c(i,j;a,b)\big).$$

Then, there are three cases:
Case 1: $MS_\mu(P \cup R(i,j;a,b)) < \min\Big\{MS_\mu(P), MS_\mu(P \cap R^c(i,j;a,b))\Big\}$. In this case, update P as follows:

$$P \longrightarrow P \cup R(i,j;a,b).$$

Calculate the new values of c_0, c_1, and MS_μ.
Case 2: $MS_\mu(P \cap R^c(i,j;a,b) < \min\Big\{MS_\mu(P), MS_\mu(P \cup R(i,j;a,b))\Big\}$. In this case, update P as follows:

$$P \longrightarrow P \cap R^c(i,j;a,b).$$

Calculate the new values of c_0, c_1, and MS_μ.
Case 3: Otherwise, do not update P.

For one by one neighborhoods, the procedure explained above reduces to essentially the standard form of the algorithm. But there is an obvious disadvantage to using large neighborhoods: Namely, with a casual implementation of the generalized algorithm, the computational cost of a single sweep through all the pixels in the image is proportional to $a \times b \times N^2$. The difference in cost compared to the original version of the algorithm is due to the fact that the local computations necessary to compute the competing energies $MS_\mu(P \cup R(i,j;a,b))$ and $MS_\mu(P \cap R^c(i,j;a,b))$ from the current energy value of $MS_\mu(P)$ require us to use values from a neighborhood of $R(i,j;a,b)$.

Yet this is being too pessimistic. With a little bit of care, and by choosing the update order appropriately, the computational cost can be reduced to a constant multiple of $\min\{a,b\} \times N^2$ per sweep. In particular, when either a or b is one, the generalized sweep can be accomplished at only twice the cost of a single sweep of the original algorithm, measured in terms of the number of floating point operations involved. We now explain how this is done.

Consider applying the generalization of the algorithm with a rectangle of size $a \times 1$. Suppose that the (i,j)-th pixel has just been visited, so that the following quantities have already been computed:

$$\begin{aligned}&MS_\mu(P), MS_\mu(P \cup R(i,j;a,1)), MS_\mu(P \cap R^c(i,j;a,1)),\\&c_0(P), c_1(P), c_0(P \cup R(i,j;a,1)), c_1(P \cup R(i,j;a,1)),\\&c_0(P \cap R^c(i,j;a,1)), c_1(P \cap R^c(i,j;a,1)).\end{aligned}$$

Suppose also that we have chosen a row-wise update order, so that the $(i+1,j)$-th pixel is visited next. Then, the following quantities need to be calculated:

$$\begin{aligned}&MS_\mu(P), MS_\mu(P \cup R(i+1,j;a,1)), MS_\mu(P \cap R^c(i+1,j;a,1)),\\&c_0(P \cup R(i+1,j;a,1)), c_1(P \cup R(i+1,j;a,1)),\\&c_0(P \cap R^c(i+1,j;a,1)), c_1(P \cap R^c(i+1,j;a,1)).\end{aligned}$$

Our observation is:

All these quantities can be computed from their analogues at the previous pixel visit at the expense of only a fixed number of operations (independent of a and N). In other words, all the quantities required for energy comparison can be updated locally.

It is very easy to demonstrate this fact. The subsets of the grid involved in the $(i+1,j)$-th pixel visit are P, $P \cup R(i+1,j;a,1)$, and $P \cap R^c(i+1,j;a,b)$. They differ from the partitions involved in the previous pixel visit by only a few points. Indeed,

$$\begin{aligned}&\Big(P \cup R(i+1,j;a,b)\Big) \triangle \Big(P \cup R(i,j;a,b)\Big) \subset \Big\{(i,j),(i+a,j)\Big\}, \text{ and}\\&\Big(P \cap R^c(i+1,j;a,b)\Big) \triangle \Big(P \cap R^c(i,j;a,b)\Big) \subset \Big\{(i,j),(i+a,j)\Big\}.\end{aligned}$$

Hence, the energies and the constants required at the $(i+1,j)$-th pixel visit can be computed from their values at (i,j)-th pixel visit by a local calculation that involves only the immediate neighbors of the (i,j)-th and $(i+a,j)$-th pixels.

This observation can be easily generalized to updates with rectangles of size $a \times b$; if the appropriate pixel visit direction is chosen, the computational cost is seen to be at the advertised level. We summarize the discussion of this section:

- The standard SC algorithm can be generalized so that rectangular neighborhoods, instead of single pixels, are switched to one of the two phases at each step.

- This generalization helps eliminate some of the spurious stationary states of the original algorithm.
- There is an efficient way to implement the generalized algorithm so that the cost of a sweep through all pixels using $a \times b$ rectangular neighborhoods is only $2\min\{a, b\}$ times more costly than a sweep of the original algorithm.

7 Numerical results

In this section we illustrate on real and synthetic images how the multiscale version of Song and Chan's algorithm leads to better segmentations. Our emphasis is on showing how efficient multilevel procedures can be designed with the help of the observations in the previous two sections. Hence we experiment with several possible techniques without necessarily committing ourselves to a specific one of them.

We first note the the synthetic images shown in *Figures 1* and *2* that constitute spurious stationary states of the original SC algorithm are de-noised effectively, provided that we use updates with rectangular neighborhoods of size 2×1 or larger.

Figures 3 and 4 show results of experiments using the image of a galaxy at 512×512 resolution ($N = 512$). Regularization parameter was $\mu = \frac{512}{5000}$. Images of this type pose a particular challenge to segmentation algorithms, as they do not contain well defined edges and regions. As a result, a good segmentation relies heavily on the regularization term (i.e. the length of the interface). This is therefore the type of image for which the original version of the SC algorithm can reach a premature stationary state. The main purpose of this experiment is to show how the generalized version of the algorithm presented in this paper can avoid these stationary states, and reach significantly cleaner segmentations while maintaining low computational cost.

The upper left hand picture in Figure 3 shows the original galaxy image to be segmented. The upper right hand binary image shows the two phases found by the original version of the SC algorithm. The corresponding energy value is a relatively high $43.22N\mu$. Moreover, the algorithm requires 93 iterations to reach this stationary state. The lower left hand binary image shows what happens if we take the stationary state found by the standard algorithm and make it the initial guess for the generalized version that uses 32×1 rectangular neighbors and traverses the pixels in a row-wise direction (so that the cost of a single sweep is only twice that of the original algorithm).

The image shows the result after only one pass: the energy is down to $35.7N\mu$, and the boundary in visibly improved. Of course, just as a rectangle of size 32×1 can be used to do updates at only twice the cost of an update of the original algorithm if pixels are visited in the row-wise direction, a rectangle of size 1×32 can be used at only twice the cost of the original algorithm if the pixels are visited in the column-wise direction. The bottom right hand image of Figure 3 shows what happens when the bottom left image of Figure 3 is taken as initial guess for a single sweep using 1×32 rectangles: The energy is further decreased to $32.78N\mu$, and we have yet a cleaner segmentation.

This process can now be repeated: the image is alternately traversed in row-wise and column-wise fashion, using 16×1 and 1×16 rectangles, respectively. After a few steps, one arrives at the upper left image of Figure 4. Its energy is $31.54N\mu$. The remaining images in Figure 4 show the result of continuing this process using smaller rectangles (8×1 and 1×8, 4×1 and 1×4, etc.) and making a few sweeps at a time. After a few steps the lower right hand side image of Figure 4 is reached. Its energy is about $31.35N\mu$.

In the experiment of Figures 3 and 4, we waited until the original SC algorithm reached its stationary state before commencing with sweeps using larger neighborhoods. However, as we have seen, the SC algorithm can require quite a lot of iterations to reach a stationary state (93 in that example). In the following experiments, only a few steps of the original SC algorithm are taken to form an initial, very fine segmentation. Then, this fine segmentation is pruned at various scales by making sweeps with neighborhoods of various sizes. Also, in the experiments so far only neighborhoods of size $a \times 1$ or $1 \times a$ were used; this made it possible to sweep through the image at only twice the cost of the original algorithm. Further experiments suggest, on the other hand, that sweeps using $a \times a$ neighborhoods, although $2a$-fold more expensive than sweeps of the original algorithm, can lead to very rapid energy decrease. These matters are explored in the remaining numerical examples.

The left hand image of Figure 5 shows a 256×256 original image to be segmented (i.e. $N = 256$). The right hand image is the result of applying 3 iterations of standard K-Means algorithm. This was taken as initial guess for the experiments presented in Figures 5, 6, 7, 8, and 9.

The graph in Figure 6 shows how the energy decreases, starting from the initial guess shown in Figure 5 and using sweeps at different scales, as a function of computational cost used. Each unit of the horizontal axis represents the cost of a single sweep through the pixels using the standard SC algorithm (which is proportional to the number of pixels). The graphs were plotted to reflect the fact that a single sweep using $a \times a$ neighborhoods is in principle only about $2a$ times more costly than a a single sweep using 1×1 neighborhoods in terms of the number of floating point operations involved; this is why the points on graphs corresponding to larger neighborhoods are spaced out. The graph obtained using 16×16 neighborhoods turned out to lie strictly above the graph of 8×8 neighborhoods. The result clearly shows that the prudent choice is to use 8×8 neighborhoods. Also shown by the result is that at the 8×8 level, sweeps beyond the first two are essentially redundant: they accomplish very little in the way of energy decrease. The regularization parameter used in this experiment was $\mu = \frac{256}{200}$.

Figure 7 shows the stationary states reached in the experiment that led to the graph of Figure 6. The top left image is the stationary state for the original algorithm (1×1 updates), the top right for 2×2 updates, the bottom left for 4×4 updates, and bottom right for 8×8 updates.

Naturally, the optimal neighborhood size turns out to be related to the regularization parameter μ, which sets the scale for the model. When the last experiment is repeated with $\mu = \frac{256}{800}$, the energy vs. computational cost graph is shown in Figure 8. The prudent choice for neighborhood size turned out to be 4; the graph for 8×8 and larger updates were strictly above that of 4×4 updates. The stationary states corresponding to the various updates are shown in Figure 9.

Of course, we cannot expect to know the optimal neighborhood size. One natural approach is then to start with the very fine initial segmentation obtained by a few steps of the K-means algorithm and/or the original SC algorithm, and then prune it with neighborhoods of various sizes in a coarse to fine fashion. For example, we can start with a few sweeps using 16×16 rectangles, followed by a few sweeps using 8×8 rectangles, etc. Figures 10 through 14 report results of experiments using this procedure. During the coarse to fine pruning process, only one sweep at each scale was found to be quite sufficient.

Figure 10 shows the result of applying the coarse to fine pruning procedure outlined above to the original image shown in Figure 5. As in the experiments of Figure 7, the regularization parameter was taken to be $\mu = \frac{256}{200}$. The energy reached is $6.75N\mu$. Figure 11 shows the energy vs. computational cost graph for this experiment. It can be seen that the procedure is essentially complete by the time the equivalent of 38 sweeps of the original SC algorithm have been made. These results should be compared with those of Figures 6 and 7.

Finally, Figures 12 and 13 show a slightly simpler segmentation scenario. Were it not for the moderate amount of noise, the original image to be segmented (shown in the upper left hand side of Figure 12) – which has a resolution of 512×512 (i.e. $N = 512$) – would be well approximated by a piecewise constant image. The upper right hand figure shows the result found by the K-Means procedure, which was started from a random binary image as initial guess. Regularization parameter was taken to be $\mu = \frac{512}{3000}$. The lower left hand figure shows the result found by the original SC algorithm, which was also started from the same random binary image. It reached the stationary state shown in 35 iterations, yielding an energy value of about $205.78 N\mu$. The lower right hand side image shows the segmentation obtained via the multilevel procedure that was used in the experiment of Figure 10; the energy found is $157.45 N\mu$.

Figure 13 shows the energy vs. computational cost graph for the multilevel procedure that produced the segmentation shown in the lower right hand side of Figure 12. Figure 14 shows the segmentation at two intermediate stages of the multilevel procedure. The right hand side figure makes it clear that at a computational cost equivalent to making 22 sweeps with the original SC algorithm, a successful segmentation can be obtained.

The multilevel procedure explained above and illustrated with the numerical examples require a number of choices to be made. For instance, the coarsest scale that the algorithm will operate at, and the number of sweeps to make at each scale need to be chosen. There are also a number of reasonable ways to come up with a very fine segmentation as an initial guess. Depending on how these choices are made, the algorithm can produce slightly different final results. This is a common feature of many segmentation algorithms. For instance, the algorithm of [3] can produce slightly different results depending on the particular choice of schedule for the regularization parameter.

8 Conclusion:

We identified a challenging class of images for the algorithm of Song and Chan and showed how it sometimes reaches a steady state prematurely. By allowing the algorithm to operate at more scales than that of a single pixel, we showed how some of the premature steady states can be avoided. Moreover, we showed how all this can be done at more or less the same computational cost as the original algorithm.

There are many ways in which our study can be furthered. First and foremost, it is necessary to understand at a more precise and quantitative level how the computational cost of these algorithms compare to, say, the

Chan-Vese technique. A particularly interesting and important question in this context is how the computational cost of the present algorithm scales with respect to resolution of the given image.

References

[1] Chan, T. F.; Vese, L. A. *Active contours without edges.* IEEE Transactions on Image Processing. **10** (2), Feb. 2001, pp. 266 – 277.

[2] Gibou, F.; Fedkiw, R. *Fast hybrid k-means level set algorithm for segmentation.* Preprint, in review. (URL=http://math.stanford.edu/˜fgibou)

[3] Koepfler, G.; Lopez, C.; Morel, J.-M. *A multiscale algorithm for image segmentation by variational methods.* SIAM J. Numer. Anal. **31** (1994), no. 1, 282-299.

[4] MacQueen, J. *Some methods for classification and analysis of multivariate observations.* In Le Cam, L. M. and Neyman, J., editors. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability.* Vol. 1, pp. 281-297.

[5] Mumford, D.; Shah, J. *Optimal approximations by piecewise smooth functions and associated variational problems.* Comm. Pure Appl. Math. **42** (1989), no. 5, pp. 577-685.

[6] Osher, S. J.; Fedkiw, R. *Level set methods and dynamic implicit surfaces.* Appplied Mathematical Sciences 153, Springer Verlag. New York, 2003.

[7] Osher, S. J.; Sethian, J. A. *Fronts propagating with curvature-dependent speed; algorithms based on Hamilton-Jacobi formulations.* J. Comput. Phys. **79** (1988), no. 1, pp. 12-49.

[8] Rommelse, J. R.; Lin, H.-X.; Chan, T. F. *A robust level set algorithm for image segmentation and its parallel implementation.* UCLA CAM Report 03-05, February 2003.

[9] Selim, S. Z.; Ismail, M. A. *K-means-type algorithms: a generalized convergence theorem and characterization of local optimality.* IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol 6, no. 1, pp. 81-86.

[10] Sharon, E.; Brandt, A.; Basri, R. *Fast multiscale image segmentation.* Proceedings IEEE Conference on Computer Vision and Pattern Recognition. 1:70-77, South Carolina, 2000.

[11] Song, B.; Chan, T. F. *A fast algorithm for level set based optimization.* UCLA CAM Report 02-68, December 2003. Under revision for publication in SIAM J. Sci. Comput.

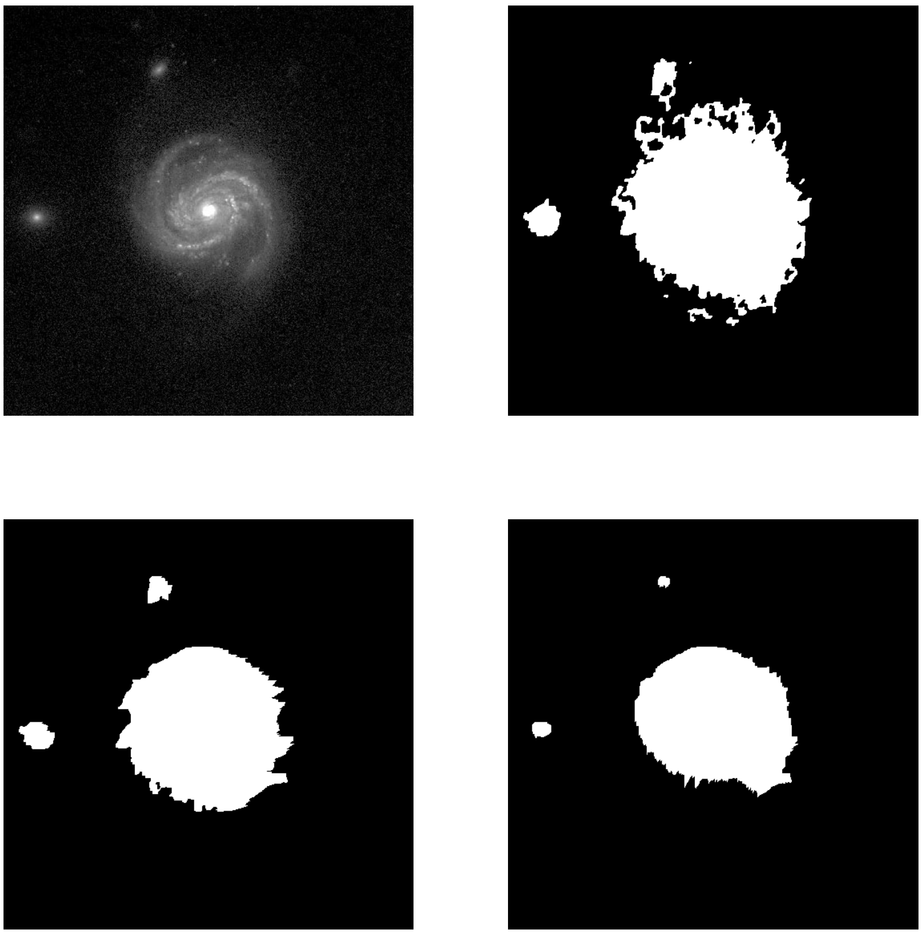

Figure 3: Results of the standard and enhanced algorithm compared using the 512×512 image of a galaxy, shown in the upper left hand figure, as the original image. The regularization parameter used was $\mu = \frac{512}{5000}$. The upper right hand figure shows the result found by the standard algorithm, which has an energy of $43.22N\mu$. The lower left hand figure is the result of starting with the upper right hand figure as initial guess and making one sweep, in the horizontal direction, using a rectangular neighborhood of size 32 by 1; this decreases the energy to $35.70N\mu$. The lower right hand figure shows the stationary state reached by making a further sweep, this time in the vertical direction, using a 1×32 neighborhood. That brings the energy to $32.78N\mu$.

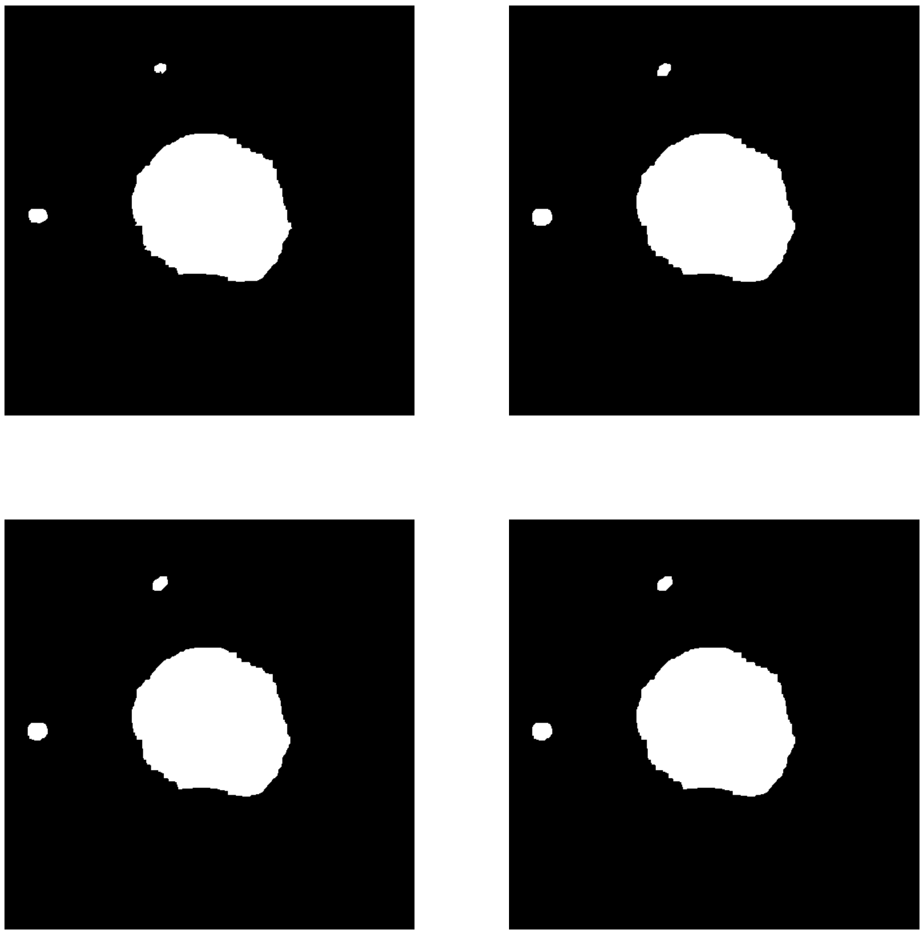

Figure 4: Alternating 32×1 and 1×32 updates in the horizontal and vertical directions, respectively, for a few more times gives the result shown in the upper left figure. Continuing, we make a few sweeps with 16×1 and 1×16 neighborhoods (result shown in upper right figure), and then with 8×1 and 1×8 neighborhoods, etc. The final result of this procedure is shown in the lower right hand figure, and has energy $31.35N\mu$.

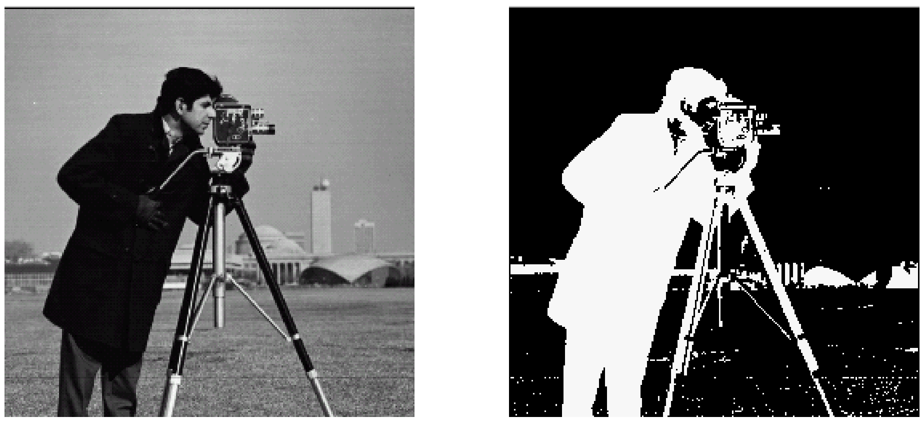

Figure 5: Original cameraman image, and the initial guess for two-phase segmentation obtained via standard sequential K-Means procedure (without perimeter regularization term).

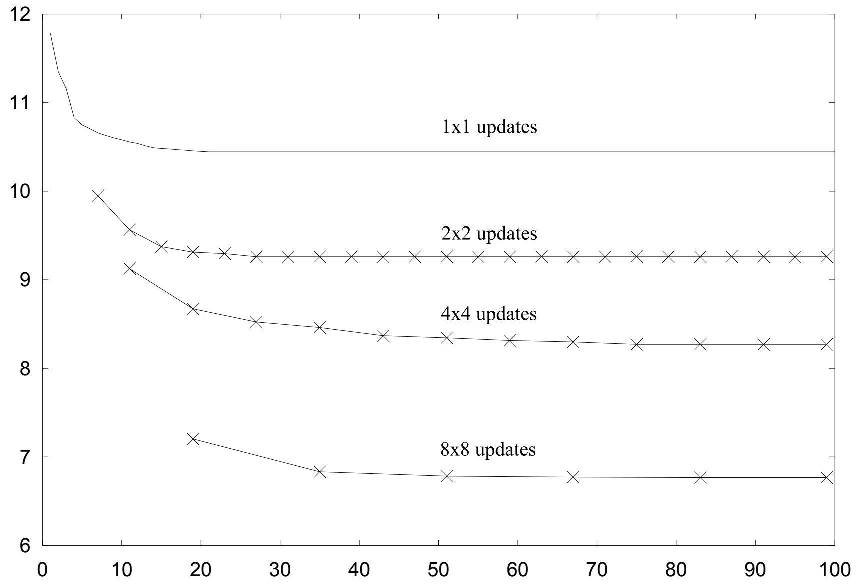

Figure 6: Plot of Energy vs. computational effort involved. The initial guess had an energy of about $19.2N\mu$ (not plotted). Regularization parameter was $\mu = \frac{256}{200}$.

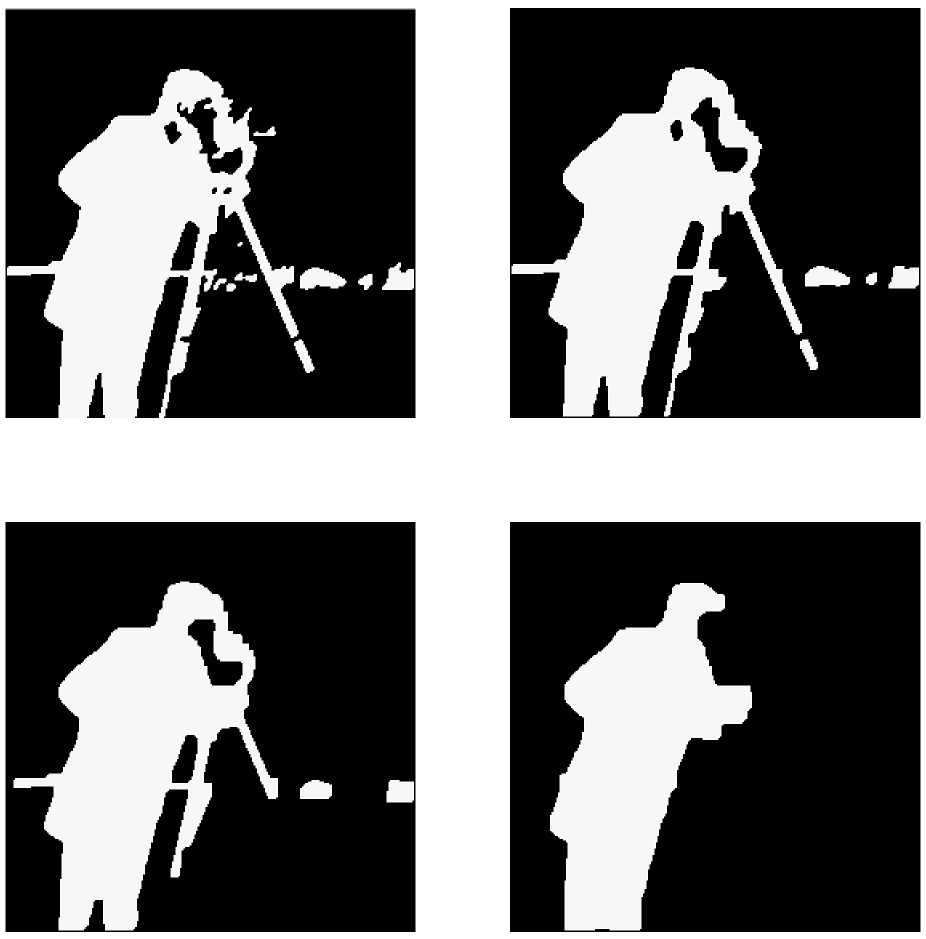

Figure 7: The stationary states for 1×1, 2×2, 4×4, and 8×8 updates. The corresponding energies reached were about $10.44N\mu$, $9.26N\mu$, $8.27N\mu$, and $6.77N\mu$ respectively. It took 21, 6, 8, and 6 sweeps through the pixels to reach them, respectively. Regularization parameter used was $\mu = \frac{256}{200}$.

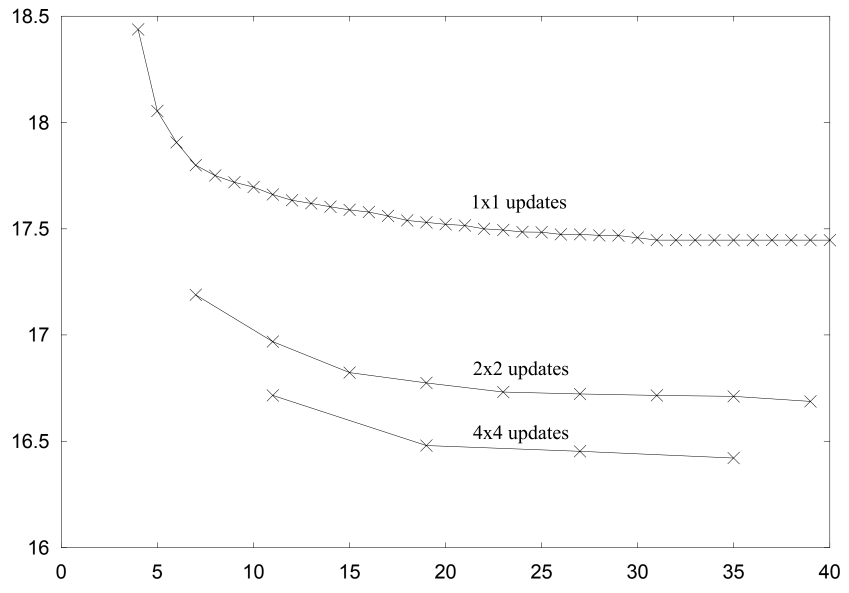

Figure 8: Plot of Energy vs. computational effort involved. Regularization parameter was $\mu = \frac{256}{800}$.

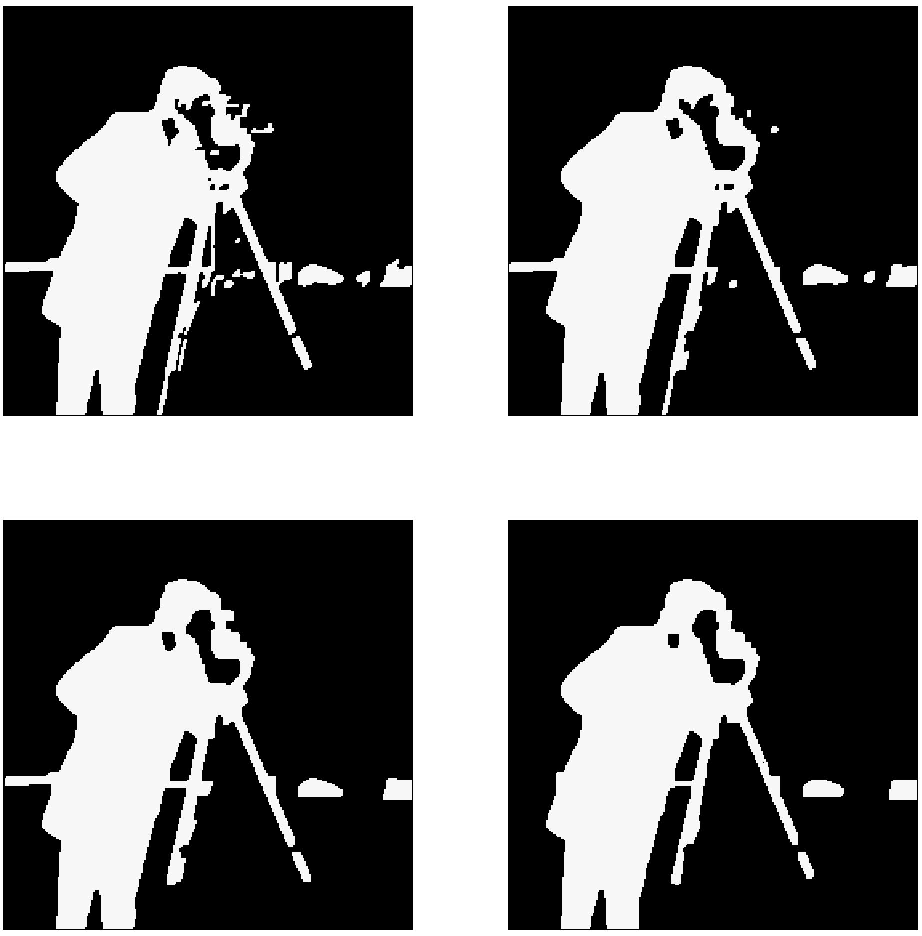

Figure 9: The stationary states for 1×1, 2×2, 4×4, and 8×8 updates for $\mu = \frac{256}{800}$. The corresponding energies reached were about $17.45 N\mu$, $16.69 N\mu$, $16.42 N\mu$, and $16.46 N\mu$, respectively.

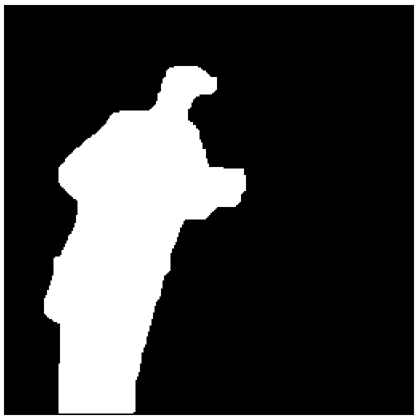

Figure 10: Segmentation obtained after a multilevel procedure to minimize energy (1) with $\mu = \frac{256}{200}$. A very fine segmentation is obtained as initial guess by using a few steps of the original SC algorithm. In this example, we took 3 steps with the algorithm using $\mu = 0$ (which makes these 3 steps equivalent to the K-means procedure), followed by 3 more sweeps using $\mu = \frac{256}{200}$. Then, this very fine segmentation is pruned at various scales using the generalized algorithm described in Section 6, by sweeping the image *once* with each of 16×16, 8×8, 4×4, 2×2, and 1×1 neighborhoods, in that order.

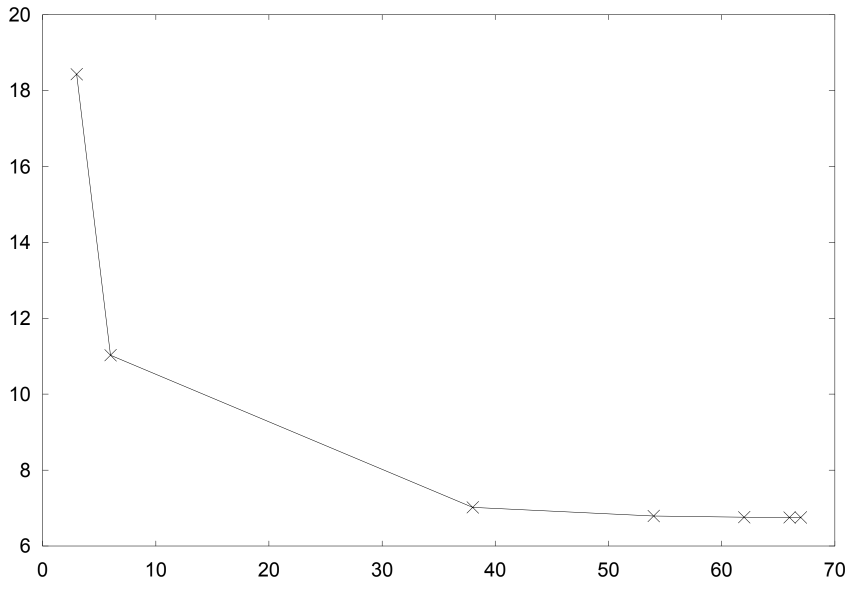

Figure 11: Plot of Energy vs. computational effort involved during the multilevel procedure. Regularization parameter was $\mu = \frac{256}{200}$.

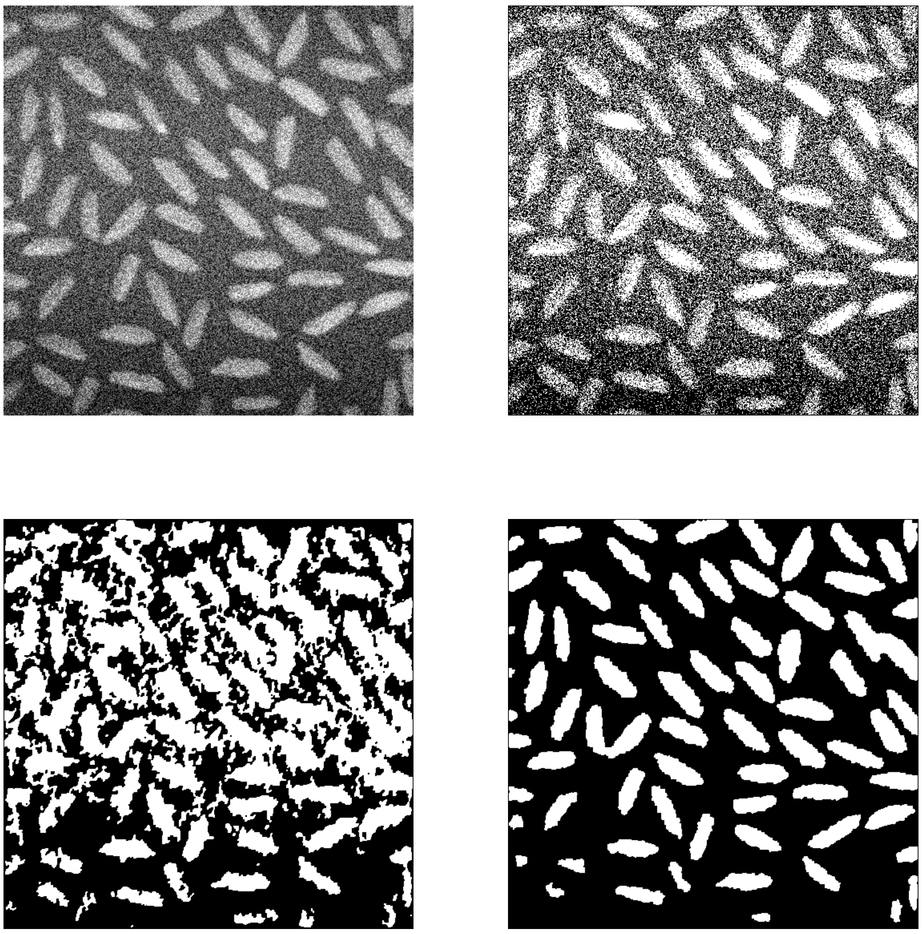

Figure 12: A moderately noisy image to be segmented. The upper left figure is the original image. The upper right hand figure shows the result of K-Means procedure ($\mu = 0$ case of the SC algorithm) after 3 sweeps. Bottom left image shows the result obtained by the original Song-Chan algorithm with $\mu = \frac{512}{3000}$, which was started from a random binary image as initial guess; the stationary state shown was reached after 35 sweeps through the image and has an energy of about $205.78N\mu$. The bottom right image show the result of the proposed multilevel procedure using the generalized algorithm. The energy obtained is $157.45N\mu$. The computational cost incurred is equivalent to about 35 sweeps of the original algorithm.

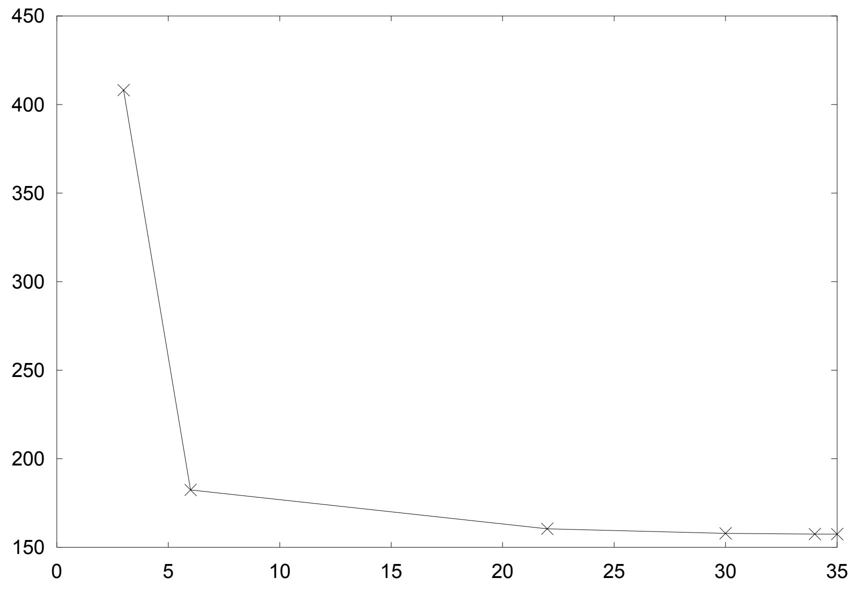

Figure 13: Plot of Energy vs. computational effort involved on the example of previous figure.

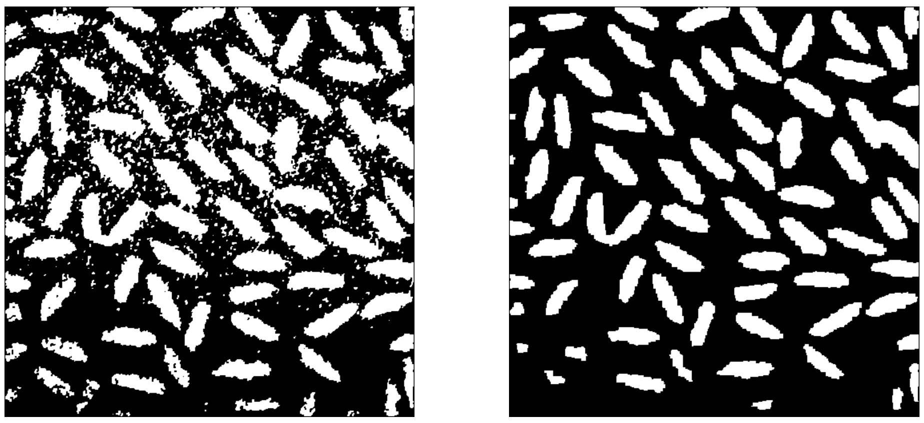

Figure 14: Intermediate stages of the multilevel procedure. Computational cost of reaching the left hand side image is about 6 sweeps of the original algorithm; it's energy is $182.34N\mu$. Computational cost of reaching the right hand side image is about 22 sweeps of the original algorithm; it's energy is $160.34N\mu$.