Chen, A. et al.. (0000) "Efficient Boundary Tracking Through Sampling,"
International Mathematics Research Notices, Vol. 0000, Article ID rnn000, 21 pages.
doi:10.1093/imrn/rnn000

Efficient Boundary Tracking Through Sampling

Alex Chen[1], **Todd Wittman**[1], **Alexander Tartakovsky**[2], **and Andrea Bertozzi**[1]

[1]Department of Mathematics, University of California, Los Angeles, CA 90059, U.S.A. and [2]Center for Applied Mathematical Sciences, University of Southern California, Los Angeles, CA 90090-1113 U.S.A.

Correspondence to be sent to: Correspondence to be sent to Alex Chen, 520 Portola Plaza Math Sciences Building 6363, Mailcode: 155505, Department of Mathematics, University of California, Los Angeles, CA 90059, U.S.A. e-mail: achen@math.ucla.edu

The proposed algorithm for image segmentation is inspired by an algorithm for autonomous environmental boundary tracking. The algorithm relies on a tracker that traverses a boundary between regions in a sinusoidal-like path. Boundary tracking is done by efficiently sampling points, resulting in a significant savings in computation time. Page's cumulative sum (CUSUM) procedure and other methods are adapted to handle a high level of noise. Applications to large data sets such as hyperspectral are of particular interest. Irregularly shaped boundaries such as fractals are also treated at very fine detail.

1 Introduction

Interpreting important features in an image often involves some simplification that discards detail in favor of generality. In image segmentation, two approaches are particularly common: region-based methods and edge-based methods. Partitioning an image into homogeneous regions gives a "cartoon-like" appearance which identifies the most important objects and their rough boundaries. Many algorithms using this approach rely on identifying some feature common to each region and classifying individual pixels according to how well they match this feature. The classification is often based on matching pixel intensities with some spatial bias, such as regularity of object boundaries. This idea of image segmentation is the method adopted in region-based methods like the algorithm of Chan and Vese [15], using the level set method [49] to minimize the piecewise constant Mumford-Shah functional [47]. In many of these methods, changes in topology are also allowed, so that multiple objects may be detected.

Another approach is to consider the statistical patterns observed in real data. The seminal work of Geman and Geman [27] describes attributes such as pixel intensity and image edges as a Markov Random Field. This model places images in a Bayesian framework, in which features can be described based on similar features found in the same image or in other data sets that share the same characteristics. Methods such as region competition rely on this statistical model and are often solved using energy minimization [68, 62]. Sometimes the segmentation can be successively refined with further user interaction [53].

These approaches are generally less sensitive to noise, due to the spatial regularity requirement and the use of information from many pixels in the image. This robustness, however, comes at the expense of computation time, since calculations often must be done on every pixel, even if some pixels are far from object boundaries or clearly belong to a certain region.

Recent advances have greatly improved the computational efficiency of segmentation methods based on the Mumford-Shah functional using more efficient methods for energy minimization [28, 21, 24, 25]. Another possibility is to make calculations only in a narrow band around the zero level set, reducing the need to make irrelevant calculations [51].

Alternatively, tracking edges or boundaries of objects gives the location at which significant changes in the image take place. Local searching algorithms such as image snakes and gradient vector flow [38, 66] are designed to work faster than region-based methods. Note that since curves are evolved near the boundaries of objects, it is only necessary to consider pixels near these boundaries.

In contrast to the region-based methods, noise is often a problem with such local searching methods since far fewer pixels are used in calculations. Moreover, such methods often use an edge-detection function

based on a gradient. While the gradient is intended to locate edges, it has the side-effect of making the edge-detector highly sensitive to noise.

Furthermore, certain shapes such as those with large concavities or sharp corners may not be detected accurately due to requirements in the regularity of the curve. These points can be somewhat resolved with a balloon term [19] designed to encourage evolution past local minima of the underlying energy functional or replacing the second order PDE for snakes evolution with a fourth order PDE designed to preserve corners [63, 12]. In [10], a new energy was proposed to replace length with local area. This allows irregularly shaped objects, especially those with fractal-like structure, to be segmented more accurately. Other edge-based methods include the use of random walks to test hypotheses for the most likely layout of edges [4] and the Canny edge detector [13].

Theoretically, the use of fewer pixels in local searching gives these methods a shorter run time than with region-based methods. These algorithms scale only with the number of pixels in the boundary, rather than the number of pixels in the image. This speed advantage is thus even more pronounced for large images.

In this work, we propose a novel segmentation method that seeks to minimize the number of calculations. By sampling points, the boundary of an object can be tracked without the need to process a large number of pixels. The method is based on an algorithm for tracking environmental boundaries [39, 36] which utilizes robots that walk in a sinusoidal path along the boundary between two regions. The robots change directions as they cross from one region into another. The robot walking method is related to the class of "bug" algorithms that seek a target while avoiding obstacles [18, 48]. Such algorithms utilize only local information and theoretically should not visit the same location twice. Only the points that are near the boundary in question are tracked, resulting in substantial savings in run-time.

Sinusoidal tracking patterns have been observed in a variety of contexts as an efficient path for travel. Ants walk in a sinusoidal manner along pheromone trails laid down by other ants, predicting the proper direction of travel and compensating if they stray too far in a certain direction [32, 20]. For atomic force microscopy, using a sinusoidal pattern for scanning can avoid imaging points that are irrelevant to the region of interest [16, 3].

Some "walking" methods for boundary tracking exist in image processing as well. Moore's algorithm [29] tracks an edge of a binary image using an ordered search for image intensity values of 1 (alternatively, 0) through a neighborhood of each pixel in the iteration. Another method follows the maximum gradient magnitude of the 3×3 neighborhood of each pixel [14]. These methods, however, clearly fail in the presence of any noise.

As with local searching methods in image processing, noise can cause problems since the tracking is done as a local search. It was proposed [36] that the use of a change-point detection algorithm such as Page's cumulative sum (CUSUM) procedure [50] would allow objects to be tracked in noise. Testbed implementations of the boundary tracking algorithm suggest that robots can indeed track boundaries efficiently in the presence of a moderate amount of sensor noise [37].

One of the greatest advantages for using the algorithm for segmentation is the computational efficiency. By considering only pixels near the boundary of an object, as with local searching methods, many pixels are not considered. Furthermore, the tracking method travels through each location only once, resulting in run-time savings even over other local searching methods. As mentioned earlier, noise can be particularly troublesome when fewer points are considered. Further improvements can be made that are not practical in the environmental tracking case. Many of these improvements are based on hypothesis testing for two regions, with the use of the CUSUM algorithm as a special case. Hypothesis testing is an important part of many statistical segmentation methods [9, 52, 26].

The boundary tracking algorithm in the context of image processing was briefly introduced in [17]. In this work, we give a more detailed treatment with expanded theory and applications. In subsequent sections, we will adapt the boundary tracking algorithm to the image processing problem and discuss improvements that can be made in this context. Section 2 briefly reviews the boundary tracking algorithm and compares the original environmental tracking and image processing problems. In Section 3, various methods for better performance in the presence of noise are suggested. Other improvements to the algorithm are discussed in Section 4. Section 5 shows numerical results and images, including examples with hyperspectral and high resolution data. Finally, discussion and issues for further study are given in Section 6.

2 A two-step locating and tracking method

The boundary tracking algorithm is first initialized with some given decision function, which determines whether it is in one of two regions. Boundary tracking is done as a two-step process using the decision function in both steps. The first step involves locating a boundary point via a global search. The boundary

point found by this global search serves as an initial point to be used in the local sampling step. At this second step, boundary points are found by using a tracker that travels near the boundary in a sinusoidal path. Sometimes the tracker can move off of the boundary, particularly when the noise is high or when the decision function is not very accurate. When this happens, it is necessary to use the global search step again to locate the boundary. This paper discusses the global search step briefly, and focusses on the local sampling algorithm.

2.1 Global search to locate a boundary point

There are several options for the global search step. The simplest is just a user-defined point near the boundary that can be directly used by the local sampling step. For a more automated approach, a random initial point in the image can be given. Then the tracker travels in a spiral away from the initial point until a boundary point is detected (see Fig. 1). It is sometimes necessary to refine this estimate, since the spiral pattern may not detect the exact crossing point accurately. Another option is to use a Levy flight to detect a boundary. Levy flights have been observed in a variety of contexts and have certain optimality properties [41, 64].

A different approach for global search is to use a hybrid method. Boundary location can be done by using a coarse segmentation by another method. The resulting segmentation gives initial boundary points to be used in the local sampling step. This method will be explained further in Section 5. For the tracking of one object, however, global detection is not usually the limiting step in accuracy or in run-time. Thus, in this work we focus on the local sampling step.

2.2 Local sampling to track the boundary

For the physical problem [36, 39], a robot is used to track an environmental boundary. The robot is placed near the boundary in question, and it then uses a bang-bang steering controller to move through the boundary of the two regions.

It is relatively straightforward to adapt the algorithm for images. Let the image domain be represented by Ω, with B the boundary between two regions Ω_1 and Ω_2, so that $\Omega = \Omega_1 \cup \Omega_2 \cup B$ and $\Omega_1 \cap \Omega_2 = \emptyset$. Define an initial starting point $\vec{x}_0 = (x_0^1, x_0^2)$ for the boundary tracker and an initial value θ_0, representing the angle from the $+x^1$ direction, so that the initial direction vector is $(\cos\theta_0, \sin\theta_0)$. Also define the step size V and angular increment ω, which depend on estimates for image resolution and characteristics of the boundary to be detected. In general, V is chosen smaller for greater detail, and ω is chosen smaller for straighter boundaries. A decision function between Ω_1 and Ω_2 must also be specified and has the following form:

$$d(\vec{x}) = \begin{cases} 1, & \text{if } \vec{x} \,\epsilon\, \Omega_1, \\ 0, & \text{if } \vec{x} \,\epsilon\, B, \\ -1, & \text{if } \vec{x} \,\epsilon\, \Omega_2. \end{cases} \tag{1}$$

The simplest example is thresholding of the image intensity $I(\vec{x})$ at a given spatial location $\vec{x}$ (in the case of a grayscale image):

$$d(\vec{x}) = \begin{cases} 1, & \text{if } I(\vec{x}) > T, \\ 0, & \text{if } I(\vec{x}) = T, \\ -1, & \text{if } I(\vec{x}) < T, \end{cases} \tag{2}$$

where T is a fixed threshold value. Later in this section we use statistical information about prior points sampled along the path to modify $d(\vec{x})$. At each step k, the direction θ_k and current location $\vec{x}_k$ are updated recursively. Specifically,

$$\vec{x}_k = \vec{x}_{k-1} + V * (\cos\theta_{k-1}, \sin\theta_{k-1}), \tag{3}$$

and θ_k is updated according to the new location of the tracker $\vec{x}_k$. A simple update for θ is the bang-bang steering controller, defined by

$$\theta_k = \theta_{k-1} + \omega d(\vec{x}_k). \tag{4}$$

Midpoints of the tracker locations the iteration before and the iteration after change are taken to be the boundary points. Linear interpolation can be used to obtain a boundary curve. Note that unlike many other segmentation methods, it is not assumed that the curve is smooth. This allows the detection of very irregular objects, especially those with large concave regions or high curvature boundaries.

An angle-correction modification [36] can be used for (4) if step k is a region crossing:

$$\theta_k = \theta_{k-1} + d(\vec{x}_k)(\bar{t}\omega - 2\theta_{ref})/2, \tag{5}$$

where $\bar{t}$ is the number of steps since the last region crossing, and θ_{ref} is a small fixed reference angle. The parameter θ_{ref} is determined empirically with a value of 0.1 giving good results in many images. Intuitively, θ_{ref} makes angle correction more conservative so that the direction of travel is not parallel to the expected direction of the boundary. This results in some distance still being traversed before the next boundary crossing.

A further consideration is the stopping condition. Several options for the termination of the algorithm are possible: the tracker completes a certain number of iterations, arrives at the image border, or arrives near the first boundary point detected with some minimum number of iterations. The latter two were used for numerical experiments.

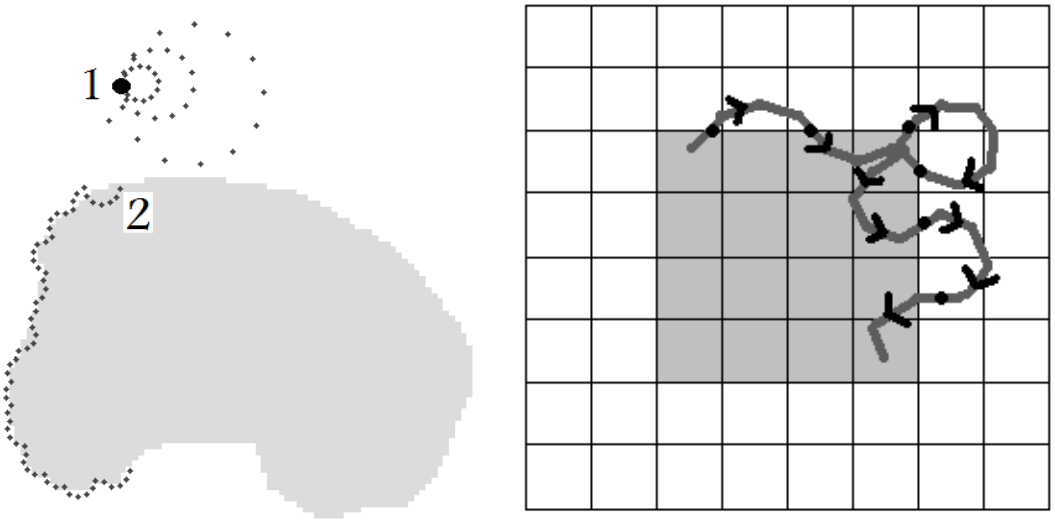

Fig. 1. Left: Global search is initialized at point 1, traveling in a spiral-like pattern. At point 2, a boundary point is found and local sampling starts, tracking the boundary more closely. Right: Basic procedure for the boundary tracking (local sampling) algorithm. Local sampling works on a sub-pixel level.

There are several differences between the environmental problem and image processing problem. The primary difference is the conversion from a continuous model to a discrete model. While a robot samples data wherever it travels, image data is pixelated so that the tracker cannot sample data at the sub-pixel level. Instead, the nearest neighbor intensity reading is used. Thus, the level of detail is limited by the image resolution. However, while sampling is only done at the pixel locations, the tracker still travels at the sub-pixel level, as indicated in Fig. 1. Moreover, the tracker can move with step sizes less than 1. The redundancy of sampling the same pixel more than once makes the algorithm more robust to noise and gives a sharper segmentation without increasing computation time much. Indeed, numerical experiments confirm that the step size giving optimal results is often less than 1.

Conversion to the image segmentation problem also has several advantages. A robot is restricted to sampling data only at its current location. In images, however, information from the pixels surrounding the tracker can be used to mitigate mistakes due to noise. Using these extra pixels, of course, negates some computational benefits, but as long as the number of extra pixels considered is small, the main advantages are retained. Various methods for using this nonlocal information while preserving the speed of the algorithm are presented in the next section. Also, robots used in experiments [33, 37] make smooth changes of direction, but this restriction is not required in the image processing case. Lastly, robots have a fixed position at any point in time and cannot make instantaneous jumps. In images, however, it is possible to place the tracker in another location in a single step. This ability allows the boundary tracker to correct mistakes detected at a later time without having to backtrack through many pixels.

3 Change-Point Decisions

The boundary tracking method presented works well when a clear and accurate decision function can be defined. However, in many applications a clear distinction between the two regions cannot be made, particularly in noisy images. For boundary tracking, errors in classification by the decision function can lead to serious errors in tracking, since the local sampling algorithm is only valid when the tracker is near

the boundary. With some method to average nearby pixels, however, the algorithm can be made much more robust to noise.

Change-point detection theory is well-suited to tracking image edges in noise. In particular, the CUSUM algorithm has been used to improve tracking performance in the environmental tracking problem [40, 37]. This section reviews relevant information from change-point detection and its application to the boundary tracking problem.

Change-point problems deal with rapid detection of abrupt changes in statistical properties (distributions) of data. One standard application of change-point detection is in manufacturing [45, 22, 55]. For a certain process, it may be acceptable to have a certain failure rate. If the process is able to operate below this tolerance level, then it is allowed to continue operation. If, however, this tolerance level is exceeded, one would like to stop as soon as possible to make repairs. Making a false stop, however, is costly as well, so it is important to balance the two considerations. Other applications include surveillance, computer network security (rapid detection of intrusions), failure detection in dynamical systems and communication networks, financial markets, seismology, navigation, speech segmentation, etc. See, e.g., [11, 60, 61, 65] and references therein.

More explicitly, given a sequence of independent observations $s_1 = I(x_1), \ldots, s_n = I(x_n)$ and two probability density functions (pdf) f (pre-change) and g (post-change), determine whether there exists N such that the pdf of s_i is f for $i < N$ and g for $i \geq N$.

One of the most efficient change-point detection methods is the CUSUM algorithm proposed by Page in 1954 [50]. Write $Z_k = \log[g(s_k)/f(s_k)]$ for the log-likelihood ratio and define recursively

$$U_k = \max\left(U_{k-1} + Z_k, 0\right),\ U_0 = 0 \tag{6}$$

the CUSUM statistic and the corresponding stopping time $\tau = \min\{k \mid U_k \geq \overline{U}\}$, where $\overline{U}$ is a threshold controlling the false alarm rate. Then τ is a time of raising an alarm. In our applications, assuming that f is the pdf of the data in Ω_1 and g is the pdf in Ω_2, the value of τ may be interpreted as an estimate of the actual change-point, i.e., the boundary crossing from Ω_1 to Ω_2.

Changes from Ω_2 to Ω_1 can also be tracked in this manner. Analogously to (6) define recursively the decision statistic $L_k = \max(L_{k-1} - Z_k, 0)$, $L_0 = 0$ and the stopping time $\tau = \min\{k \mid L_k \geq \overline{L}\}$, where $\overline{L}$ is a threshold associated with a given false detection rate. The description of the basic boundary tracking algorithm is now complete; the steps are shown in Fig. 2.

The CUSUM algorithm and optimality can be understood intuitively. Consider two distributions of data D_f and D_g with density functions f and g, respectively. In the change-point detection problem, observations are conditionally independent given the point of change. Given a sequence of observations $x_{k-l+1}, \ldots, x_k$ for a fixed natural number l (the change point), test the hypotheses

$$H_0 : x_{k-l+1}, \ldots, x_k \text{ are all from distribution } D_f,$$

$$H_1 : x_{k-l+1}, \ldots, x_k \text{ are all from distribution } D_g.$$

Then the likelihood ratio test gives the procedure $\frac{g(x_{k-l+1})\cdots g(x_k)}{f(x_{k-l+1})\cdots f(x_k)} \geq \alpha$.

This test, however, requires $k - l + 1$ observations after a change has actually occurred, since the hypotheses are that the observations all come from one distribution or the other. Instead, the number of observations l can be allowed to vary, which is beneficial for our applications where the point of change from D_f to D_g is not known in advance. In this case, a reasonable approach is to declare that a change is in effect if there exists some $l = 1, \ldots, k$ such that $\frac{g(x_l)\cdots g(x_k)}{f(x_l)\cdots f(x_k)} \geq \alpha$ for some $k \geq 1$. In other words, the change is declared when it is reasonably certain that some number of the most recent entries provides evidence that a change has occurred. The threshold α describes the confidence level for the change. This modification allows for the reduction of the number of false alarms due to noise while still allowing rapid detection in case of an actual change.

Taking the logarithm, this rule can be associated with the stopping time

$$\begin{aligned}\tau &= \min\{k \mid \sum_{i=l}^{k} \log \frac{g(x_i)}{f(x_i)} \geq \overline{U} \text{ for some } l = 1, \ldots, k\} \\ &= \min\{k \mid \max_{1 \leq l \leq k} \sum_{i=l}^{k} Z_i \geq \overline{U}\}\end{aligned}$$

where $\overline{U}$ is a certain threshold that controls the rate of false alarms and $Z_i = \frac{g(x_i)}{f(x_i)}$ is the log-likelihood ratio for the ith observation.

It is easily seen that the trajectories of the statistic $\max_{1\le l\le k}\sum_{i=l}^{k} Z_i$ coincide with the trajectories of the CUSUM statistic $U_k = \max_{l\ge 1}\sum_{i=l}^{k} Z_i$ on the positive half-plane. Note that the CUSUM statistic also obeys the recursion (6) (i.e., it is a reflected from the zero barrier random walk). Therefore, whenever the threshold $\overline{U}$ is positive (which is usually the case) the stopping time τ is nothing but the CUSUM algorithm that can be equivalently written as

$$\tau = \min\{k \mid U_k \ge \overline{U}\}$$

where U_k is the CUSUM statistic given in (6).

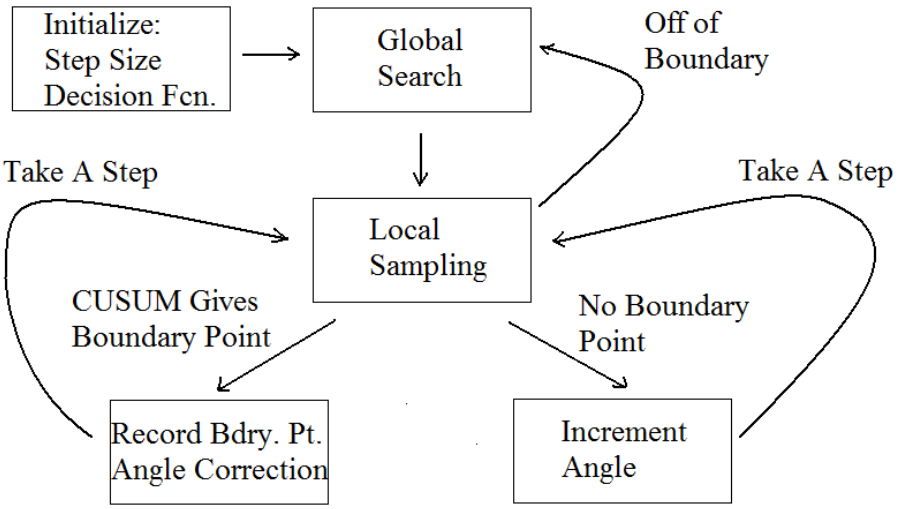

Fig. 2. Flowchart of the boundary tracking algorithm.

A few basic observations from (6) can be made. As mentioned earlier, the algorithm indicates a change from Ω_1 to Ω_2 only when observations generated by Ω_2 accumulate. Furthermore, an accumulation of many (negative) values from Ω_1 does not affect performance since the minimum value for U is 0. Lastly, the CUSUM procedure can be interpreted as a collection of one-sided sequential probability ratio tests (in this connection, see Lorden [42]) .

The pdf f, g for the two regions can be modeled in a variety of ways. The method is sufficiently general to allow of the use of discriminative or generative statistical models. One simple generative approach is to assume f and g are generated by Gaussians or a mixture of Gaussians, a common approach in image processing [30, 58, 35]. In the boundary tracking results for grayscale images, f and g have been assumed to follow Gaussian distributions.

In adapting change-point detection theory to region changes, the assumption of independent observations has been made. The samples are not independent, however, since they are taken from the tracking path. This assumption is not altogether problematic in practice, especially when noise is spatially uncorrelated. With a high level of noise, the spatially uncorrelated noise gives approximate independence. For a low level of noise, using change-point detection is less important.

As alluded to earlier in this section, the effectiveness of a change-point detection algorithm can be quantified. Generally, the important factors are the delay in detection of a real change (which is random) and false alarms. Detection delay measures the number of observations it takes to detect a change after it has actually occurred, while a false alarm occurs when a change is detected but has not actually occurred (Type 1 errors). In general, the detection delay should be small and the false alarm rate should be low. Clearly, the two goals are antagonistic. By increasing thresholds $\overline{U}, \overline{L}$ the false alarm rate can be lowered, but this will unavoidably lead to an increase in the detection delay. Thus, one has to adjust parameters in the change-point detection algorithm to moderate both indices.

Typically, operating characteristics of change-point detection algorithms are expressed via the average run length (ARL) to detection versus the ARL to false alarm. These performance indices have been first introduced by Page in [50]. The ARL is the average time to a change detection for a certain scenario. Under this notation, the false alarm rate is measured by the ARL when no change takes place (i.e., by the mean time to false detection $E_l(\tau|\tau < l)$), while the average detection delay is measured by the ARL when a change takes place immediately after observations begin, i.e., $E_0\tau$. The ARL is not the only method to measure

the effectiveness of change-point detection algorithms [11, 42, 59]. For example, why should we restrict our attention to the detection delay when the change occurs from the very beginning? The conditional average detection delay $E_l(\tau - l|\tau > l)$ for any fixed point of change $l > 0$ is a more reasonable measure. Also, in place of measuring the false alarm rate with the ARL to false alarm one may prefer to work with a probability of false alarm in a fixed time interval. See [59] for a more detailed discussion.

In 1971, Lorden [42] introduced the minimax (worst-worst case) performance measure – the essential supremum average detection delay $DD = \operatorname{ess\,sup} \sup_l E_l(\tau - l)^+ | x_1, \ldots, x_l)$ and proved that the CUSUM procedure is asymptotically optimal with respect to this detection delay measure for a low false alarm rate when the ARL to false alarm is large, among all procedures for which the ARL to false alarm is fixed at a given level. Later, Moustakides [46] proved that actually CUSUM is exactly optimal for any false alarm rate with respect to Lorden's criterion. Other optimality properties of CUSUM are discussed in [59]. For CUSUM, the following relation holds (asymptotically as ARL to false alarm is large):

$$DD \sim E_l(\tau - l|\tau > l) \sim \frac{\log ARLFA}{K(g, f)}, \quad l \geq 0,$$

where $ARLFA$ denotes the ARL to false alarm and

$$K(g, f) = E^g \left(\log \frac{g(x)}{f(x)} \right)$$

is the Kullback-Leibler information number [11]. It can also be shown that asymptotic detection delay is

$$DD \sim \frac{\overline{U}}{K(g, f)}.$$

This form of asymptotic detection delay is used to improve boundary tracking in the following section.

4 Further Improvements to the Boundary Tracking Algorithm

This section introduces further modifications to the algorithm, possible only in the image processing case, improving performance in the presence of noise. To mitigate the effects of noise further, a mean filter can be used; that is, the nearest neighbor intensity at the tracker can be replaced by an average of intensity values of nearby pixels. The idea is that nearby pixels are most likely to be in the same region, so that taking an average filters noise without using values from the other region. The same principle is at work with region-based segmentation methods, in which it is assumed that an image can be partitioned into a few contiguous regions with the same characteristics.

Mean filters can help the performance of the boundary tracker in noise. But since the tracker follows the boundary closely, it is important to use the uniform region assumption conservatively. Using a large number of pixels around the tracker would average noise but likely would also use values from the other region. Using these values can result in a much more serious error. In practice, a 3×3 window centered on the tracker works well. Instead of averaging the entries in the window, one can apply reasoning similar to the hypothesis testing mentioned for the CUSUM algorithm. As with observations along the boundary tracking path, the entries in the 3×3 window can also be treated as independent measurements. Thus, in the hypothesis testing of the CUSUM procedure, the nearest neighbor observation is replaced by the nine observations in the 3×3 window.

The replacement of an observation by a window is not standard in change-point detection theory, since it requires using extra data points not typically available. While the implementation may pose some difficulty in the environmental boundary tracking problem, there is no such problem for the corresponding image processing problem.

Besides noise, there can be other difficulties with the boundary tracker. Occasionally, the boundary tracker becomes stuck in a certain area due to ambiguities in the object boundaries or irregularity of its shape. To prevent this occurrence, the boundary tracker receives a "kick" if it has not left a certain window by a certain number of iterations. The kick is in the opposite direction to which the boundary tracker entered the window.

More explicitly, starting at a point $\vec{x}_i$, let M_1 be a constant representing the "window" size that the tracker must leave, and M_2 a constant representing the number of iterations before receiving a "kick." If $\|\vec{x}_i - \vec{x}_{i+j}\|_2 < M_1$ for each $j \leq M_2$, then the tracker location $\vec{x}_{i+M_2}$ is moved to $\vec{x}_{i+M_2} - 2(\vec{x}_{i+M_2} - \vec{x}_i)$, and the new starting point for the kicking algorithm becomes $\vec{x}_{i+M_2}$.

Theoretically, this direction is justified by assuming that the boundary tracker enters the kicking window due to chance noise effects and that a slightly different path can avoid problems. In experiments, the efficacy of kicking is confirmed in that after going off course, the tracker often finds its way back to the boundary after kicking. In the numerical experiments following, the tracker receives a kick in the opposite direction if it has stayed in the same 5×5 window for 200 iterations.

Another improvement is to use the theoretical average detection delay to backtrack along the path. In cases where the structure of noise is known or can be estimated, the average detection delay can often be calculated. Since this is the average time to detection after a change-point has actually occurred, backtracking along the path by this amount will give the correct change-point on average. This results in a more accurate detection of the actual boundary points.

For the grayscale numerical experiments, average detection delay was approximated by the asymptotic detection delay, $\frac{\overline{U}}{K(g,f)}$. For f generated by $N(0,\sigma^2)$, g generated by $N(1,\sigma^2)$,

$$\begin{aligned} K(g,f) &= E^g(\log \frac{g}{f}) \\ &= \int_{-\infty}^{\infty} g(y)\ log(\frac{g(y)}{f(y)})\ dy \\ &= \int_{-\infty}^{\infty} \left[\frac{y^2}{2\sigma^2} - \frac{(y-1)^2}{2\sigma^2}\right] g(y)\ dy \\ &= \int_{-\infty}^{\infty} \left[\frac{(x+1)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2}\right] f(x)\ dx \\ &= \frac{1}{2\sigma^2}. \end{aligned}$$

Thus, the asymptotic detection delay is $\overline{U} * 2\sigma^2$,. Backtracking by $\overline{U} * 2\sigma^2$ steps after each boundary crossing gives a slightly sharper tracking. The quantity $\overline{U} * 2\sigma^2$ is normally small enough (< 1) that there is little difference in the actual results. For high noise images, however, this backtracking has larger influence since both σ and $\overline{U}$ will be larger.

The statistical nature of the CUSUM algorithm can be used to detect inaccurate trackings. The idea is to observe whether the algorithm is making accurate tracking or detecting an object only because it is forced between the two classes. If the object is not being tracked accurately, the algorithm should switch from the local sampling algorithm back to the global searching step to find a boundary point. Once a new boundary point is found, local sampling can begin again.

Recalling the log-likelihood ratio Z_k for each point along the tracking path, tracking the statistic $C = \frac{1}{N}\sum_{k=1}^{N} |Z_k|$, where N is the total number of points in the path, gives a measurement of the confidence that the tracking is valid. If $C > \overline{T}$ for some constant threshold $\overline{T}$, then one can be reasonably sure that a large portion of data points are well-separated by the density functions f, g. In contrast for $C < \overline{T}$, the tracking is likely to be inaccurate as boundary points are classified mainly because they are forced into one class or the other. This is particularly a problem if the assumption of a two-region decomposition for Ω is false. The threshold $\overline{T}$ is determined empirically, and a value of $\overline{T} = 0.3$ typically works well.

5 Numerical and Image Examples

In Fig. 3, several results on a "U" image in heavy noise are shown. The results for boundary tracking are comparable to a region-based method. Furthermore, corners are not rounded as in the case with many energy-based segmentation methods. This fact can make boundary tracking more well-suited for segmentation problems with more man-made objects or with sharper edges. Parameters for each of the numerical examples were $\omega = 0.5, V = 0.5, \overline{U} = \overline{L} = 0.8$ unless otherwise indicated.

The segmentation accuracy of the boundary tracking algorithm is comparable to that of other segmentation methods. The run-time and storage costs, however, are much less for boundary tracking than for many global methods, especially for high dimensional data.

The algorithm compares favorably with other segmentation methods designed for fast computation time. The split Bregman globally convex segmentation (GCS) method of [28] is one such method. Table 1 shows a comparison of the two methods. Note that the boundary tracking method scales much better with an increase in image size.

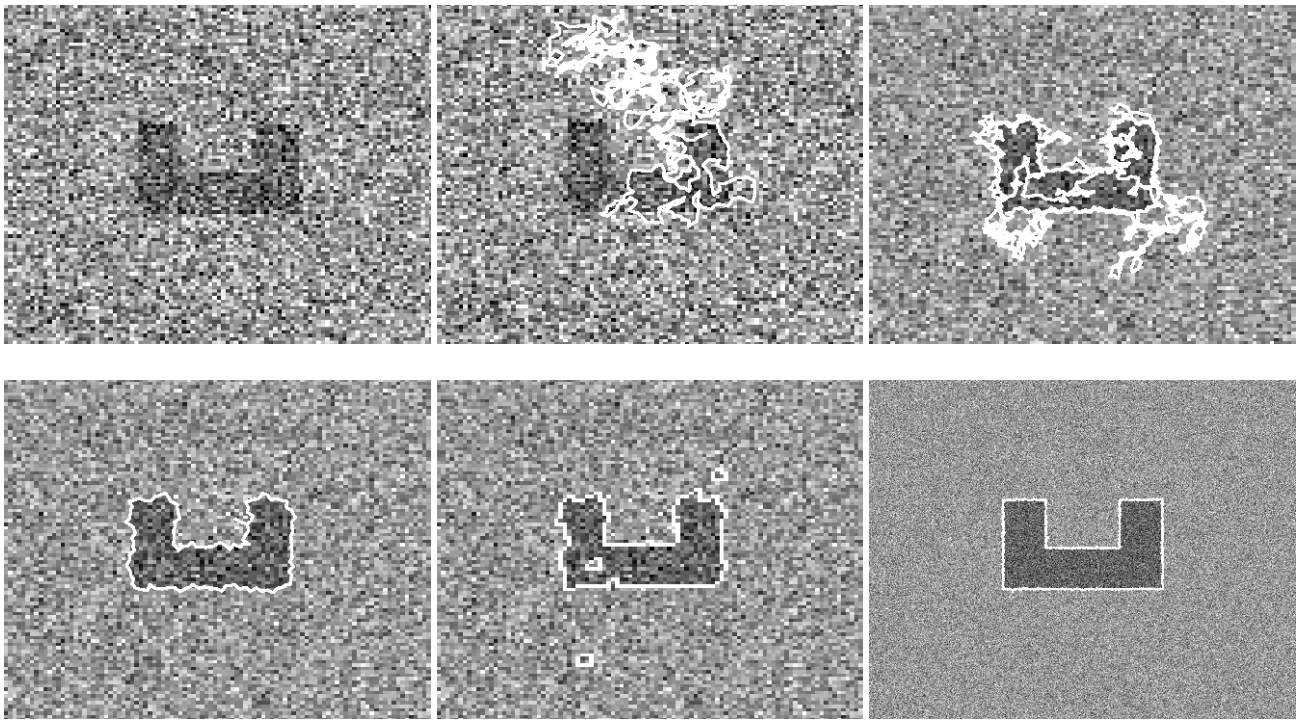

Fig. 3. Top Left: A 100×100 image was corrupted with additive Gaussian noise, N(0,0.5). Top center: Boundary tracking without change-detection. Top right: Boundary tracking with CUSUM. Bottom left: Boundary tracking with CUSUM and a 3×3 window average. Bottom center: Threshold dynamics [25]. Bottom right: Boundary tracking on a 1000×1000 version of the "U" image.

Table 1. A comparison of run-times for various images and segmentation methods. Computations were done with an Intel Core 2 Duo Processor T8300 (2.4 GHz).

A comparison of run-times (s)		
Image	Boundary Tracking	Split Breg. GCS
100×100 U	0.0025	0.034
300×300 U	0.0089	0.620
1000×1000 U	0.0420	6.600

For the tracking of more than one object, it is possible to use another segmentation method first on a subsampled version of the image to obtain topology and the rough locations of objects. This has the added benefit of giving estimates for the mean and variance, which can be used in the case of Gaussian noise. Furthermore, if the initial segmentation is reasonably accurate, the boundary tracking algorithm can be restricted to detecting only boundary points close to those detected in the initial segmentation. Thus, the boundary tracking algorithm acts as a fast refinement to another segmentation method. The resulting detection can also be used as an initialization to another segmentation method on the full data set. The last modification would give fast convergence while preserving the strengths of an alternative method.

An example on a simple, noisy image is shown in Fig. 4. The original image is 1000×1000. Threshold Dynamics was first applied to a heavily subsampled version (100×100) of the image. Then one pixel from each connected component was taken as the starting point for a boundary tracker.

The boundary tracking algorithm is ideally suited for segmentation in large data sets. Detection delays of boundary points are the same in terms of number of steps, whether for low resolution or high resolution data. Thus, relative to the size of objects, detection delays are much shorter. Furthermore, steps with smaller angular increment can be used, effectively taking straighter steps and tracking the boundary more sharply. Using a lower angular increment also increases the speed of tracking, lowering the number of iterations needed to track the object completely. A tracking of a 1000×1000 version of the "U" image is also shown in Fig. 3. Note that the "unit" of detection delay is in steps. That is, for a high resolution image, the detection delay is much smaller in terms of the characteristic width of features. Thus, the tracking result looks much

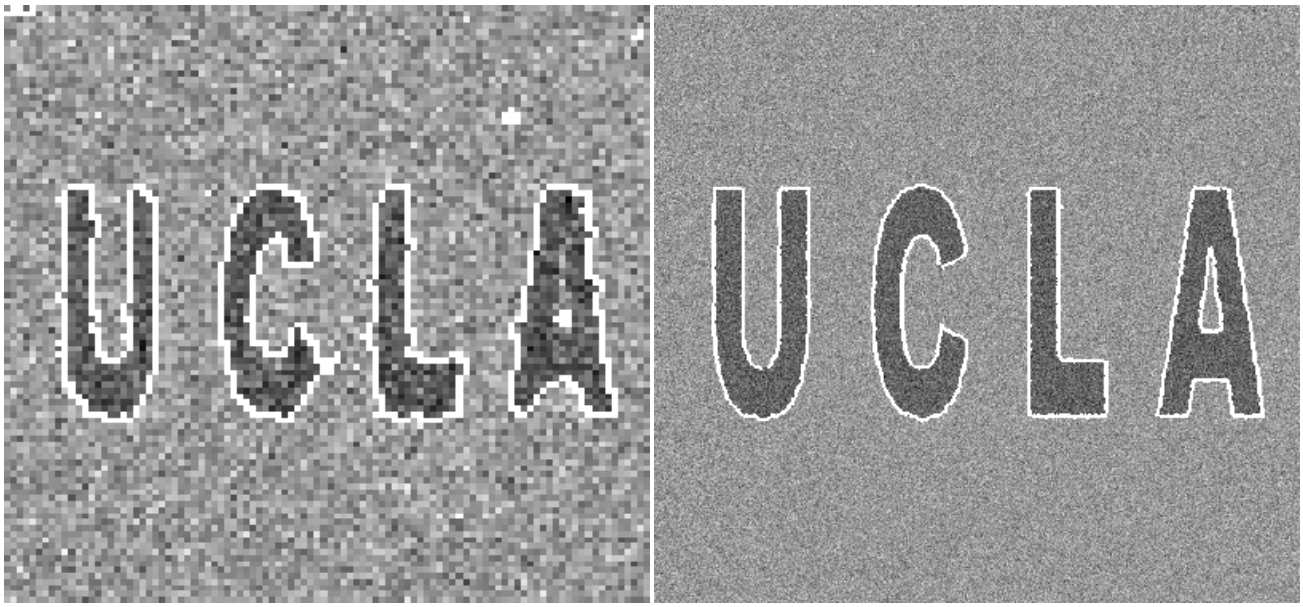

Fig. 4. A hybrid threshold dynamics - boundary tracking segmentation on a 1000×1000 image. Left: Initial segmentation by threshold dynamics. The image is subsampled by a factor of 10 on each axis. Right: Final segmentation by boundary tracking, with starting points for the trackers from the initial segmentation.

more accurate. While the jagged tracking of the 100×100 version still exists in the 1000×1000 version, the scale of the jagged behavior is small compared to the features and thus is almost invisible.

In the "San Francisco Bay" data set [1], a multispectral data set taken by the Landsat 7 satellite, the data is 3000×3500 pixels, with boundaries of the object of interest touching the edge of the image. The Normalized Difference Vegetation Index (NDVI), commonly used for water detection [54], is taken as the decision function. This index is a combination of two images at the same spatial coordinates, taken at a red wavelength band and an infrared wavelength band.

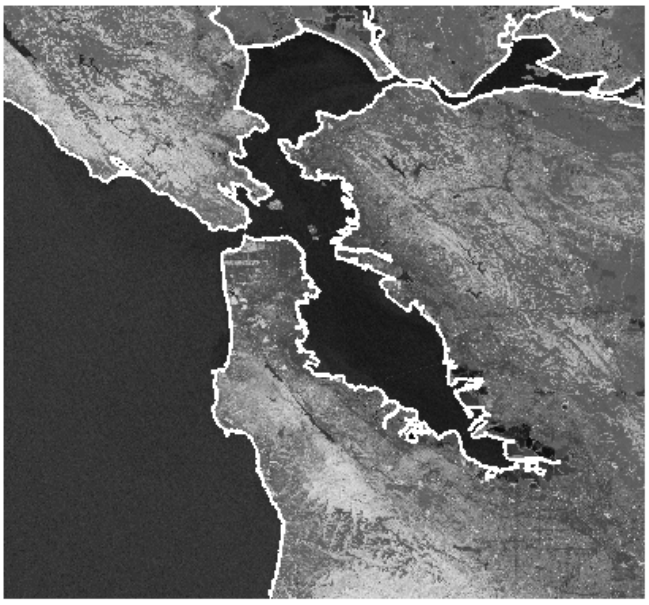

Fig. 5. Boundary tracking of the San Francisco Bay [1] coastline. In this example, multiple trackers need to be used since the coastline boundary is not connected.

The boundary tracking method also has a natural extension to hyperspectral data, which consists of a group of spatially co-registered images taken at different wavelengths in the electromagnetic spectrum. Thus, each spatial pixel now has a vector-valued "spectral signature" instead of a scalar intensity. In fact,

it has been observed that different objects have distinct spectral signatures, so that objects are readily distinguishable by observing spectral signatures.

In the tracking algorithm, all that is required is a decision function that determines whether the tracker is in one of two regions. Then replacing a simple threshold by a suitable distance can still give a good boundary detection. One distance that has been found to work especially well with hyperspectral data is the spectral angle distance [67, 56]:

$$\text{Spectral angle} = \cos^{-1}\left(\frac{u \cdot v}{\|u\|\|v\|}\right) \tag{7}$$

If there are multiple objects to be detected, one may wish to compare two classes of objects, rather than two specific objects. In this case, the decision function should compare distance to one class (the minimum distance to an object in the class) and distance to the other class. This comparison is especially useful in the case of hyperspectral imagery. The theoretical background on using such class comparisons is given in [59]. The segmentation resulting from a class comparison between building references and background references for the Urban data set, a hyperspectral data [2] set taken by HYDICE, an airborne sensor, is shown in Fig. 6.

Fig. 6. Building segmentation of the "Urban" data set [2] with spectral angle decision function.

With the change to a different decision function, the choice of f, g becomes more unclear. It is difficult or impossible to estimate the structure of the data in terms of spectral angle distance. By choosing a large number of sample points, it may, however, be possible to create an approximation of f, g from the data itself.

With hyperspectral imagery, the number of spectral bands can number in the hundreds. With this amount of data, it is often necessary to reduce the size of the data, either in the spectral dimension or through spatial subsampling. Using the boundary tracking algorithm equipped with the spectral angle distance, segmentation can be fast and accurate.

The Smith Island $\mathrm{H20000508\_southendBIP}$ data set [8, 5, 6, 7] is a large hyperspectral data set obtained by an airborne hyperspectral scanner (HyMAP) on May 8, 2000, over Smith Island, VA, a barrier island in the Virgina Coast Reserve. The boundary between land and sea is often ambiguous, causing problems especially when the full spectral information is not used. The NDVI of the data generally gives a good threshold for land and sea, but sandy and swampy areas occupy a range of values that constitute the ambiguous region. Considering spectral signatures, however, gives a clearer indication of the various types of terrain. Moreover, it is also possible to assign these ambiguous areas to either region according to user preference, simply by changing the reference points for each region.

In Fig. 7, each pixel is unmixed as a linear combination of various reference spectral signatures using the L^1 unmixing model of [31]. The reference signatures used are that of beach, dark dirt, light dirt, vegetation, light water, and dark water. Each pixel is assigned to either a land class or water water class based on the maximum abundance of the material at that location. The plots of the reference signatures are shown in Fig. 8. Fig. 9 shows that tracking the maximum abundance shows the boundary more clearly than using the NDVI result.

More generally, boundary tracking can be used to efficiently track features that can be derived from the solution of an inverse problem. Since boundary tracking is a subsampling algorithm, the number of pixels

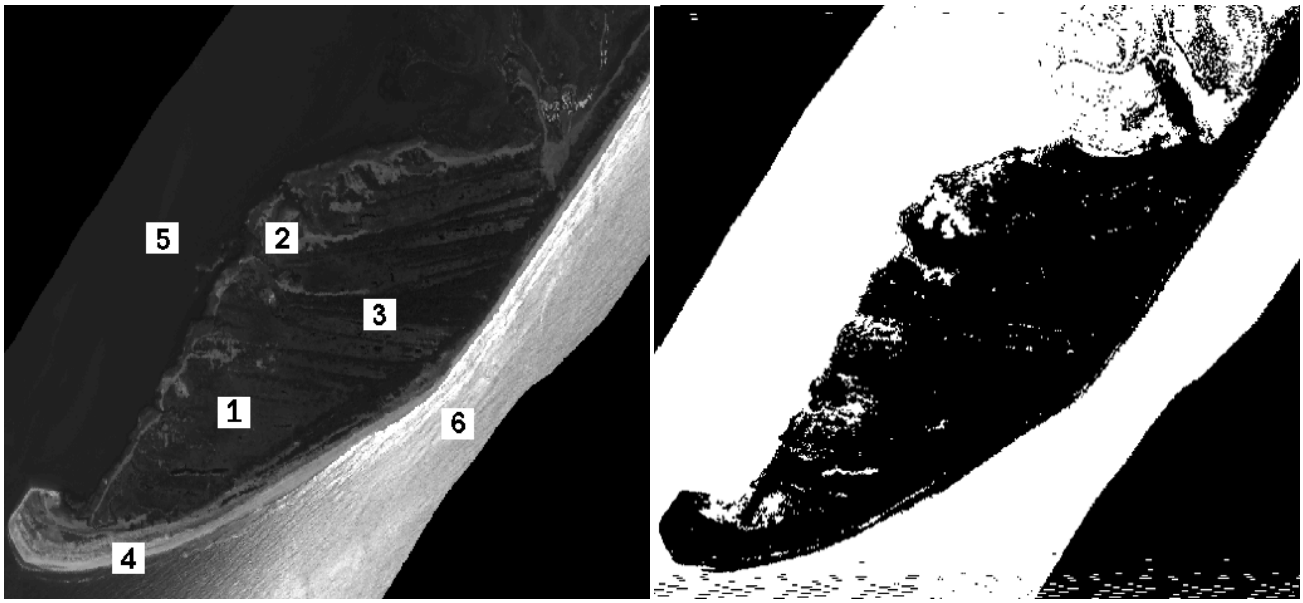

Fig. 7. The Smith Island data set [8, 5, 6, 7]. Left: Reference points used in the land/water classification by full unmixing. The reference signatures are that of 1 - dark dirt, 2 - light dirt, 3 - vegetation, 4 - beach, 5 - dark water, 6 - light water. Right: The land/water classmap using maximum abundance of the unmixed result.

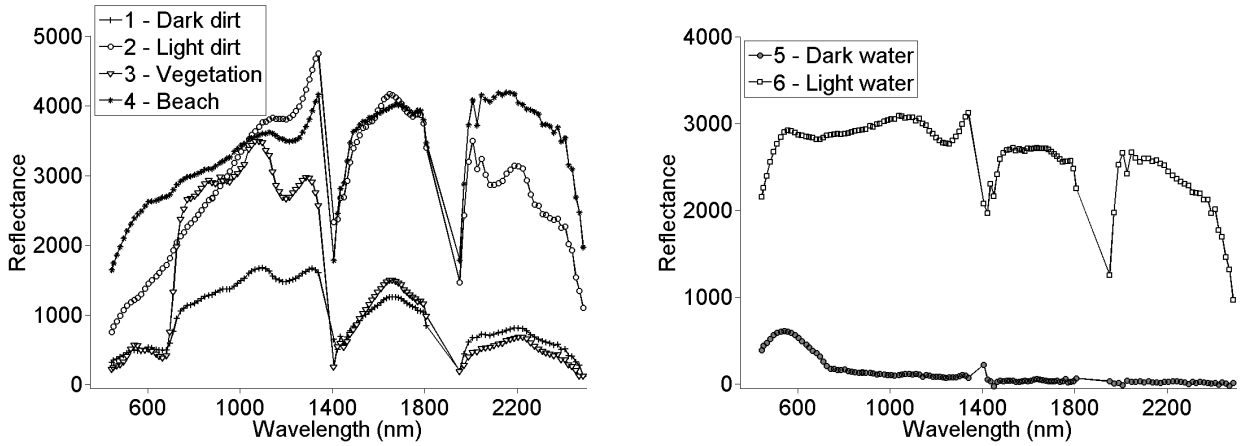

Fig. 8. Left: Plots of the reflectances of the reference signatures of the land class: dark dirt, light dirt, vegetation, beach. Right: Plots of the reflectances of the reference signatures of the water class: dark water, light water.

that require intense processing is minimal. In certain cases, the decision function for each pixel may be a complex calculation requiring a high run-time. This is especially true when calculating the decision function accurately is more computationally expensive as the boundary is approximated with greater precision. The next example for fractals is an example of this fact.

Tracking fractal boundaries such as coastlines is another interesting example of the efficiency of boundary tracking (see Fig. 5). For fractals such as the Mandelbrot set, the escape algorithm [23] can be used to calculate whether a point is in the set or outside. While it can be computationally expensive to implement the algorithm for very fine detail, tracking only the boundary can be much faster. In addition, the potential detail is unlimited since the fractal is defined in continuous space rather than the discrete space of images. Moreover, boundary tracking follows rough boundaries without introducing extra smoothing, as in many segmentation methods with a penalty on the length of the boundary. This fact makes boundary tracking a method well-suited for dealing with fractal-like structure. Fig. 11 shows various magnifications of boundary tracking on the Mandelbrot set. Table 2 gives a timing comparison between the boundary tracking algorithm and the calculation of the escape algorithm at the corresponding resolution. In the results shown in Fig. 11 and 2, the computations on the escape algorithm for boundary tracking were done only at locations that the

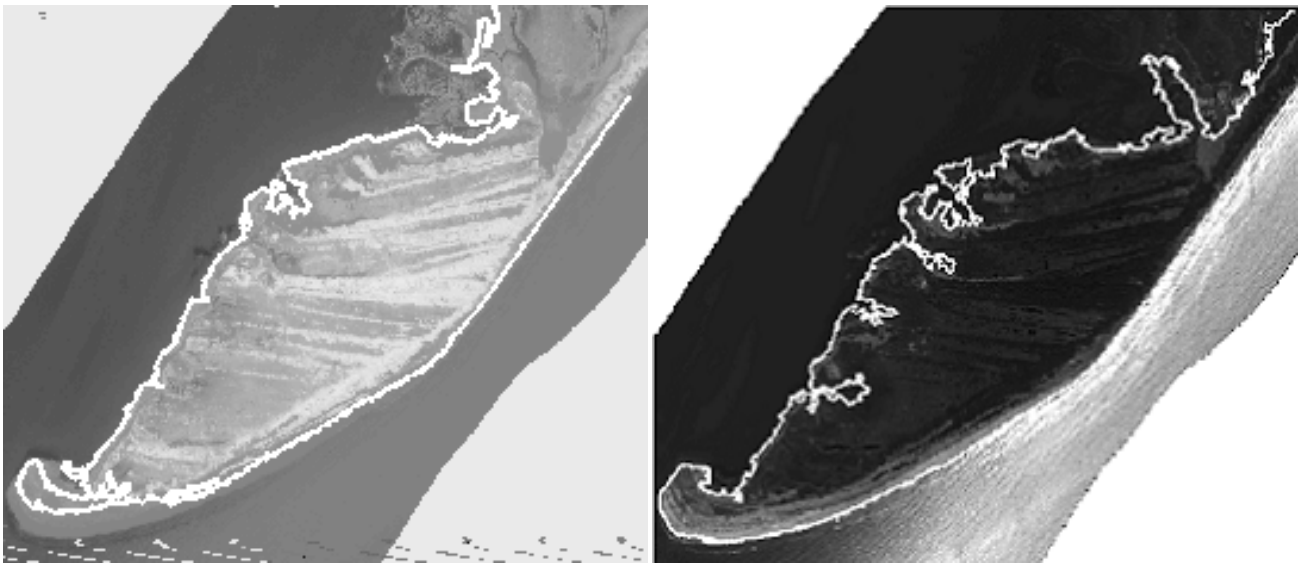

Fig. 9. Tracking the boundary of the "Smith Island" hyperspectral data set. Left: Grayscale boundary tracking using the NDVI as an indicator between land and sea. Right: Boundary tracking using unmixing on each pixel of the data set. Only pixels visited by the boundary tracker are unmixed.

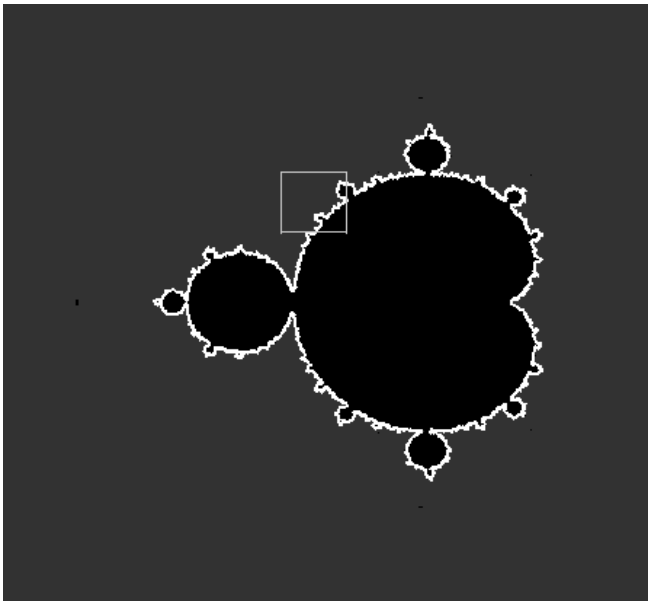

Fig. 10. Boundary tracking of the Mandelbrot set at 400×400 resolution. A higher magnification of the boxed region is shown in Fig. 11.

tracker visits, not on the entire 400×400 images. The tracking results are shown overlaid on the full images for clarity.

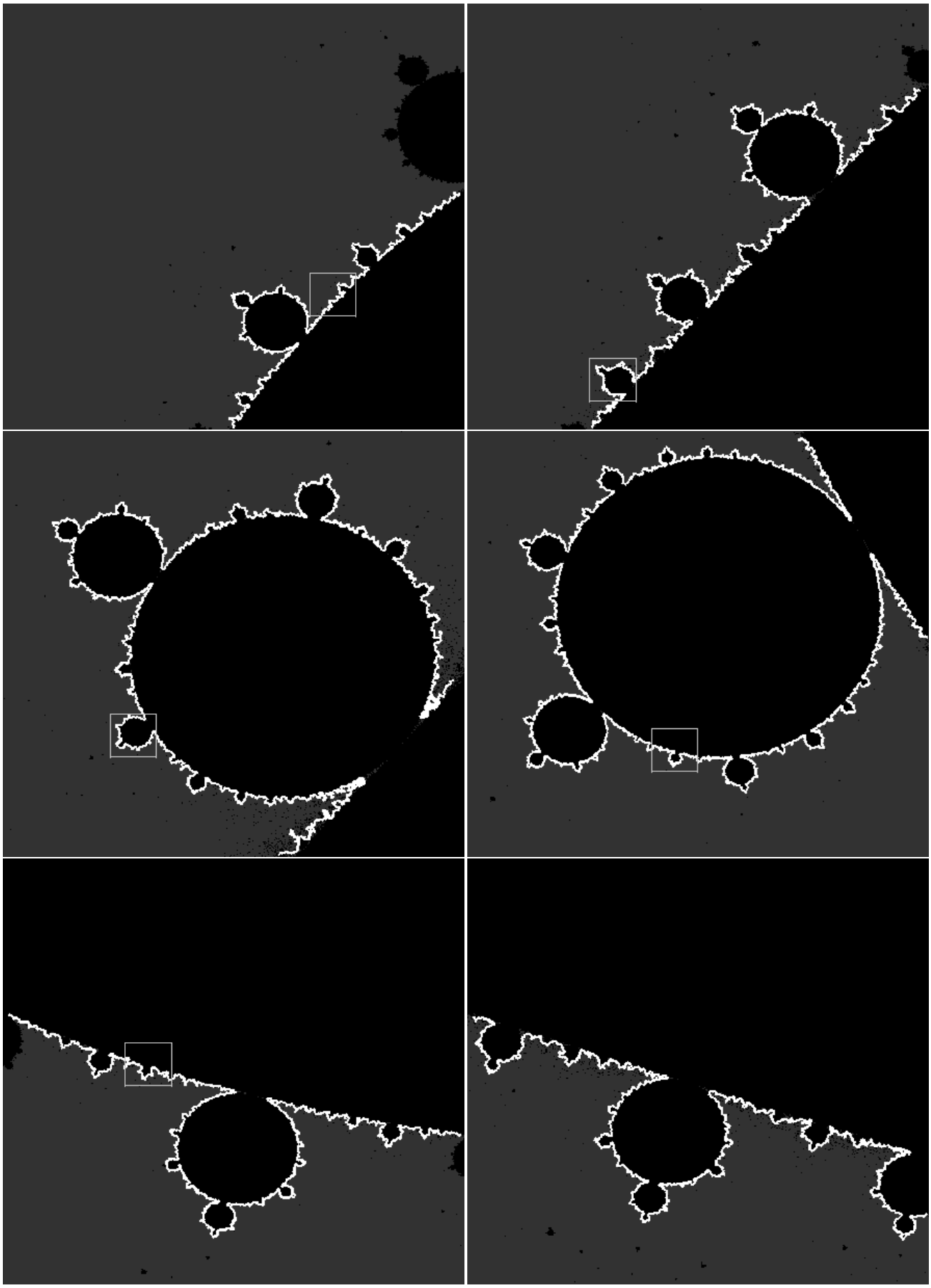

Fig. 11. Tracking the boundary of the Mandelbrot set at increasing levels of magnification with the boxed region showing the domain for the next highest level of magnification. The resolution of each image is 400×400. Top: 10 times magnification, 10^2 times magnification. Middle: 10^3 times magnification, 10^4 times magnification. Bottom: 10^5 times magnification, 10^6 times magnification.

Note that as the magnification is increased, the number of iterations in the escape algorithm needed to resolve the boundary accurately also increases. This is due to the fact that the points being examined are much closer to the actual boundary of the Mandelbrot set, so that the escape algorithm for points not in the Mandelbrot set will diverge much more slowly. This gives an additional savings in the run-time of boundary tracking relative to computing on the entire domain, as the resolution is increased.

Table 2. A table comparing the run-time of the boundary tracking algorithm (BT) and calculation of the full domain (FD) for the Mandelbrot set at the corresponding resolution (pixel width). The precision of the escape algorithm (Escape) was adjusted according to the resolution, as needed.

Mandelbrot set timing comparison			
Pixel Width	Escape	BT (s)	Full FD (s)
7.51×10^{-3}	400	0.7	5.8
7.51×10^{-4}	10^3	1.2	21.5
7.51×10^{-5}	4×10^3	7.8	152.6
7.51×10^{-6}	10^4	32.6	529.7
7.51×10^{-7}	4×10^4	145.9	2100.4
7.51×10^{-8}	10^5	218.6	6078.6
7.51×10^{-9}	4×10^5	861.5	22042.3

By tracking the boundary of the fractal at various magnification factors, it is possible to measure the fractal dimension using the box-counting method. The fractal dimension of a set is defined to be

$$D = \lim_{\epsilon \to 0} \frac{\log(N(\epsilon))}{\log(1/\epsilon)},$$

where N represents the number of boxes of size ϵ that are needed to cover the set. At a given scale ϵ, there exists a simple relationship between the length $L(\epsilon)$ and the number of boxes $N(\epsilon)$: $L(\epsilon) = \epsilon N(\epsilon)$. Using this formula, the fractal dimension can be rewritten as

$$\lim_{\epsilon \to 0} \frac{\log(L/\epsilon)}{\log(1/\epsilon)} = 1 - \lim_{\epsilon \to 0} \frac{\log(L)}{\log(\epsilon)}.$$

Using various step sizes V for the scale parameter ϵ, boundary tracking yields boundary points at various scales. Thus, a log-log plot of $\log(L)$ versus $\log(V)$ is approximately a line with slope equal to one minus the fractal dimension. A calculation of the boundary for the Julia set with $c = 1/4$ is shown in Table 3. The calculated slope is -0.0799, so the measured fractal dimension is 1.0799, which is close to the theoretical value of 1.0812 [43]. The corresponding set and log-log plot is shown in Fig. 12. Note that the Mandelbrot has comparatively more complex structure, having a fractal dimension of 2 [57], and the numerical computation of the fractal dimension is not usually done. The computation of the fractal dimension here is a demonstration of one application of boundary tracking.

Table 3. A table of the length calculations for the Julia set at various scales.

Julia set lengths		
V	Boundary Points	Length
5×10^{-5}	42 582	3.7706
10^{-4}	20 152	3.5642
3×10^{-4}	6210	3.2655
5×10^{-4}	3538	3.1364
10^{-3}	1668	2.9773
3×10^{-3}	513	2.7254
5×10^{-3}	308	2.5925
10^{-2}	137	2.4744

Similar fractal dimension calculations can be done for real coastlines, as long as the resolution of the data set is sufficient. For the Smith Island data set, a fractal dimension can be calculated using boundary tracking. The results are shown in Table 4 and Fig. 13. From the fitted slope of -0.1739 for the log-log plot, the measured fractal dimension is thus 1.1739.

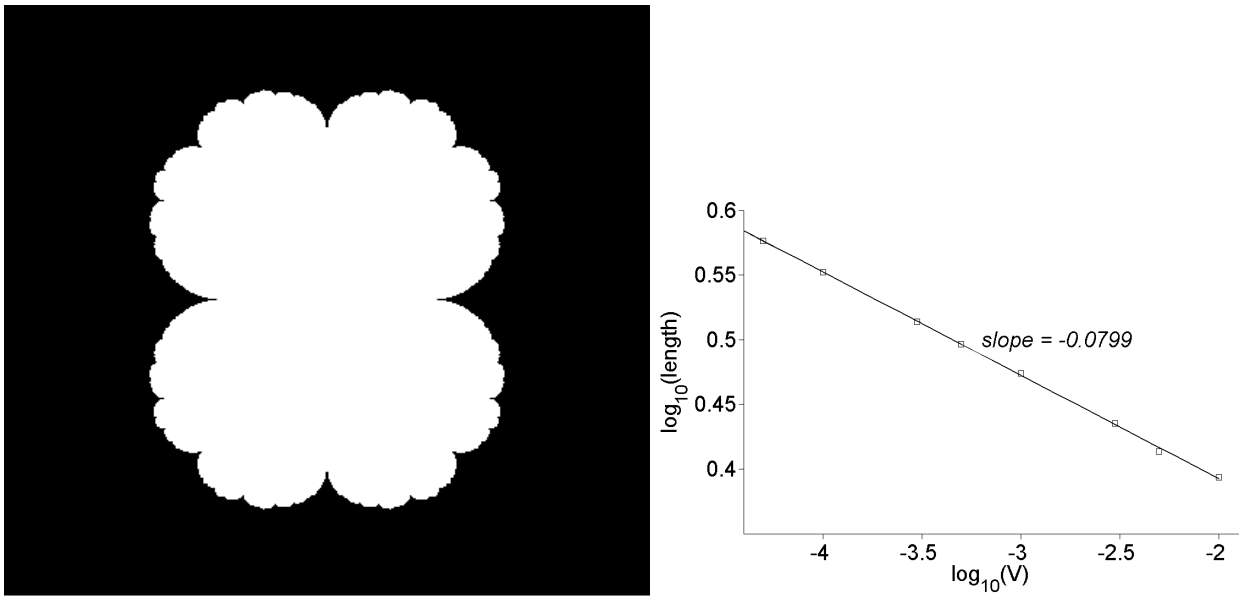

Fig. 12. Left: The Julia set for $c = 1/4$. Right: Log-log plot of $\log_{10}(L)$ versus $\log_{10}(V)$.

Table 4. A table of the length calculations for the Smith Island set at various scales.

Smith Island set lengths		
V	Boundary Points	Length
0.5	14 431	8162
1	6111	7057
2	2939	6830
4	1271	6024
8	464	4651
16	199	4651
32	93	3976

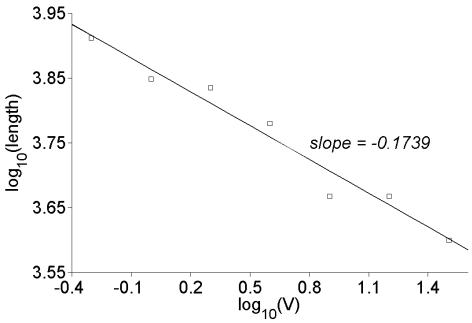

Fig. 13. Log-log plot of $\log_{10}(L)$ versus $\log_{10}(V)$ for Smith Island.

6 Discussion

The greatest advantage of the boundary tracking algorithm over region-based methods for segmentation is the computation speed. Calculations and measurements are taken over samples, with the number of samples having the same order as the boundary length of the tracked object. In region-based methods, however, calculations and measurements are done over the entire image. Thus, boundary tracking is $O(n)$, while region-based methods are $O(n^2)$, where the size of the image is $n \times n$. Then the savings becomes more apparent for larger images and those with higher resolution.

There is a substantial savings in memory requirements as well. Since only local information is used in the decision function, it is not even necessary for the entire image to be stored at once. Instead, image values

can be read as needed whenever the tracker visits a new point. On the other hand, many segmentation methods require not only the original data to be stored but also evolved data of the same size.

With high-noise data, change-point detection is vital in minimizing mistakes due to noise. Parameters are often chosen in order to minimize detection delay and false alarms. In the boundary tracking problem, the former is of paramount importance, since region changes happen very frequently. While false alarms can be tolerated to some extent, consistent detection delays can seriously hamper the algorithm's performance, causing false alarms to increase as well, by leading the tracker away from the boundary. Improving the algorithm's ability to recover from such mistakes is an important consideration.

The change-point modifications run very quickly, requiring only a minimal amount of extra storage and computation over the boundary tracking algorithm alone. Using a window replacement is slightly slower since more pixels are sampled. In this case, however, the algorithm still operates in $O(n)$. Moreover, the greater accuracy with which the boundary is tracked can lead to additional savings in computation time, as fewer tracking mistakes are made and thus fewer iterations are needed.

While it is desirable in general to minimize the false alarms and detection delay, these indices do not provide a complete understanding of a change detection algorithm's effectiveness in the boundary tracking problem. This is due to the fact that a false alarm of just one change point can cause the boundary tracker to travel far from the boundary. In subsequent readings, change-point detection may be completely accurate and have low detection delay, but give no useful information about the boundary. Thus, the effectiveness of a change point detection algorithm must be measured visually with respect to the final segmentation result as well. Nevertheless, these parameters are useful numerical indicators of a procedure's effectiveness.

The change-point detection literature studies various measures of false alarms and detection delays. Due to the interdependency of each change-point detection test that is run, simply tracking these two indices may not be a good indicator of how the algorithm performs. Also, most of the examples presented involve some a priori assumption on the probability distribution of the data. This assumption may not be accurate for some data, especially for complicated data sets like hyperspectral imagery. Some possibilities from change-point detection [11] have been studied, but it is also not clear whether these would represent an improvement over a priori assumptions on the distributions.

Another important factor in the effectiveness of the algorithm is the characteristic width of the object to be tracked. If the step size of the boundary tracker is of the same order of magnitude as the characteristic width of the object, it can be difficult to track an object effectively. But with a characteristic width much larger than boundary tracker step size (equivalently, higher resolution) data, the algorithm can be highly effective without a corresponding large jump in run-time. Note that step size cannot be decreased arbitrarily; the resolution restricts the amount of data that can be used when the step size becomes too small. Thus, narrow objects are difficult to track if the resolution of the data is not sufficient. Of course, this limitation on tracking features smaller than characteristic dimensions is true for all segmentation methods.

When there are more than two regions to be differentiated, different possibilities arise. One method is a modification to track classes of objects. This is especially relevant to the case with hyperspectral data, in which there are multiple objects with different spectral signatures. In this case, the decision function can be altered so that a given pixel is measured against proximity to an entire class, rather than just to one object. Particular care needs to be taken, however, to choose the classes carefully.

It is also important to take into account the structure of the data when using a different decision function. In some of the examples presented, images were corrupted with additive Gaussian noise. But for real images, the noise parameters or structure of the data is not known a priori. The assumption of Gaussian data, however, does seem to fit a wide variety of data.

An extension to the boundary tracking algorithm is to use multiple trackers. The algorithm is easily parallelizable, adding a stopping condition for each tracker; for example, the algorithm for one tracker terminates if it meets the path taken by any other tracker. Aside from parallelization for the sake of run-time, information from multiple trackers can potentially be used in concert to improve tracking.

The boundary tracking algorithm can be of use in many problems in which run-time is important. For video tracking, the large number of frames used can lead to a large computation time. By using the boundary tracking algorithm with initialization taken from previous frames, objects can be found and tracked quickly through each frame. Other applications include surface segmentation and more sophisticated parameter estimation models by using shape priors.

Acknowledgements

The authors would like to thank Dr. C. Bachmann and the Naval Research Laboratory for helpful discussions on geospatial tracking problems as well as the use of the "Smith Island" data set, Z. Hu and V. Mejia for earlier work on the boundary tracking algorithm in the image processing context [34, 44], and T.

Goldstein for the implementation of the split Bregman GCS method. This work was supported by Office of Naval Research grant [N000140810363]; Dept. of Defense grant ARO MURI [50363-MA-MUR]; and National Science Foundation grant [DMS-0914856].

References

[1] San Francisco Bay multispectral data set. Landsat 7 satellite images.

[2] Urban hyperspectral data set. HYDICE sensor imagery.

[3] S. Andersson. Curve tracking for rapid imaging in AFM. *IEEE Transactions on Nanobioscience*, 6(4), December 2007.

[4] N. Azzabou, N. Paragios, and F. Guichard. Random walks, constrained multiple hypotheses testing and image enhancement. *9th European Conference in Computer Vision (ECCV)*, 2006.

[5] Charles M. Bachmann. Improving the performance of classifiers in high-dimensional remote sensing applications: An adaptive resampling strategy for error-prone exemplars (aresepe). *IEEE Transactions on Geoscience and Remote Sensing*, 41(9):2101–2112, 2003.

[6] Charles M. Bachmann, Thomas L. Ainsworth, and Robert A. Fusina. Exploiting manifold geometry in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):441–454, 2005.

[7] Charles M. Bachmann, Thomas L. Ainsworth, and Robert A. Fusina. Improved manifold coordinate representations of large scale hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 44(10):2786–2803, 2006.

[8] C.M. Bachmann, T.F. Donato, G.M. Lamela, W.J. Rhea, M.H. Bettenhausen, R.A. Fusina, K.R. Du Bois, J.H. Porter, and B.R. Truitt. Automatic classification of land cover on Smith Island, VA, using HyMAP imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 40(10):2313–2330, October 2002.

[9] Xue Bai and Guillermo Sapiro. Distancecut: Interactive segmentation and matting of images and videos. *IEEE ICIP*, 2:249–252, 2007.

[10] Marco Barchiesi, Sung Ha Kang, Triet Le, Massimiliano Morini, and Marcello Ponsiglione. A variational model for infinite perimeter segmentations based on Lipschitz level set functions: denoising while keeping finely oscillatory boundaries. *SIAM Multiscale Modeling and Simulation (to appear)*, 2010.

[11] M. Basseville and I. Nikiforov. *Detection of Abrupt Changes - Theory and Applications*. Information and System Sciences Series. Prentice Hall, Englewood Cliffs, NJ, US, 1993.

[12] Andrea Bertozzi and John Greer. Low-curvature image simplifiers: Global regularity of smooth solutions and laplacian limiting schemes. *Communications on Pure and Applied Mathematics*, 57(6):764–790, 2004.

[13] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–714, 1986.

[14] K. Castleman. *Digital Image Processing*. Prentice Hall, 1996.

[15] Tony Chan and Luminita Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, February 2001.

[16] Peter I. Chang and Sean B. Andersson. Smooth trajectories for imaging string-like samples in AFM: A preliminary study. In *2008 American Control Conference*, Seattle, Washington, June 2008.

[17] A. Chen, T. Wittman, A. Tartakovsky, and A. Bertozzi. Image segmentation through efficient boundary sampling. In *Proceedings of the 2009 Workshop on Sampling Theory and Applications*, May 2009.

[18] Chia Hsun Chiang and Jing-Sin Liu. Boundary following in unknown polygonal environment based on fast marching method. In *IEEE International Conference on Advanced Robotics and its Social Impacts*, 2008.

[19] L.D. Cohen. On active contour models and balloons. *CVGI: Image Understanding*, 53(2):211–218, March 1991.

[20] I. D. Couzin and N. R. Franks. Self-organized lane formation and optimized traffic flow in army ants. *P. Roy. Soc. Lond. B. Bio.*, 270:139–146, 2003.

[21] J. Darbon. A note on the discrete binary Mumford-Shah model. In *Lecture Notes in Computer Science*, volume 4418, pages 283–294. Springer Berlin/Heidelberg, 2007.

[22] Enrique del Castillo. *Statistical Process Adjustment for Quality Control*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 2002.

[23] Adrien Douady and John H. Hubbard. Etude dynamique des polynômes complexes. *Prépublications mathématiques d'Orsay*, 2/4, 1984/85.

[24] N. El-Zehiry, S. Xu, P. Sahoo, and A. Elmaghraby. Graph cut optimization for the Mumford-Shah model. In *Proceedings of the Seventh IASTED International Conference Visualization, Imaging, and Image Processing.*, Palma de Mallorca, Spain, August 2007.

[25] S. Esedoglu and Y. R. Tsai. Threshold dynamics for the piecewise constant Mumford-Shah functional. *J. Comput. Phys.*, 211(1):367–384, 2006.

[26] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Trans. Patt. Anal. and Machine Intel.*, 18(1):1–14, January 1996.

[27] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[28] Tom Goldstein and Stanley Osher. The split Bregman algorithm for L1 regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[29] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 2008.

[30] T. Griffiths and A. Yuille. Technical introduction: A primer on probabilistic inference. 2006.

[31] Zhaohui Guo, Todd Wittman, and Stanley Osher. L1 unmixing and its application to hyperspectral image enhancement. UCLA CAM report, March 2009.

[32] W. Hangartner. Spezifitt und inaktivierung des spurpheromons von lasius fuliginosus (latr.) und orientierung der arbeiterinnen im duftfeld. *Z. vergl. Physiol.*, 57:103–136, 1967.

[33] Chung H. Hsieh, Zhipu Jin, Daniel Marthaler, Bao Q. Nguyen, David J. Tung, Andrea L. Bertozzi, and Richard M. Murray. Experimental validation of an algorithm for cooperative boundary tracking. In *Proceedings of the American Control Conference*, pages 1078–1083, Portland, 2005.

[34] Zhong Hu. Gradient-free boundary tracking. Undergraduate research project.

[35] Z. Huang and K. Chau. A new image thresholding method based on gaussian mixture model. *Applied Mathematics and Computation*, 205:899–907, 2008.

[36] Zhipu Jin and Andrea Bertozzi. Environmental boundary tracking and estimation using multiple autonomous vehicles. In *2007 46th IEEE Conference on Decision and Control*, pages 4918–4923, New Orleans, LA, USA, December 2007.

[37] A. Joshi, T. Ashley, Y. Huang, and A. Bertozzi. Experimental validation of cooperative environmental boundary tracking with on-board sensors. In *American Control Conference*, pages 2630–2635, St. Louis, MO, June 2009.

[38] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[39] M. Kemp, A. L. Bertozzi, and D. Marthaler. Multi-UUV perimeter surveillance. In *Proceedings of 2004 IEEE/OES Workshop on Autonomous Underwater Vehicles*, pages 102–107, 2004.

[40] Kevin K. Leung, Chung H. Hsieh, Yuan R. Huang, Abhijeet Joshi, Vlad Voroninski, and Andrea L. Bertozzi. A second generation micro-vehicle testbed for cooperative control and sensing strategies. In *Proceedings of the 2007 American Control Conference*, pages 1900–1907, 2007.

[41] W. Liu, M. Short, Y. Taima, and A. Bertozzi. Multiscale collaborative searching through swarming. In *Proceedings of the 7th International Conference on Informatics in Control, Automation, and Robotics (ICINCO)*, Portugal, June 2010.

[42] G. Lorden. Procedures for reacting to a change in distribution. *Annals of Mathematical Statistics*, 42:1897–1908, 1971.

[43] Curtis T. McMullen. Hausdorff dimension and conformal dynamics, III: Computation of dimension. *American Journal of Mathematics*, 120(4):691–721, 1998.

[44] Victor Mejia. Multiple robot boundary tracking. Undergraduate research project.

[45] Douglas Montgomery. *Introduction to statistical quality control*. John Wiley & Sons, Inc., 1985.

[46] G. Moustakides. Optimal stopping times for detecting changes in distributions. *Annals of Statistics*, 14:1379–1387, 1986.

[47] David Mumford and Jayant Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Math*, XLII(5):577–684, July 1989.

[48] James Ng and Thomas Bräun. Performance comparison of bug navigation algorithms. In *Journal of Intelligent and Robotic Systems*, volume 50, pages 73–84. Springer Netherlands, September 2007.

[49] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulation. *J. Comput. Phys.*, 79:12–49, 1988.

[50] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1-2):100–115, June 1954.

[51] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A pde-based fast local level set method. *J. Comput. Phys*, 155:410–438, 1999.

[52] Alexis Protiere and Guillermo Sapiro. Interactive image segmentation via adaptive weighted distances. *IEEE Transactions on Image Processing*, 16(4):1046–1057, 2007.

[53] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (SIGGRAPH)*, August 2004.

[54] J.W. Rouse, R.H. Hass, J.A. Schell, and D.W. Deering. Monitoring vegetation systems in the grain plains with ERTS. In *Third ERTS Symposium*, pages 309–317, NASA SP-351 I, 1973.

[55] Thomas P. Ryan. *Statistical Methods for Quality Improvement*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 2nd edition, 2000.

[56] P. Shippert. Introduction to hyperspectral image analysis. *Online J. of Space Commun.*, 2003.

[57] M. Shishikura. The Hausdorff dimension of the boundary of the Mandelbrot set and Julia sets. *Ann. of Math.*, 147:225–267, 1998.

[58] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. CVPR.*, June 1999.

[59] A. Tartakovsky. Asymptotic performance of a multichart CUSUM test under false alarm probability constraint. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, pages 320–325, 2005.

[60] A.G. Tartakovsky, B.L. Rozovskii, R. Blažek, and H. Kim. Detection of intrusions in information systems by sequential change-point methods. *Statistical Methodology*, 3(3):252–293, July 2006.

[61] A.G. Tartakovsky and V.V. Veeravalli. Change-point detection in multichannel and distributed systems with applications. In N. Mukhopadhyay, S. Datta, and S. Chattopadhyay, editors, *Applications of Sequential Methodologies*, pages 331–363. Marcel Dekker, Inc., New York, 2004.

[62] Z. Tu and S. C. Zhu. Image segmentation by data-driven Markov chain Monte Carlo. *IEEE Trans. on PAMI*, 24(5):657–673, May 2002.

[63] J. Tumblin and G. Turk. LCIS: A boundary hierarchy for detail-preserving contrast reduction. In *SIGGRAPH annual conference on computer graphics*, page 8390, Los Angeles, CA, August 1999.

[64] G. Viswanathan, S. Buldyrev, S. Havlin, M. da Luz, E. Raposo, and E. Stanley. Optimizing the success of random searches. *Nature*, 401(6756):911–914, 1999.

[65] A. Willsky. A survey of design methods for failure detection in dynamical systems. *Automatica*, 12:601–611, 1976.

[66] C. Xu and J.L. Prince. Gradient vector flow: A new external force for snakes. *Proc. IEEE Conf. on Comp. Vis. Patt. Recog. (CVPR)*, pages 66–71, June 1997.

[67] R. Yunas, A. Goetz, and J. Boardman. Discrimination among semi-arid landscape endmembers using the spectral angle mapper (SAM) algorithm. In *Summaries of the Third Annual JPL Airborne Geoscience Workshop*, pages 147–149, 1992.

[68] S. C. Zhu and A. L. Yuille. Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Trans. on PAMI*, 18(9):884–900, 1996.