

EVALUATING AND CORRECTING THE EFFECT OF SENSOR SPATIAL RESOLUTION
ON CLOUD FRACTION DERIVED FROM SATELLITE INSTRUMENTS

BY

ALEXANDRA L. JONES

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Atmospheric Sciences
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Associate Professor Larry Di Girolamo

ABSTRACT

This study examines the biases in cloud fraction (CF) derived from satellite instruments that are due to the effect of sensor spatial resolution. In total, 1405 15 m resolution ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer) scenes over cumulus-dominated regions including the Indian Ocean, Caribbean Sea, and Gulf of Mexico were collected for this study. Cloud masks of these scenes were treated as “truth” and used to study the dependence of CF biases on sensor resolution and cloud distributions. We found that at typical meteorological satellite resolution (~ 1 km) the traditional CF estimate derived from perfect clear-conservative masks has a median bias of ~ 0.28 in the above mentioned regions. To obtain a median bias on the order of 0.01, without correction, a cloud mask resolution less than 80 m is required. A simple equation can correct the bias to 0.025 at 150 m resolution and the more computationally expensive pattern recognition technique can correct the median bias to 0.0 at any resolution tested and for any true CF tested. We further applied the above correction techniques to an operational cloud mask, the RCCM (Radiometric Camera-by-camera Cloud Mask) of the MISR (Multi-angle Imaging SpectroRadiometer) instrument, which has spatially and temporally overlapped observations with ASTER and a resolution of 1.1 km. We applied the pattern recognition technique to correct a CF climatology in the tropical Western Atlantic dominated by small trade wind cumulus. The average CF was reduced from 0.496 to 0.199. Although the results were simplified to the average improvement, this technique does more than simply subtract a standard value for bias to improve the climatology; the degree of correction for each region derived from the RCCM is dependent on the spatial distribution of clouds within each region.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Larry Di Girolamo for giving me the space and freedom to choose my own project and make decisions regarding its direction, while providing support and guidance as needed. I would also like to thank my lab-mates, past and present, for providing many valuable lessons and moral support that have contributed to my successes as a graduate student. Support from the National Aeronautic and Space Administration (NASA) under NA06OAR4310003 is gratefully acknowledged. The MISR data and ASTER data were obtained from the Atmospheric Sciences Data Center at the NASA Langley Research Center and the Land Processes Distributed Active Data Center, respectively.

Finally, I would like to thank my friends and family who have provided emotional support throughout my transition into graduate school, as I move forward in my studies, and as I make decisions about the direction of my career. Specifically, I would like to thank Drs. Pamela and Craig Jones, Robin Comeau, Clarice Krueger, Tosha Richardson, and Justin McHugh.

TABLE OF CONTENTS

1. Introduction	1
1.1 Importance of Clouds to Climate	1
1.2 A Brief History of Cloud Fraction Observations	7
1.3 Current Issues in Cloud Fraction Observations	10
1.3.1 Distinguishing Cloud from Aerosol	10
1.3.2 Overestimation of Boundary Layer Cumulus Cloud Fraction	11
1.3.3 Considerations when Using Cloud Fraction Climatologies for Comparison	12
1.4 The Resolution Effect	13
2. Data	16
2.1 Instrument and Data Overview	16
2.2 Cloud Masks	17
2.2.1 ASTER masks	17
2.2.2 MISR masks	18
3. Pattern Recognition Technique	20
4. Resolution Effect Bias of CF from a Perfect, Clear-Conservative Cloud Mask	22
4.1 Dependence of CF biases on resolution	22
4.2 Dependence on cloud area distribution	23
5. Correcting CF from an Operational, Imperfect Clear-Conservative Cloud Mask	28
5.1 Challenges in correcting CF from MISR’s RCCM	28
5.2 Application: correcting CF climatologies	36
6. Summary and Conclusions	38
7. Reference List	40
Appendix A: Research Code	43
A.1 C-Shell Scripts	43
A.2 FORTRAN Source Code	53
Appendix B: Sample Streamer Files	87
Appendix C: ASTER Mask Thresholds	90

1. Introduction

1.1 Importance of Clouds to Climate

Clouds are an important component of the climate system because of their roles in transporting water and modulating heat transfer through latent and radiative processes. Because clouds have a profound influence on water and radiation budgets, small changes in cloud can alter the climate's response to forcings such as increased CO₂ or anthropogenic aerosol (Stephens, 2005). As such, cloud feedbacks remain one of the largest source of uncertainty in climate sensitivity, with boundary layer clouds contributing the most to the range in modeled cloud feedbacks (Bony and Dufresne, 2005; Houghton et al, 2001). They most strongly influence net radiative fluxes, especially through their impact on short wave radiation in the summer (Stephens, 2005). Net radiative flux is a crucial component of the climate system and it is sensitive to changes in the environment, especially changes in clouds (Frouin et al, 1988; Frouin and Chertock, 1992). Increasing the CO₂ mixing ratio by 20% changes the longwave irradiance leaving the surface by 0.5 Wm⁻², while increasing the amount of low clouds by 20% changes the longwave irradiance leaving the surface by 15 Wm⁻² (Frouin et al, 1988). The doubling of CO₂ can be offset by just a 4% increase in low clouds. This sensitivity suggests a maximum cloud fraction (CF) error requirement of just 1% in order to monitor and model the effects of low clouds on climate (Slingo, 1990; Ohring et al, 2005).

Variables like effective radius, optical depth (OD), liquid water path, and CF are used to quantify cloud properties (Goodman and Henderson-Sellers, 1988). Of these variables a perturbation in CF has the largest impact on net fluxes. Figure 1 shows plots of longwave, shortwave, and net radiative flux at the top of atmosphere (TOA) and surface as a function of CF and OD. The radiative transfer model, Streamer (Key, 2001), was used to create these figures for 1.5km thick boundary layer cloud cover in the tropical North Atlantic in December. The effective radius of the cloud drop size distribution is 20μm and the cloud top height was 3km. The liquid water content was allowed to vary with OD. See Appendix B for a sample input and descriptive-output files for one individual point on the figures. At typical trade wind cumulus OD of 20 (Heymsfield and McFarquhar, 2001) the net flux at the top of the atmosphere ranges from ~700 Wm⁻² for the lowest CF to ~250 Wm⁻² for the largest CF. Net fluxes at the surface have a similarly wide range with CF. Because of the sensitivity of climate to cloud, accurate cloud information is needed for both incorporation into models and to compare model output to (Goodman and Henderson-Sellers, 1988). The basic test of modeled clouds is to compare the timing and location of clouds to observations (Stephens, 2005). Models that tune clouds to TOA radiative budgets have some non-unique combination of OD and CF. For example, a cloud with an OD of 20 covering 50% of a grid box produces the same net flux at TOA as a cloud with an OD of 60 covering 35% of a grid box. Since higher OD clouds tend to produce more precipitation, an inaccurate relationship between OD and CF may lead to problems in the model's hydrologic budget. In a model intercomparison study done by Zhang et al (2005) 8 of 10 general circulation models underestimated

tropical low cloud amount compared to satellite observations of CF from ISSCP and CERES (Figure 2). However, the TOA shortwave cloud radiative forcing is within the range of observations for 8 of 10 models indicating a compensatory increase in OD of the simulated clouds. Joint histograms of OD and cloud top height observations from satellite are increasingly being used to evaluate model output (Marchand et al, 2010). However, care must still be taken in recognizing that satellite observations carry both random and systematic errors. This thesis focuses on CF errors caused by the finite resolution of satellite instruments.

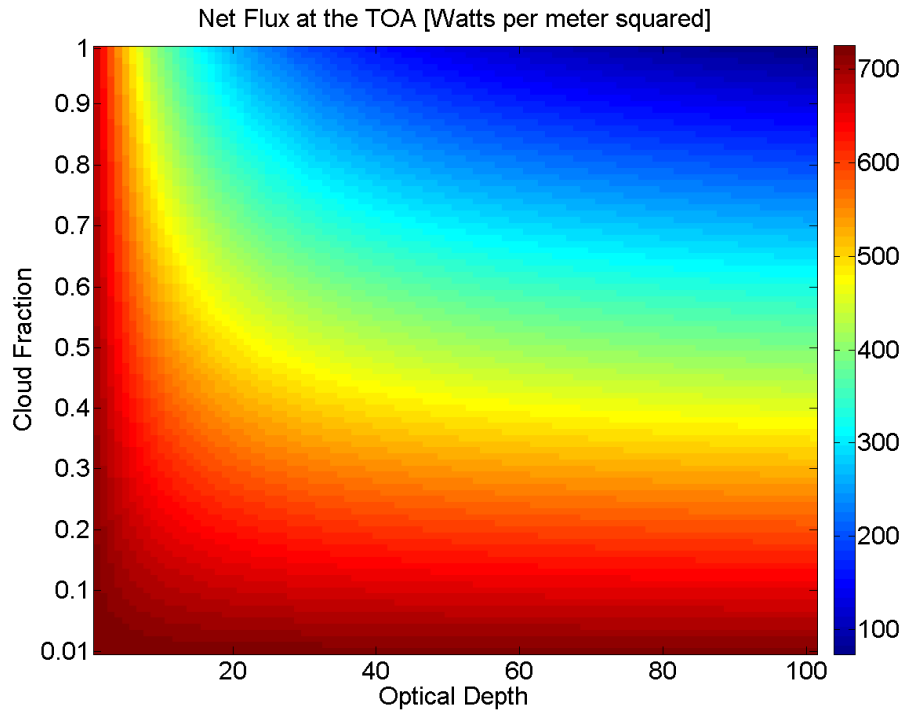


Figure 1. Plots of net radiative flux varying with optical depth and cloud fraction at the surface and top of the atmosphere. The radiative transfer model, Streamer (Key, 2001), was used to create these figures for 1.5km thick boundary layer cloud cover in the tropical North Atlantic in December. The effective radius of the cloud drop size distribution is $20\mu\text{m}$ and the cloud type height is 3km. The liquid water content was allowed to vary with OD.

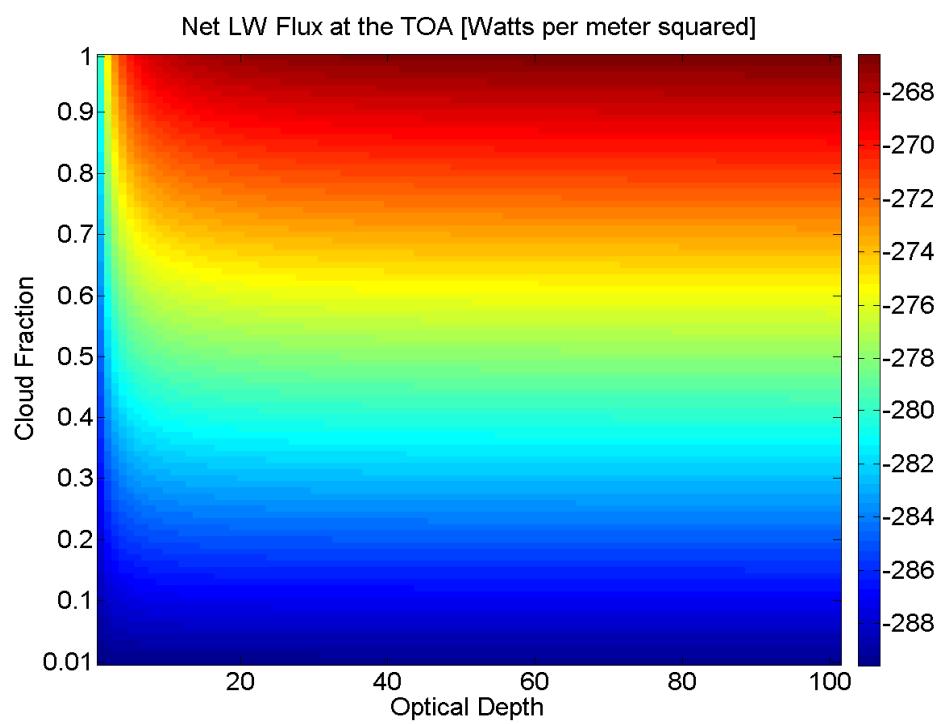
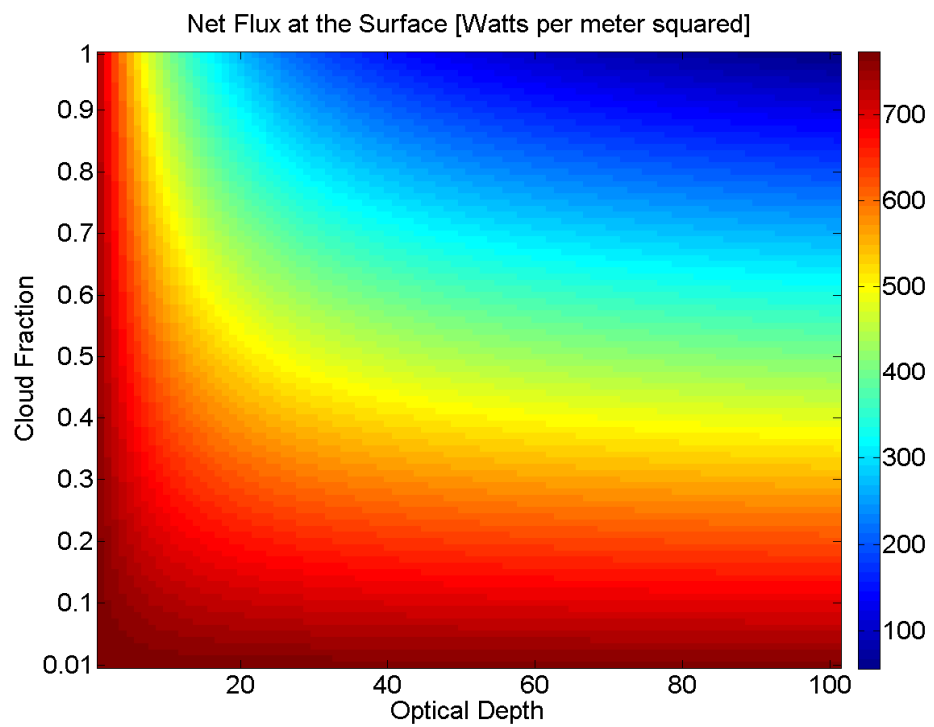


Figure 1. (continued)

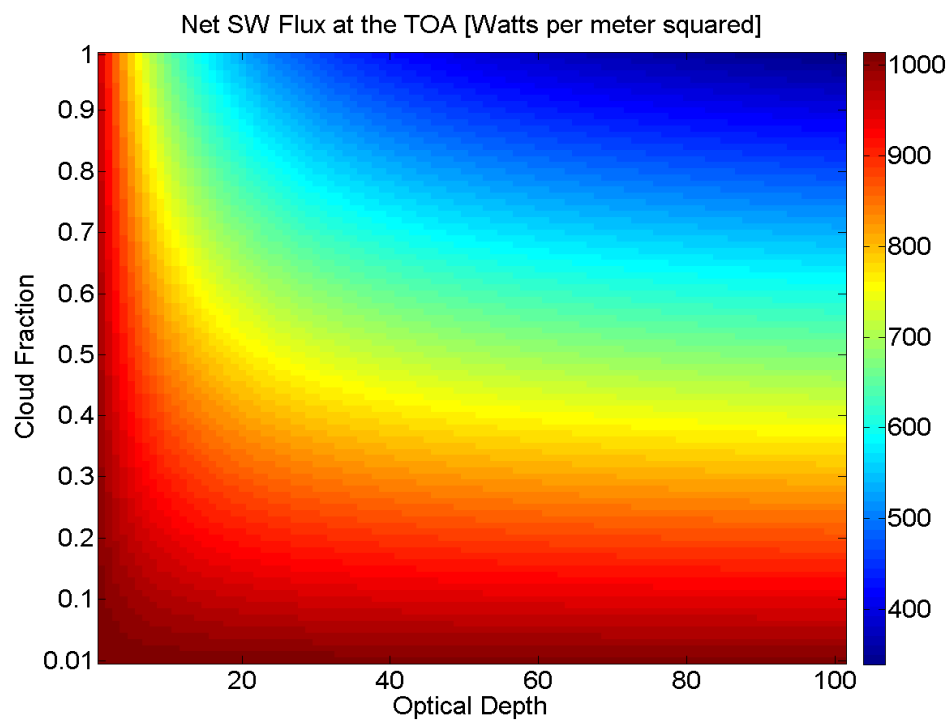
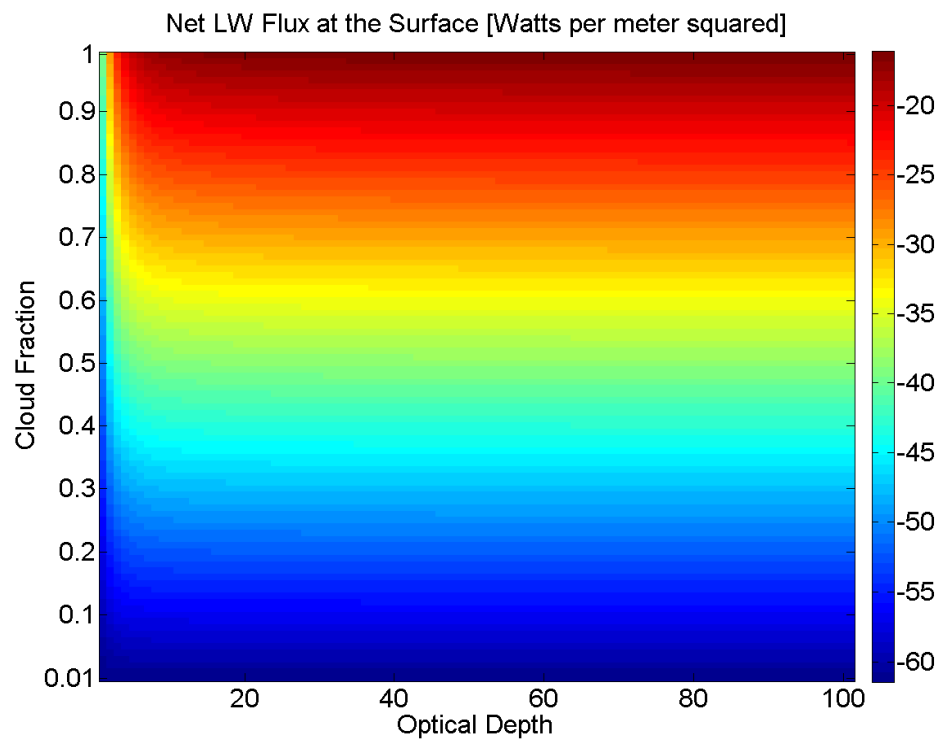


Figure 1. (continued)

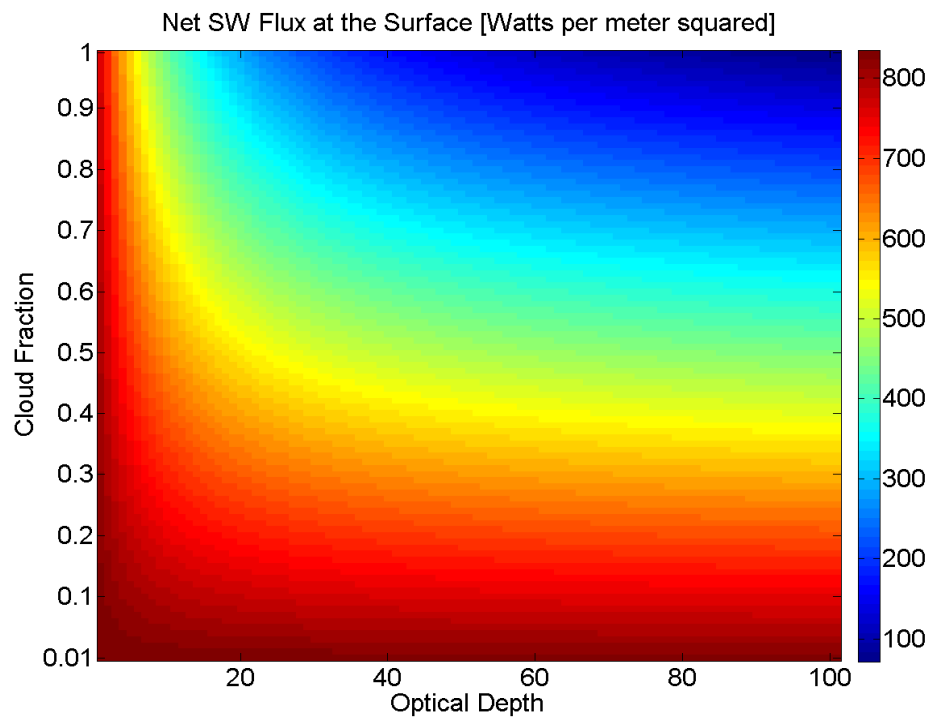


Figure 1. (continued)

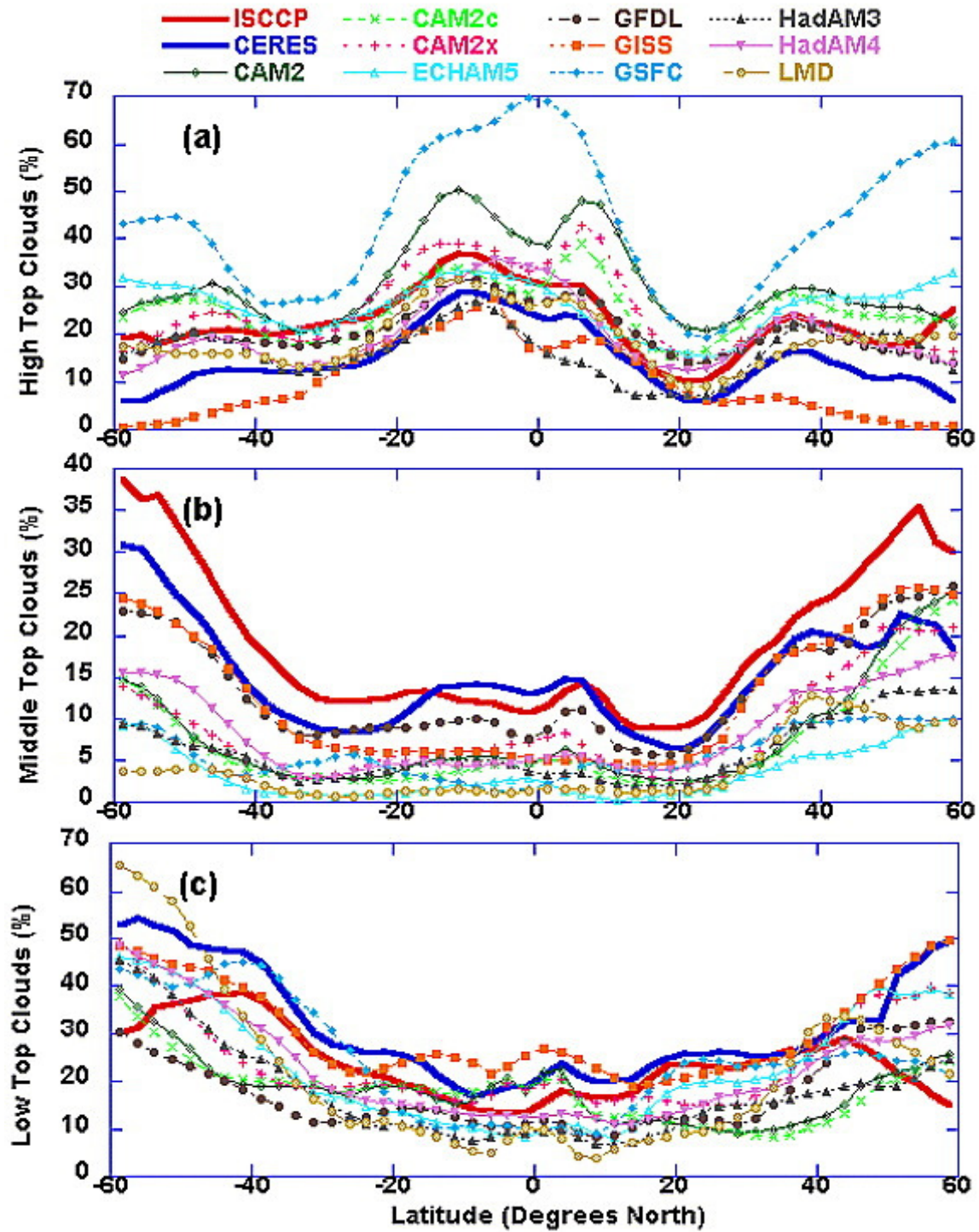


Figure 2. This figure and caption are borrowed with permission from Zhang et al (2005) “(a) High top clouds, (b) middle top clouds, and (c) low top clouds in the DJF from satellite measurements and from the models. ISCCP data are from year 2000, CERES data are from two seasons of 2001 and 2002. Model results are from one year simulations with most of them forced with prescribed monthly sea-surface temperature of year 2000.”

1.2 A Brief History of Cloud Fraction Observations

Cloud amount has historically been collected by surface observers, ground based instrumentation, aircraft mounted instrumentation, and satellite instruments. Estimations of cloud amount from these methods are actually the fraction of background obscured by clouds. The amount of background obscured by a particular cloud is a function of cloud height and area, and view angle (Figure 3). Surface observations are typically more distorted by view angle than satellite observations. However, satellite observations are only the same as “earth view”, which is the CF considered by models, if its view is nadir looking (Henderson-Sellers and McGuffie, 1990). There are benefits and drawbacks to all methods of observing cloud amount. Surface based observations will tend to miss upper level cloud when it is obscured by low level cloud. Conversely, satellites will miss low level cloud when it is obscured by upper level cloud (Goodman and Henderson-Sellers, 1988). Although surface based observations have a long record, they only exist at points on the globe where people or automated stations are available to take them. Cloud cover information over the oceans and in polar regions is especially sparse. Only a satellite can collect timely global observations of CF required to evaluate the impacts of clouds on climate.

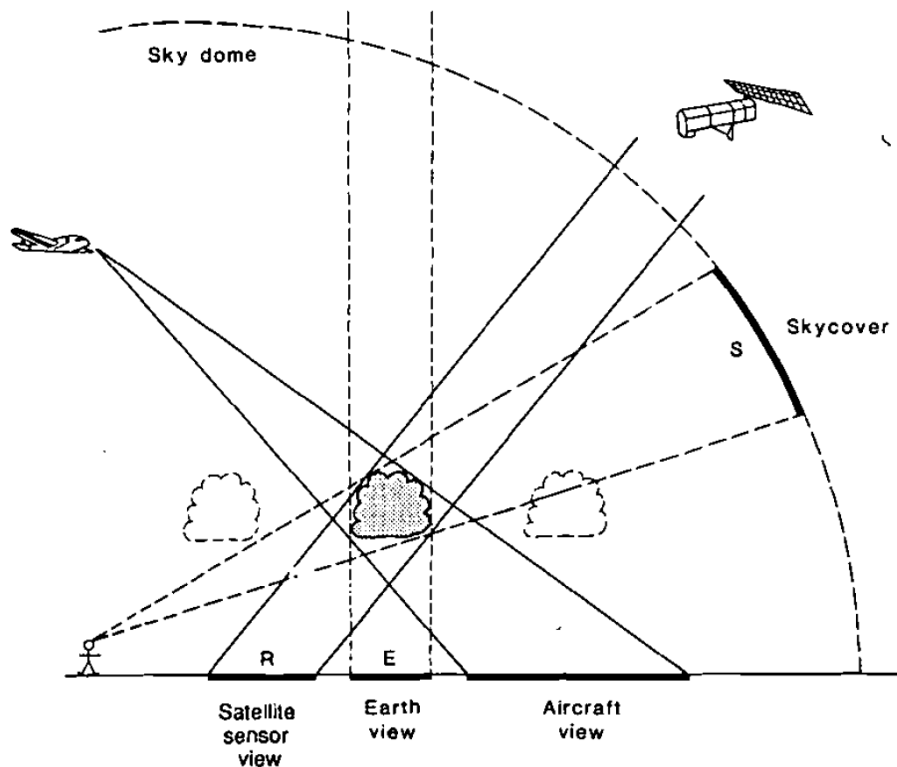


Figure 3. This figure and caption are borrowed with permission from Henderson-Sellers and McGuffie (1990). “Schematic illustration of the relative viewing geometries of the various observing systems under consideration. Observed everywhere perpendicular to the Earth, the earth view (the ideal) of cloud amount is always less than the satellite sensor view, much of which is acquired at very large scan angles. The surface observer has an even greater range of scan angles, and an observer in an aircraft has a yet more complex view of the cloud scene. The sky dome is of relevance only to the surface observer's view.”

Before CF can be determined there must first be proper cloud detection. Defining cloud is one source of uncertainty in every CF product (Di Girolamo and Davies, 1997). There is some minimum radiance that a pixel may contain and still be distinguishable from the clear-sky background. This corresponds to a minimum cloud OD that can be observed from space, which is an instrument and algorithm specific sensitivity. The best threshold value is one that provides the most sensitive and reliable separation of cloudy and clear for the wide range of conditions experienced (Rossow, 1989). Drawbacks to the threshold method relate to partially filled pixels, low OD clouds, and ill-defined background, clear-sky radiance. Figure 4 shows the effect of sensor resolution on measured radiance and cloud detection for a given cloud element. If the threshold is set very low, to capture subpixel clouds, then as pixel size increases the CF will tend toward 1.0 (Astin, 1997). An increase in pixel size can be likened to a decrease in domain size until the domain contains only one pixel. In that case CF will be equal to either 1 or 0 (Dey et al, 2008). So, CF is more sensitive to threshold as pixel size increases (Astin, 1997; Wielicki and Welch, 1986; Wielicki and Parker, 1992). The most popular cloud detection methods implement at least one radiance threshold test for each individual pixel. However, there are other methods used to determine CF (Goodman and Henderson-Sellers, 1988). Statistical cloud detection methods treat large groups of pixels at a time, partitioning multidimensional frequency histograms into representative classes via Gaussian histogram analysis, dynamic clustering, or the spatial coherence method. There are also radiative transfer approaches that determine cloud parameters by fitting radiative transfer model output to the observed radiances.

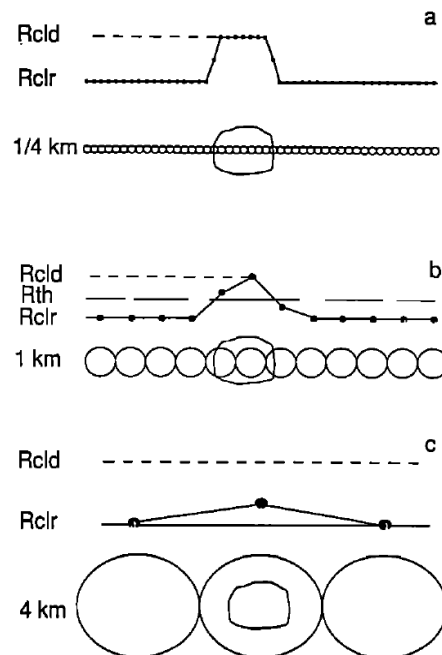


Figure 4. This figure and caption are borrowed with permission from Wielicki and Parker (1992). “Schematic of the effect of sensor spatial resolution on reflectance measured by a satellite. Sensor resolutions of 1/4, 1, and 4 km are shown for a 2-km cloud cell.”

Once cloud detection via thresholding is finished the results are stored in a cloud mask. Operational cloud masks are defined by the location of the threshold on a histogram of radiance values or other observable quantity. A threshold works perfectly when there is a discontinuity in the histogram separating cloudy pixels from clear pixels. This never happens in practice due to thin clouds, partially covered pixels, bright surfaces, instrument noise, and 3D radiative effects. Where to place the threshold depends on the purpose of the cloud mask (Figure 5) (Yang and Di Girolamo, 2008). Increasing the pixel resolution will result in fewer partially filled pixels, which will create less overlap in the radiance histogram. CF is less sensitive to the choice of threshold if the radiance of clear and cloudy pixels is more widely separated. The minimum classification error threshold is placed not in the histogram valley, but at the intersection point of the component Gaussian, or similar, distributions for clear and cloudy pixels. The CF conservative threshold gives the correct CF. The clear conservative threshold gives no misclassified cloudy pixels. The cloud conservative threshold gives no misclassified clear pixels.

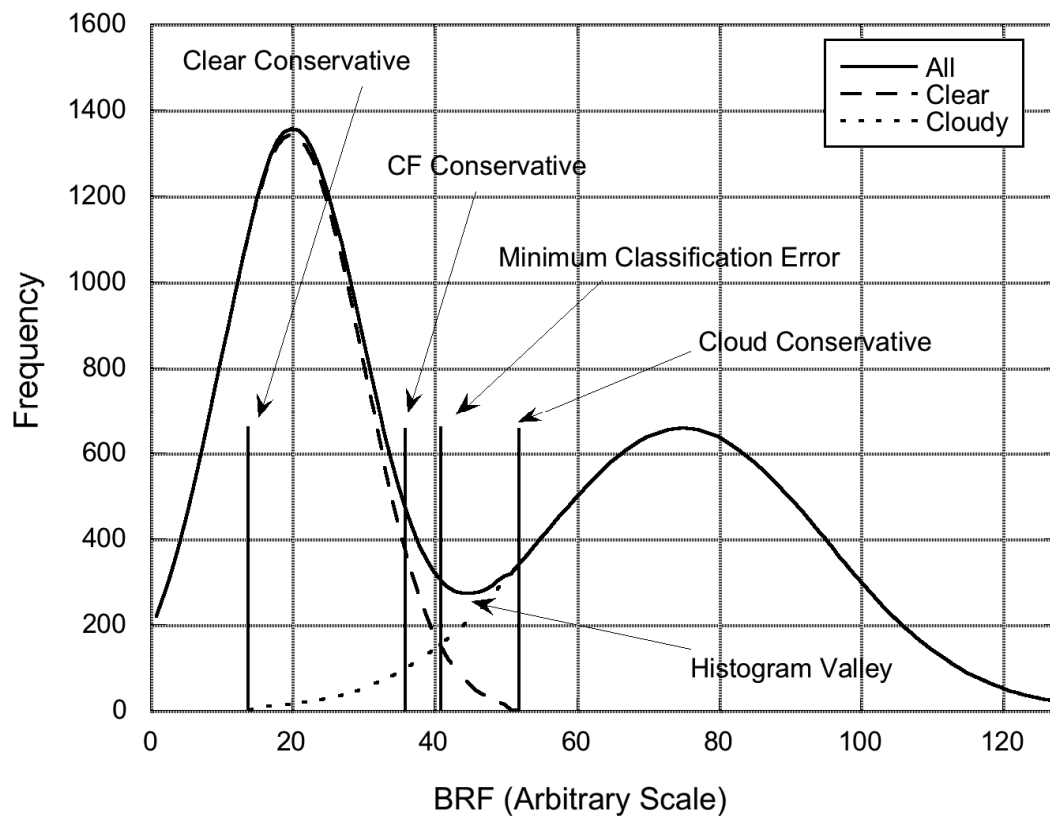


Figure 5. This figure and caption are borrowed with permission from Yang and Di Girolamo (2008). “A conceptual model of a BRF histogram of a satellite image. The distributions of clear and cloudy pixels are assumed to be Gaussian. The three histograms are for clear pixels, cloudy pixels, and all pixels. Four types of thresholds are shown in the figure: the minimum classification error (MCE) threshold, the cloud fraction (CF) conservative threshold, the clear conservative threshold, and the cloud conservative threshold.”

The traditional method of determining CF from a cloud mask is referred to as the “standard method”, where CF is equal to the number of cloudy pixels divided by the total number of pixels. Therefore,

the quality of a cloud mask determines the errors in CF. However, even if a cloud mask successfully flags every pixel that contains any amount of cloud as cloudy (such a mask is referred to as a perfect, clear-conservative cloud mask in this thesis), CF may still be biased relative to the true underlying CF because of the finite resolution of satellite observations. Imager resolution becomes an issue when the pixels contain subpixel-sized clouds or cloud edges transect the pixels. These situations create pixels that are only partially cloud covered. If the cloud detection algorithm considers those pixels cloudy, they will contribute to an overestimation of CF because subpixel clouds and cloud edges contribute to the CF in the amount of the pixel's area rather than the true cloud area. This effect is referred to as the resolution effect. The logical step would be to increase the resolution of the cloud mask and therefore the satellite instrument, but there are limitations on the physical instrument as well as data bandwidth and storage (Goodman and Henderson-Sellers, 1988). Previous studies have suggested ideal pixel resolutions for determining properties of fair weather cumulus. Wielicki and Welch (1986) tested the effect of threshold choice and resolution on CF and found that radiance data of cumulus cloud fields that were degraded to 250m and then thresholded reasonably estimated the true CF for thresholds ranging from 4 to 40 digital counts, representing a variety of cloud mask types. Shenk and Salomonson (1972) suggested that a pixel size $1/100$ of the size of the area averaged cloud diameter is necessary to accurately determine the CF derived from a perfect, clear conservative cloud mask. However, if the partially cloudy pixels are assumed to be 50% cloud covered a pixel $1/10$ the size of the area averaged cloud diameter is necessary to accurately determine the CF. For clouds with area-averaged diameters less than 1km this corresponds to a pixel resolution ranging from 10m to 100m.

1.3 Current Issues in Cloud Fraction Observations

1.3.1 Distinguishing Cloud from Aerosol

There is a continuum of radiance between heavy aerosol and thin cloud. Cloud contamination of aerosol products can arise due to the type of cloud mask and therefore the choice of threshold. The objective of the CF-conservative cloud mask is to find a threshold that balances the overestimation of the CF of small clouds with the underestimation of the CF from optically thin clouds. This means that some of the pixels marked as clear will actually contain some cloud and some of the pixels marked as cloudy may be clear. This can cause bias in aerosol products or any product that relies on the cloud mask to determine which pixels to perform clear air calculations on. Even a small amount of cloud contamination can bias those products, which can be especially important when they are used in studies of aerosol direct and indirect effects (Zhao et al, 2009). One benefit of a clear-conservative cloud mask is that clouds do not contaminate clear-sky products, such as aerosol OD. The Multi-angle Imagine Spectro-Radiometer (MISR) Radiometric Camera-by-camera Cloud Mask (RCCM) is considered a clear-conservative mask and is used in part to determine where aerosol measurements should be calculated. The level of direct cloud contamination by subpixel cumulus

clouds was found to bias aerosol optical depth climatologies by no more than 0.002, which is below the level of uncertainty in the product from other sources (Zhao et al, 2009).

1.3.2 Overestimation of Boundary Layer Cumulus Cloud Fraction

Current cloud mask products derived from meteorological satellites are reported at resolutions of about 1km at best. However, many trade wind cumulus clouds tend to be much smaller than 1km (Zhao and Di Girolamo, 2007; Wielicki and Welch, 1986). Zhao and Di Girolamo (2007) found that cumulus clouds with diameters less than 2km contribute half of the total cumulus CF. Of the cumulus clouds sampled 68% covered an area less than 2.7 km². If the clouds were circular that would be an effective diameter of 58.6 m. Subpixel scale clouds cause competing effects on the CF depending on the mask type; either they go undetected and cause an underestimate in CF or the entire area of the pixel that contains the cloud is considered cloudy and causes an overestimate in the CF. The RCCM overestimates the CF of cumulus clouds by 0.36 on average (Zhao and Di Girolamo, 2006). The significance of this overestimation is great when considering the sensitivity of radiative flux to cloud amount (Figure 1). There is a variation in the over estimation that results from the natural variability of the underlying spatial distribution of the cloud area. Scenes with few large clouds will have a lower overestimation than scenes with many small clouds, even if they have the same high resolution CF (Figure 6). Part of the reason for this is the large relative fraction of partially filled, cloud edge pixels to total cloudy pixels for small cumulus clouds (Wielicki and Parker, 1992). This means that small cumulus CF has the largest dependence on spatial resolution.

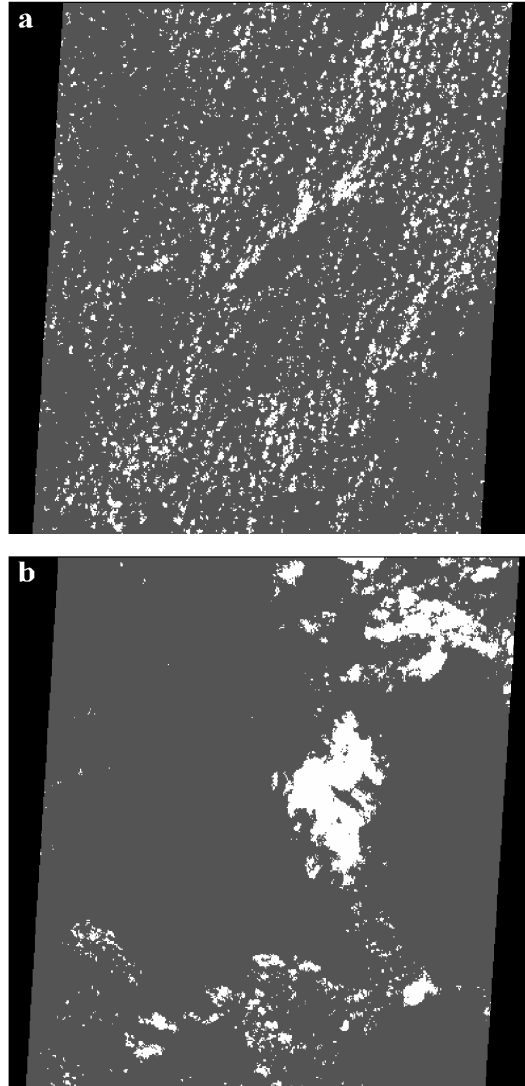


Figure 6. This figure and caption are borrowed with permission from Zhao and Di Girolamo (2006) “(a) A cloud mask for an ASTER scene, taken on Sep. 22, 2004, centered on 14.45°N, 53.80°W and (b) a cloud mask for an ASTER scene, taken on Dec. 9, 2004, centered on 19.25°N, 55.71°W. White represents cloud, grey represents clear and black represents no retrieval.” The 15 m CF are a) 0.09 and b) 0.08, whereas the 1.1km CF are a) 0.83 and b) 0.32.

1.3.3 Considerations when Using Cloud Fraction Climatologies for Comparison

Cloud system specific parameterizations, such as trade wind cumulus, require evaluation of models against observations (Randall et al, 2003). The lack of consistency between products derived from different instruments, platforms and channels is confusing and has sparked a push toward integration or assimilation of data products (Stephens and Vane, 2007). Differences between similar products derived from different observations may not be due entirely to accuracy issues. Quantitative cloud statistics are a function of both domain size and resolution (Dey et al, 2008). So, care must be taken when comparing satellite datasets of differing resolutions. The overestimations made by datasets such as the RCCM need to be taken into consideration when using daily or monthly mean CF products to evaluate model output (Zhao and Di

Girolamo, 2006). Also, when comparing satellite observations to model output it would be improper to assume the CF derived from finite resolution satellite observations is the same as the true fractional area covered by clouds. Marchand et al (2010) suggest that rather than requiring model output to be at the same resolution of satellite observations for direct comparison, more sophisticated algorithms can be developed to estimate the true CF from satellite observations.

1.4 The Resolution Effect

For this study the “resolution effect” will be defined as the component of the partially filled pixel problem that leads to the overestimation of boundary layer CF. The overestimation of CF due to the resolution effect arises from subpixel clouds contributing to the CF in the amount of the pixel’s area rather than the true cloud area. Previous studies have found that, for CF of boundary layer cumulus derived from a cloud mask that is clear-conservative, there is a predictable overestimation that varies with true CF due to the resolution effect (Zhao and Di Girolamo, 2006; and Di Girolamo and Davies, 1997). The degree of overestimation depends on the ratio of cloud size to pixel size (Shenk and Salomonson, 1972) and the distribution of true cloud area (Di Girolamo and Davies, 1997); both are functions of cloud type (Wielicki and Parker, 1992) with cumulus CF showing the greatest dependence on resolution (Wielicki and Welch, 1986).

The resolution effect was first studied by Shenk and Salomonson (1972) in the “paper cloud experiment”, in which paper clouds were laid against a dark background and grids of varying sizes were overlaid to represent pixels of various sizes. An observer then counted the pixels containing some cloud. They found that CF can be overestimated by as much as 86% for small ratios of cloud size to pixel size. A few methods have been devised to deal with the problem of partially filled pixels. Some attempt to improve CF estimates using the radiance of fully clear and fully cloudy pixels to gauge the fractional coverage of nearby pixels that have radiance values falling somewhere in between, such as those developed by Coakley and Betherton (1982) and Arking and Childs (1985). However, these methods assume distinct cloud layers. And, each must have a large patch of fully cloud covered pixels relative to the resolution of the imager. Completely clear pixels must also be present, and each surface type and cloud layer are assumed to have homogeneous properties. Another method suggests that a threshold exists at each resolution that will produce an unbiased estimate of CF (Wielicki and Welch, 1986; Wielicki and Parker, 1992). However, this implies that sometimes a pixel containing some cloud will go undetected to balance out the overestimation from partially cloudy pixels that are detected. Operational algorithms employing this CF-conservative threshold still overestimate CF for scenes composed of small cumulus clouds over high contrast backgrounds by an average of 0.18 (Zhao and Di Girolamo, 2006). And, the errors in CF derived from a cloud mask employing a CF-conservative threshold is more sensitive to sunglint than the algorithm employing a clear-conservative threshold (Zhao and Di Girolamo, 2006). Finally, Di Girolamo and Davies (1997) derived an

equation (Eq. 17 in their paper) that gives the correct CF for a perfectly clear-conservative cloud mask given the sensor resolution and the true scale of the cloud,

$$A_{17} = A_{\text{int}}(r_i) + \left[1 + \left(\frac{r_t}{r_i} \right)^2 \right] \frac{A_{\text{edge}}(r_i)}{2},$$

where A_{int} is the fraction of cloudy pixels that do not border a clear pixel on any of their eight sides or vertices (i.e. an interior cloud pixel), A_{edge} is the fraction of cloudy pixels that border a clear pixel on at least 1 of their eight sides or vertices, r_i is the image resolution and r_t is the resolution required to perfectly resolve the clouds, however defined. The major benefits of this correction are its computational simplicity and its theoretical basis. A_{17} is exact for scenes meeting the following assumptions: only cloud edge pixels can be partially cloudy, and a sufficiently large number of edge pixels must be present such that they average 50% cloudy. However, those assumptions are not always met, such as in the case of remotely sensed images, where the ratio of r_i to r_t is large. A cloud edge pixel will be mislabeled as interior if all of its neighboring pixels contain some cloud (Figure 7). To overcome this limitation Di Girolamo and Davies (1997) employed a pattern recognition technique (A_p) to estimate the true CF, since CF bias shows a dependence on the spatial distribution of cloud area. In addition to having large ratios of r_i to r_t , operational cloud masks are imperfect and do not catch every cloud, so A_{17} cannot provide an exact solution for remotely sensed cloud fraction. However, understanding the resolution effect bias for CF derived from perfect, clear-conservative cloud masks can act as a basis for understanding the resolution effect bias on CF derived from operational cloud masks.

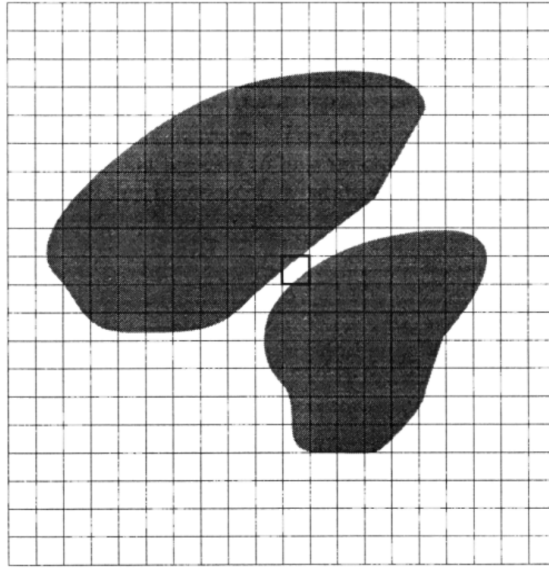


Figure 7. This figure and caption are borrowed with permission from Di Girolamo and Davies (1997). “The highlighted pixel is an example of a partially cloud-covered pixel being labeled as a cloud interior pixel rather than a cloud edge pixel.”

The goals of this study are to examine the resolution effect as a function of resolution and true CF on both perfect and operational clear-conservative cloud masks. Specifically, the results from Di Girolamo and Davies (1997) will be reconstructed with larger datasets and the following questions will be answered:

- 1) With future instruments in mind, what resolution is necessary to reduce the CF bias to 1%?
- 2) How do correction techniques perform on both a perfect, clear-conservative cloud mask and an operational clear-conservative cloud mask, the RCCM?
- 3) What is the impact of the resolution effect on a CF climatology derived for the tropical Western Atlantic from the RCCM?

2. Data

High resolution satellite datasets can provide knowledge of the true cloud fields. Long-term observations at high resolution and coincident wide-swath satellite datasets are required to fully evaluate the resolution effect. With the space-time coincident observations made by the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) and MISR, we are now able to extensively examine the resolution effect on CF. The 15-m resolution ASTER data provides an excellent reference for “cloud truth”, especially for small cumulus clouds as discussed in Zhao and Di Girolamo (2006 and 2007).

2.1 Instrument and Data Overview

The two instruments used in this study are ASTER and MISR whose relevant attributes are summarized in table 1. More detailed descriptions can be found in Abrams (2000) and Diner et al (1998) respectively. Isolating the resolution effect on CF can be difficult when there are variable background radiances and view angles (Henderson-Sellers and McGuffie, 1990), so data for this study was acquired over the ocean from the nadir camera only. The contrast between the dark ocean background and the bright cumulus clouds should largely remove the bias due to improper cloud detection as long as sunglint contamination is not severe (Zhao and Di Girolamo, 2007). However, there is relatively little oceanic ASTER data available, since its primary missions are land surface based, and ASTER does not collect data unless tasked to do so. The data used in this study was collected by previous studies: the Gulf of Mexico Atmospheric Composition and Climate Study (GoMACCS) (Parrish et al, 2009) over the Gulf of Mexico from July-September 2006, the Rain in Cumulus over the Ocean (RICO) field campaign (Rauber et al, 2007) over the Caribbean Sea from September-December 2004, and over the Indian Ocean from November 2006 to April 2007, where the Indian Ocean Experiment (INDOEX) (Ramanathan et al, 2001) had previously taken place. The spatially and temporally overlapped MISR data were also collected. The high resolution masks used in this study are derived from the ASTER instrument’s 3N channel (760-860nm, nadir view), at 15 m resolution. At such high spatial resolution subpixel clouds are negligible. Each pixel is either completely cloud covered or completely clear. This allows us to take the CF derived from the 15m cloud mask as the true CF at that instant over that area (Zhao and Di Girolamo, 2006). A similar method using Landsat data was demonstrated by Wielicki and Parker (1992) and Krijger et al (2007).

	ASTER	MISR
Satellite Platform	Terra	Terra
Image resolution	15, 30, 90 m	275, 1100 m
Number of cameras	2	9
Camera angles	0, -27.6	0, ± 26.2 , ± 45.6 , ± 60 , ± 70.5
Number of channels	14	4
Wavelengths	560-11300 nm	443-865 nm
Swath width	60 km	360 km
Cloud Mask type	Hand threshold	(RCCM) automated
Cloud Mask resolution	15 m	1.1 km

Table 1. Selected attributes of the ASTER and MISR instruments.

2.2 Cloud Masks

2.2.1 ASTER masks

Cloud masks for the ASTER data were generated using a single threshold of the 3N channel L1B V003 ASTER registered radiance at the sensor product. A single threshold in this channel is appropriate for daytime scenes with enough background contrast (Zhao and Di Girolamo, 2007; Ackerman et al, 2008). The threshold was manually selected by viewing the radiance image adjacent to the cloud mask image to visually verify the accuracy of the chosen threshold. It was considered correct if very little change occurred when it was nudged in either direction, and the mask was judged to accurately represented the cloud extent in the radiance image. Any scene that was not easily masked with a single threshold was removed from the dataset. Those scenes often contained intense sun glint, which is a common problem for cloud detection, because it raises the radiance of clear pixels, reducing the contrast in the scene (Zhao and Di Girolamo, 2004). As the scenes were masked they were also categorized based on apparent cloud morphology. The cumulus category could contain nothing but cumulus clouds. The cumulus and cirrus category was composed of scenes containing both cumulus and cirrus clouds and no other cloud type. The “other” category was composed of scenes that contained a cloud type or mixture of cloud types other than cumulus, cirrus or cumulus and cirrus. The dominant cloud type was small trade wind cumuli, but some cirrus and stratiform clouds were present in addition to larger convective clouds. See appendix C and the associated supplementary document for categorization of individual scenes.

There were 1405 60km x 60km ASTER masks produced. Of that data 73.2 % is from the Indian Ocean, 24.3 % is from the Caribbean Sea, and 2.5 % is from the Gulf of Mexico. To address the question of how varying the resolution affects the CF bias, the original resolution ASTER cloud masks were degraded to progressively lower resolutions. To avoid complications from the swath edges, the largest inscribed rectangle, containing a whole number of degraded pixels, was cut out of the 60km swath. The true CF (A_t) at each degraded resolution is the ratio of the number of cloudy 15m ASTER pixels divided by the total number of 15m ASTER pixels within the rectangular area. Therefore, the distribution of A_t is different at each resolution. The dataset for the calculations performed at constant resolution and displayed as a function of true CF contains 8571 17.6 km x 17.6 km regions over tropical oceans. This domain size was chosen so that results could be directly compared to results for an imperfect clear-conservative cloud mask that reports CF at that scale, the RCCM. Completely clear and complete cloudy regions were eliminated of the remaining data 68% are from the Indian Ocean, 30% from the Caribbean Sea, and 2% from the Gulf of Mexico. The distribution of A_t for this data is shown in figure 8.

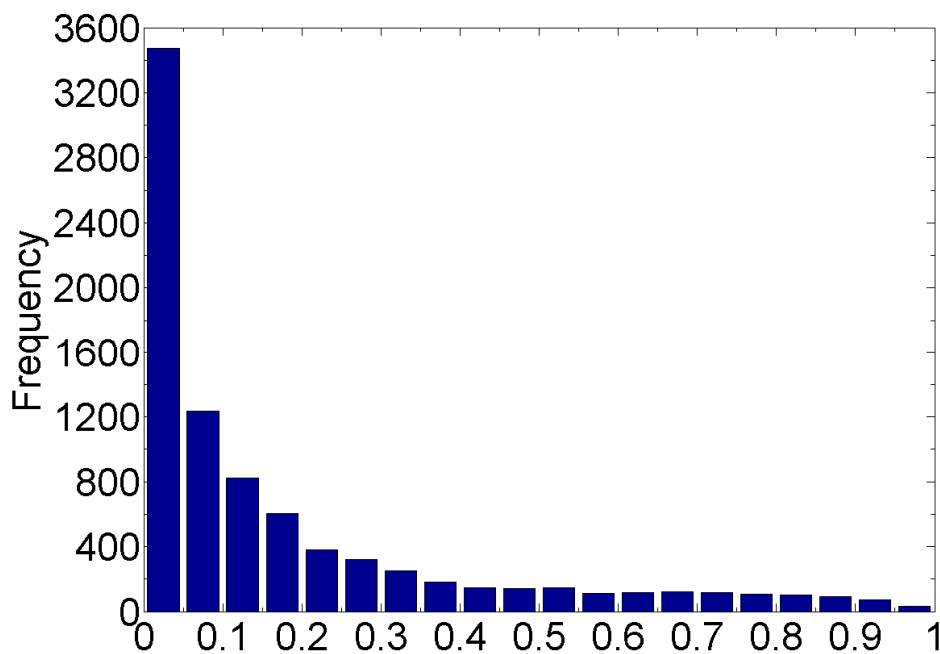


Figure 8. Histogram of true cloud fraction over 17.6km x 17.6 km domains from hand thresholded ASTER masks used in this study.

2.2.2 MISR masks

MISR's RCCM was designed to be as close as possible to a perfect, clear-conservative cloud mask (Diner et al, 1999). It has been demonstrated to behave as a clear-conservative cloud mask in terms of CF bias; however, it is not a perfect detector of clouds since it occasionally misses thin and small clouds (Zhao and Di Girolamo, 2006). Zhao and Di Girolamo (2006) found an average agreement rate of 83% between the

RCCM pixel classification and a perfect clear-conservative mask's pixel classification for 124 tropical cumulus cloud scenes, indicating that the undetected clouds are the exception rather than the rule. Another demonstration of its clear-conservative nature can be found in Zhao et al (2009). They found that the level of direct cloud contamination in the MISR aerosol products, which depend partially on the RCCM to mask out cloud, was very low, below the level of other uncertainties.

The RCCM is generated for each of MISR's 9 cameras; however, only the nadir view will be used. Each 1.1 km pixel is labeled either high confidence cloudy, low confidence cloudy, high confidence clear, low confidence clear, or no-retrieval. For this study high and low confidence categories were combined and each pixel was relabeled as either cloudy, clear, or no-retrieval. Details of the masking algorithm and product can be found in Diner et al (1999) and Zhao and Di Girolamo (2004). The RCCM version F04_0025 data used in this study contains 4325 17.6 km x 17.6 km regions from the same locations and times that the ASTER data was drawn from: 59% from the Indian Ocean, 2% from the Gulf of Mexico, and 39% from the Caribbean Sea. The distribution of A_t for this data is shown in figure 9. The number of regions was reduced by more than half compared to the ASTER data. This is due to the additional restriction of having coincident ASTER and MISR data. There are times when the number of MISR or ASTER no-retrieval pixels exceeded the maximum allowed, which was set to 5% of the total number of pixels. There are times when the MISR region straddled two ASTER scenes, in which case the region was not used. If for any other reason the number of ASTER pixels was less than 95% of the total possible number of ASTER pixels in a 17.6km x 17.6km region the region was not used.

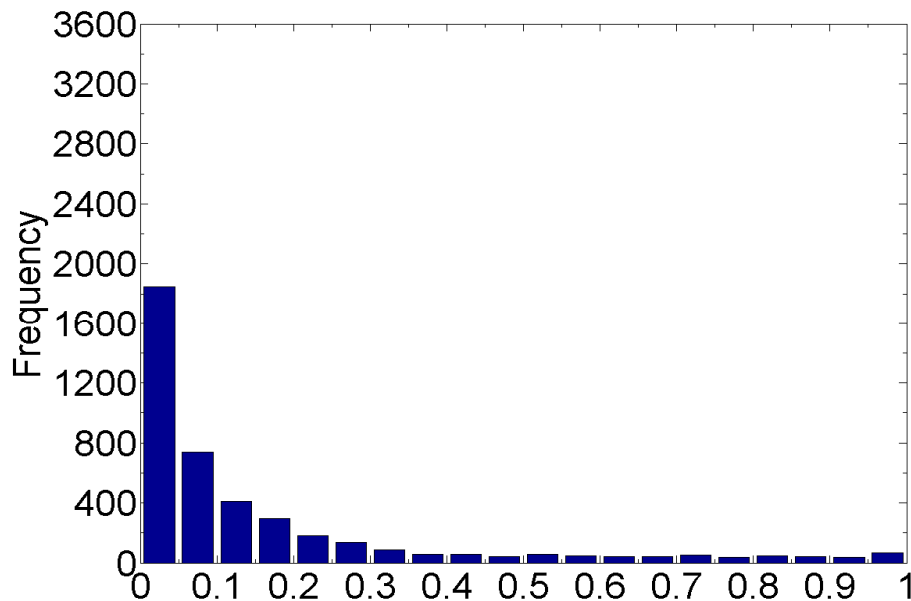


Figure 9. Histogram of true cloud fraction over 17.6km x 17.6 km domains from MISR RCCM regions used in this study.

3. Pattern Recognition Technique

This correction technique, adopted from Di Girolamo and Davies (1997), is implemented slightly differently in the current study. The idea behind a pattern recognition technique is that cloud masks that have similar spatial features will have similar true CFs. The degree of correction necessary depends on the ratio of cloud size to pixel size (Shenk and Salomonson, 1972) and the distribution of true cloud area (Di Girolamo and Davies, 1997); both are functions of cloud type (Wielicki and Parker, 1992), and cumulus CF shows the greatest dependence on resolution (Wielicki and Welch, 1986). The pattern recognition technique has an advantage over A_{17} because it relies on scene specific information on the spatial distribution of cloud area to determine the amount of correction necessary. Di Girolamo and Davies (1997) assembled a feature vector for pattern recognition composed of A_e and A_{edge} , as well as standard shape and texture measures derived from gray level difference statistics and moment invariants. These features were used to characterize the cloud area in the image. Their optimized feature vector included A_e , A_{edge} , the first moment invariant (Hu, 1962), mean, variance, and entropy of the gray levels in the scene (Weszka et al, 1976) at the scale of r_i . The feature set from Di Girolamo and Davies (1997) has been adopted for the pattern recognition technique performed in this study since there is no reason to believe that these scenes are fundamentally different from those used previously. However, the mean, variance, and entropy have been derived from the gray level co-occurrence matrix which is derived from the scene rather than deriving the features from the scene itself. The features, second-order gray level statistics, derived from the matrix will be very similar to gray level difference statistics derived from the scene (Weszka et al, 1976). Each feature is designated as $\chi_i, i \in \{1, \dots, 6\}$, so the feature vector \mathbf{X} is six-dimensional (χ_1, \dots, χ_6) . Ultimately, the goal is to match a low resolution cloud mask to one for which the true CF is known. Test scenes are chosen at random and separated from the rest of the data. The remaining scenes become the training set. The training set is composed of feature vectors derived from each of the N coarse resolution cloud masks $\{\mathbf{X}_1, \dots, \mathbf{X}_N\}$, and the associated true CF $\{A_{t1}, \dots, A_{tN}\}$ was extracted from the fine resolution mask. A nearest neighbor technique was used to find the closest match in the training set. The nearest neighbor was determined by minimizing the Euclidean distance between the feature vector of the test scene and the feature vector of each member of the training set. The test scene was assigned the true CF of the training set member whose Euclidean distance to it was shortest. This became the pattern recognition CF estimate for the test scene. This can also be described mathematically. The test scene, \mathbf{X}_0 , is assigned a pattern recognition CF, A_p , by the nearest neighbor classification rule as follows: Let \mathbf{X}_i satisfy the minimum of $\{||\mathbf{X}_0 - \mathbf{X}_i||\} \forall i \in \{1, \dots, N\}$, where $||\cdot||$ denotes the Euclidean metric. A_{ti} is assigned to \mathbf{X}_0 as A_{p0} .

For the application of this technique for this study, scenes with $A_e = 1$ or $A_e = 0$ were eliminated from the dataset since no information is available in those cases for any correction to be done. The test set was limited to 1/5 of the remaining dataset. The nearest neighbor was chosen for each member of the test set,

and then test set and training set were recombined and a new test set was chosen. To increase the robustness of the results and to ensure that the results were not dependent on the selection of the test set and training set, the process of selecting the test set from the training set was repeated 300-800 times depending on the size of the training set. This is called repeated random sub-sampling, a type of cross-validation (Geisser, 1975), a common resampling technique. The figures presented in this thesis are composed of the results of all the iterations.

4. Resolution Effect Bias of CF from a Perfect, Clear-Conservative Cloud Mask

The datasets discussed in chapter 2 will be used to recreate the results found by Di Girolamo and Davies (1997) and to expand them using current computing resources and data. Specifically, this chapter will seek to answer the following questions about the techniques used to estimate CF from a perfect, clear-conservative cloud mask: how does the bias depend on resolution; and how does the bias depend on true CF.

4.1 Dependence of CF biases on resolution

Analyzing ASTER scenes degraded from 15 m spatial resolution to 12 lower resolutions between 45 m and 1.2 km helped determine the effect of sensor resolution on CF derived from a perfect, clear-conservative cloud mask and address the question of optimal instrument resolution. The perfect, clear-conservative nature of the cloud mask was maintained by labeling cloudy each low resolution pixel containing at least one 15 m cloudy pixel. This technique was also employed by Dey et al (2008) and Krijger et al (2007). The standard method determined the CF estimate, A_e , of the degraded scene as well as the true CF as measured at 15 m resolution, A_t . Pattern recognition, A_p , and a simple equation, A_{17} , were used to estimate the CF at each of the 12 resolutions in hopes of achieving a value closer to A_t .

The results of technique bias with decreasing resolution are shown in figure 10. Here and for the rest of this thesis technique bias is defined by subtracting A_t from the estimated CF. The median and quartiles of the technique bias are plotted. They are robust and resistant statistics used to determine the central location and spread of data with an unknown distribution (Wilks, 2006), whereas mean and standard deviation apply only to normally distributed datasets. Generally speaking, a low magnitude of median bias means that the technique performs well at reducing the CF bias. A low inter-quartile range (IQR) means that the technique performs consistently over the middle 50% of the data. In other words, when the IQR is small, large oppositely signed biases are not canceling to produce a low median bias.

At the lowest resolution tested, 1155m, the standard estimate, A_e , has a median overestimation of 0.331 and IQR of 0.345. As noted in Di Girolamo and Davies (1997), the degree of overestimation for a specific scene depends on the distribution of the cloud area: whether it is few large clouds with few partially filled pixels, or many small clouds with many partially filled pixels. As the pixel size decreases, becoming closer to or less than the typical cloud size, the median bias of A_e decreases and the IQR decreases. This is due to a decrease in the number of partially filled pixels, regardless of the distribution of true cloud area, since each pixel is increasingly likely to be either completely clear or completely cloud covered.

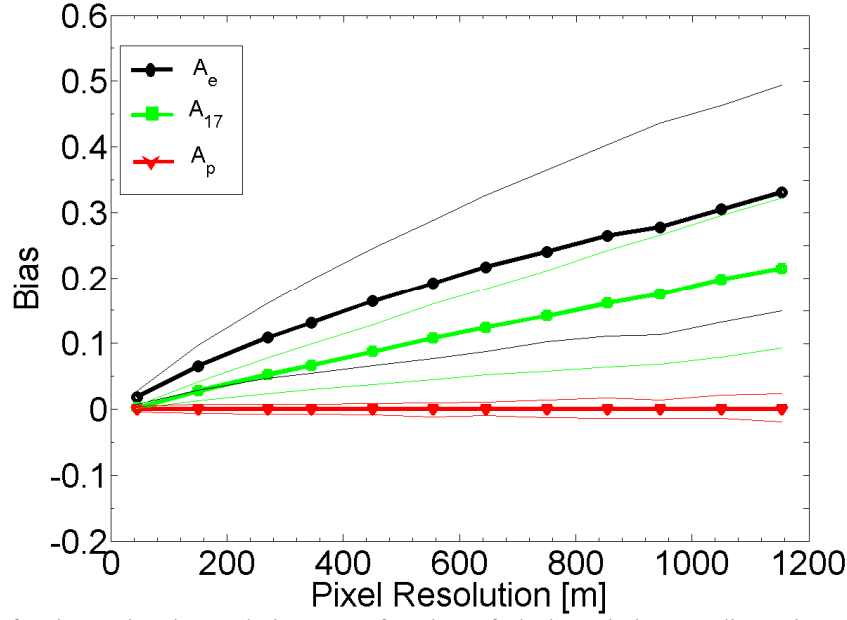


Figure 10. Cloud fraction estimation technique as a function of pixel resolution. Median values of bias are plotted in thick lines and bias quartiles are plotted in thin lines.

The simple equation correction, A_{17} , improves upon the standard estimate of CF, A_e . The median bias at 1155m is 0.214 and the upper quartile of bias, 0.322, is lower than the median bias of A_e . The upper quartile of A_{17} bias lies below the median A_e bias at every resolution tested. At typical resolutions of current cloud masks, $\sim 1\text{km}$, A_{17} overestimates the CF by 0.092-0.322 50% of the time. At 150m resolution the median A_{17} bias is 0.025. This is close to the mandate of 1% maximum CF error, which would require a resolution of 80 m or less. This is comparable to the 100 m resolution suggested by Shenk and Salamonson (1972) for area averaged cloud diameters of 1 km and under the assumption that cloud edge pixels average 50% cloudy. The pattern recognition CF, A_p , has a median bias of 0 at every resolution tested. The IQR is also very small, relative to the other techniques, with a maximum of 0.043 occurring at a resolution of 1155m. However, it is considerably more computationally expensive to compute A_p relative to A_e or A_{17} .

4.2 Dependence on cloud area distribution

Di Girolamo and Davies (1997) quantified the CF bias derived from perfect, clear-conservative cloud masks as a function of true cloud area distribution. They used both simulated cloud fields and AVHRR masks as reference “truth” and degraded both by a factor of 32 to simulate the resolution effect. For the current study the degree of degradation of the original image was chosen to be as close as possible to the resolution of current cloud mask products, 1.1km or a conglomerate of 74 15 m ASTER pixels on a side. The distribution of the true CF of the original data was described in chapter 2 and is shown in figure 8: 63% of scenes contained trade wind cumulus, 23% contained a mixture of cumulus and cirrus, and 14% contained some other type of cloud, such as deep convective, or stratiform clouds. The standard method determined

the CF estimate, A_e , of the degraded scene as well as the true CF at the original 15 m resolution, A_t . The pattern recognition and simple equation techniques for estimating CF were then applied.

The results of the technique bias as a function of A_t are shown in figure 11. The data was divided into 20 equally spaced bins, resulting in a bin width of 0.05, and the median and upper and lower quartiles of bias for each bin are plotted. The A_t referenced in the following discussions is the bin center. The peak median A_e bias of ~ 0.635 occurs for the bin centered at $A_t = 0.125$. The upper quartile of bias is 0.720, and the lower quartile of bias is 0.495, meaning 50 % of the time A_e will have a value between 0.62 and 0.85 when $A_t = 0.125$. This does not exactly match the previous results by Di Girolamo and Davies (1997) and Zhao and Di Girolamo (2006) which found peak biases at $A_t = 0.25$. However, some notable differences exist in the datasets used. Zhao and Di Girolamo (2006) used a much smaller dataset composed of scenes containing only cumulus clouds only over the Caribbean Sea. Di Girolamo and Davies (1997) relied on simulated cloud scenes that may not have represented the variety of cloud patterns observed in nature. Their distribution of A_t for the AVHRR dataset is quantifiably different from this ASTER dataset. Additionally, their data was degraded by a factor of 32 compared to our 74, which may shift the location of the peak bias. Also, differing domain sizes has an effect on the CF distribution (Dey et al, 2008).

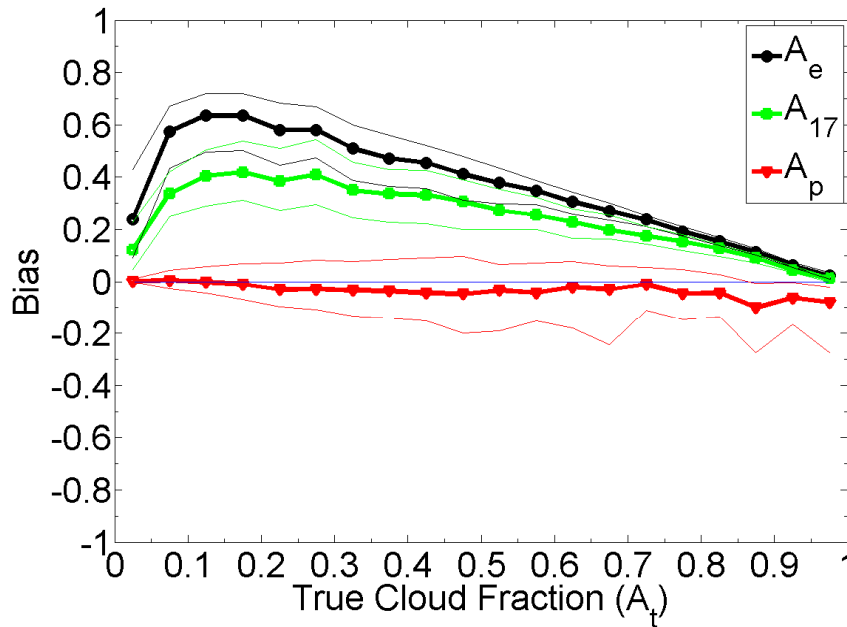


Figure 11. Cloud fraction estimation technique as a function of true cloud fraction as applied to 1.1km resolution degraded ASTER cloud masks. Median values of bias are plotted in thick lines and bias quartiles are plotted in thin lines. Values are calculated for 20 CF bins of 0.05 width and plotted at the location of the bin center.

The peak in median A_{17} bias of ~ 0.418 occurs for the bin centered at $A_t = 0.175$. The upper quartile of bias is 0.538 and the lower quartile of bias is 0.310, meaning 50% of the time A_{17} will have a value between 0.485 and 0.713 when $A_t = 0.175$. Again the upper quartile of the A_{17} bias is less than the median bias of A_e .

Overall, there is an improvement by A_{17} over A_e ; however, the magnitude of the bias is still dependent on the A_t , which is evident by the bias curve shape similarity. This result is in-line with the results for A_{17} in Di Girolamo and Davies (1997).

The pattern recognition technique has a magnitude of median bias less than 0.05 for almost all values of A_t . The slight negative bias in the mid-range of A_t is hypothesized to be due to the reduction in the number of scenes at these CFs. A larger negative bias of ~ 0.1 exists for the bin centered at $A_t=0.875$, which was also seen in Di Girolamo and Davies (1997). This highlights a natural breakdown in the pattern recognition for large values of A_e . The technique draws information from the distribution of clear and cloudy pixels in the mask. The more cloudy pixels the larger the value of A_e , however, scenes with large A_e do not necessarily have large A_t . Allocation rates provide a more detailed look at the technique biases by providing frequency information for each combination of technique-estimated CF and true CF. The allocation rates in figure 12a show that A_e saturates to a value near 1.0 fairly early, around $A_t=0.7$. At lower A_t there are still many scenes with large A_e . This means that there are a variety of A_t values that correspond to the same value of A_e , leading to larger IQR for larger A_e .

Despite the variability in the underlying distribution of true cloud area, all features derived from a scene with large A_e are limited in value since they are derived from the same mostly cloudy mask. For example, a scene with many small clouds distributed evenly throughout the domain may appear nearly 100% cloudy in the low resolution, clear-conservative mask. The number of edge pixels will be underestimated since a partially filled pixel surrounded by partially filled pixels appears as a cloud interior pixel. A scene with few large clouds will have a similar cloud mask, however, most of its cloud interior pixels are fully cloud covered. During the pattern recognition process one of these scenes may be mistakenly matched to the other. The negative bias arises because so many more scenes in the dataset have low A_t and high A_e and associated features than have high A_t and high A_e .

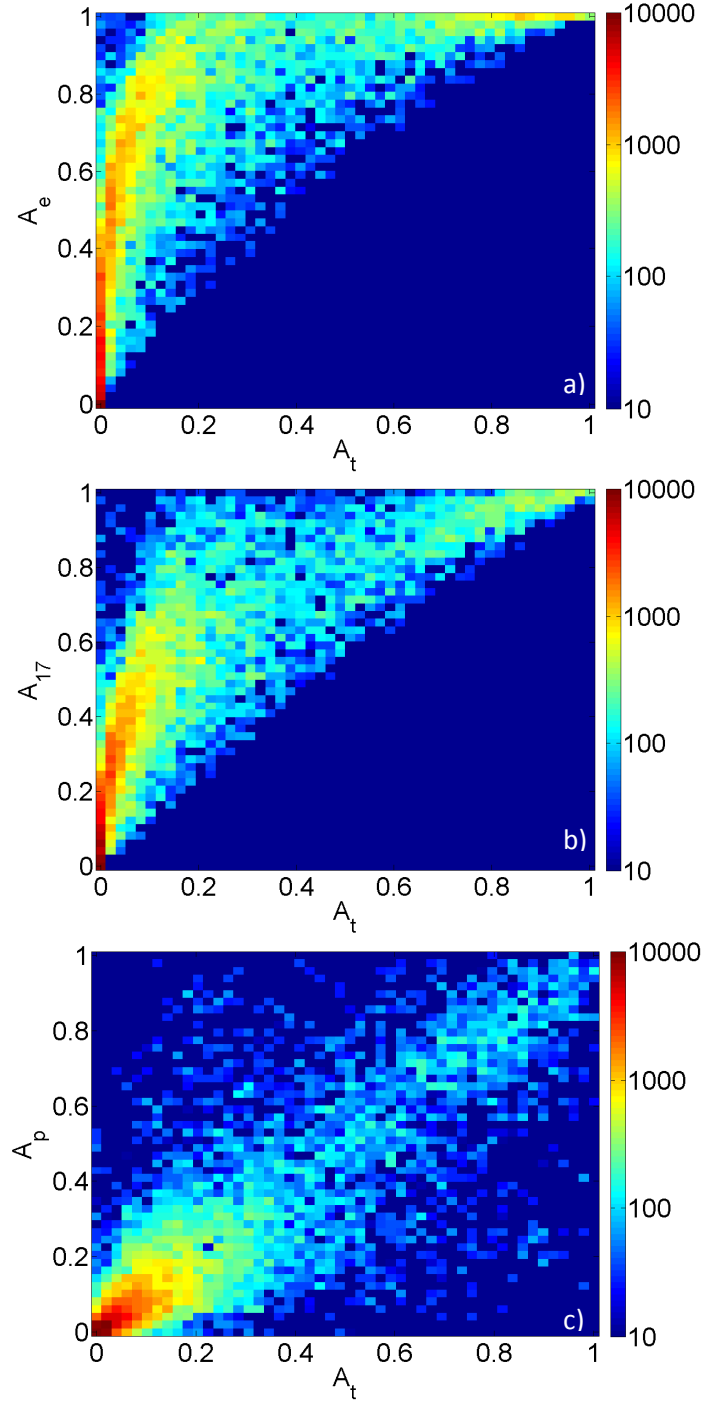


Figure 12. Allocation rates for each of the techniques a) A_e , b) A_{17} , c) A_p versus A_t as applied to the ASTER data. An allocation rate plot is a two-dimensional histogram featuring the value of a technique's CF estimate on the y-axis and the value of true CF on the x-axis.

The sign of the bias and the magnitude of the IQR show a clearer trend with increasing A_e . Figure 13 shows the technique bias as a function of A_e . Note that the A_p IQR increases steadily with increasing A_e and the negative bias is gone. For the largest values of A_e , the pattern recognition remains the technique with the

lowest magnitude of median bias. However, for the largest values of A_t , the lowest magnitude of median bias is provided by A_{17} (Figure 11). This implies that if the true CF is known to be near 1 through a priori information, such as areas dominated by expansive stratiform cloud cover, the best choice of correction technique is A_{17} . If the distribution of true CF is unknown, the best choice in technique is A_p ; little bias will result from the pattern recognition CF estimate. Ideally, future CF climatologies will feature CF estimation techniques that vary by region depending on which is most appropriate. For regions dominated by $A_t > 0.9$, A_{17} is most appropriate, although, it does not improve the bias by much compared to A_e . So, it may make more sense, computationally, to not correct CF in those regions at all.

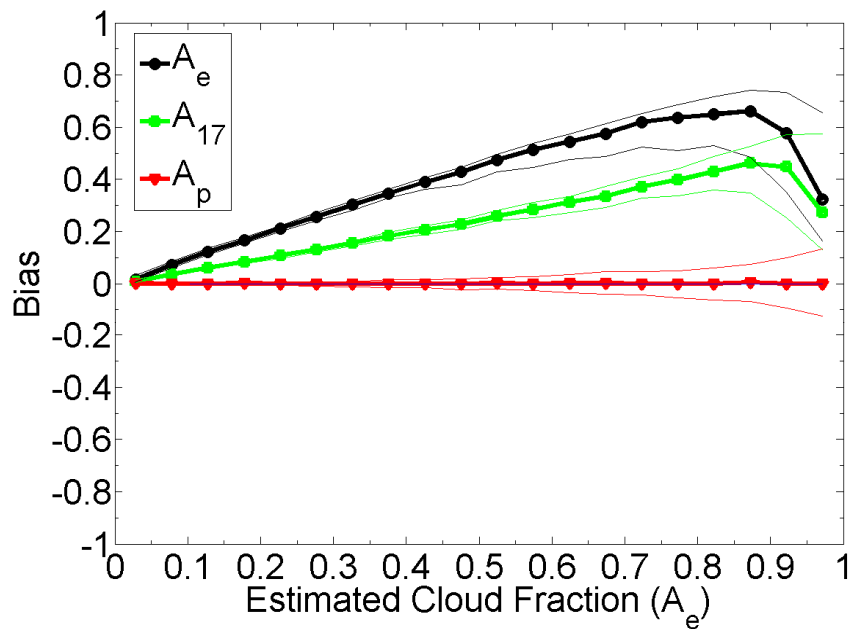


Figure 13. Cloud fraction estimation technique as a function of traditional cloud fraction estimation as applied to 1.1km resolution degraded ASTER cloud masks. Median values of bias are plotted in thick lines and bias quartiles are plotted in thin lines. Values are calculated for 20 CF bins of 0.05 width and plotted at the location of the bin center.

5. Correcting CF from an Operational Imperfect, Clear-Conservative Cloud Mask

The theoretical limitations and sources of error in estimating CF from a perfect, clear-conservative cloud mask provide a basis for understanding and correcting the CF resolution effect bias from an operational cloud mask that aims for perfect cloud detection, such as the RCCM. However, the RCCM, like any other operational mask, is not a *perfect* cloud mask. Therefore, the results should differ in some way, relative to the results from the ASTER masks at the same resolution. For example, for the same distribution of cloud area, A_e derived from the ASTER mask should be larger than A_e derived from the RCCM, since the RCCM may miss the smallest and thinnest clouds. The goals of this chapter are to determine the efficacy of CF estimation techniques given an operational cloud mask whose behavior is close to that of a perfect cloud mask, and to determine the impact of correction on a CF climatology.

5.1 Challenges in correcting CF from MISR's RCCM

As in chapter 4 the ASTER data is used as “truth” for reference, however, rather than degrading it to simulate lower resolution cloud masks the RCCM, an operational clear-conservative cloud mask, will be used. The pattern recognition technique is exactly the same as used in chapter 4 and described in chapter 3; the features are identical as well as the resampling method. However, ASTER and the RCCM report their data on different geospatial grids that cannot be perfectly aligned. Therefore, the true CF, A_t , of a MISR pixel was calculated by tallying the cloudy ASTER pixels that fell closest to the center of that MISR pixel then dividing by the total number of ASTER pixels, tallied in the same way. The error introduced by this method of co-registration is random and negligible (Zhao et al, 2009). The distribution of A_t was shown in figure 9: 74% of the scenes contained trade wind cumulus, 26% of the scenes contained stratiform or deep convective clouds. Any 17.6km x 17.6km regions falling within ASTER scenes categorized as containing cirrus were removed from the dataset, since the manual detection of thin cirrus from ASTER is much better than the automated RCCM and any discrepancies are not part of the resolution effect as defined in the thesis.

The results of the technique bias as a function of A_t as applied to the RCCM are presented in figure 14. The shape of the A_e bias curve is similar to that for the degraded ASTER data (Figure 11). The upper quartile of A_e bias for the bin centered at $A_t = 0.175$ is 0.671 and the lower quartile is 0.454, meaning 50% of the time A_e will be between 0.629 and 0.846 when median bias is at its maximum. This peak median bias of A_e occurs at the next largest bin of A_t than the degraded ASTER data and is slightly lower in magnitude, 0.582. This result is expected since the RCCM is not a *perfect* cloud mask. Therefore it is more likely to misclassify some subpixel or optically thin cloud as clear. Overall, the behavior of the A_e bias for the RCCM is very similar to the ASTER results, indicating the clear-conservative nature of the RCCM. Therefore the correction techniques should perform reasonably well on this dataset.

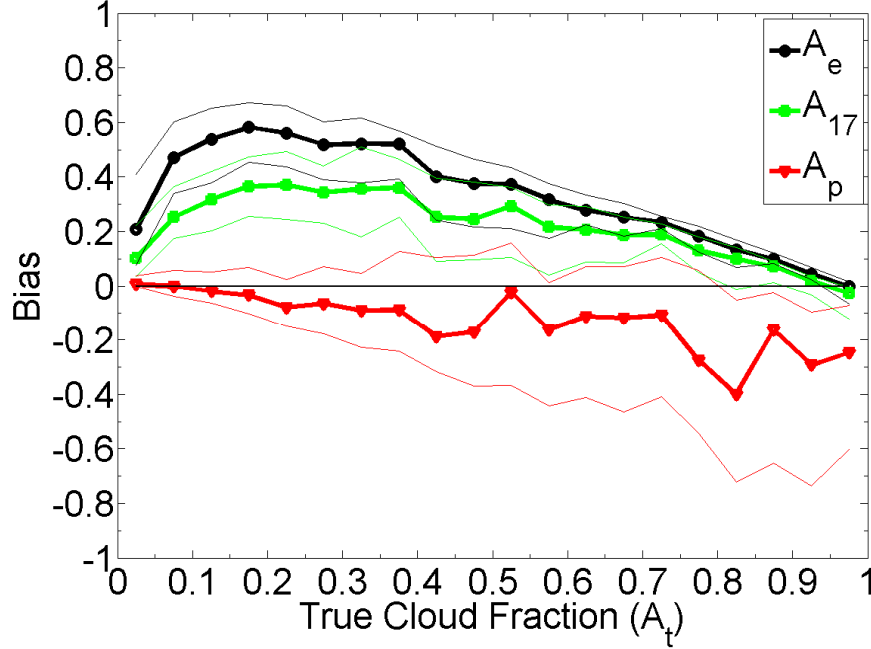


Figure 14. Cloud fraction estimation technique as a function of true cloud fraction as applied to 1.1km resolution RCCM data. Median values of bias are plotted in thick lines and bias quartiles are plotted in thin lines. Values are calculated for 20 CF bins of 0.05 width and plotted at the location of the bin center.

A_{17} again improves the magnitude of the bias but displays a dependency on A_t . The peak median bias is 0.37 and located at $A_t = 0.225$, compared to a peak of 0.418 for the bin centered at $A_t = 0.175$. The upper quartile of A_{17} bias at that point is 0.494 and the lower quartile is 0.245, meaning 50% of the time A_{17} will lie between 0.47 and 0.719 when $A_t = 0.225$, which is a similar IQR to that located around the peak median bias for the ASTER data.

Compared to the ASTER results (Figure 11) the median A_p bias becomes more negative and the IQR increases with increasing A_t . The results for the ASTER data also exhibited negative bias but the magnitude of the negative bias didn't start increasing until values of $A_t > 0.8$. The allocation rates reveal a major problem with the pattern recognition technique as applied to the RCCM (Figure 15). Figure 15c shows the allocation rates for A_p . There is very little skill exhibited at any value of A_t . It is also apparent that A_e sometimes under-predicts A_t (Figure 15a). This does not occur for the degraded ASTER data (Figure 12a) and would never occur for any perfect, clear-conservative cloud mask. Since CF derived from a perfect, clear-conservative cloud mask acts as an upper bound on true CF, A_e can never be less than A_t . The RCCM is known to have some issues distinguishing sun glint from cloud. This could be negatively impacting the efficacy of the pattern recognition technique on the RCCM dataset.

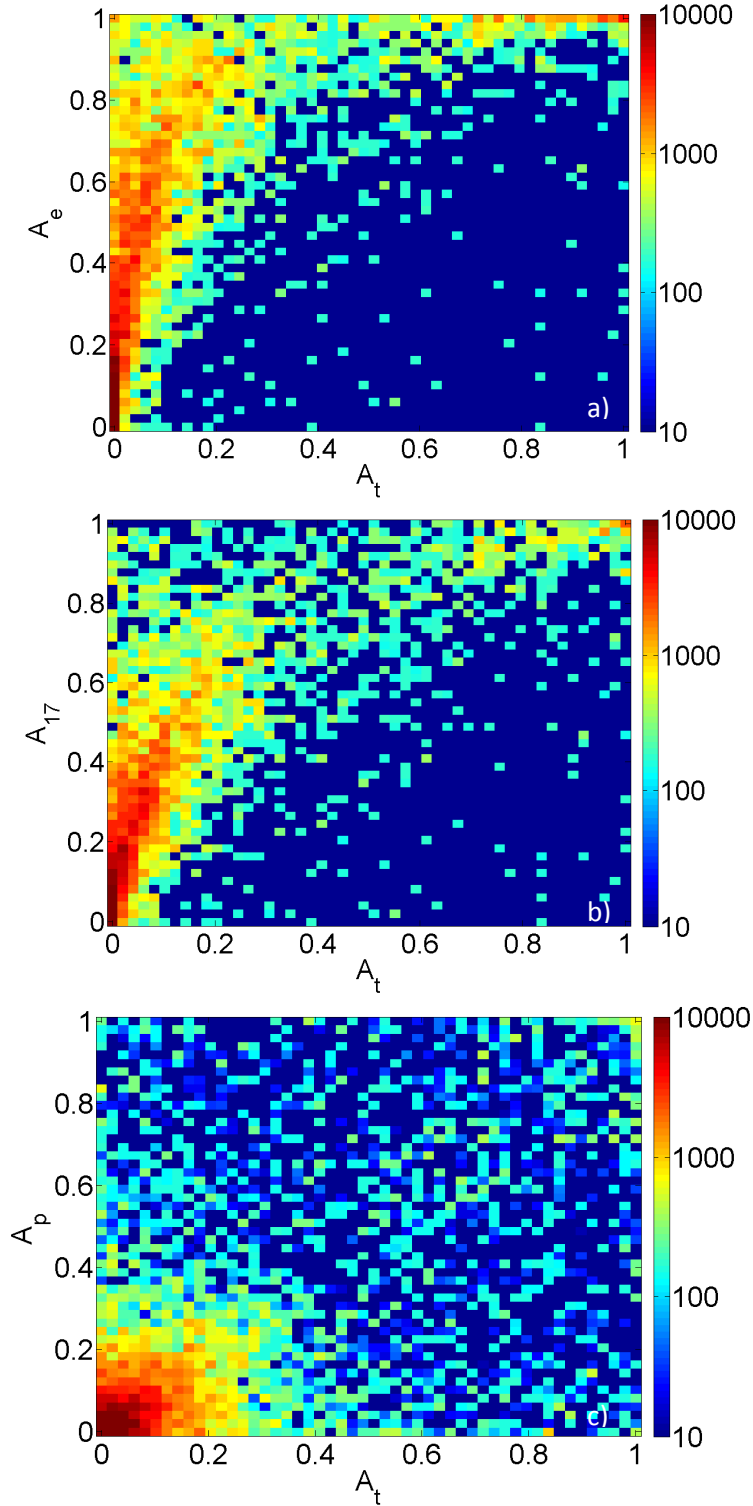


Figure 15. Allocation rates for each of the techniques a) A_e , b) A_{17} , c) A_p versus A_t as applied to the RCCM data.

Sun glint contamination results in many regions with very large A_e even for low values of A_t . This is apparent in figure 15a as A_e reaches high values for low values of A_t . Sun glint does not impact the true CF, because the threshold chosen during hand masking of the ASTER image was high enough for the sun glint to

remain part of the clear-sky background. Again, the ASTER scenes that were not easily thresholded, due to the severe sun glint contamination, were excluded in this study. The saturation to large A_e does not occur at such low values of A_t for the ASTER dataset. This can be observed by comparing the upper left corners of figures 12a and 15a. Whether the reference ASTER mask of the contaminated scene is located in the training set or test set will determine the sign of the A_p bias. If a scene contains sun glint the RCCM may inaccurately label it as cloud. In those cases A_e derived from the RCCM is too high and therefore has a positive bias. When that same region appears in an ASTER image, a high enough threshold is chosen to exclude the sun glint. When that scene is included in the training set it may be selected as a match for a scene with a legitimately large A_e and associated features. The scene that was “corrected” will be assigned a value of A_p that may be much lower than its A_t . This would cause A_p to have a positive bias. Alternatively, if that scene is in the test set its selected match from the training set may have a legitimately large A_e and associated features. In this case the value of A_p assigned to it may be much larger than its A_t , causing a negative bias.

Removing scenes with any potential sun glint contamination reduced the training set to the point that it was no longer useful. However, the more severely contaminated scenes were removed after visual scrutinization. The resulting dataset contained 50% Indian Ocean scenes, 46% Caribbean Sea scenes, and 4% Gulf of Mexico scenes. 88% contain trade wind cumulus; 1% contain a mixture of cirrus and cumulus clouds; and 11% contain clouds of another type or mixed type. The resulting distribution of A_t is shown in figure 16 and the new allocation rates are shown in figure 17. Unfortunately, many of the scenes with large A_t were found in scenes that were determined to have sunglint contamination and were therefore eliminated from the dataset, but there are many fewer instances of A_e under predicting A_t (Figure 17a), and the saturation of A_e for the restricted data does not occur at such low values of A_t . The new RCCM A_e allocation rates look similar to the ASTER A_e allocation rates (Figure 12a). The allocation rates for A_p are better, with peak allocation rates along the 1:1 line. The values of median and upper and lower quartiles of technique bias are presented as a function of A_t in figure 18. The low number of training data samples for $0.4 < A_t < 1.0$ (Figure 16) may contribute to the large fluctuations in A_p bias.

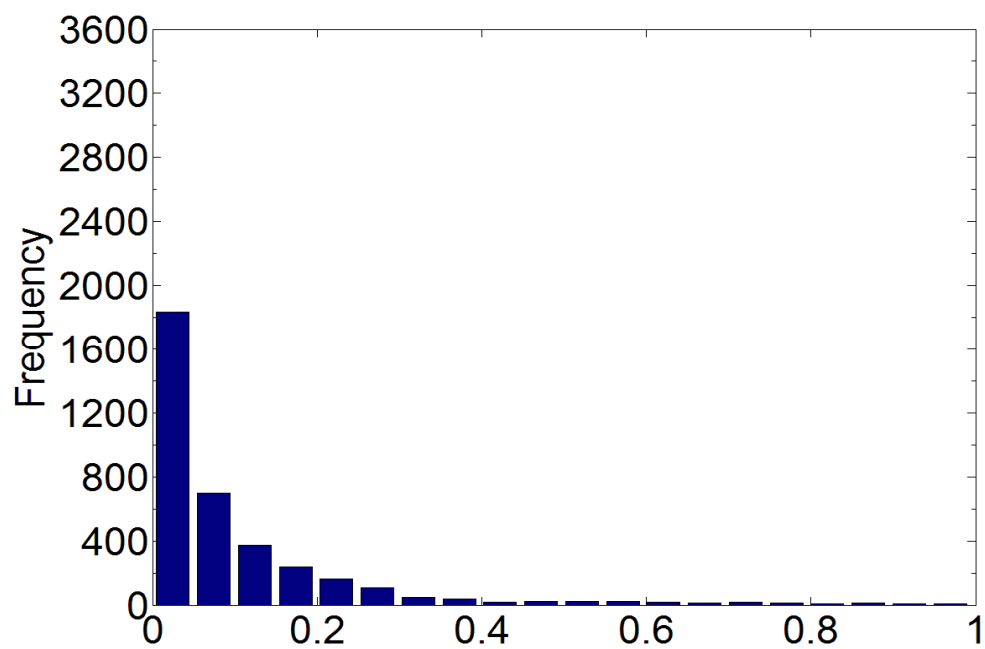


Figure 16. Histogram of true cloud fraction from MISR RCCM regions with the most severely sunglint contaminated regions removed.

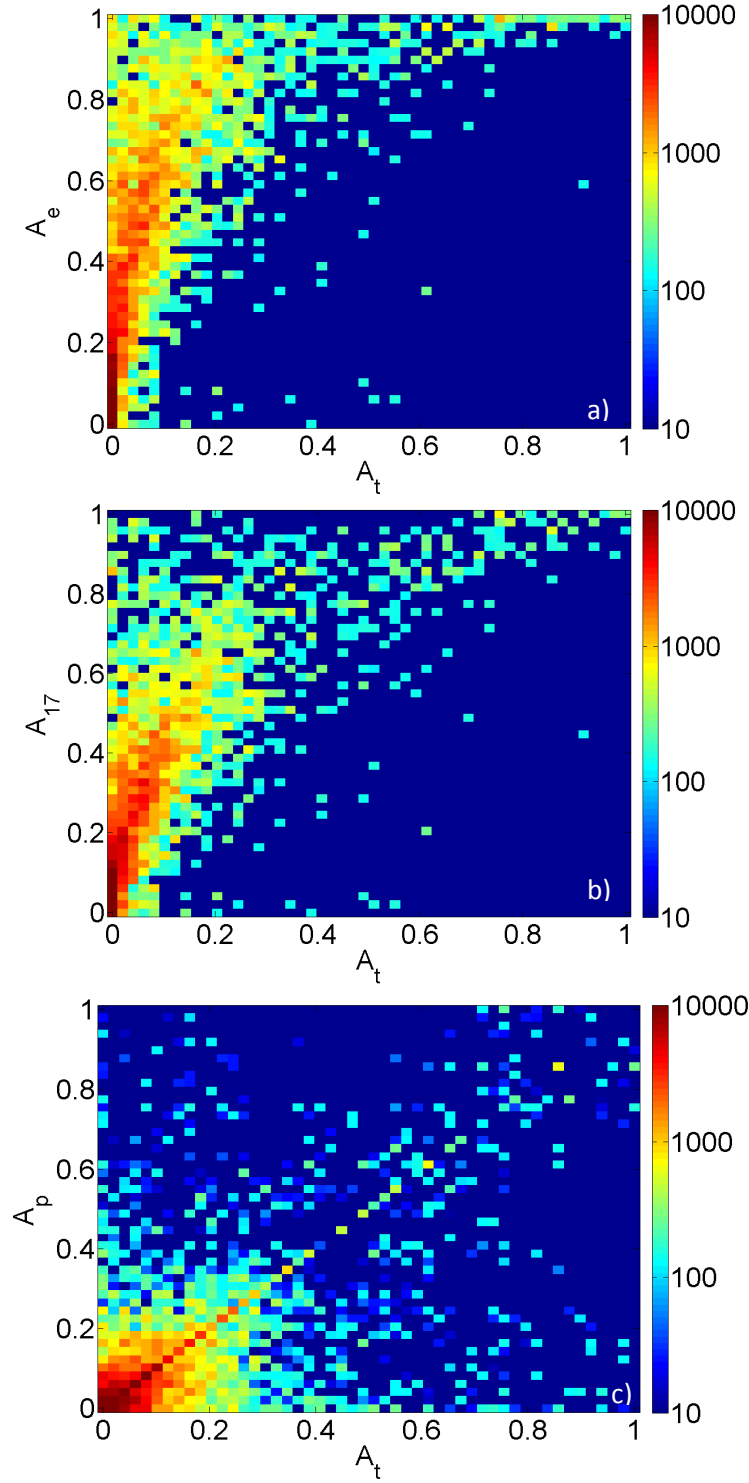


Figure 17. Allocation rates for each of the techniques a) A_e , b) A_{17} , c) A_p versus A_t as applied to the RCCM data with the most severely sunglint contaminated regions removed.

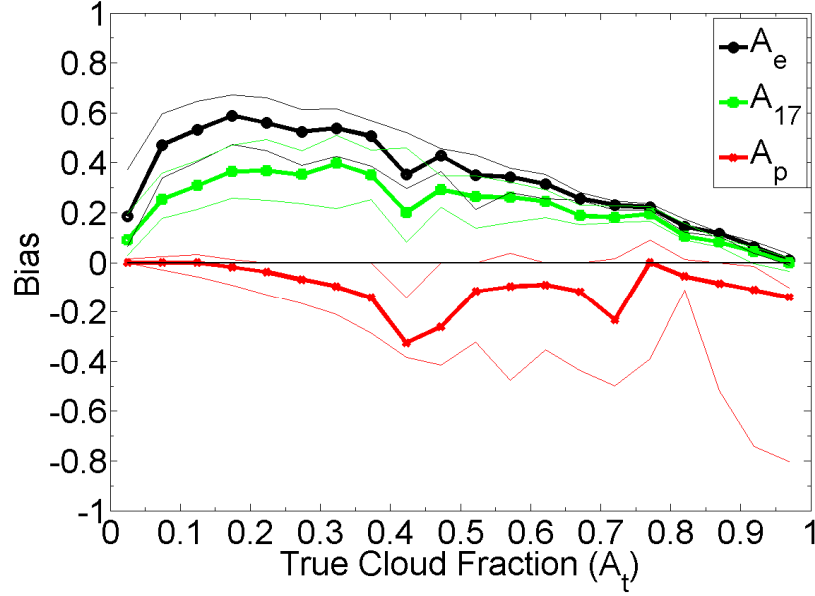


Figure 18. Cloud fraction estimation technique as a function of true cloud fraction as applied to 1.1km resolution MISR RCCM regions with most severely sunglint contaminated regions removed. Median values of bias are plotted in thick lines and bias quartiles are plotted in thin lines. Values are calculated for 20 CF bins of 0.05 width and plotted at the location of the bin center.

The frequency distribution of A_e (Figure 19) is much more uniform than the distribution of A_t (Figure 16). When looking at the technique bias versus A_e for the restricted dataset, the A_p median bias is 0 for all values of A_e and the IQR increases with increasing A_e (Figure 20). The differences between figures 18 and 19 and between figures 11 and 13 suggest that outliers with large negative bias strongly influence the median bias over the range of A_t where the distribution is sparse. Figure 20 shows that the pattern recognition technique is able to estimate A_t with little bias and with a lower magnitude of bias compared to the standard estimate or equation 17. However, the computational expense may not be worth correcting CF climatologies where A_t is known to be large since A_e has a lower bias than A_p for $A_t > 0.85$.

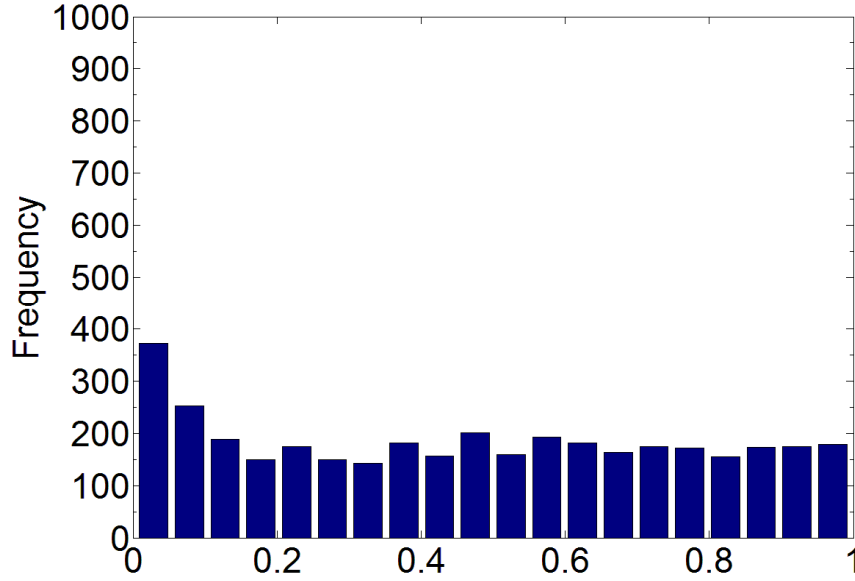


Figure 19. Histogram of standard estimation of cloud fraction from MISR RCCM regions with most severely sunglint contaminated regions removed.

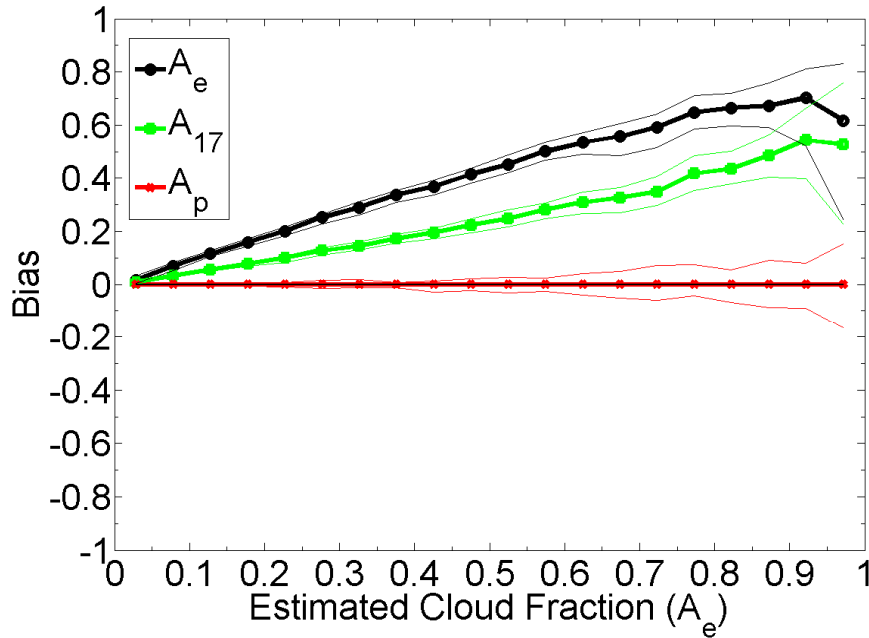


Figure 20. Cloud fraction estimation technique as a function of traditional cloud fraction estimation as applied to 1.1km resolution MISR RCCM regions with most severely sunglint contaminated regions removed. Median values of bias are plotted in thick lines and bias quartiles are plotted in thin lines. Values are calculated for 20 CF bins of 0.05 width and plotted at the location of the bin center.

These results imply that there are some technical challenges to applying the pattern recognition technique to an operational cloud mask. However, the results are promising and the pattern recognition technique can be expected to correct CF climatologies to a much smaller error compared to uncorrected CF or CF corrected by A_{17} .

5.2 Application: correcting CF climatologies

To give a first approximation of the impact of the above results the pattern recognition technique was applied to a regional CF climatology known to be overestimated by the RCCM. The region of the tropical Atlantic Ocean from 10°N - 20°N and 50°W - 60°W is dominated by trade wind cumulus, which have an expected CF well below 1.0, and therefore an appropriate location to demonstrate this correction technique.

To remove possible cirrus contamination the dataset was restricted by height rather than by cloud type, since exact cloud type of a given scene used to construct the climatology is unknown apriori. The MISR cloud classifier product, version F06_0011, contains the median cloud top height above the surface at 17.6km resolution. That value was used to filter out cloud top heights above 4km. This cutoff was chosen because it was found to approximate the maximum boundary layer depth during the RICO field campaign (Davison, 2009), which occurred during the time period that the Caribbean ASTER data was collected. The resulting training set contains 57% Indian ocean scenes, 41% Caribbean Sea scenes, and 2% Gulf of Mexico scenes. 86% of the scenes contain trade wind cumulus, and 14% of the scenes were low clouds of another type. The data to be corrected is also subject to this height restriction such that the following 10-year CF climatologies apply only to cloud cover below 4km.

All MISR orbits that fell within this region during the month of December from 2000-2009 were included in this climatology. Only 17.6 km x 17.6 km regions with valid height values were retained. The heights rely on the presence of “BestWind” stereoscopic height retrievals in the region. Regions with $0 < A_e < 1.0$ were corrected by the pattern recognition technique. Regions with $A_e=0$ or $A_e=1$ remain in the climatology, uncorrected. For a given year all 17.6 km x 17.6 km regions falling into a 0.5° x 0.5° box were averaged. Those averages were averaged to produce figure 21. Figure 21a is the uncorrected, 10-year, December CF climatology for clouds with top heights < 4km. It has an average CF of 0.496 over the 10° x 10° area. Figure 21b is the corrected, 10-year, December CF climatology for clouds with top heights < 4km. It has an average CF of 0.199 over the 10° x 10° area. The average CF was reduced by 0.297 by the pattern recognition technique, similar to Zhao and Di Girolamo’s (2006) results, where on average the RCCM overestimated the A_t of 124 RICO cumulus-only scenes by 0.36. These results also agree with long-term surface observations of the daytime, December cumulus CF in this region, ~0.2 (Warren and Hahn, 2007). Although the RCCM climatology may contain more than just the cumulus cloud type, the “low cloud” category in the Warren and Hahn Cloud Atlas includes synoptically defined low clouds (obscuring fog, stratus, stratocumulus, cumulus, and cumulonimbus), which may have cloud top heights above our 4km threshold. Therefore, it was not used for comparison to the RCCM climatology.

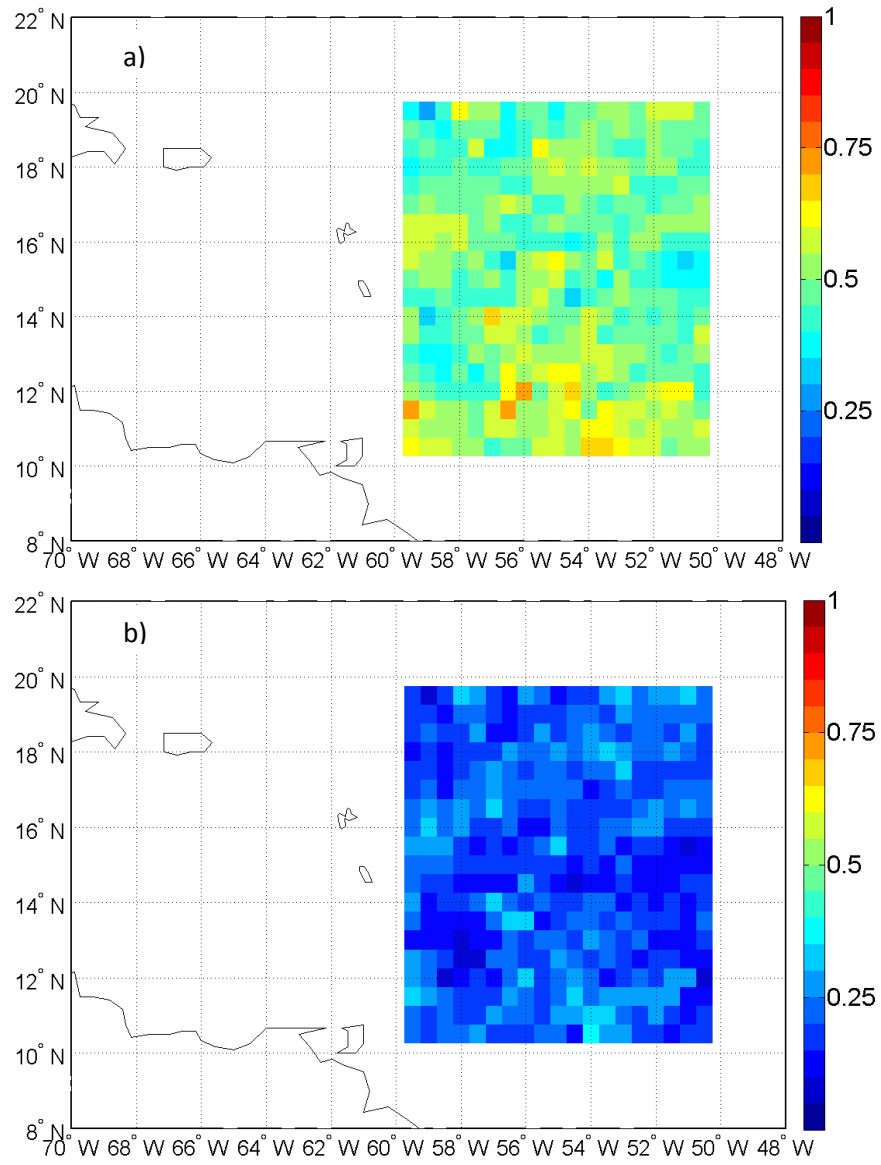


Figure 21. Sample December 10-year cloud fraction climatologies below cloud top height of 4km for a) the standard estimate of cloud fraction and b) the pattern recognition corrected cloud fractions.

6. Summary and Conclusions

This study is built on the previous work of Di Girolamo and Davies (1997) but uses larger, higher resolution satellite datasets to reexamine the biases in CF that are due to the resolution effect. In total 1405 15 m resolution ASTER scenes over cumulus-dominated regions including the Indian Ocean, Caribbean Sea, and Gulf of Mexico were collected. Cloud masks of these scenes were treated as “truth” and used to reexamine the resolution effect on CF calculated from a perfect, clear-conservative cloud mask, with a focus on how CF biases depend on sensor resolution and cloud distributions. At typical meteorological satellite resolution (~ 1 km) A_e , derived from perfect, clear-conservative masks, has a median bias of ~ 0.28 in our dataset. To obtain a median bias on the order of 0.01, without correction, the cloud mask would have to have a resolution < 80 m. The simple equation correction technique, A_{17} , given in Di Girolamo and Davies (1997), can correct the bias to 0.025 at 150 m resolution and the more computationally expensive pattern recognition technique can correct the median bias to 0.0 at any resolution tested and for any A_t tested. The efficacy of the pattern recognition technique is dependent on the availability of exploitable pattern information. As A_e approaches 1 the underlying distribution of cloud area cannot be distinguished. If A_t is known to be near 1 through a priori information the best choice of correction technique is A_{17} . However, for $A_t > 0.9$ the bias is not improved by much, so it may make more sense, computationally, not to correct the CF at all in regions dominated by $A_t > 0.9$, such as regions dominated by expansive, unbroken stratocumulus.

At 480 m resolution the degradation factor of the ASTER dataset is 32, which can be compared directly to the results of Di Girolamo and Davies (1997). They found that A_e bias ranged from an average of 0.32-0.35 depending on the dataset used. In the current study the average A_e bias is 0.18 for a degradation factor of 32. The peak in the ASTER distribution occurs at the lowest CF bin, 0.0-0.05, whereas it occurs at 0.75-0.8 for the AVHRR dataset used in Di Girolamo and Davies (1997). The overall average bias is dependant on the dataset’s distribution of A_t and the morphology of the clouds represented in each dataset.

Correction techniques provided in Di Girolamo and Davies (1997) were further applied to an operational cloud mask, the RCCM of the MISR instrument, whose observations are spatially and temporally coincident with ASTER. MISR’s RCCM was designed to flag cloudy any pixel that contains any amount of cloud. The RCCM performs similar enough to a perfect, clear-conservative cloud mask that its CF overestimation can be corrected in the same way. It is, however, subject to cloud detection issues involving thin cirrus and sun glint. For a more selectively filtered dataset the correction based on A_{17} is able to reduce the peak median bias to ~ 0.4 , but the bias still depends greatly on A_t and A_e . A_p suffers large negative biases where the training set’s distribution of A_t is sparse, but performs with a median bias of 0 for every value of A_e , which is more uniformly distributed than A_t . The A_p IQR of bias remains low but increases as A_e increases due to a decrease in pattern information available to distinguish the underlying spatial distribution of true cloud area.

Based on the performance of each correction technique, the pattern recognition technique was chosen to correct a sample CF climatology built from the RCCM in a region dominated by small trade wind cumulus. The results compared well with results from Zhao and Di Girolamo (2006) as well as long term surface observations (Warren and Hahn, 2007). The average CF over the $10^\circ \times 10^\circ$ domain was reduced from 0.496 to 0.199. This change in actual CF would have a huge impact on the climate system since an increase of low clouds by 4% can offset the doubling of CO_2 (Slingo, 1990). According to figure 1, for a cloud with an OD of 20 this would correspond to a net flux difference of $\sim 150 \text{ Wm}^{-2}$ at the TOA. Although the results are simplified to the average improvement, this technique does more than subtract a standard value to improve the climatology; it is a correction based on the spatial distribution of clouds in each $17.6 \text{ km} \times 17.6 \text{ km}$ region.

In reality, a perfect operational cloud mask does not exist. However, the results of this study demonstrated that CF biases can be reduced if the cloud mask behaves similarly to a perfect, clear-conservative one. Due to the dependence of CF overestimation on the spatial distribution of true cloud area a pattern recognition technique can be used to produce CF climatologies nearly unbiased by the resolution effect. Keeping in mind that a good training set is of great importance for the success of any pattern recognition technique, similar correction techniques can be implemented operationally by instrument teams to create their own corrected CF product. This could have a large impact on the already available cloud climatology products that are most popular with users. In the future a combination of higher resolution observations and clear-conservative cloud detection will maximize the number of clear pixels to perform clear-sky calculations on, minimize cloud contamination, and reduce CF bias due to the resolution effect.

In the future it may be beneficial to include a feature in the pattern vector that is not derived from the cloud mask but may help distinguish between the varied underlying distributions of true CF, especially for scenes with large A_c that do not have much pattern information to exploit. Some research on this was done in the early formulation of this thesis. In addition to developing a pattern recognition technique based on a feature vector derived from the cloud mask, a pattern recognition technique based on a feature vector derived from both the cloud mask and the radiance field was attempted. Rather than using radiance values directly they were converted to bidirectional reflectance factors (BRF) to normalize across values of solar zenith angle and incoming irradiance. Initial results showed little improvement over the technique described in this paper. The computation of the gray level co-occurrence matrix and the associated features was more expensive since the number of gray levels was increased from 2 to 16 and the BRF values had to be discretized over that range. Additionally, there was an increase in the number of features in the vector, which increased the amount of time it takes to compute the feature vector and the Euclidean distance.

7. Reference List

- Abrams, M. (2000), The Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER): data products for the high spatial resolution imager on NASA's Terra platform, *International Journal of Remote Sensing*, 21(5), 847-859.
- Ackerman, S. A., R. E. Holz, R. Frey, E. W. Eloranta, B. C. Maddux, and M. McGill (2008), Cloud detection with MODIS. Part II: Validation, *Journal of Atmospheric and Oceanic Technology*, 25(7), 1073-1086.
- Astin, Ivan, 1997: Sampling Errors and Bias in Satellite-Derived Fractional Cloud Cover Estimates from Exponential and Deterministic Cloud Fields as a Consequence of Instrument Pixel Size and Number. *J. Atmos. Oceanic Technol.*, **14**, 1146–1156.
- Arking, A., and J.D. Childs, (1985), Retrieval of Cloud Cover Parameters from Multispectral Satellite Images, *Journal of Climate and Applied Meteorology*, 24, 322-333.
- Bony, S., and J-L Dufresne, (2005), Marine boundary layer clouds at the heart of tropical cloud feedback uncertainties in climate models, *Geophys. Res. Lett.*, **32**, L20806, doi:10.1029/2005GL023851.
- Coakley, J. A., and F. P. Bretherton (1982), Cloud cover from high-resolution scanner data: detecting and allowing for partially filled fields of view, *Journal of Geophysical Research-Atmospheres*, 87(C7), 4917-4932.
- Davison, J. L., 2009: Radar based characterization of the tropical boundary layer. 34th Conf. on Radar Meteorology, Williamsburg, VA, Amer. Meteor. Soc.
- Dey, S., L. Di Girolamo, and G. Zhao (2008), Scale effect on statistics of the macrophysical properties of trade wind cumuli over the tropical western Atlantic during RICO, *Journal of Geophysical Research-Atmospheres*, 113.
- Di Girolamo, L., and R. Davies (1997), Cloud fraction errors caused by finite resolution measurements, *Journal of Geophysical Research D: Atmospheres*, 102(2), 1739-1756.
- Diner, D. J., et al (1998), Multi-angle Imaging SpectroRadiometer (MISR) Instrument Description and Experiment Overview, *IEEE Transactions on Geoscience and Remote Sensing*, 36(4), 1072-1085.
- Diner D. J., L. Di Girolamo, and E. Clothiaux, 1999: MISR level 1 cloud detection algorithm theoretical basis. Jet Propulsion Laboratory, JPL D-13397, Rev. B., 38 pp. [Available online at http://eosps.gsfc.nasa.gov/eos_homepage/for_scientists/atbd/docs/MISR/atbd-misr-06.pdf].
- Frouin, Robert, Beth Chertock, 1992: A Technique for Global Monitoring of Net Solar Irradiance at the Ocean Surface. Part I: Model. *J. Appl. Meteor.*, **31**, 1056–1066.
- Frouin, R., C. Gautier, and J.-J. Morcrette (1988), Downward Longwave Irradiance at the Ocean Surface From Satellite Data: Methodology and in Situ Validation, *J. Geophys. Res.*, 93(C1), 597–619.
- Gao, B-C, Y. J. Kaufman, D. Tanre, and R-R Li (2002) Distinguishing tropospheric aerosol from thin cirrus clouds, *Geophysical Research Letters*, 18, 1890.
- Geisser, S., (1975) The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70, 320-328.
- Goodman, A.H. and A. Henderson-Sellers, (1988), Cloud detection and analysis: a review of recent progress, *Atmospheric Research*, 21, 203-228.
- Henderson-Sellers, A., and K. McGuffie, (1990), Are Cloud Amounts estimated from satellite sensor and conventional surface-based observations related?, *Int. J. Remote Sensing*, 11, 543-550.

- Heymsfield, A. J., and G. M. McFarquhar (2001), Microphysics of INDOEX clean and polluted trade cumulus clouds, *J. Geophys. Res.*, 106, 28,653–28,673.
- Houghton, J., Y. Ding, D. J. Griggs, M. Noguer, P. J. van der Linden, X. Dai, K. Maskell, and C.A. Johnson, Eds., 2001: *Climate Change 2001: the Scientific Basis*. Cambridge University Press, 881 pp.
- Hu, M. K., Visual pattern recognition by moment invariants, *IEEE Trans. Inf. Theory*, IT-8, 179-187, 1962.
- Key, J., 2001: Streamer's User's guide, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin, 96 pp.
- Krijger, J. M., M. van Weele, I. Aben, and R. Frey (2007), Technical Note: The effect of sensor resolution on the number of cloud-free observations from space, *Atmospheric Chemistry and Physics*, 7(11), 2881-2891.
- Marchand, R., T. Ackerman, M. Smyth, and W. B. Rossow (2010), A review of cloud top height and optical depth histograms from MISR, ISCCP, and MODIS, *J. Geophys. Res.*, 115, D16206, doi:10.1029/2009JD013422.
- Ohring, George, Bruce Wielicki, Roy Spencer, Bill Emery, Raju Datla, 2005: Satellite Instrument Calibration for Measuring Global Climate Change: Report of a Workshop. *Bull. Amer. Meteor. Soc.*, 86, 1303–1313.
- Parrish, D. D., et al. (2009), Overview of the Second Texas Air Quality Study (TexAQS II) and the Gulf of Mexico Atmospheric Composition and Climate Study (GoMACCS), *J. Geophys. Res.*, 114, D00F13.
- Ramanathan, V., et al. (2001), Indian Ocean Experiment: An integrated analysis of the climate forcing and effects of the great Indo-Asian haze, *J. Geophys. Res.*, 106, 28,371–28,398.
- Randall, David, and Coauthors, 2003: Confronting Models with Data: The GEWEX Cloud Systems Study. *Bull. Amer. Meteor. Soc.*, **84**, 455–469.
- Rauber, R. M., et al. (2007) Rain in shallow cumulus over the ocean: the RICO campaign, *Bull. Am. Meteorol. Soc.*, 88, 1912-1928.
- Rossow, W.B., (1989), Measuring cloud properties from space: a review, *Journal of Climate*, 2, 201-213.
- Shenk, W. E., and V. V. Salomonson, A simulation study exploring the effects of sensor spatial resolution on estimates of cloud cover from satellites, *J. Appl. Meteorol.*, 11, 214-220, 1972.
- Slingo, A. (1990), Sensitivity of the earth's radiation budget to changes in low clouds, *Nature*, 343(6253), 49-51.
- Stephens, Graeme L., 2005: Cloud Feedbacks in the Climate System: A Critical Review. *J. Climate*, **18**, 237–273.
- Stephens, G. L., and D. Vane, 2007: Cloud remote sensing from space in the era of the A-Train. *J. Appl. Remote Sens.*, 1, 013507, doi:10.1117/1.0709703.
- Warren, S. G., and Hahn, C. J. (2007), Climatic Atlas of Clouds Over Land and Ocean, [available online: <http://www.atmos.washington.edu/~ignatius/CloudMap/WebO/index.html>]
- Weszka, J. S., C. R. Dyer, and A. Rosenfeld, A comparative study of texture measures for terrain classification, *IEEE Trans. Syst. Man. Cybern.*, SMC-6, 269-285, 1976.
- Wielicki, B. A., R. D. Cess, M. D. King, D. A. Randall, and E. F. Harrison, (1995) Mission to planet earth: role of clouds and radiation in climate, *Bull. Amer. Meteor. Soc.*, 76, 2125-2153.
- Wielicki, B. A. and Parker, L. (1992), On the Determination of Cloud Cover from Satellite Sensors: The Effect of Sensor Spatial Resolution, *Journal of Geophysical Research*, 97(D12), 12799-12823.

- Wielicki, B., and R. Welch (1986), Cumulus cloud properties derived using landsat satellite data, *Journal of Climate and Applied Meteorology*, 25(3), 261-276.
- Wilks, D. S. (2006), *Statistical Methods in the Atmospheric Sciences*, 2nd ed., pp 26-27, Elsevier Inc., Boston, MA.
- Yang, Y., and L. Di Girolamo (2008), Impacts of 3-D radiative effects on satellite cloud detection and their consequences on cloud fraction and aerosol optical depth retrievals, *J. Geophys. Res.*, 113, D04213, doi:10.1029/2007JD009095.
- Zhang, M. H., et al. (2005), Comparing clouds and their seasonal variations in 10 atmospheric general circulation models with satellite measurements, *J. Geophys. Res.*, 110, D15S02, doi:10.1029/2004JD005021.
- Zhao, G., and L. Di Girolamo (2006), Cloud fraction errors for trade wind cumuli from EOS-Terra instruments, *Geophysical Research Letters*, 33, L20802, doi: 10.1029/2006GL027088.
- Zhao, G., and L. Di Girolamo (2007), Statistics on the macrophysical properties of trade wind cumuli over the tropical western Atlantic, *Journal of Geophysical Research*, 112, D10204, doi: 10.1029/2006JD007371.
- Zhao, G., and L. Di Girolamo (2004), A Cloud Fraction versus View Angle Technique for Automatic In-Scene Evaluation of the MISR Cloud Mask, *Journal of Applied Meteorology*, 43(6), 860-869.
- Zhao, G., L. Di Girolamo, S. Dey, A. L. Jones, and M. Bull (2009), Examination of direct cumulus contamination on MISR-retrieved aerosol optical depth and angstrom coefficient over ocean, *Geophysical Research Letters*, 36.

Appendix A: Research Code

A.1 C-Shell Scripts

A.1.1 btrain97.csh

```
#!/bin/csh
#This script requires a number for input variable $1. It is equal to the number of
#15m pixels on a side to conglomerate into a single pixel. After each block of code
#is executed it combines the results into a single file that acts as the input to the
#pattern recognition code.

# The following block of code loops through the files containing cumulus clouds in
#one of three geographic regions and runs btrain97.exe for each. It then appends the
#output files to common files containing the data from all regions of that cloud
#type.
if ( -e totalbtraincu.txt ) then
    /bin/rm totalbtraincu.txt
endif

if ( -e filenamescu.txt ) then
    /bin/rm filenamescu.txt
endif

if ( -e badfilenamescu.txt ) then
    /bin/rm badfilenamescu.txt
endif

#foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/gomex/3N/4/*mask_4_.bin` )
#foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/india/3N/4/*mask.bin` )
foreach maskfile(`/bin/ls /home/manabe/gdi/sagnik/aster/rico/binary/4/*mask.bin` )

    echo $maskfile > btrain97.in
    echo $1      >> btrain97.in
    btrain97.exe < btrain97.in

    cat    TrainingSet.txt >> totalbtraincu.txt
    cat    name.txt        >> filenamescu.txt
    cat    badname.txt     >> badfilenamescu.txt

end

# The following block of code loops through the files containing cumulus clouds and
#cirrus clouds in one of three geographic regions and runs btrain97.exe for each. It
#then appends the output files to common files containing the data from all regions
#of that cloud type.
if ( -e totalbtraincuci.txt ) then
    /bin/rm totalbtraincuci.txt
endif

if ( -e filenamescuci.txt ) then
    /bin/rm filenamescuci.txt
endif
```

```

if ( -e badfilenamescuci.txt ) then
    /bin/rm badfilenamescuci.txt
endif

#foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/gomex/3N/5/*mask_5_.bin` )
#foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/india/3N/5/*mask.bin` )
foreach maskfile(`/bin/ls /home/manabe/gdi/sagnik/aster/rico/binary/5/*mask.bin` )

    echo $maskfile > btrain97.in
    echo $1      >> btrain97.in
    btrain97.exe < btrain97.in

    cat      TrainingSet.txt >> totalbtraincuci.txt
    cat      name.txt        >> filenamescuci.txt
    cat      badname.txt     >> badfilenamescuci.txt

end

# The following block of code loops through the files containing clouds other than
#cumulus or cumulus and cirrus in one of three geographic regions and runs
#btrain97.exe for each. It then appends the output files to common files containing
#the data from all regions of that cloud type.
if ( -e totalbtrainother.txt ) then
    /bin/rm totalbtrainother.txt
endif

if ( -e filenamesother.txt ) then
    /bin/rm filenamesother.txt
endif

if ( -e badfilenamesother.txt ) then
    /bin/rm badfilenamesother.txt
endif

#foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/gomex/3N/6/*mask_6_.bin` )
#foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/india/3N/6/*mask.bin` )
foreach maskfile(`/bin/ls /home/manabe/gdi/sagnik/aster/rico/binary/6/*mask.bin` )

    echo $maskfile > btrain97.in
    echo $1      >> btrain97.in
    btrain97.exe < btrain97.in

    cat      TrainingSet.txt >> totalbtrainother.txt
    cat      name.txt        >> filenamesother.txt
    cat      badname.txt     >> badfilenamesother.txt

end

cat totalbtraincu.txt    > totalbtrain_byres${1}new.txt
cat totalbtraincuci.txt >> totalbtrain_byres${1}new.txt
cat totalbtrainother.txt >> totalbtrain_byres${1}new.txt

```


A.1.2 gomex.csh

```
#!/bin/csh
```

```
#This file takes the steps necessary to coregister the ASTER and MISR grids for the  
#purpose of determining the true cloud fraction.
```

```
foreach binfile(`/bin/ls /home/manabe/gdi/aljones4/gomex/3N/[4-6]/*.bin`)  
    set hdfstring=(`echo $binfile | cut -d0 -f2- | cut -d_ -f1`)  
    set hdfhfile=(`echo /home/manabe/gdi/aljones4/gomex/hdf/*$hdfstring*.hdf`)  
    set orbit=`dump -h $hdfhfile | grep -A3 -i 'ORBIT' | sed -n 3P \  
        | cut -d= -f2 | cut -d\\ -f1`  
    set RCCMfile=(`ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \  
        input/binfiles/gomex/*$orbit*.bin`)  
  
    # This block of code extracts the latitude information from the ASTER hdf  
    #file, interpolates it to full resolution, and renames it.  
    cd /home/manabe-i2/a/aljones4/pattern_recog_MISR/output/lats  
    /home/glitter/b/gzhao1/bin/mread.exe $hdfhfile \  
        < /home/manabe-i2/a/aljones4/pattern_recog_MISR/input/mreadlat.in  
    /home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/interp.exe \  
        < /home/manabe-i2/a/aljones4/pattern_recog_MISR/input/interplat.in  
    mv -f LGgrid.bin /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \  
        output/lats/lat3N.bin  
  
    # This block of code extracts the longitude information from the ASTER hdf  
    #file, interpolates it to full resolution, and renames it  
    cd /home/manabe-i2/a/aljones4/pattern_recog_MISR/output/lons  
    /home/glitter/b/gzhao1/bin/mread.exe $hdfhfile \  
        < /home/manabe-i2/a/aljones4/pattern_recog_MISR/input/mreadlon.in  
    /home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/interp.exe \  
        < /home/manabe-i2/a/aljones4/pattern_recog_MISR/input/interplon.in  
    mv -f LGgrid.bin /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \  
        output/lons/lon3N.bin  
  
    # This block of code runs a program that creates files that assign each ASTER  
    #pixel to a MISR pixel.  
    cd /home/manabe-i2/a/aljones4/pattern_recog_MISR/output  
    /home/glitter/b/gzhao1/bin/misrcoordex $RCCMfile \  
        < /home/manabe-i2/a/aljones4/pattern_recog_MISR/input/misrcoordex.in  
    mv agp_line.bin /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \  
        output/gomex/line/${orbit}_${hdfstring}_line.bin  
    mv agp_sample.bin /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \  
        output/gomex/sample/${orbit}_${hdfstring}_sample.bin  
  
end
```

A.1.3 MISR_Blocks.csh

```
#!/bin/csh
```

#This code has four sections, one for each region and two for the RICO region. Each #section of code sets up the input file for MISR_blocks.exe and appends the output to #a common file and renames the output file. That common output file will become an #input for a program that calculates the feature vectors and associates them with a #true cloud fraction.

```
foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/india/3N/[4,6]/*.bin`)
  set hdfstring=(`echo $maskfile | cut -d/ -f9 | cut -d0 -f2- | cut -d_ -f1`)
  set linefile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
    output/india/line/*$hdfstring*.bin`)
  if(`echo $linefile | wc -c` > 75)then
    set IDstring=(`echo $linefile | cut -d/ -f10- | cut -d_ -f1-2`)
    set samplefile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/india/sample/$IDstring*.bin`)
    set block1=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/india/blocks/$IDstring*.txt | cut -c1-5`)
    set block2=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/india/blocks/$IDstring*.txt | cut -c6-11`)

    echo $maskfile      > MISR_blocks.in
    echo $linefile      >> MISR_blocks.in
    echo $samplefile    >> MISR_blocks.in
    echo $block1        >> MISR_blocks.in
    echo $block2        >> MISR_blocks.in

    /home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/MISR_blocks.exe \
      < MISR_blocks.in

    mv -f pixel_count.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/india/counts/${IDstring}_counts.txt
    cat total_count.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/india/counts/india_counts.txt
  endif
end
```

```
foreach maskfile(`/bin/ls /home/manabe/gdi/aljones4/gomex/3N/[4,6]/*.bin`)
  set hdfstring=(`echo $maskfile | cut -d/ -f9 | cut -d0 -f2- | cut -d_ -f1`)
  set linefile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
    output/gomex/line/*$hdfstring*.bin`)
  if(`echo $linefile | wc -c` > 75)then
    set IDstring=(`echo $linefile | cut -d/ -f10- | cut -d_ -f1-2`)
    set samplefile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/gomex/sample/$IDstring*.bin`)
    set block1=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/gomex/blocks/$IDstring*.txt | cut -c1-5`)
    set block2=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
      output/gomex/blocks/$IDstring*.txt | cut -c6-11`)

    echo $maskfile      > MISR_blocks.in
    echo $linefile      >> MISR_blocks.in
    echo $samplefile    >> MISR_blocks.in
```

```

echo $block1      >> MISR_blocks.in
echo $block2      >> MISR_blocks.in

/home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/MISR_blocks.exe \
< MISR_blocks.in

mv -f pixel_count.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/gomex/counts/${IDstring}_counts.txt
cat total_count.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/gomex/counts/gomex_counts.txt
endif
end

foreach maskfile(`/bin/ls /home/manabe/gdi/sagnik/aster/rico/binary/[4,6]/*.bin`)
set hdfstring=(`echo $maskfile | cut -d/ -f10 | cut -d0 -f2- | cut -d_ -f1`)
set linefile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnw/line/*$hdfstring*.bin`)
if(`echo $linefile | wc -c` > 75)then
set IDstring=(`echo $linefile | cut -d/ -f10- | cut -d_ -f1-2`)
set samplefile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnw/sample/*$IDstring*.bin`)
set block1=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnw/blocks/*$IDstring*.txt | cut -c1-5`)
set block2=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnw/blocks/*$IDstring*.txt | cut -c6-11`)
echo $block1 $block2

echo $maskfile      > MISR_blocks.in
echo $linefile      >> MISR_blocks.in
echo $samplefile    >> MISR_blocks.in
echo $block1        >> MISR_blocks.in
echo $block2        >> MISR_blocks.in

/home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/MISR_blocks.exe \
< MISR_blocks.in

mv -f pixel_count.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnw/counts/${IDstring}_counts.txt
cat total_count.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnw/counts/RICOnw_counts.txt
endif
end

foreach linefile(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOld/line/*.bin`)
set IDstring=(`echo $linefile | cut -d/ -f10- | cut -d_ -f1-2`)
set hdfstring=(`echo $IDstring | cut -d_ -f2`)
set maskfile=(`echo /home/glitter/b/gzhao1/cumulus/aster/clouds/ \
*$hdfstring*.bin`)
set samplefile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOld/sample/*$IDstring*.bin`)
set block1=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOld/blocks/*$IDstring*.txt | cut -c1-5`)

```

```

set block2=(`cat /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOold/blocks/*$IDstring*.txt | cut -c6-11`)

echo $maskfile      > MISR_blocks.in
echo $linefile      >> MISR_blocks.in
echo $samplefile    >> MISR_blocks.in
echo $block1        >> MISR_blocks.in
echo $block2        >> MISR_blocks.in

/home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/MISR_blocks.exe \
< MISR_blocks.in

mv -f pixel_count.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOold/counts/${IDstring}_counts.txt
cat total_count.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOold/counts/RICOold_counts.txt
end

```

A.1.4 goodmasks_totalpattern.csh

```

#!/bin/csh

# This script performs the filter to the RCCM dataset to retain only the best cloud
# masks as determined by the list in goodmasks.txt.

if ( -e ../output/RCCMgoodcu_totalTS.txt ) then
    /bin/rm ../output/RCCMgoodcu_totalTS.txt
endif

set j=1

while ($j <= `wc -l ../input/goodmasks.txt | cut -d" " -f1`)
    set zero1=(`cat ../input/goodmasks.txt | sed -n ${j}P | cut -c1`)
    set date1=(`cat ../input/goodmasks.txt | sed -n ${j}P | cut -c2-8`)
    set zero2=(`cat ../input/goodmasks.txt | sed -n ${j}P | cut -c9`)
    set time1=(`cat ../input/goodmasks.txt | sed -n ${j}P | cut -c10-15`)

    grep -Eh "$zero1?$date1.*$zero2?$time1\b"
    ../output/india/training/test2/*totalTS.txt
    ../output/gomex/training/test2/*totalTS.txt
    ../output/RICOnew/training/test2/*totalTS.txt >> ../output/RCCMgoodcu_totalTS.txt

    endif
    @ j++
end

```

A.1.5 RCCMbin_neighbor_noglint.csh

```

#!/bin/csh

```

#This code has four sections, one for each region and the RICO region is split into #two. Each section of code creates an input file for and runs #RCCM_bin97_neighbor_noglint.exe. It then appends to output to a file for each region #and renames the output file. The conglomerate files are the input for the pattern #recognition code.

```
foreach countsfile(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/india/counts/*_counts.txt`)
    set IDstring=`echo $countsfile | cut -d/ -f10- | cut -d_ -f1-2`
    set orbit=`echo $IDstring | cut -d_ -f1`
    set date=`echo $IDstring | cut -d_ -f2 | cut -c3-10`
    set tt=`echo $IDstring | cut -d_ -f2 | cut -c11-`
    set RCCMfilename=`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/binfiles/india/*$orbit*.bin`
    set classifier=`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/Classifier/training/india/*$orbit*.bin`
    set lines=`wc -l $countsfile | cut -d' ' -f1`

    echo $RCCMfilename > RCCMbin.in
    echo $countsfile >> RCCMbin.in
    echo $classifier >> RCCMbin.in
    echo 768 >> RCCMbin.in
    echo 2048 >> RCCMbin.in
    echo 80 >> RCCMbin.in
    echo $lines >> RCCMbin.in
    echo $orbit >> RCCMbin.in
    echo $date >> RCCMbin.in
    echo $tt >> RCCMbin.in

    /home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/ \
RCCM_bin97_neighbor_noglint.exe < RCCMbin.in
    cat RCCMbin.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/output/ \
india/training/test3/india_totalTS.txt

    mv -f RCCMbin.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/india/training/test3/${IDstring}_TS.txt
end
```

```
foreach countsfile(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/gomex/counts/*_counts.txt`)
    set IDstring=`echo $countsfile | cut -d/ -f10- | cut -d_ -f1-2`
    set orbit=`echo $IDstring | cut -d_ -f1`
    set date=`echo $IDstring | cut -d_ -f2 | cut -c3-10`
    set tt=`echo $IDstring | cut -d_ -f2 | cut -c11-`
    set RCCMfilename=`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/binfiles/gomex/*$orbit*.bin`
    set classifier=`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/Classifier/training/gomex/*$orbit*.bin`
    set lines=`wc -l $countsfile | cut -d' ' -f1`

    echo $RCCMfilename > RCCMbin.in
```

```

echo $countsfile >> RCCMbin.in
echo $classifier >> RCCMbin.in
echo 640 >> RCCMbin.in
echo 1152 >> RCCMbin.in
echo 63 >> RCCMbin.in
echo $lines >> RCCMbin.in
echo $orbit >> RCCMbin.in
echo $date >> RCCMbin.in
echo $tt >> RCCMbin.in

/home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/ \
RCCM_bin97_neighbor_noglint.exe < RCCMbin.in
cat RCCMbin.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/gomex/training/test3/gomex_totalTS.txt

mv -f RCCMbin.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/gomex/training/test3/${IDstring}_TS.txt
end

foreach countsfile(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnew/counts/*_counts.txt`)
set IDstring=(`echo $countsfile | cut -d/ -f10- | cut -d_ -f1-2`)
set orbit=(`echo $IDstring | cut -d_ -f1`)
set date=(`echo $IDstring | cut -d_ -f2 | cut -c3-10`)
set tt=(`echo $IDstring | cut -d_ -f2 | cut -c11-`)
set RCCMfilename=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/binfiles/RICOnew/*$orbit*.bin`)
set classifier=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/Classifier/training/RICOnew/*$orbit*.bin`)
set lines=(`wc -l $countsfile | cut -d' ' -f1`)

echo $RCCMfilename > RCCMbin.in
echo $countsfile >> RCCMbin.in
echo $classifier >> RCCMbin.in
echo 656 >> RCCMbin.in
echo 1280 >> RCCMbin.in
echo 74 >> RCCMbin.in
echo $lines >> RCCMbin.in
echo $orbit >> RCCMbin.in
echo $date >> RCCMbin.in
echo $tt >> RCCMbin.in

/home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/ \
RCCM_bin97_neighbor_noglint.exe < RCCMbin.in
cat RCCMbin.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOnew/training/test3/RICOnew_totalTS.txt

mv -f RCCMbin.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/output/ \
RICOnew/training/test3/${IDstring}_TS.txt
end

```

```

foreach countsfile(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOold/counts/*_counts.txt`)
  set IDstring=(`echo $countsfile | cut -d/ -f10- | cut -d_ -f1-2`)
  set orbit=(`echo $IDstring | cut -d_ -f1`)
  set date=(`echo $IDstring | cut -d_ -f2 | cut -c3-10`)
  set tt=(`echo $IDstring | cut -d_ -f2 | cut -c11-`)
  set RCCMfilename=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/binfiles/RICOold/*$orbit*.bin`)
  set classifier=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/input/ \
Classifier/training/RICOold/*$orbit*.bin`)
  set lines=(`wc -l $countsfile | cut -d' ' -f1`)

  echo $RCCMfilename > RCCMbin.in
  echo $countsfile >> RCCMbin.in
  echo $classifier >> RCCMbin.in
  echo 656 >> RCCMbin.in
  echo 1280 >> RCCMbin.in
  echo 72 >> RCCMbin.in
  echo $lines >> RCCMbin.in
  echo $orbit >> RCCMbin.in
  echo $date >> RCCMbin.in
  echo $tt >> RCCMbin.in

  /home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/ \
RCCM_bin97_neighbor_noglnt.exe < RCCMbin.in
  cat RCCMbin.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOold/training/test3/RICOold_totalTS.txt

  mv -f RCCMbin.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/RICOold/training/test3/${IDstring}_TS.txt
end

```

A.1.6 RCCMbin_climat_noglnt.csh

```

#!/bin/csh

# This script creates the input file required to run RCCM_bin97_climat_noglnt.exe
# then renames the output files and appends each file to a common file that is the
# input for the pattern recognition code.

foreach classifier(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/Classifier/Dec/*.bin`)

  set orbit=(`echo $classifier | cut -dP -f2 | cut -d_ -f2 | cut -c2-`)
  set pathnum=(`echo $classifier | cut -dP -f2 | cut -d_ -f1`)
  set year=(`/home/manabe-i2/a/gzhao1/Mtk-src-1.2.0/bin/ \
MtkOrbitToTimeRange $orbit | sed -n 1P | cut -d- -f1`)
  set month=(`/home/manabe-i2/a/gzhao1/Mtk-src-1.2.0/bin/ \
MtkOrbitToTimeRange $orbit | sed -n 1P | cut -d- -f2`)
  set day=(`/home/manabe-i2/a/gzhao1/Mtk-src-1.2.0/bin/ \
MtkOrbitToTimeRange $orbit | sed -n 1P | cut -d- -f3 | cut -dT -f1`)

```

```

set date=$year$month$day
set RCCMfilename=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/RCCMDec/*$orbit*.bin`)
set AGPlat=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/AGPbin/*$pathnum*lat.bin`)
set AGPlon=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/AGPbin/*$pathnum*lon.bin`)
set sfcfile=(`echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
input/sfcID/*$pathnum*.bin`)

echo $RCCMfilename > RCCMbin.in
echo $classifier >> RCCMbin.in
echo $AGPlat >> RCCMbin.in
echo $AGPlon >> RCCMbin.in
echo $sfcfile >> RCCMbin.in
echo 688 >> RCCMbin.in
echo 1536 >> RCCMbin.in
echo 73 >> RCCMbin.in
echo $date >> RCCMbin.in

/home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/ \
RCCM_bin97_climat_noglnt.exe < RCCMbin.in

cat RCCMbin.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/climat_totalTS.txt
cat RCCMbin_nopattern.txt >> /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/climat_nopattern_totalTS.txt
mv -f RCCMbin_nopattern.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/${date}_${orbit}_nopattern_TS.txt
mv -f RCCMbin.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/${date}_${orbit}_TS.txt

end

```

A.1.7 climatology_nonglnt.csh

```

#!/bin/csh

# This script sets up the input file for and runs climatology.exe, then renames and
# moves the outputs of the program.

foreach nocorfile(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/cloudfraction_climat_nocor2009Ae1.txt`)
set year=(`echo $nocorfile | cut -d_ -f6 | cut -dr -f2 | cut -c1-4`)
set corfile=(`/bin/ls /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/cloudfraction_climat${year}Ae1.txt`)
cat $corfile $nocorfile > /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/pattern_climat.txt
set lines=(`wc -l /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
output/climat_noglnt/pattern_climat.txt | cut -d' ' -f1`)

```



```

echo /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
    output/climat_noglint/pattern_climat.txt > climatology1.in
echo $lines >> climatology1.in
echo 20.0 >> climatology1.in
echo 10.0 >> climatology1.in
echo -60.0 >> climatology1.in
echo -50.0 >> climatology1.in
echo 20 >> climatology1.in
echo 20 >> climatology1.in

/home/manabe-i2/a/aljones4/pattern_recog_MISR/exec/climatology.exe \
    < climatology1.in

mv -f CF.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/output/ \
    climat_noglint/CF_${year}Ae1.txt
mv -f CF_corr.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
    output/climat_noglint/CF_corr_${year}Ae1.txt
mv -f counts.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
    output/climat_noglint/counts_${year}Ae1.txt
mv -f lats.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
    output/climat_noglint/lats.txt
mv -f lons.txt /home/manabe-i2/a/aljones4/pattern_recog_MISR/ \
    output/climat_noglint/lons.txt

end

```

A.2 FORTRAN Source Code

A.2.1 btrain97.f90

```

!-----
!This code develops the degraded ASTER dataset of feature vectors
!that will be separted into training and test sets in the pattern
!recognition code.
!
!Change the value of "numpixels" to change the resolution at which this code
!is performed.
!-----

program masktraining

implicit none
character(len=100)                :: asterfilename
integer(4)                        :: nline, ncol, &
    ierr,j,n,graylevels,i, hirescols, hiresrows, degradedcols, &    degradedrows
integer(1), allocatable, dimension(:,:) :: mask, mask1, odata1
real(8), allocatable, dimension(:,:)    :: HGLCM, VGLCM
integer(8)                            :: tallycld, tallyclr, &    numpixels,
tally2, tally1, ints, tallynr
real(8)                               :: At, Ae, Aedge, A17, &
    Ione,VAR,MEAN,ENT
integer(4),                dimension(2)    :: UL, LR

```

```

!=====
!   Input the files
!=====

print*, "input the aster mask filename"
read(*,'(1A)') asterfilename
read (*,'(I8)') numpixels

nline=4200
ncol=4980

!=====
!   Read in and process the masks
!=====

open(10, file=trim(asterfilename), action='read', form='binary', &
      access='direct', recl=1*ncol)
asterfilename=TRIM(asterfilename)
allocate(mask(1:ncol,1:nline), stat=ierr)
Do j=1, nline
    read(10,rec=j) (mask(i,j), i=1, ncol)
end do
close(10)

call find_rectangle(mask,ncol,nline,numpixels,UL,LR)

! if the results of "find_rectangle" are insufficient print the
!filename to a file and stop executing
IF(UL(1)<=numpixels .or. UL(2)<=numpixels+1 .or. LR(1)<=0 .or. &
    LR(2)<= 0 .or. UL(1)>LR(1) .or. UL(2) >LR(2))THEN
    open(90, file='badname.txt')
    write(90, '(1x, 1A)') asterfilename
    close(90)
    open(40, file='name.txt')
    write(40, '(1x, 1A)') "      "
    close(40)
    IF(allocated(mask))deallocate(mask)
    STOP
END IF

! Determine dimensions of rectangle and then remember a buffer of
!numpix on all sides
degradedcols=FLOOR((LR(1)-UL(1)+1)/(numpixels*1.0))
degradedrows=FLOOR((LR(2)-UL(2)+1)/(numpixels*1.0))
hirescols=degradedcols*numpixels
hiresrows=degradedrows*numpixels
allocate(odata1(1:hirescols+2*numpixels,1:hiresrows+2*numpixels))
call reliable_mask(UL, ncol, nline,mask, hirescols, &
    hiresrows,numpixels,odata1,tallycld,tallyclr, tallynr)
deallocate(mask)

! if there are no clouds in the domain or the domain is 100% cloud
!covered or the number of no-retrieval pixels is greater than 5% of
!the total then stop executing
IF(tallycld==0 .or. tallyclr==0 .or. tallynr > 0.05* &      (hirescols*hiresrows))THEN

```

```

        IF(allocated(odata1))deallocate(odata1)
        STOP
    END IF

!Calculate true cloud fraction
    At=tallycld*1.0/(tallycld+tallyclr)

allocate(mask1(1:degradedcols+2,1:degradedrows+2))
call degrade_mask(numpixels,hirescols, hiresrows, degradedcols, & degradedrows,
odata1, mask1,tally1,tally2)
IF(allocated(odata1))deallocate(odata1)

call find_int(degradedcols,degradedrows,mask1,ints)

!=====
!    Calculate the features
!=====
Ae=tally1*1.0/(tally1+tally2)
! if the degraded cloud mask returns no cloudy pixels write to file
!the ASTER filename and stop executing
IF(tally1==0)THEN
    open(90, file='badname.txt')
    write(90, '(1x, 1A)') asterfilename
    close(90)
    open(40, file='name.txt')
    write(40, '(1x, 1A)') "      "
    close(40)
    IF(allocated(mask1))deallocate(mask1)
    STOP
END IF

Aedge=(tally1-ints)*1.0/(tally1+tally2)
A17=(ints*1.0/(tally1+tally2)) + (1+(1.0/numpixels)**2) * Aedge/2.0
call calc_FirstMoment(mask1,degradedcols, degradedrows, Ione)

graylevels=2
allocate(HGLCM(1:graylevels,1:graylevels))
allocate(VGLCM(1:graylevels,1:graylevels))
call create_GLCMS(mask1,degradedcols, & degradedrows,graylevels,HGLCM,VGLCM)
call calc_GLDS(HGLCM,VGLCM,graylevels,VAR,MEAN,ENT)

!=====
!    PRINT features to file
!=====

open(40, file='name.txt')
write(40, '(1x, 1A)') asterfilename
close(40)

open(90, file='badname.txt')
write(90, '(1x, 1A)') "      "
close(90)

```

```

open(50, file='TrainingSet.txt')
write(50, '( 1A, 8F12.6)') asterfilename, At, Ae, A17, Aedge, Ione, &   VAR, MEAN,
ENT
close(50)
if(allocated(odata1)) deallocate(odata1)
if(allocated(mask)) deallocate(mask)
if(allocated(HGLCM)) deallocate(HGLCM)
if(allocated(VGLCM)) deallocate(VGLCM)

END PROGRAM masktraining

```

```

!-----
!
! SUBROUTINES
!
!-----
SUBROUTINE find_rectangle(mask,ncol,nline,umpix,UL,LR)
!determines the upperleft and lower right coordinates that maximize
!the rectangular area while excluding swath edges.
implicit none
integer*1, dimension(1:ncol, 1:nline), intent(in) :: mask
integer(4),                intent(in) :: nline, ncol
integer(8),                intent(in) :: umpix
integer(4),  dimension(1:2),    intent(out) :: UL, LR
integer(4)                :: i,j, m,n
integer(8)                :: area, temparea
integer(8),  dimension(1:4, 1:nline)    :: coords

coords=0
area=0
UL=0
LR=0

! Going through line-by-line and from left to right to find the Upper
!Left point for each line
Do j=1, nline
    DO i=1, ncol-1, 1
        IF (mask(i,j)>0 .and. mask(i+1,j)>0)THEN
            coords(1,j)=i
            coords(2,j)=j
            EXIT
        END IF
    END DO
END DO

!going through line by line and from right to left to find Lower Right
!point for each line
DO j=1, nline
    DO i=ncol, 2, -1
        IF (mask(i,j)>0 .and. mask(i-1,j)>0)THEN
            coords(3,j)=i
            coords(4,j)=j
            EXIT
        END IF
    END DO
END DO

```

```

        END IF
    END DO
END DO

DO m=1, nline
    IF(coords(3,m)==0 .or. coords(4,m)==0)CYCLE
    DO n=1, nline
        IF(coords(1,n)==0 .or. coords(2,n)==0)CYCLE
        temparea=(coords(4,m)-coords(2,n))*(coords(3,m) &
            -coords(1,n))
        IF(temparea>area)THEN
            area=temparea
            UL(1)=coords(1,n)+numpix
            UL(2)=coords(2,n)+1+numpix
            LR(1)=coords(3,m)-numpix
            LR(2)=coords(4,m)-1-numpix
        END IF
    END DO
END DO

END SUBROUTINE find_rectangle

SUBROUTINE relable_mask(UL,ncol, nline,mask, cols, &
    rows,numpix,odata,tally1,tally2, tally3)
! relabels pixels into 1 of three groups ( cloudy, clear, no-
! retrieval) and tallies how many of each there are

implicit none
integer(4),          dimension(1:2),  intent(in):: UL
integer(4),          intent(in):: ncol, nline, &  cols, rows
integer*1, dimension(1:ncol, 1:nline), intent(in):: mask
integer(8),          intent(in):: numpix
integer*1, dimension(1:cols+2*numpix,1:rows+2*numpix), &
    intent(out):: odata
integer*8,          intent(out):: tally1, &          tally2,tally3
integer(4)          :: i,j

odata=0
tally1=0
tally2=0
tally3=0

DO j=UL(2)-numpix, UL(2)+rows+numpix-1
    DO i=UL(1)-numpix, UL(1)+cols+numpix-1
        IF (mask(i,j)==1 .or. mask(i,j)==2) THEN
            odata(i+numpix-UL(1)+1,j+numpix-UL(2)+1)=1
            IF(j>=UL(2) .and. j<=UL(2)+rows+numpix-1 .and. i>=UL(1) &
                .and. i<=UL(1)+cols+numpix-1)THEN
                tally1=tally1+1
            END IF
        ELSE IF (mask(i,j)==3 .or. mask(i,j)==4) THEN
            odata(i+numpix-UL(1)+1,j+numpix-UL(2)+1)=0
            IF(j>=UL(2) .and. j<=UL(2)+rows+numpix-1 .and. i>=UL(1) &

```

```

                .and. i<=UL(1)+cols+numpy-1)THEN
            tally2=tally2+1
        END IF
    ELSE
        odata(i+numpy-UL(1)+1,j+numpy-UL(2)+1)=9
        IF(j>=UL(2) .and. j<=UL(2)+rows+numpy-1 .and. i>=UL(1) &
            .and. i<=UL(1)+cols+numpy-1)THEN
            tally3=tally3+1
        END IF
    END IF

END DO
END DO

END SUBROUTINE reliable_mask

SUBROUTINE degrade_mask(n,hirescols, hiresrows, degradedcols, &
    degradedrows, input, output, cnt1,cnt2)
!degrades mask to desired resolution and tallies how many clear and
!cloudy degraded pixels there are.

implicit none
integer(4), intent(in) :: hirescols,hiresrows,degradedcols, & degradedrows
integer(8), intent(in) :: n
integer(1), dimension(1:hirescols+2*n,1:hiresrows+2*n), &
    intent(in) :: input
integer(1), dimension(1:degradedcols+2,1:degradedrows+2), &
    intent(out) :: output
integer(8) intent(out) :: cnt1,cnt2
integer(4) :: I,J,m,l,clear

output=0
cnt1=0
cnt2=0

DO J=1, hiresrows+2*n, n
    Do I=1, hirescols+2*n, n
        Do m=0, n-1
            Do l=0, n-1
                clear=0
                IF (input(I+m,J+l)==1)EXIT
                clear=1
            END Do
            If (clear==0) EXIT
        END Do
        IF (clear==0) THEN
            output((i+n-1)/n, (j+n-1)/n)=1
            IF(i>n .and. i<=hirescols+n .and. j>n .and. &
                j<=hiresrows+n)THEN
                cnt1=cnt1+1
            END IF
        ELSE
            output((i+n-1)/n, (j+n-1)/n)=0

```

```

                IF(i>n .and. i<=hirescols+n .and. j>n .and. &
                    j<=hiresrows+n)THEN
                    cnt2=cnt2+1
                END IF
            END IF
        END DO
    END DO
END SUBROUTINE degrade_mask

```

SUBROUTINE find_int(degradedcols, degradedrows,input,tallyi)
!determines if each cloudy pixel is a cloud interior pixel and
!tallies how many are.

```

implicit none
integer(4),      intent(in)    :: degradedcols, degradedrows
integer(1), dimension(1:degradedcols+2, 1:(degradedrows+2)), &
    intent(in) :: input
integer(8),      intent(out) :: tallyi
integer(4)       :: I,J

tallyi=0

DO J=2, degradedrows+1
    DO I=2, degradedcols+1
        IF (input(I,J)==1 .and. input(I-1,J-1)==1 .and. &
            input(I,J-1)==1 .and. input(I+1,J-1)==1 .and. &
            input(I-1,J)==1 .and. input(I+1,J)==1 .and. &
            input(I-1,J+1)==1 .and. input(I,J+1)==1 .and. &
            input(I+1,J+1)==1)THEN
            tallyi=tallyi+1
        END IF
    END DO
END DO

END SUBROUTINE find_int

```

SUBROUTINE calc_FirstMoment(input,degradedcols, degradedrows,Ione)
! calculates the feature- first Hu invariant moment

```

implicit none
integer*4,      intent(in)    :: degradedcols, degradedrows
integer(1), dimension(1:degradedcols+2,1:degradedrows+2), &
    intent(in) :: input
real(8),       intent(out) :: Ione
integer(4)     :: i,j
integer(8)     :: Mone,Mzero,Mten,Mtwo,Mtwenty,Uzero
real(8)        :: xbar,ybar,Utwo,Utwenty,Ntwo,Ntwenty

```

```

Mzero=0
Mone=0

```

```

Mtwo=0
Mten=0
Mtwenty=0

!!!!!!!calculate raw moments!!!!!!!
DO i=2, degradedcols+1
    DO j=2, degradedrows+1
        Mzero=Mzero+input(i,j)
        Mone=Mone+(j-1)*input(i,j)
        Mtwo=Mtwo+input(i,j)*(j-1)**2
        Mten=Mten+(i-1)*input(i,j)
        Mtwenty=Mtwenty+input(i,j)*(i-1)**2
    END DO
END DO
xbar=Mten*1.0/Mzero
ybar=Mone*1.0/Mzero

!!!!!!!calculate central moments!!!!!!!
Uzero=Mzero*1.0
Utwo=Mtwo-(ybar*Mone*1.0)
Utwenty=Mtwenty-(xbar*Mten*1.0)

!!!!!!!calculate scale invariant moments!!!!!!
Ntwo=Utwo/(Uzero)**2
Ntwenty=Utwenty/(Uzero)**2

!!!!!!!calculate first moment!!!!!!!
Ione=Ntwo+Ntwenty

IF(isnan(Ione))THEN
STOP
END IF
END SUBROUTINE calc_FirstMoment

SUBROUTINE create_GLCMS(input, degradedcols, &
    degradedrows,gl,Houtput,Voutput)
! creates the gray-level co-occurrence matrix, both horizontal and
!vertical

implicit none
integer(4),          intent(in)  :: degradedcols, &
    degradedrows
integer*4,           intent(in)  :: gl
integer(1),  dimension(1:degradedcols+2,1:degradedrows+2), &
    intent(in)  :: input
real(8),  dimension(1:gl,1:gl),  intent(out) :: Houtput, Voutput
integer*4          :: i,j

Houtput=0
Voutput=0

DO i=2, degradedcols+1

```



```

DO j=2, degradedrows+1
  IF (j/=degradedrows+1) THEN
    Houtput(input(i,j)+1,input(i,j+1)+1)=
Houtput(input(i,j)+1,input(i,j+1)+1)+1
  END IF
  IF (i/=degradedcols+1) THEN
    Voutput(input(i,j)+1,input(i+1,j)+1)=
Voutput(input(i,j)+1,input(i+1,j)+1)+1
  END IF
END DO
END DO

!!!!!!normalize matrices!!!!!!
DO i=1, gl
  DO j=1, gl
    Houtput(i,j)=Houtput(i,j)/((degradedcols-1)*degradedrows)
    Voutput(i,j)=Voutput(i,j)/((degradedrows-1)*degradedcols)
  END DO
END DO

END SUBROUTINE create_GLCMS

```

```

SUBROUTINE calc_GLDS(Hinput,Vinput,gl,VAR,MEAN,ENT)
! calculates variance, mean and entropy from the gray-level co-
! occurrence matrices then averages them together to determine the
! variance, entropy and mean feature values.

```

```

implicit none
integer*4,          intent(in)  :: gl
real(8),    dimension(1:gl,1:gl), intent(in)  :: Hinput, Vinput
real(8),          intent(out)   :: VAR,MEAN,ENT
integer*4          :: i,j,n
real(8)            :: HVAR,VVAR,HMEAN, &
                   VMEAN,HENT,VENT

```

```

HVAR=0.0
HMEAN=0.0
HENT=0.0
VVAR=0.0
VMEAN=0.0
VENT=0.0

```

```

DO i=1, gl
  DO j=1, gl
    HVAR=HVAR+(Hinput(i,j))**2
    VVAR=VVAR+(Vinput(i,j))**2

    IF (Hinput(i,j)/=0) THEN
      HENT=HENT+Hinput(i,j)*log(Hinput(i,j))
    END IF
    IF (Vinput(i,j)/=0) THEN
      VENT=VENT+Vinput(i,j)*log(Vinput(i,j))
    END IF
  END DO
END DO

```

```

        END IF

        n=(i-j)**2
        HMEAN=HMEAN+n*Hinput(i,j)
        VMEAN=VMEAN+n*Vinput(i,j)
    END DO
END DO

```

```

HENT=-1.0*HENT
VENT=-1.0*VENT

```

```

VAR=(VVAR+HVAR)/2
MEAN=(VMEAN+HMEAN)/2
ENT=(HENT+VENT)/2

```

```

END SUBROUTINE calc_GLDS

```

A.2.2 interp.f90

```

!-----
!
! This code performs a linear interpolation of gidded data, assuming
! the input grid is evenly spaced and the input is of type double
! (64 bit precision)
!
!-----

```

```

program interp

```

```

!=====
!   GLOBAL VARIABLES
!=====

```

```

implicit none
Real(8), allocatable, dimension(:,:) :: ingrid
character(len=100) :: filename
integer(4) :: xin, yin, xout, yout, i, &
    j, m, n, x1, x2, y1, y2, XSubpix, YSubpix, x, y
Real(8), allocatable, dimension(:,:) :: outgrid

```

```

!=====
!   Input the files
!=====

```

```

!input the filename of the grid to be interpolated, x dimension, y dimension, desired
x dimension, desired y dimension
read(*,'(1A)') filename
read(*,'(I4)') xin
read(*,'(I4)') yin
read(*,'(I4)') xout
read(*,'(I4)') yout

```

```

allocate(ingrid(1:xin, 1:yin))
open(90, file=filename, action='read', form='binary', access='direct',
recl=xin*yin*8)

```

```

read(90,rec=1) ingrid
close(90)

!-----
!   Do the interpolation
!-----
allocate(outgrid(1:xout, 1:yout))
outgrid=0

XSubpix=xout/(xin-1)
YSubpix=yout/(yin-1)

Do i=1, xin-1
    x1=(i-1)*XSubpix+1
    x2=i*XSubpix+1
    Do j=1, yin-1
        y1=(j-1)*YSubpix+1
        y2=j*YSubpix+1
        Do m=1, XSubpix
            x=x1+m-1
            Do n=1, YSubpix
                y=y1+n-1

                outgrid(x,y)=((y-y1)*1.0/YSubpix) &
                    *((x2-x)*1.0*ingrid(i,j+1)/XSubpix &
                    +(x-x1) *ingrid(i+1,j+1)*1.0/XSubpix) &
                    +((y2-y)*1.0/YSubpix)*((x2-x) &
                    *ingrid(i,j) *1.0/XSubpix+(x-x1) &
                    *ingrid(i+1,j)*1.0/XSubpix)

            End Do
        End Do
    End Do
End Do

!-----
! Print new grid to file
!-----

open(20, file='LGgrid.bin', action='write', form='binary', &
    access='direct', recl=xout*yout*8)
write(20, rec=1) outgrid
close(20)

END PROGRAM interp

```

A.2.3 MISR_blocks.f90

```

!-----
! This code takes in ASTER mask file, corresponding line

```

```

! and sample files and the associated block numbers
! It outputs a file identifying ASTER filename, block #,
! sub-block position, total # of ASTER pixels in that
! sub-block and cloudy pixels in the sub-block.
!-----

program pixel_count

!=====
!   VARIABLES
!=====
implicit none
character(len=150)                :: maskfilename, &
    samplefilename, linefilename
integer(4)                        :: nline, ncol, i, j, ierr
integer(4), allocatable, dimension(:,:) :: block1_total, &
    block2_total, block1_cloudy, block2_cloudy
integer(4)                        :: blocknum1, blocknum2, &
    offset
integer(4), dimension(1:179)      :: offset_array
integer(1), allocatable, dimension(:,:) :: mask
integer(4), allocatable, dimension(:,:) :: sample, line

nline=4200
ncol=4980
!=====
!   Input the files
!=====
!-----read from brf.in---

read(*,'(1A)') maskfilename
read(*,'(1A)') linefilename
read(*,'(1A)') samplefilename
read(*,'(I3)') blocknum1
read(*,'(I3)') blocknum2

!=====
!   Read in files
!=====
open(10, file=trim(maskfilename), action='read', form='binary', &
    access='direct', recl=ncol*nline*1)
maskfilename=TRIM(maskfilename)
allocate(mask(1:ncol,1:nline), stat=ierr)
read(10,rec=1) mask
close(10)

open(20, file=trim(samplefilename), action='read', form='binary', &
    access='direct', recl=ncol*nline*4)
samplefilename=TRIM(samplefilename)
allocate(sample(1:ncol,1:nline), stat=ierr)
read(20,rec=1) sample
close(20)

```

```

open(30, file=trim(linefilename), action='read', form='binary', &
      access='direct', recl=ncol*nline*4)
linefilename=TRIM(linefilename)
allocate(line(1:ncol,1:nline), stat=ierr)
read(30,rec=1) line
close (30)

open(40, file='/home/manabe-i2/a/aljones4/pattern_recog_MISR/&
      input/offset.txt')
Do i=1,179
      read (40, FMT=104) (offset_array(i))
      104 FORMAT (I4)
END DO
close(40)

!-----
! Body of Code
!-----

!If ASTER scene is contained within one MISR block
IF (blocknum2 - blocknum1 .eq. 0) THEN
      offset=0
      allocate(block1_total(1:32,1:8))
      allocate(block1_cloudy(1:32,1:8))
      block1_total=0
      block1_cloudy=0

      DO i=1, ncol
            DO j=1, nline

                  ! if pixel is no retrieval move on to next pixel.
                  !it does not count towards the total number of
                  !pixels
                  IF (mask(i,j) .eq. 0)THEN
                        CYCLE
                  ELSE IF (mask(i,j)==1 .or. mask(i,j)==2)THEN
                        mask(i,j)=1
                  ELSE
                        mask(i,j)=0
                  END IF

                  !keep track of how many ASTER pixels fall within
                  !each block and how many of those are cloudy.
                  !each block is 16x16 pixels
                  block1_total(CEILING(sample(i,j)/16.0), &
                        CEILING(line(i,j)/16.0))=block1_total &
                        (CEILING(sample(i,j)/16.0),CEILING(line &
                        (i,j)/16.0)) + 1

                  block1_cloudy(CEILING(sample(i,j)/16.0), &
                        CEILING(line(i,j)/16.0))= block1_cloudy &
                        (CEILING(sample(i,j)/16.0),CEILING(line &
                        (i,j)/16.0)) + mask(i,j)

            END DO

      END DO

```

```

END DO

open (80, file="total_count.txt")
open (90, file="pixel_count.txt")

DO i=1,32
  DO j=1,8
    IF (block1_total(i,j) > 1307876)THEN
      write(90, '(5I12)') blocknum1, i, j, &
        block1_total(i,j), block1_cloudy(i,j)
      write(80, '(I12)') block1_total(i,j)
    END IF
  END DO
END DO
close(80)
close(90)

! If ASTER scene straddles 2 MISR blocks
ELSE
  offset=offset_array(blocknum1)
  allocate(block1_total(1:32,1:8))
  allocate(block1_cloudy(1:32,1:8))
  allocate(block2_total(1:32,1:8))
  allocate(block2_cloudy(1:32,1:8))
  block1_total=0
  block1_cloudy=0
  block2_total=0
  block2_cloudy=0

  DO i=1, ncol
    DO j=1, nline

      ! if pixel is no retrieval move on to next pixel
      IF (mask(i,j) .eq. 0)THEN
        CYCLE
      ELSE IF (mask(i,j)==1 .or. mask(i,j)==2)THEN
        mask(i,j)=1
      ELSE
        mask(i,j)=0
      END IF

      IF (line(i,j) <= 128)THEN
        ! it is in the upper block (block1)

        IF (offset > 0)THEN
          !the line/sample values are the same
          !relative to the block- so there is no offset

          offset=0
        END IF

        block1_total(CEILING((sample(i,j) &
          +offset)/16.0),CEILING(line(i,j)/16.0)) &
          =block1_total(CEILING((sample(i,j) &

```

```

        +offset)/16.0),CEILING(line(i,j)/16.0))+ 1

    block1_cloudy(CEILING((sample(i,j) &
        +offset)/16.0),CEILING(line(i,j)/16.0)) &
        =block1_cloudy(CEILING((sample(i,j) &
        +offset)/16.0),CEILING(line(i,j)/16.0)) &
        + mask(i,j)

ELSE
    ! it is in the lower block (block2)

    line(i,j)=line(i,j)-128

    IF (offset > 0)THEN
        ! we want to transform the offset so that we &
        ! can still use addition to get the proper &
        ! sample number

        offset=offset * -1
    ELSE
        offset=0
    END IF

    block2_total(CEILING((sample(i,j) &
        +offset)/16.0),CEILING(line(i,j)/16.0)) &
        =block2_total(CEILING((sample(i,j) &
        +offset)/16.0),CEILING(line(i,j)/16.0))+ 1

    block2_cloudy(CEILING((sample(i,j) &
        +offset)/16.0),CEILING(line(i,j)/16.0)) &
        =block2_cloudy(CEILING((sample(i,j) &
        +offset)/16.0),CEILING(line(i,j)/16.0)) &
        + mask(i,j)

END IF

END DO

END DO

open (80, file="total_count.txt")
open (90, file="pixel_count.txt")

DO i=1,32
    DO j=1,8
        IF (block1_total(i,j) > 1307876)THEN
            write(80, '(I12)') block1_total(i,j)
            write(90, '(5I12)') blocknum1, i, j, &
                block1_total(i,j), block1_cloudy(i,j)
        END IF

        IF (block2_total(i,j) > 1307876)THEN
            write(80, '(I12)') block2_total(i,j)
            write(90, '(5I12)') blocknum2, i, j, &
                block2_total(i,j), block2_cloudy(i,j)
        END IF
    
```

```

        END DO
    END DO

    close(80)
    close(90)

END IF

END PROGRAM pixel_count

```

A.2.4 RCCM_bin97_neighbor_noglint.f90

```

!-----
!This code develops the RCCM dataset that will be seperated into
!training and test sets by the pattern recognition program.
!It will not work properly near poles or anywhere the offset between
!blocks is positive
!-----

program masktraining

implicit none
character(len=150)                :: RCCMfilename, &
    countsfilename, Heightfilename, Glitterfilename
integer(4)                        :: nline, ncol, &
    ierr,RCCMcol, RCCMrow,i, lines, block, subblock_col, &
    subblock_row, total, cloudy, RCCMstart, graylevels, glitterflag
integer(1), allocatable, dimension(:,:) :: mask, subblock, &
    submask, glitter, subglitter
integer(2), allocatable, dimension(:,:) :: height
real(8), allocatable, dimension(:,:) :: HGLCM, VGLCM
integer(4)                        :: tallycld, tallyclr, &
    ints,orbit, date, time
integer(2)                        :: subblock_height
real(8)                          :: At, Ae, Aedge, A17, &
    Ione,VAR,MEAN,ENT
integer(4), dimension(1:179)      :: offset_array

!=====
!   Input the files
!=====
!-----read from .in file-----

read(*,'(1A)') RCCMfilename
read(*,'(1A)') countsfilename
read(*,'(1A)') Heightfilename
read(*,'(I4)') RCCMcol
read(*,'(I4)') RCCMrow
read(*,'(I4)') RCCMstart
read(*,'(I4)') lines

```



```

read(*,'(I)') orbit
read(*,'(I)') date
read(*,'(I)') time

!=====
! Read in files
!=====
open(10, file=trim(RCCMfilename), action='read', form='binary', &
      access='direct', recl=RCCMcol*RCCMrow*1)
RCCMfilename=TRIM(RCCMfilename)
allocate(mask(1:RCCMcol,1:RCCMrow), stat=ierr)
read(10,rec=1) mask
close(10)

open(30, file=trim(Heightfilename), action='read', form='binary', &
      access='direct', recl=(RCCMcol/16)*(RCCMrow/16)*2)
Heightfilename=TRIM(Heightfilename)
allocate(height(1:RCCMcol/16,1:RCCMrow/16), stat=ierr)
read(30,rec=1) height
close(30)

open(40, file='/home/manabe-i2/a/aljones4/pattern_recog_MISR/&
      input/offset.txt')
Do i=1,179
    read (40, FMT=104) (offset_array(i))
    104 FORMAT (I4)
END DO
close(40)

open(20, file=trim(countsfilename))
open(90, file='RCCMbin.txt')

DO i=1, lines
    read(20, '(5I12)') block, subblock_col, subblock_row, &
        total, cloudy
    At=cloudy*1.0/total
    If(At==0)THEN
        CYCLE
    END IF

    subblock_height=height(subblock_col+(RCCMstart+((RCCMrow/128)&
        -1)-block), subblock_row+((block-RCCMstart)*8))

    ! We only want to retain regions with valid heights below 4km
    IF(subblock_height>4000 .or. subblock_height<0)CYCLE

    allocate(subblock(1:18,1:18))
    CALL create_subblock(RCCMrow,RCCMcol,subblock_col, &
        subblock_row, RCCMstart,block,offset_array,mask,subblock)

    allocate(submask(1:18,1:18))
    CALL relabel_mask(subblock,tallycld, tallyclr,submask)

```

```

        deallocate(subblock)

        !----Calculate Features-----

        Ae=tallycld*1.0/(tallycld+tallyclr)

        ! We only want to process regions containing clouds
        IF(Ae==0)THEN
            deallocate(submask)
            CYCLE
        END IF

        CALL find_int(submask,ints)

        Aedge=(tallycld-ints)*1.0/(tallycld+tallyclr)

        A17=(ints*1.0/(tallycld+tallyclr)) + (1+(15/1100)**2) &
            * Aedge/2.0

        CALL calc_FirstMoment(submask,Ione)

        graylevels=2
        allocate(HGLCM(1:graylevels,1:graylevels))
        allocate(VGLCM(1:graylevels,1:graylevels))
        CALL create_GLCMS(submask,graylevels,HGLCM,VGLCM)

        CALL calc_GLDS(HGLCM,VGLCM,graylevels,VAR,MEAN,ENT)

        write(90, '( 5I12, 8F12.6, 3I10)') block, subblock_col, &
            subblock_row, cloudy, total, At, A17, Ae, Aedge, Ione, &
            VAR, MEAN, ENT, orbit, date, time

        deallocate(submask)
        deallocate (HGLCM)
        deallocate(VGLCM)

    END DO
    close(20)
    close(90)

    END PROGRAM masktraining

    !=====
    !   SUBROUTINES
    !=====

    SUBROUTINE create_subblock(totalrow,totalcol,subcol,subrow, &
        topblock,focus_block,offsets,input,output)

    implicit none
    integer(4), intent(in)                                :: totalrow, &
        totalcol, subcol, subrow, topblock, focus_block
    integer(1), dimension(1:totalcol,1:totalrow), intent(in) &
        :: input

```

```

integer(4), dimension(1:179), intent(in)      :: offsets
integer(1), dimension(1:18,1:18), intent(out) :: output
integer(4)                                     :: totaloffset, &
    endcol, endrow

```

```

totaloffset=sum(offsets(topblock:(focus_block-1)))
endcol=totalcol-((32-subcol)*16)+totaloffset
endrow=((focus_block-topblock)*128)+(subrow*16)
output(1:18,1:18)=input(endcol-16:endcol+1,endrow-16:endrow+1)

```

```

END SUBROUTINE create_subblock

```

```

SUBROUTINE relabel_mask(input1,cloud,clear,output_mask)
!This subroutine relables each pixel as either clear, cloudy, or no-
! retrieval and tallies the number of clear and cloudy pixels

```

```

implicit none
integer(1), dimension(1:18,1:18), intent(in)  :: input1
integer(1), dimension(1:18,1:18), intent(out) :: output_mask
integer(4),                                intent(out) :: cloud, clear
integer(4)                                    :: I,J

```

```

output_mask=0
cloud=0
clear=0

```

```

DO j=1, 18
    DO i=1,18

        IF (input1(i,j)==1 .or. input1(i,j)==2) THEN
            output_mask(i,j)=1
            IF (i==1 .or. i==18 .or. j==1 .or. j==18)CYCLE
            cloud=cloud+1
        ELSE IF (input1(i,j)==3 .or. input1(i,j)==4) THEN
            output_mask(i,j)=0
            IF (i==1 .or. i==18 .or. j==1 .or. j==18)CYCLE
            clear=clear+1
        ELSE
            output_mask(i,j)=9
        END IF
    END DO
END DO

```

```

END DO
END SUBROUTINE relabel_mask

```

```

SUBROUTINE find_int(input,tallyint)
! This subroutine determines if each cloudy pixel is a cloud interior
! pixel and tallies the number of them that are.

```

```

implicit none

```

```

integer(1), dimension(1:18,1:18), intent(in)  :: input
integer(4),                        intent(out) :: tallyint
integer(4)                          :: I,J

tallyint=0

DO J=2, 17
    DO I=2, 17
        IF (input(I,J)==1 .and. input(I-1,J-1)==1 .and. &
            input(I,J-1)==1 .and. input(I+1,J-1)==1 .and. &
            input(I-1,J)==1 .and. input(I+1,J)==1 .and. &
            input(I-1,J+1)==1 .and. input(I,J+1)==1 .and. &
            input(I+1,J+1)==1)THEN
            tallyint=tallyint+1
        END IF
    END DO
END DO

END SUBROUTINE find_int

```

```

SUBROUTINE calc_FirstMoment(input,Iout)
!This subroutine calculates the feature-Hu first moment invariant

```

```

implicit none
integer(1), dimension(1:18,1:18), intent(in)  :: input
real(8),                        intent(out) :: Iout
integer(4)                          :: i,j
integer(8)                          :: Mone,Mzero, &
    Mten,Mtwo,Mtwenty,Uzero
real(8)                              :: xbar,ybar,Utwo, &
    Utwenty,Ntwo,Ntwenty

```

```

Mzero=0
Mone=0
Mtwo=0
Mten=0
Mtwenty=0

```

```

!!!!!!!calculate raw moments!!!!!!!

```

```

DO i=2, 17
    DO j=2, 17
        Mzero=Mzero+input(i,j)
        Mone=Mone+(j-1)*input(i,j)
        Mtwo=Mtwo+input(i,j)*(j-1)**2
        Mten=Mten+(i-1)*input(i,j)
        Mtwenty=Mtwenty+input(i,j)*(i-1)**2
    END DO
END DO
xbar=Mten*1.0/Mzero
ybar=Mone*1.0/Mzero

```

```

!!!!!!!calculate central moments!!!!!!!

```

```

Uzero=Mzero*1.0

```

```

Utwo=Mtwo-(ybar*Mone*1.0)
Utwenty=Mtwenty-(xbar*Mten*1.0)

!!!!!!!calculate scale invariant moments!!!!!!
Ntwo=Utwo/(Uzero)**2
Ntwenty=Utwenty/(Uzero)**2

!!!!!!!calculate first moment!!!!!!!
Iout=Ntwo+Ntwenty

END SUBROUTINE calc_FirstMoment


SUBROUTINE create_GLCMS(input,gl,Hout,Vout)
!This subroutine creates the horizontal and vertical gray-level co-
! occurrence matrices

implicit none
integer*4,          intent(in)    :: gl
integer(1),  dimension(1:18,1:18), intent(in)  :: input
real  (8),  dimension(1:gl,1:gl), intent(out) :: Hout, Vout
integer*4          :: i,j

Hout=0
Vout=0

DO i=2, 17
    DO j=2,17
        IF (j/=17)THEN
            Hout(input(i,j)+1,input(i,j+1)+1)= &
                Hout(input(i,j)+1,input(i,j+1)+1)+1
        END IF
        IF (i/=17) THEN
            Vout(input(i,j)+1,input(i+1,j)+1)= &
                Vout(input(i,j)+1,input(i+1,j)+1)+1
        END IF
    END DO
END DO

!!!!!!!normalize matrices!!!!!!!
DO i=1, gl
    DO j=1, gl
        Hout(i,j)=Hout(i,j)/((16-1)*16)
        Vout(i,j)=Vout(i,j)/((16-1)*16)
    END DO
END DO

END SUBROUTINE create_GLCMS


SUBROUTINE calc_GLDS(Hin,Vin,gl,VAR,MEAN,ENT)
!This subroutine calculates horizontal and vertical variance, mean

```

! and entropy from the GLCMs then averages them to create the
! features, variance, mean, and entropy.

```

implicit none
integer*4,          intent(in)  :: gl
real(8),    dimension(1:gl,1:gl), intent(in) :: Hin, Vin
real(8),          intent(out)   :: VAR,MEAN,ENT
integer*4          :: i,j,n
real(8)            :: HVAR,VVAR,HMEAN, &
                   VMEAN,HENT,VENT

HVAR=0.0
HMEAN=0.0
HENT=0.0
VVAR=0.0
VMEAN=0.0
VENT=0.0

DO i=1, gl
  DO j=1, gl
    HVAR=HVAR+(Hin(i,j))**2
    VVAR=VVAR+(Vin(i,j))**2

    IF (Hin(i,j)/=0) THEN
      HENT=HENT+Hin(i,j)*log(Hin(i,j))
    END IF

    IF (Vin(i,j)/=0) THEN
      VENT=VENT+Vin(i,j)*log(Vin(i,j))
    END IF

    n=(i-j)**2
    HMEAN=HMEAN+n*Hin(i,j)
    VMEAN=VMEAN+n*Vin(i,j)
  END DO
END DO

HENT=-1.0*HENT
VENT=-1.0*VENT

VAR=(VVAR+HVAR)/2
MEAN=(VMEAN+HMEAN)/2
ENT=(HENT+VENT)/2

END SUBROUTINE calc_GLDS

```

A.2.5 RCCM_bin97_climat_noglint.f90

!-----
!This code develops the RCCM training set for the pattern recognition !program used
for the climatology.
!It will not work properly near poles or anywhere the offset between

```

!blocks is positive
!-----

program masktraining

implicit none
character(len=150)                :: RCCMfilename, &
    Heightfilename, LATfilename, LONfilename, SFCfilename
integer(4)                        :: ierr,RCCMcol, &
    RCCMrow,i, subblock_col, subblock_row, total, cloudy, &
    RCCMstart,graylevels, block
integer(1), allocatable, dimension(:,:) :: mask, subblock, &
    submask, sfcflag, subblock_sfc
integer(4), dimension(1:2,1:2)      :: corners
integer(2), allocatable, dimension(:,:) :: height
integer(2)                          :: subblock_height
real(8), allocatable, dimension(:,:) :: HGLCM, VGLCM
real(4), allocatable, dimension(:,:) :: LAT, LON
real(4)                             :: centerLAT, centerLON
integer(4)                          :: tallycld, &
    tallyclr,ints,date,tallysfc
real(8)                             :: Ae, Aedge, A17, &
    Ione,VAR,MEAN,ENT
integer(4), dimension(1:179)        :: offset_array

!=====
!   Input the files
!=====

!=====
!-----read from .in file-----
!=====
read(*,'(1A)') RCCMfilename
read(*,'(1A)') Heightfilename
read(*,'(1A)') LATfilename
read(*,'(1A)') LONfilename
read(*,'(1A)') SFCfilename
read(*,'(I4)') RCCMcol
read(*,'(I4)') RCCMrow
read(*,'(I4)') RCCMstart
read(*,'(I)') date

!=====
!   Read in files
!=====
open(10, file=trim(RCCMfilename), action='read', form='binary', &
    access='direct', recl=RCCMcol*RCCMrow*1)
RCCMfilename=TRIM(RCCMfilename)
allocate(mask(1:RCCMcol,1:RCCMrow), stat=ierr)
read(10,rec=1) mask
close(10)

open(20, file=trim(Heightfilename), action='read', form='binary', &
    access='direct', recl=(RCCMcol/16)*(RCCMrow/16)*2)
Heightfilename=TRIM(Heightfilename)

```

```

allocate(height(1:RCCMcol/16,1:RCCMrow/16), stat=ierr)
read(20,rec=1) height
close(20)

open(30, file=trim(SFCfilename), action='read', form='binary', &
      access='direct', recl=RCCMcol*RCCMrow*1)
SFCfilename=TRIM(SFCfilename)
allocate(sfcflag(1:RCCMcol,1:RCCMrow), stat=ierr)
read(30,rec=1) sfcflag
close(30)

open(40, file='/home/manabe-i2/a/aljones4/pattern_recog_MISR&
      /input/offset.txt')
Do i=1,179
    read (40, FMT=104) (offset_array(i)
    104 FORMAT (I4)
END DO
close(40)

open(50, file=trim(LATfilename), action='read', form='binary', &
      access='direct', recl=RCCMcol*RCCMrow*4)
LATfilename=TRIM(LATfilename)
allocate(LAT(1:RCCMcol,1:RCCMrow), stat=ierr)
read(50,rec=1) LAT
close(50)

open(60, file=trim(LONfilename), action='read', form='binary', &
      access='direct', recl=RCCMcol*RCCMrow*4)
LONfilename=TRIM(LONfilename)
allocate(LON(1:RCCMcol,1:RCCMrow), stat=ierr)
read(60,rec=1) LON
close(60)

open(90, file='RCCMbin.txt')
open(80, file='RCCMbin_nopattern.txt')

DO block=74,83
    DO subblock_col=7, 25
        DO subblock_row=1,8

            ! We only want to retain scenes with cloud top height
            ! below 4km
            IF(subblock_height > 4000 )CYCLE
            CALL create_subblock(RCCMcol,subblock_col, &
                                subblock_row,RCCMstart,block,&
                                offset_array,cornerRadius)

            allocate(subblock_sfc(1:16,1:16))
            subblock_sfc(1:16,1:16)=sfcflag(cornerRadius(2,1)&
                                :cornerRadius(2,2),cornerRadius(1,1):cornerRadius(1,2))

            allocate(subblock(1:18,1:18))

```



```

subblock(1:18,1:18)=mask(corners(2,1)-1:corners(2,2)&
                        +1,corners(1,1)-1:corners(1,2)+1)

allocate(submask(1:18,1:18))
CALL relabel_mask(subblock,subblock_sfc,tallycld, &
                  tallyclr,tallysfc,submask)
deallocate(subblock)
deallocate(subblock_sfc)

      ! We only want to retain scenes over ocean
      IF(tallysfc > 0 )THEN
          deallocate(submask)
          CYCLE
      END IF

centerLAT=SUM(LAT(corners(2,2)-8:corners(2,2) &
                 -7,corners(1,2)-8:corners(1,2)-7))/4
centerLON=SUM(LON(corners(2,2)-8:corners(2,2) &
                 -7,corners(1,2)-8:corners(1,2)-7))/4

      !-----
      ! Calculate Features
      !-----

Ae=tallycld*1.0/(tallycld+tallyclr)

      ! If the estimated cloud fraction is 0 write the
      !   information to a file that will not go through
      !   pattern recognition
      IF(Ae==0)THEN
          write(80, '(1I12, 2F12.6, 1I12, 2F12.6)') &
              date, centerLAT, centerLON, &
              subblock_height, Ae, 0
          deallocate(submask)
          CYCLE
      END IF

      !We only want to retain scenes for which we know the
      !   cloud type height
      IF(subblock_height < 0)THEN
          deallocate(submask)
          CYCLE
      END IF

CALL find_int(submask,ints)
Aedge=(tallycld-ints)*1.0/(tallycld+tallyclr)

      ! If the estimated cloud fraction is 1 write the
      !   information to a file that will not go through
      !   pattern recognition
      IF(Ae==1)THEN
          write(80, '(1I12, 2F12.6, 1I12, 2F12.6)') date, &
              centerLAT, centerLON, subblock_height, &
              Ae, Aedge
          deallocate(submask)

```

```

        CYCLE
    END IF

    A17=(ints*1.0/(tallycld+tallyclr)) &
        + (1+(15/1100)**2) * Aedge/2.0

    CALL calc_FirstMoment(submask,Ione)

    graylevels=2
    allocate(HGLCM(1:graylevels,1:graylevels))
    allocate(VGLCM(1:graylevels,1:graylevels))
    CALL create_GLCMS(submask,graylevels,HGLCM,VGLCM)

    CALL calc_GLDS(HGLCM,VGLCM,graylevels,VAR,MEAN,ENT)

    PRINT*, 'got through calcs'

    write(90, '( 1I12, 2F12.6, 1I12, 7F12.6)') date, &
        centerLAT, centerLON, subblock_height, A17, &
        Ae, Aedge, Ione, VAR, MEAN, ENT

    deallocate(submask)
    deallocate (HGLCM)
    deallocate(VGLCM)
END DO
END DO
close(90)
close(80)

END PROGRAM masktraining

!=====
!   SUBROUTINES
!=====

SUBROUTINE create_subblock(totalcol,subcol,subrow,topblock, &
    focus_block,offsets,output)

implicit none
integer(4), intent(in)                :: totalcol, subcol, &
    subrow, topblock, focus_block
integer(4), dimension(1:179), intent(in) :: offsets
integer(4), dimension(1:2,1:2), intent(out) :: output
integer(4)                                :: totaloffset, &
    endcol, endrow

totaloffset=sum(offsets(topblock:(focus_block-1)))
endcol=totalcol-((32-subcol)*16)+totaloffset
endrow=((focus_block-topblock)*128)+(subrow*16)

output(1,1)=endrow-15
output(1,2)=endrow
output(2,1)=endcol-15
output(2,2)=endcol

```

```
END SUBROUTINE create_subblock
```

```
SUBROUTINE relabel_mask(input_mask,input_sfc, cloud,clear,sfc, &
    output)
!This subroutine relables each pixel as either clear, cloudy, or no-
! retrieval and tallies the number of clear and cloudy pixels
```

```
implicit none
integer(1), dimension(1:18,1:18), intent(in)    :: input_mask
integer(1), dimension(1:16,1:16), intent(in)    :: input_sfc
integer(1), dimension(1:18,1:18), intent(out)   :: output
integer(4),                                intent(out) :: cloud, clear,sfc
integer(4)                                :: I,J

output=0
cloud=0
clear=0
sfc=0

DO j=1, 18
    DO i=1,18
        IF(i>1 .and. i<18 .and. j>1 .and. j<18)THEN
            IF(input_sfc(i-1,j-1)<6 .and. &
                input_sfc(i-1,j-1)>0)THEN
                sfc=sfc+1
            END IF
        END IF

        IF (input_mask(i,j)==1 .or. input_mask(i,j)==2) THEN
            output(i,j)=1
            IF (i==1 .or. i==18 .or. j==1 .or. j==18)CYCLE
            cloud=cloud+1

        ELSE IF (input_mask(i,j)==3 .or. &
            input_mask(i,j)==4) THEN
            output(i,j)=0
            IF (i==1 .or. i==18 .or. j==1 .or. j==18)CYCLE
            clear=clear+1

        ELSE
            output(i,j)=9

        END IF

    END DO
END DO

END SUBROUTINE relabel_mask
```

```
SUBROUTINE find_int(input,tallyint)
```

```
! This subroutine determines if each cloudy pixel is a cloud interior
! pixel and tallies the number of them that are.
```

```
implicit none
integer(1), dimension(1:18,1:18), intent(in)      :: input
integer(4),                        intent(out)     :: tallyint
integer(4)                          :: I,J

tallyint=0

DO J=2, 17
  DO I=2, 17
    IF (input(I,J)==1 .and. input(I-1,J-1)==1 .and. &
        input(I,J-1)==1 .and. input(I+1,J-1)==1 .and. &
        input(I-1,J)==1 .and. input(I+1,J)==1 .and. &
        input(I-1,J+1)==1 .and. input(I,J+1)==1 .and. &
        input(I+1,J+1)==1)THEN
      tallyint=tallyint+1
    END IF
  END DO
END DO

END SUBROUTINE find_int
```

```
SUBROUTINE calc_FirstMoment(input,Iout)
!This subroutine calculates the feature-Hu first moment invariant
```

```
implicit none
integer(1), dimension(1:18,1:18), intent(in)      :: input
real(8),                        intent(out)         :: Iout
integer(4)                          :: i,j
integer(8)                          :: Mone,Mzero, &
Mten,Mtwo,Mtwenty,Uzero
real(8)                              :: xbar,ybar, &
Utwo,Utwenty,Ntwo,Ntwenty
```

```
Mzero=0
Mone=0
Mtwo=0
Mten=0
Mtwenty=0
```

```
!!!!!!!calculate raw moments!!!!!!!!!!!!!!
```

```
DO i=2, 17
  DO j=2, 17
    Mzero=Mzero+input(i,j)
    Mone=Mone+(j-1)*input(i,j)
    Mtwo=Mtwo+input(i,j)*(j-1)**2
    Mten=Mten+(i-1)*input(i,j)
    Mtwenty=Mtwenty+input(i,j)*(i-1)**2
  END DO
END DO
xbar=Mten*1.0/Mzero
```

```

ybar=Mone*1.0/Mzero

!!!!!!!calculate central moments!!!!!!!
Uzero=Mzero*1.0
Utwo=Mtwo-(ybar*Mone*1.0)
Utwenty=Mtwenty-(xbar*Mten*1.0)

!!!!!!!calculate scale invariant moments!!!!!!
Ntwo=Utwo/(Uzero)**2
Ntwenty=Utwenty/(Uzero)**2

!!!!!!!calculate first moment!!!!!!!
Iout=Ntwo+Ntwenty

END SUBROUTINE calc_FirstMoment


SUBROUTINE create_GLCMS(input,gl,Hout,Vout)
!This subroutine creates the horizontal and vertical gray-level co-
! occurrence matrices

implicit none
integer*4,          intent(in)    :: gl
integer(1), dimension(1:18,1:18), intent(in)  :: input
real    (8), dimension(1:gl,1:gl), intent(out) :: Hout, Vout
integer*4           :: i,j

Hout=0
Vout=0

DO i=2, 17
    DO j=2,17
        IF (j/=17)THEN
            Hout(input(i,j)+1,input(i,j+1)+1)= &
                Hout(input(i,j)+1,input(i,j+1)+1)+1
        END IF

        IF (i/=17) THEN
            Vout(input(i,j)+1,input(i+1,j)+1)= &
                Vout(input(i,j)+1,input(i+1,j)+1)+1
        END IF
    END DO
END DO

!!!!!!!normalize matrices!!!!!!!
DO i=1, gl
    DO j=1, gl
        Hout(i,j)=Hout(i,j)/((16-1)*16)
        Vout(i,j)=Vout(i,j)/((16-1)*16)
    END DO
END DO

END SUBROUTINE create_GLCMS

```

```

SUBROUTINE calc_GLDS(Hin,Vin,gl,VAR,MEAN,ENT)
!This subroutine calculates horizontal and vertical variance, mean
! and entropy from the GLCMs then averages them to create the
! features, variance, mean, and entropy.

implicit none
integer*4,          intent(in)  :: gl
real(8),    dimension(1:gl,1:gl), intent(in) :: Hin, Vin
real(8),          intent(out)   :: VAR,MEAN,ENT
integer*4          :: i,j,n
real(8)           :: HVAR,VVAR,HMEAN, &
                   VMEAN,HENT,VENT

HVAR=0.0
HMEAN=0.0
HENT=0.0
VVAR=0.0
VMEAN=0.0
VENT=0.0

DO i=1, gl
  DO j=1, gl
    HVAR=HVAR+(Hin(i,j))**2
    VVAR=VVAR+(Vin(i,j))**2

    IF (Hin(i,j)/=0) THEN
      HENT=HENT+Hin(i,j)*log(Hin(i,j))
    END IF

    IF (Vin(i,j)/=0) THEN
      VENT=VENT+Vin(i,j)*log(Vin(i,j))
    END IF

    n=(i-j)**2
    HMEAN=HMEAN+n*Hin(i,j)
    VMEAN=VMEAN+n*Vin(i,j)
  END DO
END DO

HENT=-1.0*HENT
VENT=-1.0*VENT

VAR=(VVAR+HVAR)/2
MEAN=(VMEAN+HMEAN)/2
ENT=(HENT+VENT)/2

END SUBROUTINE calc_GLDS

```

A.2.6 climatology.f90

PROGRAM Climatology

! This program takes the results from the pattern recognition of the
! regions included in the climatology and averages the results to a
! lat/lon grid. The outputs are a grid containing the corrected cloud
! fractions, a grid containing the uncorrected cloud fractions, the
! lat/lon grids, and the number of regions included in the average at
! each grid point.

```
!-----
!   Variables
!-----
implicit none
integer(4)                                :: i,j, xdim,ydim, &
    grid_col, grid_row
integer(8)                                :: date, lines
integer(4), allocatable, dimension(:,:) :: counts
real(8), allocatable, dimension(:)       :: lats,lons
real(8), allocatable, dimension(:,:)    :: CF,CF_corr
real(8)                                   :: lat, lon, upper_lon, &
    lower_lon, upper_lat, lower_lat, Ae, Ap, A17
character(len=150)                       :: patternrecogfile

!-----
!   Read from .in file
!-----
read(*, '(1A)') patternrecogfile
read(*, '(1I)') lines
read(*, '(1F)') upper_lat
read(*, '(1F)') lower_lat
read(*, '(1F)') upper_lon
read(*, '(1F)') lower_lon
read(*, '(1I)') xdim
read(*, '(1I)') ydim

open(20, file=trim(patternrecogfile))

!-----
!   Create grid
!-----
allocate(lons(1:xdim))
allocate(lats(1:ydim))

CALL create_grid(xdim, upper_lon, lower_lon, lons)

CALL create_grid(ydim, lower_lat, upper_lat, lats)
```

```

allocate(counts(1:xdim, 1:ydim))
counts=0

allocate(CF(1:xdim,1:ydim))
CF=0

allocate(CF_corr(1:xdim,1:ydim))
CF_corr=0

DO i=1, lines
    read(20, '(1I10, 5F11.6)') date, lat, lon, Ae, Ap, A17

    !-----
    !  locate grid box
    !-----

    IF(lat > upper_lat .or. lat < lower_lat .or. lon > lower_lon &
        .or. lon < upper_lon)CYCLE

    CALL find_grid_box(lats,ydim, lons,xdim, lat, lon, grid_col, &
        grid_row)

    counts(grid_col, grid_row)=counts(grid_col, grid_row)+1

    CF_corr(grid_col,grid_row)=(CF_corr(grid_col,grid_row) &
        *(counts(grid_col, grid_row)-1)+Ap) &
        /counts(grid_col,grid_row)

    CF(grid_col,grid_row)=(CF(grid_col,grid_row) &
        *(counts(grid_col, grid_row)-1)+Ae) &
        /counts(grid_col,grid_row)

END DO

close(20)

!-----
!  write to files
!-----
open(70, file='CF.txt' )
open(80, file='CF_corr.txt' )
open(90, file='counts.txt' )
open(100, file='lats.txt')
open(110, file='lons.txt')

DO j=1,ydim
    write(70, '(20F12.6)') CF(1:xdim,j)
    write(80, '(20F12.6)') CF_corr(1:xdim,j)
    write(90, '(20I12)') counts(1:xdim,j)
    write(100, '(20F12.6)') lats(j), lats(j),lats(j), lats(j), &
        lats(j), lats(j),lats(j), lats(j),lats(j), lats(j), &
        lats(j), lats(j),lats(j), lats(j),lats(j), lats(j), &
        lats(j), lats(j),lats(j), lats(j)
    write(110, '(20F12.6)') lons(1:xdim)
END DO

```



```

close(70)
close(80)
close(90)
close(100)
close(110)

```

```

END PROGRAM Climatology

```

```

!-----
!      SUBROUTINES
!-----
SUBROUTINE create_grid(length, smallest, biggest, array_out)

implicit none
integer(4),          intent(in)      :: length
real(8),             intent(in)      :: smallest, biggest
real(8), dimension(1:length), intent(out) :: array_out
real(8)              :: increment
integer(4)           :: i,j

increment=abs((smallest-biggest)/(length*2))

IF(abs(smallest)>abs(biggest))THEN
  DO i=1, length*2, 2
    array_out(CEILING(i*1.0/2))=smallest+increment*i
  END DO
ELSE
  DO i=1, length*2, 2
    array_out(CEILING(i*1.0/2))=biggest-increment*i
  END DO
END IF

END SUBROUTINE create_grid


SUBROUTINE find_grid_box(y_array,ydim, x_array,xdim, y, x, col, row)
!This subroutine determines which grid box each region belongs in.

implicit none
integer(4),          intent(in)      :: ydim, xdim
real(8), dimension(1:ydim), intent(in) :: y_array
real(8), dimension(1:xdim), intent(in) :: x_array
real(8),             intent(in)      :: y, x
integer(4),          intent(out)     :: col, row
integer(4)           :: i,j
real(8)              :: euc_dist_lon, &
                      euc_dist_lat, miny, minx

! the following values are supposed to be much larger than any actual value
miny=100.0

```

```

minx=100.0

DO i=1, ydim
  euc_dist_lat=((y_array(i)- y)**2)**.5
  IF(euc_dist_lat < miny)THEN
    miny=euc_dist_lat
    row=i
  END IF
END DO

DO i=1, xdim
  euc_dist_lon=((x_array(i)- x)**2)**.5
  IF(euc_dist_lon < minx)THEN
    minx=euc_dist_lon
    col=i
  END IF
END DO

END SUBROUTINE find_grid_box

```

Appendix B: Sample Streamer Files

B.1 Streamer Input File

```
OPTIONS
.TRUE.                ; Compute fluxes (or radiances)?      (FLUXES)
.TRUE.                ; Include thermal emission in band 106? (IR106)
.FALSE.               ; Compute cloud forcing?              (CLDFRC)
8 8                   ; Number of streams, short and long    (NSTR*)
36                   ; Number of Legendre coeff., method      (NCOEF,IMTHD)
.TRUE.               ; Include gaseous absorption?           (GASABS)
.TRUE.               ; Include Rayleigh scatter (shortwave)? (RAYLISHRT)
1                    ; Surface albedo control                (ALBTYPE)
4                    ; Surface emissivity control             (EMISSTYPE)
1 .TRUE.             ; Std prof; extend input profile to 100 km?
4 1                  ; Aerosol model and profile
1 1 2 1 2 5 3        ; Height, temp, wv, oz, cloud units, band spec.
3                    ; Output levels control
.TRUE.               ; Descriptive output desired?
thesis4.des
.FALSE.              ; User-customized output?
thesis4.out
<weights file name>
.FALSE.              ; Read cloud optical properties?
<file name>
<brdf file name>

$CPRINT 'cldfrac:', cldfrac, /T, 'cldtau:', cldtau, /T, 'TOAupwellingLW:',
aveflxu(1,2), /T, 'TOAupwellingSW:', aveflxu(1,1), /T, 'TOAdownwellingLW:',
aveflxd(1,2), /T, 'TOAswflxdn:', swflxd(1), /T, 'TOAnetflx:', netflx(1), /T, \
'BOAupwellingLW:', aveflxu(25,2), /T, 'BOAupwellingSW:', aveflxu(25,1), /T,
'BOAdownwellingLW:', aveflxd(25,2), /T, 'BOAswflxdn:', swflxd(25), /T,
'BOAnetflx:', netflx(25)

$CASE
thesis flux figure data TEST
2009 12 10 16.0 15.00 60.00 -36.9
0 0 ; viewing geometry
1 129
0.06 1 1 1.0
999.0 0.99
1 1 1.0 999 3.0 1.5 100 1 0 20.0 0.1 999 999 999 ; cloud
0
2 2 2 2 2 2 0 1.0 1.0 1.0 1.0 1.0 1.0
```

B.2 Streamer Descriptive Output File

Streamer, v3.0b8 (beta test)

Input File: thesis4.inp

INITIAL OPTIONS (later changes will not be noted):

Number of Streams, Shortwave: 8, Longwave: 8
 Number of coefficients: 36
 Gaseous absorption included.
 Rayleigh scattering included.
 Default profile: Tropical
 Default aerosol optical model: Maritime
 Default aerosol vertical profile: Background trop. and strat.
 No spectral (band) weighting.
 Using internal cloud optical properties.
 No surface BRDF used.

+++++

Case Number in Input File: 1
 thesis flux figure data TEST
 Band number range: 1 - 129
 Spectral Interval: 20 1/cm (500.00 um) to 35710 1/cm (0.28 um)
 Year: 2009, Month: 12, Day: 10, Hour: 16.00
 Lat: 15.000, Lon: 60.000, Zenith Angle (degrees): 37.86

Unscaled Atmospheric Profiles (25 Levels)						
Height(km)	Press(mb)	T(K)	H2O(g/m^3)	RH(%)	O3(g/m^3)	Aer(km^-1)
1 100.00	0.00	210.80	0.000000	0.00	0.0000000	0.000000
2 70.00	0.06	219.50	0.000000	0.00	0.0000001	0.000000
3 50.00	0.85	270.00	0.000004	0.00	0.0000043	0.000001
4 45.00	1.59	264.90	0.000007	0.00	0.0000130	0.000002
5 40.00	3.05	254.30	0.000014	0.00	0.0000410	0.000005
6 35.00	6.00	243.30	0.000028	0.01	0.0000920	0.000010
7 30.00	12.20	232.30	0.000059	0.04	0.0002400	0.000041
8 25.00	25.70	221.30	0.000131	0.26	0.0003400	0.000081
9 20.00	56.50	207.60	0.000308	3.24	0.0001900	0.000366
10 15.00	132.00	203.60	0.000757	13.56	0.0000470	0.000245
11 14.00	156.00	210.40	0.000986	7.24	0.0000450	0.000275
12 13.00	182.00	217.00	0.001790	5.88	0.0000450	0.000321
13 12.00	213.00	223.80	0.006080	9.23	0.0000430	0.000398
14 11.00	247.00	230.40	0.017900	13.50	0.0000410	0.000496
15 10.00	286.00	237.20	0.049000	18.84	0.0000390	0.000708
16 9.00	329.00	243.80	0.121000	25.22	0.0000390	0.001131
17 8.00	378.00	250.60	0.250000	28.82	0.0000390	0.002094
18 7.00	432.00	257.30	0.471000	31.40	0.0000410	0.003871
19 6.00	492.00	264.00	0.860000	34.23	0.0000430	0.004790
20 5.00	559.00	270.70	1.530000	37.46	0.0000450	0.005784
21 4.00	633.00	277.40	2.660000	41.16	0.0000470	0.006337
22 3.00	715.00	283.60	4.700000	48.64	0.0000510	0.009071
23 2.00	805.00	288.40	9.290000	71.39	0.0000540	0.016154
24 1.00	904.00	294.10	13.000000	71.19	0.0000560	0.025784
25 0.00	1013.00	300.00	19.000000	74.41	0.0000560	0.000035

Total Column Amounts (scaled) -
 Water Vapor: 42462.81 g/m^2
 Ozone: 5.43 g/m^2
 Aerosol Optical Depth: 0.08 (unitless)
 Scaling Factors - w.v., O3, haze RH, CO2, O2, w.v. continuum:
 1.00 1.00 1.00 1.00 1.00 1.00
 Cloud/clear types (models) in scene (101=clear): 1

INDIVIDUAL CLOUD CHARACTERISTICS:

Model	Top Index	Bott Index	Zthick (km)	Pthick (mb)	Frac	Tau	Ttop (K)	Ptop (mb)	Re (um)	WC (g/m ³)	Phase
-------	-----------	------------	-------------	-------------	------	-----	----------	-----------	---------	------------------------	-------

1	22	24	1.500	138.1	1.00	100.0	283.6	715.0	20.0	0.863	Liquid
---	----	----	-------	-------	------	-------	-------	-------	------	-------	--------

(Note: Above cloud fractions do not include overlapping portion, if any.)

SURFACE CHARACTERISTICS:

Clear Sky Fraction: 0.00

Surface Type Fractions -

Sea Water: 1.00, Freshwater: 0.00, Meltponds: 0.00, Melting snow: 0.00

Fresh snow: 0.00, Bare Ice: 0.00, Dry Sand: 0.00, Vegetation: 0.00

Grass: 0.00, Dry grass: 0.00, Deciduous: 0.00, Coniferous: 0.00

Observed Surface Temp (K): 300.0

Emissivity (all bands): 0.9900

Broadband All-sky Surface Albedo, by type:

1: 0.060

Broadband All-sky Surface Albedo: 0.060

ALL-SKY FLUXES (W/m²), CLOUD RADIATIVE EFFECT (W/m²), HEATING RATE (degrees K/day):

	DirSW Down	DiffSW Down	TotalSW Down	LW Down	DiffSW Up	LW Up	NET	Heating Rate
1	1101.71	0.00	1101.71	0.01	761.71	266.58	73.44	0.679
25	0.00	77.18	77.18	441.91	4.63	458.06	56.41	0.000

Appendix C: ASTER Mask Thresholds

The hand derived thresholds of the level 1B, channel 3N ASTER radiance data can be found in a supplemental file named ASTERmask_thresholds.xls. The file identifier, digital number threshold value and cloud type identifier are located in a table. Cloud type identifiers are as follows:

4- cumulus

5- cumulus and overlying cirrus

6- a cloud type or mixed types other than cumulus, cirrus, or cumulus and cirrus.