# Efficient Algorithms for Atmospheric Correction of Remotely Sensed Data*


Hassan Fallah-Adl [1]
hfallaah@umiacs.umd.edu

Joseph JáJá [1†]
joseph@umiacs.umd.edu

Shunlin Liang [2]
liang@umiacs.umd.edu

Yoram J. Kaufman [3]
kaufman@climate.gsfc.nasa.gov

John Townshend [2]
jt59@umail.umd.edu

Institute for Advanced Computer Studies (UMIACS)
University of Maryland, College Park, MD 20742



**Keywords:** High Performance Computing, Atmospheric Correction, Parallel I/O, Scalable Parallel Processing, Remote Sensing.



**Abstract**

Remotely sensed imagery has been used for developing and validating various studies regarding land cover dynamics such as global carbon modeling, biogeochemical cycling, hydrological modeling, and ecosystem response modeling. However, the large amounts of imagery collected by the satellites are largely contaminated by the effects of atmospheric particles through absorption and scattering of the radiation from the earth surface. The objective of *atmospheric correction* is to retrieve the *surface reflectance* (that characterizes the surface properties) from remotely sensed imagery by removing the atmospheric effects. Atmospheric correction has been shown to significantly improve the accuracy of image classification.

This problem has received a considerable attention from researchers in remote sensing who have devised a number of solution approaches. Sophisticated approaches are computationally demanding and have only been validated on



*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.
[3]Laboratory for Atmospheres, code 913, NASA Goddard Space Flight Center.

a very small scale. We introduce a number of computational techniques that lead to a substantial speedup of an atmospheric correction algorithm based on using look-up tables that are generated from radiative transfer computations. Excluding I/O time, the previous known implementation processes one pixel at a time and requires about 2.63 seconds per pixel on a SPARC-10 machine, while our implementation is based on processing the whole image and takes about 4-20 microseconds per pixel on the same machine. We also develop a parallel version of our algorithm that is scalable in terms of both computation and I/O. Experimental results obtained show that a Thematic Mapper (TM) image (36 MB per band, 5 bands need to be corrected) can be handled in less than 4.3 minutes on a 32-node CM-5 machine, including I/O time.

# 1 Introduction

Data from the Landsat series of satellites have been available since 1972. The primary source of data from the first three satellites was the Multispectral Scanner System (*MSS* ). The Thematic Mapper (TM) of Landsats 4 and 5 represents a major improvement compared with the *MSS* in terms of spectral resolution (4 wave-bands for *MSS*, 7 narrower wave-bands for TM), and spatial resolution (79 meters for *MSS*, and 30 meters for TM). The TM data have been widely used for resource inventory, environmental monitoring, and a variety of other applications [1]. Since 1979, the Advanced Very High Resolution Radiometers (AVHRR) on board of the National Oceanic and Atmospheric Administration (NOAA) series of satellites have been in continuous polar orbit. AVHRR data have become extremely important for global studies because they carry multiple bands in the visible, the infrared and the thermal spectrum, and a complete coverage of the Earth is available twice daily with 1.1 km resolution at nadir and from two platforms. AVHRR has allowed us for the first time to improve our studies of the earth surface from the regional scale to the global scale using remote sensing techniques [1, 2].

The radiation from the earth surface, which highly characterizes surface inherent properties, are largely contaminated by the atmosphere. The atmospheric particles (aerosols and molecules) scatter and absorb the solar photons reflected by the surface in such a way that only part of the surface radiation can be detected by the sensor. On the other hand, atmospheric particles scatter the sunlight into the sensor's field of view directly, resulting in a radiation that does not contain any surface information at all. The combined atmospheric effects due to scattering and absorption are wavelength dependent, vary in time and space, and depend on the surface reflectance and its spatial variation [3]. For TM band 1, it is likely that the aerosol contribution is of the order of 50% even for relatively clear sky conditions. Although qualitative evaluation of these remotely sensed data have been very useful, the developments of the quantitative linkages between the satellite imagery and the surface characteristics greatly depend on the removal of the atmospheric effects. The objective of the so-called **atmospheric correction** is to retrieve the surface reflectance from remotely sensed imagery. It has been demonstrated [4, 5] that the atmospheric correction can significantly improve the accuracy of image classification.

Atmospheric correction algorithms basically consist of two major steps. First, the optical characteristics of the atmosphere are estimated either by using special features of the ground surface or by direct measurements of the atmospheric constituents [10] or by using theoretical models. Various quantities related to the atmospheric correction can then be computed by the radiative transfer algorithms given the atmospheric optical properties. Second, the remotely sensed imagery can be corrected by inversion procedures that derive the surface reflectance, as we will shortly discussed in more details.

In this paper, we will focus on the second step, describing our work on improving the computational efficiency of the existing atmospheric correction algorithms. In Section 2, we present background material, while a known atmospheric correction algorithm is described in Section 3. We describe in Section 4 a substantially more efficient version of the algorithm. In Section 5, we develop a parallel implementation of our algorithm, that is scalable in terms of number of processors, internal memory size, and number of I/O nodes. A case study is presented in Section 6 and concluding remarks are given in Section 7.

# 2 Background

## 2.1 Atmospheric Correction

In order to correct for the atmospheric effects, the relationship between the upward radiance $L^m$ measured by the satellite and the surface reflectance $\rho$ has to be established. The radiative transfer theory is used for this purpose. 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 [6] :

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

where $L_0$ is the upward radiance of the atmosphere with zero surface reflectance, 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 *atmospheric albedo* (the probability that a photon reflected from the surface is reflected back to the surface). From Eqn. (1) we can see that the factor $\frac{\rho}{1-s\rho}$ is the sum of the infinite series of interactions between the surface and the atmosphere $\sum_n (s\rho)^n$.

In order to invert $\rho$ 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. The locational information includes, surface level and observer heights, observation and solar zenith angles, and observation azimuth angle (Figure 1). 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 $\rho$.

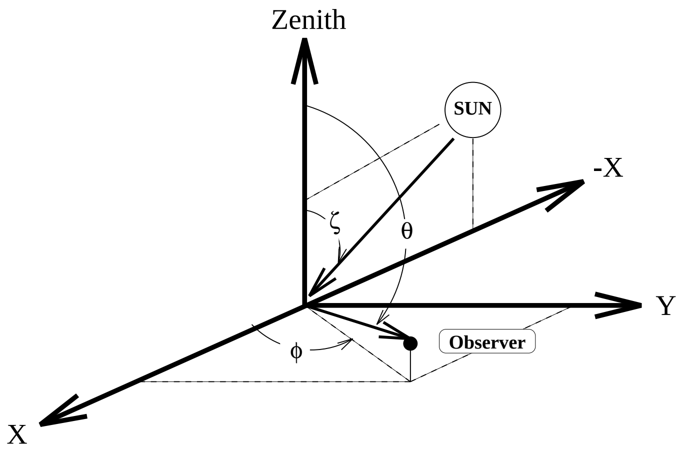



Figure 1: Angular coordinates used in the algorithm. The X-Y plane is a horizontal plane tangent to the earth's surface at the observation point. The solar zenith angle $\zeta$, observation zenith angle $\theta$, and observation azimuth angle $\phi$ are shown.

It is not easy to obtain simultaneous measurements of all atmospheric optical properties operationally because of the rapid variation of the atmosphere. The estimation of the atmospheric optical properties from imagery itself is the only way for operational atmospheric correction. One of the main parameters needed for TM and AVHRR imagery is the *aerosol optical depth*, which is defined as $\tau = -\ln T$. The so-called *"dark object"* [7] approach is used for this study. The idea behind this approach is quite simple. We search for pixels with low surface reflectance using TM band 7 [8] (or AVHRR band 3 [9, 10]) in which aerosol effect is very small, 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.

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 [6], which is an integro-differential equation from which no analytical solution is available. There are a couple of approaches for obtaining practical solutions. The first is to use a numerical iterative approach, such as the discrete-ordinate algorithm [11], and the Gauss-Seidel algorithm [12]. 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 [13], and the four-stream approximation [14]. These approximation algorithms are computationally efficient, but the accuracy is limited. An alternative is to set up off-line look-up tables [15] 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 approach has been followed in this study and is referred to as the *look-up table approach*.

## 2.2 Why High Performance Computing (HPC)?

The existing code [15] for atmospheric correction based on the look-up table approach processes one pixel at a time (and meant to be used interactively), and takes 2.63 seconds to correct a single pixel on a SPARC-10 machine, excluding I/O. A single TM image that covers an area of size $180Km \times 180Km$ consists of approximately 36 million pixels per band, and will therefore require more than 15 years to correct with the existing code, excluding the I/O time. We address this apparent intractability by attempting to achieve the following two objectives:

- Minimize the overall computational complexity while maintaining the accuracy of the algorithm.
- Maximize the scalability of the computation and the I/O as a function of the available resources (computation nodes, I/O nodes, and size of internal memory).

We believe that our algorithm, to be described in the rest of this paper, achieves both objectives simultaneously. In fact, we are able to correct a TM image in less than 13 minutes on SPARC-10 (excluding I/O) and in less than 4.3 minutes on a 32-processor CM-5 (including I/O).

## 2.3 Data Sources

In our studies, TM and AVHRR are used as the primary sources of input data to our algorithm. The essential features of these imagery types are as follows.

The TM imagery consists of 7 channels that correspond to 7 spectral bands. The resolution is $30m$ and five bands need to be corrected (band 6 corresponds to the thermal channel and the atmosphere does not have much scattering effect on band 7) [16]. An image covers approximately an area of size $180Km \times 180Km$ and requires 36MB per channel.

The Pathfinder AVHRR imagery consists of 12 channels from which only 5 channels are original band information and only 2 bands need to be corrected [17]. The remaining channels provide location, quality , cloud, and time information. Each band requires about $20MB$ and the resolution is 8Km. Each AVHRR image covers the entire globe.

# 3 Atmospheric Correction Algorithm

## 3.1 Description of Algorithm

A direct implementation of an atmospheric correction algorithm based on the look-up table approach has been described in [15], where a Fortran version is given. The overall algorithm can be sketched as follows.

**Algorithm (Atmospheric Correction)**

**Input:** *$N$ pixel values with the following information for each pixel: measurement wavelength, atmospheric correction parameters, location information and the day of measurement. Moreover, look-up tables for solar flux and for the functions $L_0$, $F_d$, $T$ and $s$ are provided.*

**Output:** *The surface reflectance at each pixel.*

**for** each pixel **do:**

**Step A:** *Read input parameters and appropriate look-up tables.*

**Step B:** *Perform initializations and required normalizations.*

**Step C:** *Compute $L_0$, $F_d$, $T$ and $s$ by interpolation, using the look-up tables.*

**Step D:** *Compute the surface reflectance $\rho$ using Eqn. (1).*

**end**

In step $B$, we compute the solar flux for the measured wavelength by linear interpolation (using the spectral irradiances and earth-sun distance for given wavelengths). Then we correct the solar flux for the day of the year for which the measurement is made and we use the result to convert the measured radiance $L^m$ from absolute units to reflectance units. We also compute the default values of the water and gaseous (carbon-dioxide and ozone) absorption by linear interpolation of the measured wavelength.

All the interpolations in step $B$ are *spline interpolations* of degree one. A spline interpolation consists of polynomial pieces on subintervals joined together with certain continuity conditions. Formally, suppose that a table of $n+1$ data points $(x_i, y_i)$ is given, where $0 \leq i \leq n$ and $x_0 < x_1 < \ldots < x_n$ is satisfied. A spline function of degree $k$ on these $n+1$ points is a function, $S$ such that, (1) $S(x_i) = y_i$, (2) on each interval $[x_i, x_{i+1})$, $S$ is a polynomial of degree $\leq k$, and (3) $S$ has a continuous $(k-1)$st derivative on $[x_0, x_n]$. Thus the spline interpolation of degree one is a piecewise linear function. By interpolating at a point $x$, we want to find $y$, the value of function $S$ at point $x$, assuming that $x_0 \leq x \leq x_n$. Hence, for a spline interpolation of degree one, we need to find an index $0 \leq j < n$ such that $x_j \leq x \leq x_{j+1}$ and approximate the value $y$ by $y_j + \frac{y_j - y_{j+1}}{x_j - x_{j+1}}(x - x_j)$.

Step $C$ is computationally intensive and involves both linear and nonlinear interpolations. We give next the details of step $C$ for computing $L_0$, a function of wavelength, location, and atmospheric parameters.

**Algorithm (Interpolate $L_0$)**

**Input:** *Wavelength, atmospheric correction parameters, location information and look-up tables.*

**begin**

1. *Interpolate $L_0$ for the measured height.*
2. *Interpolate $L_0$ for the wavelength and adjust for the excess and deficit of gaseous and water absorption.*
3. *Interpolate $L_0$ on measured solar zenith angle.*
4. *Interpolate $L_0$ on measured observation azimuth angle.*
5. *Interpolate $L_0$ on measured observation zenith angle.*
6. *Interpolate $L_0$ on measured optical thickness.*

**end**

In step 1, linear interpolations and extrapolations are performed. The interpolations in step 2 are piecewise exponential interpolations and those required by steps 3, 4, and 5 are spline interpolations of degree one. For piecewise exponential interpolation at a point $x$, $x_j \leq x < x_{j+1}$, $y = e^{(\alpha \log y_j + (1-\alpha) \log y_{j+1})}$ where, $\alpha = \frac{\log(\frac{x}{x_{j+1}})}{\log(\frac{x_j}{x_{j+1}})}$. The interpolation on measured optical thickness required by Step 6 is non-linear and consists of the following substeps. First we find the minimum of a non-linear function over a subinterval; a costly process given the complexity of the function to be minimized. Second, we perform a linear or a nonlinear interpolation depending on the outcome of the first substep.

## 3.2 Implementation

The direct atmospheric correction algorithm has been coded in Fortran [15] and tested on about 20 pixels. It uses eight look-up tables, one table for each of the 2 AVHRR bands of 0.639 and 0.845 $\mu m$ and one for each of the 6 TM bands of 0.486, 0.587, 0.663, 0.837, 1.663 and 2.189 $\mu m$. Each table contains the values of $L_0$, $F_d$, $T$ and $s$ for nine solar zenith angles (10,20,30,40,50,60,66,72,and 78 degree), 13 observation zenith angles (0 to 78 degree, every 6 degree), 19 observation azimuth angles(0 to 180 degree, every 10 degree, plus 5 and 175 degrees), 4 aerosol optical thicknesses (0.0, 0.25, 0.50, and 1.0), and 3 observation heights (0.45, 4.5, and 80.0 kilometers). One more table is used which gives the solar spectral irradiances for 60 wavelengths in the range 0.486 to 2.189 $\mu m$.

The code corrects single pixels of an image. The wavelength, solar and observation angles, and aerosol optical thickness data are assumed to be in the range discussed above as no extrapolation is performed. If any of the selected parameters does not match any of the values used to construct the look-up tables, the algorithm interpolates on that parameter. For observation height, the algorithm both interpolates and extrapolates, and chooses the one that shows minimal atmospheric effect (lower values of intensity and higher values of transmittance are selected).

The computation for each pixel begins by reading the input data and look-up tables, followed by approximating the values of $L_0$, $F_d$, $T$ and $s$ by interpolations. Finally the surface reflectance of that pixel is computed using Eqn. (1).

The resulting algorithm takes 2.63 seconds to correct a single pixel on a SPARC-10 system. Hence this code is not suitable for handling images as a single TM image that covers an area of size ($180km \times 180km$) will take more than 15 years to correct all 5 bands. In the next section, we present our modified version of the algorithm which runs substantially faster and is quite efficient for handling images.

# 4 Optimizing Overall Computational Complexity

We introduce a number of techniques that lead to a substantially more efficient version of the atmospheric correction algorithm. We group our new techniques into 5 major types described briefly next.

**Reordering and Classifying Operations.** In the previous algorithm, all operations are repeated for each pixel. We rearrange the operations into three groups (1) image based, (2) window based, and (3) pixel based. The first group includes the operations that are independent of pixel values. For example, reading the look-up tables, the initializations and the interpolations based on the height of the sensor are image based operations. The corresponding operations can be performed for one pixel and used for all the remaining pixels. The second class of operations, window based, are reserved for those that depend on parameters which remain fairly constant over a window of size $w \times w$ for a suitable value of $w$. For example, the aerosol optical thickness remains fairly constant over a small neighborhood. Atmospheric conditions and the resolution of the image determine the value of $w$. It follows that the computations and the interpolations belonging to the second group depend only on parameters that can be considered constant over windows. These operations can be performed only once for each window if they are performed before any pixel based computation. The remaining computations and interpolations belong to the third group and depend on pixel values or some other parameters that are different for each pixel. We reorganized the operations of the atmospheric correction algorithm, so that the first group (image based) operations appear first followed by those of the second group (window based), and then by the third group operations. This has resulted in a substantial reduction of the total number of operations used.

**Performing Interpolations on Sub-cubes for Each Group.** In the original algorithm, each interpolation causes one of the dimensions to be removed. Moreover, all the interpolations are piecewise interpolations. To reduce the number of operations, we first identify the appropriate indexes along all the dimensions for which interpolations are required within a certain class (image, window, or pixel). Then we extract the sub-cube, obtained from the intersection of those indexes. With this technique not only is the computational complexity reduced, but also the computational complexity becomes almost independent of the size of the look-up tables.

**Data Dependent Control.** Another technique is to use the characteristics of different data inputs to reduce the overall computational complexity. For high resolution data most of the parameters are constant for the whole image while for coarse data most parameters change from window (pixel) to window (pixel). On the other hand, the computational complexity per pixel is much more important for high resolution data than for coarse data due to the large difference in the amount of data involved.

**Changing Nonlinear Interpolations to Linear Interpolations.** We have replaced some of the nonlinear interpolations by linear interpolations at the expense of increasing the sizes of the look-up tables. This technique decreases the number of computations because those nonlinear interpolations are window based operations and increasing the look-up table size mostly affects image based operations. For example, we replaced the nonlinear interpolations on measured optical thickness, which induce window based operations, with linear interpolations.

**Removing Unnecessary Interpolations.** We replaced the interpolations with simpler operations whenever possible. For example, since we are only dealing with satellite images, we do not need to do the interpolation for the observation height since it can be assumed as constant.

These techniques were quite effective in improving the performance while preserving the quality of the corrected imagery. Table 1 shows the performance of our atmospheric correction algorithm, for different window sizes and for different types of input data. The execution times do not include I/O time and are obtained on a SPARC-10 machine. Clearly, the new code is substantially faster than the code in [15] and can be used to correct all 5 bands of a TM image covering an area of size $180km \times 180km$ with a window size of $19 \times 19$ in less than 13 minutes on a SPARC-10 machine (excluding I/O).

| Data Type | Window Size | Time($\mu$sec/pixel) |
|---|---|---|
| TM | $1 \times 1$ | 22.95 |
| TM | $3 \times 3$ | 7.00 |
| TM | $5 \times 5$ | 4.08 |
| TM | $19 \times 19$ | 4.01 |
| TM | $99 \times 99$ | 3.94 |
| AVHRR | $1 \times 1$ | 527.50 |
| AVHRR | $3 \times 3$ | 75.35 |
| AVHRR | $5 \times 5$ | 35.76 |
| AVHRR | $19 \times 19$ | 15.25 |

Table 1: Performance on a SPARC-10 machine.

It is interesting to notice that substantial speedups are achieved by increasing the window size up to a certain point. After that point ($5 \times 5$ for TM), the speedups start to level off. Also, the TM data is corrected much faster than AVHRR data by our algorithm for the following reasons. First, most of the computations for TM are image based operations while in AVHRR they are mostly window based operations. Second, we can skip some of the interpolations for TM data, such as those for observation angles.

It should be mentioned that the input for our code is raw data and does not need any preprocessing, while the input for the code [15] is formatted and needs another program to transform the raw data into the special formatted input.

# 5 Parallel implementation

Atmospheric correction of global data sets requires the extensive handling of large amounts of data residing in external storage, and hence the optimization of the I/O time must be considered in addition to the computation time. 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. We concentrate first on modeling the I/O performance; we then give a description of the parallel algorithm together with its overall theoretical performance followed by experimental results on a 32-node CM-5.

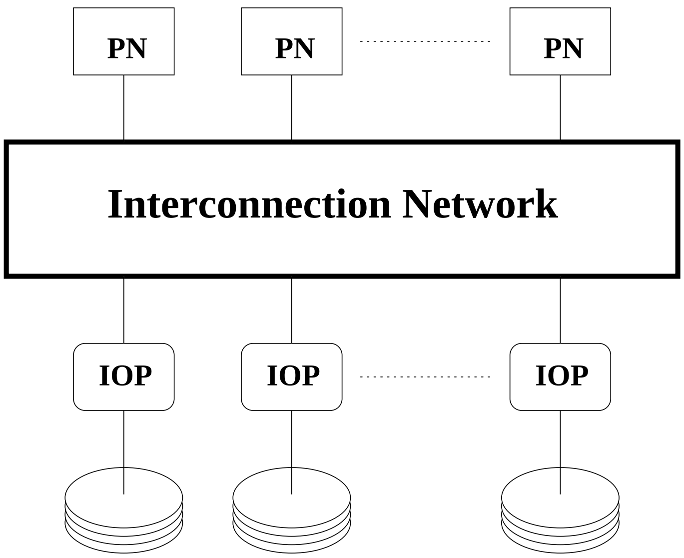



Figure 2: Processing nodes and I/O nodes are connected by an interconnection network.

## 5.1 I/O Model

Our parallel model consists of a number $p$ of computation nodes $P_0$, $P_1$, …, $P_{p-1}$ and a number $d$ of I/O nodes, $N_0$, $N_1$, …, $N_{d-1}$, connected by an interconnection network (Figure 2). For our purposes, we view each I/O node as holding a large disk (or a disk array). Data are transferred into and out of external storage in units of *blocks*, each block consisting of a number $b$ of contiguous records. Each of the $d$

disks can simultaneously transfer a block into the network. Therefore the theoretical total I/O bandwidth is $d$ times the bandwidth provided by a single I/O node. The actual bandwidth depends on several factors including the interconnection network, the ratio of $p$ and $d$, and the network I/O interface.

Most current parallel systems use the technique of *disk striping* in which consecutive blocks are stored on different disks. For our case, an $n \times n$ image (say, in row-major order form) will be striped across the $d$ disks in block units. When the image is accessed in parallel, we can adopt the single disk model with a very large bandwidth. In the single disk model, the time to transfer data is the sum of two components $t_a + \frac{N}{d}t_e$, where $t_a$ is the *access setup time*, $N$ the data size, and $t_e$ the transfer time per element. Thus the I/O time depends essentially on the two parameters, $t_a$ and $t_e$. On a serial machine, the access setup time is mostly the *seek time* $t_s$ which is around $20msec$ but on parallel machines it is considerably larger, whereas the transfer time per element is substantially smaller (inverse of total bandwidth). In spite of the fact that $t_a$ and $t_e$ vary from one application to another, they can be approximated reasonably well by constants.

The I/O performance can be estimated by using the number $n_p$ of *passes through the data* and the number $n_a$ of *disk accesses*. In this case, the total data transfer time is given by $(n_a \times t_a) + (n_p \times N \times t_e)$. Since the access setup time is much larger than the transfer time, minimizing the number of disk accesses is usually much more important than minimizing the number of passes. For our implementation of the atmospheric correction algorithm to be discussed shortly, it is possible to minimize the I/O time by minimizing both parameters independently. For other problems (e.g., matrix transposition), it is possible to come up with algorithms that use more passes but requires less total I/O time by reducing the number of disk accesses [18]. A brief description of the CM-5 I/O system and its relationship to our model is presented in Appendix A.

## 5.2 Parallel Algorithm

We now sketch our parallel atmospheric correction algorithm and how it achieves its computation and I/O scalability. The algorithm is designed in the *Single Program Multiple Data* (SPMD) model using the *multiblock PARTI library*. Therefore each processor runs the same code but on different parts of the image. The *Multiblock PARTI library* [19, 20] is a runtime support library for parallelizing application codes efficiently which provides a *shared memory* view on distributed memory machines. This makes our code portable to a large set of parallel machines since the library is available on the CM-5, the Intel iPSC/860 and Paragon, the IBM SP-1 and the PVM message passing environment for networks of workstations.

A straightforward parallel implementation of the algorithm applied to each band of a $N \times N$ image[4] requires only one pass through the image and can be done as follows. We read a block of maximum possible size, (say $n \times n$) of the image that can fit in the internal memory with corresponding parameters for that part of the

[4] Here we have assumed that the numbers of columns and rows are equal for simplicity. The same type of analysis can be carried out in the more general case.

image. We apply our atmospheric correction procedure on the current block and write back the results. The procedure is repeated until the entire image is processed. This method requires $n$ disk accesses per iteration, and hence $Cn\frac{N^2}{MP} \approx C\frac{N^2}{\sqrt{MP}}$ disk accesses per band are required, where $M$ is the memory size per processor and C is some constant. However we can modify the algorithm to minimize the number of disk accesses as follows. During each iteration, instead of reading a block, we read a slab consisting of the maximum possible number (say $r$) of consecutive rows of the image that can fit in the internal memory with corresponding parameters. We apply the atmospheric correction procedure on the slab and write back the results and repeat the procedure until the entire image is processed. Now we need only one disk access per iteration and hence the total number of disk accesses is reduced to $C\frac{N^2}{MP}$ which is clearly optimal. The new algorithm still requires only one pass through the image and therefore the total transfer time is also optimal and is given by

$$T_{I/O} \approx bN^2\{C_1\frac{t_a}{Mp} + C_2\frac{t_e}{d}\}$$

where $b$ is the number of bands that we correct ($b = 5$ for TM and $b = 2$ for AVHRR), $C_1$ is the cost of constant number of disk accesses per pass, and $C_2$ is the number of bytes per pixel that we read or write. Given these relations, the I/O time can be controlled by changing the number of processors and the number of disk I/O nodes.

In our algorithm, interprocessor communication is only introduced by the possibility of partitioning windows across processors. This can be eliminated by replacing $r$ with $r' = \lfloor\frac{r}{w.p}\rfloor \times w \times p$. This can be done only if at least $w$ rows can fit in the internal memory of each processor, which is a realistic assumption for all reasonable values of w (i.e. $\leq 500$).

The computation time for both TM and AVHRR can be estimated by

$$T_{comp} \approx b\{C_3 + C_4\frac{N^2}{pw^2} + C_5\frac{N^2}{p}\},$$

where $C_3$, $C_4$ and $C_5$ are some machine dependent constants. In fact $C_3$ is the required time for image based computations, $C_4$ is the required time per window for window based operations and $C_5$ is the required time per pixel for pixel based operations. These constants can be accurately estimated for a given machine. As an example, for TM data on CM-5, $C_3 \approx 26sec$, $C_4 \approx 62\mu sec$, and $C_5 \approx 18\mu sec$. Note that $C_3$ includes the I/O time for reading all the look-up tables. These numbers are valid for large data and agree with the observed experimental results.

Summarizing , the total time is given by

$$T_{total} \approx b\{C_3 + C_4\frac{N^2}{w^2p} + C_5\frac{N^2}{p} + C_1\frac{N^2}{Mp}t_a + C_2\frac{N^2}{d}t_e\}$$

The above performance analysis indicate that our algorithm is scalable in terms of the parameters $p$, $M$, and $d$ because for each term with $N$ in the numerator we have $p$, $M$, or $d$ in the denominator. Also, for a desired value of $T_{total}$, we can derive the number of processors and the number $d$ of disk I/O nodes to achieve this.

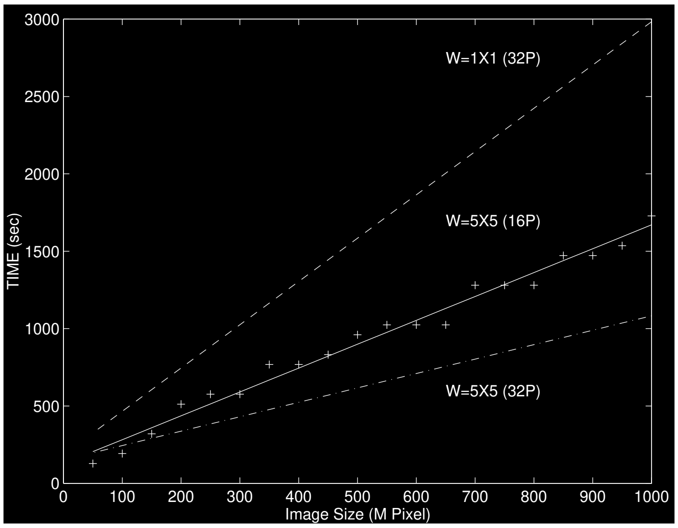



Figure 3: Atmospheric correction performance of TM imagery on CM-5, for different image sizes, two window sizes, and a different number of processors.

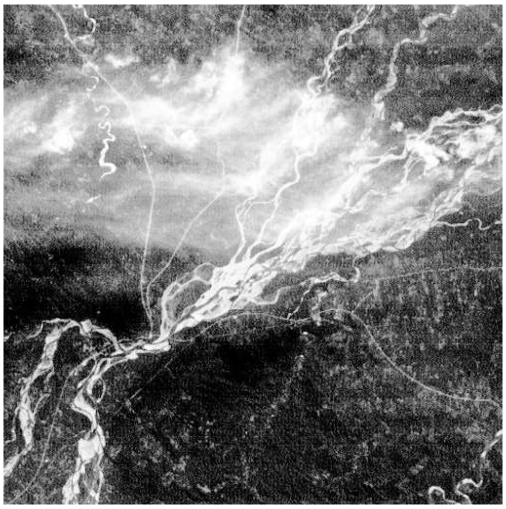
The results of running our code on a 32-node CM-5 are illustrated in Figure 3. The figure shows the total time in seconds for different image sizes, two window sizes and a different number of processors. We have used the least squares approximation to smooth the curves, but we have also included the real data for one of the cases ($w = 5 \times 5$ on 16 processors) to show the linearity of the experimental data. These experimental results are consistent with the analysis carried out in our model. From the graph, the running time scales properly with the image size and with the number of processors for different window sizes.

# 6 Case Study

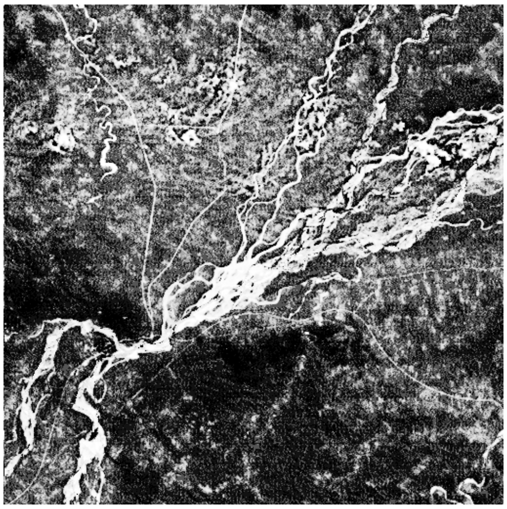
The application of our algorithm on a real TM imagery is presented in this section. Figure 4 shows a subset of TM image bands 1, 3, and 7 acquired on *Aug. 17, 1989* in *Amazon Basin area.* As we can see, a large portion of the image in bands 1 and 3 in the middle is occupied by hazy aerosols and thin clouds but band 7 is less contaminated because it is in large wavelengths and scattering effect is negligible. In order to remove the aerosol contamination, the first step is to implement the so-called "dark-object" approach to estimate aerosol optical depth using band 7. We developed a preliminary program which estimates the aerosol optical depth for TM images and applied it to the mentioned image [5].

After having obtained aerosol optical depth for each channel, the surface reflectance is retrieved using the new atmospheric correction algorithm described in Section 5. The retrieved reflectance is shown in Figure 4. It is evident that most of hazy aerosols in channels 1 and 3 have been removed. Also it is interesting to see that the corrected channel 3 looks even more clear than channel 7. The retrieved

[5] Details of this algorithm will be published after validation.

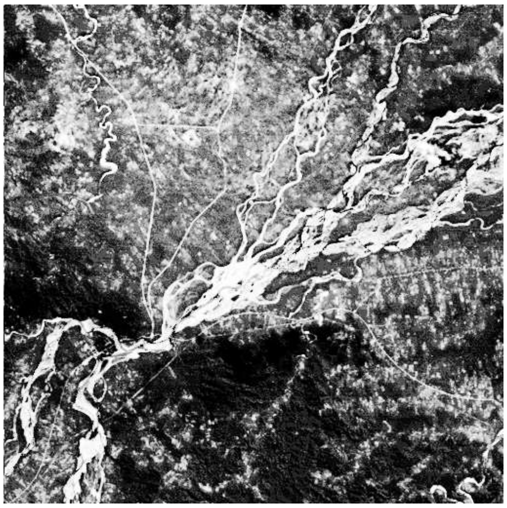


Band 1, **Before** Correction

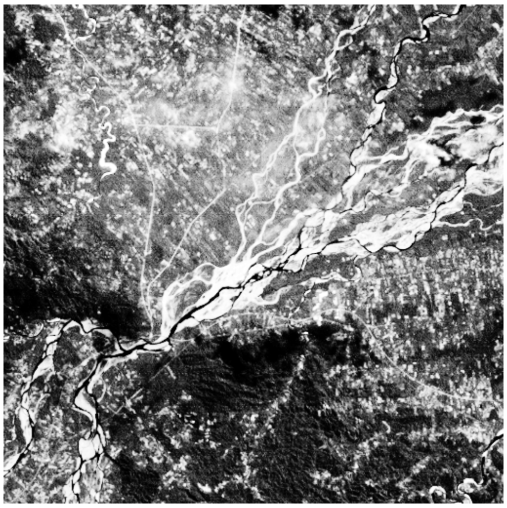


Band 1, **After** Correction

Band 3, **Before** Correction



Band 3, **After** Correction



Band 7

Figure 4: TM imagery ($512 \times 512$).

surface reflectance underneath the hazy aerosols will be further evaluated with the knowledge of the ground truth.

# 7 Conclusion

We have introduced a number of techniques to obtain a very efficient atmospheric correction algorithm based on the look-up table approach. As a result our algorithm can correct all 5 bands of a TM image, covering an area of size $180km \times 180km$, in less than 13 minutes on a SPARC-10 machine (excluding I/O). A parallel version of the algorithm that is scalable in terms of the number of processors, the number of I/O nodes, and the size of internal memory, was also described and analyzed. Experimental results on a 32-node CM-5 machine are provided.

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

We thank Dr. Alan Sussman for helping us in using multiblock PARTI library. Ms. Shana Mattoo provided the original NASA code described in [15] and have answered many of our questions regarding the atmospheric correction algorithm based on look-up tables. Her help is greatly appreciated. Also, thanks to Professor Ralph Dubayah for providing comments regarding this project.

# References

[1] C. O. Justice, J. R. G. Townshend, and V. T. Kalb, *Representation of vegetation by continental data set derived from NOAA-AVHRR data,* International Journal of Remote Sensing, 1991, 12:999-1021.

[2] J. R. G. Townshend, *Global datasets for land applications from the Advanced Very High Resolution Radiometer: an introduction,* International Journal of Remote Sensing, 1994, 15:3319-3332.

[3] Y. J. Kaufman, *The atmospheric effect on remote sensing and its correction,* Chapter 9 in Optical Remote Sensing, technology and application, G. Asrar, Ed., Wiley, 1989.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] B. N. Holben, E. Vermote, Y. J. Kaufman, D. Tanre, V. kalb, *Aerosols retrieval over land from AVHRR data - Application for atmospheric correction,* IEEE Trans. Geosci. Remote Sens., 1992, 30:212-222.

[10] Y. J. Kaufman, A. Gitelson, A. Karnieli, E. Ganor, R. S. Fraser, T. Nakajima, S. Mattoo, and B. N. Holben, *Size distribution and scattering phase function of aerosol particles retrieved from sky brightness measurements,* JGR-Atmospheres, 1994, 99:10341-10356.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] J. R. G. Townshend, J. Cushnie, J. R. Hardy, and A. Wilson, *Thematic Mapper data: Characteristics and use*, 1983.

[17] J. R. G. Townshend, M. E. James, S. Liang, and S. Goward, *A long term data set for global terrestrial observations: Report of the AVHRR Pathfinder Land Science Working Group,* NASA-TM, in press, 1995.

[18] S. D.Kaushik, C.-H. Huang, J. R. Johnson, R. W. Johnson, P. Sadayappan, *Efficient Transposition algorithms for large Matrices,* ACM, 1993.

[19] G. Agrawal, A. Sussman, and J. Saltz, *Compiler and runtime support for structured and block structured applications,* In Proceedings Supercomputing '93, IEEE Computer Society Press, November 1993.

[20] A. Sussman and J. Saltz, *A manual for the multiblock PARTI runtime primitives version 4,* Technical Report CS-TR-3070 and UMIACS-TR-93-36, University of Maryland, May 1993.

[21] *CM-5 I/O System Programming Guide,* Version 7.2, Thinking Machines Corporation, Cambridge,MA, September 1993.

[22] *Connection Machine CM-5 Technical Summary,* Technical report, Thinking Machines Corporation, Cambridge,MA, October 1991.

[23] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong, S. W. Yang, and R. Zak, *The Network Architecture of the Connection Machine CM-5,* Extended Abstract, Thinking Machines Corporation, Cambridge,MA, July 1992.

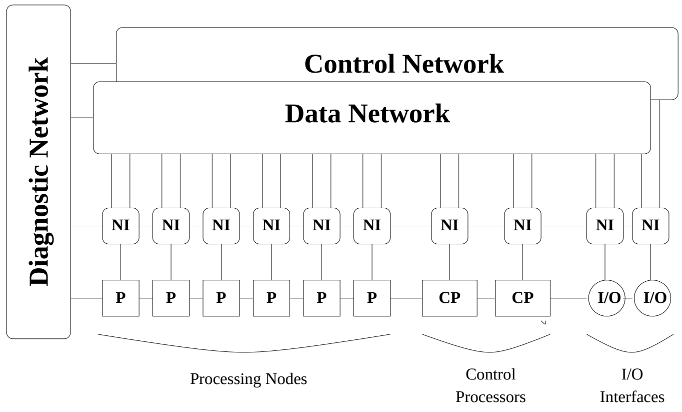



Figure 5: The organization of the Connection Machine CM-5. The machine has three networks: a data network, a control network, and a diagnostic network.The data and control networks are connected to processing nodes, control processors, and I/O channels via a network interface.

# A CM-5 I/O System

Here we examine briefly the CM-5 I/O system and how it relates to our model [21, 22, 23]. The CM-5 contains three communication networks (Figure 5): a data network, a control network and a diagnostic network. The processing nodes (PNs), the control processors and the I/O nodes are interconnected by the three networks. The data network provides high-performance point-to-point data communications between system components whereas the control network provides cooperative and system management operations. The diagnostic network allows back-door access to all system hardware to test system integrity and to detect and isolate errors. The *Scalable Disk Array* (SDA) provides striping across all the disks and communicates with the processors via the data network using high-bandwidth I/O interfaces. A basic read or write operation consists of two phases. A read's two phases are:

- *Raw data transfer*: transferring data between the disks and the PNs' user memory. The transfer rate of this phase is usually dominated by the speed of the SDA unless the number of processing nodes is small compared to the number of disks.

- *Data routing*: Reordering the data in PN memory so that every PN has the correct data in the correct order for the application. The performance of this phase depends on several factors such as (1) the number of processing nodes, (2) the application's geometry, and (3) the number of bytes per shape position or array element.

In general data is stored in serial order on the SDA. When we read a file, data is read as fast as possible during the first phase and sent to the PNs in such a way as to make optimal use of the Data Network. In the second phase, the data is parallelized according to the specified shape and routed to the correct locations. In the write operation we have the two phases in reverse order. It can be shown that the time for the second phase is much smaller than the time for the first phase, and hence our single disk model works well for the CM-5. We measured the $t_a$ and $t_e$ on a 32 node CM-5 with two I/O nodes and the results are shown in table 2.

| OPERATION | SHAPE | $t_a(msec)$ | $t_e(microsec)$ |
|---|---|---|---|
| READ | 1 DIMENSION | 230 | 0.098 |
| READ | 2 DIMENSION | 217 | 0.160 |
| READ | 3 DIMENSION | 235 | 0.166 |
| WRITE | 1 DIMENSION | 700 | 0.212 |
| WRITE | 2 DIMENSION | 755 | 0.210 |
| WRITE | 3 DIMENSION | 757 | 0.214 |

Table 2: $t_a$ and $t_e$ for a 32 node CM-5.