Fast Algorithms for Estimating Aerosol Optical Depth and Correcting Thematic Mapper Imagery *

Hassan Fallah-Adl [1] Joseph JáJá [1†] Shunlin Liang [2]

Institute for Advanced Computer Studies (UMIACS)
University of Maryland, College Park, MD 20742
{hfallaah, joseph, liang}@umiacs.umd.edu

Keywords: High Performance Computing, Scalable Parallel Processing, Remote Sensing, Atmospheric Correction, Aerosol Optical Depth.

Abstract

Remotely sensed images collected by the satellites are usually contaminated by the effects of the atmospheric particles through absorption and scattering of the radiation from the earth surface. The objective of *atmospheric correction* is to retrieve the *surface reflectance* from remotely sensed imagery by removing the atmospheric effects, which is usually performed in two steps. First, the optical characteristics of the atmosphere are estimated and then the remotely sensed imagery is corrected by inversion procedures that derive the surface reflectance.

In this paper we introduce an efficient algorithm to estimate the optical characteristics of the Thematic Mapper (TM) imagery and to remove the atmospheric effects from it. Our algorithm introduces a set of techniques to significantly improve the quality of the retrieved images. We pay a particular attention to the computational efficiency of the algorithm, thereby allowing us to correct large TM images quite fast. We also provide a parallel implementation of our algorithm and show its portability and its scalability on several parallel machines.

*This work is partly supported under the NSF Grand Challenge Grant No. BIR-9318183.
†The work of this author is partially supported by NSF under Grant No. CCR-9103135.
[1]UMIACS and Department of Electrical Engineering.
[2]Department of Geography.

1 Introduction

Remote sensing techniques have been extensively applied in different disciplines. However, the radiation from the earth surface which is highly correlated with surface inherent properties are largely contaminated by the atmosphere. It has been demonstrated [1, 2] that the removal of atmospheric effects can significantly improve the accuracy of image classification. So far, no operational method is avaliable to remove the atmospheric effects on large scales. A general scheme and some efficient algorithms are described in our previous work [3]. In this paper, we develop a fast algorithm to estimate the aerosol optical thickness from Thematic Mapper (TM) imagery and present various examples of correcting TM imagery that illustrate the effectiveness of our approach.

bands	1	2	3	4	5	7
central wavelength (nm)	503.0	594.0	677.0	800.0	1710.0	2200.0

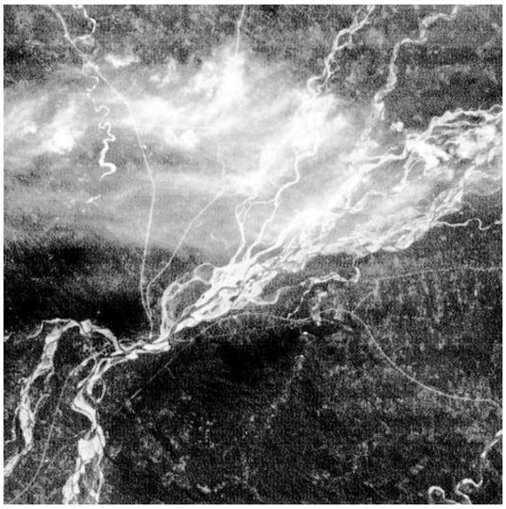
Table 1: Spectral bands of Thematic Mapper (TM) Imagery.

The Thematic Mapper (TM) of Landsats 4 and 5 provides a substantial amount of imagery that has a high spatial resolution (30 meters), and a high spectral resolution and that has been widely used for resource inventory, environmental monitoring, and a variety of applications [4]. The spectral characteristics of TM imagery is summarized in Table 1. The first three channels are in the visible spectrum corresponding to blue, green and red. Channels 4 and 5 are in the near-infrared spectrum and channel 7 is a middle-infrared band. Since channel 6 is in the thermal region which has a different spatial resolution (120 meters) and different physical properties, it is rarely used in the environmental sciences and will not be considered in this study. Figure 1 presents a scene of TM imagery acquired on *Aug. 17, 1989* in the *Amazon Basin area*, which is partially covered by hazy aerosols and some thin clouds. Aerosol has much larger disturbances in shorter wavelength. Although some cloud residuals are visible in channel 7, aerosol scattering effects have almost disappeared. The objective of the atmospheric correction is to retrieve the surface reflectance from observed pixel values at the top of the atmosphere.

2 Background

Assuming that the atmosphere is bounded by a Lambertian surface (i.e., reflects solar energy isotropically), the upward radiance at the top of the cloud-free, horizontally homogeneous atmosphere can be expressed by [5] :

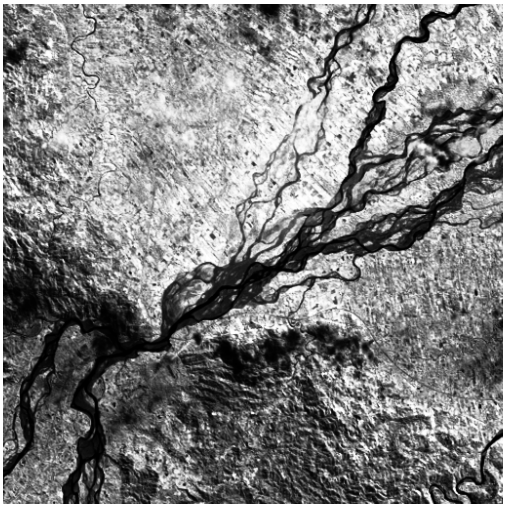
$$L^m = L_0 + \frac{\rho F_d T}{\pi(1 - s\rho)}, \tag{1}$$



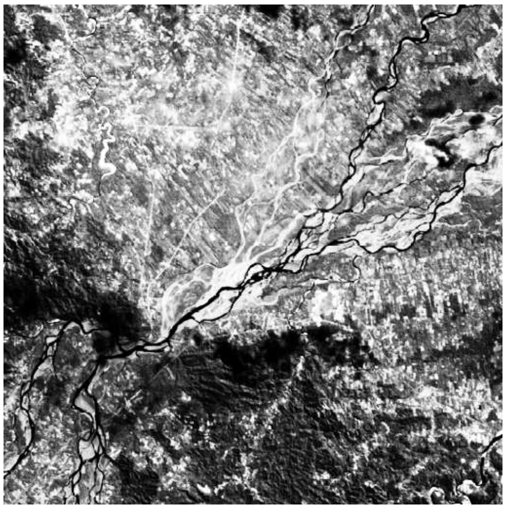
Band 1

Band 2

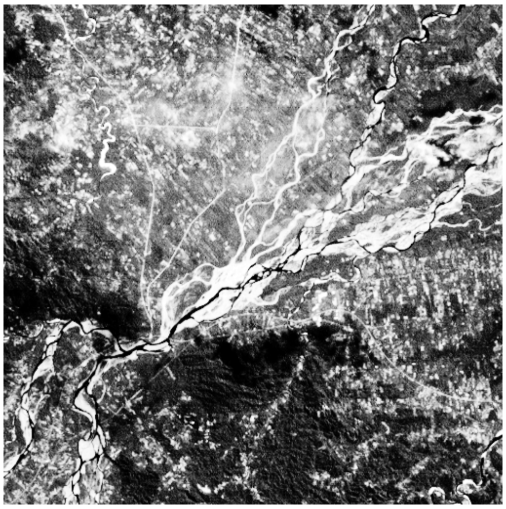
Band 3



Band 4



Band 5



Band 7

Figure 1: TM imagery (512×512).

where L_0 is the upward radiance of the atmosphere with zero surface reflectance (i.e., $\rho = 0$), often called *path radiance*, F_d is the *downward flux* (total integrated irradiance) at the ground, T is the *transmittance* from the surface to the sensor (the probability that a photon travels through a path without being scattered or absorbed), and s is the *spheric albedo* of the atmosphere (the probability that a photon reflected from the surface is reflected back to the surface).

In order to invert ρ from L^m through Eqn. (1) , we need to determine the quantities L_0, F_d, T, and s which are functions of the wavelength, atmospheric optical properties, and a set of locational parameters, such as surface elevation, sensor heights, viewing zenith and azimuth angles, and solar zenith angle. There are two main tasks involved. The first is to estimate the atmospheric properties and the second is to calculate the functions required to invert the surface reflectance ρ.

2.1 Estimating Atmospheric Properties

It is not realistic to assume that simultaneous measurements of all atmospheric optical properties are operationally available due to the rapid variation of the atmosphere. In fact, most of available TM imagery are not accompanied with any simultaneous measurements at all. Climatology data based on very coarse spatial and temporal resolutions can not be used for correcting individual images. The estimation of the atmospheric optical properties from the imagery itself is the only operational scheme to achieve the atmospheric correction. One of the main parameters needed for the correction of TM imagery is the *aerosol optical depth*, which is the measure of atmospheric turbility. If the atmosphere is perfectly clear, aerosol optical depth is zero. When atmosphere becomes more turbid, it becomes larger and its value may be larger than 1. There are several methods for estimating aerosol optical depth from the imagery, including contrast estimation [6, 7], "dark-object" approach [8, 9], but practical implementations of these have not been reported. The so-called *"dark object"* approach is used for this study because of its simplicity and effectiveness [8]. The idea behind this approach is quite simple. We search for pixels with low surface reflectance using TM band 7 [9] in which aerosol effect is negligible, and then we assign a small surface reflectance to those dark pixels and the aerosol optical depth can be figured out from Eqn. (1). Note that in this case the deviation of the assigned reflectance from the *"true"* reflectance will not result in a large uncertainty for the estimation of aerosol optical depth since both are very small. The assumptions behind this approach can be summarized as follows: (1) The image contains at least a few pixels which characterize dense dark vegetation; (2) The atmosphere is bounded by a Lambertian surface as the lower boundary condition; And (3) the assumed aerosol scattering phase function and the single-scattering albedo will not result in significant errors in the retrieval of the aerosol optical depth [9].

2.2 Surface Reflectance Retrieval

Given the aerosol optical depth, the determination of L_0, F_d, T, and s in Eqn. (1) is not a simple task due to the fact that these quantities are related to the solutions of the radiative transfer equation [5], which is an integro-differential equation for which no analytical solution is available. There are several approaches to obtain practical solutions. The first is to use a numerical iterative approach, such as the discrete-ordinate algorithm [10], and the Gauss-Seidel algorithm [11]. The resulting solutions are accurate but the methods involved are computationally very expensive and not feasible for large scale studies. Another approach is to simplify the radiative transfer equation by using approximations, such as the two-stream approximation [12], and the four-stream approximation [13]. These approximation algorithms are computationally efficient, but the accuracy is limited. An alternative is to set up off-line look-up tables [14] for certain input values. With the additional tables, the quantities (L_0, F_d, T, and s) can be efficiently calculated with high accuracy using interpolations. This look-up table approach has been explored in our previous study and further improvements will also be presented in this paper.

2.3 Computational Complexity

Estimating optical depth requires the extensive handling of large amounts of data residing in external storage and hence the optimization of both computation time and I/O time must be considered. To get a better feel about the volume of data involved, we should mention that a single standard TM image that covers an area of size about $180Km \times 180Km$ consists of approximately 36 million pixels per band, which adds up to about 216 million pixels. Therefore, we are dealing with more than 10^{12} pixels for the entire globe. In order to handle such massive amounts of data, our main objectives are to :

- design very efficient serial algorithms that correct different types of landscape in TM imagery.
- develop parallel versions of these algorithms, that are scalable in terms of the number of processors, the number of I/O nodes, and the size of internal memory.

We develop in this paper a new atmospheric correction algorithm that appears to work very well on a variety of images and that satisfies our two stated objectives. The rest of the paper introduces the algorithms and provides examples of our implementation.

3 Estimating Aerosol Optical Depth

In this section we first describe the basic sequential algorithm and then present a number of techniques that were used to improve the quality of the output and to reduce the overall computational complexity.

3.1 Description of Algorithm

Our algorithm for estimating the aerosol optical depth for TM imagery is based on the dark object method. The algorithm operates on windows of size $w \times w$ and can be sketched as follows.

Aerosol Optical Depth Estimation Algorithm :

Step A: *For each window of the input image, determine if it contains a dark object. In the affirmative, estimate the aerosol optical depth for the first five bands.*

Step B: *For each window without a dark object, estimate the aerosol optical depth by interpolating on the neighboring windows with dark objects.*

The determination of the window size w is not simple due to two conflicting requirements. On one hand, increasing the window size will increase the chances of finding a dark pixel in each window and will reduce the overall computational complexity. On the other hand, the larger the window size, the less accurate the computed optical thickness is. In general, atmospheric conditions and the resolution of the image determine the value of w.

In step A, we search for dark objects in each window and if one or more dark pixels are encountered, aerosol optical depth is estimated for that window. We next give the details of executing step A. Note that the algorithm for computing aerosol optical depth given surface reflectance and upward radiance needed for step 4 and 8 will be given later.

Algorithm (Estimate Aerosol Optical Depth for Each Window)

Input: *Pixel values (Upward Reflectance[3]) in the window, location information and look-up tables.*

Output: *Aerosol optical depths of the first five channels for the window.*

begin

1. *Identify all those pixels in the window whose channel 7 (Ch7) values are smaller than a threshold ($L_7^m \leq 0.1$) as dark objects. Exit if no dark pixels are found.*

2. *Calculate the average values of dark pixels in each of channels 1 through 5 and channel 7 (L_1^m, L_2^m, L_3^m, L_4^m, L_5^m, L_7^m).*

3. *Assume the surface reflectance ρ_7 in Ch7 equal to its reflectance at the top of the atmosphere (L_7^m) and calculate surface reflectance in channels 1 and 3 using $\rho_1 = 0.25\rho_7$ and $\rho_3 = 0.50\rho_7$, which are derived from statistical analysis in several test sites [9].*

4. *Estimate aerosol optical depths of channels 1 and 3 (τ_1, τ_3) by using L_1^m, L_3^m, ρ_1, ρ_3, look-up tables. The parameter τ_1 should be larger than or equal to τ_3.*

[3] Input pixel values are digital counts which are first converted to radiance values and then to reflectance units (apparent reflectance).

5. If $\tau_1 < \tau_3$, repeat steps 1 through 5 with smaller thresholds, until $\tau_1 \geq \tau_3$ or the algorithm terminates when the window does not have any dark pixels.

6. Set the aerosol optical depth as $\tau_i = a\lambda_i^{-b}$, where τ_i and λ_i are the aerosol optical depth and the wavelength for channel i, $1 \leq i \leq 5$. Calculate parameters a and b from τ_1, τ_3.

7. Calculate τ_2, τ_4, and τ_5 using $\tau_i = a\lambda_i^{-b}$, $1 \leq i \leq 5$.

8. Check whether $\tau_i^{min} \leq \tau_i \leq \tau_i^{max}$, where $\tau_i^{min} = 0.0$ and τ_i^{max} can be calculated by assuming $\rho_i = 0.0$ and using L_i^m and the look-up tables.

9. If any of the aerosol optical depths is out of bound, fit the exponential curve again by using τ_1 and τ_i^{max}, where i denotes the band whose optical depth is out of range. Repeat this procedure until no optical depth is out of bound.

end

The procedure for determining the aerosol optical depth, given surface reflectance and upward radiance is very similar to the method for determining the surface reflectance given aerosol optical depth and upward radiance [3]. We use look-up tables to compute L_0, F_d, T and s (as functions of aerosol optical depth) by interpolation followed by computing the upward radiance (as a function of aerosol optical depth) using Eqn. (1). We then use spline interpolation of degree one on the upward radiance to compute the aerosol optical depth.

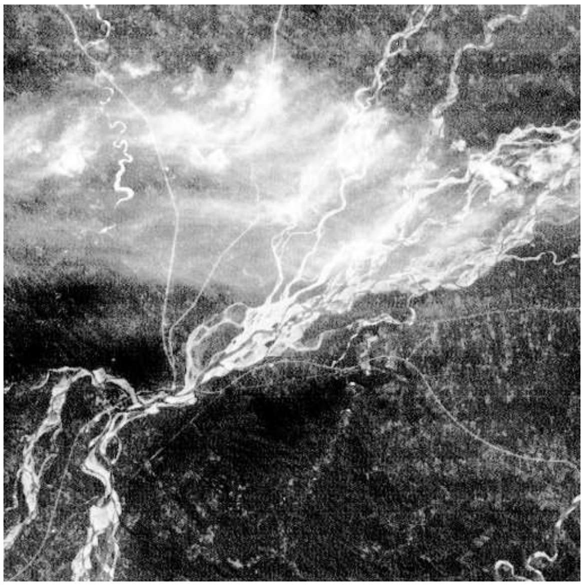
The algorithm just described handles the case when there is at least one dark object in the window. Otherwise, we continue in the main algorithm with step B where we estimate optical depth by interpolating on the optical depths of the neighboring windows. This can be repeated until we have the optical depth for all the windows.

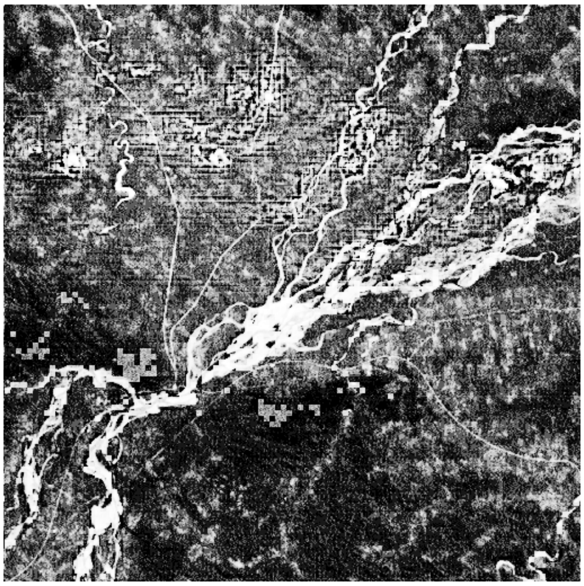
A straightforward implementation of the strategy sketched above coupled with our earlier atmospheric correction algorithm [3] reveals a number of shortcomings of the algorithm. These shortcomings include:

Large variations in the quality of corrected images. The algorithm performed well only when the atmospheric conditions did not change rapidly, the optical depth was not high, and when the image did not include any cloud or water. Wherever there was a sharp change in optical depth, window effects appeared on corrected images (Figure 2.b). Also, the algorithm was not able to correct water areas, specially when the image had a large body of water.

Channels 4 and 5 were not corrected appropriately. Not only did the algorithm not correct Channels 4 and 5, but in fact it typically caused adverse effects on the tested images.

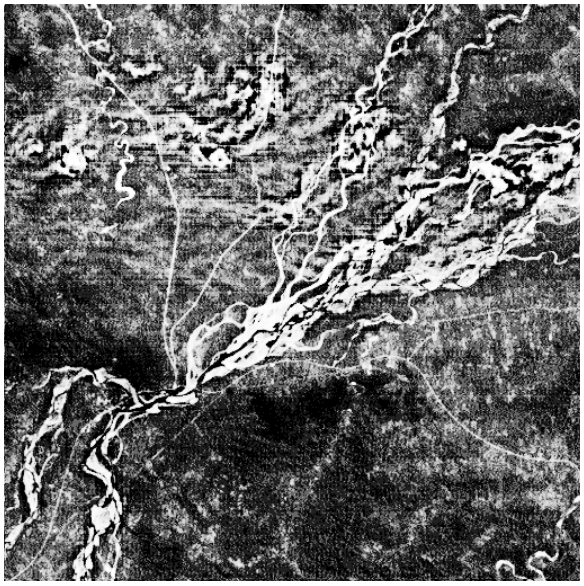
We modified the algorithm to take care of these problems as well as improve its computational efficiency.

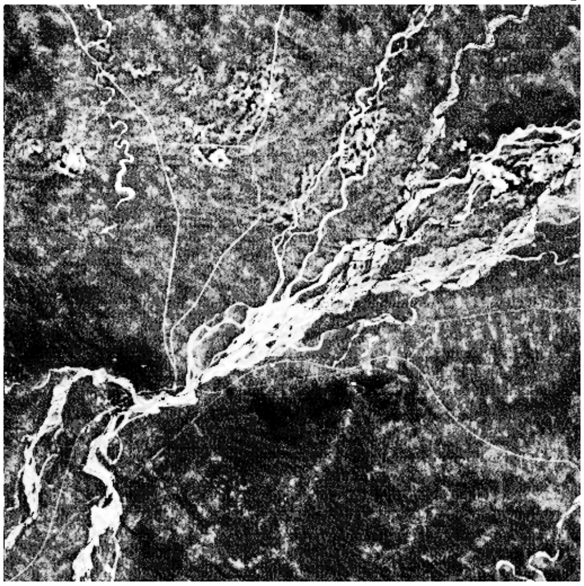




a.Band 1, **Before** Correction

b.Band 1, **After** Correction, **No** Filtering





c.Band 1, **After** Correction, **with** Filtering

d.Band 1, **After** Correction, **Final** Version

Figure 2: TM imagery (512×512).

3.2 Modified Algorithm

The following techniques were used to handle the problems cited earlier.

Moving Window. An attempt to incorporate a number of smoothing techniques into our algorithm (just before running the correction algorithm) was not successful in removing the window effects (Figure 2.c). Instead we used a moving window technique that is somewhat similar to image convolution. More precisely, we build a window of appropriate size around each pixel and apply the same window algorithm as before. We find the optical depth of each pixel (not a window), and therefore we expect to generate more accurate estimates. It turns out with this modification, there are no window effects and the tested images were corrected much more accurately than before (Figure 2.d).

A major problem with our moving window method is the resulting increase in the number of computations (by a factor of about w^2). We will later present a technique to counter this problem.

To get rid of what appeared to be a random noise, we perform a smoothing step once at the end of the algorithm, which is done over a small window of pixels. Interpolation in step B in the main algorithm can be replaced by an averaging operation. We thus combined step B with the smoothing step to further improve the timing of the algorithm.

Correcting Water Areas. Unlike other surfaces, water has in general higher reflectance in the first three channels and a very low reflectance in the other channels. Therefore water will likely be detected as a dark object by our algorithm and, because of high reflectance in lower channels, it will lead to a higher estimate of the optical depth. Thus, after correction, water areas will become dark in the first three channels, an incorrect output. To solve this problem we do not choose water pixels as dark objects. In other words if a pixel's vegetation index is less than some threshold $((L_4^m - L_3^m)/(L_4^m + L_3^m) < 0.1)$, we detect it as a water pixel and remove it from the list of dark objects.

With this new approach, all the windows that fall completely in water areas, will not have any dark objects. In this case, we determine the optical depth in the following way. If a window does not have any dark object and its corresponding pixel is a water pixel, we examine the pixel's reflectance in channel 3 (since it is cleaner than channels 1 and 2). If the reflectance is high, we conclude that it is a shallow, and clean water and we assign zero optical depth to it. Therefore in the corrected image its reflectance will also be high in the first three channels. But if its reflectance is low, we conclude that it is deep or contaminated water and we calculate the optical depth, assuming $\rho = 0$. Therefore in the corrected image its reflectance will also be low.

By applying the above technique we were able to extend the algorithm's scope to images with different kinds of water areas. Also it even improved the quality of the correction in land areas because occasionally there are scattered water pixels in land areas, which should be excluded from the dark objects list. Otherwise, they can increase the value of the optical depth.

Correcting Channels 4 and 5. As mentioned before, channels 4 and 5 were not corrected and often the algorithm had a reverse effect on them. This is mostly due to the fact that, in channels 4 and 5, as you increase the optical thickness, the surface reflectance increases for high values of upward radiance. Thus we can only remove water and gaseous effects from the image in channels 4 and 5 and it is better to use zero optical depth for these channels. Also we should note that channels 4 and 5 look at least as clean as channel 7 and hence we should not expect substantial scattering correction of those channels. This modification results in a more computationally efficient algorithm as well.

Simplifying Optical Depth Estimation for channel 2. We have used the model $\tau_i = a\lambda_i^{-b}$ primarily to estimate the optical depth for channels 4 and 5. Now that we have decided to assign zero optical depth for these channels, we do not need to use that curve and we can instead estimate the optical depth for channel 2 by averaging the optical depths of channels 1 and 3. Also we do not need to check for $\tau_i^{min} \leq \tau_i \leq \tau_i^{max}$, because we know that τ_1 and τ_3 satisfy these boundary conditions and thus τ_2 will satisfy these conditions as τ_2 is just the average. Obviously $\tau_4 = \tau_5 = 0$ do not exceed the maximum values.

3.3 Improving the Efficiency of the Modified Algorithm

The next set of techniques allow a significant improvement of the computational efficiency of the algorithm.

Smart Window Move. As mentioned before, we find the optical depth for each pixel and hence the number of operations are of the order of N^2w^2. However we view the window as sliding from left to right, one column at a time, while performing some operation at each move. By recording the intermediate results at previous columns we reduce the number of operations to the order of N^2w.

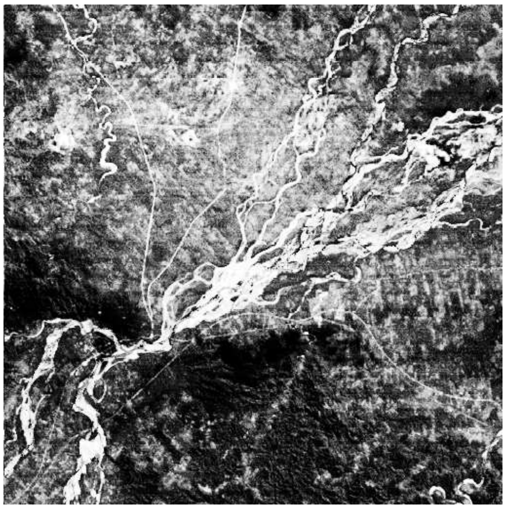
The same technique can be used for the smoothing step, which can be viewed as matrix convolution where the values of all elements in the convolution matrix are equal to $1/w^2$. This later technique made the algorithm substantially faster.

Eliminating the necessity to check $\tau_1 \geq \tau_3$. A potential computational bottleneck of the algorithm is the step to check the condition $\tau_1 \geq \tau_3$ and to repeat the steps prior to it using smaller thresholds, whenever the condition is not satisfied. After examining the situations under which this condition is not satisfied, we decided to do the following. If the condition is not satisfied and it is a water pixel, then we follow the same procedure as in the case of a window without any dark object where the corresponding pixel is a water pixel. Otherwise we accept them as they are[4].

Reading Look-up Tables Faster. Another technique to speed-up the program is to replace the formatted reading of look-up tables with unformatted read. We modified the look-up tables off-line to be suitable for unformatted I/O.

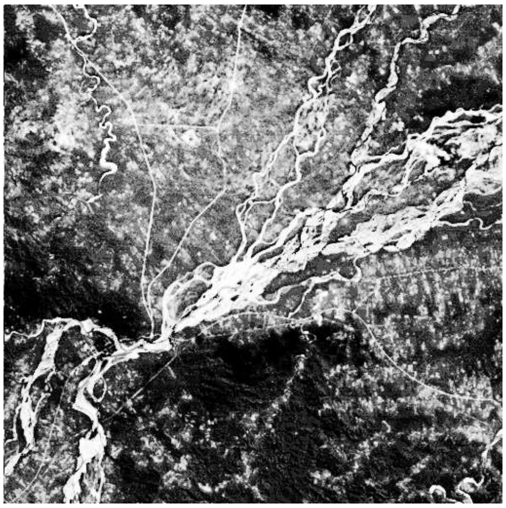
[4]Some experiments report that τ_3 may be larger than τ_1 occasionally. Since more investigations are needed to identify and explain these cases, we do not explore this situation further in this study.

Band 2, **Before** Correction

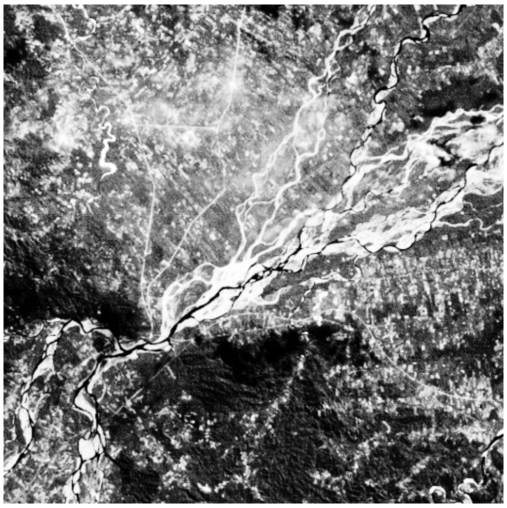


Band 2, **After** Correction

Band 3, **Before** Correction



Band 3, **After** Correction



Band 7

Figure 3: TM imagery (512×512).

4 Combining Parameter Estimation and Correction Algorithms

In an earlier work, we designed an atmospheric correction algorithm [3] that assumed constant optical depth over windows of a suitable size. As we have seen before, this assumption may not always be valid. We therefore need to develop an algorithm that makes no such assumption, which will significantly increase its computational complexity if the optical depth is to be computed for each pixel in a straightforward manner. Toward this end, we introduce efficient techniques that exploit the fact that most of the operations in the atmospheric correction algorithm are also present in the parameter estimation algorithm. As a result, we end up with a very efficient algorithm that combines parameter estimation and atmospheric correction. The details are given next.

Given that the optical depth varies from pixel to a neighboring pixel, the atmospheric correction algorithm has to perform an interpolation for each pixel to generate an estimate of the optical depth at that pixel, a costly operation. We get around this problem as follows.

We digitize the optical depth values by scaling these values by a large integer and then create a look-up table, where each entry in the table corresponds to a digitized optical depth value. Each row in the look-up table can hold the result of the interpolation for the corresponding optical depth and includes a validation flag.

At the beginning of the correction algorithm we set the validation flag to false for all the entries in the table. Wherever we need to interpolate for an optical depth value, we check the corresponding validation flag in the table. If the flag is true then we read the information directly from the table, otherwise we perform the interpolation, save the result in the table, and set the corresponding flag to true for future references.

Optical depth values in the algorithm range from 0.0 to 2.0 and a table of size 500 provides sufficient precision. Therefore, in the worst case, the new algorithm will require only 500 interpolations to compute all possible values of the optical depth, while our previous algorithm would have required hundreds of thousands of interpolations for a standard TM image. For the remainder of this paper, we only refer to the combined algorithm unless otherwise stated.

Experimental results show that our algorithm requires less than 60 minutes to correct standard TM image with 50 M Pixels per band (300 M Pixels total) on an IBM RS6000 workstation. To speedup the computation, we develop an implementation on a parallel machine, a topic addressed in the next section.

5 Parallel Implementation

In general, the entire image will not fit in the main memory and hence we have to deal with the issue of storing and accessing the image externally. In the sequential case, there is basically one way to store the image on the disk available. For a parallel machine, we seek to achieve an efficient layout of the input imagery on the disks and an efficient mapping of the computation across the processors in such a way the total computation and I/O time is minimized. In this section we present different parallel implementations of our algorithm under two I/O models representing possible configurations on current parallel machines. We later analyze and compare the performance of the different algorithms.

5.1 Parallel File Systems

Parallel file systems are usually installed on a set of disks which can physically be configured as: (1) *shared disks*, (2) *distributed disks*, or (3) *semi-distributed disks*.

In the *shared disks* configuration, we have a number p of computation nodes P_0, P_1, ..., P_{p-1} and a number d of I/O nodes, N_0, N_1, ..., N_{d-1}, each holding one or more disks, connected through an interconnection network. In some *shared disks* configurations, such as the *CMMD* on the CM-5 machines and the *pfs* on the PARAGON machines, users can not control the file distribution across the disks and the data is striped over the disks based on some predefined system parameters. With other parallel file systems, users can dynamically control the file distribution over the disks. This flexibility can lead to more efficient implementations of some algorithms. The *piofs* parallel file system on the SP-1 and SP-2 machines with dedicated I/O nodes is an example of this type of systems.

In the *distributed disks* architecture, there are no dedicated I/O nodes and one or more disks are attached to each of the computation nodes, which means that every node is responsible for both computation and I/O. The *piofs* parallel file system on the SP-1 and SP-2 machines with the disks distributed over all the nodes, is an example of this type of architecture, which allows the user to control the file distribution dynamically.

In the *semi-distributed disks* configuration, the I/O nodes are a subset of the computation nodes, and thus these nodes are responsible for both computation and I/O, while the remaining nodes are dedicated only for computation. The *piofs* on the SP-1 and SP-2 machines with the disks distributed over a small subset of the nodes, is an example of this architecture.

Some parallel file systems provide both *independent* and *synchronous* I/O. To access a file in synchronous mode, the file should be opened globally by all nodes. Synchronous mode allows nodes to access sequential portions of a file and the data read from or written into the file come from each node in sequence, from node 0 to the highest numbered node. For example, in a read operation, each node requests an arbitrary number of bytes of data and the data is distributed across the nodes sequentially by node number. Each node's block of data begins where the preceding node's data block ended. The *CMMD* and *pfs* both provide synchronous I/O mode, while the *piofs* does not support synchronous I/O but allows users to divide a file logically into multiple subfiles.

5.2 Parallel Algorithm

We now sketch the parallel implementations of our algorithm and how they achieve their computation and I/O scalability. All the algorithms are designed in the *Single*

Program Multiple Data (SPMD) model, and hence each processor runs the same code but on different parts of the image.

The I/O performance can be estimated by using the number of *passes through the data* and the number of *disk accesses*. Our parallel implementations of the algorithm to be discussed shortly, require only one pass through the image and hence we only need to minimize the number of disk accesses. Furthermore, we should balance the computation among the nodes and minimize the interprocessor communication time.

We balance the computation by partitioning each image equally among the processors. To minimize the number of disk accesses [3], each processor during each iteration processes a slab (as opposed to a block) consisting of the maximum possible number of consecutive rows that can fit in its internal memory. More precisely, each processor reads a slab sequentially, process it, and writes back the result. This procedure is repeated until the entire image is processed. Clearly, the number of disk accesses is minimum and thus the total I/O transfer time is optimal.

Interprocessor communication time depends on the image partitioning policy among the computation nodes. There are basically two approaches. The first to be called *fine-grain* partitioning, processes $r \times p$ consecutive rows during each iteration such that the first r rows goes to the first processor, the second r rows goes to the second processor, and so on, where r is the maximum number of rows that can fit in the main memory of each node. This method requires synchronization after each read or write, and requires interprocessor communication as each processor must get $\frac{w-1}{2}$ rows from its neighbors (if they exist) during each iteration, where w is the window size used by our algorithm. Such an approach seems to be effective for the shared disks configuration, specially with optimized synchronous I/O mode.

In the second method, to be called *coarse-grain* partitioning, we partition the $N \times M$ input image into p equal subimages, each subimage consisting of $\frac{N}{p}$ consecutive rows. Each processor works on a subimage independent of the other processors. During each iteration, a processor reads not only r rows of its subimage but also $\frac{w-1}{2}$ extra rows from the top and bottom boundaries of the slab. Under this scheme, no interprocessor communication or synchronization is required, but $w - 1$ extra rows should be read during each iteration. Such a scheme seems to be well suited for distributed or semi-distributed disks configuration with dynamic file distribution capability over the disks.

5.3 Experimental results

The performance data obtained by running our codes on a 16 node IBM SP-2, a 128 node IBM SP-1, a 512 node Thinking Machine CM-5, and a 512 node Intel PARAGON machines are plotted in Figures 4 through 9. All the results are for a standard TM image with 50 million pixels per band and thus the input consists of 300 M pixels. In the remaining discussion, computation time also includes communication time , unless otherwise stated.

Figure 4 shows the computation and I/O times on a SP-2 machine with *distributed disks* configuration which allows dynamic data distribution over the disks. Obviously, the computation time scales very well with the number of processors and is slightly

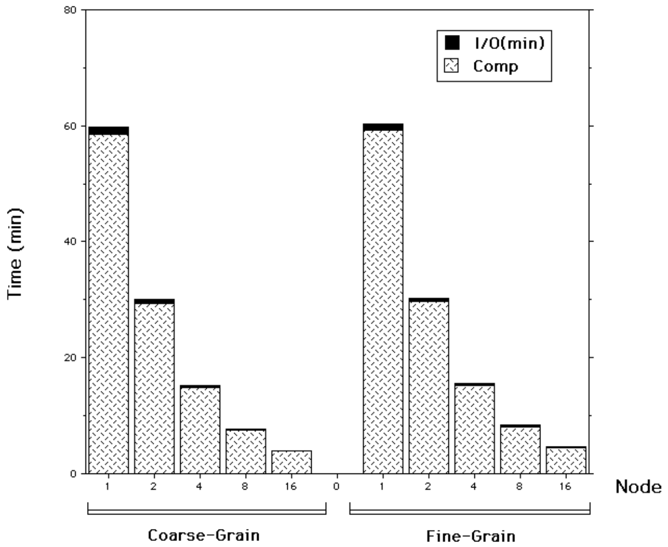


Figure 4: Computation and I/O times for a standard TM imagery on a SP-2 machine with distributed disks configuration, for different number of processors.

higher for *fine-grain* approach because of the extra time needed for the communication and synchronization.

The I/O time is negligible, compared to the computation time and we do not expect it to scale with the number of processors, but to decrease slightly as we increase the number of nodes, because all the nodes participate in the I/O even if we use a subset of nodes for computation. More importantly, the I/O time for *coarse-grain* approach is smaller, because this approach allows us to use the dynamic data distribution feature of the *piofs* file system.

Figure 5 shows the computation time on a SP-1 machine with *semi-distributed disks* configuration. The computation time for both algorithms scale well, but it is slightly higher for the fine-grain approach. The I/O time is not shown, because it heavily depends on the machine load, as a result of the fact that the I/O nodes are also computation nodes.

The performance results for the *fine-grain* approach on a CM-5 machine with a *shared disks* configuration is shown in Figure 6. clearly, the computation time scales well and the I/O time decreases by a small amount as you increase the number of processors, which is consistent with our expectation because the number of I/O nodes remains constant. Also, as we increase the number of processors, the I/O time dominates the computation time.

The *coarse-grain* algorithm did not perform well on CM-5 machine because the parallel file system on CM-5 is optimized for synchronous I/O and does not perform well in independent mode.

Figure 7 compares the estimation and correction times on the CM-5 machine, for

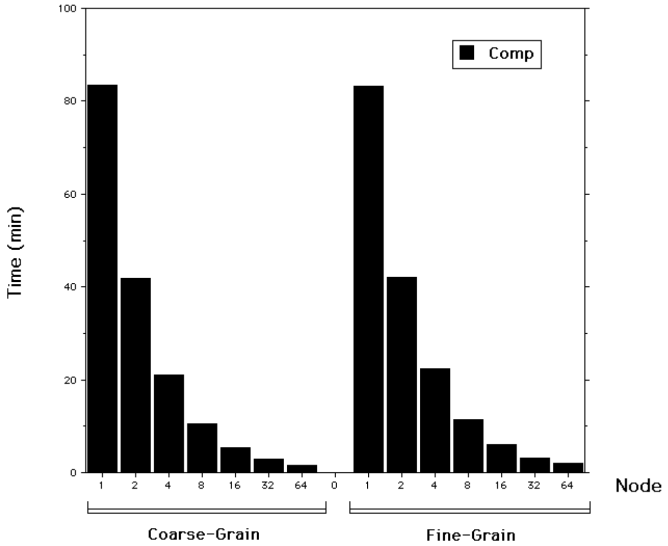


Figure 5: Computation time for a standard TM imagery on a SP-1 machine with semi-distributed disks configuration, for different number of processors.

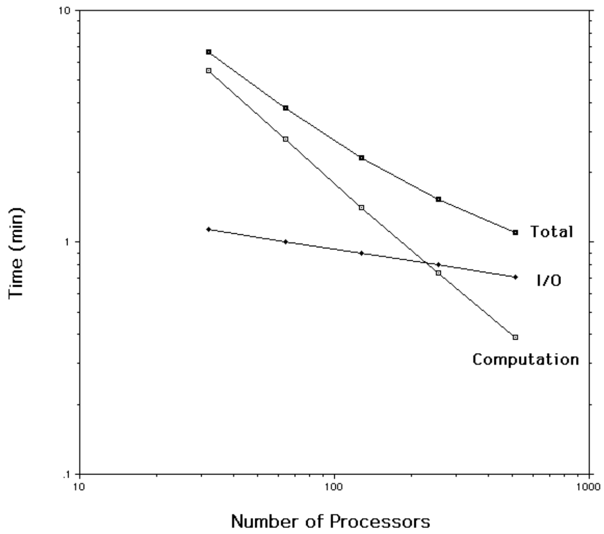


Figure 6: Computation, I/O, and total times for the fine-grain approach with a standard TM imagery as input on a CM-5 machine with shared disks configuration, for different number of processors.

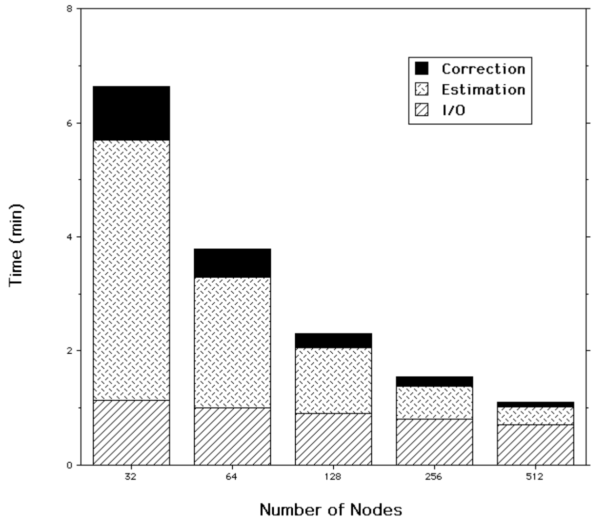


Figure 7: Estimation, Correction, and I/O times for a standard TM imagery on a CM-5 machine for different number of processors

different number of processors. As before, we can see that both the estimation and correction times scale well and the correction time takes a small portion of the overall computation time.

Figure 8 shows the computation and I/O times on a PARAGON machine with a *shared disks* configuration. The computation time for *coarse-grain* approach scales well up to 512 nodes but it scales only up to 128 nodes for *fine-grain* algorithm and that is because of the fact that the *fine-grain* approach requires synchronization and communication in each iteration which are expensive operations on a large PARAGON machine with mesh structured network.

On the PARAGON machine, Unlike the CM-5, the I/O time increases for larger number of nodes, even though both machines have the *shared disks* configuration. This inconsistency is mainly the result of the architectural differences in the interconnection networks of the two machines. The interconnection network on our PARAGON machine is a 16×32 mesh and 16 I/O nodes are connected along one side (with 16 nodes). Therefore, whenever the number of nodes is above 16, there will be contention over the network and increasing the number of processors will make the problem worse. On the other hand, the CM-5 machine uses a fat tree structure, in which the network bandwidth increases proportional to the number of nodes.

Obviously, the total time scales only up to 64 nodes because of the I/O behavior.

Figure 9 compares the total times for different machines. The *coarse-grain* approach is used for all the machines, except the CM-5, for which we have used the *fine-grain* algorithm.

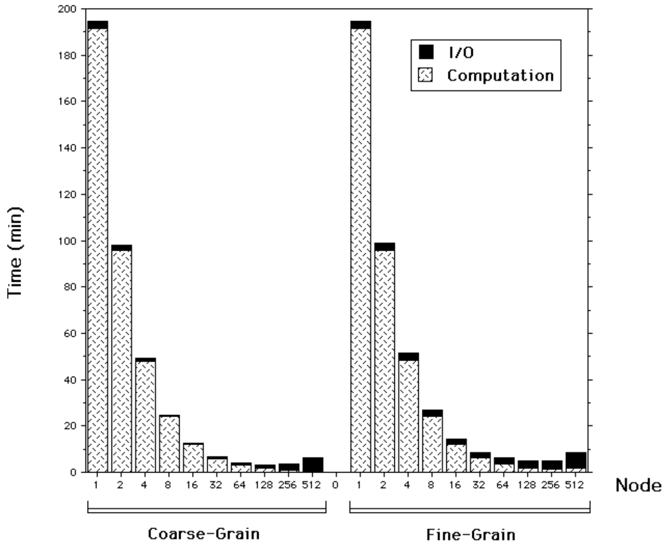


Figure 8: Computation and I/O times for a standard TM imagery on a PARAGON machine for different number of processors

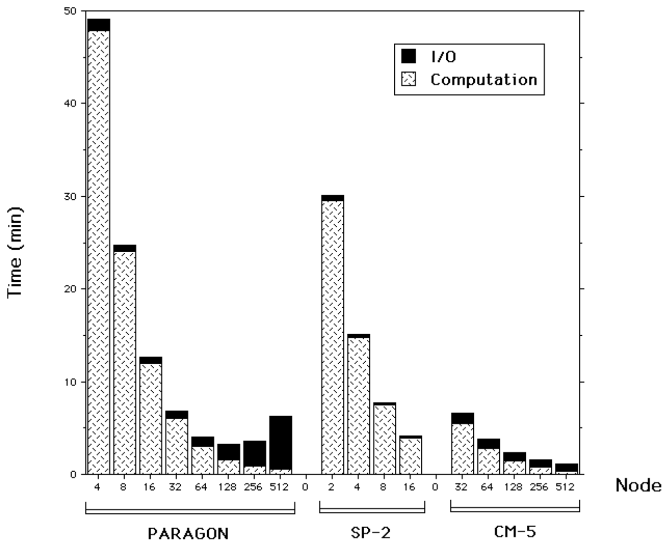


Figure 9: Performance comparison of the total times for a standard TM imagery on different machines for different number of processors

6 Conclusion

We introduced an efficient algorithm to estimate the optical characteristics of the Thematic Mapper (TM) imagery and to remove the atmospheric effects from it. We also presented the parallel implementations of our algorithm, which are scalable and I/O optimal. Experimental results on different parallel machines showed their portability and scalability.

This work constitutes a part of a large multidisciplinary grand challenge project on applying high performance computing to land cover dynamics. Other aspects include parallel algorithms and systems for image processing and spatial data handling with emphasis on object oriented programming and parallel I/O of large scale images and maps.

Acknowledgment

Dr. Yoram Kaufman in NASA Goddard Space Center has continuously provided us with valuable support and advice. The authors are grateful to acknowledge his help.

This research was performed in part using the CACR, 512 node PARAGON machine, operated by Caltech on behalf of the Center for Advanced Computing Research. We have utilized the 512 node CM-5 machine at the National Center for Supercomputing Application (NCSA), University of Illinois at Urban-Champaign. We also like to thank the Argonne National Laboratory at Argonne, Illinois for letting us to use their 128 node SP-1 machine.

We would like to acknowledge the use of a 16-node IBM SP-2-TN2, which was provided by an IBM Shared University award and an NSF Research Infrastructure Initiative Grant No. CDA9401151.

References

[1] R. S. Fraser and Y. J. Kaufman, *The relative importance of scattering and absorption in remote sensing,* IEEE Trans. Geosciences and Remote Sensing, 1985, 23:625-633.

[2] R. S. Fraser, O. P. Bahethi, and A. H. Al-Abbas, *The effect of the atmosphere on classification of satellite observations to identify surface features,* Remote Sens. Environment, 1977, 6:229.

[3] H. Fallah-Adl, J. JáJá, S. Liang, Y. J. Kaufman, and J. R. G. Townshend, *Efficient algorithms for atmospheric correction of remotely sensed Data*, In Proceedings Supercomputing '95, IEEE Computer Society Press, December 1995.

[4] J. R. G. Townshend, J. Cushnie, J. R. Hardy, and A. Wilson, *Thematic Mapper data: Characteristics and use*, Natural Environment Research Council, Swindon, 1983.

[5] S. Chandrasekar, *Radiative Transfer,* London: Oxford University Press, 1960.

[6] Y. J. Kaufman and J. H. Joseph, *Determination of surface albedos and aerosol extinction characteristics from satellite imagery,* J. Geophys. Res., 1982, 20:1287-1299.

[7] D. Tanre, P. Y. Deschamps, C. Devaux, and M. Herman, *Estimation of Saharan aerosol optical thickness from blurring effects in Thematic Mapper data,* J. Geophys. Res., 1988,93:15,955 - 15,964.

[8] Y. J. Kaufman and C. Sendra, *Automatic atmospheric correction,* Intl. Journal of Remote sensing, 1988, 9:1357-1381.

[9] Y. J. Kaufman, L. A. Lorraine, B. C. Gao, and R. R. Li, *Remote sensing of aerosol over the continents: Dark targets identified by the 2.2 um channel*, in preparation, 1995.

[10] J. Lenoble, *Radiative Transfer in Scattering and Absorbing Atmospheres: Standard Computational Procedures,* A. Deepak Publ., Hampton, Virginia, 1985.

[11] S. Liang and A. H. Strahler, *Calculation of the angular radiance distribution for a coupled atmosphere and canopy,* IEEE Trans. Geosci. Remote Sens., 1993, 31:491-502.

[12] W. E. Meador and W. R. Weaver, *Two-stream approximations to radiative transfer in planetary atmospheres: A unified description of existing methods and a new improvement,* J. Atmos. Sci., 1980, 37:630-643.

[13] S. Liang and A. H. Strahler, *Four-stream solution for atmosphere radiative transfer over an non-Lambertian surface,* Appl. Opt., 1994, 33:5745-5753.

[14] R. S. Fraser, R. A. Ferrare, Y. J. Kaufman, B. L. Markham, and S. Mattoo, *Algorithm for atmospheric corrections of aircraft and satellite imagery,* Intl. Journal of Remote sensing, 1992, 13:541-557.