TOWARDS A POLYALGORITHM FOR LAND USE AND LAND COVER CHANGE DETECTION

by

Rishu Saxena

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science and Applications

Layne T. Watson, Chair
Randolph H. Wynne, Co-Chair
Sharath Raghvendra

December 15, 2017
Blacksburg, Virginia

Keywords: Remote sensing; time series analysis; event detection; change detection; big data; scalability; high performance computing.

TOWARDS A POLYALGORITHM FOR LAND USE AND LAND COVER CHANGE DETECTION

by

Rishu Saxena

(ABSTRACT)

Earth observation satellites (EOS) such as Landsat provide image datasets that can be immensely useful in numerous application domains. One way of analyzing satellite images for land use and land cover change (LULCC) is time series analysis (TSA). Several algorithms for time series analysis have been proposed by various groups in remote sensing; more algorithms (that can be adapted) are available in the general time series literature. However, in spite of an abundance of algorithms, the choice of algorithm to be used for analyzing an image stack is presently an open question. A concurrent issue is the prohibitive size of Landsat datasets, currently of the order of petabytes and growing. This makes them computationally unwieldy — both in storage and processing. An EOS image stack typically consists of multiple images of a fixed area on the Earth's surface (same latitudes and longitudes) taken at different time points. Experiments on multicore servers indicate that carrying out meaningful time series analysis on one such interannual, multitemporal stack with existing state of the art codes can take several days.

This work proposes using multiple algorithms to analyze a given image stack in a polyalgorithmic framework. A polyalgorithm combines several basic algorithms, each meant to solve the same problem, producing a strategy that unites the strengths and circumvents the weaknesses of constituent algorithms. The foundation of the proposed TSA based polyalgorithm is laid using three algorithms (LandTrendR, EWMACD, and BFAST). These algorithms are precisely described mathematically, and chosen to be fundamentally distinct from each other in design and in the phenomena they capture. Analysis of results representing success, failure, and parameter sensitivity for each algorithm is presented. Scalability issues, important for real simulations, are also discussed, along with scalable implementations, and speedup results. For a given pixel, Hausdorff distance is used to compare the distance between the change times (breakpoints) obtained from two different algorithms. Timesync validation data, a dataset that is based on human interpretation of Landsat time series in concert with historical aerial photography, is used for validation. The polyalgorithm yields more accurate results than EWMACD and LandTrendR alone, but counterintuitively not better than BFAST alone. This nascent work will be directly useful in land use and land cover change studies, of interest to terrestrial science research, especially regarding anthropogenic impacts on the environment, and in much broader applications such as health monitoring and urban transportation.

TOWARDS A POLYALGORITHM FOR LAND USE AND LAND COVER CHANGE DETECTION

by

Rishu Saxena

(GENERAL AUDIENCE ABSTRACT)

Numerous manmade satellites circling around the Earth regularly take pictures (images) of the Earth's surface from up above. These images naturally provide information regarding the land cover of any given piece of land at the moment of capture (for e.g., whether the land area in the picture is covered with forests or with agriculture or housing). Therefore, for a fixed land area, if a person looks at a chronologically arranged series of images, any significant changes in land use can be identified. Identifying such changes is of critical importance, especially in this era where deforestation, urbanization, and global warming are major concerns.

The goal of this thesis is to investigate the design of methodologies (algorithms) that can efficiently and accurately use satellite images for answering questions regarding land cover trend and change. Experience shows that the state-of-the-art methodologies produce great results for the region they were originally designed on but their performance on other regions is unpredictable. In this work, therefore, a 'polyalgorithm' is proposed. A 'polyalgorithm' utilizes multiple simple methodologies and strategically combines them so that the outcome is better than the individual components. In this introductory work, three component methodologies are utilized; each component methodology is capable of capturing phenomenon different from the other two. Mathematical formulation of each component methodology is presented. Initial strategy for combining the three component algorithms is proposed. The outcomes of each component methodology as well the polyalgorithm are tested on human interpreted data. The strengths and limitations of each methodology are also discussed. Efficiency of the codes used for implementing the polyalgorithm is also discussed; this is important because the satellite data that needs to be processed is known to be huge (petabytes sized already and growing). This nascent work will be directly useful especially in understanding the impact of human activities on the environment. It will also be useful in other applications such as health monitoring and urban transportation.

ACKNOWLEDGEMENTS

First, I would like to thank Dr. Layne Watson for agreeing to be my advisor. With the professional background that I came from and my vision for future career, I cannot think of anyone else at present who could have better mentored me. I am thankful to him for the invaluable insights I received from him on many topics, and wish that I had more opportunity to discuss with and learn from him. I also thank him for his immense patience, especially after my daughter's birth. I would like to thank Dr. Randolph Wynne for letting me be a part of his group, for the feedbacks I received from him whenever I needed/asked him, and all the resources he arranged for. I would like to thank him equally for his support and patience in the past year and a half after my daughter's birth. I also thank Dr. Sharatha Raghavendra for being a part of my committee.

Understanding and using EWMACD became easier because of Dr. Evan Brooks' availability and his willingness to discuss my doubts. I would like to thank him for that. I thank Dr. Valerie Thomas for her support, and especially for covering up for me at various conferences, and Dr. Xinwei Deng (from Department of Statistics, Virginia Tech) for helping me with statistics. Dr. Yang Zhiqiang from Oregon State University shared with us the TimeSync data that was used for validating outcomes in this work. I had been looking for such validation data and am thankful to him. I would especially like to thank Jill Derwin for several useful discussions, her support, and adjusting to my limitations so many times during the past years. I am thankful to Dr. Christine Blinn as well for discussions during the initial phase of this work. I wish I had more opportunity to collaborate with each of these people. I would like to thank Tyler Chang and Chaitra Raghunath for the academic help I received from them and for their support, and Cameron Houser and Sherin Ghannam for their support and encouragement. I am thankful to Rob Hunter for his administrative help. Finally, I could not be more grateful to Dr. Karen DePauw for allowing me the extra time that she did.

Last but not the least, I would like to thank my family and friends for their constant support and faith in me. I would like to thank my husband for his support and patience. I dedicate this thesis to Aarohi — my daughter and my compass star.

The work presented in this thesis was supported in part by NSF Grant CNS-1565314 and USDA Grant 422350.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1.

INTRODUCTION

Land use change is described as changes in how humans use the surface of the Earth (e.g., for agriculture, plantations, pastures, managed woods, conservation, settlements, or leaving it alone as natural ecosystem). Changes in land use lead to changes in albedo, thereby directly affecting the temperatures of the surrounding area. Significant and lasting changes in land use and land cover (LULC) have more profound effects. The past century has seen an exponential growth in human activities such as deforestation and urbanization causing significant changes in land cover in several parts of the world ([32]). Simultaneously, significant changes in the global climate have also been observed, driven in part by LULC change (LULCC) (e.g., [23]). LULCC also has impacts on a wide variety of other ecosystem services. Monitoring LULCC across the globe, therefore, has become the need *du jour*. Land use change detection comprises any methodology used for determining the occurrence and nature of change in LULC.

Earth observation satellites (EOS) such as Landsat capture images of the Earth's surface at regular intervals using multiple spectral frequencies. These images hold valuable information that, if harnessed well, can be immensely helpful in understanding, monitoring, and managing our natural resources, as well as studying LULCC. One way of analyzing these satellite images for LULCC studies is time series analysis (or, temporal trajectory analysis). For time series analysis, several images of the scene under consideration, taken over a period of time, are stacked together chronologically and subsequently analyzed. Commonly, the time series for each pixel is treated individually; the full image stack is thus a collection of many time series. The choice of spectral band(s) varies from application to application. The objective is to discover a 'trend' in how different relevant variables (indicators) evolve over time. In change detection analysis, when the trajectory of one or more of the variables departs from the normal, a change is detected. Time series analysis for LULCC studies has been receiving increasing attention in the last decade, specifically, after the Landsat data became freely accessible in 2008 ([86]). Several time series analysis algorithms have been proposed by different groups in the remote sensing community.

Despite a plethora of time series analysis algorithms available in remote sensing, design and selection of algorithms for LULCC detection in remote sensing appears to be almost always context specific. Most of the methods proposed to date seem to perform well on the type of data that they are designed for. Their performance on randomly picked datasets from across the globe has not been studied. The onus of choosing an appropriate algorithm that will perform well on their particular dataset falls on the user. Unfortunately, no single algorithm designed so far seems to work for all datasets ([16]). For example, the Western Antarctica as well as the Greenland Ice Sheets are beginning to collapse due to global warming, the melting leading to continually receding snow covers at the respective locations. For these regions, using LULC algorithms based on periodicity assumptions is expected to lead to incorrect predictions and/or false alarms, although the nature and extent of this has not been studied yet. Even if there were no global warming, mild shifts in the 'phase' and 'amplitude' of seasons are known to take place ([61]). Time warping techniques ([61]) to deal with these issues may be helpful in some contexts, but their accuracy and scalability has not yet been satisfactorily investigated. Approaches based on periodicity and a moving window are possible, with additional computational costs.

A polyalgorithm is an effective strategy to unite the strengths and circumvent the weaknesses of multiple algorithms that are individually also designed to solve the same problem. The concept of polyalgorithm was introduced by Rice and Rosen (1966). A polyalgorithm uses a combination of several basic methods in a framework. Each of these basic methods is applicable to the same problem, with only their performance and/or success being different for different datasets (inputs). The construction of this framework involves experimenting with an increasingly heterogeneous set of situations to evolve a robust algorithm that is capable of choosing the subset of algorithms (from a collection of algorithms) suitable for a given input, and has performance metrics to integrate their outputs. The details of algorithm selection and processing stay hidden from the user. Polyalgorithms have been designed in the past for solving various problems, for example, nonlinear systems of equations ([70], [68], [69], [67]), matrix computations on parallel architectures ([52], [30]), and certain chemical models ([27]).

This work lays the foundation for a polyalgorithm for LULCC detection. Three currently existing, fundamentally different from each other, change detection algorithms are utilized. A similar work in this direction is [88], wherein a framework is developed to evaluate five

different algorithms on the input dataset, compare them based on certain scores, and then return the best results. Similar approaches are also gaining ground recently in the field of classification algorithms ([20], [46], [87]). Most recently, in [33], multiple change detection algorithms are utilized to build a decision trees based ensemble algorithm for LULCC.

The rest of this work is organized as follows: Chapter 2 presents background on state-of-the-art change detection algorithms available in remote sensing, puts them in the context of the general time series literature, and explains the choice of algorithms used in this work. Chapter 3 defines the notation. Chapters 4, 5, and 6 describe three different trend and change detection algorithms — EWMACD, BFAST, and LandTrendR; experimental results demonstrating the successes, failures, and sensitivity to parameters for each algorithm are presented. Prospects for a viable polyalgorithm are discussed in Chapter 7. Optimizations, parallelization, and scalability of codes for the three algorithms is discussed in Chapter 8. Chapter 9 concludes with an assessment and future work.

Chapter 2.

BACKGROUND

2.1. Remote Sensing Literature

Most of the LULC algorithms proposed in the remote sensing literature can be divided into two categories: *bitemporal analysis* and *temporal trajectory analysis*. Bitemporal analysis was more popular before 2008 (when the availability of satellite data to the public was very limited) and forms the classical way of analyzing images — these algorithms analyze changes occurring between two images (dates). The more preferred bitemporal algorithms rely on image differencing ([4], [31], [14], [15]), and linear transformations ([71], [58], [24], [25]). Other strategies used to design bitemporal algorithms include image ratioing ([37]), image regression ([41]), and composite analysis ([76]). Detailed surveys of these algorithms can be found in ([19], [54]). Multi-Index Integrated Change Analysis (MIICA) ([38]), a recent popular algorithm, utilizes two Landsat image pairs and four different derived spectral indices for change detection. The interested reader is referred to [11]for a further decent categorization of these algorithms.

For time series analysis based change detection algorithms, the popular strategy has been to design pixel based algorithms, wherein the time series for one pixel at a time is analyzed. One strategy is to segment the time series into piecewise linear segments. Specifically, the time span is partitioned into intervals where each interval corresponds to a sustained trend in observed values. The boundaries of these intervals correspond to points of change or the start of a new trend. The number of intervals depends on how many changes in trends occurred for that time series. This approach is adopted, for example, in ([17], [43], [39], [40], [79], [80]). In ([29]), on the other hand, seven shapes that can possibly occur in time series spectral data are identified. Constrained regression is done using splines that can generate these shapes. Some methods leverage the fact that climate related phenomena such as vegetation, temperatures, and the like are expected to follow a periodic pattern and utilize models based on Fourier series (trigonometric polynomials) ([7], [89]). In Vegetation Change Tracker (VCT) ([34]), another popular method, for *each* image, cloud, shadow, and water are first masked using histograms. A derived index based

on the mean and standard deviation of observed values of multiple bands in that image is calculated. (So, unlike LandTrendR, BFAST, and EWMACD, VCT does *not* work on normalized difference vegetation index(NDVI) values.) One (that with the best derived index) image per year is selected for further processing. Any masked values appearing in these selected images are filled in by interpolating two temporally nearest available values in the previous and subsequent years. Then a suite of decision rules is used to detect and classify forest disturbances. Wavelets were utilized in ([10]).

Data mining approaches have been proposed for classification and change detection ([28], [56], [60], [59], [82], [81]). In ([28]), a decision tree classifier was used to detect an outbreak of mountain pine beetle. This algorithm was originally implemented only on a subset of all available Landsat images (specifically, one scence per year was chosen from a 14 year span). In [59], sequential pattern mining was proposed for finding trends (and changes) in land cover; all images were utilized. Dynamic time warping (DTW) was proposed in [61]for comparing and analyzing remote sensing time series as well as characterizing change.

Improved trend approximation can be obtained if, instead of treating each pixel independently, information from nearby pixels (spatial information) is also utilized. One such approach is VeRDET ([35]), which utilizes two-dimensional total variation regularization (TVR) ([72]; [26]) to modify the images so that they have small-scale spatial patches (reduced spatial heterogeneity). TVR is then used again for fitting a piecewise linear polynomial to the time series of each pixel. In [60], each image in a stack is first segmented to generate region associated indices (in addition to the already existing spectral indices). Each pixel is thus characterized by both a spectral and a spatial index. Unsupervised classification algorithms are then used over this expanded set of indices for time series analysis and change detection.

2.2. The Time Series Literature

While research on algorithms for LULC using satellite image time series is relatively new, development and use of time series analysis started nearly a century ago, with applications such as econometrics, seismology, weather prediction, electrocardiography, mathematical finance, control systems, and more. This has led to a plethora of algorithms for analysis and forecasting in the time series literature. Autoregressive moving average (ARMA) models, introduced by Wold in 1938 [85], are polynomial models for representing any stochastic process and making predictions [84]. For nonstationary data, the nonstationarity is eliminated

by preprocessing the data. Specifically, the observed (input) values are replaced by the differences between their values and the previous values. The rest of the processing is carried out on this modified data using ARMA. This is known as the autoregressive integrated moving average (ARIMA) approach. The classic 1970 book by Box and Jenkins [9]presented their method — a full framework for analyzing time series. This framework is polynomial based applying either ARMA or ARIMA models to find the best fit to a given time series. Approaches based on Fourier transforms [1], wavelets [12], support vector machines [83], piecewise linear approximations [36], [47], [49], [45], and the references therein are other currently popular methods for approximating time series. Regardless of the formalization, all the time series algorithms in the literature fit in one of the following classes:

(i) Kernel regression methods. These methods represent the time series as a linear combination of basis functions. Typically, a linear system of equations is solved to determine the coefficients. Any analysis and predictions are done based on this representation. ARMA models (polynomial regression), Box-Jenkins models (polynomial regression), Fourier transform based approaches (trigonometric polynomials), and wavelet transforms would all classify as kernel regression.

(ii) Top-down approaches. In these algorithms, an approximation is first made to the whole time series. Then, typically using error estimate criteria, finer partitions of the time series are sought so that each new partition is a refinement of the previous partition. This is repeated until either a maximum number of iterations is reached or each segment of the partition satisfies a convergence criterion. The 'Iterative End Points Fits' algorithm ([66]) is an example of a top-down approach. Other top-down algorithms include [21], [22], [51], and [75].

(iii) Bottom-up approaches. Bottom-up approaches represent the most elementary units of the data first; on each iteration, increasingly larger (or coarser or more 'complex') structures are composed from the simpler structures and their evaluations in the previous iteration. Dynamic programming and all recursive algorithms classify as bottom-up algorithms, which are frequently implemented using backtracking.

Table 1 General time series literature classification of some remote sensing algorithms.

	Algorithms in remote sensing	Segmentation approaches in general time series literature
(*i*)	CCDC, EWMACD, SHAPE-SELECT--FOREST	Kernel regression method
(*ii*)	LandTrendR, VeRDET	Top-down approach
(*iii*)	BFAST, MIICA, VCT	Bottom-up approach

A list of some recent change detection algorithms in remote sensing and their classification is displayed in Table 1.

2.3. Our Approach

A strategy to construct a polyalgorithm would be to include algorithms that are fundamentally distinct from each other in terms of the phenomenon they capture as well as in construction. However, the suitability of available algorithms to capture different scenarios is currently not fully known. Therefore, choosing an algorithm from each of the classes in Table 1, ensuring some variation in type of phenomenon captured, may be used as a first step towards polyalgorithm construction. In this paper, LandTrendR, EWMACD, and BFAST are studied. LandTrendR, a top-down approach, generates a piecewise linear model to represent the input time series. A broad trend can therefore be captured. It is not expected to be sensitive to seasonal deviations/anomalies, though. For a stationary time series whose periodicity or variation changes, the performance of LandTrendR is yet unknown. EWMACD is a kernel regression method based on harmonic regression, designed to detect any persistent deviations from the stable pattern observed during a training period chosen by the user. This algorithm performs very well in regions where the land cover exhibits strong periodicity. However, its performance on time series with aperiodic variations and/or unstable training periods is limited. BFAST, a bottom-up approach, is more generic and models both linear trends and seasonal variations. BFAST's periodic linear model is more accurate than that of LandTrendR, since BFAST recursively evaluates the possibility of every single time point being a breakpoint, and then chooses the most

optimal set of breakpoints. Unfortunately, such an exhaustive search makes BFAST more expensive than other algorithms [73]. Also, the correctness of BFAST seasonal fits is yet to be tested. Finally, all of these algorithms were originally designed for forest covers and have only been tested on the same so far; their performance on nonforest topographies (e.g., urban developments, arid areas) is yet to be seen.

Sections 4, 5, and 6 elaborate on each of these three algorithms and present instances of their success and failure.

Chapter 3.

PRELIMINARIES

Notation and definitions. For an $m \times n$ matrix A, an n-vector x, $I \subset \{1, \ldots, m\}$, $J \subset \{1, \ldots, n\}$, let A_{IJ} denote the submatrix of A formed from the rows indexed by I and the columns indexed by J, and x_J denote the subvector of x indexed by J. $A_{I\cdot}$ ($A_{\cdot J}$) are the rows (columns) of A indexed by I (J), respectively. An image is an $R \times C$ matrix D, where each D_{rc} (pixel) is an $S \times B$ matrix, whose (s, b) element $(D_{rc})_{sb}$ is the signal value at time index s and frequency band index b. S' is the number of missing data values and $\bar{S}$ is the total number of timestamps provided in the data, i.e., $\bar{S} = S + S'$.

Input data.

Images taken by satellites Landsat 7 and 8 are used for the experiments presented in this paper. Landsat 7 is equipped with an enhanced thematic mapper (ETM+) instrument, which takes images at 30m resolution, using eight different spectral bands (one image per band). Landsat 8 employs an operational land imager (OLI) for the same task. Any given area of the Earth's surface is captured every 16 days. The experiments presented in this paper utilize normalized difference vegetation index ([78], [48])

$$\text{NDVI} = \frac{\text{NIR} - \text{R}}{\text{NIR} + \text{R}},$$

where NIR is the near infrared band (band 4) and R is the visual red band (band 3). In the remote sensing community, NDVI is widely considered to be a good metric for identifying the presence of vegetation cover. NDVI is directly related to the photosynthetic capacity of plant canopies. Broadly, forests typically tend to have high positive values (e.g., 0.6–0.8), scrubs/shrubs lean towards slightly smaller values, and any other topographies with smaller canopy cover (e.g., meadows, grazing areas) have even lesser values. Persistently decreasing NDVI values usually indicate decreasing foliage — which could be because of harvest (steep negative slope), insects (gentler negative slope), seasons (descending portions of periodic curves), or any other reason. Persistently increasing NDVI values indicate leaf

cover increasing. For the experiments presented here, negative values of NDVI are deemed irrelevant as they correspond to water, clouds, or missing observations, and are masked out.

Reference data.

TimeSync data, a dataset that is based on human interpretation of Landsat time series, is used for validating the results for experiments in 1D. This dataset is prepared using TimeSync Landsat images visualization and change data collection tool ([17]). This tool enables disturbance characterizations for pixel-level samples of Landsat time series data, relying on human interpretations of change as viewed in image chip series, spectral index trajectories, high spatial resolution image temporal snapshots from Google Earth, and other supporting products. The current dataset was built using Landsat image stacks belonging to six different path/rows (scenes) and spanning the years 1984 to 2014. From each scene, 300 pixels were chosen with random sampling, and without regard to land cover. There are 1800 change pixels in the dataset. For each of these pixels, the following attributes were noted: occurrence of disturbance, the first year of detection (a year between 1986 and 2011), the duration for gradual disturbances (in number of years), and the causal agent class (harvest, fire, mechanical, decline, wind, other).

Evaluation. The three algorithms presented in this paper are applied to NDVI data. TimeSync dataset is used as reference data set. Due to focus on change detection, only experience with change pixels is discussed in this paper. The time period of 2000–12 is utilized. Success/failure of each algorithm is compared to TimeSync data by checking the distrurbance year(s) stated in the dataset with the year(s) predicted in the outcome of respective algorithms. Note that TimeSync itself is an interpreted dataset, not ground truth. For the pixels explicitly displayed in this paper, visual comparison of outcome with trajectory is also done. Published values of parameters are used. Sensitivity of algorithms to parameters is also presented. In Section 7, mathematical distances (Hausdorff distances) are utilized to further provide quantify the differences in outcomes of different algorithms.

Chapter 4.

EXPONENTIALLY WEIGHTED MOVING AVERAGE CHANGE DETECTION

Exponentially weighted moving average change detection (EWMACD), proposed in ([6], [7]), is a kernel regression approach modeling the time series as a linear combination of trigonometric polynomials. The model is trained over data collected in the initial two (or more) years. When the observations in the subsequent years deviate from the values forecast by the model for a 'substantial' length of time (persistence), a change is declared (recovery or disturbance). The training period as well as the persistence are parameters of the algorithm. A positive flag is raised in instances of sustained growth while sustained losses are indicated by negative flags. EWMACD is able to capture seasonal changes, but is also very sensitive to several algorithm parameters.

Algorithm EWMACD.

for band $b := 1$ **step** 1 **until** B
  for row $r := 1$ **step** 1 **until** R
    for column $c := 1$ **step** 1 **until** C **do**
      Step 1: Write the time series data in the column $(D_{rc})_{\cdot b}$ as

$$(D_{rc})_{\cdot b} = \begin{pmatrix} u \\ v \end{pmatrix},$$

where the M-dimensional vector u is deemed training data and the $(S - M)$-dimensional vector v as the test data. Let

$$X = \begin{bmatrix} 1 & \sin t_1 & \cos t_1 & \cdots & \sin Kt_1 & \cos Kt_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \sin t_M & \cos t_M & \cdots & \sin Kt_M & \cos Kt_M \end{bmatrix}$$

be the Gram matrix for the time points t_1, ..., t_M, using K harmonics, where $M > 2K + 1$. The least squares fit to the training data u is then written as

$$u(t) = \alpha_0 + \sum_{i=1}^{K} (\alpha_{2i-1} \sin it + \alpha_{2i} \cos it)$$

with coefficients

$$\alpha = \left(X^t X\right)^{-1} X^t u$$

and residual

$$E(\alpha) = u - X\alpha.$$

REMARK 1. In practice α is computed via a QR factorization of X, not by computing $(X^t X)^{-1}$ explicitly.

Next let

$$I = \{i \mid |E(\alpha)_i| < \gamma_1\},$$

where γ_1 is a user defined threshold and $|I| > 2K + 1$. Calculate the coefficients for an improved fit to the underlying signal as

$$\alpha^* = \left((X_{I.})^t X_{I.}\right)^{-1} (X_{I.})^t \, u_I.$$

With the refined coefficients α^*, calculate the residuals for

(i) the complete time series $(D_{rc})_{.b}$ as

$$E^*(\alpha^*) = (D_{rc})_{.b} - \bar{X}\alpha^*,$$

where $\bar{X}_{s.} = (1, \sin t_s, \cos t_s, \ldots, \sin Kt_s, \cos Kt_s)$, for $s = 1, \ldots, S$.

(ii) the outlier-free time series as $\left(E^*(\alpha^*)\right)_{\bar{I}}$, where $\bar{I} = \{s \mid |E^*(\alpha^*)_s| < \gamma_2\}$, γ_2 is a user defined threshold, and

(iii) the outlier-free training set $\hat{I} = \bar{I} \cap \{1, \ldots, M\}$ as

$$\left(E^*(\alpha^*)\right)_{\hat{I}} = u_{\hat{I}} - X_{\hat{I}.}\alpha^*,$$

where $|\hat{I}| > 2K + 1$.

REMARK 2. In the present implementation,

$$\gamma_2 = \begin{cases} 1.5\eta, & i \in [1, M], \\ 20\eta, & i \in (M, S], \end{cases}$$

where η is the standard deviation of the first M elements of the residual vector $E^*(\alpha^*)$.

Step 2: Define the control limit vector τ by

$$\tau_i = \mu + \sigma L \sqrt{\frac{\lambda}{2-\lambda}\left(1 - (1-\lambda)^{2i}\right)},$$

for $i = 1, 2, \ldots, |\bar{I}|$, where $\mu = 0$ is used here, σ is the standard deviation of the outlier-free training data errors $\left(E^*(\alpha^*)\right)_{\hat{I}}$, L is the multiple of this standard deviation σ, and $\lambda \in (0, 1]$ is the weight given to the most recent residual in the exponentially weighted moving average (EWMA) defined next. L is typically set to 3 or slightly smaller depending on the value of λ.

Step 3: Let $\bar{I} = \{j_1, j_2, \ldots, j_{|\bar{I}|}\}$, $j_1 < j_2 < \cdots < j_{|\bar{I}|}$. Define the vector z by

$$z_1 = \big(E^*(\alpha^*)\big)_{j_1},$$
$$z_i = (1-\lambda)z_{i-1} + \lambda\big(E^*(\alpha^*)\big)_{j_i}, \quad i = 2, \ldots, |\bar{I}|.$$

This is the exponentially weighted moving average (EWMA) of the residual $\big(E^*(\alpha^*)\big)_{\bar{I}}$.
Step 4: Define the flag history S-vector f by

$$f_s = \begin{cases} \operatorname{sgn}(z_i)\big\lfloor |z_i/\tau_i| \big\rfloor, & s = j_i \in \bar{I}, \\ 0, & \text{otherwise.} \end{cases}$$

If there is a run of $+1$ or -1 in the values $\operatorname{sgn}(\Delta f_s) = \operatorname{sgn}(f_{s+1} - f_s)$ of length ϖ, called the 'persistence', signal a change at the index s beginning the (nonzero) run.

REMARK 3. Missing data is automatically handled by not assuming that the time points t_i are equally spaced. Alternatively, missing data for time point t_k can be handled by including t_k in the sequence $(t_1, t_2, \ldots, t_S)$, but excluding t_k from the training sequence $(t_1, t_2, \ldots, t_M)$ and k from the sets I, $\bar{I}$, and $\hat{I}$, which is equivalent to treating $(D_{rc})_{kb}$ as an outlier and to setting the flag $f_k = 0$. **end**

end

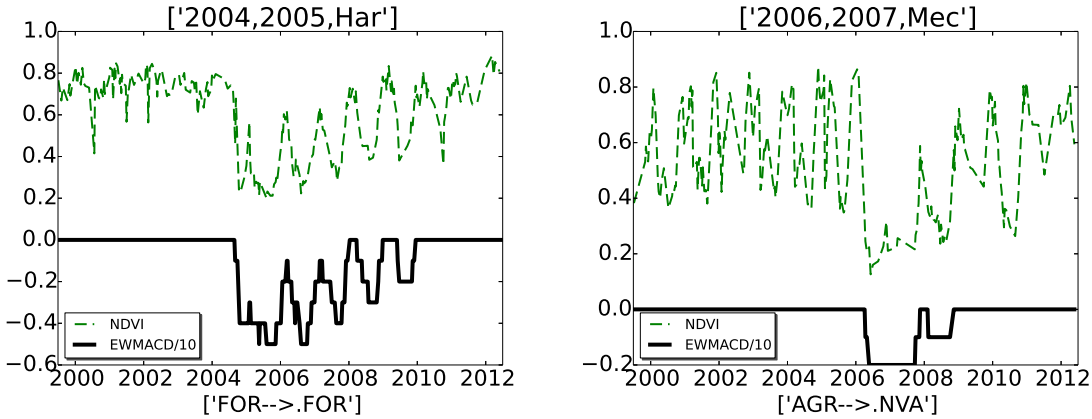
end

The complexity of this approach is $\mathcal{O}\big((2K+1)^2 S\big)$. EWMACD outcomes are expected to be zeros when the time series trajectory matches the trajectory in the training period. Non-zeros are expected to appear when the trajectory deviates from the training period for a substantial length of time (determined by control chart parameters and the persistence). The first timepoint when the outcome trajectory deviates from the prior constant (or zero) value is the estimated date of disturbance, the subsequent nonzero sequence of values provides recovery (or, loss) period information. A sharp loss (e.g., harvest) is expected to manifest as a sudden drop in the outcome as well. A gentler loss (e.g., mountain pine beetle) is expected to appear in the form of the outcome trajectory gradually drifting away from zero (with an overall negative slope). Similarly, recovery will appear in the form of the trajectory following an overall positive slope, revealing recovery period information. In Edyn ([8]), an improved version of EWMACD that retrains data after a disturbance is sensed, is proposed (this development not implemented here). In any case, since the training data in the postshock period is unstable, the training period of Edyn after the first break is expected to be unstable. For all the experiments presented in this work, the first two years

of data is used for training EWMACD. The rest of the parameter values for EWMACD are as follows: $K = 2$, $L = 0.5$, $\lambda = 0.3$, $\varpi = 7$.

Success. Figure 1 displays the outcome of EWMACD on NDVI data for two pixels. Figure 1(left) has a stable trajectory with high NDVI values until 2004. The NDVI values drop suddenly in 2004 from 0.8 to 0.2, indicating the possibility of an event. Gradual recovery is seen after that. Per TimeSync data, this is a forest pixel where harvest occurred in 2004–05. The trajectory and the TimeSync information are, thus, in agreement for this pixel. Using the published parameters, EWMACD captures the occurrence and timing of the harvest accurately. The nonzero portion of the curve following this until 2010 displays the recovery period information, which exhibits seasonal effects, likely related to the relatively flat training curve.

In Figure 1(right), a stable trajectory with mean 0.6 and much variance is seen until 2006. In 2006, the values drop to 0.2 (perhaps very little leaf cover left). Subsequently, some leaf cover is regained but the trajectory is different from the pre-2006 trajectory. TimeSync data states that this pixel is used for agriculture initially, is cleared in 2006–07, and used for nonvegetated anthropogenic purposes after that. The trajectory and the TimeSync data are in agreement. EWMACD accurately detects the sudden absence of vegetation cover. Gradual increase in NDVI is indicated thereafter.



Forest → Forest

Harvest in 2004

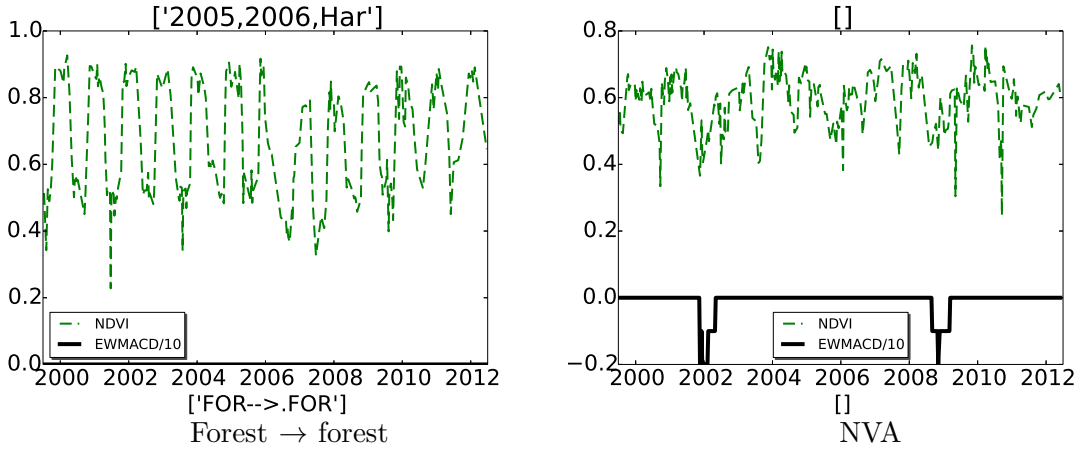
Agriculture → NVA

Clearing in 2006

Figure 1. EWMACD success. Flag history (divided by 10) signals times of change by runs of increases or decreases in the flags. Harvest in a forest (left) and fire in a nonforest pixel (right).

Failure. Failure of EWMACD when the training period is not stable is trivial. Instances of failure when the training period is stable are discussed here. Figure 2(left) displays a forest pixel. The NDVI trajectory shows a disturbance in 2006. TimeSync data states a harvest in 2005–06. EWMACD misses this event. The outcome does not change with different parameters either, possibly because the change is relatively brief compared to the training data.

In Figure 2(right), the trajectory appears stable except for two visible drops in 2002 and 2008. TimeSync data classifies this pixel as one with nonvegetated anthropogenic land cover, although the NDVI values appear high for the land cover to be so. There is no recorded event in the 2000–12 time period, thus there is potential disagreement between the trajectory and the TimeSync data. EWMACD shows two instances of loss, which seem to agree with the trajectory but not so with TimeSync. The second signal (in 2008–09) vanishes when three harmonics are used instead of two, indicating that this trajectory could be following time periods that are different from those used in EWMACD. The first signal, however, remains through a range of parameters, and could potentially be attributed to an insufficient training period.



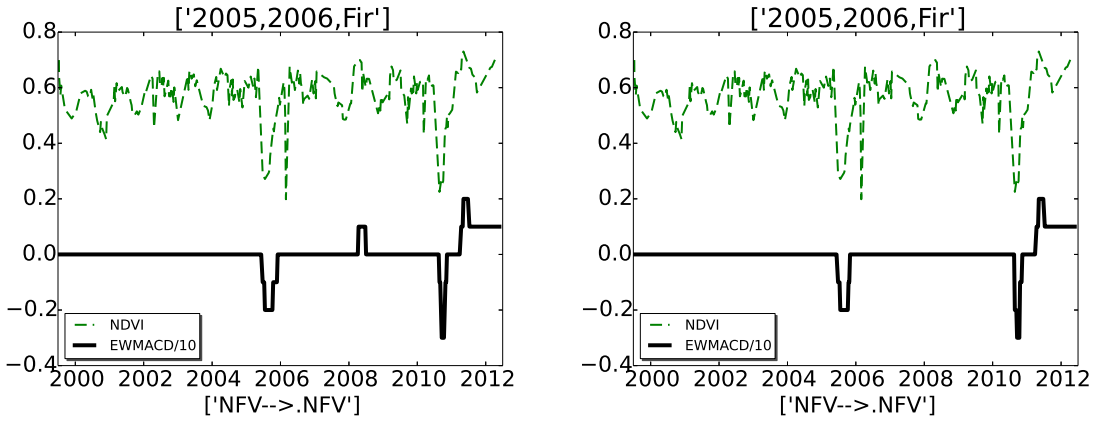
Forest → forest

Harvest in 2005–06

NVA

No recorded change

Figure 2. EWMACD failure. Misses a harvest in a forest (left); false alarms in a developed area (right).



$\lambda = 0.5$ $\lambda = 0.58$

Figure 3. EWMACD sensitivity. A false alarm appears for $\lambda = 0.5$, but not for $\lambda = 0.58$. For $\lambda = 0.6$, the second event also disappears.

Sensitivity. The accuracy of EWMACD is expected to be influenced by the parameters L and λ used for the control limits calculation (Step 2 of the algorithm), the persistence parameter (ϖ), and the training period. Sensitivity to λ is presented here. The forest pixel displayed in Figure 3 is where a fire is known to have occurred in 2005–06. The NDVI trajectory shows a drop in 2005–06. For $\lambda = 0.5$ (Figure 3(left)), EWMACD detects the fire in 2005–06. In addition, it shows a second flag in 2008, and a third flag in 2010–11 followed by recovery. The second flag is likely a false alarm. The third flag corresponds to a visible dip (potentially, an event not recorded) in NDVI at that time. The false alarm is not appear for $\lambda = 0.58$ (Figure 3(right)). On using $\lambda = 0.6$, the flag in 2010–11 also disappears (which may make the outcome agree with TimeSync data but not with the trajectory). Using $\lambda = 0.4$, on the other hand, produces more false alarms between 2000 and 2006.

In general, decreasing L and ϖ has the effect of increasing the sensitivity of the algorithm. This means that the number of false alarms may also increase. λ determines the restrospectiveness of the algorithm. Decreasing λ should mask abrupt changes (the kind displayed in examples here) in favor of highlighting subtle, chronic changes (the kind TimeSync might also pick up). Finally, since this algorithm utilizes a training period, it is important that there be no change during the training period. When the training period does happen to contain a change, the signals' signs are expected to skew in the direction opposite to the direction of 'the change during the training period', but this needs to be investigated further. In Edyn ([8]), an improved version of EWMACD, the harmonic model coefficients and the persistence are dynamically updated. Specifically, when EWMACD finds an event, the following (two) years are used to retrain the model, and the updated model is used for sensing change thereafter. This may alleviate the postchange false alarms to some extent.

Chapter 5.

LANDTRENDR

LandTrendR was proposed in ([44]). Starting with an ordinary least squares fit to the entire time series, LandTrendR partitions the time series step by step, adding breakpoints (called vertices here) at each step. A set of potential breakpoints is thus generated in a straight top-down approach. Once these vertices have been generated, they are refined in multiple passes: (i) first the least influential vertices (corresponding to most obtuse angles) are discarded; (ii) then, of the now remaining vertices, each vertex is dropped one by one, based on a continuous piecewise linear fit (a 'model') using anchored regression (anchored regression comprises doing a least squares fit to the data but with one end fixed and only the slope (or the second end) to be determined); and (iii) the goodness of each model is evaluated in terms of F-statistics ('improvement compared to the *mean* model'). Of all the potential models, the model with lowest p value of the F-statistic is chosen as the final model. LandTrendR's original implementation utilized only one image per year from the Landsat image stacks, out of the 23 or so available per year. However, in this work, all available images are utilized.

Algorithm LandTrendR.

for band $b := 1$ **step** 1 **until** B **do**
 for row $r := 1$ **step** 1 **until** R **do**
  for column $c := 1$ **step** 1 **until** C **do**
  Step 1: Despike Let $u = (D^0_{rc})_{\cdot b}$ denote the raw time series data. For each time point t_i, $1 < i < S$, define $\Delta u_i = (D^0_{rc})_{(i+1)b} - (D^0_{rc})_{ib}$, $\nabla u_i = (D^0_{rc})_{ib} - (D^0_{rc})_{(i-1)b}$, $\mu\delta u_i = (D^0_{rc})_{(i+1)b} - (D^0_{rc})_{(i-1)b}$, $k_i = 1 - |\mu\delta u_i| / \max\{|\nabla u_i|, |\Delta u_i|\}$, and correction

$$\kappa_i = (\delta^2 u_i)k_i/2 = \big((D^0_{rc})_{(i-1)b} - 2(D^0_{rc})_{ib} + (D^0_{rc})_{(i+1)}\big)k_i/2.$$

  For each i such that $k_i = \max\limits_{1<j<S} k_j$, update $(D_{rc})_{ib} := (D^0_{rc})_{ib} + \kappa_i$. Repeat iteratively until $\max\limits_{1<j<S} k_j < \upsilon$, some given despiking tolerance.

[1] also called intercept-only model, intercept being equal to the average of the observations under consideration.

Step 2: Find potential breakpoints Let $S^1 = (t_1, \ldots, t_S)$ be the original sequence of time points and $I^1 = (2, \ldots, S-1)$ denote the corresponding sequence of interior indices. Let

$$X = \begin{pmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_S \end{pmatrix}$$

be the Gram matrix for the time points $t_1, \ldots, t_S$, for ordinary least squares linear regression. The least squares fit to this data is given by

$$u(t) = \alpha_0 + \alpha_1 t$$

with coefficients

$$\alpha = (X^t X)^{-1} X^t u$$

and residuals

$$E^1(\alpha) = u - X\alpha.$$

Find the smallest index i_1 corresponding to the maximum absolute deviation, i.e.,

$$i_1 = \min\big\{ i \mid i \in I^1 \text{ and } |E^1(\alpha)_i| = \max_{j \in I^1} |E^1(\alpha)_j| \big\}.$$

Split the sequence S^1 into two subsequences,

$$S^1_l = (t_1, \ldots, t_{i_1}) \text{ and } S^1_r = (t_{i_1}, \ldots, t_S).$$

Do linear regression on each of these and compute their respective mean squared errors, MSE_l and MSE_r. Suppose $|\text{MSE}_l| \leq |\text{MSE}_r|$. Then let $S^2 = S^1_r$ with interior index set $I^2 = (i_1 + 1, \ldots, S-1)$ be the next candidate sequence for 'breakpoint search'.
Again, find the smallest index i_2 corresponding to the maximum absolute deviation

$$i_2 = \min\big\{ i \mid i \in I^2 \text{ and } |E^2(\alpha^2)_i| = \max_{j \in I^2} |E^2(\alpha^2)_j| \big\}.$$

Again, split S^2 into two subsequences $S^2_l = (t_{i_1}, \ldots, t_{i_2})$ and $S^2_r = (t_{i_2}, \ldots, t_S)$, compute the least squares fit for each of these, choose the interval with higher MSE, and find the index i_3 corresponding to maximum absolute deviation. Recursively apply the algorithm until there are $\mu + \nu + 1$ breakpoints (including t_1 and t_S), where μ is the maximum number of segments allowed and ν is the maximum number of vertex overshoots (see Step 3 below) allowed. (In the rare circumstance that $\text{MSE}_l = \text{MSE}_r = 0$ at some iteration, there may be fewer than $\mu+\nu+1$ breakpoints.) Let $\bar{S} = (t_1, t_{i_1}, \ldots, t_{i_{\mu+\nu-1}}, t_S)$ be the final sequence of (sorted) breakpoints thus obtained, $\bar{I} = (1, i_1, \ldots, i_{\mu+\nu-1}, S)$ and $\bar{V} = (v_1, v_{i_1}, \ldots, v_S)$ be the corresponding index and 'vertex' sequences, where $v_i = (t_i, u_i)$.

Step 3: Cull by angle change

Define the sequence of angles

$$\alpha_j = \arccos\left(\frac{(v_{\bar{I}_j} - v_{\bar{I}_{j-1}}) \cdot (v_{\bar{I}_{j+1}} - v_{\bar{I}_j})}{\|(v_{\bar{I}_j} - v_{\bar{I}_{j-1}})\|\|(v_{\bar{I}_{j+1}} - v_{\bar{I}_j})\|}\right), \text{ for } j = 2, 3, \ldots, \mu + \nu.$$

Find $\bar{a} = \min\{i \mid \alpha_i = \min_j \alpha_j\}$, delete $v_{\bar{a}}$ from the sequence $\bar{V}$, and recalculate from this the angles with the new vertices. Repeat until reaching the sequence $V^* = (v_1, v_{l_1}, \ldots, v_{l_{\mu-1}}, v_S)$ with index sequence $L^* = (1, l_1, \ldots, l_{\mu-1}, S)$.

Step 4: Fit trajectories

Moving from $i = 1, \ldots, \mu$, consider consecutive vertices $v_{L_i^*}$, $v_{L_{i+1}^*} \in V^*$, one at a time, and an anchored regression fit

$$u_{AR}(t) = y_{L_i^*} + \alpha(t - t_{L_i^*}),$$

where $y_{L_i^*}$ is the 'fitted' value inferred from the fit in the preceding interval $\left(t_{L_{i-1}^*}, t_{L_i^*}\right)$, α is the solution to the least squares regression problem $u_{AR} \approx u$ at the points $t_{L_i^*+1}, \ldots, t_{L_{i+1}^*}$. For the special case $i = 1$, the coefficient $y_{L_1^*}$ is also estimated. The final result of this step will be a continuous piecewise linear function $P^*(t)$ covering the full domain. Further, let $Y^* = (y_1, y_{l_1}, \ldots, y_{l_{\mu-1}}, y_S)$ be the sequence of fitted values at the breakpoints with indices L^*. Call the tuple $\mathcal{M}^* = (P^*(t), L^*, V^*)$ a regression model. In addition, let $(y_1, y_2, \ldots, y_S)$ be the sequence of fitted values over all time points in S^1 as predicted by $\mathcal{M}^*$.

Step 5: Model statistics

The improvement in prediction from regression compared to the mean model is given by the random variable

$$X_1^2 = \sum_{i=1}^{S}(u_i - \bar{u})^2 - \sum_{i=1}^{S}(u_i - y_i)^2 = \sum_{i=1}^{S}(y_i - \bar{u})^2,$$

where $\bar{u}$ is the mean value of the observations. The squared distance of the observed values from the values predicted by the regression model is the random variable

$$X_2^2 = \sum_{i=1}^{S}(u_i - y_i)^2.$$

Assuming that $y_i - \bar{u}$ and $u_i - y_i$ are independent, normally distributed, and have variance one, X_1^2 and X_2^2 have a χ^2 distribution with degrees of freedom $d_1 = \mu$, $d_2 = S - \mu - 1$, respectively. Therefore, the ratio

$$F = \frac{X_1^2/d_1}{X_2^2/d_2}$$

has an F-distribution and F-statistics can be used for measuring the 'goodness' of fit of the regression model. Let f be the F-statistic for model $\mathcal{M}$. Calculate the p-value of this F-statistic:

$$Q(f|d_1, d_2) = 1 - I_{\frac{d_1 f}{d_1 f + d_2}}\left(\frac{d_1}{2}, \frac{d_2}{2}\right) = I_{\frac{d_2}{d_2 + d_1 f}}\left(\frac{d_2}{2}, \frac{d_1}{2}\right).$$

$Q(f|d_1, d_2)$ is the probability that $F > f$ and $I_x(a, b)$ denotes the regularized incomplete Beta function given by

$$I_x(a, b) = \frac{1}{B(a, b)} \int_0^x t^{a-1}(1-t)^{b-1} dt, \qquad a > 0, \quad b > 0,$$

where

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} = \int_0^1 t^{a-1}(1-t)^{b-1} dt$$

is the (complete) Beta function, and

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt, \qquad \Re\, z > 0$$

is the Gamma function.
For model $\mathcal{M}^{(\mu)} = \mathcal{M}^*$, where the superscript corresponds to the number of segments in the model, let the p-value calculated in this step be $p^{(\mu)}$.

Step 6: Generate more (simpler) models
Begin with the model $\mathcal{M}^{(\mu)} = (P^{(\mu)}, L^{(\mu)}, V^{\mu)}) = (P^*(t), L^*, V^*)$.

(i) Assume that a disturbance always corresponds to a positive slope while a negative slope indicates recovery.

First look for negative slopes at interior vertices. Let $V^{(\mu)}_{i^{(\mu)}} \neq V^{(\mu)}_1$ be the left vertex (leftmost in case of a tie) of the segment with steepest negative slope. Delete $L^{(\mu)}_{i^{(\mu)}}$ from the index sequence $L^{(\mu)}$, giving the shorter sequence $L^{(\mu-1)}$.

However, if no negative slopes are found or, if the steepest negative segment happens to be the leftmost segment of the current model, then for each interior vertex $v^{(\mu)}_{L_i}$ in the model, consider the point to point connect using the vertices immediately to the left and right of $v^{(\mu)}_{L_i}$, i.e., $v^{(\mu)}_{L_{i-1}}$ and $v_{L^{(\mu)}_{i+1}}$:

$$u^{(\mu)}_{PP}(t) = \frac{y_{L^{(\mu)}_{i+1}} - y_{L^{(\mu)}_{i-1}}}{t_{L^{(\mu)}_{i+1}} - t_{L^{(\mu)}_i}}(t - t_{L^{(\mu)}_{i-1}}) + y_{L^{(\mu)}_{i-1}}, \quad i \in L^{(\mu)} \setminus \{1, S\},$$

and calculate the $\mathrm{MSE}_{L^{(\mu)}_i}$ for vertex $v_{L^{(\mu)}_i}$ as

$$\mathrm{MSE}_{L^{(\mu)}_i} = \frac{1}{t_{L^{(\mu)}_{i+1}} - t_{L^{(\mu)}_{i-1}}} \sum_{L^{(\mu)}_{i-1} \le k \le L^{(\mu)}_{i+1}} \left(u^{(\mu)}_{PP}(t_k) - u_k\right)^2, \quad i \in L^{(\mu)} \setminus \{1, S\}.$$

Then define $i^{(\mu)} = \min\{i \mid \mathrm{MSE}_{L_i^*} = \min_j \mathrm{MSE}_{L_j^*}\}$, the index of the vertex dropping which leads to least MSE. Delete $L_{i^{(\mu)}}^{(\mu)}$, resulting in the shorter sequence $L^{(\mu-1)}$.

(ii) Remove the corresponding vertex from $V^{(\mu)}$ giving $V^{(\mu-1)}$, and as in Step 4 generate the new piecewise linear fit $P^{(\mu-1)}(t)$ and model $\mathcal{M}^{(\mu-1)} = \big(P^{(\mu-1)}(t), L^{(\mu-1)}, V^{(\mu-1)}\big)$.

(iii) Calculate the p-value $p^{(\mu-1)}$ for this model.

Proceeding in this way, generate a total of μ models $\mathcal{M}^{(i)}$, $i = \mu, \ldots, 1$.

Step 7: Pick best model $\mathcal{M}_{i^*}$.
Let i^* be the smallest index i corresponding to the models $\mathcal{M}^{(i)}$ whose p-value is less than a user defined recovery threshold τ, i.e., $i^* = \min\{i \mid p^{(i)} \le \tau,\ i = 1, \ldots, \mu\}$.
REMARK. Check the linear segment slopes. If, for any model $\mathcal{M}^{(i)}$ under consideration, the recovery (to a global baseline) happens quicker than the quickest disturbance ϱ (from a global baseline), that model is discarded.

Step 8: Alternate approach.
If no models are found using Steps 1–7, repeat Steps 4–7 with the following modifications:
Step 4′. Instead of computing the continuous piecewise linear approximation $P^*(t)$ one segment at a time, going from left to right, compute $P^*(t)$ using all the data at once. This is done by expressing $P^*(t)$ as a linear combination of B-splines of order 2 with knot sequence $(t_1, t_1, t_{l_1}, t_{l_2}, \ldots, t_{l_{\mu-1}}, t_S, t_S)$, and then solving a linear least squares problem for the $\mu+1$ coefficients of these B-spline basis functions. (Note that there is no need to use the Levenberg-Marquardt algorithm as proposed in ([44])).
Step 6′. Skip directly to the point to point connect approach, without looking for negative slopes at all.

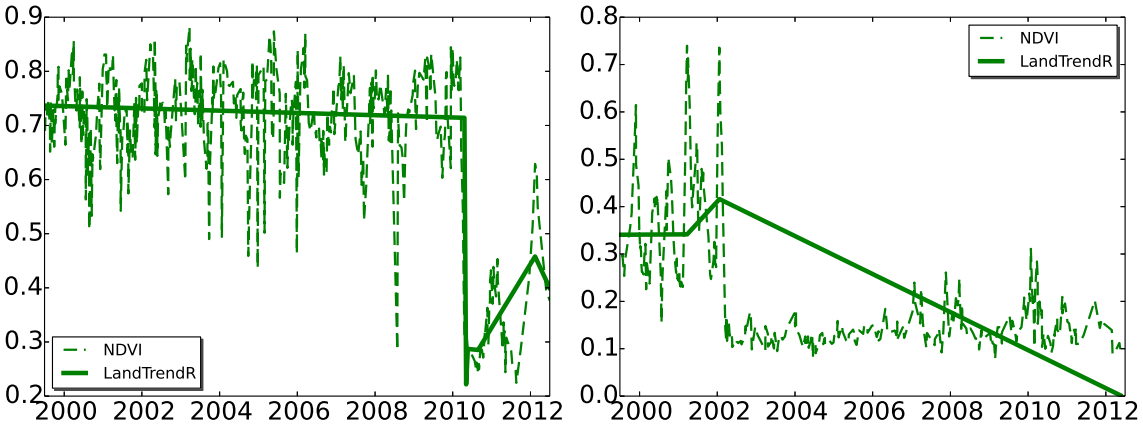
end
end
end

The complexity of the algorithm is $\mathcal{O}(n_b S)$, where $n_b = \mu + \nu - 1$. LandTrendR output consists of a set of breakpoints as well as a fit to the data. The fit is always continuous piecewise linear. The fit is displayed in the results discussed here. Since the fit is piecewise linear, the breakpoints are obvious. The following parameter values are used for LandTrendR in this work: $\upsilon = 0.9$, $\mu = 6$, $\nu = 3$, $\tau = 0.2$, $\varrho = 1.0$.

Success. Figure 4 presents examples of LandTrendR success. The pixel of Figure 4(left) is forest, per TimeSync data. In the 12 year period 2000–2012, one significant instance of harvest is recorded in 2010–11. The NDVI trajetory stays stable until 2010, and then a sharp fall is seen in the NDVI values in 2010–11, matching the harvest. LandTrendR

captures the trajectory accurately, indicating stability until 2010, a sharp loss in vegetation cover in 2010, followed by gradual recovery.

Figure 4(right) has a trajectory with mean 0.3–0.4, high variance until 2002. After 2002, the NDVI values are almost constant, close to 0.1, indicating absence of any vegetation cover. TimeSync data indicates that this pixel is an agricultural area that is cleared in 2002–03, and used for nonvegetated anthropogenic purposes starting 2003. The trajectory and the TimeSync information are thus in agreement. LandTrendR indicates some variation in trend pre-2002 and then successfully captures the 2002–03 harvest.



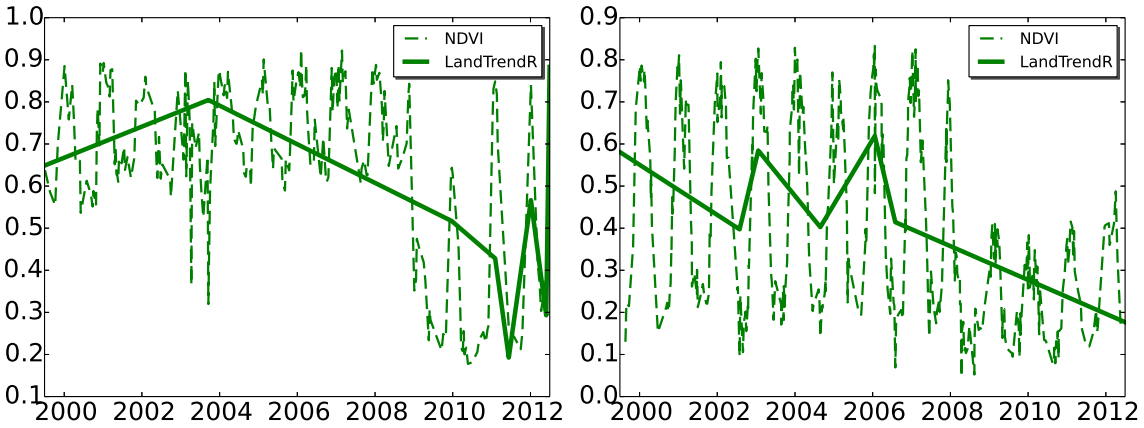
Forest → forest
Harvest in 2010–11

Agriculture → NVA
Harvest in 2002–03

Figure 4. LandTrendR success.

Failure. The two pixels of Figure 5 present instances of LandTrendR failure. Per TimeSync data, Figure 5(left) is a forest. The trees are harvested sometime in between 2008–10, and the land is subsequently used for agricultural purposes. The NDVI trajectory exhibits totally different mean and variance before and after 2009. In 2009, a sharp drop in NDVI values is seen, confirming the TimeSync information. The LandTrendR approximation shows an overall decrease in NDVI values between 2004 and 2010, but there is no breakpoint at the 2009 drop. The breakpoints that it does predict appear to be mostly false alarms.

Figure 5(right) is for a nonforest vegetation area. TimeSync states a mechanical change in 2008–10, after which the area is used for nonvegetated anthropogenic purposes, consistent with the NDVI trajectory: the mean NDVI value has a sharp drop in 2008, and the trajectory



Forest → agriculture
Mechanical in 2008–10

NFV → NVA
Mechanical in 2008–10

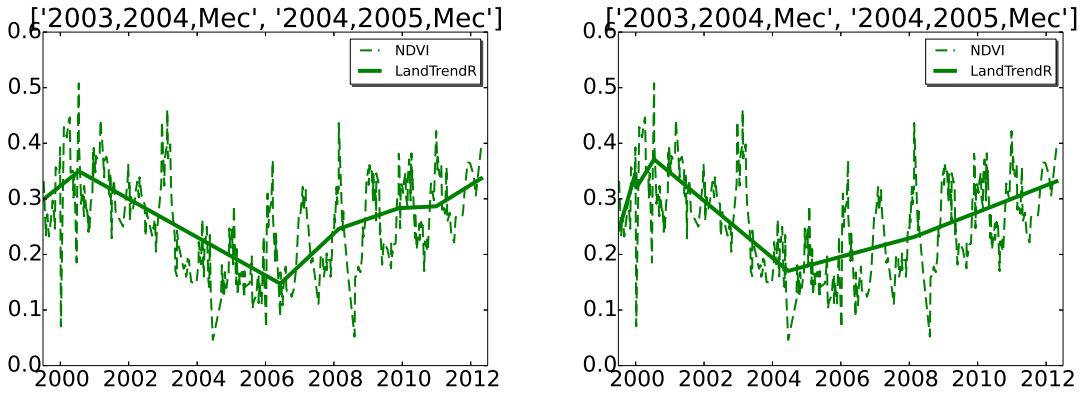
Figure 5. LandTrendR failure.

after that also has much lower variance. LandTrendR senses a decrease in vegetation cover but there is no breakpoint at the time where the sharp drop occurs. The trend, in general, does not match the trajectory.

Sensitivity. LandTrendR extensively harnesses least squares fitting. The determination of the initial breakpoint set is based on iteratively finding indices where deviation of the least squares fit to the data in certain intervals is maximum (cf. Step 2). Since least squares fit can be easily affected by outliers, this methodology for initial breakpoint set construction is also prone to being affected by outliers. In particular, sometimes points on only one side of a disturbance make it in to $\bar{S}$, the sequence of initial vertices, and even these may get dropped in reaching the sequence V^*. This makes LandTrendR's success in breakpoint determination sensitive to outliers. Of relevance to the polyalgorithm are also the parameters μ and ν. With smaller values of these parameters, 'localization' of breakpoints appears to be less prevelant.

Figure 6 shows the final outcomes of LandTrendR for two values of the parameter υ. TimeSync states nonforest vegetation type for this pixel prior to 2003, a mechanical event in 2003–04, conversion of land cover type to urban after that, a second mechanical event in 2004–05, and continued urbanized land cover for the rest of the time. The NDVI values start quite small (0.3) and stay low throughout. The changes are not evident in the trajectory, although a gradual decline in NDVI values is seen from 2002 to 2006, and less noise is seen after 2006. Figure 6(left) corresponds to $\upsilon = 0.8$. LandTrendR gets the overall approximation to the trend correct but misses both the breakpoints for $\upsilon = 0.8$. Figure 6(right) corresponds to $\upsilon = 0.9$. For $\upsilon = 0.9$, a breakpoint is placed in 2004–05 though the 2003–04 breakpoint is still missing.

As of now, LandTrendR has not shown much sensitivity to other parameters.



$\upsilon = 0.8$ $\upsilon = 0.9$

Figure 6. LandTrendR sensitivity.

Chapter 6.

BREAKS FOR ADDITIVE AND SEASONAL TRENDS

BFAST ([79]) decomposes the given time series iteratively into three components: trend, seasonal, and noise. BFAST computes and evaluates least squares fits in windows of increasing size. Qualitatively, (i) first the possibility of there being any structural change in the given time series is determined by computing the partial sums of residuals of least squares fits in windows (OLS-MOSUM). The limiting process of these partial sums is the increments of a Brownian bridge process ([13]). If the observations do have a structural change, an ordinary linear least squares fit will result in large residuals and, hence, in large partial sums. Therefore, the occurrence of large values in the process is an indication of the presence of a structural change — this probability being calculated from the Brownian bridge table. (ii) If a structural change is indicated, a search for change location is done. *Each* interior time point t is considered a breakpoint (change location) candidate. A recursive residual is the error at time t_j from the linear least squares fit over the window $[t_i, \ldots, t_{j-1}]$. The breakpoints (change locations) are chosen so as to minimize the sum of squared recursive residuals over all windows in between (omitting) the breakpoints. This is done for both trend and seasonal components of the time series, consecutively.

Algorithm BFAST.

for band $b := 1$ **step** 1 **until** B **do**
 for row $r := 1$ **step** 1 **until** R **do**
  for column $c := 1$ **step** 1 **until** C **do**
   begin
   Let $T = (t_1, \ldots, t_S)$ be the sequence of given time points and the S-vector u denote the time series data in the column $(D_{rc})_{\cdot b}$, i.e., $u = (D_{rc})_{\cdot b}$. Assume that the general model is of the form

$$u = \mathcal{V} + \mathcal{W} + \epsilon,$$

   where $\mathcal{V}$ and $\mathcal{W}$ denote the iteratively computed trend and seasonal components, respectively, present in the data and ϵ is the noise. The trend $\mathcal{V}$ may be piecewise linear and the seasonal component $\mathcal{W}$ may be piecewise harmonic. Let N be the maximum number of iterations, n be the iteration number, and $\mathcal{V}^n$ and $\mathcal{W}^n$ be the trend and seasonal components, respectively, computed at the nth iteration. Let $h \in (0, 1)$ denote the proportion of data points by which two consecutive breakpoints t_i and t_j (including

t_1 and t_S) must be separated. Thus $\lceil Sh \rceil \leq j - i - 1$. Take the length of moving windows to be $\lceil Sh \rceil$, initialize the iteration number $n := 1$, and initialize the seasonal component as $\mathcal{W}^0(T) = (w_1^0, \ldots, w_S^0)$.

Step 1.1: Determine the possibility of breakpoints in trend.
Eliminate the seasonal component from the data

$$u^n = u - \mathcal{W}^{n-1}(T).$$

The ordinary least squares (OLS) estimator for the trend is given as

$$\alpha = (X^t X)^{-1} X^t u^n$$

where X is the Gram matrix for linear regression given by

$$X = \begin{pmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_S \end{pmatrix}.$$

The prediction error (or residual vector or the OLS residual) is defined as

$$E^o = u^n - X\alpha,$$

where the superscript 'o' is used to signify the fact that these residuals are OLS regression based. Consider the process defined by the moving sums (MOSUM) of these OLS residuals

$$Q^o = \left\{ \frac{1}{\sigma\sqrt{\lceil Sh \rceil}} \sum_{i=k-\lceil Sh \rceil+1}^{k} E_i^o \right\}_{k=\lceil Sh \rceil}^{S},$$

where σ is the sample standard deviation of all the OLS residuals.
Compute the OLS-MOSUM test statistic

$$\hat{f}^o = \max_{1 \leq k \leq S-\lceil Sh \rceil+1} |Q_k^o|$$

as the maximum absolute value of this process, then compute the asymptotic critical value of the OLS-MOSUM test using the two-sided boundary-crossing probability

$$p_T = P[f^o > \hat{f}^o],$$

where p_T is read from the Brownian Bridge table.
A p-value less than a user defined parameter $\tau_{\mathcal{V}} \in (0, 1)$ indicates the presence of breakpoints.

REMARK 1: As discussed in ([13]), under the null hypothesis, the OLS-MOSUM process converges in distribution to the increments of a Brownian Bridge process.

Step 1.2: Locate trend breakpoints.

Suppose $p_T \leq \tau_{\mathcal{V}}$. To locate the breakpoints, consider all possible partitions of the domain, compute OLS fits for each partition, and settle with a partition that yields minimum squared error.

Let $X_{[i,j]}$ denote the matrix formed from rows i through j of the matrix X, and $\alpha_{[i,j]}$ denote the least squares coefficients computed using the matrix $X_{[i,j]}$ with time points $t_i, \ldots, t_j$, and data $u^n_{[i,j]} = u^n_{\{i,\ldots,j\}}$. For $i = 1, \ldots, S - \lceil Sh \rceil + 1$, consider each window $[t_i, \ldots, t_{j-1}]$, $i + 2 \leq j \leq S$, and the linear fit in this window. The recursive residual at point t_j is then defined as the weighted prediction error

$$E^r_{ij} = \frac{u^n_{[j,j]} - X_{[j,j]}\alpha_{[i,j-1]}}{\sqrt{1 + X_{[j,j]}\left(X^t_{[i,j-1]}X_{[i,j-1]}\right)^{-1}X^t_{[j,j]}}}.$$

The superscript 'r' is used to signify the fact that the process/statistic is recursive residual based.

Suppose a breakpoint has been found at t_i. Then the cost of placing the next breakpoint at t_k is calculated as the accumulated sum of squared recursive residuals in the interval $[t_i, t_{k-1}]$, i.e.,

$$\rho_{ik} = \sum_{j=i+2}^{k-1} \left(E^r_{ij}\right)^2.$$

All possible positions for the breakpoints can thus be calculated by considering the moving sums of squared recursive residuals, i.e., the process defined by

$$Q^r = \left\{ \left\{ \sum_{j=i+2}^{k} \left(E^r_{ij}\right)^2 \right\}_{k=i+2}^{S} \right\}_{i=1}^{S-\lceil Sh \rceil + 1}.$$

Given the number μ of desired interior breakpoints, let $k_1, \ldots, k_\mu$ be integers such that $k_{i+1} - k_i > \lceil Sh \rceil$, $k_1 > \lceil Sh \rceil + 1$, and $k_\mu < S - \lceil Sh \rceil$. Determine $K = (1, k_1, \ldots, k_\mu, S)$ to minimize the moving sums of squared recursive residuals

$$\sum_{i=3}^{k_1-1} \left(E^r_{1,i}\right)^2 + \sum_{i=k_1+2}^{k_2-1} \left(E^r_{k_1,i}\right)^2 + \sum_{i=k_2+2}^{k_3-1} \left(E^r_{k_2,i}\right)^2 + \cdots + \sum_{i=k_\mu+2}^{S} \left(E^r_{k_\mu,i}\right)^2.$$

Then $(t_{k_1}, \ldots, t_{k_\mu})$ are the interior breakpoints in the trend component.

REMARK 2: The breakpoints $t_1, t_{k_1}, \ldots, t_{k_\mu}, t_S$ are optimal in the sense of the above moving sums of squared recursive residuals criterion.

REMARK 3: If $p_T > \tau_{\mathcal{V}}$, then there are only two breakpoints (t_1 and t_S) and no interior breakpoints. So this step is skipped and there is simply one linear fit over the entire domain $[t_1, t_S]$ (Step 1.3).

Step 1.3: Let $k_0 = 1$, $k_{\mu+1} = S$, and $I_0 = [t_{k_0}, t_{k_1})$, $I_1 = [t_{k_1}, t_{k_2})$, …, $I_\mu = [t_{k_\mu}, t_{k_{\mu+1}}]$. For each interval I_i, determine the linear regression coefficients

$$\gamma^i = \left(X^t_{[k_i,k_{i+1}]} X_{[k_i,k_{i+1}]}\right)^{-1} X_{[k_i,k_{i+1}]} u^n_{[k_i,k_{i+1}]}$$

and construct the (discontinuous) piecewise linear fit

$$\mathcal{V}^n(t) = \sum_{i=0}^{\mu} \Gamma^i(t),$$

where

$$\Gamma^i(t) = \begin{cases} \gamma^i_0 + \gamma^i_1 t, & t \in I_i, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathcal{V}^n(T) = (v^n_1, \ldots, v^n_S)$ be the sequence of values estimated at $t_1, \ldots, t_S$ using this piecewise linear fit.

Step 2.1: Determine the possibility of breakpoints in seasons.
Eliminate the estimated trend component from the observed data

$$\tilde{u}^n = u - \mathcal{V}^n(T).$$

The Gram matrix for the seasonal (harmonic) component is given by

$$Y = \begin{pmatrix} 1 & \sin t_1 & \cos t_1 & \cdots & \sin Kt_1 & \cos Kt_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \sin t_S & \cos t_S & \cdots & \sin Kt_S & \cos Kt_S \end{pmatrix},$$

where K is the degree of the trigonometric polynomial used for regression. The trigonometric regression coefficients for the seasonal component are computed as

$$\beta = (Y^t Y)^{-1} Y^t \tilde{u}^n.$$

The prediction error for this fit is defined as

$$E^o = \tilde{u}^n - Y\beta.$$

The OLS-MOSUM process for these errors is given by

$$Q^o = \left\{ \frac{1}{\sigma\sqrt{\lceil Sh \rceil}} \sum_{i=k-\lceil Sh \rceil+1}^{k} E^o_i \right\}_{k=\lceil Sh \rceil}^{S},$$

and the OLS-MOSUM test statistic is

$$\hat{g}^o = \max_{1 \le j \le S-\lceil Sh \rceil+1} |Q^o_j|.$$

The two-sided boundary-crossing probability

$$p_S = P[g^o > \hat{g}^o]$$

is read from the Brownian Bridge table.
A p-value less than a user defined parameter $\tau_{\mathcal{W}} \in (0,1)$ indicates the presence of seasonal breakpoints.

Step 2.2: Locate seasonal breakpoints.
Suppose $p_S \leq \tau_{\mathcal{W}}$. Using the same notation as for the trend breakpoints,

$$E_{ij}^r = \frac{\tilde{u}_{[j,j]}^n - Y_{[j,j]}\beta_{[i,j-1]}}{\sqrt{1 + Y_{[j,j]}\left(Y_{[i,j-1]}^t Y_{[i,j-1]}\right)^{-1} Y_{[j,j]}^t}}$$

is the recursive residual at time t_j, obtained by trigonometric regression in the time window $[t_i, t_{j-1}]$.
Given the number ν of desired seasonal interior breakpoints and a minimum number of data points separating breakpoints (as for the trend), let l_1, ..., l_ν be integers such that $l_{i+1} - l_i > \lceil Sh \rceil$, $l_1 > \lceil Sh \rceil + 1$, and $l_\nu < S - \lceil Sh \rceil$. Determine $L = (1, l_1, \ldots, l_\nu, S)$ to minimize the moving sums of squared recursive residuals

$$\sum_{i=3}^{l_1-1} \left(E_{1,i}^r\right)^2 + \sum_{i=l_1+2}^{l_2-1} \left(E_{l_1,i}^r\right)^2 + \sum_{i=l_2+2}^{l_3-1} \left(E_{l_2,i}^r\right)^2 + \cdots + \sum_{i=l_\nu+2}^{S} \left(E_{l_\nu,i}^r\right)^2.$$

Then $(t_{l_1}, \ldots, t_{l_\nu})$ are the interior breakpoints in the seasonal component.

REMARK 4: If $p_S > \tau_{\mathcal{W}}$, then there are only two breakpoints (t_1 and t_S) and no interior breakpoints. So this step is skipped and there is simply one trigonometric polynomial fit over the entire domain $[t_1, t_S]$ (Step 2.3).

Step 2.3: Let $l_0 = 1$, $l_{\nu+1} = S$, and $J_0 = [t_{l_0}, t_{l_1})$, $J_1 = [t_{l_1}, t_{l_2})$, ..., $J_\nu = [t_{l_\nu}, t_{l_{\nu+1}}]$. For each interval J_j determine the trigonometric polynomial regression coefficients

$$\delta^j = \left(Y_{[l_j,l_{j+1}]}^t Y_{[l_j,l_{j+1}]}\right)^{-1} Y_{[l_j,l_{j+1}]} \tilde{u}_{[l_j,l_{j+1}]}^n$$

and construct the (discontinuous) piecewise trigonometric polynomial

$$\mathcal{W}^n(t) = \sum_{j=0}^{\nu} \Delta^j(t), \text{ where } \Delta^j(t) = \begin{cases} \delta_1^j + \sum_{k=1}^{K} \delta_{2k}^j \sin kt + \delta_{2k+1}^j \cos kt, & t \in J_j, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathcal{W}^n(T) = (w_1^n, \ldots, w_S^n)$ be the sequence of values estimated at t_1, ..., t_S using this piecewise trigonometric polynomial approximation.

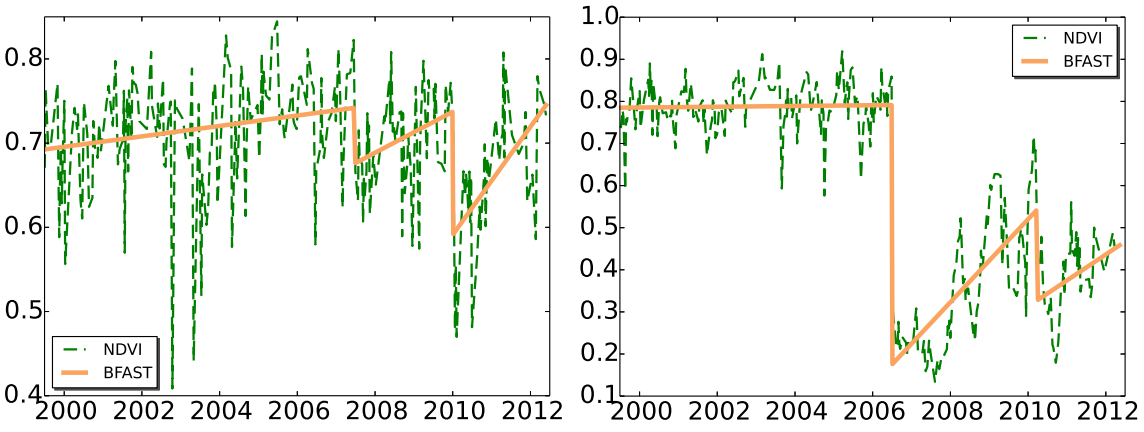
Step 3: Compare the breakpoints between iterations $n-1$ and n.
If the Hamming distance between the two breakpoint vectors $(t_{k_1}, \ldots, t_{k_\mu}, t_{l_1}, \ldots, t_{l_\nu})$ at iterations $n-1$ and n is less than some defined tolerance or the number of iterations

has reached N, then exit. Otherwise, increment the iteration number n and repeat Steps 1.1 to 3.
end
end
end

The complexity of BFAST is $\mathcal{O}(S^3)$, with the calculation of all the recursive residuals being $\mathcal{O}(S^3)$ and dynamic programming to find the optimal breakpoint sequence being $\mathcal{O}(S^2)$. While the experiments so far assure that the method captures the linear trend correctly, its ability to capture phenological (seasonal) changes has not been studied sufficiently yet. Like LandTrendR, BFAST offers a fit to the data as well as a set of breakpoints. The number of breakpoints is fixed, though, and the piecewise linear fit offered is possibly discontinuous (an advantage). The following parameters are used for BFAST in this work: $K = 1$, $\nu = 2$, $h = 0.15$, $\tau_{\mathcal{V}} = \tau_{\mathcal{W}} = 0.05$, $N = 2$.

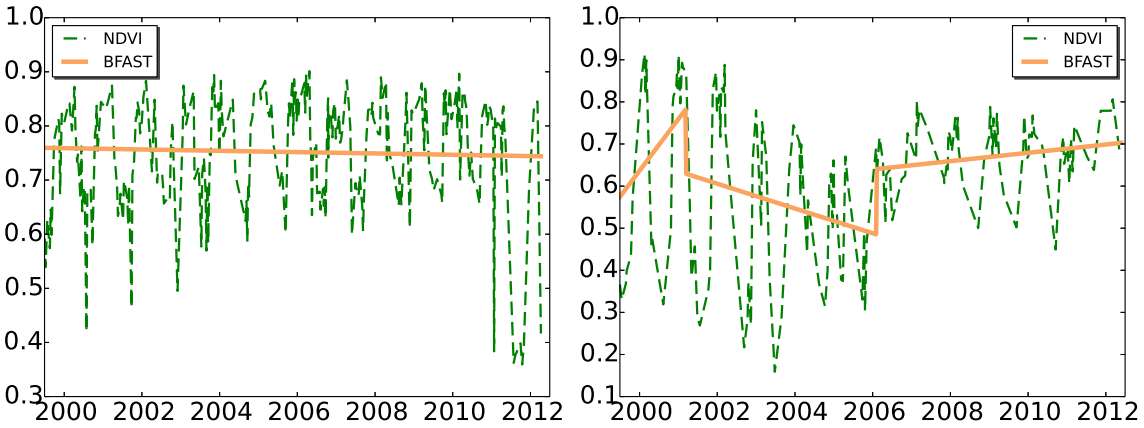
Success. Figure 7 presents two examples of BFAST's success, the first one being a forest pixel and the second one a nonforest pixel. Figure 7(left) has two instances of fire — one in 2006–07, and another in 2009–10, both seen in the NDVI trajectory as well. The occurrence of these events is well sensed by BFAST.

Figure 7(right) depicts the functioning of BFAST on forest as well as nonforest land covers. Specifically, this pixel is a forest initially, a mechanical activity causes a sudden loss in vegetation cover in 2006–07, and there is nonvegetated anthropogenic (NVA) usage following that. Based on the NDVI trajectory, some recovery in leaf cover appears to have taken place post2007. A second mechanical event is known to have happened in 2011–12. BFAST is able to capture both events and also replicates the trends in each of the three segments (separated by these breakpoints) correctly. These results endure after minor changes in the parameters.



Forest → forest, Forest → forest Forest → NVA, NVA → NVA

Figure 7. BFAST success.



Forest → forest, Forest → forest
Harvest in 2002–03, and 2005–06

Forest → NFV
Mechanical in 2003–04

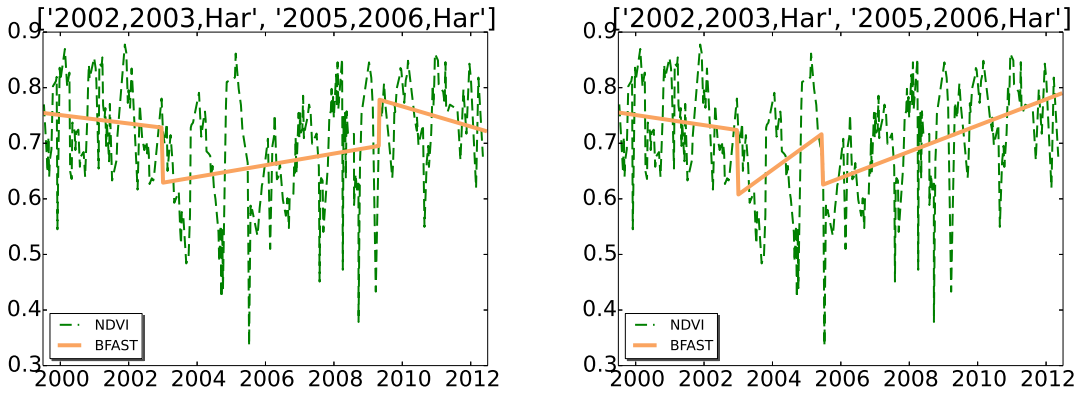
Figure 8. BFAST failure.

Failure. Some examples where BFAST fails are presented here. However, such instance are rather rare. In general, it is difficult to find instances where BFAST fails to capture events and/or trends in forests, its main limitation being the *a priori* chosen number of breakpoints. Specifically, if the number of breaks in the time series is more than the number of breaks specified by the user, only the strongest breaks are captured.

Figure 8(left) corresponds to a forest pixel. There is a harvest in the year 2011–12, reflected in the NDVI trajectory as well. BFAST is not able to capture this change. If two harmonics are used (instead of one) or if the breakpoint spacing proportion h is reduced to $h = 0.10$, this change is captured.

Figure 8(right) represents a pixel that undergoes a mechanical change in 2003–04, going from forest until 2003 (prior to change) to nonforest vegetation after the change. The NDVI trajectory reflects changes in trend and periodicity; two breakpoints are evident — one in 2004 and another in 2006. BFAST places breakpoints in 2001 and 2006, neither agreeing with TimeSync data. However, the 2006 breakpoint does agree with the NDVI trajectory. This outcome did not change with slight variations in parameters.

Sensitivity. BFAST is mostly able to capture the trends well. For the pixels where it fails, the output for some (not all) of them appears to improve by using more harmonics (K, cf. Step 2.1). Figure 9 corresponds to a forest pixel known to have been harvested twice — in 2002 and 2005. When published parameters are used, BFAST detects the first harvest but misses the second one. Using two harmonics, the occurance as well as the timing of both harvests are sensed accurately. Another instance of failure/parameter sensitivity was discussed in Figure 8(left): for this pixel the breakpoint location is too close to the right boundary of the domain. In such situations, the default value of h prevents the (correct) detection of breakpoints.



$K = 1$ $K = 2$

Figure 9. BFAST sensitivity.

Chapter 7.

PROSPECTS FOR A VIABLE POLYALGORITHM

Once the algorithms have been implemented, the next step is to develop a procedure to choose the most appropriate outcome for the input. One possibility is to draw a consensus between algorithm outcomes: if more that half the algorithms being used predict breakpoint sets that are in proximity of each other, one of those algorithms gets chosen as the most appropriate algorithm. To this end, a method to measure the distance between the breakpoint sets is needed. Hausdorff distance from topology is used here.

7.1 Proposed strategy

Consider the set of real numbers R and the usual distance function $\bar{d}(x, y) = |x - y|$ defined on R. Suppose $A = \{a_1, \ldots, a_l\}$ is the set of breakpoints from algorithm $\mathcal{A}$, and $B = \{b_1, \ldots, b_m\}$ is the set of breakpoints from algorithm $\mathcal{B}$. Then A and B are subsets of R. Using the distance function $\bar{d}(x, y)$, define the distance from a point $a \in A$ to the set B as

$$\hat{d}(a, B) = \inf\{\bar{d}(a, b) \mid b \in B\}.$$

Then define the distance from set A to set B as

$$d(A, B) = \sup\{\hat{d}(a, B) \mid a \in A\}.$$

Note that d is not a metric since possibly $d(A, B) \neq d(B, A)$. Hausdorff distance is defined as

$$H(A, B) = \max\{d(A, B), d(B, A)\},$$

and H is a metric.

If every point in A happens to be close to some point B and vice versa, the Hausdorff distance is small. If some point in A is very far from every point in B or vice versa, the Hausdorff distance is large.

Some other facts important in designing the procedure are: (i) EWMACD works based on training data. If the training data is not from a period of stability, EWMACD outcomes cannot be trusted. In general, a prior knowledge of change during the training period is not available. (ii) Since Hausdorff distance is symmetric, H only determines the *pair* of algorithms that are closest to each other; some method to zero in on a single final algorithm is needed. (iii) The case that an algorithm predicts stability (correctly or incorrectly) for a pixel will often arise, so the empty breakpoint set special case must be handled.

In this early work, the following overall procedure for an LULCC polyalgorithm is used:

(1) Run the three algorithms on the input.

(2) For EWMACD, consider only isolated breakpoints. Specifically, during periods of stability, $f_i = f_{i-1}$. For any i, if $f_i \neq f_{i-1}$, check the previous, say 50, time points. If $f_{i-1} = f_j, j = i-50, \ldots, i-2$, then use t_i as a breakpoint. Otherwise, t_i is only a part of an ongoing recovery or loss period, does not mark a new event, and is ignored.

(3) If BFAST predicts a change during the training period of EWMACD, EWMACD results are not used for that pixel (only BFAST and LandTrendR are used).

(4) If $A = \emptyset$ and $B \neq \emptyset$, $d(A, B)$ is undefined and not used. If $A \neq \emptyset$ and $B = \emptyset$, $d(A, B) = \infty$. If $A = B = \emptyset$, $d(A, B) = 0$. The interpretation of this is that the two algorithms agree totally on breakpoints for this pixel.

(5) If $d(A, B) \ll d(B, A)$, by the principle of parsimony, choose $\mathcal{A}$ over $\mathcal{B}$.

(6) If $d(A, B) > d_\tau, \forall A, B$, where d_τ is a user defined threshold, then no consensus has been found between the algorithms, and the pixel is declared to be stable. Due to the limited number of algorithms included at present, and lack of control over the number of breakpoints for BFAST and LandTrendR, $d_\tau = 13$ (the number of years in the data) is used. This is because two algorithms that agree and have found a breakpoint correctly, may have temporally distant false alarms, resulting in either of them not getting selected. As the polyalgorithm evolves, lower values of d_τ can be used.

7.2 Results

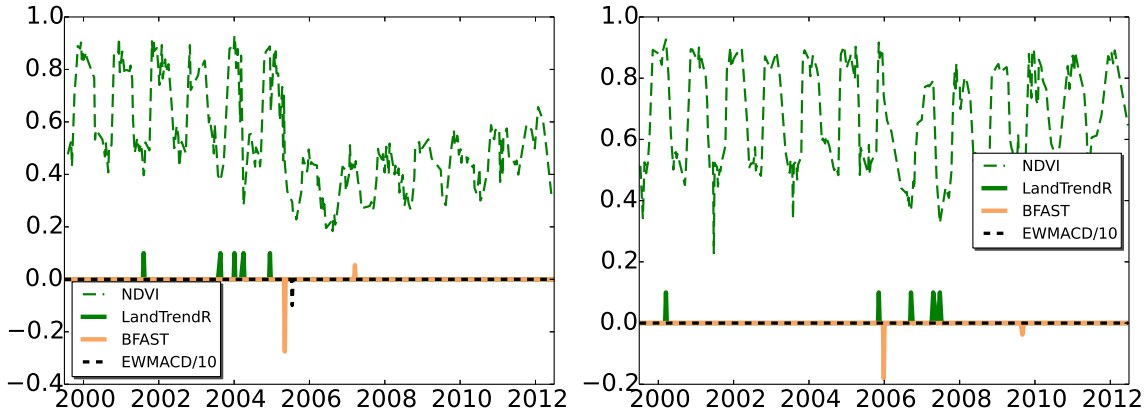
Each component algorithm produces a different kind of outcome (flags for disturbance, continuous fit, discontinuous fit). In the rest of this work, only the breakpoints produced by each algorithm are considered (the focus of this work being correct breakpoint placement); any 'fit' or 'recovery period information' (originally included in the figures in Sections 3, 4, and 5) is not included here. For BFAST, $K = 2$ is used. For the formulae appearing in the

descriptions in the rest of this section, the breakpoint set produced by EWMACD is denoted by E, that by BFAST by B, and that by LandTrendR by L. For BFAST, the height of a jump corresponds to the signed magnitude of the corresponding discontinuity in the porposed model. For EWMACD, the height of the jump corresponds to the signed magnitude of the flag predicted by EWMACD at the corresponding location. For LandTrendR, the model is continuous. So a value of $+1$ is used at its breakpoints for display purposes.

Figure 10 displays instances of polyalgorithm success. The pixel of Figure 10(a) has a stable NDVI trajectory until 2005, suffers a sudden drop in NDVI to almost no vegetation in 2005, and has small but gradually increasing NDVI values thereafter. Based on this trajectory, there should be exactly one breakpoint in 2005. Per TimeSync data, this pixel is a forest that is cleared of vegetation in 2005–06 and is used for anthropogenic purposes after that, agreeing with the TimeSync data and NDVI trajectory. On running the polyalgorithm for this pixel, EWMACD produces one breakpoint, precisely at the location where the drop in NDVI occurs. BFAST produces two breakpoints (as requested), one at the NDVI drop location and another in 2007. LandTrendR produces four breakpoints, the first three being false alarms between 2001 and 2005, and the fourth one being close to the drop location. The breakpoint-sets are: $E = \{2006.03\}$, $B = \{2005.85, 2007.72\}$, $L = \{2002.1, 2004.2, 2004.5, 2004.8, 2005.5\}$. The $d(\cdot, \cdot)$ values are displayed along with the figures. Since $d(E, B) = 0.18$ is the smallest one, EWMACD gets chosen as the final algorithm and the final breakpoint is January 10th, 2006, quite in agreement with both the trajectory as well as TimeSync data. Note that both BFAST and LandTrendR also catch the break in 2005–06.

Figure 10(b) has a stable trajectory until 2006, a sudden drop in NDVI in 2006, with gradual recovery in the subsequent years. There is one breakpoint in the trajectory. Per TimeSync data, this pixel is a forest, with a single harvest in 2007–08. EWMACD assesses this pixel as stable. LandTrendR has a breakpoint in 2000, and multiple breakpoints in 2006 to 2008. BFAST has breakpoints in 2006 and 2010. The breakpoint sets are: $E = \{\}$, $B = \{2006.5, 2010.17\}$, and $L = \{2000.7, 2006.36, 2007.2, 2007.8, 2007.9\}$. The $d(\cdot, \cdot)$ values are displayed along with the figure. BFAST gets selected as the final algorithm, labeling 2006 and 2010 (a false alarm) as breakpoint locations.

Figure 10(c) has a drop in NDVI in 2005, from vegetated to almost zero vegetation. The NDVI recovers in 2005–07. The new vegetation NDVI has a smaller mean. For this



$d(B,E)=1.7,\ d(E,B)=0.18,\ d(B,L)=2.25$

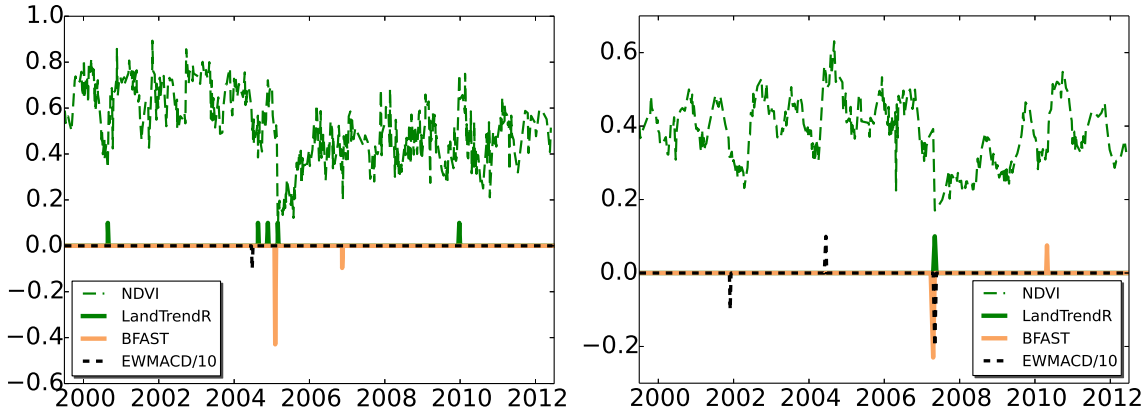
$d(L,B)=3.7,\ d(L,E)=3.9,\ d(E,L)=0.6$

(a) Forest → NVA, Mechanical in 2005.

$d(B,E)=\infty,\ d(E,B)=*,\ d(B,L)=2.2$

$d(L,B)=5.7,\ d(L,E)=\infty,\ d(E,L)=*$

(b) Forest, Harvest in 2007.



$d(B,E)=2.4,\ d(E,B)=0.63,\ d(B,L)=1.7$

$d(L,B)=4.4,\ d(L,E)=5.5,\ d(E,L)=0.17$

(c) NFV → NVN, Flood in 2004.

$d(B,E)=3.02,\ d(E,B)=5.4,\ d(B,L)=2.97$

$d(L,B)=0.04,\ d(L,E)=0.04,\ d(E,L)=5.5$

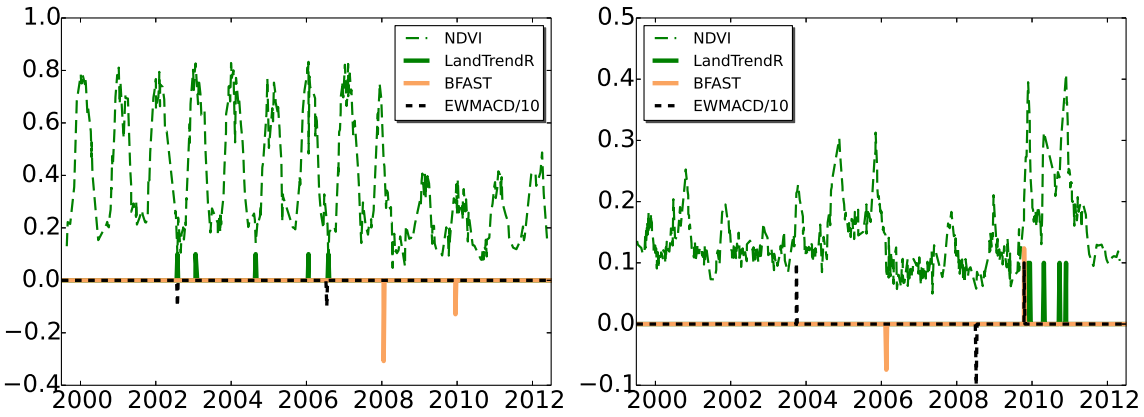
(d) Non-forest vegetated, Fire in 2007.

Figure 10 Polyalgorithm outcomes.

trajectory, two breakpoints are expected: one in 2005, and another in 2007. Per TimeSync data, this pixel is initially covered with nonforest vegetation (NFV), experiences a flood in 2004, and has nonvegetated natural (NVN) land cover thereafter. TimeSync does not note the end of recovery period/beginning of the stable period. Each of the three algorithms detects the flooding event, with their breakpoint placement for the flood being not exactly the same but in close proximity to each other. EWMACD places one breakpoint in 2004, and none in 2006. BFAST produces two breakpoints, the first in 2005, and the second in

2007. BFAST output is thus most in agreement with the NDVI trajectory. LandTrendR places multiple breaks in 2004–05, one each in 2000 and 2010, and none in 2006–07. The polyalgorithm selects EWMACD as the final algorithm. EWMACD's selection can be attributed to (i) fewest number of breakpoints, and (ii) each of its breakpoints being close to some breakpoint of LandTrendR. The outcome of each of the three component algorithms as well as the polyalgorithm agrees with TimeSync data, but BFAST offers the best match to the trajectory. This pixel with flooding does not reflect the general situation with flooding, where all the algorithms perform poorly.

The low NDVI values in Figure 10(d) indicate relatively low vegetation cover. There is one sharp drop in 2007, recovery until 2010, and stability thereafter. TimeSync data classifies this pixel as one covered with nonforest vegetation, with an instance of fire in 2007. The trajectory and TimeSync are in agreement, except that TimeSync does not record the end of a recovery period. EWMACD detects four breakpoints, BFAST two, and LandTrendR two. The three algorithms agree only on 2007 being the breakpoint, which is the desired outcome. The polyalgorithm chooses LandTrendR as the final algorithm.



$d(B,E)$=3.4, $d(E,B)$=5.5, $d(B,L)$=3.4
$d(L,B)$=5.47, $d(L,E)$=1.9, $d(E,L)$=0.07

(a) NFV → NVA, Mechanical in 2008.

$d(B,E)$=2.36, $d(E,B)$=2.39, $d(B,L)$=3.76,
$d(L,B)$=1.115, $d(L,E)$=1.118, $d(E,L)$=6.15

(b) NFV, Fire in 2006.

Figure 11 Current limitations of the polyalgorithm.

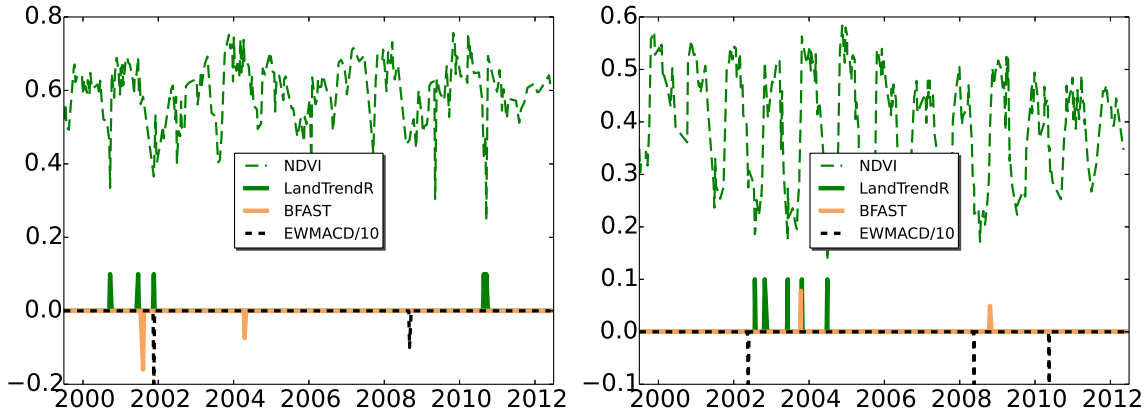
Sometimes, a 'majority' of the component algorithms (two or more out of three here) capture the same set of incorrect breakpoints leading to polyalgorithm failure. Per TimeSync, the pixel of Figure 11(a) is initially covered with nonforest vegetation, cleared in 2008–10, and used for anthropogenic purposes after that. Per the NDVI trajectory, there is a major lasting change in 2008. BFAST detects the occurrence of this change correctly. However, both EWMACD and LandTrendR detect changes in 2002–03 and 2006–07 (in addition to more changes signalled by LandTrendR). Due to this, the EWMACD outcome gets finally chosen by the polyalgorithm. Clearly, the breakpoints based on this final selection (EWMACD) are not correct.

Figure 11(b) shows how an algorithm's localized output can lead to the selection of a suboptimal algorithm. The NDVI trajectory for this pixel has four breakpoints: 2004 (beginning of a recovery period), 2006 (an abrupt drop in NDVI), 2010 (another recovery period), and 2011 (another abrupt drop in NDVI), TimeSync records specify a fire in 2006, and no other event. Ideally, a change detection algorithm would have four breakpoints. EWMACD places breakpoints in 2004, 2008, and 2010. It misses the events in 2006 and 2011, because these events occur amidst unstable periods. In all, EWMACD disagrees with TimeSync information, places breakpoints at two out of four locations expected based on NDVI trajectory. BFAST, with the allowed two breakpoints, places breaks in 2006 and 2010. LandTrendR, on the other hand, with its binary search pattern of stencil choosing, places multiple breakpoints in a very small period 2009–11. Overall, all three algorithms agree on the breakpoint in 2010, but only BFAST catches the fire in 2006–07, only EWMACD catches the beginning a of recovery period in 2004, and only LandTrendR catches the event in 2011. With all breakpoints of LandTrendR accumulated in one region, and one breakpoint (2010) each of EWMACD and BFAST close to all these breakpoints, $d(L, B)$ and $d(L, E)$ are small numbers and the polyalgorithm selects LandTrendR for the final outcome, disagreeing with TimeSync information, and agreeing with the trajectory only on the 2010–12 part. Note that none of the three algorithms captures all four breakpoints suggested by the trajectory (although BFAST got its limit of two correct).

Figure 11(b) shows how an algorithm's localized output can lead to the selection of a suboptimal algorithm. The NDVI trajectory for this pixel has four breakpoints: 2004 (beginning of a recovery period), 2006 (an abrupt drop in NDVI), 2010 (another recovery period), and 2011 (another abrupt drop in NDVI), TimeSync records specify a fire in 2006,

and no other event. Ideally, a change detection algorithm would have four breakpoints. EWMACD places breakpoints in 2004, 2008, and 2010. It misses the events in 2006 and 2011, because these events occur amidst unstable periods. In all, EWMACD disagrees with TimeSync information, places breakpoints at two out of four locations expected based on NDVI trajectory. BFAST, with the allowed two breakpoints, places breaks in 2006 and 2010. LandTrendR, on the other hand, with its binary search pattern of stencil choosing, places multiple breakpoints in a very small period 2009–11. Overall, all three algorithms agree on the breakpoint in 2010, but only BFAST catches the fire in 2006–07, only EWMACD catches the beginning a of recovery period in 2004, and only LandTrendR catches the event in 2011. With all breakpoints of LandTrendR accumulated in one region, and one breakpoint (2010) each of EWMACD and BFAST close to all these breakpoints, $d(L, B)$ and $d(L, E)$ are small numbers and the polyalgorithm selects LandTrendR for the final outcome, disagreeing with TimeSync information, and agreeing with the trajectory only on the 2010–12 part. Note that none of the three algorithms captures all four breakpoints suggested by the trajectory (although BFAST got its limit of two correct).

Finally, instances where the TimeSync data differs from the NDVI trajectory are presented. In Figure 12(a), all three algorithms yield outcomes that disagree with TimeSync information. The NDVI trajectory appears mostly stable except two instances of NDVI drop — in 2002 and 2008. TimeSync does not record any event during the time period under consideration. Per TimeSync, this pixel is being used for nonvegetated anthropogenic purposes (although the NDVI values appear to be high for the pixel to not have any vegetation). Each of the component algorithms predicts breakpoints. Due to BFAST's outcome, EWMACD's selection for the final algorithm is ruled out. (This explains the ∞ values of $d(E, \cdot)$ and $d(\cdot, E)$ displayed with the figure. Note that, this implies that even if EWMACD had correctly captured the stability, it would not get selected as the final algorithm because of BFAST's outcome.) BFAST gets selected as the final algorithm. In Figure 12(right), TimeSync reports mechanical change in 2009–10 but no such change is evident in the NDVI trajectory. On the other hand, the trajectory exhibits three distinct trends: descending NDVI values from 2000 to 2003, a flat trend from 2003 to 2008, and an ascent in NDVI values from 2008–12. Thus, breakpoints in 2003 and 2008 are appropriate. EWMACD and BFAST place breaks 2002-04 and 2008-09. LandTrendR is chosen as the



$d(B,E)=\infty,\ d(E,B)=\infty,\ d(B,L)=2.4$

$d(L,B)=6.4,\ d(L,E)=\infty,\ d(E,L)=\infty$

(a) NVA, stable.

$d(B,E)=1.4,\ d(E,B)=1.6,\ d(B,L)=4.3$

$d(L,B)=1.2,\ d(L,E)=2.1,\ d(E,L)=5.8$

(b) NVA, Mechanical in 2009.

Figure 12 NDVI trajectory vs. TimeSync records.

final algorithm. A third instance of disagreement between TimeSync and NDVI trajectory was discussed in Figure 11(b).

7.3 Discussion

BFAST and LandTrendR have high false positive rates. In particular, for BFAST, the number of breakpoints the algorithm must calculate is fixed (user defined, defaults to two or none in this work), so BFAST is expected to produce one false alarm for each pixel that happens to have only a single event. This leads to an overall high false alarm count for the algorithm. For the current set of parameters LandTrendR is also characterized by several false alarms for any given trajectory. In addition, for short term trajectories, the available TimeSync data contains only major events such as fire, harvest, and the like (and, sometimes, does not agree with the NDVI trajectory (Figure 12)). Each of the three algorithms frequently registers the beginning of any new trend in trajectory as a breakpoint as well. Since 'nonmajor-changes' such as beginning or end of recovery period are not included in TimeSync data, these breakpoints often get (incorrectly) counted towards false alarms. With such limitations, a fair assessment of false alarms is currently not possible.

In case of BFAST, when there exists an ongoing gradual event in the beginning of the trajectory (e.g., a harvest from 2000–02), BFAST places a breakpoint at the end of this change, or, in other words, at the beginning of the next (recovery or stability) trend (so,

in 2002 or 2003 for the harvest that extended from 2000–02). This breakpoint is correct with respect to the trajectory. However, for this example, TimeSync records 2000–02 as the change years. So while checking against TimeSync data, no change is found in the years on record and BFAST gets a false negative. Consequently, BFAST's true positives are slightly undercounted and false negatives are slightly overcounted. The same is true for the polyalgorithm's false negatives/true positives.

Estimates of true positives (TPs, instances where an algorithm correctly identifies a breakpoint) and false negatives (FNs, instances where an algorithms misses a breakpoint) using TimeSync data as reference are presented in Table 2. Four specific types of events are considered: harvest, mechanical, flood, and fire. The total number of pixels that contain these events (not the same as the total number of events) is listed in parentheses beside the event name. Table 3 shows the percentage of algorithm selection in the current polyalgorithm.

Table 2 Breakpoint detection relative to TimeSync data.

		EWMACD	LandTrendR	BFAST	PolyAlg.
Harvest	TP	271	304	368	340
(558 pixels)	FN	340	307	243	275
Mechanical	TP	33	44	49	46
(58 pixels)	FN	39	28	23	26
Flood	TP	8	11	9	8
(19 pixels)	FN	13	10	12	13
Fire	TP	79	72	88	82
(130 pixels)	FN	54	61	45	51

Table 3 Percentage of algorithm selection

	EWMACD	LandTrendR	BFAST	none
Harvest	32.8%	50.5%	14.9%	1.7%
Mechanical	25.8%	25.8%	46.5%	1.7%
Flood	42%	26.3%	31.6%	0%
Fire	27.7%	20.8%	49.2%	2.3%

Based on the numbers in the tables, the polyalgorithm demonstrates improved results compared to either EWMACD or LandTrendR alone. However, BFAST turns out to be the highest performing algorithm, even better than the polyalgorithm itself. The results collectively suggest that, for a significant number of instances, (i) EWMACD and LandTrendR miss 'major' events while these events are correctly identified by BFAST, but (ii) EWMACD and LandTrendR still get chosen as the final algorithm by the polyalgorithm. The selection in these instances could potentially be because of (i) both algorithms finding similar sets of incorrect breakpoints and therefore being in consensus, or (ii) situations similar to Figures 11 and 12(b) — one of the component algorithms yields a clustered set of breakpoints, which also happens to be close to any breakpoint from one of the other component algorithms. Experiments on long term trajectories with the same polyalgorithm strategy have not shown any significant changes in success rates. Including more algorithms will partially alleviate this situation. To robustly combat this situation, the polyalgorithm strategy itself needs to evolve further. Minor variations in polyalgorithm strategy have shown promise but having more algorithms will be beneficial prior to any significant strategy changes.

BFAST vs. polyalgorithm. BFAST is very thorough in its breakpoint search, so its good performance is not surprising. It appears that using BFAST alone may be better than using multiple algorithms and the polyalgorithm. However, BFAST does not scale well ([73], [74]). Furthermore, the user-defined, fixed number of breakpoints ($\mu = 2$ in this study) that BFAST uses is a significant limitation. For the same stack of images (scene), different pixels may have zero, one, two, or more breakpoints. In addition, for longer lengths of time more breakpoints may be desirable. Two breakpoints are barely enough to capture one loss plus the subsequent recovery. Eventually, therefore, other algorithms that allow on-the-fly determination of breakpoints and are also scalable will need to be included as component algorithms. BFAST can instead be used, for example, in subintervals of the dataset to validate breakpoints produced by other component algorithms, or, to check stability in the training period of algorithms dependent on training data. The polyalgorithmic approach appears to be the most promising way of change detection over a variety of land covers, but a viable polyalgorithm remains to be worked out.

Dependence on parameters. The present study is based on a fixed set of parameters (the published parameters). In some (to many) instances, variation of input parameters for any one algorithm could well cause output variability comparable to that between

algorithms. Using larger values of persistance ϖ and L for EWMACD, for instance, make significant differences in the output. Similarly, using $\nu = 0$ for LandTrendR appears to reduce clustering of breakpoints. The current polyalgorithm used the published value $\nu = 0$. The number of breakpoints used for BFAST and LandTrendR, in particular, has significant influence on the polyalgorithm outcomes.

Imbalance in number of breakpoints. Currently, EWMACD has an independent number of breakpoints, BFAST has a user defined, fixed number of breakpoints, while LandTrendR has a user defined upper limit on the number of breakpoints. Different algorithms producing different numbers of breakpoints almost always leads to a significant imbalance in the Hausdorff directional distance based voting. To alleviate this imbalance, one possibility is to run EWMACD, if it produces a minimum number of breakpoints (minimum being decided based on the total number of years in the study), set μ for LandTrendR and $\nu = \mu$ for BFAST equal to it. Preliminary experiments with this approach have shown improved results for BFAST, LandTrendR and the polyalgorithm. However, more algorithms need to be included (at least, to combat the dependence of EWMACD itself on its training period) and outcomes studied further before drawing conclusions.

Chapter 8.

SCALABILITY AND PARALLELIZATION

The size of satellite image datasets to be used for LULCC is prohibitive, currently of the order of terabytes and growing. Specifically, a real Landsat satellite image stack, on an average, consists of approximately 10^7 to 10^8 pixels per image, with 23 to 46 images per year, and time series analysis utilizes at least a few years. Experiments on multicore servers indicate that carrying out meaningful time series analysis on a single interannual, multitemporal stack with existing state of the art codes can take several days. Efforts to implement scalable versions of remote sensing algorithms are, therefore, often made. An HPC platform for time series analysis of satellite images obtained from MODIS was presented in ([5]). In contrast with Landsat images, MODIS images have coarser spatial resolution but much better temporal resolution. Knowing the need for scalable computations in remote sensing, several architectures have also been proposed. These mainly include massive parallel clusters (HIVE), heterogeneous clusters, grid computing, GPUs and the like ([42], [50], [63]). Considerable HPC work on classification (e.g., [62]) has also been carried out. This section describes the implementation, the optimizations, and the parallelization of each of the three algorithms, EWMACD, BFAST, and LandTrendR. The codes are written in two languages: Python and Fortran 2008. Details of the implementations in each language are described separately. Python is an interpreted language, being increasingly used for scientific computing, with avenues to improve its performance for scientific computing being explored. Fortran is a compiled language.

A study area in South Carolina is used for the experiments presented here. Specifically, an image stack taken from Landsat path 18, row 37, with 198 time points, covering the time span January 2009 to December 2014, and one band, viz., the NDVI, is used. Figure 13 shows the satellite images taken on dates January 3rd, 2009, and February 16th 2014, corresponding to the beginning and the final year of the image stack (henceforth referred to as SC1837). Each image in this stack has $7411 \times 8801 = 65224211 > 10^7$ pixels. Thus, the image stack dimensions are $R = 7411$, $C = 8801$, $S = 198$, $B = 1$. All the experiments

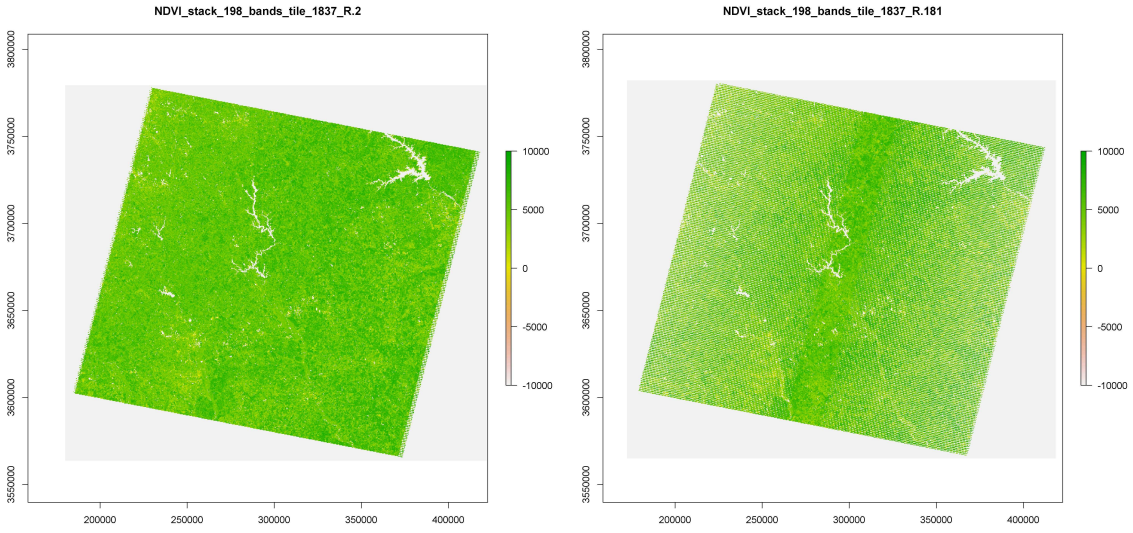


Figure 13. Beginning (left) and end (right) of image stack for NDVI values for which TCC data exists (2009 –2014). The x- and y-axes represent relative pixel coordinates of the extracted image.

presented in this article were carried out on Navajo, a machine hosted by the Computer Science department at Virginia Tech.

The sequential baseline implementation of BFAST in Fortran (faster than that in Python) takes over 7 hours to analyze a $1000 \times 1000 = 10^6$ pixel subset of the full scene. For EWMACD, its sequential baseline implementation in Python takes about an hour for the 10^6 pixels subset while Fortran sequential baseline implementation takes 2 minutes, 18 seconds for this subset. For EWMACD, thus, Fortran is approximately 26 times faster than Python on Navajo. Fortran EWMACD code takes 1 hour and 24 minutes to process the full image stack (full scene, 198 time points). Sequential baseline implementation of LandTrendR in Fortran takes about 6.65 hours to complete. Given the infeasibility of the serial baseline implementation of BFAST for a full scene, the better than BFAST but still insufficient performance of EWMACD and LandTrendR and the knowledge that (i) the image stack discussed in this work consists of only 198 time points (2009–2014) while there are currently 900 time points (1984–2014) actually available, (ii) the run times discussed here are for a single path/row only while there are more than 450 path/rows in the US alone, and other similar facts, scaling the codes is imperative for any meaningful analysis. Herewith, are

presented (i) the optimizations used to address the computational hotspots and hardware bottlenecks, and (ii) the parallel implementations for both Python and Fortran.

All three algorithms involve a fair number of arrays, Therefore, attention is given to the creation and use of array variables. In addition, a major concern in designing parallel codes for Landsat imagery is load balancing across pixels. Specifically, since the time series processing for any given pixel is independent of that for any other pixel, the algorithms are apparently embarrassingly parallel with respect to pixels. Landsat images, however, suffer from missing observations (due to factors beyond human control), thereby resulting in 'invalid' pixels (a pixel is declared invalid if there are fewer than $2K+1$ observations in the entire time span, where K signifies the number of harmonics being used for the EWMACD). These invalid pixels are nonuniformly distributed across the data, which induces very high work load imbalance across the pixels.

8.1. Python

Arrays in Python are handled using NumPy, high level inbuilt support for working with multidimensional arrays. NumPy automatically vectorizes array operations making them very efficient. Together with SciPy, the fundamental scientific computing library of Python, Numpy supports most linear algebra operations, a feature of interest with regard to the algorithms considered here. In addition, Python also supports creation of anonymous functions using a keyword 'lambda'. This allows direct array to array transformations, preserving the efficiency of vector instructions. For example, $sin(jt_i), i = 0, \ldots, M$ may be carried out as $sin(map(lambda\ x : x * j, t[0 : M]))$. For the algorithms considered here, not all array operations are directly portable to Numpy; lambda expressions facilitate the porting in these instances. Overall, significant gain in performance over the baseline implementation is noticed on replacing Python primitives with Numpy primitives. For EWMACD, approximately 25% of the total time originally went in determining the validity of a pixel, and 20% in least squares fitting, 20% in calculation of residuals. On replacing the Python primitives with NumPy primitives, these numbers reduce to 0.7%, 16.5%, and 15.4%, respectively. Other procedures such as computing the control charts, flagging history gain prominence as computational hot spots, each one taking 16% of the total time. Unlike least squares fitting and residual calculations, this latter set of procedures is not expressed as concise mathematical operations, NumPy cannot be utilized for their optimization.

Python sequential codes are (weakly) parallelized using the multiprocessing module available in Python. Multiprocessing creates subprocesses with disjoint virtual memory space. The image stack is chunked pixelwise and these chunks are distributed to the processes. Within a subprocess, that chunk is processed sequentially. The full scene is processed using 64 subprocesses in approximately 64 minutes. For this full scene run, a load imbalance is also noticed across subprocesses, because the chunking of pixels utilized here does not ensure load balancing. An improved way to distribute chunks would be to actually partition the image into a larger number of chunks (for example, 3000 chunks) and then distribute these chunks across the subprocesses nonuniformly. However, this is currently not implemented. Further, Python is an interpreted language with a global interpreter lock (GIL) (https://wiki.python.org/moin/GlobalInterpreterLock). To bypass GIL enforced serialization and run multiple threads in parallel without compromising thread safety requires nontrivial programming time and is still risky. With the current implementation and the run times recorded here, processing the full scene using Python EWMACD code with a single process will take more than a day and is, therefore, not attempted.

8.2. Fortran

Fortran has intrinsic support for high dimensional arrays. The algorithms implemented here necessitate the use of several intermediate (work) arrays. Since allocation/deallocation of arrays is expensive, global arrays are used for all work arrays. Fortran vector instructions are utilized wherever possible. LAPACK and BLAS libraries are used for all linear algebra operations.

Fortran sequential codes are parallelized using OpenMP. In contrast with Python's multiprocessing, OpenMP is a shared memory paradigm. The input and output arrays are the only large arrays; these are declared as shared variables. The remaining significant hardware bottleneck is the load imbalance across pixels. Attempting to weed out invalid pixels in a preprocessing step and execute the PARALLEL DO loop for only valid pixels leads to CPU underutilization (from 99.9% to 70–80%), simultaneously increasing the OpenMP time. This presumably is due to memory contention: the memory access pattern for the latter approach is such that multiple threads try to access the same memory bank(s). Furthermore, even amongst the valid pixels, the total number of observations available for

one pixel can be much less than the number of observations available for some other pixel. So, even with this preprocessing approach the work load balance is not guaranteed.

Finally, allocating/deallocating arrays within each thread is inefficient. Allocatable global arrays in modules can be used by threads via THREADPRIVATE, but this data copy mechanism does not work with dynamic loop scheduling, which is desirable because of the large variance in pixel analysis times (including missing data for a pixel). The best alternative is using Fortran automatic arrays with OpenMP PRIVATE.

The computational hot spots for Fortran codes are identified. For EWMACD, more than 50% of the time is spent in least squares fitting, specifically in DGELS (LAPACK) calls. LAPACK is already optimized for the hardware. 22% of the total time is in the calculation of residuals. This subroutine has two DO loops with dependencies and cannot be vectorized. For BFAST, approximately 97% of the OpenMP time is spent in computing the recursive residuals. Essentially, linear and harmonic least squares fits are done in every permissible interval, and the least squares fitting is already optimized. For LandTrendR, approximately 29% of the time is spent in despike step. Approximately 7% time is spent in linear regression.

In summary, after considering and testing several alternatives, the best approach found was to (1) perform the raw binary stream input data order (r, c, s, b) conversion to (s, c, r, b) order in parallel; (2) cull invalid (including missing) pixels inside the subroutines EWMACD and BFAST, which are called from within a PARALLEL DO; (3) convert the nested DO loop `DO r=1,R ; DO c=1,C` into a single PARALLEL DO loop `DO k=1,R*C,A` ; (4) use OpenMP `SCHEDULE(DYNAMIC,1)`; (5) process a chunk of A pixels indexed by k on each call to EWMACD and BFAST; (6) use automatic rather than allocatable arrays for all small work arrays in the subroutines, and allocate/deallocate just one large work array in both EWMACD and BFAST; (7) perform all I/O outside parallel OpenMP constructs to reduce memory and disk contention. Note that manually collapsing the nested loops and chunking within the pixel processing subroutine is more efficient than collapsing and chunking at the OpenMP directive level, since the latter would call the subroutines, which allocate and

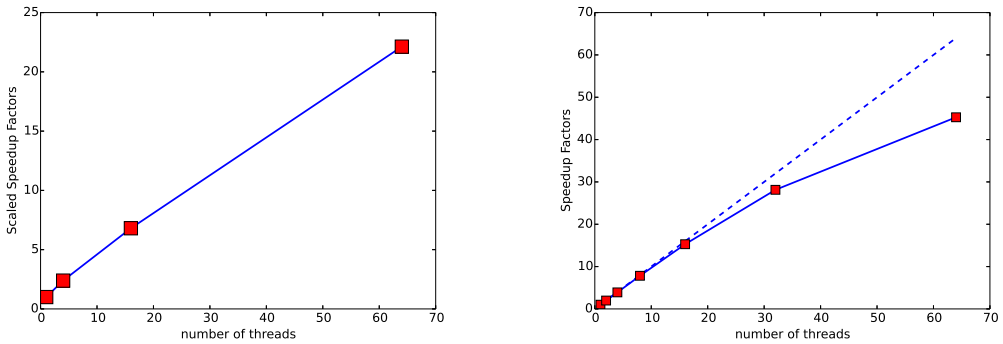


Figure 14. Scalability of Fortran codes: BFAST scaled speedup for a base size $B = 50 \times 50$ (left); speedup of EWMACD code (right). Both the graphs are for parallel sections only.

deallocate numerous work arrays, for each pixel index. The difficulty of load balancing "embarrassingly parallel" applications is analyzed theoretically in ([2]).

Figure 14(left) shows the scaled speedup for BFAST, i.e., increasing both the problem size and the number of cores. The isoefficiency (constant efficiency as both the problem size and number of cores are scaled up) decreases significantly, indicating some combination of poor load balancing (the pixel chunk size $A = 100$), main memory contention, and increasing thread management overhead, as yet unresolved.

Parallel results for the full scene, processed with EWMACD, are shown in Figure 14(right). Using 64 cores, the full scene is processed in 1.86 minutes, with a speedup of roughly 46. The input binary image is 25GB. For this image, the code needs 74GB of memory. When a single thread (core) is used, the cache miss rate is 0.566% and 1.03 instructions per cycle are executed. For 64 cores, the cache miss rate is 0.721% and 0.52 instructions per cycle are executed. On 64 cores, the FLOPS performance of the EWMACD code is 48.54 GFLOPs. The peak theoretical performance for this machine is 358.4 GFLOPs, yielding performance to peak ratio of $48.54/358.4 = 13.54\%$.

For LandTrendR, on using the current Fortran implementation and 64 cores, the full scene is processed in 11.38 minutes, thus yielding a speedup of 35. It is possible to further improve the implementation, an aspect left to future work. The code takes approximately 74GB memory, similar to EWMACD.

For all the codes, the input (as well as output) image stacks are in binary file format. The logical mathematical description of an image stack uses the index order (r, c, s, b), where

r, c, s, and b denote the row, column, time point, and spectral band, respectively. Python accesses arrays in row order. However, chunking of pixels for multiprocessing requires all the information for a given pixel to be contiguous. Without this, the data cannot be correctly chunked. Fortran accesses array elements in column order. To ensure good memory locality and due to the hardware effects of cache misses and paging, the image stack in Fortran implementation must be stored and processed in the index order (s, c, r, b). Failure to use this latter index order can result in a cache miss rate as high as 28%. So for both Python and Fortran, the input image is stored in time sequential format.

In python codes, Numpy is used for loading, reading and writing binary files. In Fortran codes, Fortran I/O with streaming access is utilized to read and write these files. Kernprof was used for profiling Python codes, valgrind for Fortran. Perf was used to determine cache misses. Visualizations are done in Python. The results presented in this work were obtained on a single machine: 64 core AMD Opteron 6276, 1.4GHz CPU, 2MB cache per core, 265 GB main memory, CentOS. Python 2.7 and gfortran compiler version 4.8 were used for respective languages.

Python vs. Fortran vs. R performance. On Navajo, for a 10^6 pixels scene with 198 time points, EWMACD Python code takes 68 minutes to complete serially and 1.38 minutes using 64 processes, exhibiting a speedup of approximately 49; EWMACD Fortran code takes 138.43 seconds serially and 2.96 seconds using 64 threads, a speedup of about 46. The speedups are similar but the actual execution times are vastly different for the two languages: Fortran codes are 26 times faster than their Python counterparts. Further, in going from 32 to 64 processes, the performance gain for Python EWMACD is only 23% while Fortran EWMACD gains 40% (ideal would be 50%). This indicates weaker cache utilization by Python. An overall 7% cache miss rate is observed for Python codes after all optimizations; for Fortran it is 0.72%. For the 64Mpixels image (larger full scene), Fortran parallel code is 35 times faster than Python parallel code, possibly due to lack of load balancing in Python. Finally, processing this 10^6 pixel scene using EWMACD standard codes written in R, another language popularly used in remote sensing community, takes about 140 minutes. Serial Fortran codes for EWMACD have thus yielded a speedup of more than 60 over standard R codes. Since Fortran code for EWMACD itself exhibits a speedup of about 46 (on 64 cores), parallel Fortran codes for EWMACD have a speedup of 2760 over R (which does not directly support parallel computing).

8.3. Summary

Python is an excellent language for developing code prototypes. Algorithms are implemented, analyzed, and tested in Python using minimal programmer time. Good scalability is achieved for NumPy-friendly expressions. Expressions that do not conform to NumPy constructs are slower and limit the scalability of the code. Experiments here indicate that Python's cache utilization must be improved, especially with a large number of cores, for Python to offer performance comparable with Fortran. This work makes a quantitative assessment (development time, performance, scalability) of Python codes vs. Fortran codes with respect to LULCC. Given the impending computation demands with these datasets, Fortran is indispensable. Development of Fortran codes takes longer but the codes are orders of magnitude faster and scale better. Strategically, therefore, code prototypes are developed in Python and then utilized to develop Fortran codes. For the polyalgorithm to be scalable, BFAST's implementation for a single pixel needs to be aggressively explored, lest BFAST be used only on a need basis (current run time is > 30 hours for a full scene with 198 time points).

Chapter 9.

CONCLUSION AND FUTURE WORK

At present, the polyalgorithm offers some improvement over EWMACD or LandTrendR alone, but not over BFAST. Since BFAST is the most rigourous/exhaustive component algorithm, and the polyalgorithm is not as good as BFAST yet, there is surely room for improvement of the polyalgorithm. The experiments carried out in this work are just the beginning of further work towards a viable polyalgorithm. Several directions of research/development must be pursued in this regard.

First, the number of algorithms included here is not sufficient for a robust consensus vote. Two out of three algorithms often miss trends and/or important changes of interest to users. With only three algorithms in the framework, the polyalgorithm also misses these changes/events. At least two more algorithms must be included for robust voting. Algorithms based on splines (e.g., [29]) and those utilizing spatial information have potential for inclusion.

Second, the current algorithm selection is not exhaustive in data coverage. For example, for trajectories where there is no significant change in trend and the training period of EWMACD is also unstable, only LandTrendR is applicable. An algorithm suitable for such trajectories is needed.

Third, in addition to determining the 'best' algorithm for a given pixel, it may also be possible to incorporate parameter tuning/adaptation in the polyalgorithm. For example, for a random sample of pixels with known validated changes, the polyalgorithm may run each algorithm for a range of parameters rather than for the fixed set of (published) parameters. The polyalgorithm will then also suggest good parameter choices for each algorithm on the full data set. Adding such intelligence to any of the algorithms, and to the polyalgorithm, is a significant research challenge.

Fourth, resource allocation is a concern. Parallelism is presently implemented only across pixels (first level) using OpenMP, i.e., multiple cores of a node take different pixels and process them simultaneously. For a fixed pixel, the component algorithms are currently

executed serially. With this arrangement, load balancing within individual algorithms is already an issue. If a national level image analysis has to be carried out or if parameter tuning has to be included in the polyalgorithm, parallelism across algorithms (second level) becomes essential. A third layer of parallelism, namely, parallelism within algorithms, is also desirable. Load balancing between these three (coarse to fine grained) levels of parallelism in the polyalgorithm will be challenging.

Finally, the directional Hausdorff distance $d(\cdot, \cdot)$ alone is not sufficient to determine the polyalgorithm outcome. The instances of failure of the current basic strategy are primarily due to insufficient number of component algorithms and, equally important, due to to multiple algorithms producing proximal but incorrect results. To fully circumvent the latter situation, the polyalgorithm strategy needs to evolve futher. Possibilities include: (i) including parameter variation (ii) breakpoint acceptance test (for example, residuals for a piecewise linear fit based on proposed breakpoints). (iii) tests for goodness of fit and correlation coefficient, (iv) preemptively checking whether a component algorithm is even suitable for a given time series (e.g., if the training period of EWMACD is not stable, then EWMACD is simply not used for that pixel). Including more algorithms will allow designing more sophisticated strategies.

Given the inconsistent trend predictions between the different algorithms, the sometimes erratic behavior of a given algorithm on a given image stack, the sensitivity to parameters for some algorithms, and the prohibitive execution times for serial codes, there is clearly a need for a parallel polyalgorithm (an intelligent, adpative union of multiple algorithms). This and earlier work ([73]), assessing the scalability and memory footprint, the parameter sensitivity, and the range of applicability of individual algorithms are but first steps toward such a parallel polyalgorithm for hyperspectral Landsat image stacks. In summary, the contributions of this work are (1) a systematic investigation of the issues involved in creating a polyalgorithm; (2) precise mathematical descriptions of BFAST, EWMACD, and LandTrendR; (3) efficient, portable, parallel Python 2.7 and Fortran 2008 code for all algorithms; (4) directional Hausdorff distance for comparing changes; (5) a unified I/O framework for running and comparing LULCC algorithms.

REFERENCES

[1] Agrawal, R., Faloutsos, C., and Swami, A., 1993, "Efficient similarity search in sequence databases", in *Lomet D.B. (eds) Foundations of Data Organization and Algorithms*, 69–84.

[2] Ahn, T.-H., Sandu, A., Watson, L.T., Shaffer, C.A., Cao, Y., and Baumann, W.T., 2015, "A framework to analyze the performance of load balancing schemes for ensembles of stochastic simulations", *International Journal Parallel Programming*, 43(4), 597–630.

[3] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D., 1992, *LAPACK Users' Guide*, SIAM.

[4] Banner, A., and Lynham, T., 1981, "Multitemporal analysis of Landsat data for forest cutover mapping — a trial of two procedures", in *Proceedings of the 7th Canadian Symposium on Remote Sensing, Winnipeg, Canada*, 233–240.

[5] Bergh, F.V.D., Wessels, K.J., Miteff, S., Zyl, T.L.V., Gazendam, A.D., and Bachoo, A.K., 2012, "HiTempo: a platform for time-series analysis of remote-sensing satellite data in a high-performance computing environment", *International Journal of Remote Sensing*, 33(15), 4720–4740.

[6] Brooks, E.B., Thomas, V.A., Wynne, R.H., and Coulston, J. W., 2012, "Fitting the multitemporal curve: a Fourier series approach to the missing data problem in remote sensing analysis", *IEEE Transactions on Geosciences and Remote Sensing*, 59, 3340–3353.

[7] Brooks, E.B., Wynne, R.H., Thomas, V.A., Blinn, C.E., and Coulston, J.W., 2014, "On-the-fly massively multitemporal change detection using statistical quality control charts and Landsat data", *IEEE Transactions on Geoscience and Remote Sensing*, 52(6), 3316–3332.

[8] Brooks, E.B., Zhiqiang, Y., Thomas, V.A., and Wynne, R.H., 2017, "Edyn: Dynamic Signaling of Changes to Forests Using Exponentially Weighted Moving Average Charts", *Forests*, 8(304), 18.

[9] Box, G.E.P., and Jenkins, G.M., 1970, *Time Series Analysis: Forecasting and Control*, San Francisco: Holden Day. (Revised edition published 1976).

[10] Cai, S., and Desheng, L., 2015, "Detecting Change Dates from Dense Satellite Time Series Using a Sub-Annual Change Detection Algorithm", *Remote Sensing*, 7, 8705–8727.

[11] Campbell, J.B., and Wynne, R.H., 2011, *Introduction to Remote Sensing, Fifth Edition*, Guilford Publications.

[12] Chan, K., and Fu, W., 1999, "Efficient time series matching by wavelets", in *Proceedings of the 15th IEEE International Conference on Data Engineering (ICDE)*, 8 pages.

[13] Chu, C-S.J., Hornik, K., and Kuan, C.-M., 1995, "Mosum tests for parameter constancy", *Biometrika*, 82, 603–617.

[14] Cohen, W.B., and Fiorella, M., 1998, *Remote Sensing Change Detection: Environmental Monitoring Applications and Methods, edited by C. D. Elvidge and R. S. Lunetta*, Ann Arbor Press, 89–102.

[15] Cohen, W.B., Fiorella, M., Gray, J., Helmer, E., and Anderson, K., 1998, "An efficient and accurate method for mapping forest clear cuts in the Pacific Northwest using Landsat imagery", *Photogrammetric Engineering and Remote Sensing*, 64, 293–300.

[16] Cohen, W.B., Healey, S.P., Zhiqiang, Y., Stehman, S.V., Brewer, C.K., Brooks, E.B., Gorelick, N., Huang, C., Hughes, M.J., Kennedy, R.E., Loveland, T.R., Moisen, G.G., Schroeder, T.A., Vogelmann, J.E., Woodcock, C.E., Yang, L., and Zhu, Z., 2017, "How Similar Are Forest Disturbance Maps Derived from Different Landsat Time Series Algorithms", *Forests*, 8, 98.

[17] Cohen, W.B., Yang, Z., and Kennedy, R., 2010, "Detecting trends in forest disturbance and recovery using yearly Landsat time series: 2. TimeSync-Tools for calibration and validation", *Remote Sensing of Environment*, 114(12), 2911–2924.

[18] Coppin, P.R., and Bauer, M.E., 1994, "Processing of multitemporal Landsat TM imagery to optimise extraction of forest cover change features", in *IEEE Transactions on Geoscience and Remote Sensing*, 32, 918–927.

[19] Coppin, P., Jonckheere, I., Nackaerts, K., Muys, B., and Lambin, E., 2004, "Digital change detection methods in ecosystems monitoring: A review", *International Journal of Remote Sensing*, 25(5), 1565–1596.

[20] Dietterich, T.G., Kittler, J., and Roli, F., 2001, *Ensemble methods in machine learning, Multiple Classifier Systems*, LNCS Vol. 1857, Springer, 1–15.

[21] Douglas, D.H., and Peucker, T.K., 1973, "Algorithms for the reduction of the number of points required to represent a digitized line or its carricature", *Canadian Cartographer*, 10(2), 112–122.

[22] Duda, R.O., and Hart, P.E., 1973, *Pattern Classification and Scene Analysis*, Wiley, New York.

[23] Fall, S., Niyogi, D., Gluhovsky, A., Pielke, R.A. Sr., Kalnay, E., and Rochon, G., 2010, "Impacts of land use and land cover on temperature trends over the continental United States: assessment using the North American Regional Reanalysis", *International Journal of Climatology*, 30(13), 1980–1993.

[24] Fung, T., and LeDrew, E., 1987, "Application of principal components analysis to change detection", *Photogrammetric Engineering and Remote Sensing*, 53, 1649–1658.

[25] Fung, T., 1990, "An assessment of TM imagery for land cover change detection", *IEEE Transactions on Geoscience and Remote Sensing*, 28, 681–684.

[26] Goldstein, T., and Osher, S., 2009, "The split Bregman method for L1-regularized problems", *SIAM Journal on Imaging Sciences*, 2(2), 323–343.

[27] Gomeni, R., and Gomeni, C., 1979, "AUTOMOD: A polyalgorithm for an integrated analysis of linear pharmacokinetic models", *Computers in Biology and Medicine*, 9(1), 39–48.

[28] Goodwin, N.R., Coops, N.C., Wulder, M.A., Gillanders, S., Schroeder, T.A., and Nelson, T., 2008, "Estimation of insect infestation dynamics using a temporal sequence of Landsat data", *Remote Sensing of Environment*, 112, 3680–3689.

[29] Gretchen, G.M., Meyer, M.C., Schroeder, T.A., Liao, X., Schleeweis, K.G., Freeman, E.A., and Toney, C., 2016, "Shape selection in Landsat time series: a tool for monitoring forest dynamics", *Global CHange Biology*, 22(10), 3518–3528.

[30] Häfner, H., Schönauer, W., and Weiss, R., 1999, "The program package LINSOL — basic concepts and realization", *Applied Numerical Mathematics*, 30(2–3), 213–224.

[31] Hame, T.H., 1986, "Satellite image aided change detection", *In Remote sensing-aided forest inventory, Research Notes No. 19, Department of Forest Mensuration and Management, University of Helsinki, Helsinki, Finland*, 47–60.

[32] Hansen, M.C., Potapov, P.V., Moore, R., Hancher, M., Turubanova, S.A., Tyukavina, A., Thau, D., Stehman, S.V., Goetz, S.J., Loveland, T.R., Kommareddy, A., Egorov, A., Chini, L., Justice, C.O., and Townshend, J.R.G., 2013, "High-resolution global maps of 21st-century forest cover change", *Science*, 342, 850–853.

[33] Healey, S.P., Cohen, W.B., Yang, Z., Brewer, C.K., Brooks, E.B., Gorelick, N., Hernandez, A.J., Huang, C., Hughes, M.J., Kennedy, R.E., Loveland, T.R., Moisen, G.G., Schroeder, T.A., Stehman, S.V., Vogelmann, J.E., Woodcock, C.E., Yang, L., and Zhu, Z., 2017, "Mapping forest change using stacked generalization: An ensemble approach", *Remote Sensing of Environment*, http://dx.doi.org/10.1016/j.rse.2017.09.029.

[34] Huang, C., Goward, S.N., Masek, J.G., Thomas, N., Zhu, Z., and Vogelmann, J.E., 2010, "An automated approach for reconstructing recent forest disturbance history using dense Landsat time series stacks", *Remote Sensing of Environment*, 114(1), 183–198.

[35] Hughes, M.J., 2014, *New Remote Sensing Methods for Detecting and Quantifying Forest Disturbance and Regeneration in the Eastern United States*, University of Tennessee - Knoxville.

[36] Hunter, J., and McIntosh, N., 1999, *Artificial Intelligence in Medicine*, Springer.

[37] Jensen, J.R., 1983, "Urban change detection mapping using Landsat digital data", *The American Cartographer*, 81, 127–147.

[38] Jin, S., Yang, L., Danielson, P., Homer, C., Fry, J., and Xian, G., 2013, "A comprehensive change detection method for updating the National Land Cover Database to circa 2011", *Remote Sensing of Environment*, 132, 159–175.

[39] de Jong, R., Verbesselt, J., Schaepman, M.E., and de Bruin, S., 2012, "Trend changes in global greening and browning: Contribution of short-term trends to longer-term change", *Global Change Biology*, 18, 642–655.

[40] de Jong, R., Verbesselt, J., Zeileis, A., and Schaepman, M., 2013, "Shifts in global vegetation activity trends", *Remote Sensing*, 5(3), 1117–1133.

[41] Joyce, A.T., and Burns, G.S., 1981, "Evaluation of land cover change detection techniques using Landsat MSS data", in *Proc. of the 7th PECORA Symposium, Sioux Falls, SD, USA (Bethesda, MD: ASPRS)*, 252–260.

[42] Kalluri, S.N.V., JaJa, J., Bader, D.A., Zhang, Z., Townshend, J.R.G., and Fallah-Adl, H., 2000, "High performance computing algorithms for land cover dynamics using remote sensing data", *International Journal of Remote Sensing*, 21(6 & 7), 1513–1536.

[43] Kennedy, R.E., Cohen, W.B., and Schroeder, T.A., 2007, "Trajectory-based change detection for automated characterization of forest disturbance dynamics", *Remote Sensing of Environment*, 110, 370–386.

[44] Kennedy, R.E., Yang, Z., and Cohen, W.B., 2010, "Detecting trends in forest disturbance and recovery using yearly Landsat time series: 1. LandTrendr – Temporal segmentation algorithms", *Remote Sensing of Environment*, 114, 2897–2910.

[45] Keogh, E., Chu, S., Hart, D., and Pazzani, M., 2001, "An online algorithm for segmenting time series", in *Proc. of IEEE International Conference Data Mining*, 289–296.

[46] Kittler, J., Hatef, M., Duin, R.P.W., Matas, J., 1998, "On combining classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3), 226–239.

[47] Koski, A., Juhola, M., and Meriste, M., 1995, "Syntactic recognition of ECG signals by attributed finite automata", *Pattern Recognition*, 28(12), 1927–1940.

[48] Kriegler, F.J., Malila, W.A., Nalepka, R.F., and Richardson, W., 1969, "Preprocessing transformations and their effects on multispectral recognition", in *Proc. of the Sixth International Symposium on Remote Sensing of Environment*, 97–131.

[49] Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D., and Allen, J., 2000, "Mining of Concurent Text and Time Series", in *Proc. of the 6th International Conference on Knowledge Discovery and Data Mining*, 37–44.

[50] Lay, David C., *Linear Algebra and its Applications*, Addison Wesley.

[51] Lee, C.A., Gasster, S.D., Plaza, A.J., Chang, C.-T., and Huang, B., 2011, "Recent developments in high performance computing for remote sensing: a review", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3), 508–527.

[52] Li, C., Yu, P., and Castelli, V., 1998, "MALM: A framework for mining sequence database at multiple abstraction levels", in *Proc. of the 9th INternational Conference on Information and Knowledge Management*, 267–272.

[53] Li, J., 1996, *A Polyalgorithm for Parallel Dense Matrix Multiplication on two-dimensional process grid topologies*, Thesis: Missisippi State University.

[54] LLNL, *OpenMP*, https://computing.llnl.gov/tutorials/openMP/#THREADPRIVATE.

[55] Lu, D., Mausel, P., Brondizio, E., and Moran, E., 2004, "Change detection techniques", *International Journal of Remote Sensing*, 25(37), 2365–2401.

[56] Lunetta, R.S., Knight, J.F., Ediriwickrema, J., Lyon, J.G., and Worthy, L.D., 2006, "Land-cover change detection using multi-temporal MODIS NDVI data", *Remote Sensing of Environment*, 105(2), 142–154.

[57] Mougel, P.N., and Folcher, N.S., 2012, "A data mining approach to discover collections of homogeneous regions in satellite image time series", in *Geoscience and Remote Sensing Symposium (IGARSS)*, 4360–4363.

[58] Myeneni, B., R., Keeling, C.D., Tucker, C.J., Asrar, G., and Nemani, R.R., 1997, "Increased plant growth in the northern high latitudes from 1981 to 1991", *Nature*, 386, 698–702.

[59] Neilsen, A., Conradsen, K., and Simpson, J., 1998, "Multivariate alteration detection (MAD) and MAF post processing in multi-spectral bi-temporal image data: new approaches to change detection studies", *Remote Sensing of Environment*, 64, 1–19.

[60] Petitjean, F., Gancarski, P., Masseglia, F., and Forestier, G., 2010, "Analysing satellite image time series by means of pattern mining", *Lecture Notes in Computer Science*, 6283, 45–52.

[61] Petitjean, F., Kurtz, C., and Gancarski, P., 2012, "Spatio-Temporal Reasoning for the Classification of Satellite Image Time Series", in *Pattern Recognition Letters*, 14 pages.

[62] Petitjean, F., Inglada, J., and Gancarski, P., 2011, "Satellite Image Time Series Analysis under Time Warping", *IEEE Transactions on Geoscience and Remote Sensing*, 50(8), 3081–3095.

[63] Phillips, R.D., Watson, L.T., and Wynne, R.H., 2007, "Hybrid image classification and parameter selection using a shared memory parallel algorithm", *Computers & Geosciences*, 33(7), 875–897.

[64] Plaza, A.J., and Chang, C.-I., 2007, *High Performance Computing in Remote Sensing*, CRC Press.

[65] Plisnier, P.D., Serneels, S., and Lambin, E.F., 2000, "Impact of ENSO on East African ecosystems: multivariate analysis based on climatologic and remote sensing data", *Global Ecology and Biogeography Letters*, 9, 481–497.

[66] Ramer, U., 1972, "An iterative procedure for the polygonal approximation of planar curves", *Computer Graphics and Image Processing*, 1, 244–256.

[67] Rice, J.R., 1967, "On the Construction of Polyalgorithms for Automatic Numerical Analysis", in *Interactive Systems for Experimental Applied Mathematics*, 301–313.

[68] Rice, J.R., 1969, "A Polyalgorithm for the Automatic Solution of Nonlinear Equations", in *Proc. of the 1969 24th National Conference*, 179–183.

[69] Rice, J.R., 2014, *Numerical Methods in Software and Analysis*, Elselvier.

[70] Rice, J.R., and Rosen, S., 1966, "NAPSS – a numerical analysis problem solving system", in *Proc. of the ACM National Conference*, 51–56.

[71] Richards, J.A., 1984, "Multitemporal analysis of Landsat imagery for monitoring forest cutovers in Nova Scotia", *Canadian Journal of Remote Sensing*, 11, 188–194.

[72] Rudin, L., Osher, S., and Fatemi, E., 1992, "Nonlinear total variation based noise removal algorithms", *Physica D: Nonlinear Phenomena*, 60, 259–268.

[73] Saxena, R., Watson, L.T., Thomas, V.A., and Wynne, R.H., 2017, "Scaling constituent algorithms of a trend and change detection polyalgorithm", in *Proc. High Performance Computing Symp. (HPC 2017), 2017 Spring Simulation Multiconference, Soc. for Modelling and Simulation Internat., Vista, CA*, 12 pages.

[74] Saxena, R., Watson, L.T., Wynne, R.H., and Thomas, V.A., 2017, "Scalability of land use monitoring codes", in *Proc. 2017 Internat. Conf. on Scientific Computing, H.R. Arabnia, M.R. Grimaila, D.D. Hodson, and F.G. Tinetti (eds.) CSREA Press, Las Vegas, NV*, 3–9.

[75] Serneels, S., Said, M., and Lambin, E.F., 2001, "Land-cover changes around a major East African wildlife reserve: the Mara ecosystem", *International Journal of Remote Sensing*, 22, 3397–3420.

[76] Shatkay, H., and Zdonik, S., 1996, "Approximate queries and representations for large data sequences", in *Proc. of the 12th IEEE International Conference on Data Engineering*, 546–553.

[77] Thomson, F., Davis, G., and Colwell, J.E., 1980, "Detection and measurement of changes in the production and quality of renewable resources", in *USDA Forest Service Final Report 145300–4-F*, ERIM, Ann Arbor, MI, USA.

[78] Toney, C., Liknes, G., Lister, A., and Meneguzzo, D., 2012, "Assessing alternative measures of tree canopy cover: photo-interpreted NAIP and ground-based measures", in *Proc. of Monitoring Across Borders: 2010 Joint Meeting of the Forest Inventory and Analysis (FIA) Symposium and the Southern Mensurationists. Edited by W. McWilliams and F.A. Roesch. USDA Forest Service, Southern Research Station, Asheville, North Carolina, e-Gen. Tech. Rep. SRS-157*, 209–215.

[79] Tucker, C.J., 1979, "Red and photographic infrared linear combinations for monitoring vegetation", *Remote sensing of Environment*, 8(2), 127-150.

[80] Verbesselt, J., Hyndman, R., Newnham, G., and Culvenor, D., 2010, "Detecting trend and seasonal changes in satellite image time series", *Remote Sensing of Environment*, 114, 106–115.

[81] Verbesselt, J., Hyndman, R., Zeileis, A., and Culvenor, D., 2010, "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series", *Remote Sensing of Environment*, 114, 2970–2980.

[82] Vintrou, E., Ienco, D., Begue, A., and Tesseire, M., 2013, "Data mining, a promising tool for large area cropland mapping", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(5), 2132–2138.

[83] Vintrou, E., Desbrosse, A., Begue, A., Traore, S., Baron, C., and Seen, D.L., 2012, "Crop area mapping in West Africa using landscape stratification of MODIS time series and comparison with existing global land products", *International Journal of Applied Earth Observation and Geoinformation*, 14, 83–93.

[84] Vlasveld, R.Q., 2014, *Temporal Segmentation using Support Vector Machines in the context of Human Activity Recognition*, Utrecht University.

[85] Whittle, P., 1951, *Hypothesis Testing in Time Series Analysis*, Uppsala: Almqvist & Wiksells Boktryckeri AB.

[86] Wold, H., 1938, *A Study in the Analysis of Stationary Time Series*, Almqvist & Wiksell.

[87] Woodcock, E.C., Allen, R., Anderson, M., Belward, A., Bindschadler, R., Cohen, W., Gao, F., Goward, S.N., Helder, D., Helmer, E.M., Nemani, R., Oreopoulos, R., Schott, J., Thenkabail, P., Vermonte, E., Vogelmann, J., Wulder, M., and Wynne, R., 2008, "Free access to Landsat imagery", *Science*, 320, 1011–1011.

[88] Wozniak, M., Grana, M., Corchado, E., 2014, "A survey of multiple classifier systems as hybrid systems", *Information Fusion*, 16, 3–17.

[89] Zhan, X., Sohlberg, A.R., Townshend, J.R.G., DiMiceli, C., Carroll, M.L., Eastman, J.C., Hansen, M.C., and DeFries, R.S., 2002, "Detection of land cover changes using MODIS 250 m data", *Remote Sensing of Environment*, 83, 336–350.

[90] Zhu, Z., and Woodcock, C.E., 2014, "Continuous Change Detection and Classification of Land Cover Using All Available Landsat Data", *Remote Sensing of Environment*, 144, 152–171.