TIME SERIES ANALYSIS OF NOAA-AVHRR COMPOSITE IMAGES OF LAKE NAIVASHA BASIN

BY:

FATHY ABDEL HAMED HAMMOUDA

APRIL, 1999

TIME SERIES ANALYSIS OF NOAA-AVHRR COMPOSITE IMAGES OF LAKE NAIVASHA BASIN

BY:

FATHY ABDEL HAMED HAMMOUDA

THESIS SUBMITTED TO THE INTERNATIONAL INSTITUTE FOR AEROSPACE SURVEY AND EARTH SCIENCES (ITC), ENSCHEDE, THE NETHERLANDS, IN PARTIALL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE MASTER OF SCIENCE DEGREE IN WATER RESOURCES SURVEYS WITH EMPHASIS ON WATERSHED MANAGEMENT.

EXAMINATION BOARD:

Prof. Dr. A.M.J. Meijerink (chairman of the examination board)
Ir. C.A. Mücher (external examiner)
Dr. A. S. M. Gieske (supervisor)
Drs. R. Becht

INTERNATIONAL INSTITUTE FOR AEROSPACE SURVEY AND EARTH SCIENCES
ENSCHEDE, THE NETHERLANDS

Acknowledgement

I wish to express my gratitude to the Ministry of Public Works and Water Resources in Egypt and the Netherlands Government for giving me the chance to gain more scientific knowledge and experience.

I would like to express my sincere gratitude to Prof. A. M. J. Meijerink, for his support and guidance.

I am deeply indebted to Dr. A. S. M. Gieske for his guidance and critical comments during my MSc. work.

I am grateful to, Ir. A. M. van Lieshout, for his support and help.
I would like to acknowledge all the staff of the Water Resources and Environmental Studies Division.

Thanks are also addressed to the WRAP (Water Resources Assessment Project) and KNWTP (Kenya Netherlands Wetlands Training Programme) personnel for their help during the field work period.

Special thanks go to Eng. Mahmoud Bakr for his help during the program developing phase and Eng. Ahmad Salah for his comments during the preparation phase of this work.

Thanks to all my classmates and my colleges in Egypt.

I am grateful to Eng. Galal Bedda and Eng. Hasab El-Nabi Khafagi for their support for me to be nominated for this fellowship.

Special thanks go to my mother, my brothers and sisters, and my mother in law for their spiritual support.

ABSTRACT

Knowledge of the hydrological behavior is a crucial step in water resources management of large-scale catchments. One of the most important tools in acquiring spatial and temporal information of large-scale catchments is remote sensing imagery. In this study, a time series of public domain NOAA-AVHRR ten days composite images has been processed and analyzed using different time series analysis techniques. The processing has been done using GIS software "ILWIS 2.2" and new C-programs, which were developed during the study. The catchment has been divided into Terrain Mapping Units as primary hydrological zones. The images of the hydrological response variables and ancillary data layers have been combined within the program. The time series of images has been partly analyzed using Fourier analysis and Exploratory models. The hydrological behavior of the different Terrain Mapping Units has been assessed qualitatively based on the dynamic behavior of the derived response variables. It was shown that some of the Terrain Mapping Units could be combined in one hydrological unit. Also the statistical techniques used to reduce the noise-to-signal ratio in the vegetation profile proved to be a successful alternative for radiometric correction techniques, which require detailed information of the underlying interactions between the signal and the atmosphere.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

STATEMENT OF THE PROBLEM

Optimization of the usage of the available water resources in a certain catchment area is a crucial issue, because of the limited amount of water against the increasing demand due to, among others, the increase of population. To achieve this optimization, a thorough understanding of the hydrological behavior of the catchment is a vital prerequisite in the sense that this is the first step in the management process of the water resources in the catchment.

One way to acquire a better understanding is to use remote sensing data especially for large catchments. By using derived hydrological response variables, for example vegetation indices, surface temperature, and surface albedo from remote sensing imagery, and the relation between these response variables and the different components of the hydrological cycle, an assessment of the hydrological behavior and the catchment water balance can be achieved.

In this research the possibility of making use of a series of NOAA-AVHRR satellite images, which have been downloaded from the Internet, in getting information about the hydrological behavior of the study area was investigated. This series of images is processed using the existing techniques of image processing available in ILWIS2.2 software. Part of the processing has been done using a specially developed Program written in C language. This C-program has been used also in creating an interface between the GIS package (ILWIS2.2) and mathematical functions, which were used to process the available data and analyze the output of the image processing phase.

One important research question is how to use time series of raw satellite images data to derive reliable hydrological response variables to end up with a qualitative assessment of the hydrological behavior of a catchment area.

1.1 OBJECTIVES

General objective:

Establishing a method to reach a qualitative assessment of the hydrological behavior for the different zones in the catchment area using time series analysis of public domain sequential satellite images and related spatial data.

Specific objectives:

- Constructing time series for the hydrological response variables derived from NOAA-AVHRR and Landsat satellite images and meteorological data for the different zones of the study area.
- Establishing techniques to decrease the amount of noise imposed in the time series data.
- Time series analysis of the NOAA images using different techniques
- Creating interfaces between GIS and time series analysis techniques.

1.2 RESPONSE VARIABLES

Response variables are variables, which can be used to measure the response of the hydrological system to external climatic factors. One of the most important variables expressing response to the changes of the hydrological components in time, is vegetation. Vegetation indices like the Normalized Difference Vegetation Index (NDVI) and the Soil Adjusted Vegetation Index (SAVI), which are derived from satellite imagery have been developed to express the state of vegetation, and the variation of the associated hydrological components such as evapotranspiration and precipitation with time. Another variable, which can be measured using remote sensing, is surface temperature. In the warm tropical regions of Africa, surface temperature is largely a function of evaporation, when water availability is not limited (Lambin and Ehrlich, 1996). The surface reflectance of land surface (Albedo) is one of the factors, which affects the amount of net radiation available for the energy balance process, which in turn may be used to calculate actual evapotranspiration. All these response variables, which have a direct relationship with the hydrological cycle components, can be derived using remote sensing data.

The value of the above mentioned response variables, the Normalized Difference Vegetation Index (*NDVI*), surface temperature T_0, and surface reflectance r_0 stems from the fact that the Surface Energy Balance Algorithm for Land (SEBAL) utilizes these variables and their interrelationships to map land surface flux densities for a wide

spectrum of land types (Bastiaanssen, 1995). And also these variables can be derived from satellite images.

In this research, the above mentioned response variables were derived from NOAA-AVHRR images. And with the combination of ancillary data e.g. rainfall, air temperature, land cover, and Digital Elevation Model (DEM), a qualitative assessment of the hydrological behavior of the different Terrain Mapping units was established.

1.2.1 Vegetation Indices

Relation between vegetation indices and the hydrological cycle components (especially evapotranspiration) has been studied by many researchers. The theoretical model of Monteith (1972), relates phytomass production to that part of solar radiation which activates photosynthesis. The vegetation indices have been related both theoretically and empirically to the amount of absorbed photosynthetically active radiation (Bartholome, 1991). Since phytomass production depends on the water uptake, the NDVI values can be related to the evapotranspiration loss (Meijerink et al., 1994).

Cihlar et al. (1991), in a study in Canada found a highly significant association between coincident NDVI and actual evapotranspiration values with a correlation coefficient (R) equal to 0.77 or between NDVI and potential evapotranspiration for the previous 15-day period (r=0.86). It was also found that high correlation between cumulative NDVI and cumulative evapotranspiration over the growing season (r=0.96). As cited by Chilar et al. (1991), the correlation between NDVI and actual evapotranspiration (AE) has been found in arid areas by Henrickson and Durkin (1986); Kerr et al. (1989); Seguin et al., (1989); Smith et al. (1990). Also vegetation indices have been related with some of the factors in the radiation balance equation e.g. the surface emissivity, which in turn affects the outgoing long wave radiation. As an example the equation derived by Van de Griend and Owe (1993), which relates the broad-band emissivity for 8 to 14 μm and NDVI :

$$\epsilon_0 = 1.009 + 0.047 \ln NDVI \qquad (1.1)$$

Furhermore, some other studies have related vegetation indices and the Soil Heat Flux/Net Radiation Ratio to overcome the difficulty of the determination of the Soil Heat Flux as a component of the energy balance Equation. An example of this relation the one developed by Bastiaanssen and Roebeling (1993), the equation states that:

$$G/R_n = T_0(t)/r_0(t)(0.0032\, r_0^{avg} + 0.0062\, r_0^{avg\,2})[1-0.978(NDVI)^4] \qquad (1.2)$$

Where:
G is the Soil Heat Flux
R_n is the Net Solar Radiation
NDVI is the Normalized Difference Vegetation Index
$T_0(t)$ is the instantaneous surface temperature
$r_0(t)$ is the instantaneous surface albedo

r_0^{avg} is the daily average albedo

1.2.2 SURFACE ALBEDO

One of the parameters that are required over the land surface is the reflectance of the surface or surface albedo. This is because by the determination of surface albedo and the incoming short wave radiation the amount of the net short wave radiation absorbed by the land surface, which is used in the energy balance equation, can be derived from albedo and incoming shortwave radiation. Surface albedo is the ratio between the radiation reflected from a surface and the radiation incident on the surface. Albedo varies with underlying surface type (e.g. coniferous forest, crops, urban development), snow cover, surface wetness and solar zenith angle (Saunders, 1990). The soil surface albedo decreases with the increase of soil moisture content. The plant albedo can have large variations depending on the season and the phase of growth. The spectral albedo of vegetative covers, for example, especially those with rich green leaves exhibit some peculiar selective features. Albedo is generally low in the visible spectrum and high in the infra-red region. The selective feature is with respect to the chlorophyll absorption band interval near $\lambda = 0.65$ μm, where albedo is minimum. Spectral albedo has a strong dependence on the type of vegetation, its age, and the season. For vegetative covers albedo increases as the solar height decreases and is always at its minimum at solar noon. At low solar height, albedo is higher at higher wavelengths (Iqbal, 1983).

The broad band albedo can be calculated using the bi-directional reflectance of individual channels. In the case of NOAA-AVHRR channels, Saunders suggested that for two channel radiometers at visible and near-infrared wavelenghts, the broad band bi-directional reflectance ρ is given by

$$\rho = w_1 R_1 + w_2 R_2 \qquad (1.3)$$

where R_1 and R_2 are the filtered bi-directional reflectances for channel 1 and channel 2 and w_1 and w_2 are the factors which weigh each reflectance by the amount of incoming solar radiation in each channel. More exactly w_1, and w_2 are defined as

$$w_1 = I_s(1)/I_s \qquad w_2 = I_s(2)/I_s \qquad (1.4)$$

Where I_s is the solar radiance integrated over all wavelengths, $I_s(1)$ is the solar radiance integrated over all wavelengths less than the wavelength mid-way between the two channels λ and $I_s(2)$ is the solar radiance integrated over all wavelengths greater than λ.

One of the approaches in albedo analysis has been described in the NOAA Technical Report NESDIS 49 (1990). The report states that to drive reliable reflectance parameters from AVHRR, one must, in an operational environment, convert an instantaneous, highly directional, narrow band radiation measurement into an estimate of daily average broadband flux over the whole upward hemisphere. This conversion is accomplished by applying models of the following types:

1- Bi-directional (Solar zenith, satellite zenith, relative azimuth angles) – this is a set of patterns, each valid for a particular interval of solar zenith angle and for an individual scene type. Each pattern shows the relative brightness as a function of the direction from which the scene is viewed.
2- Directional (Solar zenith angle): The relationship between albedo and solar zenith angle for a given scene type. This relationship can also be used to show the variation of albedo with the time of day.
3- Narrow to broadband : A weighted combination of albedos or radiances from narrow band channels to yield an estimate of the broadband value. The weights used may also optimally be a function of the scene type.

The final equation that was used in this research and suggested in the NOAA Technical Report NESDIS 49 (1990) report for the instantaneous broadband albedo is as follows:

$$BB = 0.7459 + 0.347\ A1 + 0.65\ A2 \qquad (1.5)$$

Where:
BB is the broadband albedo (%)
A1 is the reflectance of channel 1 (%)
A2 is the surface reflectance of channel 2 (%).

Since reflectance values for channel 1 and channel 2 for the data set used in this research were atmospherically corrected only for Ozone and Rayleigh scattering and there is no correction for water vapor and aerosols The broad band albedo calculated from equation 1.5 can be considered as planetary albedo. To get surface albedo using planetary albedo, values of transmittance and minimum values of planetary albedo (to calculate the path radiance assuming a clear day) for every image in the time series should be known. Since water bodies have been masked and there is no ground data to calculate transmittance, it was found that there are two solutions. Either to assume values for both the transmittance and minimum values in every image or by considering planetary albedo as the surface albedo and verifying the calculated values with values in the literature. The second solution was adopted in this research.

1.2.3 SURFACE TEMPERATURE

The importance of deriving the radiative surface temperature stems from the fact that it is one of the parameters which is used to estimate the upwelling long wave radiation, which is part of the radiation balance equation. Recently the surface temperature combined with NDVI has been also used for land-cover mapping and land-cover change analysis (Lambin et al., 1996).

Longwave radiation can only be transmitted through the atmosphere in those ranges of the spectrum where the molecular absorption by water vapour, gases, and suspended materials is minimized, i.e. atmospheric windows in the 9-14 μm spectral

range. Relative to water vapour, the scattering and absorption of other gases is small. Atmospheric correction schemes are indispensable and may vary from highly sophisticated physically based radiative transfer models to simple empirical methods (*Bastiaanssen, 1995*). As has been cited by Bastiaanssen also split-window techniques have been developed to account for the differential water vapour absorption in the 10 to 13 μm range assuming the surface emissivity to be constant over the latter spectral range (Price, 1984). Other methods relies on in-situ measurements which are compared with satellite based measurement. The approach that was used in this research uses the split window technique to derive the surface temperature from the NOAA AVHRR channels 4, 5. This approach that has been derived by Caselles et al. (1994) further discussed in Chapter 3.

1.3 HYDROLOGICAL VARIABLES

Several basic hydrologic concepts are related to the simple model of a *system*. A system is any conceptually defined region of space capable of receiving a sequence of *inputs* of a conservative quantity, storing some amount of that quantity, and discharging *outputs* of that quantity (Dingman, 1994).

Hydrological variables, precipitation, evapotranspiration, stream outflow, ground water outflow, ground water inflow, and the storage are the different components of the water balance in a regional catchment area. To manage the water resources of a catchment area, which can be regarded as a system, there are two approaches. The first is quantitative estimation of the above mentioned components. Quantification of the water balance components can be done by point measurements e.g. rainfall, stream flow and evapotranspiration. Other components can be estimated through models like ground water inflow and outflow. Different techniques have been used to convert the point measurements to areal estimation. Remote sensing can also be used in quantifying evapotranspiration with models like SEBAL (Bastiaanssen, 1995). The second approach, which was suggested by Grayson et al. (1993), is the use of simple spatial modeling combined with qualitative reasoning. This approach is consistent with the availability of data and with our current ability to represent the system (Meijerink et al., 1994).

1.4 METHODOLOGY

Most of the data processing has been automated using a program written in C programming language. This automation was for the following reasons:

- The large number of images and maps required that should be processed and combined
- The need to create an interface between the GIS software (ILWIS 2.2) and mathematical functions like Fast Fourier Transformation to facilitate the data analysis.
- The need to perform some mathematical and statistical operations for noise removal in space and time.

The program consists of different functions, each function performs a certain data processing algorithm. The inputs to the program are map lists files for the channels 1, 2, 4, 5, ancillary maps, air temperature maps, and rainfall data text files. The output was time series of hydrological response variables, amplitude and phase of Fourier components at different frequencies. The output of the different functions has been tested with other software like SYSTAT for testing the output of the Fast Fourier Transformation Function. Ground truth data has been collected during the field work for different land covers and also value ranges of surface reflectance and surface temperature for example for irrigated area has been used to check the reasonability of the calculated values against values from literature.

1.5 THE STUDY AREA

Republic of Kenya is located in East Africa, of area about 580,370 square kilometers. Sudan and Ethiopia border the country on the north, on the east the country is bordered by Somalia and the Indian Ocean, on the south by Tanzania and on the west by Lake Victoria and Uganda (Figure 1.1).

LAKE NAIVASHA BASIN

The Naivasha basin is bounded by the Aberdares Mountains to the East and the Mau escarpment to the west, The total area of the catchement is 3200 km^2 and is shown on Figure (1.3). This Figure also shows the existing drainage network in the area.
The main sub-divisions in the catchement are:-(Mavuti, 1983)

(i) The Malewa River Basin, including the Turasha River Basin (1,730 km^2).
Drainage into the Malewa starts among the steep forested eastern slopes from the Kinangop plateau (2,483 m) and the Aberdares (3960 m) where the average annual rainfall is 1087.5 mm. Initial flow takes place in a westerly direction via a number of steeply graded tributaries that, at the lower slopes of the range, develop into four main tributaries, the Mugutyu, Turasha, Kitiri, and Makngi.

(ii) The Gilgil river basin (420 km^2).
The Gilgil drains a long narrow basin (the Bahati Highlands to the north of the Elementeita-Nakuru basin) in the western part of the Naivasha catchement.
It has few tributaries and rises at an altitude of approximately 2,772 m, in an area where the average rainfall is 1300 mm. The two important tributaries of it are Marundati and Little Gilgil rivers.
(iii) The Karati Catchement; the lake itself; and the areas around the lake to the east, south and west (1,238 Km^2).
Karati is the other river that flows occasionally into the lake, it drains about 134.7 (Km^2) and is normally dry for long periods. It rises at altitude 2,6488 m. where annual rainfall is about 775 mm.

Figure 1.1 General overview of the country location

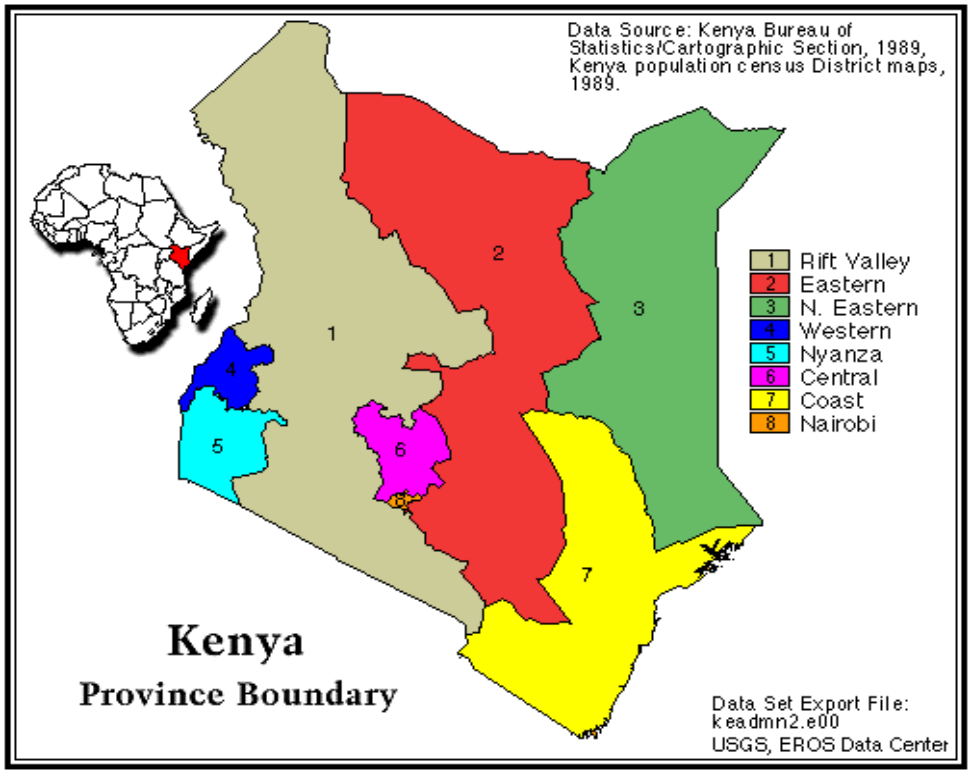


Figure 1.2 Location Map of the Kenya Rift Valley

Figure 1-3 A map showing the lake Naivasha drainage basin (Mavuti 1983)

The three rivers enter the lake from the north through a swamp area (11.7 km^2 in area) that spreads out across the north end of the lake. Prior to entering the lake the river diverges into dendritic pattern and disappears under the floating mat of the swamp, no channel could be detected from the air.

Geology of Lake Naivasha Basin

The volcanic rocks of the Naivasha area consist of a mixed assemblage of acid and basic lava. In the Southeast and Southwest of the basin fumaroles, Hot Springs and steam vents are found and in the Njorowa Gorge. These are being harvested for geothermal power generation. Several of the craters within the basin can be seen along the western side of the lake, one of which contains standing water (Crater Lake).

Soil

According to KSS(1980), the distribution of soils in the area is complex. The soils are derived mainly from weathered volcanic and basement rock system. Generally soils in the study area can be grouped into two: soils developed on the Lacustrine plain and those developed on the volcanic plain.

Soils developed on the Lacustrine plain are moderately well drained, very deep, very dark grayish brown to pale brown, silty clay to clay loam.

Soils developed on the volcanic plain are well drained, moderately deep to very deep, dark brown to pale brown, with non-calcareous to moderately calcareous topsoil, and moderately to strongly calcareous deep soil.

Climate

The climate is humid to sub-humid in the highlands and semi-arid in the rift valley. The mean monthly maximum temperature range between 24.6°C to 28.3°C, and mean monthly minimum temperature between 6.8°C to 8.0°C. the average monthly temperature ranges between 15.9°C to 17.8°C. the average annual rainfall ranges from about 1300 mm in the Kinangop plateau to about 600 mm in the rift floor. The rainy seasons are typically April to June and October to November.

The hydrology of Lake Naivasha basin

The Naivasha Basin, whether or not it is technically a closed basin, behaves hydrologically as if it were one and fluctuates in the manner of a closed lake (Richardson and Richardson 1972). Lake level changes occur continuously. Although these changes in level generally reflect periods of drought versus excessive rainfall there is no simple correlation between any one of input vector on a monthly basis. In other words a month with High River discharge may result in Lake Levels other than those which would be

predicted on the basis of discharge alone. This may be due to differences in climatic factors, as well as seepage effects which caused a delayed response to changes in the balance, and are of considerable importance in this basin.

References

Bartholome E., (1991). Remote Sensing and Agricultural Production Monitoring in Sahelian Countries. In: Belward and Valenzuela eds. Remote Sensing Applications and GIS for Resources Management in Developing Countries. *(Kluwer Acad. Publ. Dordrecht) Eurocourses, 189-214.*

Bastiaanssen, W. G. M. (1995). Regionalization of surface flux densities and moisture indicators in composite terrain, A remote sensing approach under clear skies in Mediterranean climates. *PhD thesis, Agricultural University of Wageningen, The Netherlands, pp. 273.*

Bastiaanssen, W. G. M., and Roebeling, R. A. (1993). Analysis of Land Surface Processes in two Agricultural Regions in Spain using Thematic Mapper Simulator data. In: Bolle, Feddes and Kalma (Ed.), Exchange processes at the Land Surface for a Range of space and Time scales. *JAHS Publ., 212, 407-416*

Cihlar J., St. Laurent L., and Dyer J. A. (1991). Relation between the Normalized Difference Vegetation Index and Ecological variables. *Remote Sensing for Environment , 35: 279-289.*

Dingman, S. L. (1994). Physical Hydrology. *Printice-Hall, Inc., Simon and Schuster / A Viacom Company, New Jersey, pp. 575.*

Gaudet, J. J. and Melack, J. M. (1979). Major Ion Chemistry and Solute budget in a Tropical African Lake Basin, *Feshwater Biology.*

Kilham, p. (1971). Biochemistry of African Lakes and Rivers, *PhD thesis, Duke University, Durham. NC. pp 199.*

Grayson R.B., Blosch, G., Barling and Moore, I. D. (1993). Process, Scale and Constrains to Hydrologic Modeling in GIS. HydroGIS. Application of Hydrology and Water Resources Management. Kovar Ed. K. and Nachthbel, H. P. *IAHS publ. No.211 pp 83-92.*

Kerr. Y. H., Imbernon, J., Dedieu.G, Hautecoeur, O., Lagouarde, J, and Seguin,B. (1989). NOAA-AVHRR and its Uses for Rainfall and Evepotranspiration Monitoring. *International Journal of Remote Sensing. 10:847-854.*

Lambin E. F., and Ehrlich D. (1996). The Surface Temperature-Vegetation Index Space for Land Cover and Land Cover Change Analysis. *International Journal of Remote Sensing, 17: no. 3,463-487.*

Mavuti, K. M., (1983). Wetlands and Shallow Water Bodies in Kenya: Characteristics and General Status of Knowledge, *Promotuion of Science and Tech. Kenya, 5.pp 48-58.*

Van de Griend A. A., and Owe M. (1993). On the Relationship between Thermal Emissivity and the Normalized Difference Vegetation Index for Natural Syrfaces. *International Journal of Remote Sensing, 14, no. 6, 1119-1131.*

Iqbal, M. (1983). An Introduction to Solar Radiation. *Academic Press, a Subsidiary of Harcourt Brace Javanovich, Publishers.*

Meijerink A. M. J., De Brower H. A. M., Mannerts C. M., and Valenzuela. C. R. (1994). Introduction to the use of Geographic Information Systems for practical Hydrology. *ITC, Publication Number 23.*

Monteith, J.L. (1976). Vegetation and the atmosphere. *Vol. 1Principles. Vol. 2, Case studies. London, Acad.press.279 and 439 pp.*

NOAA Technical Report NESDIS 49, (1990). Implementation of Reflectance Models in Operational AVHRR Radiation Budget Processing.

Price, J. C., (1984). Land Surface Temperature Measurements from the Split Window Channels of the NOAA-7 Advanced very High resolution Radiometers, *J. Geophys. Res.89: 17-22.*

Richardson J. and Richardson A., (1972). History of an African lake and its Climate Implications. Ecol.Monogr. 42: 499-534.

Saunders R. W. (1990). The Determination of Broadband Surface Albedo from AVHRR Visible and Near-Infrared Radiances. *International Journal Of Remote Sensing, 11, No. 1, 49-67.*

Seguin, B., Assad,E., Freteaud,J. P. Imbernon, J., Kerr,Y., and Lagouarde,J.P. (1989).Use of Meteorological Satellites for Water Balance Monitoring in Sahelian regions. International Journal of Remote Sensing.10:1101-1117.

Smith, M. O., Ustin, S. L., Adams, T. B., and Gillespie, A. R. (1990). Vegetation in Deserts: II. Environmental influences on Regional abundance. Remote Sensing of Environment. 31:27-52.

KSS (1980). Soil condition at Kulia farm (Naivasha). KSS report

CHAPTER 2

DATA AVAILABILITY AND QUALITY

2.1 PUBLIC DOMAIN DATA

Part of the data used in this research is data downloaded from the "http://edcwww.cr.usgs.gov/landdaac" web site in the Internet. NOAA-AVHRR images, the Digital Elevation of the World at resolution of 30 seconds arc (about 1 Km at the equator), and the downloaded Thematic Mapper images are used in this research. Characteristics of the different types of data available and its quality and problems encountered in preparing and organizing these data are discussed hereafter in this chapter and also the available ancillary data for the study area.

2.1.1 NOAA AVHRR Series

The NOAA-AVHRR sensors onboard the NOAA polar orbiting series of meteorological satellites are used increasingly for deriving quantitative (hydrologic) surface information (Bakker et al. 1997). During the past decade the AVHRR-derived vegetation indices proved to be a useful tool in depicting the large scale distribution and phenological changes of vegetation over particular regions of the world (Gutman, 1991). The availability of NOAA-AVHRR as free data set in the Internet (the data set that was used in this research) made it possible to make use of the available data. However caution has to be taken before using this data set. For this reason the characteristics of this data set of images has to be studied carefully before using it in estimating any derived variables, in the analysis of those variables and consequently drawing any conclusions from this analysis, which may be misleading conclusions. In the following paragraphs the characteristics of the data set and its availability as has been quoted from the above mentioned web site are discussed hereafter.

Available NOAA AVHRR Images

Currently, NOAA-AVHRR composite images are available from April 1992 to January 1996. There is a data gap period during the time that NOAA-11's orbit decayed to the point that the local overpass times were so late in the afternoon that severe sun angles were encountered which made the data scientifically questionable. The data gap starts in October 1994 and ends in January 1995. The rest of the data starting from February 1995 till January 1996 were recorded using NOAA-14 satellite.

The data set is made up of 5 channels raw AVHRR data, at 1.1-km resolution (at nadir) from each daily afternoon pass over all land and coastal zones using data from

NOAA's polar-orbiting TIROS. The available channels and bands in each 10-day record are in Table 2.1.

BAND	DESCRIPTION	BAND	DESCRIPTION
1	AVHRR channel 1	6	VDVI
2	AVHRR channel 2	7	Satellite zenith
3	AVHRR channel 3	8	Solar zenith
4	AVHRR channel 4	9	Relative azimuth
5	AVHRR channel 5	10	Date index

Table 2.1; Available channels in the data set.

Processing Methods

The implementation of the data processing flow is generally a stepwise process that incrementally applies higher order processing in a logical and efficient manner (Eidenshink and Faundeen, 1998). The steps identified are:

- radiometric calibration
- atmospheric correction
- computation of NDVI
- geometric registration
- compositing

The following is a discussion of the methods and parameters that have been recommended for each step.

Radiometric Calibration

Radiometric calibration of the AVHRR visible and near-infrared channels (channels 1 and 2) is difficult because there is not always reliable preflight calibration, no onboard calibration, and difficulty with inflight calibration. Preflight calibration coefficients can change while the instrument is in storage or after launch because of the space environment. Degradation of AVHRR sensors after launch is well documented (e.g., Rao, 1987; Price, 1987; Holben et al., 1990). These studies have used a variety of approaches such as ground-based measurements from stable sites such as homogeneous desert targets to monitor the degradation of the sensors. Although the results from the different approaches often are not in close agreement, there is an increasing consensus on suitable coefficients. The calibration method that is recommended for this data set accounts for sensor degradation by using coefficients developed by Teillet and Holben (1994). Their calculation uses a desert calibration to develop time-dependent calibration coefficients for the AVHRR sensor on NOAA-11. Use of calibration coefficients involves extrapolation of the most recent calibration results for processing data on a near real-time basis. Therefore, the time-dependent coefficients are based on a piecewise linear fit of the desert results.

Teillet and Holben (1994) describe the equations for radiometric calibration to radiance and reflectance for the visible and near infrared channels. The calibration

coefficients for AVHRR thermal channels 3, 4, and 5 are derived onboard the satellite using a view of a stable blackbody and deep space as a reference. The calibration process converts raw digital counts to radiance as described by Kidwell (1991). The radiance values for all channels are stored with 10-bit precision.

Atmospheric Correction

The impact of atmospheric effects on the AVHRR channel 1 and 2 data and NDVI can be significant. Four principle atmospheric factors, water vapor, aerosols, ozone, and Rayleigh scattering, are considered to have the most impact.

The corrections for ozone and Rayleigh scattering are straightforward (Teillet, 1991). Appropriate Rayleigh scattering correction must include an adjustment for topography. Recommended reference values for Rayleigh optical depths for standard pressure and temperature conditions are available (Teillet, 1990). The local elevation adjustment can be derived using ETOPO5, a 5-minute resolution gridded digital elevation data, which is the best available global digital terrain data at this time and for this task has sufficient accuracy.

The correction for ozone should be based on actual measurements derived from the Total Ozone Mapping Spectrometer (TOMS) or other appropriate sensors. However, access and utilization of these data can be difficult. Using the concentration values from standard climatic tables with latitudinal and seasonal dependence is acceptable and is the approach that has been implemented for this data set.

Several approaches for correction of water vapor exist but there is no agreement on a feasible method. The basic problem is determination of the spatial and temporal variability of water vapor concentrations. The same circumstances affect aerosol corrections. Therefore no water vapor or aerosol correction will be applied.

The input to the atmospheric correction process is radiance values from the calibrated visible and near-infrared channels. The output of the atmospheric correction process is surface reflectance (in percent) of the visible and near-infrared, albeit without corrections for water vapor and aerosols.

Computation of NDVI

The NDVI is the difference of near-infrared (NIR, channel 2) and visible (VIS, channel 1) reflectance values normalized over the sum of channels 1 and 2 ((NIR-VIS)/(NIR+VIS)). The NDVI equation produces values in the range of -1.0 to 1.0, where increasing positive values indicate increasing green vegetation and negative values indicate non-vegetated surface features such as water, barren, ice, and snow or clouds. To obtain the most precision, the NDVI is derived from calibrated channels 1 and 2 data in 16-bit precision.

Geometric Registration

Geometric registration involves precise transformation of the image from the sensor-based projection to an earth surface- based projection. This process includes calculating a satellite model, matching ground and image-based control points, and transformation and resampling the data to a map projection coordinate system. The satellite model is also used to compute satellite zenith, solar zenith, and relative azimuth viewing angles for each pixel.

Based on a comprehensive evaluation of map projections for global data sets, the Interrupted Goode Homolosine is recommended for the data set. The Interrupted Goode Homolosine has two important features. First, it is an equal area projection that facilitates spatial analysis. Second, it essentially divides the world into 12 regions that can be mosaicked into a global map. The regionalization of the global map has advantages for data handling.

Control points for AVHRR data are commonly identifiable features along coastlines, lakes, and rivers. Some prominent physiographic features are used in arid regions where hydrologic features are sparse.

Yet another approach is to use hydrologic features from vector data sets such as the Digital Chart of the World (DCW) and the World Vector Shoreline (WVS). The vector data representing coastlines, shorelines, and other major hydrologic features are rasterized to AVHRR resolution and warped into satellite projection. A comparable edge feature is derived from the AVHRR data at approximately the same location using edge extraction filtering methods. The correlation is performed on several control points and adjustments are made. The advantage of this approach is that DCW and WVS are available worldwide. The accuracy of the DCW, however, is variable, depending on the geographic location and original map source. The WVS is considered more accurate for shorelines than DCW, but DCW is the best source of inland water bodies and rivers.

The prototype procedure that was developed using the DCW had an accuracy of 0.8 rms to 1.3 rms pixels based on image to map verification and, therefore, meets the recommended standards in most cases. The accuracy of the registration was improved slightly by constraining the correlation process. However, the accuracy was still slightly greater than 1.0 rms pixels. The most limiting factor is still the overall accuracy of the DCW and WVS. However, the important point was to achieve multi-temporal (image to image) registration rather than absolute positional accuracy in order to prevent "blurring" in the compositing process. Verification of image to image registration had an accuracy of less than 1 pixel rms.

COMPOSITING

The first factor to consider in the compositing process is the length of the compositing period. Compositing periods of 7, 10, and 14 days have been used most commonly. The choice of the period is usually based on the length of time necessary to

obtain a composite with minimal cloud contamination and/or the amount of time necessary to observe meaningful changes in surface characteristics. The compositing period that is recommended for the prototype product is approximately 10 days created by month. Thus, January has three composites of 10, 10, and 11 days; February has 10, 10, and 9 or 8 depending on whether it is a leap year, and so on. This procedure has the advantage of creating calendar month composites, which is a common reporting period for agronomic and biophysical characteristics.

The recommended method is maximum NDVI compositing. The NDVI is examined pixel by pixel for each observation during the compositing period to determine the maximum value. The retention of the highest NDVI value reduces the number of cloud-contaminated pixels and selects the pixels nearest to nadir (Holben, 1986).

Processing Flow

The first step was to read in the portion of an orbital segment corresponding to the region of the Interrupted Goode Homolosine projection that was being processed. The calibration of the five AVHRR channels to radiance was completed next. The date of the observation was determined and the appropriate time adjusted coefficients for channel 1 and 2. The next step was to perform the control point matching and terrain correction process and develop the adjustments to the orbital model. The control points and the terrain data were warped into the satellite projection for the correlation process. A transformation grid was computed but not applied in this step. The satellite model was used to compute the three solar/satellite viewing angles. The angles are eventually used in the atmospheric correction process.

Next, the geometric registration was completed for the five AVHRR spectral channels and the three viewing angles. The transformation grid developed in the control point correlation process was applied. The grid postings were at 10-km intervals to be consistent with the ETOPO5 digital elevation data. Intervening pixel locations were interpolated.

The next step was to compute the NDVI and perform the maximum NDVI compositing. The output of the compositing process was defined as a 10-band image that includes the NDVI value for each pixel selected by the maximum value compositing for the 10- day period, the radiometrically calibrated channel 1-5 values, the satellite viewing geometry data, and a date index value.

The final step was atmospheric correction. Typically, atmospheric correction is performed in conjunction with calibration. However, in a recent study (Cihlar and Huang, personal communication) it was shown that using atmospherically corrected data in the maximum NDVI compositing process increased the probability of selecting pixels with higher satellite zenith angles, and preferably in the backscatter direction. Based on these findings and one other important factor, the atmospheric correction will be applied following the compositing process.

Data Scaling Characteristics

The Global Land 1 km AVHRR Data Set products are processed to maintain the maximum precision of the data. The AVHRR channels 1-5 are scaled to 10-bit precision within a 16-bit (signed) integer data type. The NDVI, viewing geometry, and data pointer are stored as byte data. Tables 2.2 and 2.3 describe the data range, scale, offset, and methods for unscaling the data.
Equations for scaling and unscaling values:

scaled = (actual + shift) * scale + offset

actual = (scaled - offset) / scale - shift

- Values to be scaled are stored in signed data types, except unsigned byte.
- Each value to be scaled must be greater than or equal to 0. This allows the values 0 to 9 to be reserved for special masks (water, interrupted area, no data, etc). To accomplish this, a shift value is added to each pixel prior to scaling.
- Care must be taken when unscaling the data. The mask values must be reset to some value not present in the data range or they must be reset to a fixed value. For example, if all mask values are set to zero, they will be indistinguishable from a data value of zero.

Field	Actual	Shift	Shifted
VDVI	0 – 200	0	0 – 200
Satellite Zenith	-180 – 180	90	0 – 180
Solar zenith	0 – 180	0	0 – 180
Relative Azimuth	-180 – 180	180	0 – 360
Reflectance	0 – 100	0	0 – 100
Radiance	0 – 540	0	0 – 540
Thermal	160 – 340	-160	0 – 180

Table 2.2 Parameters used to calculate the actual values (1)

Data layer	Units	Q-bits	Offset	Scale
NDVI	-	8	10	1
Satellite Zenith	Degrees	8	10	1
Solar Zenith	Degrees	8	10	1
Relative azimuth	Degrees	8	10	1
Ch1 reflectance	%	16	10	10
Ch2 Reflectance	%	16	10	10
Ch3 Btemp	Kelvin	6	10	5.602
Ch4 Btemp	Kelvin	16	10	5.602
Ch5 Btemp	Kelvin	16	10	5.602
Date	Index	8	10	1

Table 2.3 Parameters used to calculate actual values (2)

2.1.2 THEMATIC MAPPER IMAGES

- The TM downloaded images from the Internet have the following characteristics:
- Images are resampled every 16 row, which means that only the digital numbers of one row from every 16 row is considered to be representative for the other 15 rows.
- Only channels 3, 4, 5 are available.
- The study area lies in four images with two paths and two rows.

Two problems have been encountered and hindered constructing time series from the TM images. Problems are as follows:

- The cloud contamination which exists in most of the images, which reduces the temporal resolution in the series.
- The existing images are not evenly distributed in time and even they are not near-evenly distributed.
- Since the study area does not exist in one image, but in four images, this has caused a problem because the images have been taken in different dates. This problem hindered the combination of the four images.

The above mentioned problems are shown in the following charts, which show the amount of images available, the cloud contamination extent, and the dates of imaging for the four parts of the study area.

The higher the value of the cloud cover in the Y axis in the chart, the higher the number of contaminated pixels. Full coverage of clouds occurs at the value 10.

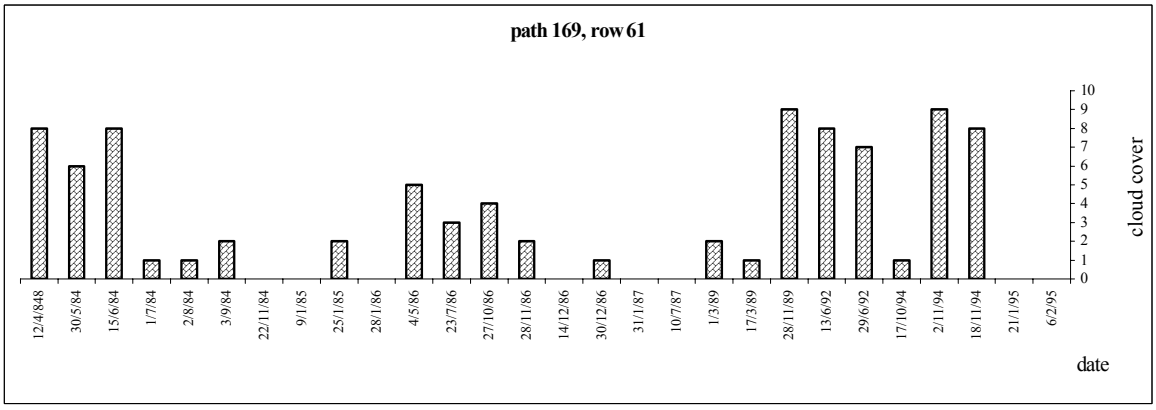


Figure 2.1 available images and cloud condition for path 169, and row 61.

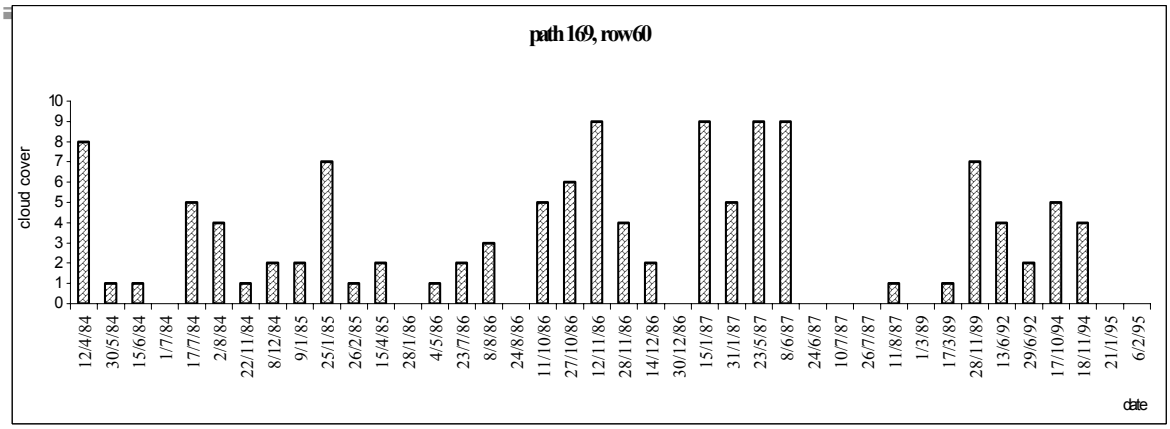


Figure 2.2 available images and cloud condition for path 169, and row 60.

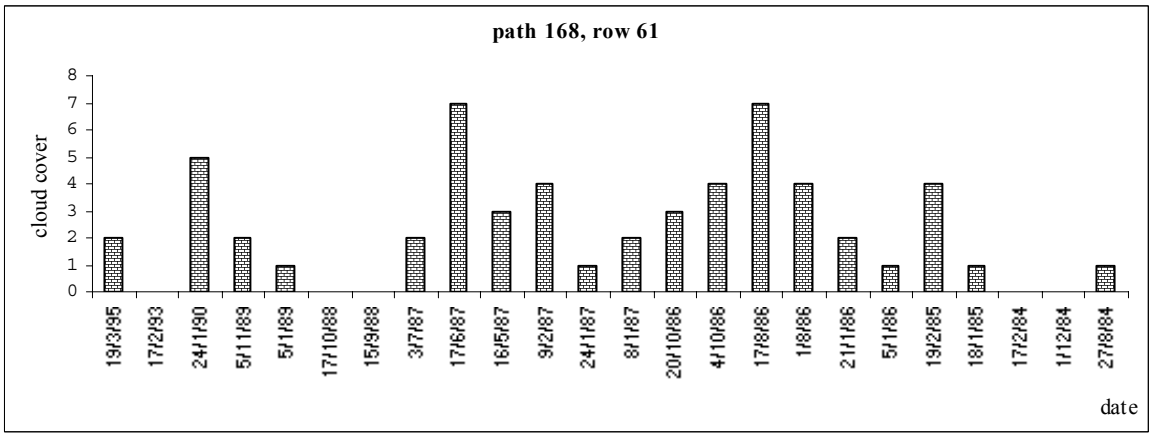


Figure 2.3 available images and cloud condition for path 168, and row 61.

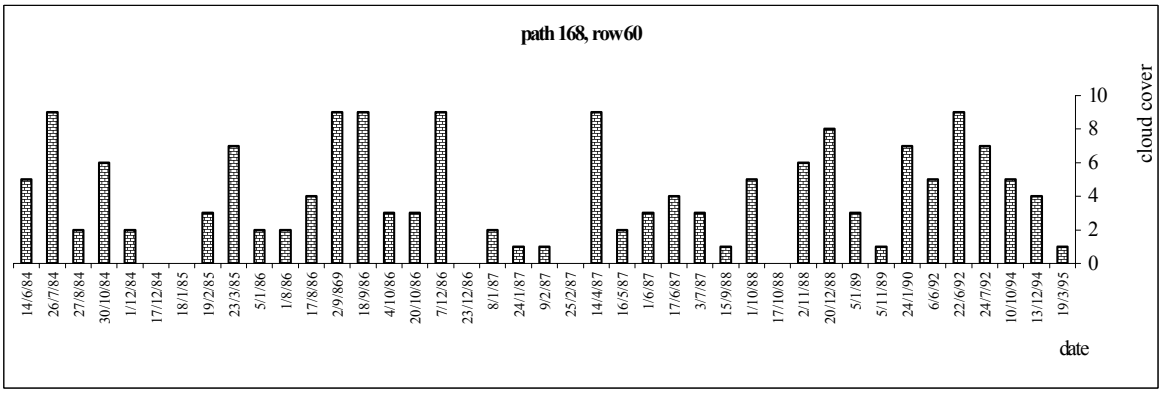


Figure 2.4 available images and cloud condition for path 168, and row 60.

Location of the different paths are as follows:

- Path 168, row 60 : N00 51 55 Northwest Longitude : E036 42 41 Northeast Latitude : N00 37 40 Northeast Longitude : E038 21 23 Southeast Latitude : S00 51 55 Southeast Longitude : E038 02 13 Southwest Latitude : S00 37 40 Southwest Longitude : E036 23 31
- path 168, row 61: Northwest Latitude : S00 34 51 Northwest Longitude : E036 24 17 Northeast Latitude : S00 49 07 Northeast Longitude : E038 02 58 Southeast Latitude : S02 18 42 Southeast Longitude : E037 43 49 Southwest Latitude : S02 04 26 Southwest Longitude : E036 05 04
- Path 169, row 60: Northwest Latitude : N00 51 58 Northwest Longitude : E035 11 48 Northeast Latitude : N00 37 43 Northeast Longitude : E036 50 30 Southeast Latitude : S00 51 52 Southeast Longitude : E036 31 20 Southwest Latitude : S00 37 37 Southwest Longitude : E034 52 38
- Path 169, row 61: Northwest Latitude : S00 34 51 Northwest Longitude : E034 51 35 Northeast Latitude : S00 49 07 Northeast Longitude : E036 30 16 Southeast Latitude : S02 18 42 Southeast Longitude : E036 11 07 Southwest Latitude : S02 04 26 Southwest Longitude : E034 32 22

2.1.3 The Digital Elevation Model (DEM)

The Digital Elevation Model (DEM) of the study area has been derived from public domain GTOPO30. GTOPO30 is a global digital elevation model (DEM) resulting from a collaborative effort led by the staff at the U.S. Geological Survey's EROS Data Center. A C-program has been designed to read the binary file of the study area and then to convert it to an ILWIS map. The characteristics of the data set and a description of the processing steps of the input file are explained in the next paragraphs.

Characteristics of the data set

The characteristics of the GTOPO30 data set, as has been quoted from the "readme"file is as follows:

- Elevations in GTOPO30 are regularly spaced at 30-arc seconds (approximately 1 kilometer).
- The DEM is provided as 16-bit signed integer data in a simple binary raster.
- Data sources for Africa have been derived from the Digital Terrain Elevation Data and the Digital Chart of the World
- The accuracy for these data sources are expected to be + or – 30 meters at 90 percent confidence for the Digital Terrain Elevation Data, and for + or – 160 meters for the Digital chart of the world.

Data processing

As has been mentioned the data processing for the DEM file has been done by a designed C-program. The inputs for the program are the binary file of the Digital

Elevation Model "e020n40" and information from the header file about the organization of the binary file. The output of the program is an image file in the ILWIS format (binary file) contains the elevation data for every pixel for an area includes the study area. After generating the DEM map by the program the following steps have been done using ILWIS2.2 to combine the DEM with other data sources for the study area:

- The output file was georeferenced to the coordinates of the corners given in the header file of the input file.
- After georeferencing, a submap of the DEM has been created to include only the study area.
- The submap was resampled using the "noaageor" georeference used by the rest of the data set.
- A masked map using a polygon map of the catchment boundary has been used to generate a DEM map for the catchment area only. This has been done using map calculation formula of ILWIS2.2.
- A final submap with the same size like other maps has been created.

To test the validity of the DEM, the following steps were taken:

- An existing DEM for part of the study area containing the lake, which has been generated from digitized contour lines, has been used for comparison.
- The existing DEM has been aggregated with the average function
- The output map of the aggregation process has been resampled to the georeference of the rest of the data set.
- The public domain DEM has been masked to the same area as the existing DEM.
- The public domain DEM has been compared with the aggregated existing DEM, using lake area and some ground truth data collected from the field. It was found that there is a difference of about 150 meters in the lake area.
- A C-program has been designed to extract data from the two files of the final DEM maps of the Internet DEM and the existing DEM.
- The input to the program is a map list containing the two DEM files.
- The output of the program is a text file contains two columns of data.
- Every row contains two cells containing the elevation of a particular pixel from the two files.
- The text file is then opened in EXCEL software.
- The scatter plot, the regression line, and the correlation coefficient between the two original data sets, and between the public domain DEM –150 meters and the existing DEM has been established
- Also the existing DEM has been sorted in order to devide the data set into two classes. The first class contains values till 2399 m altitude and the second class contains values from 1400 meters till 3018. The correlation has been studied for the two classes between the two DEMs.
- The results of the comparison demonstrated that the values of the public domain DEM after subtracting a value of 150 meters are more correlated to the existing DEM.

- Comparison of the two DEMs for the classes and the correlation coefficients are shown in Figures 2.8, 2.9,2.10.

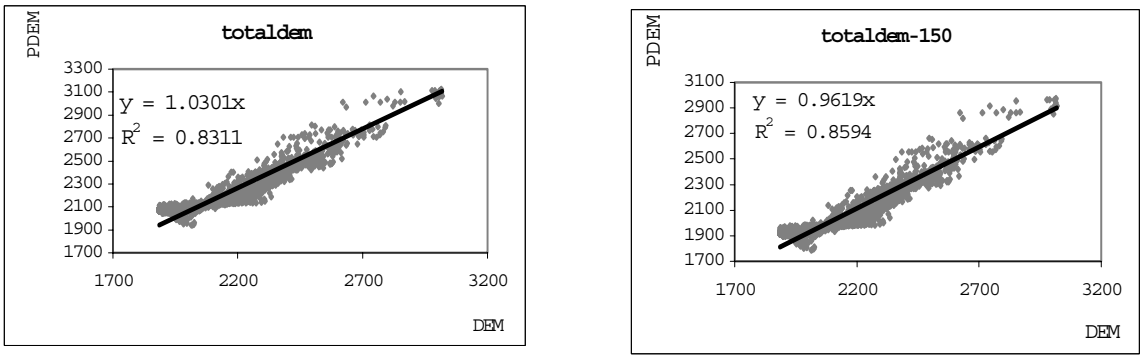


Fig. 2.5 scatter plot of the public domain DEM and the digitized-contour derived DEM.

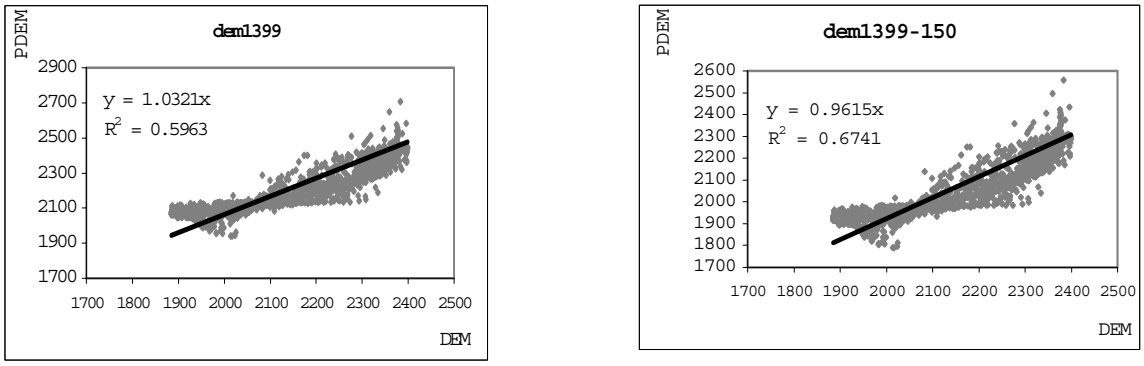


Figure 2.6 scatter plot of the public domain DEM and the digitized-countour derived DEM for altitudes up to 1399 meters

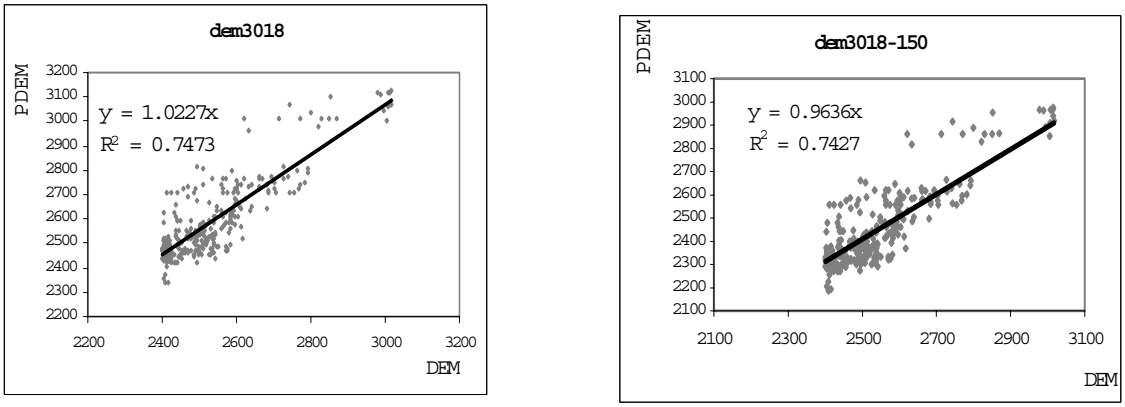


Figure 2.7 scatter plot of the public domain DEM and the digitized-contour derived DEM for altitudes from 1400 meters till 3018 meters.

As can be seen from the charts and the values of R^2, altitudes obtained by subtracting 150 meters from the public domain DEM are more correlated to the values of the existing DEM, which gives an indication that we can rely on the public domain DEM to an extent depending on how much accuracy we need. In our case the DEM will be used to create a threshold for the cloud detection process in the part of image processing of the time series of NOAA images. The accuracy is sufficient for this purpose, because the change in the temperature with altitude is about 1 ^{o}C every 150 meters, which means an error of 150 meters in the DEM amounts to only 1 ^{o}C in the value of the threshold. Therefore there is a negligible effect on the final results of the cloud detection algorithm.

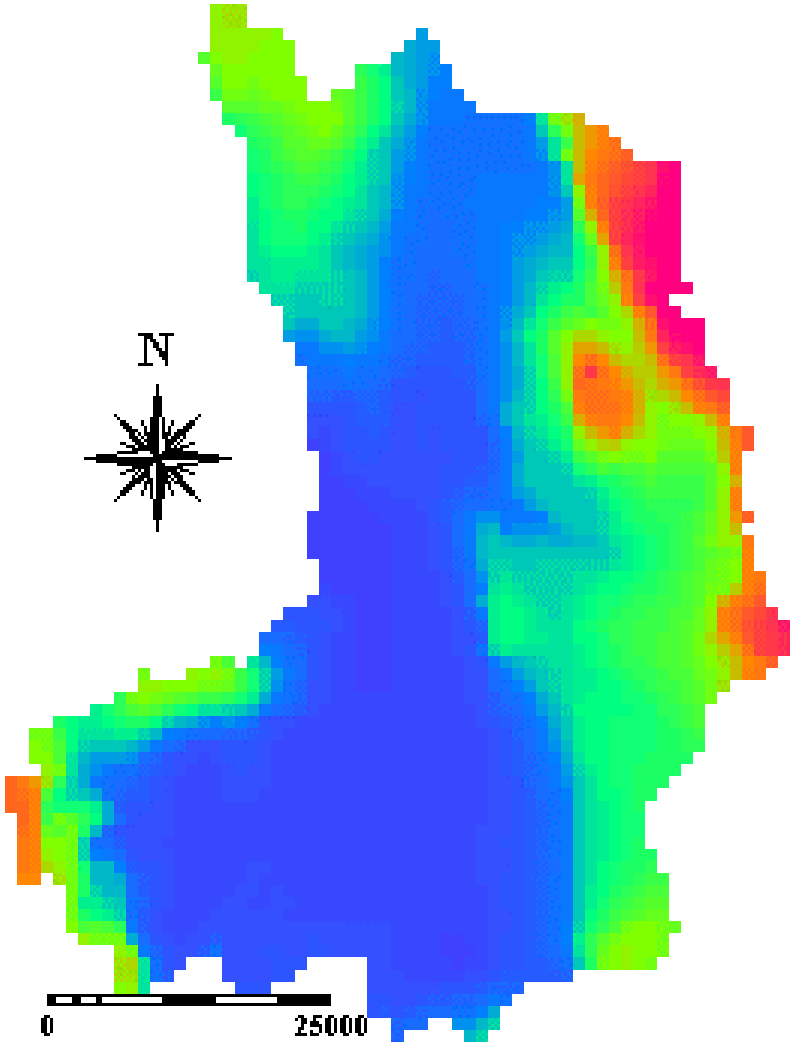


Figure 2.8 the public domain Digital Elevation Model

2.2 ANCILLARY DATA

Ancillary data for the study area have been used in this research. A Thematic Mapper satellite image dated January, 21st 1996, a terrain mapping unit map developed by Hamududu (1998), and meteorological data for rainfall and temperature data has been used. In the following paragraphs a description of those data is presented.

2.2.1 Thematic Mapper imagery

Channels 5, 4, 3 have been used to produce a color composite for land cover classification together with some ground truth data collected from the field. This land cover classification has been compared with last-year land cover classification and it has been found that they are almost the same. Channels 4, and 3 have been used to produce a Normalized Difference Vegetation Index (NDVI) image that was used to georeference the NOAA images.

2.2.2 RAINFALL DATA

For the study period, rainfall records for only four stations are available. The four stations are Naivasha W. D. (ID 9036032), North Kinangop forest station (ID 9036025), OlKalau railway station(ID 9036055), Gilgil Kwetu Farm (ID 9036029). The daily records have been aggregated to ten days for all the stations. Only monthly data was available for North Kinangop station, so to get the decade data for this station the monthly data was distributed according to the nearest station (Naivasha W. D.). The four stations are located in different altitudes, so they represent the overall rainfall in the catchment to some extent. The average monthly rainfall for the study period calculated from the four stations record was compared to the average monthly rainfall for long periods calculated from many stations. It was found that there are some variations because of the difference in the period length. The rainfall data processing is explained in chapter 3. Two methods for areal distribution of point rainfall data have been checked, the inverse distance and the Thiessen polygon. Values obtained by the inverse distance for the annual rainfall for the different classes were far below the minimum value of rainfall in the catchment. Values obtained by the Thiessen polygon were within the average but for the forest area, values of annual rainfall were below the average because of the limited number of stations (only 4 stations) and also there were no stations available close to the forest area. The altitude-rainfall relationship can not be established because of the limited number of rainfall stations and the short duration of the study period.

2.2.3 Temperature Data

Because of the lack of temperature data in the study area, a relation between altitude and monthly temperature has been used in this study. The relation between altitude and temperature has been quoted from a report by the Ministry of Agriculture and

Livestock Development. According to the report, the relations are based on data from 160 stations in Kenya. Data on absolute and mean, maximum and minimum, monthly and annual temperatures for the 160 stations are given in a publication of the East African Meteorological Department (EAMD 1970). Also the EAMD publication gives the equations relating the temperatures in Celsius (^{o}C) to the altitude in meters.

Table 2.4, which was quoted from the report, shows the equations for the different months and for the average, minimum, and maximum temperature.

	Mean maximum	Mean minimum	Average
January	36.7 – 0.00571 xm	25.0 – 0.00745 xm	30.8 – 0.00659 xm
February	37.4 – 0.00577 xm	25.8 – 0.00761 xm	31.6 – 0.00669 xm
March	37.6 – 0.00577 xm	26.3 – 0.00755 xm	31.9 – 0.00666 xm
April	35.8 – 0.00574 xm	26.1 – 0.00705 xm	30.9 – 0.00643 xm
May	34.6 – 0.00587 xm	25.1 – 0.00682 xm	29.9– 0.00633 xm
June	33.9 – 0.00584 xm	23.8 – 0.00666 xm	28.9 – 0.00627 xm
July	33.5 – 0.00600 xm	22.8 – 0.00636 xm	28.2 – 0.00617 xm
August	33.7 – 0.00610 xm	23.8 – 0.00646 xm	28.5 – 0.00627 xm
September	35.4 – 0.00604 xm	24.1 – 0.00705 xm	29.7 – 0.00653 xm
October	36.4 – 0.00614 xm	24.8 – 0.00712 xm	30.6 – 0.00663 xm
November	36.2 – 0.00643 xm	25.2 – 0.00709 xm	30.7 – 0.00676 xm
December	35.5 – 0.00594 xm	25.1 – 0.00725 xm	30.3 – 0.00656 xm

Table 2.4 regression equations for estimating the monthly mean, maximum, minimum and average temperature in (^{o}C) from the altitude in meter (m)

Verifying the calculated air temperature

Air temperature values calculated using the above regression equations were partly verified using recorded air temperature from unofficial station situated at UTM coordinates (37M,0190815,991968). The data records were taken in the period from May 1998 till January 1999. The average monthly values for every month were calculated from temperature records at the time of the satellite overpass. The correlation between the two averages was found to be 0.94. As can be seen from Table 2.5, the difference between the two values is acceptable, which means that the calculated average maximum monthly values using the regression equations is reliable.

Month	May	June	July	Aug.	Sep.	Oct.	Nov.	Dec.	Jan.
Calculated	**22**	**21**	**20.7**	**19.7**	**23.3**	**24**	**22.7**	**28**	**27.4**
Recorded	**22.4**	**21.5**	**21**	**21**	**22.8**	**23.6**	**22.6**	**24**	**25**

Table 2.5 Comparison between average monthly values for recorded and calculated air temperature

References:

Bakker, W. H., Parodi G. N., and Timmermans W. J. (1997). NOAA AVHRR preprocessing: An Application Of a Cloud-Detection Technique. *ITC Journal 1997-1.*

Eidenshink, J. c., and Faundeen, J. L. (1998). The 1-KM AVHRR Global Land Data Set: First Stages inImplementation Internet Website http://edcwww.cr.usgs.gov/landdaac/1km/paper .html#proc2

Gutman, G. G.(1991). Vegetation Indices from AVHRR: AN Update and Future Prospective. *Remote Sensing of the Environment 35:121-136.*

Hamudud, B. H., (1998). Erosion Assessment for Large Basins using Remote Sensing and GIS: a Case Study of Lake Naivasha, Kenya. *ITC's MSc. Thesis.*

Kidwell, K. B., (1991). NOAA Polar Orbiter Data Users' Guide, National Oceanic and Atmospheric Adminstration, World Weather Building, Room 100, Washington D.C.

Price, J. C., (1987). Calibration of Satellite Radiometers and The Comparison of Vegetation Indices. *Remote Sensing of the Environment, 21, 15-27.*

Holben, B. N., Kaufman, Y. J., and Kendall, J. D., (1990). NOAA-11 AVHRR visible and near infrared Inflight Calibration*: The International Journal Of Remote Sensing, 11, 1511-1519.*

Rao, N. C. R. (1987). Pre-Launch Calibration of channels 1 and 2 of Advanced Very High Resolution Radiometer: *NOAA Technical Report NESDIS 36, Satellite Research Laboratory, Natonal Environmental Satellite, Data, and Information Service, Wash*ington, D.C.

Teillet, P. M., (1990). Rayleigh optical depth comparisons from various sources: *Applied Optics, 28, 1897-1900.*

Teillet, P. M., (1991). Radiometric and atmospheric correction procedures for AVHRR preprocessing in the solar reflective channels. *Proceedings of the Fifth International Colloquium on Spectral Signatures of Objects in Remote Sensing, Courchevel, France, 1991, 511-516.*

Teillet, P. M., and Holben, B. n., (1994). Towards Operational Radiometric Calibration of NOAA Imagery in the Visible and Near-Infrared Channels: *Canadian Journal of Remote Sensing, 20, 1.*

CHAPTER 3

DATA PROCESSING

Different sources of noise exist in satellite images and this noise can affect the interpretation of the images to a great extent. Before starting the analysis phase of any study this noise should be removed to a degree depending on the accuracy required and the purpose of the analysis. Some of the sources of noise are clouds, atmosphere, soil background (especially in studying vegetation indices), inaccurate georeferencing (especially when dealing with sequence of images in time series analysis) because shifts in the position of pixels may lead to inaccurate results especially at the boundaries of different units, and image resolution. One more source of errors lies in the application of classification criteria. In this chapter processing of the data series of images has been carried out to remove the effect of the different types of noise and prepare the data for the analysis.

Most of the phases of the image processing part and noise removal have been implemented using programs written in C programming language. Data were prepared for the program using ILWIS 2.2 software. The different functions of this program and the preparation of the data procedure are explained in the following paragraphs.

Approaches that have been used in this study to deal with the different sources of noise and noise removal techniques are explained in the following paragraphs. The choice of the noise removal technique depends to a great extent on the characteristics of the available data. As an example, water bodies in the set of images have been masked, which made it impossible to use them in the atmospheric correction phase. Moreover the low resolution of the NOAA Images hindered the possibility of using techniques such as the radiometric scene normalisation using pseudo-invariant features.

3.1 IMAGE CLASSIFICATION

As has been mentioned above, the criteria and the accuracy of the classification that is used to divide the study area into classes which expresses the objective of the research is of great importance in the interpretation of the results phase. In this research the classification criteria was based on dividing the study area into Terrain Mapping Units in which each unit has its own landform with different attributes of geology, soil, and land cover. The Terrain Mapping Units map (Figure 3.1) was developed by Hamududu, (1998). Table 3.1 shows the different Terrain Mapping Units and their attributes.

Map Unit	Lithology	Land cover
Mountain	Pyroclastic rocks, Maiela Pumice	Forest
Volcanic Complex	Olkaria Comendite, Lava Flows,	Shrubs , vegetation
Scarp	Basalts,Older Tuffs	Shrubs, vegetation
Volcanic Plateau	Tertiary Volcanic Rocks	Agriculture, Grass
Foot Slope	Colluvium (Igneous)	Shrubs, agriculture
Volcanic Plain	Alluvium, Maiela Pumice	Agriculture crops
Upper Lucastrine Plain	Lucastrine Sediments	Irrigated Agriculture
Lower Lucastrine Plain	Lucastrine Sediments	Natural vegetation
Lava	Stony, Rock Land	Bare rock, n. vegetation
River Valley	Alluvial Deposits	Agriculture crops
Water	Alluvium	water

Table 3.1 Terrain Mapping Units TMUs (Hamududu, 1998)

3.2 RESPONSE VARIABLES IMAGES DERIVATION

The response variables mentioned in Chapter 1, the Normalized Difference Vegetation Index (NDVI), the Soil Adjusted Vegetation Index (SAVI), surface reflectance (albedo), and Surface Temperature, were derived from the available downloaded channels of NOAA-AVHRR data by using the map calculation functions of ILWIS2.2 software. Map lists for the time series of images were prepared for further processing with the C-program. The steps followed to derive the response variables images are described in the next sections.

3.2.1 The data base of images

The database of images, which were used to derive the response variables images consists of the following:

A. The downloaded images from Internet are raw images, which means that they are scaled images with values previously mentioned in chapter 2. NOAA-AVHRR channels downloaded are the following:
   1- the visible channel (reflectance)
   2- the near infrared channel (reflectance)
   3- the brightness temperature (channel 4)
   4- the brightness temperature (channel 5)
   5- the Normalized Difference Vegetation Index "NDVI" (channel 6)

The file names of images follow a certain convention, which expresses the date and the channel number of the image. For example the file name "febf9304" means that this file is for an image of channel 4 and for the first decade of February 1993.

B- Using ILWIS2.2 map calculation functions and the downloaded tabulated values mentioned in chapter 2, the actual value images were derived from the downloaded scaled ones (raw images). The file naming convention for the actual images data set is explained as:
"afef9304" means the actual value image for the first decade of February 1993.

Figure 3.1 Terrain Mapping Units map

The next step is to calculate the response variables images from the actual value images.

To construct a continuous time series of images, the gap exists for October, November, Decimber 1993, and January 1995 has to be covered. Decade NDVI images for these months were created for these months using the maximum value from the same decade and the same month available from the pervious and next year. Images for channels 1, 2, 4, and 5 were created by assigning the corresponding value of the maximum NDVI for the same channel to the pixels in the images. Also the 1994-year gap was covered by shifting back the 1995 images to 1994.

3.2.2 Response variables images

Some of the response variables images were calculated using ILWIS2.2 software e.g. NDVI and SAVI, and others like the surface temperature and the broad band albedo have been calculated for every pixel using the C-program during the processing phase of the program for the time series. It was preferred to calculate these images in the program to facilitate the modification of the coefficients used in the equations.

The Soil Adjusted Vegetation Index (SAVI)

The Soil Adjusted Vegetation Index images (SAVI) have been calculated using the following formula:

SAVI = (channel 2 – channel 1) * 1.5 / (channel 2 + channel 1+ 50) (1)

The value 50 has been used instead of 0.5 because values in the channel 2 and channel 1 images are reflectance in percentage. The output images file names follow the following convention:
"svifef93" which means SAVI for the first decade of February 1993.

The Normalized Difference Vegetation Index (NDVI)

The Normalized Difference Vegetation Index images (NDVI) were calculated using the following formula:

NDVI = (channel 2 – channel 1) / (channel 2 + channel 1) (2)

The output images file names follow the following convention:
"ndvfef93" which means NDVI for the first decade of February 1993.

Air Temperature Images

Decade air temperature maps were calculated using the Digital Elevation Model (DEM) and the formula developed by the Ministry of Agriculture and Livestock Development. Those maps were used later in the cloud detection

function in the C-program to calculate a variable threshold for the Gross cloud detection test, which will be described latter.

3.3 GEOMETRIC CORRECTION

As has been mentioned before in chapter 2, the images in the data set were geometrically registered. However, to combine the data set images with data sets from other sources, the images had to be georefernced again. In recognizing ground control points with images of low resolution, as is the case of NOAA images, it is proved only possible to use lakes in the study area like Lake Naivasha and Lake Nakuru as ground control features. The master image for image to image geoeferencing was a TM image available in the ITC. The following procedure has been adopted:

- A NDVI image was calculated from TM channel 4 and channel 3 images of January 1996 using ILWIS2.2 map calculation formula.
- The TM NDVI image was aggregated to approximately the same pixel size as a NOAA image using the average function to simulate the way by which NOAA sensors imaging land surfaces.
- For the same period, January 1996, a NOAA NDVI image was calculated.
- The NDVI NOAA image and the aggregated TM NDVI image were spectrally compared for specific features, which could be easily recognized in both images.
- Those features were Lake Naivasha and Lake Nakuru.
- Spectrally similar pixels were identified in the two images and used to georeference the new image.

Pixel Position shift Verification

One of the sources of noise in studying time series of images is the possible shift that could happen in the position of the pixels. This shift causes noise in time series analysis because the shifted pixel in that case does not represent the same ground position in the time dimension. Actually this problem is not so critical when dealing with low resolution images like NOAA, since in NOAA images we are aggregating large areas anyway. However this problem should be solved because it affects pixels located at the boundaries between different land cover classes.

The idea to determine the amount of shift in the position of pixels is to choose two transects crossing the lake and testing the boundary of the lake at the two transects. One of these transects is in the east-west direction and the other in north-south direction. The two transects were chosen to cross Lake Naivasha at the sections of biggest depth near the lakeshore. This is to avoid the possibility that the recorded shift (if any) was caused by variations in lake level area. The two transects were chosen according to the contour lines of the lake bed level (Figure 3.2). A C-program has been designed for this purpose. A map list of the time series of images for SAVI (Soil Adjusted Vegetation Index) was prepared using ILWIS2.2 software as an input to the program.

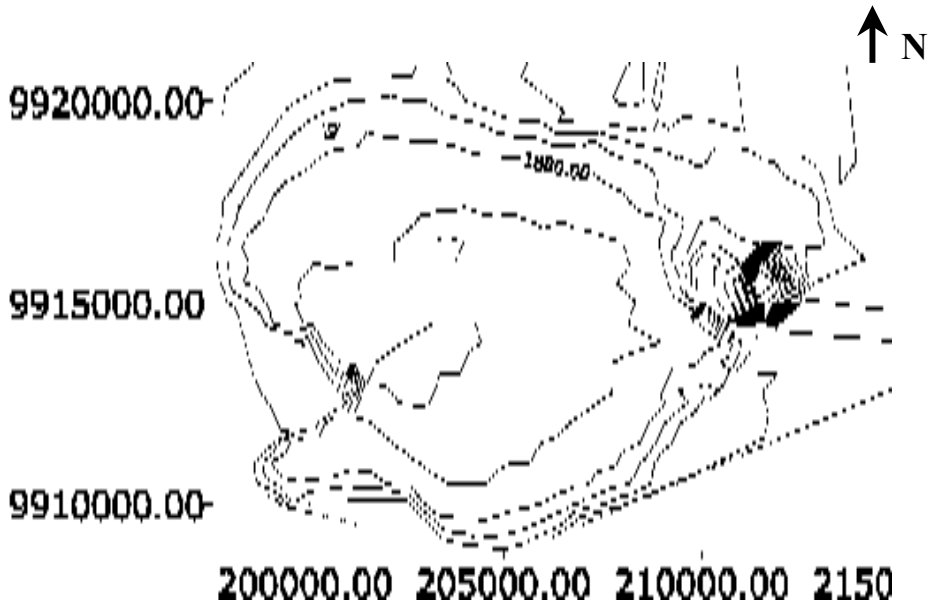


Figure 3.2 Contour lines derived from the Echo-Sounding survey of the Lake Naivasha

The pseudo code for the program is as follows:

- read the map list file
- Open the first file in the list
- Read pixel values for the chosen transect
- Store the pixel values in a two dimensional array
- Repeat the above mentioned steps for the entire images in the map list
- Every row in the two dimensional array represents the pixel values in the transect in one date
- The two-dimensional array is then converted to an image which contains the time series of the transect, including the lake.
- The output of the program is a time series image for the transects in the east-west direction and the north-south direction

By analyzing the East-West image (Figure 3.3), it is clear that during the period of study the eastern edge of the lake is straight. Therefore there was no shift in the pixels in the eastern part of the lake. The extension on the western side is not a shift because there is no corresponding shift in the eastern direction. This may be explained by the masking technique used for water bodies in the data set.

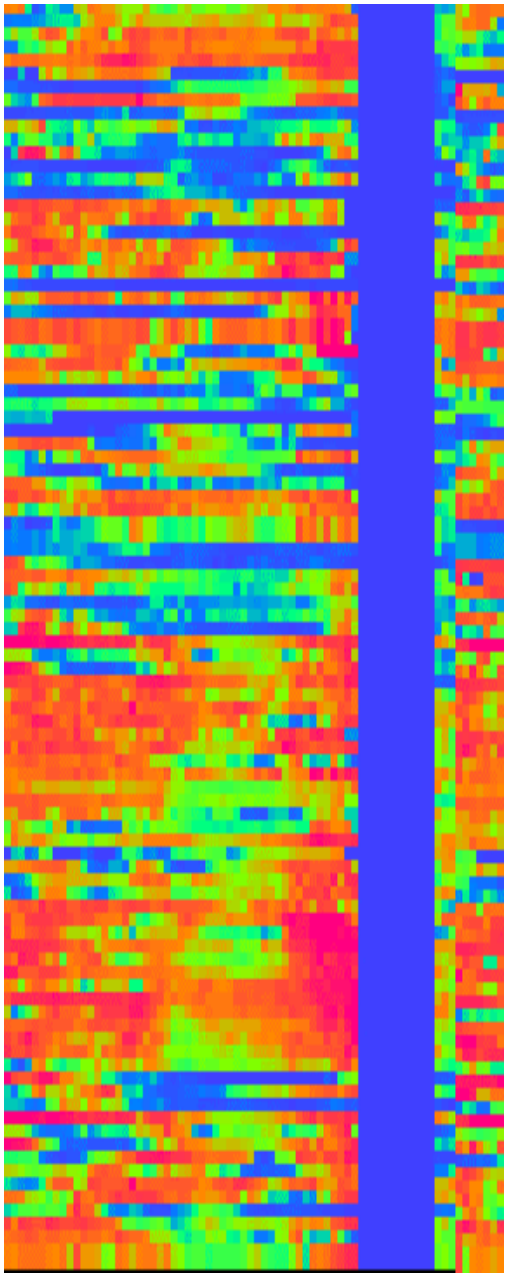


Figure 3.4 North-South transect through the lake, time period Jun-92-Sep-93, Feb-95-Jan-96.

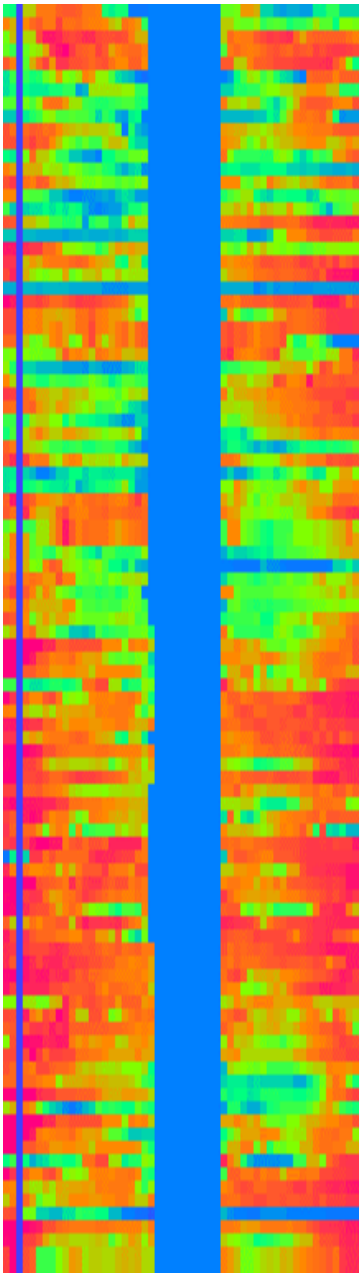


Figure 3.3 East-West transect through the lake, time period Jun-92-Sep-93, Feb-95-Jan-96.

By analyzing the image series North-South direction (Figure 3.4), it is clear that through this transect there is neither shift in the lake nor extension or shrink in the lake area. The two images justify that there is no shift in the pixels in the study area with time, which means that the value we get for any pixel represents the value of the same position in the ground for the whole period of the time series. Our analysis is based on the average value of every class, which guarantees that the effect of noise due to unrecognized shift or due to mixed pixels at the boundary of the different classes can be neglected.

Masking the images to the study area

After georeferencing the images, as will be explained, and by using the same coordinate system for the different maps, it was possible to combine the derived images with the existing maps of the study area. By using ILWIS2.2 map calculation functions and a raster map of the boundary of the catchment masked images of the response variables images were calculated. The new images contain only data within the catchment. Submaps of the masked images were derived to decrease the size of the image file.

Map lists for the response variables masked submaps were created which contain the maps arranged in time order for the time series-processing phase.

3.4 CLOUD CONTAMINATION

To use NOAA_AVHRR images in extracting information about the land surface characteristics and to make estimates of surface variables such as land surface temperature and vegetation indices, cloud free pixels have to be identified (Saunders and Kriebel, 1988). For the study area in Kenya there are two rainy seasons and hence the chance of having cloud contaminated pixels is high, which constitutes a problem in studying the time series of images of the surface variables. In this research a cloud detection algorithm developed by Saunders et al. (1988) was used followed by another screening for the supposedly cloud-free pixels using the mean and the standard deviation for every class. A cloud contamination noise removal algorithm was also applied for the vegetation index time series to try to remove the rest of the noise.

3.4.1 Cloud Contamination Detection

First of all cloud contaminated pixels have to be identified. Various cloud detection algorithms have been developed for NOAA_AVHRR data, which can be divided into three main categories (Bakker et al., 1997*)*:

- statistical method (relying on histogram analysis)
- threshold techniques (applied to each pixel)
- pattern recognition methods (based on the analysis of large-scale texture)

The technique that is used in this research, is the " threshold technique " developed by Saunders and Kriebel, (1988). Some of the problems encountered in applying cloud detection methods relate to the differences in climatic regions and underlying surface types. It is easier to detect clouds over rather

homogeneous surfaces than over heterogeneous land surfaces (Bakker et al., 1997). This is because the value of the threshold depends on the topography in the sense that high altitude areas have a lower threshold than lower altitude areas. Land cover also affects the chosen threshold value. The cloud detection algorithm, developed by Saunders and Kriebel (1988), is based on five tests. Only if all tests are negative the pixel is identified as cloud-free (Figure 3.5).

The first test (Gross Cloud Check) uses an infrared threshold for the 12 μm brightness temperature (channel 5). If the temperature is below this threshold, the pixel is flagged as cloud contaminated.

The second test suggested is the spatial coherence test, which relies on the fact that cloud top radiances often vary over small areas, so this test is suitable for large uniform areas. The third test is the dynamic reflectance threshold test, which relies on determining the cloud-free reflectance peak, and assigning a threshold value higher than this peak. Channel 1 is used in this test for land surface because of the low reflectance of the different land covers. The fourth test relies on the fact that the ratio between the near-infrared reflectance (channel 2) and visible reflectance (channel 1) for clouds is close to unity. Over land with growing vegetation the reflectance increases markedly at near-infrared wavelengths compared to shorter wavelengths (Swain and Davis, 1978), so the ratio is expected to be greater than one. Determination of the threshold value depends on the pixel histogram values. If there is a clear cloud-free peak the threshold value is chosen higher by 0.2 than the cloud-free peak. Due to the large variability over land there is no well-defined peak so in this case a default threshold of 1.4 is set where all pixels with a value of the ratio between the infra-red channel and the red channel (R2/R1) less than 1.4 are assumed to be cloud-contaminated (Saunders and Kriebel, 1988). The fifth test looks at the difference between the recorded brightness temperature in channels, 4 and 5 which is higher for clouds than for land surfaces (Bakker et al., 1997)

In this research the gross cloud test, which uses channel 5 brightness temperature and the near-infra red reflectance (channel 2) to visible reflectance (channel 1) ratio test have been used. The spatial coherence test has not been used since it is used over sea only. Neither has the thin cirrus test because it requires data about the total column of water vapor, which is not available for the study area. The dynamic reflectance threshold has not been used because of the difficulty of using it with many images in an automated manner and also the channel 2-channel 1 ratio test is supposed to be enough.

The cloud detection function in the C-program has been designed to detect the cloud contaminated pixels according to the selected steps suggested by Saunders and Kriebel, (1988), which are the Gross cloud test and the ratio between channel 2 and channel 1 test.

Gross cloud test

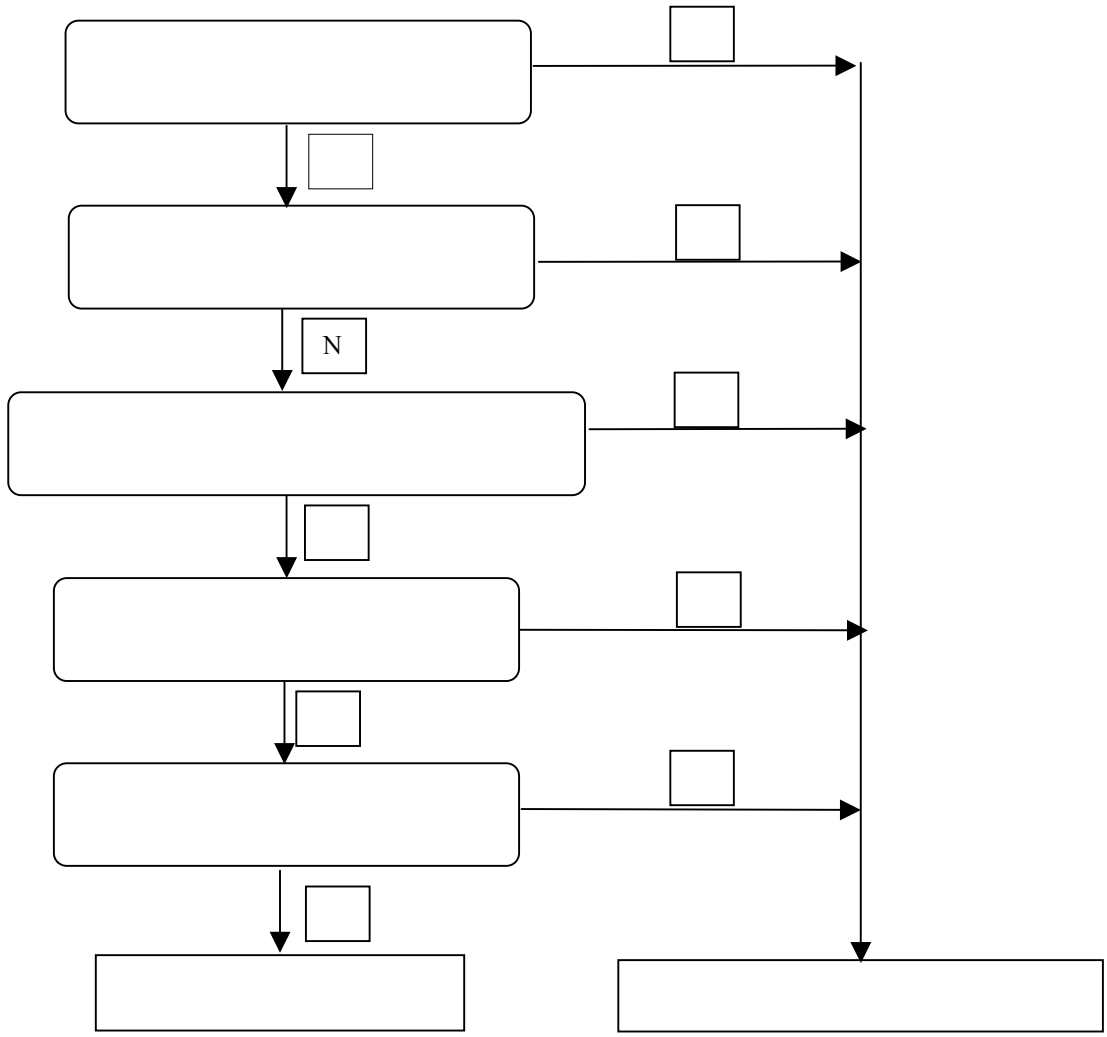
Because the study area is a mountainous area special care has to be taken when selecting the threshold, because the decrease in air temperature with altitude must be taken into account to avoid misclassifying cold cloud-free pixels

as cloud contaminated. A non-constant threshold has been adopted in this research. The variable threshold was determined based on the following:

1- DEM of the study area
2- A relation between altitude and temperature for the different months of the year. The relation that has been applied here is the relation between the mean maximum temperature and the altitude. The mean maximum temperature has been adopted since the satellite overpasses the study area at 2:30 after noon and at this time the air temperature is supposed to be around the maximum. If there is a deviation from this assumption it could be compensated for in the threshold. The relation between the altitude and the temperature has been mentioned in Chapter 2.
3- The threshold is determined by subtracting a value of 6 degrees from the air temperature calculated in step 2.

A map list for the time series of channel 5 brightness temperature images for the catchment area has been prepared using ILWIS 2.2 for the cloud detection process function, including the entire time series.

Figure 3.5 Outline of the algorithm applied to AVHRR data for daytime to detect cloud contaminated pixels. Adapted from (*Saunders and Kriebel, 1988*)



Where:

T5 : channel 5 brightness temperature
T4 : channel 4 brigthness temperature
NIR/VIS : the ratio between infra-red (channel 2) and visible channel (channel 1)
ALB : the albedo of the visible channel
ALB_{thr} : the threshold Albedo
SD :the standard deviation of a 3*3 pixelarray of channel 4
SD_{thr} : the threshold standard deviation

The NIR/VIS ratio test

The second test for cloud detection, which has been applied in this research, is the ratio between the infra-red and visible channels (NIR/VIS) test. Images of the ratio NIR/VIS has been derived using ILWIS 2.2 for the entire time series from channel 1 and channel 2 images. The histogram of a sample of the ratio images has been checked, and a threshold of 1.4 was selected for the study area. This value of the threshold was selected for two reasons, the first one is that there is no too much variability in the land cover of the study area as the majority of the area is covered with vegetation and almost there is no bare soil, so according to Saunders who suggested a value of 1.6 if there is a large variability in the land cover, a value less value than 1.6 has been selected. The second reason is that in studying time series some of the images in the series have clear cloud peak and the others have no cloud peak which suggests a value higher than 1.2, which was suggested by Saunders for clear-cloud peak image histograms.

A map list of the time series of the ratio images has been prepared and was the input for the cloud detection function together with the map list of channel 5 brightness temperature. Every pixel in the response variable images is checked for cloud conatmination, and the pixel, which does not satisfy the conditions of the two tests ie. pixels whose brightness temperature below the brightness temperature threshold or whose NIR/VIS ratio below the ratio threshold is flagged as cloud contaminated.

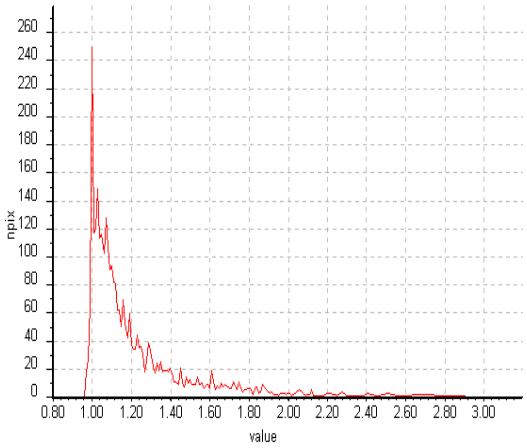


Figure 3.6 Histogram for (NIR/VIS) for the middle decade of Nov 1992

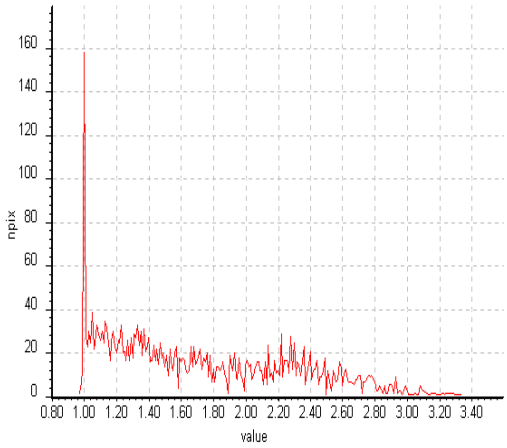


Figure 3.7 Histogram for (NIR/VIS) for the second decade of Jun. 1993

The output of the cloud detection function is new time series of images, which is supposed to include only cloud-free pixels.

3.4.2 The Mean and Standard Deviation screening

TO remove the redundant noise caused by the atmosphere and the haze, another screening algorithm was applied for the Surface Temperature, Vegetation Index and Surface Albedo for every class in the study area. The different steps of the algorithm have been implemented using a C-program function with the following pseudo code:

- The input for the function is the supposedly cloud-free pixels for every class.
- The average and the standard deviation of the remaining pixels are calculated.
- Every pixel in the class is restricted to vary within the standard deviation of the means of the different response variables. The threshold is as follows:

$$| \langle X \rangle - X | < \text{std}(X) \qquad (3)$$

where:

<X>is the mean for every class
X is the value of the pixel
std(X) is the standard deviation of the class.

- The average of the Pixels that passes the threshold test is calculated for every class for the above mentioned response variables.
- The output of the function is the input for the smoothing function of the Vegetation Index response variable. But for the surface temperature and Albedo the output of the function is the input for the time series developing function.

This screening procedure was suggested by Gutman (1991). It is assumed that this procedure with the cloud detection and cloud contaminated pixels removal algorithm in the previous section, eliminated most of the cloud and haze contaminated pixels in the data set.

3.4.3 The Spatial Resampling of the Cloud Contaminated Pixels

To assign values for every pixel and to generate images which are supposed to be cloud free, a 3 by 3 pixels window filter has been run in the whole time series of images. The advantage of generating such cloud free images is that they can be used later in the Surface Energy Budget Algorithm for Land, which runs on a pixel by pixel basis. The algorithm for this function is as follows:

- The 3 by 3 window runs line by line starting from the second line and the second column.
- The centered pixel of the window is checked if it is flagged or not
- If the pixel is not flagged it remains unchanged and the next pixel is checked
- If the pixel is flagged, the neighbouring 8 pixels are checked.
- The pixel is assigned the value of the first non-flagged pixel.

- If all the 8 neighboring pixels are contaminated, the pixel is assigned the average value of the class to which it belongs.
- The above algorithm is applied to the whole time series of the response variable images.

The Performance of the Cloud Detection Algorithm

The performance of the cloud detection algorithm has been checked by the following steps:

- An image has been generated by applying the cloud detection and the spatial filter algorithm for the NDVI image of the last decade in January 93.
- The generated image has been visually compared with the original image (Figures 3.10, 3.11).
- The histograms of the original and the generated image have been compared (Figures 3.12, 3.13).

By visual comparison of the two images it can be verified that almost all the cloud contaminated pixels have been removed either by using a value from the neighboring pixels (if clouds do not cover a large area) or by using the average value of the class (in case the clouds cover a large contiguous area). The average percentage for every TMU in the catchment were calculated (Figure 3.8), and also the time series of the percentage cloud cover for the whole catchment based on a weighted average of the area of every unit were developed (Figure 3.9)

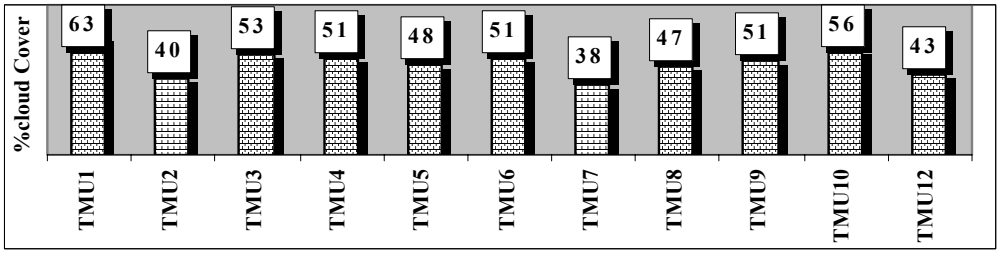


Figure 3.8 Average% cloud cover for the dufferent TMU's in the catchment

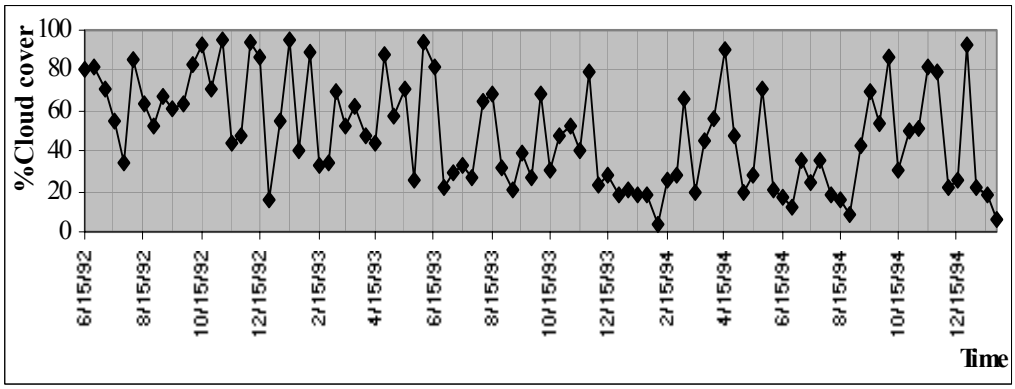


Figure 3.9 Time series of %cloud coverfor the whole catchment

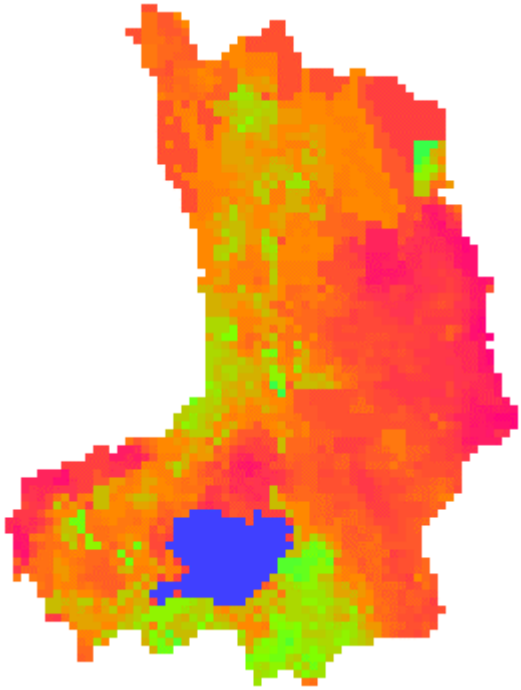


Figure 3.10 NDVI image after applying the cloud detection and the spatial filter

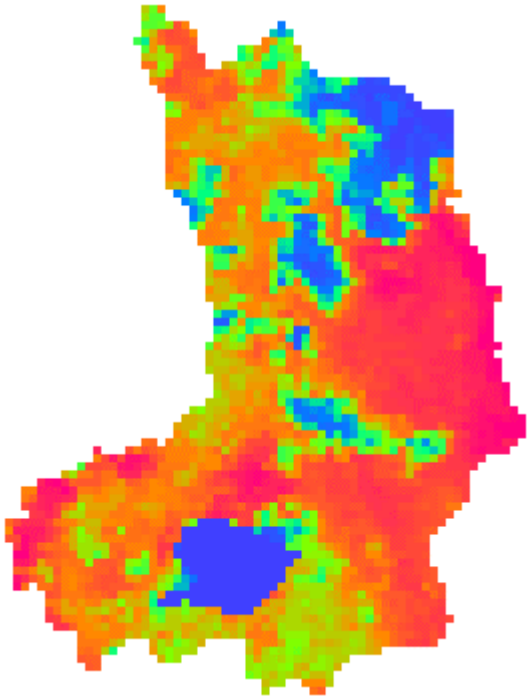


Figure 3.11 NDVI image before applying the cloud detection and the spatial filter

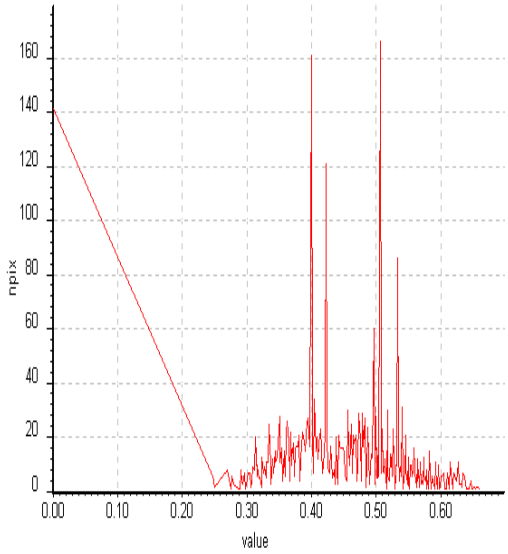


Figure 3.12 Histogram for the corrected NDVI image

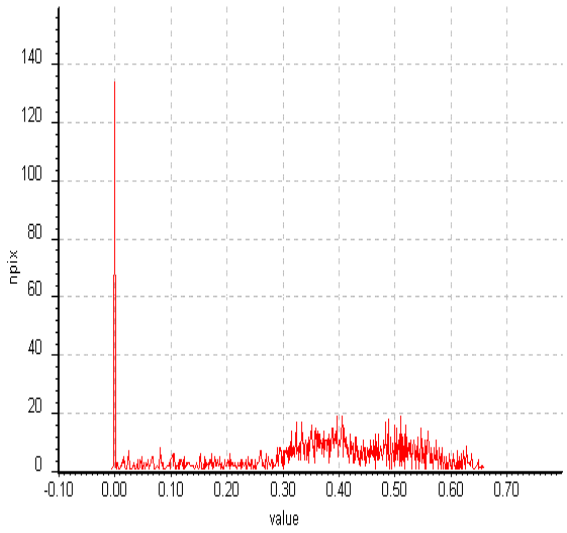


Figure 3.13 Histogram for the non-filtered NDVI image

3.5 ATMOSPHERIC CORRECTION

One of the most complex factors, which affect the signal received by the sensor, is the atmosphere. This complexity stems from the high spatial and temporal variability of constituents of the atmosphere, which cause attenuation of the signal and consequently affect the signal-to-noise ratio. Most of the approaches suggested to address variable atmospheric attenuation of surface observations require either knowledge of the meteorological variability of the atmosphere parameters, or knowledge of the observed surface reflectance (Goward et al., 1991).

The constituents of a cloudless atmosphere may affect a signal by scattering and absorption. Scattering can either increase or decrease the channel signal, whereas absorption can only decrease the signal. The value of any derived response variable like vegetation indices can either decrease or increase, according to the magnitude of the attenuation in each channel used to derive the vegetation index.

The constituents in a cloudless atmosphere that affect NOAA-AVHRR data are air molecules, which cause Rayleigh scattering, oxygen, ozone, other trace gases, water vapor, which cause absorption, and aerosols, which cause both scattering and absorption (Holben, 1986). As has been mentioned in chapter 2 in the characteristics of the collected data set, the data has been corrected for two principal atmospheric factors, which are ozone and Rayleigh scattering. Water vapor and aerosol are highly variable in time and space, as such causing the greatest variation in atmospherically attenuated AVHRR short-wave data. Holben, (1986) has reported that water vapor in the atmosphere decreases the near-infra red (NIR) channel response, which is shown to be eight per cent of the signal response, assuming complete absorption. He also simulated the effect of changes in water vapor in the Normalized Difference Vegetation Index (NDVI) and he found out that the overall effect of water vapor is to decrease the NDVI by 0.02 units.

Aerosols also lower the NDVI. Holben, (1986) reported also simulations from Dave radiation transfer models, show that a change in optical thickness owing to aerosols from 0.1 to 0.5 decreased the NDVI by 0.12 units in the extreme backscatter direction and by 0.06 units in the nadir. Other simulations show that extreme off-nadir viewing reduces NDVI of different class types and that soil and vegetation classes have higher NDVI values than water and clouds for a particular atmospheric composition.

As mentioned above there are many approaches for atmospheric correction. Some of these approaches rely on the availability of temporal and spatial values of aerosols and water vapor. Other approaches employ techniques, which are used specially for multi-temporal comparisons. These techniques approximately eliminate the atmospheric effects by comparing the digital numbers of ground features that have little spectral variations in time (Meijerink, et al., 1994).

Since there is no data available for the temporal and spatial distribution of aerosols and water vapor for the study area during the period of the study, because of the low resolution of the NOAA images in the sense that identifying spectrally invariant-with-time features in the image could not be done, and because water bodies have been masked in this data set, other techniques for radiometric corrections were employed in this research.

According to Gutman, (1991), atmospheric effects (clouds, water vapor, and aerosol) reduce the contrast between the visible and near-infrared reflectance, thus usually decreasing the vegetation indices. Therefore a straightforward way to select the clearest observation is by taking the maximum of vegetation indices over a period of time. Based on Gutman, this is applicable for the data set used in this research, which is a ten-day composite series.

The method that was used in this research has been devised by van Dijk et al. (1987). This method reduces the noise-to-signal ratio found in remotely sensed data without precise knowledge of the underlying interactions (V*an Dijk et al., 1987*). The filtering technique of the response variable is based on the assumption that reflectance X(t) is the sum of signal S(t) and noise N(t), where S(t) and N(t) are independent of each other (Parzen, 1967; Tukey, 1977) as cited by Van Dijk et al. (1987). The noise N(t) is known to consist of very high frequency components compared to the signal S(t). Smoothing is an attempt to reduce the amount of noise and make the signal the main component of the curve. For example, in studying vegetation index profiles, the vegetation index, which reflects the green-up of vegetation during the growing season has frequency waves of several weeks and can be regarded as signal components of the profile (Van Dijk et al., 1987). The radiometric disturbance caused by atmospheric attenuation and viewing angles and other sources is changing frequently and consequently adds a high frequency component or noise to the signal.

Two techniques for filtering have been used in this study. The first one is the compound smoothing. As cited by Van Dijk et al., (1987), Velleman and Hoaglin (1981) developed the “compound smoother” “4253H, twice”. The number in the name of the smoother indicates the spans used. The 4253H compound smoother first filters the data using a running median of span four, then two, then five and three. The “H” stands for “Hanning.” The Hanning (or weighting system) filter used by Velleman and Hoaglin (1981) consisted of the weights 0.25, 0.5, and 0.25. After filtering the data four times using the running median method with the various spans, the data are filtered using the running weighted mean method with the above weights. Generally, running median filtering smoothes a data sequence too much and removes patterns of interest (Van Dijk et al., 1987). Velleman and Hoaglin (1981), as cited by Van Dijk et al. (1987), overcame this problem by smoothing the residuals and adding the results to the smoothed data sequence. The second smoother uses Fourier analysis.

In this research, a developed C-program function for noise removal has been used. The following steps were taken to prepare the input data for the function using ILWIS 2.2:

- Terrain mapping units map (tmu1.mpr)

- A map list for the time series of images for the response variable (NDVI)

The pseudo code for the function is as follows:

- The program opens the classified map file
- The program determine the class to which the pixel belongs
- Opens the file of the response variable of the first image in the series
- Picks the value of the same pixel from the response variable image file
- The pixel is checked if it is flagged as cloud contaminated or not
- If the pixel is not flagged as cloud contaminated its value is added to an accumulator variable and a count of the number of pixels of that class is incremented. If the pixel is flagged as cloud contaminated it is excluded and not added to the accumulator.
- The average value for each class is calculated for that first image in the series.
- The above mentioned steps are repeated for every class and for the entire time series.
- The output of the function is a smoothed and noise-removed time series of the average of the cloud-free pixels of every class in the land cover classified map.

To compare the effect of processing techniques in the time series profile and noise removal process, time series profiles for the following cases have been generated using the C-program. A sample for three mapping units is shown in Figure 3.13.

- A time series for a selected pixel from every class.
- A time series for the average value of every class without applying the cloud contamination function.
- A time series for the average value of every class after applying the cloud contamination function
- A time series for the average of the class after applying the cloud detection function and applying the spatial filter and excluding the outliers using the standard deviation algorithm.
- A time series for the average value of every class after applying the cloud contamination function and applying the “4253h, twice” smoother

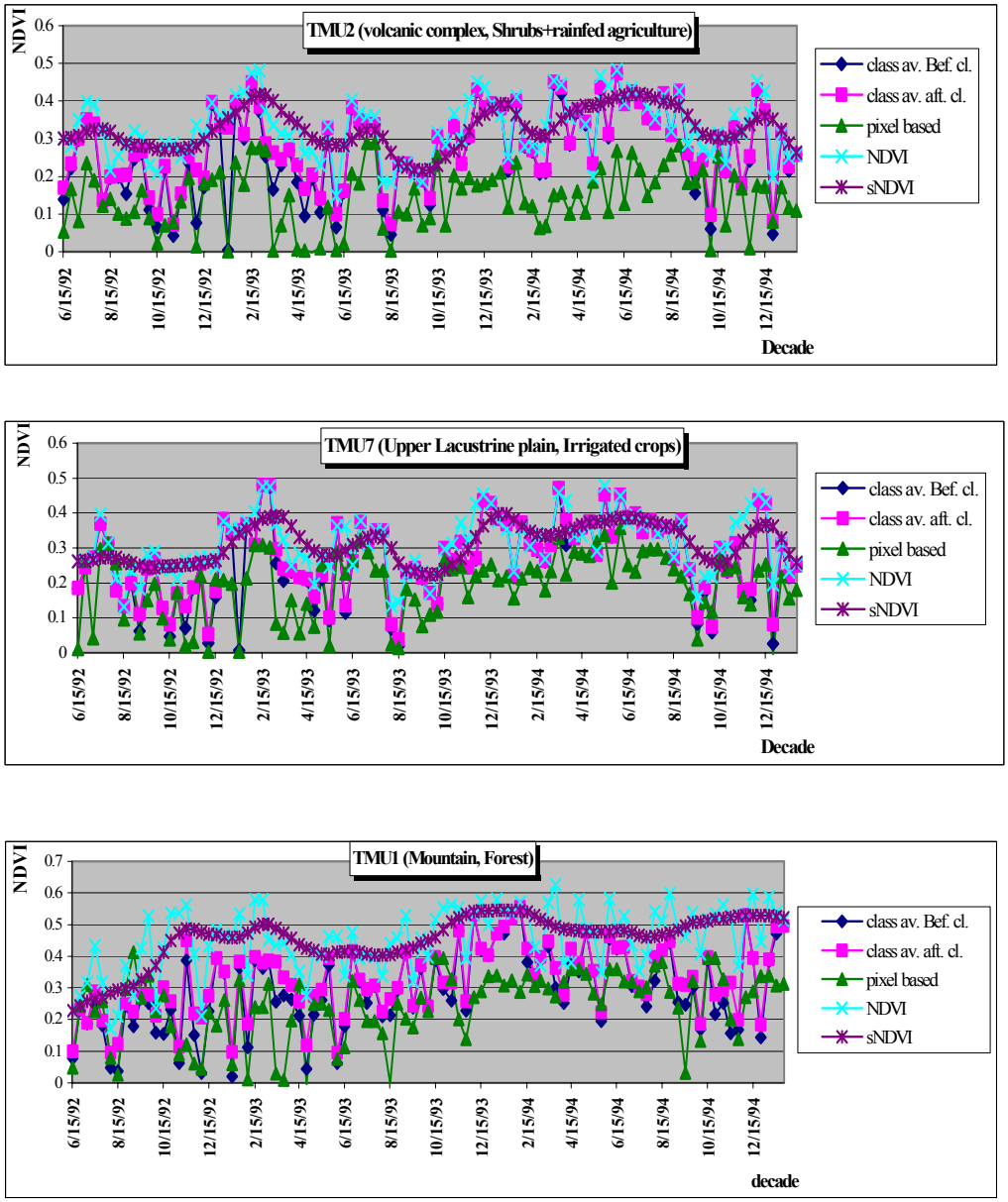


Figure 3.14 Comparison between time series before and after data processing

WHERE:

- Class av. Bef cl. is the class average before applying the cloud detection algorithm
- Class av. Aft. Cl. is the class average after applying the cloud detection algorithm
- Pixel based is the time series for a selected pixel from the class
- NDVI is the time series for NDVI after applying the standard deviation threshold for the class
- sNDVI is the smoothed NDVI time series using the "4253H,twice" filter.

3.6 SURFACE TEMPERATURE

As has been mentioned in chapter 1, the split window technique is used in this research to derive surface temperature. This technique uses channels 4, 5 in NOAA AVHRR series. The split window technique is based on the differential absorption of water vapor continuum inside the atmospheric window 10.5 – 12.5 μm (Caselles et al., 1994). This technique has been applied first for the sea surface temperature with the assumption that sea surface is behaving as a black body i.e. emissivity equals to unity. In the split window channels natural surface emissivities are close to unity, but there is some spectral variation which depends on the surface type. So the best method for correcting thermal land surface temperature seems to be the adaptation of the split window technique derived for sea surface temperature accounting for both atmospheric and emissivity effects. (Caselles, et al., 1994).

In this research two techniques have been used for comparison and to help choosing the most suitable one for the study area. Both were developed by Caselles et al. (1994).

They also developed the quadratic split-window algorithm in NOAA data set. This nonlinear algorithm takes into account the atmospheric variability on a global scale. The proposed equation is as follows:

$$T=T_4+[a_0 +a_1 (T_4 - T_5)](T_4 – T_5) + B(\varepsilon) \quad (4)$$

Where:
T is the surface temperature
T_4 is the channel 4 brightness temperature
T_5 is the channel 5 brightness temperature
a_0 , a_1 are coefficients of the regression
$B(\varepsilon)$ is the emissivity dependent parameter.

$$B(\varepsilon) = \alpha(1- \varepsilon 4) - \beta\Delta\varepsilon \quad (5)$$

α and β are parameters, which depend on the atmosphere type and the surface temperature.

They also suggested values for the parameters in the equation, which were derived from different simulations of different atmospheric conditions and different land cover types. Values of the different coefficients of the equation are stated in Table 3.1. Caselles has derived parameters for α and β for the HAPEX-Sahel area (13-14N; 2-3E), in which he suggested to use α equal to 40 K and β equal to 30 K, which have been adopted in the present study because they are suitable for the study area.

Case	A0	A1	α	β
Mid latitude winter	1	0.58	50	150
Mid latitude summer	1	0.58	50	75
Tropical	1	0.58	40	30

Table 3.2 coefficient values for the Split Window Technique

Emissivity values have been calculated based on the following equation, which has been derived by Van de Griend et al. (1993):

$$\varepsilon = a + b * \ln(NDVI) \qquad (6)$$

The equation has been derived for semi arid area and for savana environment, which coincides with the study area. The value for (a) is 1.0094 and for (b) is 0.047. an assumption has been made for the value of emmisivity in band 4 that it is equal to broad band emmisivity and the difference in emmisivity between channel 4 and 5 is equal to –0.004.

The second equation is a linear one derived tested in Spain. The equation was used here because of the semi-arid environment similarity between the study area and the area in which the equation has been tested. The equation is as follows:

$$T = T_4 + 2.13(T_4 - T_5) + .18 + 50(1 - \varepsilon_4) - 200\,\Delta\varepsilon \qquad (7)$$

Where:

T is the surface temperature

T_4 is the channel 4 brightness temperature

T_5 is the channel 5 brightness temperature

ε_4 is the emmisivity in band 4

$\Delta\varepsilon$ is the difference in emmisivity between channels 4 and 5

The value of ε_4 can be assumed to be equal to the broad band emmisivity and $\Delta\varepsilon$ equal to –0.004.

So values for surface temperature using the above mentioned equations for the different classes in a decade basis for the study period have been calculated based on the filtered mean value of NDVI, channel 4 brightness temperature, and channel 5 brightness temperature using the C-program.

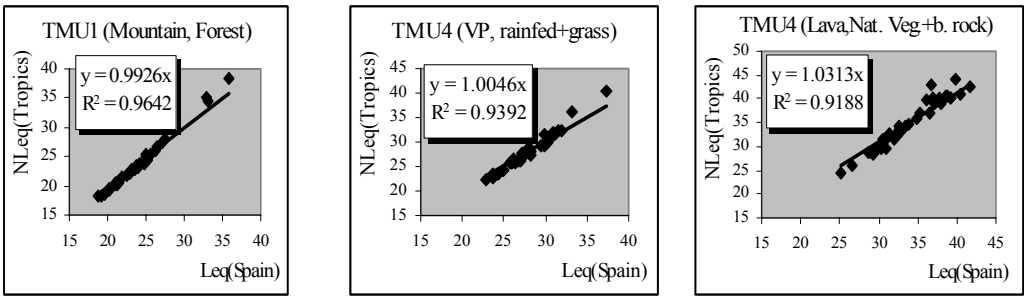


Figure 3.15 Correlation between the two approaches for calculating Surface Temperature

Since there are numerous equations for the split window technique, the correlation between the two selected equations for the different classes was checked to verify the use of one of them and the accuracy of the surface temperature estimation. Three Terrain Mapping Units were selected to represent the catchment. The first one is the mountainous area with forest land cover, the second is the Lava with natural vegetation and bare rocks land cover, and the third one is the volcanic plateau with rain-fed agriculture and grass land cover. From figure 3.15, it is clear that there is a high correlation between the two methods in the three units. Correlation between the two equations was studied for the different classes in the catchment and it was found to be within the values obtained for these three classes. Based on this, surface temperature from the global equation was used in this research.

3.7 RAINFALL DATA

To construct the decade time series of rainfall for the whole classes the following processing steps were taken:

- From the daily data records, the decade records were calculated
- The decade rainfall records for the four stations were stored in four text files to be used later as inputs to the rainfall time series generation function in the C-program
- A point map for the four stations was created and then rasterized.
- A Thiessen polygon map was created.
- Then the Thiessen polygon map was resampled to the same pixel size of NOAA as the rest of the images for overlaying processes, the resampled map was masked to the catchment area and a final submap of the masked map was created.

After preparing the Thiessen polygon map and the rainfall files, C-language code was developed to read the rainfall data and assign decade rainfall records to every pixel in the different classes, which was aggregated for the different classes in the catchment. The pseudo processing steps are as follows:

- Read the rainfall records text files and store the records in an array for every polygon (station).
- Through an overlay process the map, which contains the classes was combined with the Thiessen polygon map.
- For every class the total number of pixels which belongs to the class and to one of the polygons is determined.
- The decade rainfall for every class is calculated as a weighted average of the polygon decade rainfall in which the class is falling.
- Store the decade rainfall records in a two-dimensional array for further processing.

References:

Bakker, W. M., Parodi, G. N., and Timmermans, W. J., (1997). NOAA AVHRR Preprocessing: An Application of a Cloud-Detection Technique. *ITC Journal 1997-1.*

Caselles, C. C., Sobrino J. A., and Valor, E. (1994). On the Atmospheric Dependence of the Split Window Equation for Land Surface Temperature. *International Journal of Remote Sensing, 15: 1, 105-122.*

Gutman, G. G., (1991). Vegetation Indices from AVHRR: An Update and Future Prospects. *Remote Sensing of Environment. 35:121-136*

Holben,B. N. (1986). Characteristics of Maximum Value Compsite Images From Temporal AVHRR DATA. *International Journal Of Remote Sensing, 7: NO. 11, 1417-1434.*

Saunders, R. W. and Kriebel, K. T. (1988). An Improved Method for Detecting Clear Sky and Cloudy Radiances from NOAA-AVHRR Data. *International Journal of Remote sensing 9: 1,123-150.*

Schott, J. R., Salvaggio C., and William J. V. (1988). Radiometric Scene Normalization Using Pseudoinvariant Features. *Remote Sensing of Environment 26:1-16.*

Van Dijk, A., Susan L. C., Clarence M. S. and Vayne L. D., (1987). Smoothing Vegetation Index Profiles: An Alternative Method for Reducing Radiometric Disturbance in NOAA AVHRR Data. *Photogrametric Engineering and Remote Sensing, 53:8, 1059-1067.*

Parazen, E., (1967). Time series Analysis Papers. *Holden-Day Inc., 500 Samsome St., San Francisco.*

Tukey, J. W., 1977. Exploratory Dataanalysis. *Addison-Wesley publishing company, Reading, Massachusettes.*

Velleman, P., and Hoaglin, D. C., (1981). Applications, Basics and Computing of Exploratory Data Analysis. *Duxbury Press, Boston, Mass.*

CHAPTER 4

RESULTS AND DISCUSSION

This chapter consists of three parts. In the first part the temporal and spatial consistency of the output results of the individual response variables for the different Terrain Mapping Units (TMU's) is discussed. Secondly, a pair-wise consistency is applied by studying the interrelationship between the response variables spatially and temporally. Thirdly, the hydrological behavior of the different TMU's is qualitatively assessed. Hydrological zonation of the catchment is made based on the analysis of the time series of a surface temperature-NDVI ratio together with other information and knowledge about the study area. Fourier analysis was used to analyze the NDVI time series, since this is a periodic function in time. For surface temperature and albedo, in spite of proving that they follow a seasonal pattern, but because of the high frequency behavior of those two variables due to their immediate response to instantaneous events like rainfall, the analysis was based on statistical values for the different TMU's and for the rainy and dry seasons.

4.1 Individual Consistency Test

The response variables, the Normalized Difference Vegetation Index (NDVI), the surface temperature and the surface reflectance are tested individually in space and in time to check their behavior and consistency.

4.1.1 The Normalized Difference Vegetation Index (NDVI)

The dynamic behavior of vegetation for the different classes was studied using Fourier analysis to decompose its time series into the main components. The amplitude and phase of these main components were then used to analyze the dynamic behavior of the vegetation. It was noticed for almost all the TMU's that six and seven-month components have been detected. This might be explained by the short period of the time series, which, to some extent, could be affected by climatic changes from one year to another, for example the onset of the rainy season, which in turn affects the rainfall pattern during the short study period. The analysis has been done for a combination of the six-month and seven-month components into one component ("A" component). The amplitude values in the graphs were not normalized by the number of observations. The use of the phase angle to determine the time at which the Fourier components reach the highest value is explained in Annex C. The results are explained hereafter.

TMU1 (Forest area in the mountain terrain-mapping unit)

For the forest area the average NDVI was found to be 0.45. The main component was the one year component (Figure 4.1) but this one year component can not be reliable because the length of the time series is short (2.5 years). To some extent the annual component detects the annual variability, which means that there is a mono-modal pattern. The "A" component exists, which means that there is a bimodal growth pattern related to the two rainy seasons. The magnitude of the amplitude of the "A" component is low with respect to the 11-month component.

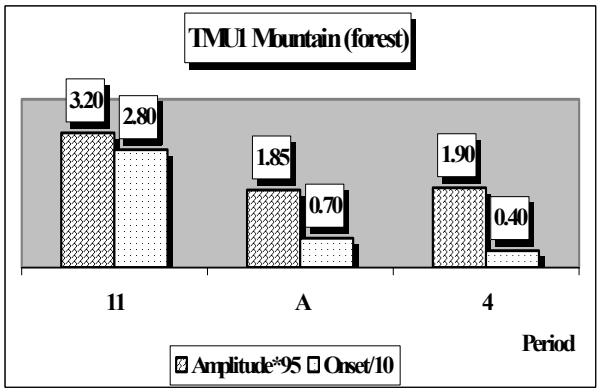


Figure 4.1 Amplitude and Onset for the NDVI of TMU1

TMU2 (volcanic complex, shrubs+vegetation)

The average NDVI value is 0.32, which is relatively low. The main component is the "A" component related to the rainfall pattern in the area. There is also a four-month component, which can be explained by some agriculture activities in this TMU. The phase was detected to be only 7 decades later from the start of the time series, which is almost the middle decade of August (Figure 4.2).

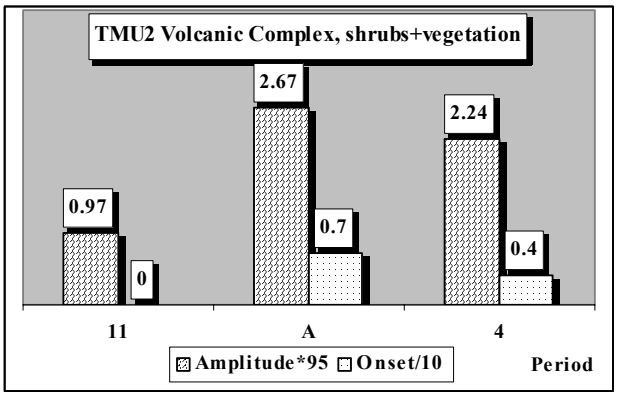


Figure 4.2 Amplitude and Onset for the NDVI of TMU2

TMU3 (scarp, shrubs+vegetation)

The average NDVI is 0.43, which is relatively high. This is because this area is close to the mountainous area, which receives the highest amount of rainfall. There are two components the "A" and the four-month components. The "A" follows the bimodal rainfall pattern. The phase for the "A" component is 7 decades, almost the middle of August, which means that NDVI value reaches the highest value by that

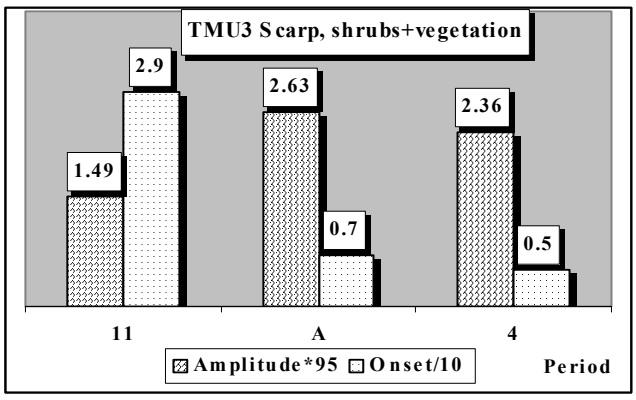


Figure 4.3 Amplitude and Onset for the NDVI of TMU3

time. The four-month component can not be explained because the Scarp is steep, and therefore not many agriculture activities by farmers would be expected (Figure 4.3).

TMU4 (volcanic plateau, agriculture+grass)

The average NDVI value is also high (0.42). The "A" and the four-month components are dominant. The explanation seems to be that the small-scale plots of agriculture, which caused the four-month component, rely on distributed boreholes in the area. Onset for the "A" component is seven decades from the beginning of the time series, which means at the middle decade of August. Onset for the four-month component is three decades earlier than for the "A" component, which seems reasonable, because agriculture crops reach the highest NDVI values earlier than vegetation or grass (Figure 4.4).

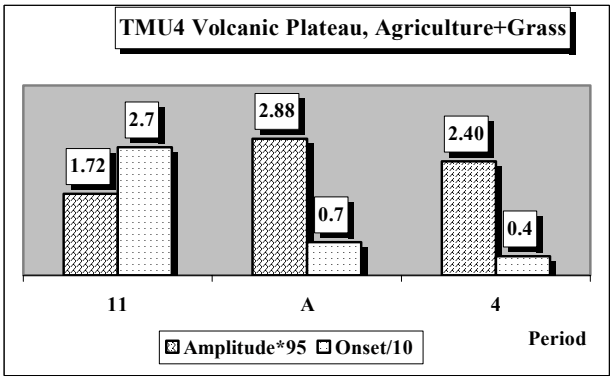


Figure 4.4 Amplitude and Onset for the NDVI of TMU4

TMU5 (foot slope, shrubs+agriculture)

The average NDVI value is 0.38, which seems high in view of the fact that this area is the foot slope area of the mountains, where the amount of rainfall is expected to be high. The two main components are also the "A" and the four-month components (Figure 4.5). For the four-month component the agriculture activities in the area do not rely too much on rainfall but rather on some shallow wells and on small streams from the mountains. Onset is seven decades for the "A" component (mid of August) and four decades for the four-month component (the mid of July).

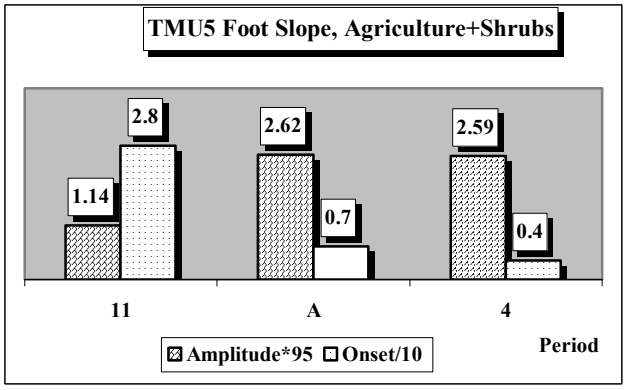


Figure 4.5 Amplitude and Onset for the NDVI of TMU5

TMU6 (volcanic plain, vegetation)

The average NDVI value is relatively high (0.36). The main component is the "A" component (Figure 4.6). It seems that this area is affected too much by the seasonal rainfall variations. Part of this unit which is located close to Eburu forest, is used for planting wheat, using rainfall. This may explain the dominance of the "A" component and the high response to the rainfall pattern. Onset is almost seven decades from the beginning of the time series, which means middle of August.

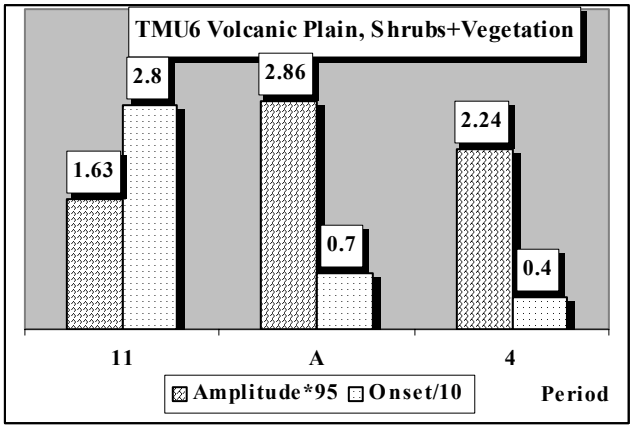


Figure 4.6 Amplitude and Onset for the NDVI of TMU6

TMU7 (upper lacustrine plain, irrigated crops)

The average NDVI is (0.31). The main components are the "A" and four month components (Figure 4.7). The four-month component was detected because of the agriculture activity, which relies on the irrigation from the lake. The "A" component may be explained by the existence of some pixels, which were misclassified as irrigated areas. This unit belongs to the natural vegetation areas around the irrigated farms.

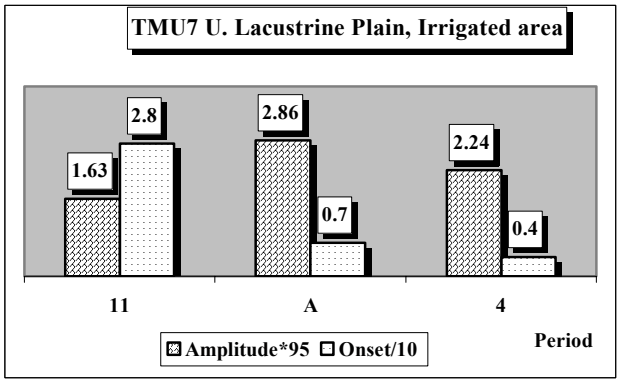


Figure 4.7 Amplitude and Onset for the NDVI of TMU7

TMU8 lower Lacustrine plain (natural vegetation)

The average NDVI value is moderate (0.37). This area is located around the lake. The most dominant component is the "A" component (Figure 4.8). The amplitude value is relatively high, which can be explained by the high dependence of that area on rainfall as the main source of water. Onset is also seven decades

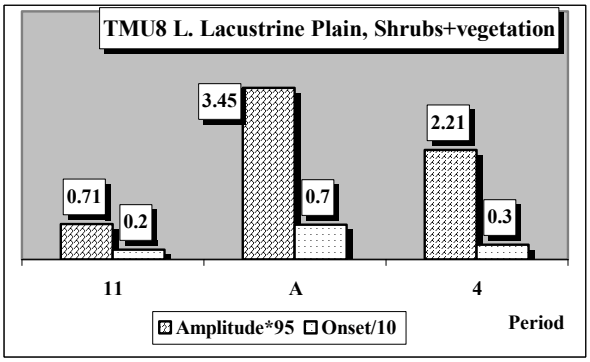


Figure 4.8 Amplitude and Onset for the NDVI of TMU8

from the beginning of the time series.

TMU9 (lava, natural vegetation+bare rock)

The average of this class is low (0.31) because of the presence of outcrops. The main component is the "A" component for the dependence of natural vegetation on the rainfall. The existence of the one-year component (if to be considered) seems to be reasonable for this area, which can be affected by the annual variation in rainfall like vegetation in desert areas. Although this area is located close to the mountainous area, which is covered with forest and receives the highest amount of rainfall, the average NDVI is low, which gives some indication about the accuracy of the classification. Onset for the six-month component is seven decades (Figure 4.9).

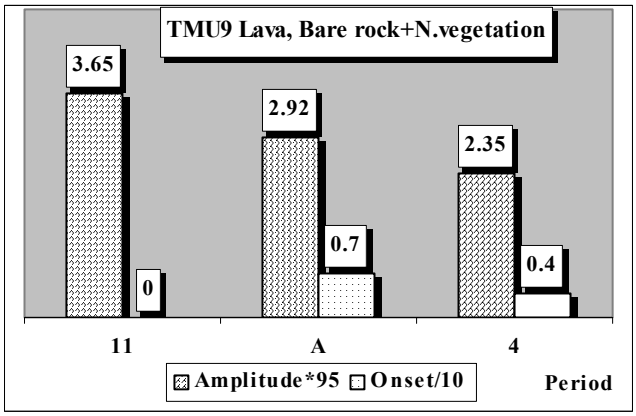


Figure 4.9 Amplitude and Onset for the NDVI of TMU9

TMU10 (river valley, agriculture, grass)

The average NDVI value is high (0.41). The main component is the "A" component and this is for rain-fed agriculture on the slopes of the rivers and for natural vegetation (Figure 4.10). The high value of the amplitude indicates the high dependence of vegetation in this area on rainfall. The four-month component is for some agriculture plots near the river. Onset for the six-month component is also seven decades (middle of August).

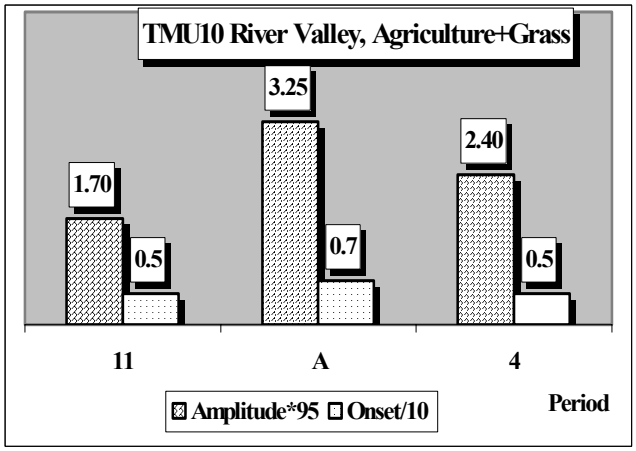


Figure 4.10 Amplitude and Onset for the NDVI of TMU10

TMU12 (volcanic plain1, natural vegetation)

The average NDVI is low (0.3), which is reasonable for natural vegetation. The main component is the "A" component, which is relatively very high. This also can be explained by the high dependence of natural vegetation on rainfall. Onset for this component is seven decades, which means middle of August (Figure 4.11).

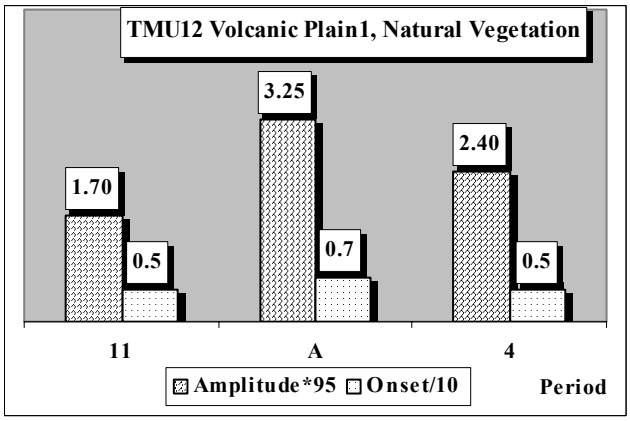


Figure 4.11 Amplitude and Onset for the NDVI of TMU12

Comments

It can be concluded that most of the Terrain Mapping Units land covers in the catchment follow a bimodal rainfall pattern, which characterizes the catchment area. The existence of a four-month component in some classes was explained and it seems reasonable based on the information available from the fieldwork, but some detected components can not be explained. The one-year component exists as a dominant one for the forest area and for the bare rock and natural vegetation area. Phase also seems to be reasonable since most of the land covers reaches its highest value almost by the end of the rainy season. The output of the NDVI time series appears to be consistent and can be used for further analysis.

4.1.2 The Surface Temperature

It was mentioned in chapter 2 that the data set has been recorded by two satellites, NOAA 11 and NOAA 14. It has been found that there is a difference between channel 4 and 5 values in the two data sets. Values recorded by NOAA 14 tend to give higher values than values recorded by NOAA 11. This may be explained by the different calibration coefficients or by the difference in the time of the satellite overpass, since for NOAA 11 the time of the satellite overpass over the equator is 2:30 but for NOAA 14 is 1:30, which my cause a difference in temperature.

The consistency of the calculated temperature is tested based on the spatial and temporal pattern. The spatial consistency is tested with respect to the land cover of the Terrain Mapping Unit (TMU). The temporal consistency is based on the pattern of surface temperature with respect to the wet and dry seasons in the study area from the time series of surface temperature for the different units.

Spatial pattern of surface temperature

Values of the average surface temperature for the different TMU's together with the land cover are shown in Figure 4.12. As can be seen the distribution of the temperature exhibits high correlation with the land cover. For example, the lowest temperature is found in the high mountainous area with forestland cover, and the

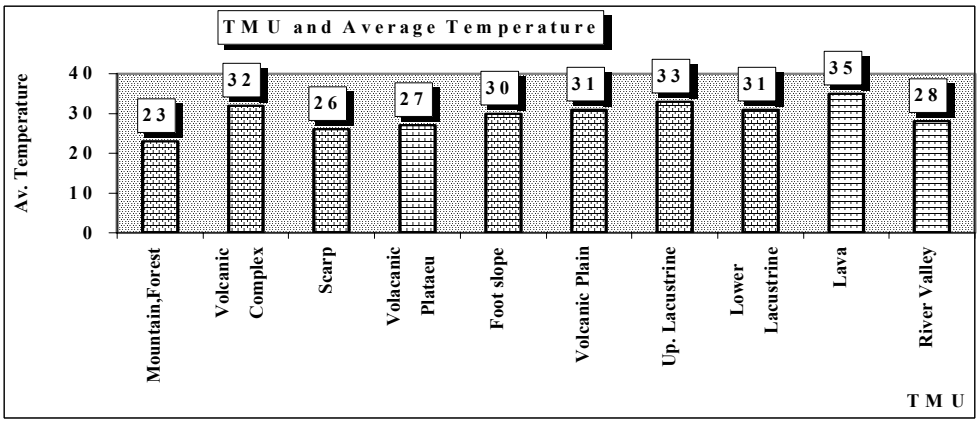


Figure 4.12 TMU's Average surface temperatures

highest temperature is in the Lava area where bare rocks show up with sparse vegetation cover. Also the other values seem to be reasonable like in the river valley and in the volcanic plateau, which are supposed to be low. Values in the irrigated area also seem to be reasonable if we consider that the calculated temperature is an aggregated average of many plots varying from bare soil to plots in the mature stage of crops passing through plots with early stage growing crops. The correlation between surface temperature and land cover implicitly includes other factors like altitude and amount of rainfall, which is highly correlated to altitude. For example, forest is located in high altitude areas, which receive relatively high amount of rainfall, which in turn decreases the average value of surface temperature. On the other hand, lower altitude areas receive less rainfall and consequently we can expect higher values for the average surface temperature. Also as has been mentioned in Chapter 2 on the relation between altitude and air temperature, air temperature decreases with altitude. The spatial correlation between surface and air temperature has been investigated for one decade in the dry season and

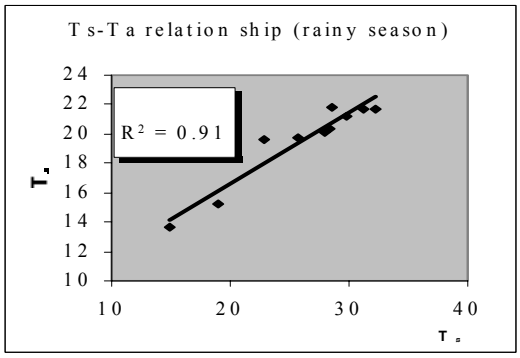


Figure 4.13 Surface temperature air temperature relationship

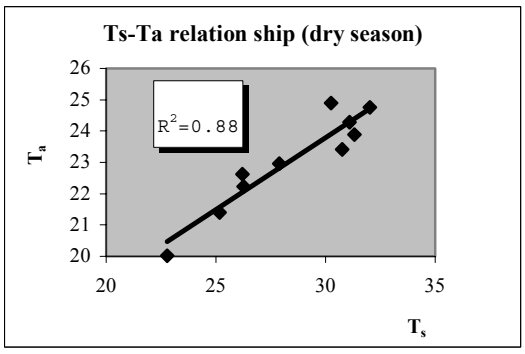


Figure 4.14 Surface temperature air temperature relationship

another one in the rainy season as is shown in Figures 4.13 and 4.14. From the graphs it is clear that there is a high correlation between surface temperature and air temperature.

Temporal pattern of surface temperature

From Figure 4.15, it is clear that surface temperature follows a seasonal pattern for the different Terrain Mapping Units since we find low temperature values in May, June and July, whereas we find high temperature values in January, February and March. Figure 4.15 shows the average temperature for the different Terrain Mapping Units for both the rainy and dry seasons.

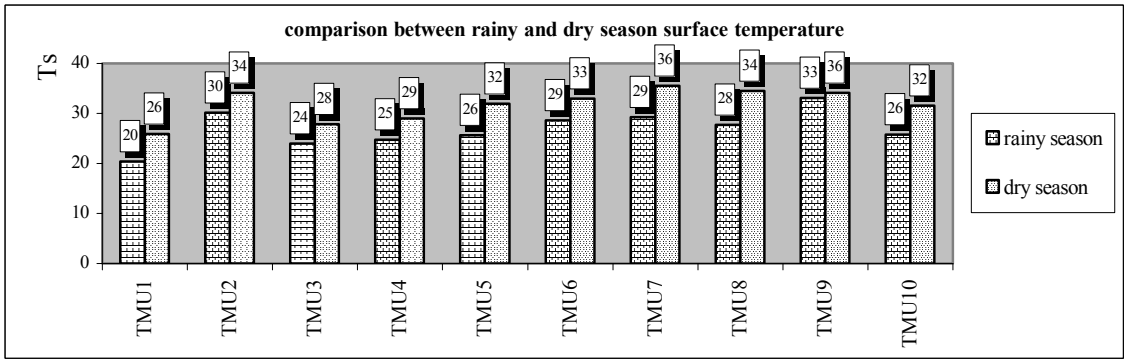


Figure 4.15 Average surface temperature in the dry and rainy seasons comparison

Comments

From the above discussion it seems that the output information regarding surface temperature is logically acceptable, reasonable and spatially and temporarily consistent with both the Terrain Mapping Units including the different characteristics of each TMU and seasonal pattern respectively. The accuracy of the values of surface temperature depends to a great extent on the model used to calculate it. In this research a comparison has been made between the two models, the linear one and the nonlinear global one developed by Caselles et al (1994). It was found that the values almost close to each other with small differences. At this stage it can be said that it is possible to rely on the obtained values of surface temperature for further calculations in evapotranspiration models.

4.1.3 The Surface Reflectance (Albedo)

Surface reflectance is an instantaneous value that is affected by many factors like, among others, surface wetness, canopy structure and land cover. So surface albedo depends also on the season and on the phase of vegetation growth. The consistency of the calculated surface albedo is tested based on the spatial and temporal pattern, as was done for surface temperature.

Spatial pattern of surface albedo

The average annual value for surface albedo for the different classes ranges between 17% to 20%. The lowest value was for forest area (17%) and it is reasonable compared to values in literature. Also the average value for the Lava TMU was found to be the lowest, which appears to be correct since this area is located in high areas receiving more frequent and higher rainfall. In this area the fraction of exposed soil is high so when it rains the soil becomes wet and consequently surface albedo becomes less. Average annual values for the different classes are shown in Figure 4.16.

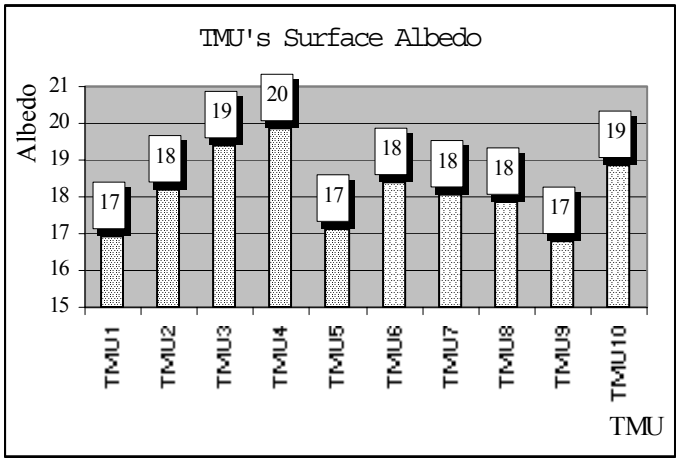


Figure 4.16 Average annual albedo for the different TMU's

Temporal Pattern of Surface Albedo

Average values of surface albedo for the rainy and dry seasons are shown in figure 4.17. From the graph we can see that the average value for the rainy season is lower than the dry season for all the classes. And this seems to be reasonable since rainfall is one of the factors that affects the surface reflectance or in other words the surface wetness.

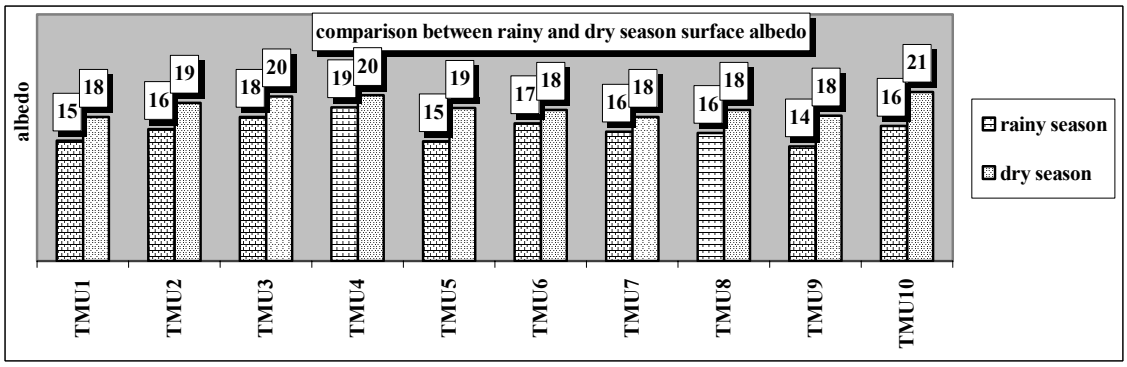


Figure 4.17 Comparison between rainy and dry season surface albedo

Comments

It seems from the above discussion that the spatial and temporal values for surface reflectance are reasonable and consistent with the factors affecting this response variable.

4.2 The Pairwise interrelationship between response variables

In this part the pairwise relationship between the response variables is studied in space, for the different classes, and in time for consistency. First the relationship between the NDVI and surface temperature is described and then surface temperature and surface albedo.

4.2.1 The NDVI-surface temperature relationship

Investigating the relationship between NDVI and surface temperature is difficult, because of many reasons. One important reason is that satellite measurement of surface temperature is an instantaneous one, which means that the measured temperature can also be affected by instantaneous events like rainfall. On the other hand NDVI measurements are not instantaneous but a continuous function expressing the dynamic behavior of vegetation. This means that there is a time lag between the vegetation and surface temperature response to rainfall. Also the remaining noise in the derived response variables in both surface temperature and NDVI may introduce some difficulty. In warm regions of Africa as in the study area, changes in Vegetation Indices like NDVI are expected to be negatively correlated to changes in surface temperature, as has been reported by Lambin and Ehrlich (1996). The monthly maximum NDVI values have been studied as a function of the monthly maximum surface temperature for one year starting from October 92 until September 93 for the whole catchment to avoid the effect of instantaneous events as much as possible. It was found that there is a negative relationship between NDVI and surface temperature expressed by the slope of the trend line with different correlation values and different slope of the trend line. The correlation between the two variables varied between the different months. However, the negative relation was found to be clear for the different months. The correlation was found to be high for almost all the months during the year. It can be noticed from Figures 4.18, 4.19 that the slope of the trend line is steeper in the dry season, and this can be explained by the strong influence of the rainfall on the NDVI-surface temperature relationship. This coincides with what was mentioned above by Lambin, and the same holds for the influence of the evapotranspiration on the relationship. In the rainy season no land covers suffer from shortage of water, so it is expected that latent heat flux takes the highest share of the available energy for the different classes, which means that the range of values of surface temperature between the extremes of the NDVI is small. But for the dry

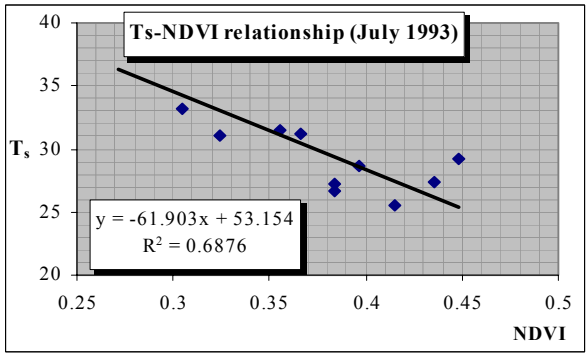


Figure 4.18 Surface temperature-NDVI relationship

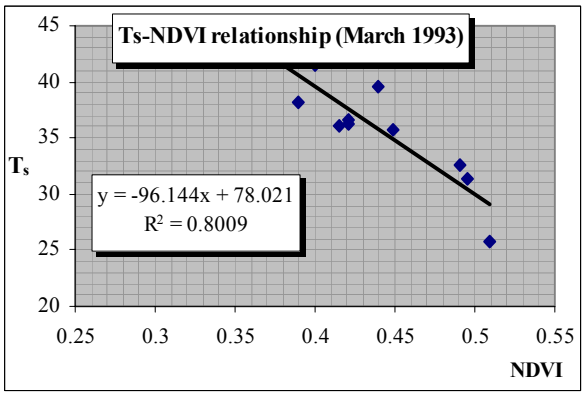


Figure 4.19 Surface temperature-NDVI relationship

season, the opposite occurs, most of the classes suffer from water shortage, which means less share for latent heat flux and high share for sensible heat flux, which in turn, means higher range of surface temperature between the extremes of NDVI. The higher surface temperature value, which corresponds to a certain NDVI value in the dry season as compared with the surface temperature at the same values of NDVI in the rainy season, can be noticed in the graphs. This may be explained by the high resistance to evapotranspiration due to low soil water availability in the dry season.

4.2.2 The surface temperature Albedo relationship

In this part the relationship between surface temperature and albedo was investigated. The relationship has been found to be positive until a certain albedo threshold value and then turns to be negative. Bastiaanssen et al. (1995) proved that

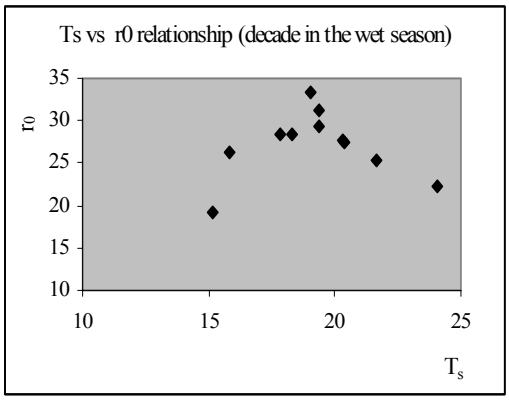


Figure4.20 Surface temperature_albedo relationship

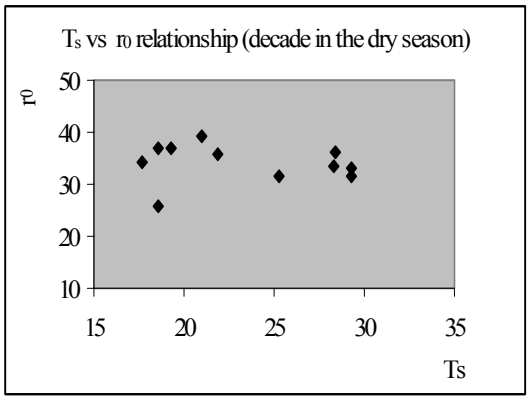


Figure 4.21 Surface temperature-albedo relationship

relationship pattern in Egypt by taking measurements across a transect from wet to dry areas. In this research since the data set is ten days composite, which means that pixel values were not necessarily extracted from the same day, so to compensate for this the relationship was tested by taking a transect through all the classes and in one decade in the dry season and another one in the rainy season. As can be seen from Figures 4.20, 4.21, the relationship exists. However, there is some scatter, which can be explained by the instantaneous response of the two variables to events like rainfall and also because of the ten-day composite characteristic of the data set. The relation has been tested for other decades in both the dry and wet season. A lot of scatter was found that does not express the expected relationship due to, as has been mentioned above, the difficulty imposed by the ten-days composite characteristic of the data set.

Comments

From the above discussion of the consistency of the individual and the pair-wise response variables relationships, it can be said that the degree of consistency is high. In that case we can rely on this output. Firstly to drive a qualitative assessment of the evapotranspiration of the different units in the study area, which, in turn, leads to a qualitative assessment of the hydrological behavior of the these units and consequently for the study area as a whole. Secondly, further work can be done using the Surface Energy Balance Algorithm for Land (SEBAL), for the quantification of the evapotranspiration. By using the other components of the water balance like

rainfall and discharge in an annual basis, a quantitative assessment of the water balance in the area can be reached for the study time period.

4.3 Hydrological Zonation of the Catchment

The time series for the arctangent of the surface temperature-vegetation index (NDVI) ratio, together with the previously analyzed time series for the response variables, were used to study the hydrological behavior for the different Terrain Mapping Units (TMU's) in the catchment to come up with the final hydrological zonation of the study area. As can be seen from Figure 4.24, The bimodal pattern of the time series coincides with the bimodal pattern of rainfall in the study area. The average values of the arctang(Ts/NDVI) and the six-month amplitude values for the different Terrain Mapping Units were plotted in the feature space to illustrate the distinction between these classes.

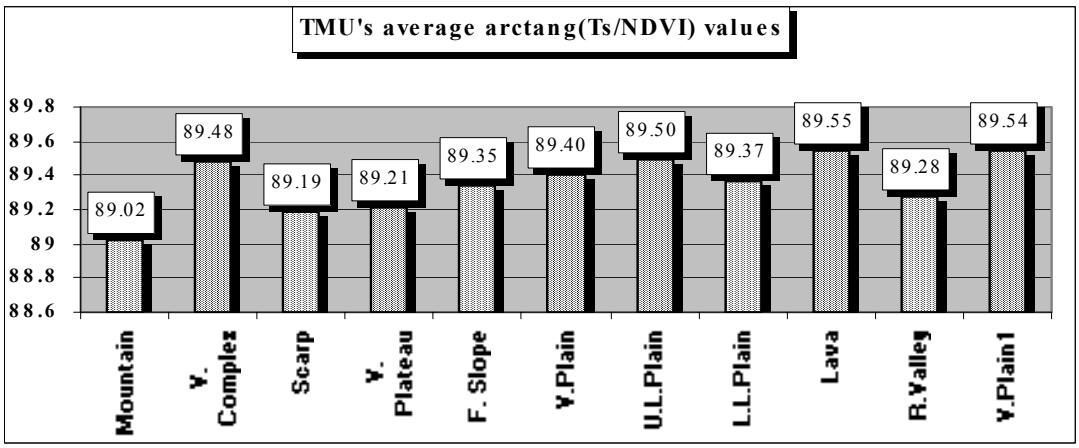


Figure 4.22 Average value of the Arctang(Ts/NDVI) for the different Terain Mapping Units

From the time series and the feature space Figures 4.24, 4.26, the following can be noted:

- For "TMU1", mountain with forest land cover, the bimodal pattern is not obvious and the pattern is irregular. The six-month component was detected with amplitude value of 1.31. The value of the ratio is low, which means low surface resistance to evapotranspiration or high soil moisture content. It seems also that this high wetness condition exists during the

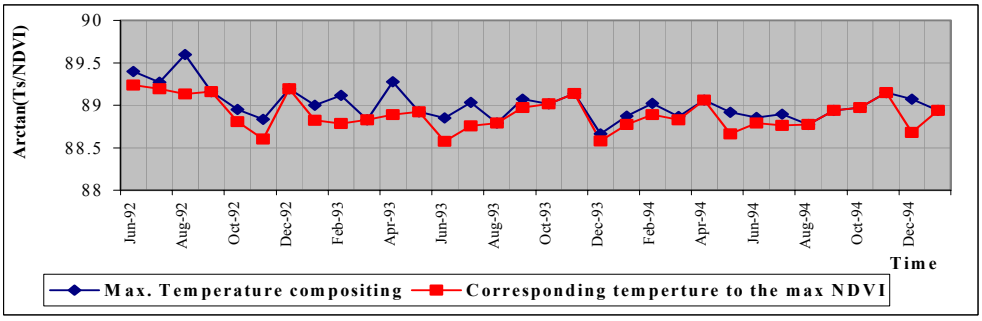


Figure 4.23 Comparison between arctang(Ts/NDVI) time series using the maximum monthly temperature and the temperature corresponding to the maximum NDVI

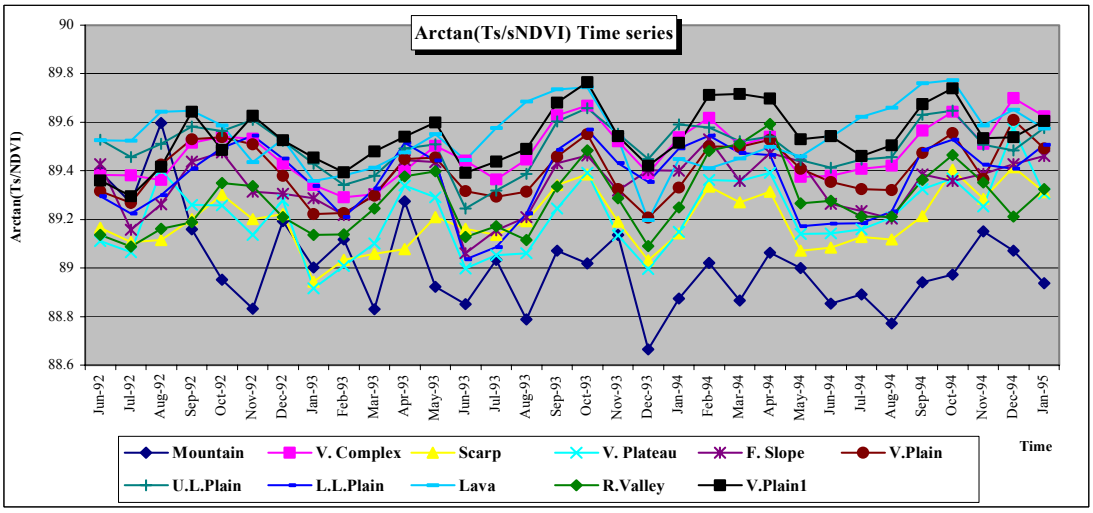


Figure 4.24 Time series of the Arctan(Ts/NDVI) for the different TMU's

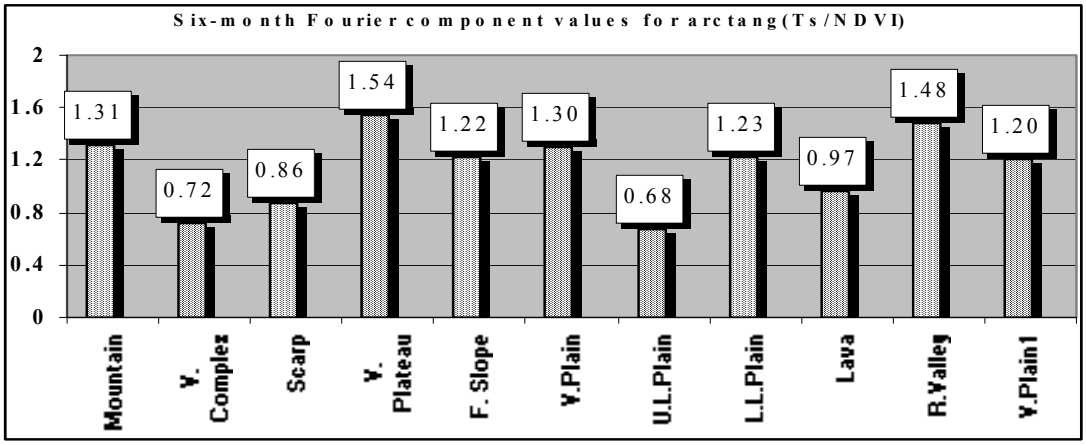


Figure 4.25 Amplitude values of the six-month Fourier component for the different TMU's

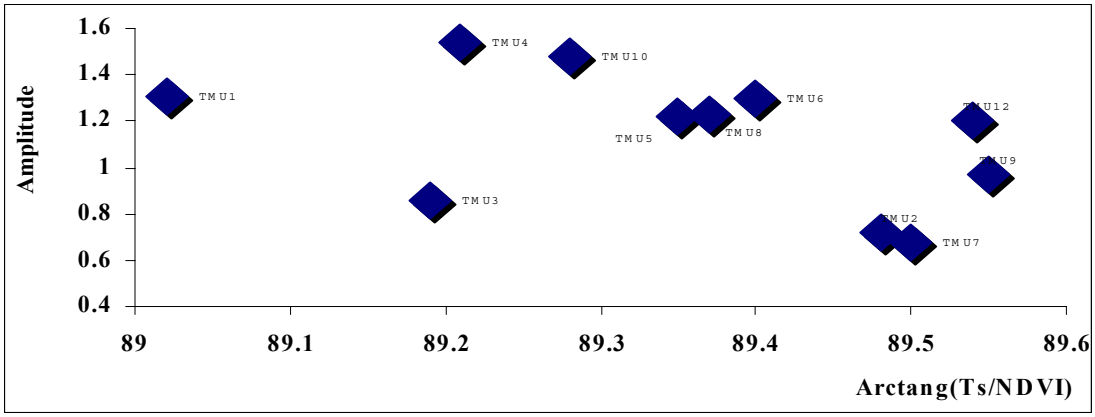


Figure 4.26 Feature space of the Amplitude and Arctang(Ts/NDVI)

whole year, which means that the rate of evapotanspiration from that unit can reach the potential condition for the whole year. Also the average value for albedo is low, which means that higher net radiation is available for evapotranspiration. The high frequency behavior of the arctangent of the surface temperature–NDVI ratio for the forest area can be related to the compositing technique. A comparison between the Arctang(Ts/NDVI) using the maximum monthly surface temperature composite and the arctang(Ts/NDVI) time series using the temperature corresponding to the maximum NDVI is shown in Figure 4.23. The smoother shape of the time series derived using the second technique can be seen in the graph, which shows more erratic behavior of the time series, derived from the first case to less erratic behavior obtained through the second compositing technique.

- The Scarp, the Volcanic Plateau, and the River valley "TMU's" almost have the same average values. The bimodal pattern exists, but with relatively low six-month component amplitude value for the scarp. This may give an indication of the low variability of the amount of evapotranspiration, which can be related to the nearly continuous availability of soil moisture for this unit. The Volcanic Plateau and the River Valley have almost the same amplitude value for the six-month component, which gives an indication of the similar dynamic behavior of the two units regarding evapotranspiration.
- For the Lava TMU, in spite of its geographic location close to the forest area, it seems that soil water availability is the lowest among other units in the catchment. This can be expected for this area with bare rock and natural vegetation land cover. The amplitude of the variations is low for the six-month component, which means that in spite of the high amount of rainfall received in this area, most of the water disappears as runoff, because of the thin or absent soil cover. Also the high frequency behavior of the Lava time series is explained by the quick response of the Ts-NDVI ratio of this kind of land cover to rainfall events.
- The dynamic pattern of the Ts-NDVI ratio for the Foot slope, the Volcanic Plain and the lower Lacustrine Plain TMU's is almost the same as can be seen from the time series and the six-month component amplitude value. The average values for the ratio are almost the same, which gives an indication that the three units have the same hydrological behavior from the point of view of evapotranspiration.
- The Upper Lacustrine Plain and the Volcanic Complex have almost the same average value for the ratio and also the amplitude values for the six-month component are almost the same and both are very low. The low amplitude value for the upper Lacustrine Plain can be explained by the agriculture in this area, which depends on the irrigation from the lake. This means that the rainfall pattern does not affect the ratio and the consequently the evapotranspiration process.
- The Volcanic Plain1 unit has low value of the ratio and a moderate six-month component. The low value can be explained by the land cover of the unit, which is sparse natural vegetation because the area does not receive high amount of rainfall, which in turn gives an indication of the high resistance to evapotranspiration for this unit.

Comments

The hydrological zonation was based on the evapotranspiration as one component of the hydrological cycle. The relationship between remote sensing derived response variables, after testing their individual and pair-wise consistency, and evapotranspiration was used in this regard. The dynamic behavior of the periodic functions of NDVI and the surface temperature-NDVI ratio expressed as their variability around the mean value and the mean value for the whole time series were used. The mean value of the surface temperature-NDVI ratio expresses the difference between the different zones in the catchment with respect to resistance to evapotranspiration. The variability around the mean value was used to express the sensitivity of the different zones to the bimodal pattern of rainfall.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

In this study a step-wise methodology was established to process a time series of images together with ancillary data. The processing has been implemented based on the integration between GIS software "ILWIS 2.2" and a C-program developed during the MSc. work. The program handles and processes the time series of images and the ancillary maps in an automated way and creates an interface between the GIS software, and exploratory methods for noise removal and Fast Fourier Transformation functions to decompose the time series into limited components.

Although the data set consists of ten-day composite images of the maximum NDVI, it was found that the cloud contamination persists in the study area. Based on the cloud detection criteria implemented in this study, about 49% of the area on average was contaminated by clouds in the time considered here.

For such cloud contaminated images, it was found that working with preliminary classified homogeneous units, Terrain Mapping Units, is more realistic than working on a pixel basis.

The level of consistency of the output results proved that the processing methodology developed in this work, is a success from a qualitative point of view.

In the absence of ground data measurements, and the state of the constituents of the atmosphere during the times of the satellite overpass, the statistical approach that was used for noise removal proved to be a reliable alternative for other atmospheric correction techniques. Also the technique is to be preferred over the one, which, assumes values for the atmosphere constituents because the gain from using the hypothesis is offset by the uncertainties in the assumed values.

It was found that images taken by NOAA-14 have higher pixel values than the corresponding pixel values in images taken by NOAA-11 in the different channels. In case of studying time series of images, care should be taken when the images are taken from different satellites because of the different calibration coefficients used for different sensors.

The smooth pattern of the Arctangent of the surface temperature-NDVI ratio gives a clue that the ratio of these variables is less sensitive to the noise than the individual variables. The ratio also seems to be related to the resistance to evapotranspiration of the different land covers in the study area.

5.2 RECOMMENDATIONS

A well-distributed network rainfall gauges in the catchement with accurate measurements is required for thorough understanding of the hydrological behavior of the different units in the catchment and consequently for the whole management process of the water resources.

The surface temperature-NDVI ratio function seems to be highly correlated to the hydrological behavior of the different units in the catchment with respect to the evapotranspiration component of the hydrological cycle and consequently to the soil moisture content. So it is recommended that further research be done on this relation together with other parameters like rainfall to arrive at a regional estimation of evapotranspiration for the different units in the catchment.

Further work can be done with respect to the ecological characteristics that affect the hydrological behavior of the different zones in the catchment. The soil, geology, land cover and topography can be combined with remote sensing derived hydrological response variables for this purpose.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

In this study a step-wise methodology was established to process a time series of images together with ancillary data. The processing has been implemented based on the integration between GIS software "ILWIS 2.2" and a C-program developed during the MSc. work. The program handles and processes the time series of images and the ancillary maps in an automated way and creates an interface between the GIS software, and exploratory methods for noise removal and Fast Fourier Transformation functions to decompose the time series into limited components.

Although the data set consists of ten-day composite images of the maximum NDVI, it was found that the cloud contamination persists in the study area. Based on the cloud detection criteria implemented in this study, about 49% of the area on average was contaminated by clouds in the time considered here.

For such cloud contaminated images, it was found that working with preliminary classified homogeneous units, Terrain Mapping Units, is more realistic than working on a pixel basis.

The level of consistency of the output results proved that the processing methodology developed in this work, is a success from a qualitative point of view.

In the absence of ground data measurements, and the state of the constituents of the atmosphere during the times of the satellite overpass, the statistical approach that was used for noise removal proved to be a reliable alternative for other atmospheric correction techniques. Also the technique is to be preferred over the one, which, assumes values for the atmosphere constituents because the gain from using the hypothesis is offset by the uncertainties in the assumed values.

It was found that images taken by NOAA-14 have higher pixel values than the corresponding pixel values in images taken by NOAA-11 in the different channels. In case of studying time series of images, care should be taken when the images are taken from different satellites because of the different calibration coefficients used for different sensors.

The smooth pattern of the Arctangent of the surface temperature-NDVI ratio gives a clue that the ratio of these variables is less sensitive to the noise than the individual variables. The ratio also seems to be related to the resistance to evapotranspiration of the different land covers in the study area.

5.2 RECOMMENDATIONS

A well-distributed network rainfall gauges in the catchement with accurate measurements is required for thorough understanding of the hydrological behavior of the different units in the catchment and consequently for the whole management process of the water resources.

The surface temperature-NDVI ratio function seems to be highly correlated to the hydrological behavior of the different units in the catchment with respect to the evapotranspiration component of the hydrological cycle and consequently to the soil moisture content. So it is recommended that further research be done on this relation together with other parameters like rainfall to arrive at a regional estimation of evapotranspiration for the different units in the catchment.

Further work can be done with respect to the ecological characteristics that affect the hydrological behavior of the different zones in the catchment. The soil, geology, land cover and topography can be combined with remote sensing derived hydrological response variables for this purpose.

THE C-PROGRAM FUNCTIONS

The C-program that was used to process the data is following the structured programming style. The program is limited to the data used in this research. It consists of many functions, and each function was designed to implement a certain task. In the following paragraphs a brief explanation of the different function tasks is described.

The "void four1(float data[],int nn,int isign)" function

This function performs the Fourier and inverse Fourier transformation. An array contains the values to be transformed, half of the number of elements in the array, and a flag to indicate either the Fourier transformation or the inverse Fourier transformation should be implemented.

The "int conv(int s)" function

This function is used to calculate the multiple of 2 nearest higher number to prepare the data used here for the Fourier transformation function, because for the Fast Fourier Transformation the number of elements of the array passed to the function should be a multiple of 2.

The "void spatial_filter()" function

This function is used to generate a new two dimensional array for the NDVI, surface temperature and surface albedo for the whole time series. Pixels in the new array are assigned a value of one of the uncontaminated neighboring pixels in the 3*3 window filter or otherwise the average value of the Terrain Mapping Unit to which the pixel belongs.

The "void write_to_file()" function

This function is used to convert the array which contains the pixel values to a binary file so that it can be read by ILWIS software as an image.

The "void read_thies_pol()" function

This function reads the Thiesen polygon image file to determine to which polygon the pixel belongs and put the values in a two dimensional array.

The "void fil_class_rain()" function

This function creates a time series of rainfall for every TMU in the catcment based on the Thiesen polygon method for areal distribution.

The "void fil_rain_gilgil_arr(), void fil_rain_kinang_arr(), void fil_rain_naivwd_arr(), and void fil_rain_olkalau_arr()" functions

These functions are used to read rainfall data from text files containing time series of rainfall for the Gilgil, North Kinangop, Naivasha W. D. and Olkalau meteorological stations and store this data into a one dimensional array.

The "void calc_air_temp()" function

This function read the binary file of the ILWIS images for the time serires of air temperature, calculate the average air temperature for every TMU in the catcment, and store the average value time series in a two dimensional array

The "void calc_residuals(float fin[][128],float finav[][128])" function

This function calculates the residuals after applying the moving median smoother filter on the time series. The residual value equals the smoothed value – the original value for the NDVI time series

*The "void getcolrow(FILE *fcolrow)" function*

This function is used to calculate the number of rows and columns in the image.

*The "void filarray(FILE *f, int flag)" function*

This function is used to read the ILWIS binary file images for the whole time series and fill a three dimensional array with the values of the pixels for the different time series of images used in the resarch.

The "void inv_four_for_classes(float inv_four_arr[][256])" function

This function is used to generate the inverse Fourier transformation of the time series of the NDVI to get a smoothed time series after removing the high frequency noise from the time series.

The "float std_calc(float st[], float avr, short no)" function

This function is used to calculate the standard deviation of values stored in an array.

The "float get_max_value(float arval[], short num)" function

This function is used to calculate the maximum value stored in an array

The "void calc_final_smooth_s4253h()" function

This function is used to calculate the time series by applying the "S4253H,twice" for the original NDVI time series.

The "void lcover()" function

This function is used to read the binary file of the ILWIS image for the Terrain Mapping Units map and stores the value in a two dimensional array.

The "void fil_class_array()" function

This function processes the three dimensional arrays which were filled in the "void filarray(FILE *f, int flag)"function to generate the final time series for the different units for the different variables used in the research.

The "void fil_rest_of_array(float rest_arr[][128])" function

This function pads zeros to the end of the array which contains the NDVI time series to fulfil the condition of the Fast Fourier Transformation condition

The "void four_for_arctndvi(float four_arr[][40],short flg)" function

This function calculates the Fourier components for the Arctan of the surface temperature-NDVI ratio

The "void four_for_classes(float four_arr[][128])" function

This function calculates the Fourier components for the time series of the NDVI average values for the different classes

The "void get_max_monthly(float maxmon[][128],float maxmontemp[][128], short flag)" function

This function calculates generates the time series of the maximum monthly NDVI and surface temperature for the different classes.

The "void arr_sort(float arr,short flag)" function

This function sorts an array passed to it and returns the sorted array.

The "void cal_med_array(float aver[][128], short flag)" function

This function calculates the median value of an array of values. This function is called by the "void calc_final_smooth_s4253h()" function which generates the smoothed NDVI function.

The "void get_max_temp_alb()" function

This function is used to generate the maximum surface temperature and albedo for every Terrain Mapping Unit in every image in the time series

The "float get_min_value(float marval[], short no)" function

This function calculates the minimum value in an array

The "void calc_evapo()" function

This function calculates the evapotranspiration based on an empirical formula

The "void calc_arct()" function

This function generates the monthly time series of the Arctangent of the surface temperature-NDVI ratio

DOWNLOADED IMAGES

The downloaded images are stored in the “” directory. A list of the downloaded images is listed in the following table. A value of “1” means the image is available in the web site, and a value of “0” means that the image is not available.

Decade	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
1-10 April 1992	1	1	0	1	1	1
10-20 April 1992	1	1	0	1	1	1
20-30 April 1992	1	1	1	1	1	1
1-10 May 1992	1	1	0	1	1	1
10-20 May 1992	1	1	0	1	1	1
20-31 May 1992	1	1	1	1	1	1
1-10 June 1992	1	1	0	1	1	1
10-20 June 1992	1	1	0	1	1	1
20-30 June 1992	1	1	1	1	1	1
1-10 July 1992	1	1	0	1	1	1
10-20 July 1992	1	1	0	1	1	1
20-31 July 1992	1	1	1	1	1	1
1-10 August 1992	1	1	0	1	1	1
10-20 August 1992	1	1	0	1	1	1
20-30 August 1992	1	1	1	1	1	1
1-10 September 1992	1	1	0	1	1	1
10-20 September 1992	1	1	0	1	1	1
10-30 September 1992	1	1	1	1	1	1
1-10 October 1992	1	1	0	1	1	1
10-20 October 1992	1	1	0	1	1	1
20-30 October 1992	1	1	1	1	1	1
1-10 November 1992	1	1	0	1	1	1
10-20 November 1992	1	1	0	1	1	1
20-30 November 1992	1	1	1	1	1	1
1-10 December 1992	1	1	0	1	1	1
10-20 December 1992	1	1	0	1	1	1
20-30 December 1992	1	1	1	1	1	1
1-10 January 1993	1	1	0	1	1	1
10-20 January 1993	1	1	0	1	1	1
20-30 January 1993	1	1	1	1	1	1
1-10 February 1993	1	1	0	1	1	1
10-20 February 1993	1	1	0	1	1	1
20-28 February 1993	1	1	1	1	1	1
1-10 March 1993	1	1	0	1	1	1
10-20 March 1993	1	1	0	1	1	1
20-31 March 1993	1	1	1	1	1	1
1-10 April 1993	1	1	0	1	1	1
10-20 April 1993	1	1	0	1	1	1
20-30 April 1993	1	1	1	1	1	1
1-10 May 1993	1	1	0	1	1	1
10-20 May 1993	1	1	0	1	1	1
20-31 May 1993	1	1	1	1	1	1
1-10 June 1993	1	1	0	1	1	1
10-20 June 1993	1	1	0	1	1	1
20-30 June 1993	1	1	1	1	1	1
1-10 July 1993	1	1	0	1	1	1

Decade	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
10-20 July 1993	1	1	0	1	1	1
20-31 July 1993	1	1	1	1	1	1
1-10 August 1993	1	1	0	1	1	1
10-20 August 1993	1	1	0	1	1	1
20-30 August 1993	1	1	1	1	1	1
1-10 September 1993	0	0	0	0	0	0
10-20 September 1993	1	1	0	1	1	1
10-30 September 1993	1	1	1	1	1	1
1-10 October 1992	0	0	0	0	0	0
10-20 October 1993	0	0	0	0	0	0
20-30 October 1993	0	0	0	0	0	0
1-10 November 1993	0	0	0	0	0	0
10-20 November 1993	0	0	0	0	0	0
20-30 November 1993	0	0	0	0	0	0
1-10 December 1993	0	0	0	0	0	0
10-20 December 1993	0	0	0	0	0	0
20-30 December 1993	0	0	0	0	0	0
1-10 January 1994	0	0	0	0	0	0
10-20 January 1994	0	0	0	0	0	0
20-30 January 1994	0	0	0	0	0	0
1-10 February 1994	0	0	0	0	0	0
10-20 February 1994	0	0	0	0	0	0
20-28 February 1994	0	0	0	0	0	0
1-10 March 1994	0	0	0	0	0	0
10-20 March 1994	0	0	0	0	0	0
20-31 March 1994	0	0	0	0	0	0
1-10 April 1994	0	0	0	0	0	0
10-20 April 1994	0	0	0	0	0	0
20-30 April 1994	0	0	0	0	0	0
1-10 May 1994	0	0	0	0	0	0
10-20 May 1994	0	0	0	0	0	0
20-31 May 1994	0	0	0	0	0	0
1-10 June 1994	0	0	0	0	0	0
10-20 June 1994	0	0	0	0	0	0
20-30 June 1994	0	0	0	0	0	0
1-10 July 1994	0	0	0	0	0	0
10-20 July 1994	0	0	0	0	0	0
20-31 July 1994	0	0	0	0	0	0
1-10 August 1994	0	0	0	0	0	0
10-20 August 1994	0	0	0	0	0	0
20-30 August 1994	0	0	0	0	0	0
1-10 September 1994	0	0	0	0	0	0
10-20 September 1994	0	0	0	0	0	0
10-30 September 1994	0	0	0	0	0	0
1-10 October 1994	0	0	0	0	0	0
10-20 October 1994	0	0	0	0	0	0
20-30 October 1994	0	0	0	0	0	0
1-10 November 1994	0	0	0	0	0	0
10-20 November 1994	0	0	0	0	0	0
20-30 November 1994	0	0	0	0	0	0
1-10 December 1994	0	0	0	0	0	0
10-20 December 1994	0	0	0	0	0	0
20-30 December 1994	0	0	0	0	0	0
1-10 January 1995	0	0	0	0	0	0

Decade	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
10-20 January 1995	0	0	0	0	0	0
20-30 January 1995	0	0	0	0	0	0
1-10 February 1995	1	1	0	1	1	1
10-20 February 1995	1	1	0	1	1	1
20-28 February 1995	1	1	1	1	1	1
1-10 March 1995	1	1	0	1	1	1
10-20 March 1995	1	1	0	1	1	1
20-31 March 1995	1	1	1	1	1	1
1-10 April 1995	1	1	0	1	1	1
10-20 April 1995	1	1	0	1	1	1
20-30 April 1995	1	1	1	1	1	1
1-10 May 1995	1	1	0	1	1	1
10-20 May 1995	1	1	0	1	1	1
20-31 May 1995	1	1	1	1	1	1
1-10 June 1995	1	1	0	1	1	1
10-20 June 1995	1	1	0	1	1	1
20-30 June 1995	1	1	1	1	1	1
1-10 July 1995	1	1	0	1	1	1
10-20 July 1995	1	1	0	1	1	1
20-31 July 1995	1	1	1	1	1	1
1-10 August 1995	1	1	0	1	1	1
10-20 August 1995	1	1	0	1	1	1
20-30 August 1995	1	1	1	1	1	1
1-10 September 1995	1	1	0	1	1	1
10-20 September 1995	1	1	0	1	1	1
10-30 September 1995	1	1	1	1	1	1
1-10 October 1995	1	1	0	1	1	1
10-20 October 1995	1	1	0	1	1	1
20-30 October 1995	1	1	1	1	1	1
1-10 November 1995	1	1	0	1	1	1
10-20 November 1995	1	1	0	1	1	1
20-30 November 1995	1	1	1	1	1	1
1-10 December 1995	1	1	0	1	1	1
10-20 December 1995	1	1	0	1	1	1
20-30 December 1995	1	1	1	1	1	1
1-10 January 1996	1	1	0	1	1	1
10-20 January 1996	1	1	0	1	1	1
20-30 January 1996	1	1	1	1	1	1

PHASE ANGLE TO DECADES CONVERSION

The time at which the sinusoidal function of the Fourier components reaches its peak, starting from the beginning of the time series, is calculated as follows:

1- the phase angle (in radians) for the component is calculated using the Fast Fourier Transformation function
2- the sign of the imaginary and real values of the angle is checked to determine in which quarter the angle falls
3- if the angle falls in the first and fourth quarter the following formula is applied

$$X=\lambda/2\pi * (\pi/2 - \alpha)$$

Where:

- X is the period in decades at which the function reaches its peak
- λ is the total period in decades for the Fourier component
- α is the calculated phase angle in radians

4- if the angle falls in the second quarter the following formula is applied

$$X =\lambda/2\pi * (\pi/2 - \alpha + \pi)$$

5- if the angle in the third quarter the following formula is used

$$X =\lambda/2\pi * (\alpha + \pi)$$

LISTING OF THE C-PROGRAM

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <assert.h>
#define swap(a,b) tempr=(a);(a)=(b);(b)=tempr;
#define albcons 7459;
float
c1c2ratio[88][68][128],flist[88][68][128],chan1[88][68][128],chan2[88][68][128],templist[88][68][128],threshold[88][68][128],tdlist[88][68][128],
qdlist[88][68][128];
float
avchan1[15][128],avchan2[15][128],avchan4[15][128],avchan5[15][128],avtemp[15][128],av[15][128],templist4[88][68][128],templist5[88][68][1
28];
float
rairtemp[15][128],cairtemp[15][128],airtemp[15][128],alb[15][128],amp[15][128],surtemp[15][128],ssurtemp[15][128],imagpart[15][128],realpart[
15][128];
float
psurtemp[15][128],isurtemp[15][128],ysurtemp[15][128],nsurtemp[15][128],phas[15][128],finalf[15][128],final[15][128],resid[15][128],fresid[15]
[128];
float arctndvi[15][40],arctphase[15][40],arctamp[15][40],iarct[15][40],rarct[15][40];
float farctndvi[15][40],farctphase[15][40],farctamp[15][40],fiarct[15][40],frarct[15][40];
float maxfalb[100],maxftemp[100];
char cover[88][68];
short test[40],rawamp,rawphase,r,lin,col;
float tdata1[256],invfour[15][256];
float gilgil[100],naivwd[100],kinang[100],olkalau[100];
float rain[15][128],raindist[15][100],navertemp[15][128],avertemp[15][128],averef[15][128];
float fchan1[88][68][128],fchan2[88][68][128],fchan4[88][68][128],fchan5[88][68][128],ndvi[88][68][128];
float ptemp[88][68][128],palb[88][68][128],mondvi[15][35],montemp[15][35],fmondvi[15][35],fmontemp[15][35];
unsigned short i,j,k,l,dex,x,y,z,freq,cnt;
char ch;
float
malb[128],mtemp[128],mintemp[15][128],evindex[15][128],lhflux[15][128],maxalb[15][128],maxtemp[15][128],evapo[15][128],amontemp[15][128],famontemp[15][128];
short n,strint,num,newstrint;
short acov1,acov2,acov3,acov4,acov5,acov6,acov7,acov8,acov9,acov10,acov11,acov12;
float cloud[15][120];
short ccloud[15][128],acov[13];
FILE *mlist,*tlist,*qlist,*thresh,*temp4,*ch4,*ch5;
char fstr1[80],fstr1rev[40],fstr2rev[40],fstr[80],str1[80],str2[80],str3[5],str4[80];
FILE *image1;
FILE *outtest;
char thpol[88][68];
FILE *pt, *clas;
FILE *ptrphase;
char stramp[60],strphase[60], buf1[3],buf2[3],*stramp1,*strphase1;
char str5[50],str6[13],str7[13],strline[4],stcol[4],strend[4],strend1[4];
void four1(float data[],int nn,int isign);
int conv(int s);
void spatial_filter();
void read_thies_pol();
void write_to_file();
void fil_class_rain();
void redistribute_air_temp();
void fil_rain_gilgil_arr();
void fil_rain_naivwd_arr();
void spatial_filter();
void calc_rain();
void calc_air_temp();
void fil_rain_kinang_arr();
void fil_rain_olkalau_arr();
void calc_residuals(float fin[][128],float finav[][128]);
void getcolrow(FILE *fcolrow);
void filarray(FILE *f, int flag);
void inv_four_for_classes(float inv_four_arr[][256]);
float std_calc(float st[], float avr, short no);
float get_max_value(float arval[], short num);
void calc_final_smooth_s4253h();
void lcover();
void fil_class_array();
void fil_rest_of_array(float rest_arr[][128]);
void four_for_arctndvi(float four_arr[][40],short flg);
void four_for_classes(float four_arr[][128]);
void get_max_monthly(float maxmon[][128],float maxmontemp[][128], short flag);
void arr_sort(float arr,short flag);
void cal_med_array(float aver[][128], short flag);
void get_max_temp_alb();
float get_min_value(float marval[], short no);
void calc_evapo();
void calc_arct();
void time_series();
void evapo_map();
void main()
{
 getcolrow(mlist);
 filarray(mlist,1);
 filarray(temp4,5);
 filarray(thresh,4);
 filarray(tlist,2);
 filarray(qlist,3);
 filarray(ch4,6);
 filarray(ch5,7);
```

```
 lcover();
 fil_rain_gilgil_arr();
 fil_rain_naivwd_arr();
 fil_rain_kinang_arr();
 fil_rain_olkalau_arr();
 read_thies_pol();
 fil_class_rain();
newstrint=conv(strint);
calc_air_temp();
fil_class_array();
redistribute_air_temp();
cal_med_array(av,1);
fil_rest_of_array(final);
calc_residuals(final,av);
cal_med_array(resid,2);
fil_rest_of_array(fresid);
calc_final_smooth_s4253h();
four_for_classes(av);
inv_four_for_classes(invfour);
calc_residuals(finalf,av);
fil_rain_gilgil_arr();
fil_rain_naivwd_arr();
fil_rain_kinang_arr();
fil_rain_olkalau_arr();
fil_class_rain();
calc_rain();
get_max_monthly(av,avertemp,1);
get_max_monthly(finalf,avertemp,2);
calc_evapo();
calc_arct();
four_for_arctndvi(arctndvi,1);
four_for_arctndvi(farctndvi,2);
time_series();
outtest=fopen("d:\\msc\\thesis\\newtmu44.txt","w");
for(x=1;x<13;x++)
{
   fprintf(outtest,"class code is %d\n\n ",x);
                  for(i=0,j=0;i<newstrint,j<2*newstrint;i++,j+=2)
                                    fprintf(outtest,"%.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %f %.3f %.3f %.5f %.5f %.1f %.3f %.3f %.3f %.3f %.3f
%.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f\n
",av[x][i],finalf[x][i],invfour[x][j]/128,avchan4[x][i],avchan5[x][i],nsurtemp[x][i],surtemp[x][i],alb[x][i],resid[x][i],(double)i/(double)newstrint,am
p[x][2*i],phas[x][2*i+1],imagpart[x][i],realpart[x][i],rain[x][i],raindist[x][i],cairtemp[x][i],airtemp[x][i],rairtemp[x][i],mondvi[x][i],montemp[x][i],
fmondvi[x][i],fmontemp[x][i],amontemp[x][i],famontemp[x][i],mintemp[x][i],maxtemp[x][i],maxalb[x][i],navertemp[x][i],avertemp[x][i],averef[x
][i],evapo[x][i],maxfalb[i],maxftemp[i],(float)(acov[x]-ccloud[x][i])/(float)acov[x]*100);
   fprintf(outtest,"\n\n");
}
fprintf(outtest,"\n\n");
for(i=0;i<32;i++)
{
for(x=1;x<13;x++)
if(x!=11)
fprintf(outtest,"%.5f %.5f %.5f %.3f %.3f ",arctndvi[x][i],iarct[x][i],rarct[x][i],arctphase[x][i],arctamp[x][i]);
fprintf(outtest,"\n");
}
fprintf(outtest,"\n\n");
for(i=0;i<32;i++)
{
for(x=1;x<13;x++)
if(x!=11)
fprintf(outtest,"%.5f %.5f %.5f %.3f %.3f ",farctndvi[x][i],fiarct[x][i],frarct[x][i],farctphase[x][i],farctamp[x][i]);
fprintf(outtest,"\n");
}
fprintf(outtest,"\n\n");
for(i=0;i<32;i++)
{
for(x=1;x<13;x++)
if(x!=11)
fprintf(outtest,"%.5f ",atan(montemp[x][i]/mondvi[x][i])*180*7/22);
fprintf(outtest,"\n");
}
                  fclose(outtest);
}
                  void four1(float data[],int nn,int isign)
                  {
                                    int n,mmax,m,j,istep,i;
                     double wtemp,wr,wpr,wpi,wi,theta;
                     float tempr,tempi;
                     n=nn<<1;
                     j=1;
                     for(i=1;i<n;i+=2)
                                    {
                                       if(j>i)
                                       {
                                                         swap(data[j],data[i]);
                                                         swap(data[j+1],data[i+1]);
                                       }
                                    m=n>>1;
                                    while(m>=2 && j>m)
                                    {
                                                      j-=m;
                                                      m>>=1;
                                    }
                                    j+=m;
                                    }
                  mmax=2;
                  while(n>mmax)
```

```
		{
				istep=2*mmax;
				theta=6.2832/(isign*mmax);
				wtemp=sin((0.5)*theta);
				wpr=-2.0*wtemp*wtemp;
				wpi=sin(theta);
				wr=1.0;
				wi=0.0;
	for(m=1;m<mmax;m+=2)
				{
					for(i=m;i<=n;i+=istep)
					{
						j=i+mmax;
						tempr=wr*data[j]-wi*data[j+1];
						tempi=wr*data[j+1]+wi*data[j];
						data[j]=data[i]-tempr;
						data[j+1]=data[i+1]-tempi;
						data[i]+=tempr;
						data[i+1]+=tempi;
					}
					wr=(wtemp=wr)*wpr-wi*wpi+wr;
					wi=wi*wpr+wtemp*wpi+wi;
				}
				mmax=istep;
		}
}
		int conv(int s)
		{
		double x, y, n,z,q,m;
		q = log10((double) s);
  z=log10(2.0);
		x=q/z;
		y = modf( x, &n );
		if(y!=0.0)
		{
				n+=1;
				m=pow(2,n);
	return((int) m);
		}
 return(s);

		}
void getcolrow(FILE *fcolrow)
{
if((fcolrow=fopen("c:\\fnada\\newnoaa\\ndmaplst.mpl","r"))==NULL)
 {
		printf("cannot open the file");
  exit(1);
 }
  while(fgets(str1,80,fcolrow)!=NULL)
 {
		if(strstr(str1,"Size=")!=NULL)
		{
				i=5;
				j=0;
				while(str1[i]!=' ')
					strline[j++]=str1[i++];
				i+=1;
				for(j=0;i<strlen(str1)-1;j++,i++)
					stcol[j]=str1[i];
		break;
		}
 }
fclose(fcolrow);
lin=atoi(strline);
col=atoi(stcol);
}
void filarray(FILE *f,int flag)
{
 switch(flag)
 {
 case 1:
	if((f=fopen("c:\\fnada\\newnoaa\\ndmaplst.mpl","r"))==NULL)
				{
					printf("cannot open the file");
					exit(1);
				}
				break;
 case 2:
	if((f=fopen("c:\\fnada\\newnoaa\\templst5.mpl","r"))==NULL)
				{
					printf("cannot open the file");
					exit(1);
				}
				break;
 case 3:
	if((f=fopen("c:\\fnada\\newnoaa\\qmaplst.mpl","r"))==NULL)
				{
					printf("cannot open the file");
					exit(1);
				}
				break;
  case 4:
	if((f=fopen("c:\\fnada\\newnoaa\\thmaplst.mpl","r"))==NULL)
				{
					printf("cannot open the file");
```

```
                                        exit(1);
                                }
                                break;
 case 5:
      if((f=fopen("c:\\fnada\\newnoaa\\templst4.mpl","r"))==NULL)
                                {
                                        printf("cannot open the file");
                                        exit(1);
                                }
                                break;
 case 6:
      if((f=fopen("c:\\fnada\\newnoaa\\ch1mapls.mpl","r"))==NULL)
                                {
                                        printf("cannot open the file");
                                        exit(1);
                                }
                                break;
 case 7:
      if((f=fopen("c:\\fnada\\newnoaa\\ch2mapls.mpl","r"))==NULL)
                                {
                                        printf("cannot open the file");
                                        exit(1);
                                }
                                break;
}
while(fgets(str1,80,f)!=NULL)
 {
                if(strstr(str1,"Maps=")!=NULL)
                {
                 j=0;
                 for(i=0;i<strlen(str1);i++)
                  if(isdigit(str1[i]))
                    str3[j++]=str1[i];
                 strint=atoi(str3);
                 fgets(str4,80,f);
                 for(n=0;n<strint;n++)
                 {
     strcpy(str6,"");
                  strcpy(fstr1,"");
                  fgets(fstr1,80,f);
                cnt=0;
                                for(i=0;i<strlen(fstr1);i++)
                                        if(fstr1[i]=='=')
                                        {
                                                for(j=0,i+=1;i<strlen(fstr1)-1;j++,i++)
                                                 str6[j]=fstr1[i];
          break;
                                        }
                                str6[strlen(str6)-1]='#';
                                strcpy(str5,"c:\\fnada\\newnoaa\\");
                                strcat(str5,str6);
                if( (image1 = fopen(str5, "rb" )) == NULL )
  {
                                printf("cannot open the file  %s\n",str5);
                exit(1);
                }
                switch(flag)
                {
                case 1:
                                        for(j=0;j<lin;j++)
                                        for(k=0;k<col;k++)
                                if(fread(&r, sizeof(r),1, image1) == 1)
                                        flist[j][k][n]=((float)r)*0.001;
        break;
                case 2:
                                        for(j=0;j<lin;j++)
                                        for(k=0;k<col;k++)
                                         if(fread(&r, sizeof(r),1, image1) == 1)
                                         {
                                                templist[j][k][n]=((float)r)*0.01;
                                                templist5[j][k][n]=((float)r)*0.01;
                                                if(templist[j][k][n]<threshold[j][k][n]-6)
                                                        tdlist[j][k][n]=-2.0;
                                                else
                                                        tdlist[j][k][n]=flist[j][k][n];
                                         }
                        break;
                case 3:
                                        for(j=0;j<lin;j++)
                                        for(k=0;k<col;k++)
                                if(fread(&r, sizeof(r),1, image1) == 1)
                                        if((float)r*0.1<1.4)
                                                qdlist[j][k][n]=-100;
        break;
                case 4:
                                        for(j=0;j<lin;j++)
                                        for(k=0;k<col;k++)
                                if(fread(&r, sizeof(r),1, image1) == 1)
                                        threshold[j][k][n]=((float)r)*0.01;
        break;
                case 5:
                                        for(j=0;j<lin;j++)
                                        for(k=0;k<col;k++)
                                         if(fread(&r, sizeof(r),1, image1) == 1)
                                                templist4[j][k][n]=((float)r)*0.01;
                                        break;
```

```
   case 6:
                                        for(j=0;j<lin;j++)
                                        for(k=0;k<col;k++)
                                if(fread(&r, sizeof(r),1, image1) == 1)
                                        chan1[j][k][n]=((float)r)*0.1;
          break;
   case 7:
                                        for(j=0;j<lin;j++)
                                        for(k=0;k<col;k++)
                                if(fread(&r, sizeof(r),1, image1) == 1)
                                        chan2[j][k][n]=((float)r)*0.1;
          break;
                }
fclose(image1);
   }
  break;
}
}
fclose(f);
}
void fil_class_array()
{
short c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12;
short i,j,k;
short ch,act,acr;
float clas1,clas2,clas3,clas4,clas5,clas6,clas7,clas8,clas9,clas10,clas11,clas12;
float air1,air2,air3,air4,air5,air6,air7,air8,air9,air10,air11,air12;
float temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8,temp9,temp10,temp11,temp12;
float ntemp1,ntemp2,ntemp3,ntemp4,ntemp5,ntemp6,ntemp7,ntemp8,ntemp9,ntemp10,ntemp11,ntemp12;
float ch11,ch12,ch13,ch14,ch15,ch16,ch17,ch18,ch19,ch110,ch111,ch112;
float ch21,ch22,ch23,ch24,ch25,ch26,ch27,ch28,ch29,ch210,ch211,ch212;
float ch41,ch42,ch43,ch44,ch45,ch46,ch47,ch48,ch49,ch410,ch411,ch412;
float ch51,ch52,ch53,ch54,ch55,ch56,ch57,ch58,ch59,ch510,ch511,ch512;
float cl1[300],cl2[400],cl3[230],cl4[1000],cl5[80],cl6[700],cl7[100],cl8[80],cl9[50];
float cl10[250],cl11[160],cl12[100];
float cl1ch1[300],cl2ch1[400],cl3ch1[230],cl4ch1[1000],cl5ch1[80],cl6ch1[700],cl7ch1[100],cl8ch1[80],cl9ch1[50];
float cl10ch1[250],cl11ch1[160],cl12ch1[100];
float cl1ch2[300],cl2ch2[400],cl3ch2[230],cl4ch2[1000],cl5ch2[80],cl6ch2[700],cl7ch2[100],cl8ch2[80],cl9ch2[50];
float cl10ch2[250],cl11ch2[160],cl12ch2[100];
float cl1ch4[300],cl2ch4[400],cl3ch4[230],cl4ch4[1000],cl5ch4[80],cl6ch4[700],cl7ch4[100],cl8ch4[80],cl9ch4[50];
float cl10ch4[250],cl11ch4[160],cl12ch4[100];
float cl1ch5[300],cl2ch5[400],cl3ch5[230],cl4ch5[1000],cl5ch5[80],cl6ch5[700],cl7ch5[100],cl8ch5[80],cl9ch5[50];
float cl10ch5[250],cl11ch5[160],cl12ch5[100];
float std1,absval;
float tp1[300],tp2[400],tp3[230],tp4[1000],tp5[100],tp6[700],tp7[100],tp8[80],tp9[50];
float tp10[250],tp11[160],tp12[100];
float ntp1[300],ntp2[400],ntp3[230],ntp4[1000],ntp5[100],ntp6[700],ntp7[100],ntp8[80],ntp9[50];
float ntp10[250],ntp11[160],ntp12[100];
float ref1[300],ref2[400],ref3[230],ref4[1000],ref5[100],ref6[700],ref7[100],ref8[80],ref9[50];
float ref10[250],ref11[160],ref12[100];
float refl1,refl2,refl3,refl4,refl5,refl6,refl7,refl8,refl9,refl10,refl11,refl12;
float avtp1,avtp2,avtp3,avtp4,avtp5,avtp6,avtp7,avtp8,avtp9,avtp10,avtp11,avtp12;
float avntp1,avntp2,avntp3,avntp4,avntp5,avntp6,avntp7,avntp8,avntp9,avntp10,avntp11,avntp12;
float avref1,avref2,avref3,avref4,avref5,avref6,avref7,avref8,avref9,avref10,avref11,avref12;
float ac,ep1,ep2;
short h,count1,count2,count3,count4,count5,count6;
for(i=1;i<13;i++)
for(k=0;k<strint;k++)
ccloud[i][k]=0;
lcover();
for(k=0;k<strint;k++)
{
c1=c2=c3=c4=c5=c6=c7=c8=c9=c10=c11=c12=0;
clas1=clas2=clas3=clas4=clas5=clas6=clas7=clas8=clas9=clas10=clas11=clas12=0;
air1=air2=air3=air4=air5=air6=air7=air8=air9=air10=air11=air12=0;
temp1=temp2=temp3=temp4=temp5=temp6=temp7=temp8=temp9=temp10=temp11=temp12=0;
refl1=refl2=refl3=refl4=refl5=refl6=refl7=refl8=refl9=refl10=refl11=refl12=0;
ch11=ch12=ch13=ch14=ch15=ch16=ch17=ch18=ch19=ch110=ch111=ch112=0;
ch21=ch22=ch23=ch24=ch25=ch26=ch27=ch28=ch29=ch210=ch211=ch212=0;
ch41=ch42=ch43=ch44=ch45=ch46=ch47=ch48=ch49=ch410=ch411=ch412=0;
ch51=ch52=ch53=ch54=ch55=ch56=ch57=ch58=ch59=ch510=ch511=ch512=0;
for(i=0;i<lin;i++)
for(j=0;j<col;j++)
{
switch(cover[i][j])
{
case 1:
        if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
        {
                clas1+=flist[i][j][k];
                ch11+=chan1[i][j][k];
                ch21+=chan2[i][j][k];
                cl1ch1[c1]=chan1[i][j][k];
                cl1ch2[c1]=chan2[i][j][k];
                cl1[c1]=flist[i][j][k];
                ch41+=templist4[i][j][k];
                ch51+=templist5[i][j][k];
                cl1ch4[c1]=templist4[i][j][k];
                cl1ch5[c1]=templist5[i][j][k];
                cl1[c1]=flist[i][j][k];
                ref1[c1]=.7459+.347 * cl1ch1[c1] + .65 * cl1ch2[c1];
                tp1[c1]=cl1ch4[c1]+(1+.58*(cl1ch4[c1]-cl1ch5[c1]))*(cl1ch4[c1]-cl1ch5[c1])+.51+40*(1-(1.0094+.047 *
                (float)(log((double)cl1[c1]))))+30*.004;
                temp1+=cl1ch4[c1]+(1+.58*(cl1ch4[c1]-cl1ch5[c1]))*(cl1ch4[c1]-cl1ch5[c1])+.51+40*(1-(1.0094+.047 *
                (float)(log((double)cl1[c1]))))+30*.004;
```

```
                        ntp1[c1]=cl1ch4[c1]+2.13*(cl1ch4[c1]-cl1ch5[c1])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl1[c1])))-200*(-
                        .004);
                        ntemp1+=cl1ch4[c1]+2.13*(cl1ch4[c1]-cl1ch5[c1])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl1[c1])))-200*(-
                        .004);
                        refl1+=.7459+.347 * cl1ch1[c1] + .65 * cl1ch2[c1];
                        air1+=threshold[i][j][k];
                        c1++;
                        ccloud[1][k]+=1;
                }
                break;
case 2:
                if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
                {
                        clas2+=flist[i][j][k];
                        ch12+=chan1[i][j][k];
                        ch22+=chan2[i][j][k];
                        cl2ch1[c2]=chan1[i][j][k];
                        cl2ch2[c2]=chan2[i][j][k];
                        ch42+=templist4[i][j][k];
                        ch52+=templist5[i][j][k];
                        cl2ch4[c2]=templist4[i][j][k];
                        cl2ch5[c2]=templist5[i][j][k];
                        cl2[c2]=flist[i][j][k];
                        ref2[c2]=.7459+.347 * cl2ch2[c2] + .65 * cl2ch2[c2];
                        tp2[c2]=cl2ch4[c2]+(1+.58*(cl2ch4[c2]-cl2ch5[c2]))*(cl2ch4[c2]-cl2ch5[c2])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl2[c2])))+30*.004;
                        temp2+=cl2ch4[c2]+(1+.58*(cl2ch4[c2]-cl2ch5[c2]))*(cl2ch4[c2]-cl2ch5[c2])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl2[c2])))+30*.004;
                        ntp2[c2]=cl2ch4[c2]+2.13*(cl2ch4[c2]-cl2ch5[c2])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl2[c2])))-200*(-
                        .004);
                        ntemp2+=cl2ch4[c2]+2.13*(cl2ch4[c2]-cl2ch5[c2])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl2[c2])))-200*(-
                        .004);
                        refl2+=.7459+.347 * cl2ch2[c2] + .65 * cl2ch2[c2];
                        air2+=threshold[i][j][k];
                        c2++;
                        ccloud[2][k]+=1;
                }
                break;
case 3:
                if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
                {
                        clas3+=flist[i][j][k];
                        ch13+=chan1[i][j][k];
                        ch23+=chan2[i][j][k];
                        cl3ch1[c3]=chan1[i][j][k];
                        cl3ch2[c3]=chan2[i][j][k];
                        ch43+=templist4[i][j][k];
                        ch53+=templist5[i][j][k];
                        cl3ch4[c3]=templist4[i][j][k];
                        cl3ch5[c3]=templist5[i][j][k];
                        cl3[c3]=flist[i][j][k];
                        ref3[c3]=.7459+.347 * cl3ch1[c3] + .65 * cl3ch2[c3];
                        tp3[c3]=cl3ch4[c3]+(1+.58*(cl3ch4[c3]-cl3ch5[c3]))*(cl3ch4[c3]-cl3ch5[c3])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl3[c3])))+30*.004;
                        temp3+=cl3ch4[c3]+(1+.58*(cl3ch4[c3]-cl3ch5[c3]))*(cl3ch4[c3]-cl3ch5[c3])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl3[c3])))+30*.004;
                        ntp3[c3]=cl3ch4[c3]+2.13*(cl3ch4[c3]-cl3ch5[c3])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl3[c3])))-200*(-
                        .004);
                        ntemp3+=cl3ch4[c3]+2.13*(cl3ch4[c3]-cl3ch5[c3])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl1[c3])))-200*(-
                        .004);
                        refl3+=.7459+.347 * cl3ch1[c3] + .65 * cl3ch2[c3];
                        air3+=threshold[i][j][k];
                        c3++;
                        ccloud[3][k]+=1;
                }
                break;
case 4:
                if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
                {
                        clas4+=flist[i][j][k];
                        ch14+=chan1[i][j][k];
                        ch24+=chan2[i][j][k];
                        cl4ch1[c4]=chan1[i][j][k];
                        cl4ch2[c4]=chan2[i][j][k];
                        ch44+=templist4[i][j][k];
                        ch54+=templist5[i][j][k];
                        cl4ch4[c4]=templist4[i][j][k];
                        cl4ch5[c4]=templist5[i][j][k];
                        cl4[c4]=flist[i][j][k];
                        ref4[c4]=.7459+.347 * cl4ch1[c4] + .65 * cl4ch2[c4];
                        tp4[c4]=cl4ch4[c4]+(1+.58*(cl4ch4[c4]-cl4ch5[c4]))*(cl4ch4[c4]-cl4ch5[c4])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl4[c4])))+30*.004;
                        temp4+=cl4ch4[c4]+(1+.58*(cl4ch4[c4]-cl4ch5[c4]))*(cl4ch4[c4]-cl4ch5[c4])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl4[c4])))+30*.004;
                        ntp4[c4]=cl4ch4[c4]+2.13*(cl4ch4[c4]-cl4ch5[c4])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl4[c4])))-200*(-
                        .004);
                        ntemp4+=cl4ch4[c4]+2.13*(cl4ch4[c4]-cl4ch5[c4])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl4[c4])))-200*(-
                        .004);
                        refl4+=.7459+.347 * cl4ch1[c4] + .65 * cl4ch2[c4];
                        air4+=threshold[i][j][k];
                        c4++;
                        ccloud[4][k]+=1;
                }
                break;
case 5:
                if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
```

```
		{
			clas5+=flist[i][j][k];
			ch15+=chan1[i][j][k];
			ch25+=chan2[i][j][k];
			cl5ch1[c5]=chan1[i][j][k];
			cl5ch2[c5]=chan2[i][j][k];
			ch45+=templist4[i][j][k];
			ch55+=templist5[i][j][k];
			cl5ch4[c5]=templist4[i][j][k];
			cl5ch5[c5]=templist5[i][j][k];
			cl5[c5]=flist[i][j][k];
			ref5[c5]=.7459+.347 * cl5ch1[c5] + .65 * cl5ch2[c5];
			tp5[c5]=cl5ch4[c5]+(1+.58*(cl5ch4[c5]-cl5ch5[c5]))*(cl5ch4[c5]-cl5ch5[c5])+.51+40*(1-(1.0094+.047 *
			(float)(log((double)cl5[c5]))))+30*.004;
			temp5+=cl5ch4[c5]+(1+.58*(cl5ch4[c5]-cl5ch5[c5]))*(cl5ch4[c5]-cl5ch5[c5])+.51+40*(1-(1.0094+.047 *
			(float)(log((double)cl5[c5]))))+30*.004;
			ntp5[c5]=cl5ch4[c5]+2.13*(cl5ch4[c5]-cl5ch5[c5])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl5[c5])))-200*(-
			.004);
			ntemp5+=cl5ch4[c5]+2.13*(cl5ch4[c5]-cl5ch5[c5])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl5[c5])))-200*(-
			.004);
			refl5+=.7459+.347 * cl5ch1[c5] + .65 * cl5ch2[c5];
			air5+=threshold[i][j][k];
			c5++;
			ccloud[5][k]+=1;
		}
		break;
case 6:
		if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
		{
			clas6+=flist[i][j][k];
			ch16+=chan1[i][j][k];
			ch26+=chan2[i][j][k];
			cl6ch1[c6]=chan1[i][j][k];
			cl6ch2[c6]=chan2[i][j][k];
			ch46+=templist4[i][j][k];
			ch56+=templist5[i][j][k];
			cl6ch4[c6]=templist4[i][j][k];
			cl6ch5[c6]=templist5[i][j][k];
			cl6[c6]=flist[i][j][k];
			ref6[c6]=.7459+.347 * cl6ch1[c6] + .65 * cl6ch2[c6];
			tp6[c6]=cl6ch4[c6]+(1+.58*(cl6ch4[c6]-cl6ch5[c6]))*(cl6ch4[c6]-cl6ch5[c6])+.51+40*(1-(1.0094+.047 *
			(float)(log((double)cl6[c6]))))+30*.004;
			temp6+=cl6ch4[c6]+(1+.58*(cl6ch4[c6]-cl6ch5[c6]))*(cl6ch4[c6]-cl6ch5[c6])+.51+40*(1-(1.0094+.047 *
			(float)(log((double)cl6[c6]))))+30*.004;
			ntp6[c6]=cl6ch4[c6]+2.13*(cl6ch4[c6]-cl6ch5[c6])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl6[c6])))-200*(-
			.004);
			ntemp6+=cl6ch4[c6]+2.13*(cl6ch4[c6]-cl6ch5[c6])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl6[c6])))-200*(-
			.004);
			refl6+=.7459+.347 * cl6ch1[c6] + .65 * cl6ch2[c6];
			air6+=threshold[i][j][k];
			c6++;
			ccloud[6][k]+=1;
		}
		break;
case 7:
		if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
		{
			clas7+=flist[i][j][k];
			ch17+=chan1[i][j][k];
			ch27+=chan2[i][j][k];
			cl7ch1[c7]=chan1[i][j][k];
			cl7ch2[c7]=chan2[i][j][k];
			ch47+=templist4[i][j][k];
			ch57+=templist5[i][j][k];
			cl7ch4[c7]=templist4[i][j][k];
			cl7ch5[c7]=templist5[i][j][k];
			cl7[c7]=flist[i][j][k];
			ref7[c7]=.7459+.347 * cl7ch1[c7] + .65 * cl7ch2[c7];
			tp7[c7]=cl7ch4[c7]+(1+.58*(cl7ch4[c7]-cl7ch5[c7]))*(cl7ch4[c7]-cl7ch5[c7])+.51+40*(1-(1.0094+.047 *
			(float)(log((double)cl7[c7]))))+30*.004;
			temp7+=cl7ch4[c7]+(1+.58*(cl7ch4[c7]-cl7ch5[c7]))*(cl7ch4[c7]-cl7ch5[c7])+.51+40*(1-(1.0094+.047 *
			(float)(log((double)cl7[c7]))))+30*.004;
			ntp7[c7]=cl7ch4[c7]+2.13*(cl7ch4[c7]-cl7ch5[c7])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl7[c7])))-200*(-
			.004);
			ntemp7+=cl7ch4[c7]+2.13*(cl7ch4[c7]-cl7ch5[c7])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl7[c7])))-200*(-
			.004);
			refl7+=.7459+.347 * cl7ch1[c7] + .65 * cl7ch2[c7];
			air7+=threshold[i][j][k];
			c7++;
			ccloud[7][k]+=1;
		}
		break;
case 8:
		if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
		{
			clas8+=flist[i][j][k];
			ch18+=chan1[i][j][k];
			ch28+=chan2[i][j][k];
			cl8ch1[c8]=chan1[i][j][k];
			cl8ch2[c8]=chan2[i][j][k];
			ch48+=templist4[i][j][k];
			ch58+=templist5[i][j][k];
			cl8ch4[c8]=templist4[i][j][k];
			cl8ch5[c8]=templist5[i][j][k];
			cl8[c8]=flist[i][j][k];
			ref8[c8]=.7459+.347 * cl8ch1[c8] + .65 * cl8ch2[c8];
```

```
                        tp8[c8]=cl8ch4[c8]+(1+.58*(cl8ch4[c8]-cl8ch5[c8]))*(cl8ch4[c8]-cl8ch5[c8])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl8[c8])))+30*.004;
                        temp8+=cl8ch4[c8]+(1+.58*(cl8ch4[c8]-cl8ch5[c8]))*(cl8ch4[c8]-cl8ch5[c8])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl8[c8])))+30*.004;
                        ntp8[c8]=cl8ch4[c8]+2.13*(cl1ch4[c8]-cl8ch5[c8])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl8[c8])))-200*(-
                        .004);
                        ntemp8+=cl8ch4[c8]+2.13*(cl8ch4[c8]-cl8ch5[c8])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl8[c8])))-200*(-
                        .004);
                        refl8+=.7459+.347 * cl8ch1[c8] + .65 * cl8ch2[c8];
                        air8+=threshold[i][j][k];
                        c8++;
                        ccloud[8][k]+=1;
                }
                break;
case 9:
                if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
                {
                        clas9+=flist[i][j][k];
                        ch19+=chan1[i][j][k];
                        ch29+=chan2[i][j][k];
                        cl9ch1[c9]=chan1[i][j][k];
                        cl9ch2[c9]=chan2[i][j][k];
                        ch49+=templist4[i][j][k];
                        ch59+=templist5[i][j][k];
                        cl9ch4[c9]=templist4[i][j][k];
                        cl9ch5[c9]=templist5[i][j][k];
                        cl9[c9]=flist[i][j][k];
                        ref9[c9]=.7459+.347 * cl9ch1[c9] + .65 * cl9ch2[c9];
                        tp9[c9]=cl9ch4[c9]+(1+.58*(cl9ch4[c9]-cl9ch5[c9]))*(cl9ch4[c9]-cl9ch5[c9])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl9[c9])))+30*.004;
                        temp9+=cl9ch4[c9]+(1+.58*(cl9ch4[c9]-cl9ch5[c9]))*(cl9ch4[c9]-cl9ch5[c9])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl9[c9])))+30*.004;
                        ntp9[c9]=cl9ch4[c9]+2.13*(cl9ch4[c9]-cl9ch5[c9])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl9[c9])))-200*(-
                        .004);
                        ntemp9+=cl9ch4[c9]+2.13*(cl9ch4[c9]-cl9ch5[c9])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl9[c9])))-200*(-
                        .004);
                        refl9+=.7459+.347 * cl9ch1[c9] + .65 * cl9ch2[c9];
                        air9+=threshold[i][j][k];
                        c9++;
                        ccloud[9][k]+=1;
                }
                break;
case 10:
                if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100)
                {
                        clas10+=flist[i][j][k];
                        ch110+=chan1[i][j][k];
                        ch210+=chan2[i][j][k];
                        cl10ch1[c10]=chan1[i][j][k];
                        cl10ch2[c10]=chan2[i][j][k];
                        ch410+=templist4[i][j][k];
                        ch510+=templist5[i][j][k];
                        cl10ch4[c10]=templist4[i][j][k];
                        cl10ch5[c10]=templist5[i][j][k];
                        cl10[c10]=flist[i][j][k];
                        ref10[c10]=.7459+.347 * cl10ch1[c10] + .65 * cl10ch2[c10];
                        tp10[c10]=cl10ch4[c10]+(1+.58*(cl10ch4[c10]-cl10ch5[c10]))*(cl10ch4[c10]-cl10ch5[c10])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl10[c10])))+30*.004;
                        temp10+=cl10ch4[c10]+(1+.58*(cl10ch4[c10]-cl10ch5[c10]))*(cl10ch4[c10]-cl10ch5[c10])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl10[c10])))+30*.004;
                        ntp10[c10]=cl10ch4[c10]+2.13*(cl10ch4[c10]-cl10ch5[c10])+.18+50*(1-(1.0094+.047 *
                        (float)(log((double)cl10[c10])))-200*(-.004);
                        ntemp10+=cl10ch4[c10]+2.13*(cl10ch4[c10]-cl10ch5[c10])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl10[c10])))-
                        200*(-.004);
                        refl10+=.7459+.347 * cl10ch1[c10] + .65 * cl10ch2[c10];
                        air10+=threshold[i][j][k];
                        c10++;
                        ccloud[10][k]+=1;
                }
                break;
case 12:
                if((short)flist[i][j][k] !=-32 && tdlist[i][j][k]!=-2.0 && qdlist[i][j][k]!=-100 )
                {
                        clas12+=flist[i][j][k];
                        ch112+=chan1[i][j][k];
                        ch212+=chan2[i][j][k];
                        cl12ch1[c12]=chan1[i][j][k];
                        cl12ch2[c12]=chan2[i][j][k];
                        ch412+=templist4[i][j][k];
                        ch512+=templist5[i][j][k];
                        cl12ch4[c12]=templist4[i][j][k];
                        cl12ch5[c12]=templist5[i][j][k];
                        cl12[c12]=flist[i][j][k];
                        ref12[c12]=.7459+.347 * cl12ch1[c12] + .65 * cl12ch2[c12];
                        refl12+=.7459+.347 * cl12ch1[c12] + .65 * cl12ch2[c12];
                        tp12[c12]=cl12ch4[c12]+(1+.58*(cl12ch4[c12]-cl12ch5[c12]))*(cl12ch4[c12]-cl12ch5[c12])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl12[c12])))+30*.004;
                        temp12+=cl12ch4[c12]+(1+.58*(cl12ch4[c12]-cl12ch5[c12]))*(cl12ch4[c12]-cl12ch5[c12])+.51+40*(1-(1.0094+.047 *
                        (float)(log((double)cl12[c12])))+30*.004;
                        ntp12[c12]=cl12ch4[c12]+2.13*(cl12ch4[c12]-cl12ch5[c12])+.18+50*(1-(1.0094+.047 *
                        (float)(log((double)cl12[c12])))-200*(-.004);
                        ntemp12+=cl12ch4[c12]+2.13*(cl12ch4[c12]-cl12ch5[c12])+.18+50*(1-(1.0094+.047 * (float)(log((double)cl12[c12])))-
                        200*(-.004);
                        air12+=threshold[i][j][k];
                        c12++;
                        ccloud[12][k]+=1;
```

```
                }
                break;
}
}
if(c1!=0)
{
                av[1][k]=clas1/(c1);
                airtemp[1][k]=air1/c1;
                avchan1[1][k]=ch11/c1;
                avchan2[1][k]=ch21/c1;
                avchan4[1][k]=ch41/c1;
                avchan5[1][k]=ch51/c1;
                avtp1=temp1/c1;
                avref1=refl1/c1;
                avntp1=ntemp1/c1;
                ac=0;
                count1=count2=count3=count4=count5=count6=0;
                std1=std_calc(cl1,av[1][k],c1);
                for(h=0;h<c1;h++)
                {
                  absval=(float)(fabs((double)(av[1][k]-cl1[h])));
                  if(absval<std1)
                  {
                                ac+=cl1[h];
                                count1+=1;
                  }
                }
if(count1!=0)
  av[1][k]=ac/count1;
  ac=0;
  std1=std_calc(cl1ch1,avchan1[1][k],c1);
  for(h=0;h<c1;h++)
  {
                absval=(float)(fabs((double)(avchan1[1][k]-cl1ch1[h])));
                if(absval<std1)
                {
                                ac+=cl1ch1[h];
                                count3+=1;
                }
  }
if(count3!=0)
  avchan1[1][k]=ac/count3;

 ac=0;
  std1=std_calc(cl1ch2,avchan2[1][k],c1);
  for(h=0;h<c1;h++)
  {
                absval=(float)(fabs((double)(avchan2[1][k]-cl1ch2[h])));
                if(absval<std1)
                {
                                ac+=cl1ch2[h];
                                count4+=1;
                }
  }
if(count4!=0)
  avchan2[1][k]=ac/count4;
 ac=0;
  std1=std_calc(cl1ch4,avchan4[1][k],c1);
  for(h=0;h<c1;h++)
  {
                absval=(float)(fabs((double)(avchan4[1][k]-cl1ch4[h])));
                if(absval<std1)
                {
                                ac+=cl1ch4[h];
                                count5+=1;
                }
  }
if(count5!=0)
  avchan4[1][k]=ac/count5;

 ac=0;
  std1=std_calc(cl1ch5,avchan5[1][k],c1);
  for(h=0;h<c1;h++)
  {
                absval=(float)(fabs((double)(avchan5[1][k]-cl1ch5[h])));
                if(absval<std1)
                {
                                ac+=cl1ch5[h];
                                count6+=1;
                }
  }
if(count6!=0)
  avchan5[1][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp1,avtp1,c1);
  for(h=0;h<c1;h++)
  {
                absval=(float)(fabs((double)(avtp1-tp1[h])));
                if(absval<std1)
                {
                                tp1[act]=tp1[h];
                                ac+=tp1[h];
                                act+=1;
                }
  }
```

```
if(act!=0)
{
                maxtemp[1][k]=get_max_value(tp1,act);
                avertemp[1][k]=ac/act;
                mintemp[1][k]=get_min_value(tp1,act);
}
ac=0;
act=0;
  std1=std_calc(ntp1,avntp1,c1);
  for(h=0;h<c1;h++)
  {
                absval=(float)(fabs((double)(avntp1-ntp1[h])));
                if(absval<std1)
                {

                                ac+=ntp1[h];
                                act+=1;
                }
  }
if(act!=0)
                navertemp[1][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref1,avref1,c1);
  for(h=0;h<c1;h++)
  {
                absval=(float)(fabs((double)(avref1-ref1[h])));
                if(absval<std1)
                {
                                ref1[acr]=ref1[h];
                                ac+=ref1[h];
                                acr+=1;
                }
  }
if(acr!=0)
{
                maxalb[1][k]=get_max_value(ref1,acr);
                averef[1][k]=ac/acr;
}

}

else
{

  av[1][k]=av[1][k-1];
  avchan1[1][k]=avchan1[1][k-1];
  avchan2[1][k]=avchan2[1][k-1];
                airtemp[1][k]=airtemp[1][k-1];
                maxtemp[1][k]=maxtemp[1][k-1];
                avertemp[1][k]=avertemp[1][k-1];
}
if(c2!=0)
{

   av[2][k]=clas2/(c2);
                airtemp[2][k]=air2/c2;
                avchan1[2][k]=ch12/c2;
                avchan2[2][k]=ch22/c2;
                avchan4[2][k]=ch42/c2;
                avchan5[2][k]=ch52/c2;
                avtp2=temp2/c2;
                avref2=refl2/c2;
                avntp2=ntemp2/c2;
                ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl2,av[2][k],c2);
  for(h=0;h<c2;h++)
  {
                absval=(float)(fabs((double)(av[2][k]-cl2[h])));
                if(absval< std1)
                {
                                ac+=cl2[h];
                                count1+=1;
                }
  }
if(count1!=0)
  av[2][k]=ac/count1;
ac=0;
  std1=std_calc(cl2ch1,avchan1[2][k],c2);
  for(h=0;h<c2;h++)
  {
                absval=(float)(fabs((double)(avchan1[2][k]-cl2ch1[h])));
                if(absval<std1)
                {
                                ac+=cl2ch1[h];
                                count3+=1;
                }
  }
if(count3!=0)
  avchan1[2][k]=ac/count3;

 ac=0;
  std1=std_calc(cl2ch2,avchan2[2][k],c2);
  for(h=0;h<c2;h++)
  {
```

```
                absval=(float)(fabs((double)(avchan2[2][k]-cl2ch2[h])));
                if(absval<std1)
                {
                        ac+=cl2ch2[h];
                        count4+=1;
                }
  }
if(count4!=0)
  avchan2[2][k]=ac/count4;

ac=0;
  std1=std_calc(cl2ch4,avchan4[2][k],c2);
  for(h=0;h<c2;h++)
  {
                absval=(float)(fabs((double)(avchan4[2][k]-cl2ch4[h])));
                if(absval<std1)
                {
                        ac+=cl2ch4[h];
                        count5+=1;
                }
  }
if(count5!=0)
  avchan4[2][k]=ac/count5;

 ac=0;
  std1=std_calc(cl2ch5,avchan5[2][k],c2);
  for(h=0;h<c2;h++)
  {
                absval=(float)(fabs((double)(avchan5[2][k]-cl2ch5[h])));
                if(absval<std1)
                {
                        ac+=cl2ch5[h];
                        count6+=1;
                }
  }
if(count6!=0)
  avchan5[2][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp2,avtp2,c2);
  for(h=0;h<c2;h++)
  {
                absval=(float)(fabs((double)(avtp2-tp2[h])));
                if(absval<std1)
                {
                        tp2[act]=tp2[h];
                        ac+=tp2[h];
                        act+=1;
                }
  }
if(act!=0)
{
  maxtemp[2][k]=get_max_value(tp2,act);
  avertemp[2][k]=ac/act;
  mintemp[2][k]=get_min_value(tp2,act);
}

ac=0;
act=0;
  std1=std_calc(ntp2,avntp2,c2);
  for(h=0;h<c2;h++)
  {
                absval=(float)(fabs((double)(avntp2-ntp2[h])));
                if(absval<std1)
                {

                        ac+=ntp2[h];
                        act+=1;
                }
  }
if(act!=0)
                navertemp[2][k]=ac/act;

ac=0;
acr=0;
  std1=std_calc(ref2,avref2,c2);
  for(h=0;h<c2;h++)
  {
                absval=(float)(fabs((double)(avref2-ref2[h])));
                if(absval<std1)
                {
                        ref2[acr]=ref2[h];
                        ac+=ref2[h];
                        acr+=1;
                }
  }
if(acr!=0)
{
  maxalb[2][k]=get_max_value(ref2,acr);
  averef[2][k]=ac/acr;
}
}
else
{
   av[2][k]=av[2][k-1];
                airtemp[2][k]=airtemp[2][k-1];
```

```
                avchan1[2][k]=avchan1[2][k-1];
                avchan2[2][k]=avchan2[2][k-1];
}
if(c3!=0)
{
          av[3][k]=clas3/(c3);
                airtemp[3][k]=air3/c3;
                avchan1[3][k]=ch13/c3;
                avchan2[3][k]=ch23/c3;
                avchan4[3][k]=ch43/c3;
                avchan5[3][k]=ch53/c3;
                avtp3=temp3/c3;
                avref3=refl3/c3;
                avntp3=ntemp3/c3;
                ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl3,av[3][k],c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(av[3][k]-cl3[h])));
                if(absval< std1)
                {
                          ac+=cl3[h];
                          count1+=1;
                }
  }
if(count1!=0)
  av[3][k]=ac/count1;
ac=0;
  std1=std_calc(cl3ch1,avchan1[3][k],c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(avchan1[3][k]-cl3ch1[h])));
                if(absval<std1)
                {
                          ac+=cl3ch1[h];
                          count3+=1;
                }
  }
if(count3!=0)
  avchan1[3][k]=ac/count3;

 ac=0;
  std1=std_calc(cl3ch2,avchan2[3][k],c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(avchan2[3][k]-cl3ch2[h])));
                if(absval<std1)
                {
                          ac+=cl3ch2[h];
                          count4+=1;
                }
  }
if(count4!=0)
  avchan2[3][k]=ac/count4;
  ac=0;
  std1=std_calc(cl3ch4,avchan4[3][k],c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(avchan4[3][k]-cl3ch4[h])));
                if(absval<std1)
                {
                          ac+=cl3ch4[h];
                          count5+=1;
                }
  }
if(count5!=0)
  avchan4[3][k]=ac/count5;

 ac=0;
  std1=std_calc(cl3ch5,avchan5[3][k],c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(avchan5[3][k]-cl3ch5[h])));
                if(absval<std1)
                {
                          ac+=cl3ch5[h];
                          count6+=1;
                }
  }
if(count6!=0)
  avchan5[3][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp3,avtp3,c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(avtp3-tp3[h])));
                if(absval<std1)
                {
                          tp3[act]=tp3[h];
                          ac+=tp3[h];
                          act+=1;
                }
  }
if(act!=0)
```

```
{
  maxtemp[3][k]=get_max_value(tp3,act);
  avertemp[3][k]=ac/act;
  mintemp[3][k]=get_min_value(tp3,act);
}

ac=0;
act=0;
  std1=std_calc(ntp3,avntp3,c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(avntp3-ntp3[h])));
                if(absval<std1)
                {
                        ac+=ntp3[h];
                        act+=1;
                }
  }
if(act!=0)
        navertemp[3][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref3,avref3,c3);
  for(h=0;h<c3;h++)
  {
                absval=(float)(fabs((double)(avref3-ref3[h])));
                if(absval<std1)
                {
                        ref3[acr]=ref3[h];
                        ac+=ref3[h];
                        acr+=1;
                }
  }
if(acr!=0)
{
  maxalb[3][k]=get_max_value(ref3,acr);
  averef[3][k]=ac/acr;
}
}
else
{
   av[3][k]=av[3][k-1];
        airtemp[3][k]=airtemp[3][k-1];
        avchan1[3][k]=avchan1[3][k-1];
        avchan2[3][k]=avchan2[3][k-1];
}
if(c4!=0)
{
        av[4][k]=clas4/(c4);
        airtemp[4][k]=air4/c4;
        avchan1[4][k]=ch14/c4;
        avchan2[4][k]=ch24/c4;
        avchan4[4][k]=ch44/c4;
        avchan5[4][k]=ch54/c4;
        avtp4=temp4/c4;
        avref4=refl4/c4;
        avntp4=ntemp4/c4;
        ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl4,av[4][k],c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(av[4][k]-cl4[h])));
                if(absval< std1)
                {
                        ac+=cl4[h];
                        count1+=1;
                }
  }
if(count1!=0)
  av[4][k]=ac/count1;
  ac=0;
  std1=std_calc(cl4ch1,avchan1[4][k],c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(avchan1[4][k]-cl4ch1[h])));
                if(absval<std1)
                {
                        ac+=cl4ch1[h];
                        count3+=1;
                }
  }
if(count3!=0)
  avchan1[4][k]=ac/count3;
  ac=0;
  std1=std_calc(cl4ch2,avchan2[4][k],c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(avchan2[4][k]-cl4ch2[h])));
                if(absval<std1)
                {
                        ac+=cl4ch2[h];
                        count4+=1;
                }
  }
if(count4!=0)
```

```
  avchan2[4][k]=ac/count4;
  ac=0;
  std1=std_calc(cl4ch4,avchan4[4][k],c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(avchan4[4][k]-cl4ch4[h])));
                if(absval<std1)
                {
                                ac+=cl4ch4[h];
                                count5+=1;
                }
  }
if(count5!=0)
  avchan4[4][k]=ac/count5;

 ac=0;
  std1=std_calc(cl4ch5,avchan5[4][k],c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(avchan5[4][k]-cl4ch5[h])));
                if(absval<std1)
                {
                                ac+=cl4ch5[h];
                                count6+=1;
                }
  }
if(count6!=0)
  avchan5[4][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp4,avtp4,c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(avtp4-tp4[h])));
                if(absval<std1)
                {
                                tp4[act]=tp4[h];
                                ac+=tp4[h];
                                act+=1;
                }
  }
if(act!=0)
{
  maxtemp[4][k]=get_max_value(tp4,act);
  avertemp[4][k]=ac/act;
  mintemp[4][k]=get_min_value(tp4,act);
}
ac=0;
act=0;
  std1=std_calc(ntp4,avntp4,c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(avntp4-ntp4[h])));
                if(absval<std1)
                {

                                ac+=ntp4[h];
                                act+=1;
                }
  }
if(act!=0)
                navertemp[4][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref4,avref4,c4);
  for(h=0;h<c4;h++)
  {
                absval=(float)(fabs((double)(avref4-ref4[h])));
                if(absval<std1)
                {
                                ref4[acr]=ref4[h];
                                ac+=ref4[h];
                                acr+=1;
                }
  }
if(acr!=0)
{
  maxalb[4][k]=get_max_value(ref4,acr);
  averef[4][k]=ac/acr;
}
}
else
{

  av[4][k]=av[4][k-1];
  airtemp[4][k]=airtemp[4][k-1];
  avchan1[4][k]=avchan1[4][k-1];
  avchan2[4][k]=avchan2[4][k-1];
}
if(c5!=0)
{
                av[5][k]=clas5/(c5);
                airtemp[5][k]=air5/c5;
                avchan1[5][k]=ch15/c5;
                avchan2[5][k]=ch25/c5;
                avchan4[5][k]=ch45/c5;
```

```
                avchan5[5][k]=ch55/c5;
                avtp5=temp5/c5;
                avref5=refl5/c5;
                avntp5=ntemp5/c5;
                ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl5,av[5][k],c5);
  for(h=0;h<c5;h++)
  {
                absval=(float)(fabs((double)(av[5][k]-cl5[h])));
                if(absval< std1)
                {
                                ac+=cl5[h];
                                count1+=1;
                }
  }
if(count1!=0)
  av[5][k]=ac/count1;
  ac=0;
  std1=std_calc(cl5ch1,avchan1[5][k],c5);
  for(h=0;h<c5;h++)
  {
                absval=(float)(fabs((double)(avchan1[5][k]-cl5ch1[h])));
                if(absval<std1)
                {
                                ac+=cl5ch1[h];
                                count3+=1;
                }
  }
if(count3!=0)
  avchan1[5][k]=ac/count3;
  ac=0;
  std1=std_calc(cl5ch2,avchan2[5][k],c5);
  for(h=0;h<c5;h++)
  {
                absval=(float)(fabs((double)(avchan2[5][k]-cl5ch2[h])));
                if(absval<std1)
                {
                                ac+=cl5ch2[h];
                                count4+=1;
                }
  }
if(count4!=0)
  avchan2[5][k]=ac/count4;
  ac=0;
  std1=std_calc(cl5ch4,avchan4[5][k],c5);
  for(h=0;h<c5;h++)
  {
                absval=(float)(fabs((double)(avchan4[5][k]-cl5ch4[h])));
                if(absval<std1)
                {
                                ac+=cl5ch4[h];
                                count5+=1;
                }
  }
if(count5!=0)
  avchan4[5][k]=ac/count5;
  ac=0;
  std1=std_calc(cl5ch5,avchan5[5][k],c5);
  for(h=0;h<c5;h++)
  {
                absval=(float)(fabs((double)(avchan5[5][k]-cl5ch5[h])));
                if(absval<std1)
                {
                                ac+=cl5ch5[h];
                                count6+=1;
                }
  }
if(count6!=0)
  avchan5[5][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp5,avtp5,c5);
  for(h=0;h<c5;h++)
  {
                absval=(float)(fabs((double)(avtp5-tp5[h])));
                if(absval<std1)
                {
                                tp5[act]=tp5[h];
                                ac+=tp5[h];
                                act+=1;
                }
  }
if(act!=0)
{
  maxtemp[5][k]=get_max_value(tp5,act);
  avertemp[5][k]=ac/act;
  mintemp[5][k]=get_min_value(tp5,act);
}
ac=0;
act=0;
  std1=std_calc(ntp5,avntp5,c5);
  for(h=0;h<c5;h++)
  {
                absval=(float)(fabs((double)(avntp5-ntp5[h])));
                if(absval<std1)
```

```
                {

                        ac+=ntp5[h];
                        act+=1;
                }
  }
if(act!=0)
        navertemp[5][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref5,avref5,c5);
  for(h=0;h<c5;h++)
  {
        absval=(float)(fabs((double)(avref5-ref5[h])));
        if(absval<std1)
        {
                ref5[acr]=ref5[h];
                ac+=ref5[h];
                acr+=1;
        }
  }
if(acr!=0)
{
  maxalb[5][k]=get_max_value(ref5,acr);
  averef[5][k]=ac/acr;
}
}
else
{

        av[5][k]=av[5][k-1];
        airtemp[5][k]=airtemp[5][k-1];
        avchan1[5][k]=avchan1[5][k-1];
        avchan2[5][k]=avchan2[5][k-1];
}
if(c6!=0)
{
        av[6][k]=clas6/(c6);
        airtemp[6][k]=air6/c6;
        avchan1[6][k]=ch16/c6;
        avchan2[6][k]=ch26/c6;
        avchan4[6][k]=ch46/c6;
        avchan5[6][k]=ch56/c6;
        avtp6=temp6/c6;
        avref6=refl6/c6;
        avntp6=ntemp6/c6;
        ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl6,av[6][k],c6);
  for(h=0;h<c6;h++)
  {
        absval=(float)(fabs((double)(av[6][k]-cl6[h])));
        if(absval< std1)
        {
                ac+=cl6[h];
                count1+=1;
        }
  }
if(count1!=0)
  av[6][k]=ac/count1;
ac=0;
  std1=std_calc(cl6ch1,avchan1[6][k],c6);
  for(h=0;h<c6;h++)
  {
        absval=(float)(fabs((double)(avchan1[6][k]-cl6ch1[h])));
        if(absval<std1)
        {
                ac+=cl6ch1[h];
                count3+=1;
        }
  }
if(count3!=0)
  avchan1[6][k]=ac/count3;
 ac=0;
  std1=std_calc(cl6ch2,avchan2[6][k],c6);
  for(h=0;h<c6;h++)
  {
        absval=(float)(fabs((double)(avchan2[6][k]-cl6ch2[h])));
        if(absval<std1)
        {
                ac+=cl6ch2[h];
                count4+=1;
        }
  }
if(count4!=0)
  avchan2[6][k]=ac/count4;
ac=0;
  std1=std_calc(cl6ch4,avchan4[6][k],c6);
  for(h=0;h<c6;h++)
  {
        absval=(float)(fabs((double)(avchan4[6][k]-cl6ch4[h])));
        if(absval<std1)
        {
                ac+=cl6ch4[h];
                count5+=1;
        }
```

```
  }
if(count5!=0)
  avchan4[6][k]=ac/count5;
 ac=0;
  std1=std_calc(cl6ch5,avchan5[6][k],c6);
  for(h=0;h<c6;h++)
  {
                absval=(float)(fabs((double)(avchan5[6][k]-cl6ch5[h])));
                if(absval<std1)
                {
                                ac+=cl6ch5[h];
                                count6+=1;
                }
  }
if(count6!=0)
  avchan5[6][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp6,avtp6,c6);
  for(h=0;h<c6;h++)
  {
                absval=(float)(fabs((double)(avtp6-tp6[h])));
                if(absval<std1)
                {
                                tp6[act]=tp6[h];
                                ac+=tp6[h];
                                act+=1;
                }
  }
if(act!=0)
{
  maxtemp[6][k]=get_max_value(tp6,act);
  avertemp[6][k]=ac/act;
  mintemp[6][k]=get_min_value(tp6,act);
}
ac=0;
act=0;
  std1=std_calc(ntp6,avntp6,c6);
  for(h=0;h<c6;h++)
  {
                absval=(float)(fabs((double)(avntp6-ntp6[h])));
                if(absval<std1)
                {

                                ac+=ntp6[h];
                                act+=1;
                }
  }
if(act!=0)
                navertemp[6][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref6,avref6,c6);
  for(h=0;h<c6;h++)
  {
                absval=(float)(fabs((double)(avref6-ref6[h])));
                if(absval<std1)
                {
                                ref6[acr]=ref6[h];
                                ac+=ref6[h];
                                acr+=1;
                }
  }
if(acr!=0)
{
  maxalb[6][k]=get_max_value(ref6,acr);
  averef[6][k]=ac/acr;
}
}

else
{
                av[6][k]=av[6][k-1];
                airtemp[6][k]=airtemp[6][k-1];
                avchan1[6][k]=avchan1[6][k-1];
                avchan2[6][k]=avchan2[6][k-1];
}
if(c7!=0)
{
                av[7][k]=clas7/(c7);
                airtemp[7][k]=air7/c7;
                avchan1[7][k]=ch17/c7;
                avchan2[7][k]=ch27/c7;
                avchan4[7][k]=ch47/c7;
                avchan5[7][k]=ch57/c7;
                avtp7=temp7/c7;
                avref7=refl7/c7;
                avntp7=ntemp7/c7;
ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl7,av[7][k],c7);
  for(h=0;h<c7;h++)
  {
                absval=(float)(fabs((double)(av[7][k]-cl7[h])));
                if(absval< std1)
                {
```

```
                                ac+=cl7[h];
                                count1+=1;
                        }
  }
if(count1!=0)
  av[7][k]=ac/count1;
  ac=0;
  std1=std_calc(cl7ch1,avchan1[7][k],c7);
  for(h=0;h<c7;h++)
  {
                        absval=(float)(fabs((double)(avchan1[7][k]-cl7ch1[h])));
                        if(absval<std1)
                        {
                                ac+=cl7ch1[h];
                                count3+=1;
                        }
  }
if(count3!=0)
  avchan1[7][k]=ac/count3;
  ac=0;
  std1=std_calc(cl7ch2,avchan2[7][k],c7);
  for(h=0;h<c7;h++)
  {
                        absval=(float)(fabs((double)(avchan2[7][k]-cl7ch2[h])));
                        if(absval<std1)
                        {
                                ac+=cl7ch2[h];
                                count4+=1;
                        }
  }
if(count4!=0)
  avchan2[7][k]=ac/count4;
  ac=0;
  std1=std_calc(cl7ch4,avchan4[7][k],c7);
  for(h=0;h<c7;h++)
  {
                        absval=(float)(fabs((double)(avchan4[7][k]-cl7ch4[h])));
                        if(absval<std1)
                        {
                                ac+=cl7ch4[h];
                                count5+=1;
                        }
  }
if(count5!=0)
  avchan4[7][k]=ac/count5;
  ac=0;
  std1=std_calc(cl7ch5,avchan5[7][k],c7);
  for(h=0;h<c7;h++)
  {
                        absval=(float)(fabs((double)(avchan5[7][k]-cl7ch5[h])));
                        if(absval<std1)
                        {
                                ac+=cl7ch5[h];
                                count6+=1;
                        }
  }
if(count6!=0)
  avchan5[7][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp7,avtp7,c7);
  for(h=0;h<c7;h++)
  {
                        absval=(float)(fabs((double)(avtp7-tp7[h])));
                        if(absval<std1)
                        {
                                tp7[act]=tp7[h];
                                ac+=tp7[h];
                                act+=1;
                        }
  }
if(act!=0)
{
  maxtemp[7][k]=get_max_value(tp7,act);
  avertemp[7][k]=ac/act;
  mintemp[7][k]=get_min_value(tp7,act);
}
ac=0;
act=0;
  std1=std_calc(ntp7,avntp7,c7);
  for(h=0;h<c7;h++)
  {
                        absval=(float)(fabs((double)(avntp7-ntp7[h])));
                        if(absval<std1)
                        {

                                ac+=ntp7[h];
                                act+=1;
                        }
  }
if(act!=0)
                navertemp[7][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref7,avref7,c7);
  for(h=0;h<c7;h++)
```

```
  {
                absval=(float)(fabs((double)(avref7-ref7[h])));
                if(absval<std1)
                {
                        ref7[acr]=ref7[h];
                        ac+=ref7[h];
                        acr+=1;
                }
  }
if(acr!=0)
{
  maxalb[7][k]=get_max_value(ref7,acr);
  averef[7][k]=ac/acr;
}
}
else
{
                av[7][k]=av[7][k-1];
                airtemp[7][k]=airtemp[7][k-1];
                avchan1[7][k]=avchan1[7][k-1];
                avchan2[7][k]=avchan2[7][k-1];
}
if(c8!=0)
{
                av[8][k]=clas8/(c8);
                airtemp[8][k]=air8/c8;
                avchan1[8][k]=ch18/c8;
                avchan2[8][k]=ch28/c8;
                avchan4[8][k]=ch48/c8;
                avchan5[8][k]=ch58/c8;
                avtp8=temp8/c8;
                avref8=refl8/c8;
                avntp8=ntemp8/c8;
                ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl8,av[8][k],c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(av[8][k]-cl8[h])));
                if(absval< std1)
                {
                        ac+=cl8[h];
                        count1+=1;
                }
  }
if(count1!=0)
  av[8][k]=ac/count1;
ac=0;
  std1=std_calc(cl8ch1,avchan1[8][k],c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(avchan1[8][k]-cl8ch1[h])));
                if(absval<std1)
                {
                        ac+=cl8ch1[h];
                        count3+=1;
                }
  }
if(count3!=0)
  avchan1[8][k]=ac/count3;
 ac=0;
  std1=std_calc(cl8ch2,avchan2[8][k],c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(avchan2[8][k]-cl8ch2[h])));
                if(absval<std1)
                {
                        ac+=cl8ch2[h];
                        count4+=1;
                }
  }
if(count4!=0)
  avchan2[8][k]=ac/count4;
  ac=0;
  std1=std_calc(cl8ch4,avchan4[8][k],c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(avchan4[8][k]-cl8ch4[h])));
                if(absval<std1)
                {
                        ac+=cl8ch4[h];
                        count5+=1;
                }
  }
if(count5!=0)
  avchan4[8][k]=ac/count5;
  ac=0;
  std1=std_calc(cl8ch5,avchan5[8][k],c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(avchan5[8][k]-cl8ch5[h])));
                if(absval<std1)
                {
                        ac+=cl8ch5[h];
                        count6+=1;
                }
```

```
  }
if(count6!=0)
  avchan5[8][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp8,avtp8,c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(avtp8-tp8[h])));
                if(absval<std1)
                {
                                tp8[act]=tp8[h];
                                ac+=tp8[h];
                                act+=1;
                }
  }
if(act!=0)
{
  maxtemp[8][k]=get_max_value(tp8,act);
  avertemp[8][k]=ac/act;
  mintemp[8][k]=get_min_value(tp8,act);
}

ac=0;
act=0;
  std1=std_calc(ntp8,avntp8,c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(avntp8-ntp8[h])));
                if(absval<std1)
                {

                                ac+=ntp8[h];
                                act+=1;
                }
  }
if(act!=0)
              navertemp[8][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref8,avref8,c8);
  for(h=0;h<c8;h++)
  {
                absval=(float)(fabs((double)(avref8-ref8[h])));
                if(absval<std1)
                {
                                ref8[acr]=ref8[h];
                                ac+=ref8[h];
                                acr+=1;
                }
  }
if(acr!=0)
{
  maxalb[8][k]=get_max_value(ref8,acr);
  averef[8][k]=ac/acr;
}
}
else
{
              av[8][k]=av[8][k-1];
              airtemp[8][k]=airtemp[8][k-1];
              avchan1[8][k]=avchan1[8][k-1];
              avchan2[8][k]=avchan2[8][k-1];
}
if(c9!=0)
{
              av[9][k]=clas9/(c9);
              airtemp[9][k]=air9/c9;
              avchan1[9][k]=ch19/c9;
              avchan2[9][k]=ch29/c9;
              avchan4[9][k]=ch49/c9;
              avchan5[9][k]=ch59/c9;
              avtp9=temp9/c9;
              avref9=refl9/c9;
              avntp9=ntemp9/c9;
ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl9,av[9][k],c9);
  for(h=0;h<c9;h++)
  {
                absval=(float)(fabs((double)(av[9][k]-cl9[h])));
                if(absval< std1)
                {
                                ac+=cl9[h];
                                count1+=1;
                }
  }
if(count1!=0)
  av[9][k]=ac/count1;
  ac=0;
  std1=std_calc(cl9ch1,avchan1[9][k],c9);
  for(h=0;h<c9;h++)
  {
                absval=(float)(fabs((double)(avchan1[9][k]-cl9ch1[h])));
                if(absval<std1)
                {
```

```
                                        ac+=cl9ch1[h];
                                        count3+=1;
                        }
  }
if(count3!=0)
  avchan1[9][k]=ac/count3;
  ac=0;
  std1=std_calc(cl9ch2,avchan2[9][k],c9);
  for(h=0;h<c9;h++)
  {
                        absval=(float)(fabs((double)(avchan2[9][k]-cl9ch2[h])));
                        if(absval<std1)
                        {
                                        ac+=cl9ch2[h];
                                        count4+=1;
                        }
  }
if(count4!=0)
  avchan2[9][k]=ac/count4;
  ac=0;
  std1=std_calc(cl9ch4,avchan4[9][k],c9);
  for(h=0;h<c9;h++)
  {
                        absval=(float)(fabs((double)(avchan4[9][k]-cl9ch4[h])));
                        if(absval<std1)
                        {
                                        ac+=cl9ch4[h];
                                        count5+=1;
                        }
  }
if(count5!=0)
  avchan4[9][k]=ac/count5;
 ac=0;
  std1=std_calc(cl9ch5,avchan5[9][k],c9);
  for(h=0;h<c9;h++)
  {
                        absval=(float)(fabs((double)(avchan5[9][k]-cl9ch5[h])));
                        if(absval<std1)
                        {
                                        ac+=cl9ch5[h];
                                        count6+=1;
                        }
  }
if(count6!=0)
  avchan5[9][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp9,avtp9,c9);
  for(h=0;h<c9;h++)
  {
                        absval=(float)(fabs((double)(avtp9-tp9[h])));
                        if(absval<std1)
                        {
                                        tp9[act]=tp9[h];
                                        ac+=tp9[h];
                                        act+=1;
                        }
  }
if(act!=0)
{
  maxtemp[9][k]=get_max_value(tp9,act);
  avertemp[9][k]=ac/act;
  mintemp[9][k]=get_min_value(tp9,act);
}
ac=0;
act=0;
  std1=std_calc(ntp9,avntp9,c9);
  for(h=0;h<c9;h++)
  {
                        absval=(float)(fabs((double)(avntp9-ntp9[h])));
                        if(absval<std1)
                        {

                                        ac+=ntp9[h];
                                        act+=1;
                        }
  }
if(act!=0)
                navertemp[9][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref9,avref9,c9);
  for(h=0;h<c9;h++)
  {
                        absval=(float)(fabs((double)(avref9-ref9[h])));
                        if(absval<std1)
                        {
                                        ref9[acr]=ref9[h];
                                        ac+=ref9[h];
                                        acr+=1;
                        }
  }
if(acr!=0)
{
 maxalb[9][k]=get_max_value(ref9,acr);
 averef[9][k]=ac/acr;
```

```
}
}
else
{
		av[9][k]=av[9][k-1];
		airtemp[9][k]=airtemp[9][k-1];
		avchan1[9][k]=avchan1[9][k-1];
		avchan2[9][k]=avchan2[9][k-1];
}
if(c10!=0)
{
		av[10][k]=clas10/(c10);
		airtemp[10][k]=air10/c10;
		avchan1[10][k]=ch110/c10;
		avchan2[10][k]=ch210/c10;
		avchan4[10][k]=ch410/c10;
		avchan5[10][k]=ch510/c10;
		avtp10=temp10/c10;
		avref10=refl10/c10;
		avntp10=ntemp10/c10;
ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl1,av[10][k],c10);
  for(h=0;h<c10;h++)
  {
		absval=(float)(fabs((double)(av[10][k]-cl10[h])));
		if(absval< std1)
		{
			ac+=cl10[h];
			count1+=1;
		}
  }
if(count1!=0)
  av[10][k]=ac/count1;
  ac=0;
  std1=std_calc(cl10ch1,avchan1[10][k],c10);
  for(h=0;h<c10;h++)
  {
		absval=(float)(fabs((double)(avchan1[10][k]-cl10ch1[h])));
		if(absval<std1)
		{
			ac+=cl10ch1[h];
			count3+=1;
		}
  }
if(count3!=0)
  avchan1[10][k]=ac/count3;
  ac=0;
  std1=std_calc(cl10ch2,avchan2[10][k],c10);
  for(h=0;h<c10;h++)
  {
		absval=(float)(fabs((double)(avchan2[10][k]-cl10ch2[h])));
		if(absval<std1)
		{
			ac+=cl10ch2[h];
			count4+=1;
		}
  }
if(count4!=0)
  avchan2[10][k]=ac/count4;
  ac=0;
  std1=std_calc(cl10ch4,avchan4[10][k],c10);
  for(h=0;h<c10;h++)
  {
		absval=(float)(fabs((double)(avchan4[10][k]-cl10ch4[h])));
		if(absval<std1)
		{
			ac+=cl10ch4[h];
			count5+=1;
		}
  }
if(count5!=0)
  avchan4[10][k]=ac/count5;
  ac=0;
  std1=std_calc(cl10ch5,avchan5[10][k],c10);
  for(h=0;h<c10;h++)
  {
		absval=(float)(fabs((double)(avchan5[10][k]-cl10ch5[h])));
		if(absval<std1)
		{
			ac+=cl10ch5[h];
			count6+=1;
		}
  }
if(count6!=0)
  avchan5[10][k]=ac/count6;
ac=0;
act=0;
  std1=std_calc(tp10,avtp10,c10);
  for(h=0;h<c10;h++)
  {
		absval=(float)(fabs((double)(avtp10-tp10[h])));
		if(absval<std1)
		{
			tp10[act]=tp10[h];
			ac+=tp10[h];
```

```
                                act+=1;
                        }
  }
if(act!=0)
{
  maxtemp[10][k]=get_max_value(tp10,act);
  avertemp[10][k]=ac/act;
  mintemp[10][k]=get_min_value(tp10,act);
}
ac=0;
act=0;
  std1=std_calc(ntp10,avntp10,c10);
  for(h=0;h<c10;h++)
  {
                absval=(float)(fabs((double)(avntp10-ntp10[h])));
                if(absval<std1)
                {

                                ac+=ntp10[h];
                                act+=1;
                }
  }
if(act!=0)
                navertemp[10][k]=ac/act;
ac=0;
acr=0;
  std1=std_calc(ref10,avref10,c10);
  for(h=0;h<c10;h++)
  {
                absval=(float)(fabs((double)(avref10-ref10[h])));
                if(absval<std1)
                {
                                ref10[acr]=ref10[h];
                                ac+=ref10[h];
                                acr+=1;
                }
  }
if(acr!=0)
{
  maxalb[10][k]=get_max_value(ref10,acr);
  averef[10][k]=ac/acr;
}
}
else
{
                av[10][k]=av[10][k-1];
                airtemp[10][k]=airtemp[10][k-1];
                avchan1[10][k]=avchan1[10][k-1];
                avchan2[10][k]=avchan2[10][k-1];
}
if(c12!=0)
{
   av[12][k]=clas12/(c12);
                airtemp[12][k]=air12/c12;
                avchan1[12][k]=ch112/c12;
                avchan2[12][k]=ch212/c12;
                avchan4[12][k]=ch412/c12;
                avchan5[12][k]=ch512/c12;
                avtp12=temp12/c12;
                avref12=refl12/c12;
                avntp12=ntemp12/c12;
ac=0;
  count1=count2=count3=count4=count5=count6=0;
  std1=std_calc(cl12,av[12][k],c12);
  for(h=0;h<c12;h++)
  {
                absval=(float)(fabs((double)(av[12][k]-cl12[h])));
                if(absval< std1)
                {
                                ac+=cl12[h];
                                count1+=1;
                }
  }
if(count1!=0)
  av[12][k]=ac/count1;
ac=0;
  std1=std_calc(cl12ch1,avchan1[12][k],c12);
  for(h=0;h<c12;h++)
  {
                absval=(float)(fabs((double)(avchan1[12][k]-cl12ch1[h])));
                if(absval<std1)
                {
                                ac+=cl12ch1[h];
                                count3+=1;
                }
  }
if(count3!=0)
  avchan1[12][k]=ac/count3;
 ac=0;
  std1=std_calc(cl12ch2,avchan2[12][k],c12);
  for(h=0;h<c12;h++)
  {
                absval=(float)(fabs((double)(avchan2[12][k]-cl12ch2[h])));
                if(absval<std1)
                {
                                ac+=cl12ch2[h];
```

```
                                count4+=1;
                        }
        }
if(count4!=0)
        avchan2[12][k]=ac/count4;
        ac=0;
        std1=std_calc(cl12ch4,avchan4[12][k],c12);
        for(h=0;h<c12;h++)
        {
                        absval=(float)(fabs((double)(avchan4[12][k]-cl12ch4[h])));
                        if(absval<std1)
                        {
                                ac+=cl12ch4[h];
                                count5+=1;
                        }
        }
if(count5!=0)
        avchan4[12][k]=ac/count5;
       ac=0;
        std1=std_calc(cl12ch5,avchan5[12][k],c12);
        for(h=0;h<c12;h++)
        {
                        absval=(float)(fabs((double)(avchan5[12][k]-cl12ch5[h])));
                        if(absval<std1)
                        {
                                ac+=cl12ch5[h];
                                count6+=1;
                        }
        }
if(count6!=0)
        avchan5[12][k]=ac/count6;
ac=0;
act=0;
        std1=std_calc(tp12,avtp12,c12);
        for(h=0;h<c12;h++)
        {
                        absval=(float)(fabs((double)(avtp12-tp12[h])));
                        if(absval<std1)
                        {
                                tp12[act]=tp12[h];
                                ac+=tp12[h];
                                act+=1;
                        }
        }
if(act!=0)
{
        maxtemp[12][k]=get_max_value(tp12,act);
        avertemp[12][k]=ac/act;
        mintemp[12][k]=get_min_value(tp12,act);
}
ac=0;
act=0;
        std1=std_calc(ntp12,avntp12,c12);
        for(h=0;h<c12;h++)
        {
                        absval=(float)(fabs((double)(avntp12-ntp12[h])));
                        if(absval<std1)
                        {

                                ac+=ntp12[h];
                                act+=1;
                        }
        }
if(act!=0)
                navertemp[12][k]=ac/act;
ac=0;
acr=0;
        std1=std_calc(ref12,avref12,c12);
        for(h=0;h<c12;h++)
        {
                        absval=(float)(fabs((double)(avref12-ref12[h])));
                        if(absval<std1)
                        {
                                ref12[acr]=ref12[h];
                                ac+=ref12[h];
                                acr+=1;
                        }
        }
if(acr!=0)
{
        maxalb[12][k]=get_max_value(ref12,acr);
        averef[12][k]=ac/acr;
}
}
else
{
                avtemp[12][k]=30;
                airtemp[12][k]=airtemp[12][k-1];
    av[12][k]=av[12][k-1];
                avchan1[12][k]=avchan1[12][k-1];
                avchan2[12][k]=avchan2[12][k-1];
}
cloud[1][k]=((float)acov1-(float)c1)/(float)acov1*100;
cloud[2][k]=((float)acov2-(float)c2)/(float)acov2*100;
cloud[3][k]=((float)acov3-(float)c3)/(float)acov3*100;
cloud[4][k]=((float)acov4-(float)c4)/(float)acov4*100;
```

```
cloud[5][k]=((float)acov5-(float)c5)/(float)acov5*100;
cloud[6][k]=((float)acov6-(float)c6)/(float)acov6*100;
cloud[7][k]=((float)acov7-(float)c7)/(float)acov7*100;
cloud[8][k]=((float)acov8-(float)c8)/(float)acov8*100;
cloud[9][k]=((float)acov9-(float)c9)/(float)acov9*100;
cloud[10][k]=((float)acov10-(float)c10)/(float)acov10*100;
cloud[12][k]=((float)acov12-(float)c12)/(float)acov12*100;
}
for(ch=1;ch<13;ch++)
if(av[ch][0]==0.0)
{
        av[ch][0]=(av[ch][1]+av[ch][2])/2;
        avchan1[ch][0]=(avchan1[ch][1]+avchan1[ch][2])/2;
        avchan2[ch][0]=(avchan2[ch][1]+avchan2[ch][2])/2;
}
for(k=1;k<strint;k++)
for(ch=1;ch<13;ch++)
if(av[ch][k]==0.0)
{
        av[ch][k]=(av[ch][k-1]+av[ch][k+1])/2;
        avchan1[ch][k]=(avchan1[ch][k-1]+avchan1[ch][k+1])/2;
        avchan2[ch][k]=(avchan2[ch][k-1]+avchan2[ch][k+1])/2;
}
for(k=0;k<strint;k++)
for(ch=1;ch<13;ch++)
alb[ch][k]=.7459+0.347 * avchan1[ch][k] + 0.65 * avchan2[ch][k];
for(ch=1;ch<13;ch++)
if(avchan4[ch][0]==0.0)
{
        avchan4[ch][0]=(avchan4[ch][1]+avchan4[ch][2])/2;
        avchan5[ch][0]=(avchan5[ch][1]+avchan5[ch][2])/2;
}
for(k=0;k<strint;k++)
for(ch=1;ch<13;ch++)
if(ch!=11)
if(maxtemp[ch][k]==0)
{
        maxtemp[ch][k]=(maxtemp[ch][k-1]+maxtemp[ch][k+1])/2;
   avertemp[ch][k]=(avertemp[ch][k-1]+avertemp[ch][k+1])/2;
        navertemp[ch][k]=(navertemp[ch][k-1]+navertemp[ch][k+1])/2;
}

for(k=0;k<strint;k++)
for(ch=1;ch<13;ch++)
if(ch!=11)
if(mintemp[ch][k]==0)
{
        mintemp[ch][k]=(mintemp[ch][k-1]+mintemp[ch][k+1])/2;
}

for(k=0;k<strint;k++)
for(ch=1;ch<13;ch++)
if(ch!=11)
if(maxalb[ch][k]==0)
{
        maxalb[ch][k]=(maxalb[ch][k-1]+maxalb[ch][k+1])/2;
   averef[ch][k]=(averef[ch][k-1]+averef[ch][k+1])/2;
}
for(k=1;k<newstrint;k++)
for(ch=1;ch<13;ch++)
if(ch!=11)
if(avchan5[ch][k]==0.0)
{
        avchan4[ch][k]=(avchan4[ch][k-1]+avchan4[ch][k+1])/2;
        avchan5[ch][k]=(avchan5[ch][k-1]+avchan5[ch][k+1])/2;
}
for(k=0;k<strint;k++)
for(ch=1;ch<13;ch++)
if(ch!=11)
{
surtemp[ch][k]=avchan4[ch][k]+(1+.58*(avchan4[ch][k]-avchan5[ch][k]))*(avchan4[ch][k]-avchan5[ch][k])+.51+40*(1-(1.0094+.047 *
(float)(log((double)av[ch][k]))))+30*.004;
nsurtemp[ch][k]=avchan4[ch][k]+2.13*(avchan4[ch][k]-avchan5[ch][k])+.18+50*(1-(1.0094+.047 * (float)(log((double)av[ch][k]))))-200*(-.004);
}
}
void lcover()
{
FILE *cov;
short i,j;
    if((cov=fopen("d:\\msc\\tmu1r.mp#","rb"))==NULL)
        {
                printf("cannot open the file");
                exit(1);
        }

          for(i=1;i<13;i++)
                  acov[i]=0;
          acov1=acov2=acov3=acov4=acov5=acov6=acov7=acov8=acov9=acov10=acov11=acov12=0;
          for(i=0;i<lin;i++)
                for(j=0;j<col;j++)
                {
                        fread(&cover[i][j], sizeof(cover[i][j]), 1, cov) ;
                        switch(cover[i][j])
                        {
                        case 1:
                                acov1+=1;
```

```
                                                acov[1]+=1;
                                                break;

                                        case 2:
                                                acov2+=1;
                                                acov[2]+=1;
                                                break;
                                        case 3:
                                                acov3+=1;
                                                acov[3]+=1;
                                                break;
                                        case 4:
                                                acov4+=1;
                                                acov[4]+=1;
                                                break;
                                        case 5:
                                                acov5+=1;
                                                acov[5]+=1;
                                                break;
                                        case 6:
                                                acov6+=1;
                                                acov[6]+=1;
                                                break;
                                        case 7:
                                                acov7+=1;
                                                acov[7]+=1;
                                                break;
                                        case 8:
                                                acov8+=1;
                                                acov[8]+=1;
                                                break;
                                        case 9:
                                                acov9+=1;
                                                acov[9]+=1;
                                                break;
                                        case 10:
                                                acov10+=1;
                                                acov[10]+=1;
                                                break;
                                        case 11:
                                                acov11+=1;
                                                acov[11]+=1;
                                                break;
                                        case 12:
                                                acov12+=1;
                                                acov[12]+=1;
                                                break;
                                        }
}
                        fclose(cov);
}

void read_thies_pol()
{
FILE *th;
short i,j;
    if((th=fopen("d:\\msc\\raindata\\thiepolf.mp#","rb"))==NULL)
            {
                    printf("cannot open the file");
                    exit(1);
            }

              for(i=0;i<lin;i++)
                    for(j=0;j<col;j++)
                      fread(&thpol[i][j], sizeof(thpol[i][j]), 1, th) ;
    fclose(th);
}
void fil_rest_of_array(float rest_arr[][128])
{
short k;
for(k=strint;k<newstrint;k++)
{
            rest_arr[1][k]=0;
            rest_arr[2][k]=0;
            rest_arr[3][k]=0;
            rest_arr[4][k]=0;
            rest_arr[5][k]=0;
            rest_arr[6][k]=0;
            rest_arr[7][k]=0;
            rest_arr[8][k]=0;
            rest_arr[9][k]=0;
            rest_arr[10][k]=0;
            rest_arr[11][k]=0;
            rest_arr[12][k]=0;
}
}
void four_for_classes(float four_arr[][128])
{
            float ang;
            short x,y,z,freq,i,l;
            for(i=1;i<13;i++)

                    {
                    l=0;
                    for(x=0,y=0,z=1;x<2*newstrint,y<newstrint,z<2*newstrint;x+=2,y++,z+=2)
                                            {
```

```
                                                tdata1[x]=four_arr[i][y];
                                                tdata1[z]=0;
                                        }
                                        four1(tdata1-1,newstrint,1);
                                        for(x=0,freq=1;x<2*newstrint,freq<2*newstrint;x+=2,freq+=2)
                                        {
                                                invfour[i][x]=tdata1[x];
                                                invfour[i][freq]=tdata1[freq];
                                                imagpart[i][l]=tdata1[freq];
                                                realpart[i][l]=tdata1[x];
                                                amp[i][x]=sqrt(pow(tdata1[x],2)+pow(tdata1[freq],2));
                                                ang=-tdata1[freq]/tdata1[x];
                                                phas[i][freq]=atan((double)ang);
                                                l+=1;
                                        }
                                }

        }

void arr_sort(float arr[],short flag)
{
        int out, in;
        float tp;
        for(out=0;out<flag-1;out++)
                for(in=out+1;in<flag;in++)
                        if(arr[out]>arr[in])
                        {
                                tp=arr[in];
                                arr[in]=arr[out];
                                arr[out]=tp;
                        }
}
void cal_med_array(float aver[][128],short flag)
{
        int i,j,k;
        float carr[5];
   float med2[15][128],med3[15][128],med4[15][128],med5[15][128];
        for(k=1;k<13;k++)

                        {
                                med4[k][0]=aver[k][0];
                                med4[k][1]=aver[k][1];
                                med4[k][strint-1]=aver[k][strint-1];
                                for(i=2;i<strint-1;i++)
                                {
                                        for(j=0;j<4;j++)
                                                carr[j]=aver[k][i+j-2];
                                        arr_sort(carr,4);
                                        med4[k][i]=(carr[1]+carr[2])/2;
                                }
                        }

                for(k=1;k<13;k++)

                        {
                                med2[k][0]=med4[k][0];
                                for(i=1;i<strint;i++)
                                        med2[k][i]=(med4[k][i]+med4[k][i-1])/2;
                        }

                for(k=1;k<13;k++)

                        {
                                med5[k][0]=med2[k][0];
                                med5[k][1]=med2[k][1];
                                med5[k][strint-1]=med2[k][strint-1];
                                med5[k][strint-2]=med2[k][strint-2];
                                for(i=2;i<strint-2;i++)
                                {
                                        for(j=0;j<5;j++)
                                                carr[j]=med2[k][i+j-2];
                                        arr_sort(carr,5);
                                        med5[k][i]=carr[2];
                                }
                        }
                for(k=1;k<13;k++)

                        {
                                med3[k][0]=med5[k][0];
                                med3[k][strint-1]=med5[k][strint-1];
                                for(i=1;i<strint-1;i++)
                                {
                                        for(j=0;j<3;j++)
                                                carr[j]=med5[k][i+j-1];
                                        arr_sort(carr,3);
                                        med3[k][i]=carr[1];
                                }
                        }
                switch(flag)
                {
                case 1:
                        for(k=1;k<13;k++)

                        {
                                final[k][0]=med3[k][0];
                                final[k][strint-1]=med3[k][strint-1];
```

```
                                        for(i=1;i<strint-1;i++)
                                        final[k][i]=.25*med3[k][i-1]+0.5*med3[k][i]+0.25*med3[k][i+1];
                                }
                        case 2:
                                for(k=1;k<13;k++)

                                {
                                        fresid[k][0]=med3[k][0];
                                        fresid[k][strint-1]=med3[k][strint-1];
                                        for(i=1;i<strint-1;i++)
                                        fresid[k][i]=.25*med3[k][i-1]+0.5*med3[k][i]+0.25*med3[k][i+1];
                                }
                        }
}

void calc_residuals(float fin[][128],float finav[][128])
{
        short i,j;
        for(i=1;i<13;i++)
                for(j=0;j<strint;j++)
                        resid[i][j]=fin[i][j]-finav[i][j];
}
void calc_final_smooth_s4253h()
{
        short i,j;
        for(i=1;i<13;i++)

                                for(j=0;j<newstrint;j++)
                                        finalf[i][j]=final[i][j]+fresid[i][j];
}

void inv_four_for_classes(float inv_four_arr[][256])
{
        float ang;
        short x,y,z,freq,i;
        for(i=1;i<13;i++)

                        {
                                for(x=0,y=0,z=1;x<2*newstrint,y<newstrint,z<2*newstrint;x+=2,y++,z+=2)
                                        {
                                                if(y<44)
                                                {
                                                        tdata1[x]=inv_four_arr[i][x];
                                                        tdata1[z]=inv_four_arr[i][z];
                                                }
        else
                                                {
                                                        tdata1[x]=0;
                                                        tdata1[z]=0;
                                                }
                                        }
                                        four1(tdata1-1,newstrint,-1);
                                        for(x=0,freq=1;x<2*newstrint,freq<2*newstrint;x+=2,freq+=2)
                                        {
                                                invfour[i][x]=tdata1[x];
                                                invfour[i][freq]=tdata1[freq];
                                        }
                        }

        }
float std_calc(float st[], float avr, short no)
{
float acc, std;
short i;
acc=0;
for(i=0;i<no;i++)
        acc+=pow((double)st[i],2);
std=sqrt(acc/no-pow(avr,2));
return(std);
}

void fil_rain_gilgil_arr()
{
FILE *fptr;
short i;
        if((fptr=fopen("d:\\msc\\raindata\\gilgil.txt","r"))==NULL)
  {
        printf("cannot open the file");
    exit(1);
  }
i=0;
while(getc(fptr)!=EOF)
{
fscanf(fptr,"%f",&gilgil[i]);
i+=1;
}
}
void fil_rain_naivwd_arr()
{
FILE *fptr;
short i;
        if((fptr=fopen("d:\\msc\\raindata\\naivwd.txt","r"))==NULL)
  {
        printf("cannot open the file");
    exit(1);
  }
```

```
i=0;
while(getc(fptr)!=EOF)
{
fscanf(fptr,"%f",&naivwd[i]);
i+=1;
}
}
void fil_rain_kinang_arr()
{
FILE *fptr;
short i;
        if((fptr=fopen("d:\\msc\\raindata\\nkinang.txt","r"))==NULL)
  {
        printf("cannot open the file");
    exit(1);
  }
i=0;
while(getc(fptr)!=EOF)
{
fscanf(fptr,"%f",&kinang[i]);
i+=1;
}
}
void fil_rain_olkalau_arr()
{
FILE *fptr;
short i;
        if((fptr=fopen("d:\\msc\\raindata\\olkalau.txt","r"))==NULL)
  {
        printf("cannot open the file");
    exit(1);
  }
i=0;
while(getc(fptr)!=EOF)
{
fscanf(fptr,"%f",&olkalau[i]);
i+=1;
}
}
void fil_class_rain()
{
        short i,j,k;
        float c12,c13,c16,c132,c22,c23,c26,c232,c32,c33,c36,c332,c42,c43,c46,c432,c52,c53,c56,c532;
        float c62,c63,c66,c632,c72,c73,c76,c732,c82,c83,c86,c832,c92,c93,c96,c932,c102,c103,c106,c1032;
        float c112,c113,c116,c1132,c122,c123,c126,c1232;
        c12=c13=c16=c132=c22=c23=c26=c232=c32=c33=c36=c332=c42=c43=c46=c432=c52=c53=c56=c532=0;
                c62=c63=c66=c632=c72=c73=c76=c732=c82=c83=c86=c832=c92=c93=c96=c932=c102=c103=c106=c1032=0;
                c112=c113=c116=c1132=c122=c123=c126=c1232=0;
                for(i=0;i<lin;i++)
                        for(j=0;j<col;j++)
                                switch(cover[i][j])
                                {
                                case 1:
                                        switch(thpol[i][j])
                                        {
                                        case 2:
                                                c12+=1;
                                                break;
                                        case 3:
                                                c13+=1;
                                                break;
                                        case 6:
                                                c16+=1;
                                                break;
                                        case 32:
                                                c132+=1;
                                                break;
                                        }
                                        break;
                                case 2:
                                        switch(thpol[i][j])
                                        {
                                        case 2:
                                                c22+=1;
                                                break;
                                        case 3:
                                                c23+=1;
                                                break;
                                        case 6:
                                                c26+=1;
                                                break;
                                        case 32:
                                                c232+=1;
                                                break;
                                        }
                                        break;
                                case 3:
                                        switch(thpol[i][j])
                                        {
                                        case 2:
                                                c32+=1;
                                                break;
                                        case 3:
                                                c33+=1;
                                                break;
                                        case 6:
```

```
                        c36+=1;
                        break;
                case 32:
                        c332+=1;
                        break;
                }
                break;
case 4:
                switch(thpol[i][j])
                {
                case 2:
                        c42+=1;
                        break;
                case 3:
                        c43+=1;
                        break;
                case 6:
                        c46+=1;
                        break;
                case 32:
                        c432+=1;
                        break;
                }
                break;
case 5:
                switch(thpol[i][j])
                {
                case 2:
                        c52+=1;
                        break;
                case 3:
                        c53+=1;
                        break;
                case 6:
                        c56+=1;
                        break;
                case 32:
                        c532+=1;
                        break;
                }
                break;
case 6:
                switch(thpol[i][j])
                {
                case 2:
                        c62+=1;
                        break;
                case 3:
                        c63+=1;
                        break;
                case 6:
                        c66+=1;
                        break;
                case 32:
                        c632+=1;
                        break;
                }
                break;
case 7:
                switch(thpol[i][j])
                {
                case 2:
                        c72+=1;
                        break;
                case 3:
                        c73+=1;
                        break;
                case 6:
                        c76+=1;
                        break;
                case 32:
                        c732+=1;
                        break;
                }
                break;
case 8:
                switch(thpol[i][j])
                {
                case 2:
                        c82+=1;
                        break;
                case 3:
                        c83+=1;
                        break;
                case 6:
                        c86+=1;
                        break;
                case 32:
                        c832+=1;
                        break;
                }
                break;
case 9:
                switch(thpol[i][j])
                {
                case 2:
```

```
                                                c92+=1;
                                                break;
                                        case 3:
                                                c93+=1;
                                                break;
                                        case 6:
                                                c96+=1;
                                                break;
                                        case 32:
                                                c932+=1;
                                                break;
                                        }
                                        break;
                                case 10:
                                        switch(thpol[i][j])
                                        {
                                        case 2:
                                                c102+=1;
                                                break;
                                        case 3:
                                                c103+=1;
                                                break;
                                        case 6:
                                                c106+=1;
                                                break;
                                        case 32:
                                                c1032+=1;
                                                break;
                                        }
                                        break;
                                case 11:
                                        switch(thpol[i][j])
                                        {
                                        case 2:
                                                c112+=1;
                                                break;
                                        case 3:
                                                c113+=1;
                                                break;
                                        case 6:
                                                c116+=1;
                                                break;
                                        case 32:
                                                c1132+=1;
                                                break;
                                        }
                                        break;
                                case 12:
                                        switch(thpol[i][j])
                                        {
                                        case 2:
                                                c122+=1;
                                                break;
                                        case 3:
                                                c123+=1;
                                                break;
                                        case 6:
                                                c126+=1;
                                                break;
                                        case 32:
                                                c1232+=1;
                                                break;
                                        }
                                        break;
                                }
        for(k=0;k<strint;k++)
        {
        rain[1][k]=c12/(c12+c13+c16+c132)*kinang[k]+c13/(c12+c13+c16+c132)*gilgil[k]+c16/(c12+c13+c16+c132)*olkalau[k]+c132/(c12
+c13+c16+c132)*naivwd[k];
        rain[2][k]=c22/(c22+c23+c26+c232)*kinang[k]+c23/(c22+c23+c26+c232)*gilgil[k]/(c22+c23+c26+c232)*olkalau[k]+c232/(c22+c23
+c26+c232)*naivwd[k];
        rain[3][k]=c32/(c32+c33+c36+c332)*kinang[k]+c33/(c32+c33+c36+c332)*gilgil[k]+c36/(c32+c33+c36+c332)*olkalau[k]+c332/(c32
+c33+c36+c332)*naivwd[k];
        rain[4][k]=c42/(c42+c43+c46+c432)*kinang[k]+c43/(c42+c43+c46+c432)*gilgil[k]+c46/(c42+c43+c46+c432)*olkalau[k]+c432/(c42
+c43+c46+c432)*naivwd[k];
        rain[5][k]=c52/(c52+c53+c56+c532)*kinang[k]+c53/(c52+c53+c56+c532)*gilgil[k]+c56/(c52+c53+c56+c532)*olkalau[k]+c532/(c52
+c53+c56+c532)*naivwd[k];
        rain[6][k]=c62/(c62+c63+c66+c632)*kinang[k]+c63/(c62+c63+c66+c632)*gilgil[k]+c66/(c62+c63+c66+c632)*olkalau[k]+c632/(c62
+c63+c66+c632)*naivwd[k];
        rain[7][k]=c72/(c72+c73+c76+c732)*kinang[k]+c73/(c72+c73+c76+c732)*gilgil[k]+c76/(c72+c73+c76+c732)*olkalau[k]+c732/(c72
+c73+c76+c732)*naivwd[k];
        rain[8][k]=c82/(c82+c83+c86+c832)*kinang[k]+c83/(c82+c83+c86+c832)*gilgil[k]+c86/(c82+c83+c86+c832)*olkalau[k]+c832/(c82
+c83+c86+c832)*naivwd[k];
        rain[9][k]=c92/(c92+c93+c96+c932)*kinang[k]+c93/(c92+c93+c96+c932)*gilgil[k]+c96/(c92+c93+c96+c932)*olkalau[k]+c932/(c92
+c93+c96+c932)*naivwd[k];
        rain[10][k]=c102/(c102+c103+c106+c1032)*kinang[k]+c103/(c102+c103+c106+c1032)*gilgil[k]+c106/(c102+c103+c106+c1032)*ol
kalau[k]+c1032/(c102+c103+c106+c1032)*naivwd[k];
        rain[11][k]=c112/(c112+c113+c116+c1132)*kinang[k]+c113/(c112+c113+c116+c1132)*gilgil[k]+c116/(c112+c113+c116+c1132)*ol
kalau[k]+c1132/(c112+c113+c116+c1132)*naivwd[k];
        rain[12][k]=c122/(c122+c123+c126+c1232)*kinang[k]+c123/(c122+c123+c126+c1232)*gilgil[k]+c126/(c122+c123+c126+c1232)*ol
kalau[k]+c1232/(c122+c123+c126+c1232)*naivwd[k];
        }
}
void calc_air_temp()
{
        short i,j,k;
```

```
		float c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12;
		float air1,air2,air3,air4,air5,air6,air7,air8,air9,air10,air11,air12;
for(k=0;k<strint;k++)
{
c1=c2=c3=c4=c5=c6=c7=c8=c9=c10=c11=c12=0;
air1=air2=air3=air4=air5=air6=air7=air8=air9=air10=air11=air12=0;
for(i=0;i<lin;i++)
for(j=0;j<col;j++)
switch(cover[i][j])
{
case 1:
				air1+=threshold[i][j][k];
				c1++;
		break;
case 2:
				air2+=threshold[i][j][k];
				c2++;
		break;
case 3:
				air3+=threshold[i][j][k];
				c3++;
		break;
case 4:
				air4+=threshold[i][j][k];
				c4++;
		break;
case 5:
				air5+=threshold[i][j][k];
				c5++;
		break;
case 6:
				air6+=threshold[i][j][k];
				c6++;
		break;
case 7:
				air7+=threshold[i][j][k];
				c7++;
		break;
case 8:
				air8+=threshold[i][j][k];
				c8++;
		break;
case 9:
				air9+=threshold[i][j][k];
				c9++;
		break;
case 10:
				air10+=threshold[i][j][k];
				c10++;
		break;
case 11:
				air11+=threshold[i][j][k];
				c11++;
		break;
case 12:
				air12+=threshold[i][j][k];
				c12++;
		break;

}
cairtemp[1][k]=air1/c1;
cairtemp[2][k]=air2/c2;
cairtemp[3][k]=air3/c3;
cairtemp[4][k]=air4/c4;
cairtemp[5][k]=air5/c5;
cairtemp[6][k]=air6/c6;
cairtemp[7][k]=air7/c7;
cairtemp[8][k]=air8/c8;
cairtemp[9][k]=air9/c9;
cairtemp[10][k]=air10/c10;
cairtemp[11][k]=air11/c11;
cairtemp[12][k]=air12/c12;
}

}
void redistribute_air_temp()
{
		short i,j,k;
		for(i=1;i<13;i++)
		{
				rairtemp[i][0]=surtemp[i][0]/(surtemp[i][0]+surtemp[i][1])*cairtemp[i][0];
				rairtemp[i][1]=surtemp[i][1]/(surtemp[i][0]+surtemp[i][1])*cairtemp[i][1];
		}
		for(i=1;i<13;i++)
				for(k=3;k<95;k+=3)
				{
						rairtemp[i][k-1]=surtemp[i][k-1]/(surtemp[i][k-1]+surtemp[i][k]+surtemp[i][k+1])*3*cairtemp[i][k];
						rairtemp[i][k]=surtemp[i][k]/(surtemp[i][k-1]+surtemp[i][k]+surtemp[i][k+1])*3*cairtemp[i][k];
						rairtemp[i][k+1]=surtemp[i][k+1]/(surtemp[i][k-1]+surtemp[i][k]+surtemp[i][k+1])*3*cairtemp[i][k];
				}
}
void calc_rain()
{
		short i,j,k;
		float c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12;
		float r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12;
```

```
		float d1,d2,d3,d4;
		for(k=0;k<strint;k++)
{
c1=c2=c3=c4=c5=c6=c7=c8=c9=c10=c11=c12=0;
r1=r2=r3=r4=r5=r6=r7=r8=r9=r10=r11=r12=0;
for(i=0;i<lin;i++)
for(j=0;j<col;j++)
switch(cover[i][j])
{
case 1:
				if(i==50 && j==60)
					c1=kinang[k];
				else if(i==66 && j==41)
					c1=naivwd[k];
				else if(i==24 && j==25)
					c1=gilgil[k];
				else if(i==14 && j==32)
					c1=olkalau[k];
				else
				{
					d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
					d2=((i-66)*(i-66)+(j-41)*(j-41));
					d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
					d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

		c1=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
				}
				r1+=c1;
				break;
case 2:
				if(i==50 && j==60)
					c2=kinang[k];
				else if(i==66 && j==41)
					c2=naivwd[k];
				else if(i==24 && j==25)
					c2=gilgil[k];
				else if(i==14 && j==32)
					c2=olkalau[k];
				else
				{
					d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
					d2=((i-66)*(i-66)+(j-41)*(j-41));
					d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
					d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));
		c2=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
				}
				r2+=c2;
		break;
case 3:
				if(i==50 && j==60)
					c3=kinang[k];
				else if(i==66 && j==41)
					c3=naivwd[k];
				else if(i==24 && j==25)
					c3=gilgil[k];
				else if(i==14 && j==32)
					c3=olkalau[k];
				else
				{
					d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
					d2=((i-66)*(i-66)+(j-41)*(j-41));
					d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
					d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

		c3=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
				}
				r3+=c3;
		break;
case 4:
				if(i==50 && j==60)
					c4=kinang[k];
				else if(i==66 && j==41)
					c4=naivwd[k];
				else if(i==24 && j==25)
					c4=gilgil[k];
				else if(i==14 && j==32)
					c4=olkalau[k];
				else
				{
					d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
					d2=((i-66)*(i-66)+(j-41)*(j-41));
					d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
					d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

		c4=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
				}
				r4+=c4;
		break;
case 5:
				if(i==50 && j==60)
					c5=kinang[k];
				else if(i==66 && j==41)
					c5=naivwd[k];
				else if(i==24 && j==25)
					c5=gilgil[k];
				else if(i==14 && j==32)
```

```
                        c5=olkalau[k];
                }
                else
                {
                        d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                        d2=((i-66)*(i-66)+(j-41)*(j-41));
                        d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                        d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

        c5=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                }
                r5+=c5;
        break;
case 6:
                if(i==50 && j==60)
                        c6=kinang[k];
                else if(i==66 && j==41)
                        c6=naivwd[k];
                else if(i==24 && j==25)
                        c6=gilgil[k];
                else if(i==14 && j==32)
                        c6=olkalau[k];
                else
                {
                        d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                        d2=((i-66)*(i-66)+(j-41)*(j-41));
                        d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                        d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

        c6=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                }
                r6+=c6;
        break;
case 7:
                if(i==50 && j==60)
                        c7=kinang[k];
                else if(i==66 && j==41)
                        c7=naivwd[k];
                else if(i==24 && j==25)
                        c7=gilgil[k];
                else if(i==14 && j==32)
                        c7=olkalau[k];
                else
                {
                        d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                        d2=((i-66)*(i-66)+(j-41)*(j-41));
                        d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                        d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

        c7=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                }
                r7+=c7;
        break;
case 8:
                if(i==50 && j==60)
                        c8=kinang[k];
                else if(i==66 && j==41)
                        c8=naivwd[k];
                else if(i==24 && j==25)
                        c8=gilgil[k];
                else if(i==14 && j==32)
                        c8=olkalau[k];
                else
                {
                        d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                        d2=((i-66)*(i-66)+(j-41)*(j-41));
                        d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                        d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

        c8=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                }
                r8+=c8;
        break;
case 9:
                if(i==50 && j==60)
                        c9=kinang[k];
                else if(i==66 && j==41)
                        c9=naivwd[k];
                else if(i==24 && j==25)
                        c9=gilgil[k];
                else if(i==14 && j==32)
                        c9=olkalau[k];
                else
                {
                        d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                        d2=((i-66)*(i-66)+(j-41)*(j-41));
                        d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                        d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));
        c9=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                }
                r9+=c9;
        break;
case 10:
                if(i==50 && j==60)
                        c10=kinang[k];
                else if(i==66 && j==41)
                        c10=naivwd[k];
```

```
                        else if(i==24 && j==25)
                                c10=gilgil[k];
                        else if(i==14 && j==32)
                                c10=olkalau[k];
                        else
                        {
                                d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                                d2=((i-66)*(i-66)+(j-41)*(j-41));
                                d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                                d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));

        c10=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                        }
                        r10+=c10;
                break;
case 11:
                        if(i==50 && j==60)
                                c11=kinang[k];
                        else if(i==66 && j==41)
                                c11=naivwd[k];
                        else if(i==24 && j==25)
                                c11=gilgil[k];
                        else if(i==14 && j==32)
                                c11=olkalau[k];
                        else
                        {
                                d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                                d2=((i-66)*(i-66)+(j-41)*(j-41));
                                d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                                d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));
        c11=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                        }
                        r11+=c11;
                break;
case 12:
                        if(i==50 && j==60)
                                c12=kinang[k];
                        else if(i==66 && j==41)
                                c12=naivwd[k];
                        else if(i==24 && j==25)
                                c12=gilgil[k];
                        else if(i==14 && j==32)
                                c12=olkalau[k];
                        else
                        {
                                d1=sqrt((i-50)*(i-50)+(j-60)*(j-60));
                                d2=((i-66)*(i-66)+(j-41)*(j-41));
                                d3=sqrt((i-24)*(i-24)+(j-25)*(j-25));
                                d4=sqrt((i-14)*(i-14)+(j-32)*(j-32));
        c12=d1/(d1+d2+d3+d4)*kinang[k]+d2/(d1+d2+d3+d4)*naivwd[k]+d3/(d1+d2+d3+d4)*gilgil[k]+d4/(d1+d2+d3+d4)*olkalau[k];
                        }
                                r12+=c12;
                        break;
}
raindist[1][k]=r1/acov1;
raindist[2][k]=r2/acov2;
raindist[3][k]=r3/acov3;
raindist[4][k]=r4/acov4;
raindist[5][k]=r5/acov5;
raindist[6][k]=r6/acov6;
raindist[7][k]=r7/acov7;
raindist[8][k]=r8/acov8;
raindist[9][k]=r9/acov9;
raindist[10][k]=r10/acov10;
raindist[11][k]=r11/acov11;
raindist[12][k]=r12/acov12;
}
}
void spatial_filter()
{
        short i,j,k;
        for(k=22;k<23;k++)
                for(i=1;i<lin-1;i++)
                        for(j=1;j<col-1;j++)
                                if((short)flist[i][j][k]==-32)
                                {
                                        ndvi[i][j][k]=flist[i][j][k];
                                        ptemp[i][j][k]=flist[i][j][k];
                                        palb[i][j][k]=flist[i][j][k];
                                }
                                else
                                {
                                if(tdlist[i][j][k]!=-2 && (short)flist[i][j][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i][j][k];
                                        fchan1[i][j][k]=chan1[i][j][k];
                                        fchan2[i][j][k]=chan2[i][j][k];
                                        fchan4[i][j][k]=templist4[i][j][k];
                                        fchan5[i][j][k]=templist5[i][j][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i-1][j-1][k]!=-2 && (short)flist[i-1][j-1][k]!=-32)
                                {
```

```
                                        ndvi[i][j][k]=flist[i-1][j-1][k];
                                        fchan1[i][j][k]=chan1[i-1][j-1][k];
                                        fchan2[i][j][k]=chan2[i-1][j-1][k];
                                        fchan4[i][j][k]=templist4[i-1][j-1][k];
                                        fchan5[i][j][k]=templist5[i-1][j-1][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i-1][j][k]!=-2 && (short)flist[i-1][j][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i-1][j][k];
                                        fchan1[i][j][k]=chan1[i-1][j][k];
                                        fchan2[i][j][k]=chan2[i-1][j][k];
                                        fchan4[i][j][k]=templist4[i-1][j][k];
                                        fchan5[i][j][k]=templist5[i-1][j][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i-1][j+1][k]!=-2 && (short)flist[i-1][j+1][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i-1][j+1][k];
                                        fchan1[i][j][k]=chan1[i-1][j+1][k];
                                        fchan2[i][j][k]=chan2[i-1][j+1][k];
                                        fchan4[i][j][k]=templist4[i-1][j+1][k];
                                        fchan5[i][j][k]=templist5[i-1][j+1][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i][j-1][k]!=-2 && (short)flist[i][j-1][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i][j-1][k];
                                        fchan1[i][j][k]=chan1[i][j-1][k];
                                        fchan2[i][j][k]=chan2[i][j-1][k];
                                        fchan4[i][j][k]=templist4[i][j-1][k];
                                        fchan5[i][j][k]=templist5[i][j-1][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i][j+1][k]!=-2 && (short)flist[i][j+1][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i][j+1][k];
                                        fchan1[i][j][k]=chan1[i][j+1][k];
                                        fchan2[i][j][k]=chan2[i][j+1][k];
                                        fchan4[i][j][k]=templist4[i][j+1][k];
                                        fchan5[i][j][k]=templist5[i][j+1][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i+1][j-1][k]!=-2 && (short)flist[i+1][j-1][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i+1][j-1][k];
                                        fchan1[i][j][k]=chan1[i+1][j-1][k];
                                        fchan2[i][j][k]=chan2[i+1][j-1][k];
                                        fchan4[i][j][k]=templist4[i+1][j-1][k];
                                        fchan5[i][j][k]=templist5[i+1][j-1][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i+1][j][k]!=-2 && (short)flist[i+1][j][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i+1][j][k];
                                        fchan1[i][j][k]=chan1[i+1][j][k];
                                        fchan2[i][j][k]=chan2[i+1][j][k];
                                        fchan4[i][j][k]=templist4[i+1][j][k];
                                        fchan5[i][j][k]=templist5[i+1][j][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else if(tdlist[i+1][j+1][k]!=-2 && (short)flist[i+1][j+1][k]!=-32)
                                {
                                        ndvi[i][j][k]=flist[i+1][j+1][k];
                                        fchan1[i][j][k]=chan1[i+1][j+1][k];
                                        fchan2[i][j][k]=chan2[i+1][j+1][k];
                                        fchan4[i][j][k]=templist4[i+1][j+1][k];
                                        fchan5[i][j][k]=templist5[i+1][j+1][k];
                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 * fchan2[i][j][k];
                                        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                        c1c2ratio[i][j][k]=fchan2[i][j][k]/fchan1[i][j][k];
                                }
                                else
                                {
```

```
switch(cover[i][j])
{

        case 1:
                ndvi[i][j][k]=av[1][k];
                fchan1[i][j][k]=avchan1[1][k];
                fchan2[i][j][k]=avchan2[1][k];
                fchan4[i][j][k]=avchan4[1][k];
                fchan5[i][j][k]=avchan5[1][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k])))+30*.004;
        c1c2ratio[i][j][k]=avchan2[1][k]/avchan1[1][k];
                break;
        case 2:
                ndvi[i][j][k]=av[2][k];
                fchan1[i][j][k]=avchan1[2][k];
                fchan2[i][j][k]=avchan2[2][k];
                fchan4[i][j][k]=avchan4[2][k];
                fchan5[i][j][k]=avchan5[2][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k])))+30*.004;
        c1c2ratio[i][j][k]=avchan2[2][k]/avchan1[2][k];
                break;
        case 3:
                ndvi[i][j][k]=av[3][k];
                fchan1[i][j][k]=avchan1[3][k];
                fchan2[i][j][k]=avchan2[3][k];
                fchan4[i][j][k]=avchan4[3][k];
                fchan5[i][j][k]=avchan5[3][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k])))+30*.004;
        c1c2ratio[i][j][k]=avchan2[3][k]/avchan1[3][k];
                break;
        case 4:
                ndvi[i][j][k]=av[4][k];
                fchan1[i][j][k]=avchan1[4][k];
                fchan2[i][j][k]=avchan2[4][k];
                fchan4[i][j][k]=avchan4[4][k];
                fchan5[i][j][k]=avchan5[4][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k])))+30*.004;
        c1c2ratio[i][j][k]=avchan2[4][k]/avchan1[4][k];
                break;
        case 5:
                ndvi[i][j][k]=av[5][k];
                fchan1[i][j][k]=avchan1[5][k];
                fchan2[i][j][k]=avchan2[5][k];
                fchan4[i][j][k]=avchan4[5][k];
                fchan5[i][j][k]=avchan5[5][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k])))+30*.004;
        c1c2ratio[i][j][k]=avchan2[5][k]/avchan1[5][k];
                break;
        case 6:
                ndvi[i][j][k]=av[6][k];
                fchan1[i][j][k]=avchan1[6][k];
                fchan2[i][j][k]=avchan2[6][k];
                fchan4[i][j][k]=avchan4[6][k];
                fchan5[i][j][k]=avchan5[6][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k])))+30*.004;
        c1c2ratio[i][j][k]=avchan2[6][k]/avchan1[6][k];
                break;
        case 7:
                ndvi[i][j][k]=av[7][k];
                fchan1[i][j][k]=avchan1[7][k];
                fchan2[i][j][k]=avchan2[7][k];
                fchan4[i][j][k]=avchan4[7][k];
                fchan5[i][j][k]=avchan5[7][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
        ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k])))+30*.004;
        c1c2ratio[i][j][k]=avchan2[7][k]/avchan1[7][k];
                break;
        case 8:
                ndvi[i][j][k]=av[8][k];
                fchan1[i][j][k]=avchan1[8][k];
                fchan2[i][j][k]=avchan2[8][k];
                fchan4[i][j][k]=avchan4[8][k];
                fchan5[i][j][k]=avchan5[8][k];
                palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
```

```
                                                ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                                c1c2ratio[i][j][k]=avchan2[8][k]/avchan1[8][k];
                                                        break;
                                                case 9:
                                                        ndvi[i][j][k]=av[9][k];
                                                        fchan1[i][j][k]=avchan1[9][k];
                                                        fchan2[i][j][k]=avchan2[9][k];
                                                        fchan4[i][j][k]=avchan4[9][k];
                                                        fchan5[i][j][k]=avchan5[9][k];
                                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
                                                ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                                c1c2ratio[i][j][k]=avchan2[9][k]/avchan1[9][k];
                                                        break;
                                                case 10:
                                                        ndvi[i][j][k]=av[10][k];
                                                        fchan1[i][j][k]=avchan1[10][k];
                                                        fchan2[i][j][k]=avchan2[10][k];
                                                        fchan4[i][j][k]=avchan4[10][k];
                                                        fchan5[i][j][k]=avchan5[10][k];
                                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
                                                ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                                c1c2ratio[i][j][k]=avchan2[10][k]/avchan1[10][k];
                                                        break;
                                                case 11:
                                                        ndvi[i][j][k]=av[11][k];
                                                        fchan1[i][j][k]=avchan1[11][k];
                                                        fchan2[i][j][k]=avchan2[11][k];
                                                        fchan4[i][j][k]=avchan4[11][k];
                                                        fchan5[i][j][k]=avchan5[11][k];
                                                        palb[i][j][k]=.7459+0.347 * fchan1[i][j][k] + 0.65 *
fchan2[i][j][k];
                                                ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                                c1c2ratio[i][j][k]=avchan2[11][k]/avchan1[11][k];
                                                        break;
                                                case 12:
                                                        ndvi[i][j][k]=av[12][k];
                                                        fchan1[i][j][k]=avchan1[12][k];
                                                        fchan2[i][j][k]=avchan2[12][k];
                                                        fchan4[i][j][k]=avchan4[12][k];
                                                        fchan5[i][j][k]=avchan5[12][k];
                                                        palb[i][j][k]=.7459+.347 * fchan1[i][j][k] + .65 *
fchan2[i][j][k];
                                                ptemp[i][j][k]=fchan4[i][j][k]+(1+.58*(fchan4[i][j][k]-
fchan5[i][j][k]))*(fchan4[i][j][k]-fchan5[i][j][k])+.51+40*(1-(1.0094+.047 * (float)(log((double)ndvi[i][j][k]))))+30*.004;
                                                c1c2ratio[i][j][k]=avchan2[12][k]/avchan1[12][k];
                                                        break;

                                                }
                                        }
}
}
void write_to_file()
{
FILE *ptrndvi,*ptralb,*ptrtemp,*c1c2r;
short i,j,k,nd,stemp,albedo,rc1c2;
char filendvi[50],filetemp[50],filealb[50], buf1[6],filext[5],filec1c2r[50];
for(k=22;k<23;k++)
{
strcpy(filendvi,"");
strcpy(filetemp,"");
strcpy(filealb,"");
strcpy(buf1,"");
strcpy(filext,"");
strcpy(filendvi,"d:\\msc\\ndvi");
strcpy(filext,".mp#");
itoa(k,buf1,10);
strcat(buf1,filext);
strcat(filendvi,buf1);
if( (ptrndvi = fopen(filendvi, "wb" )) == NULL )
   {
        printf("cannot open the file %s\n",filendvi );
        exit(1);
   }
strcpy(filetemp,"d:\\msc\\temp");
strcpy(filext,".mp#");
strcpy(buf1,"");
itoa(k,buf1,10);
strcat(buf1,filext);
strcat(filetemp,buf1);
if( (ptrtemp = fopen(filetemp, "wb" )) == NULL )
   {
        printf("cannot open the file %s\n",filetemp );
        exit(1);
   }
strcpy(filealb,"d:\\msc\\alb");
strcpy(filext,".mp#");
strcpy(buf1,"");
itoa(k,buf1,10);
strcat(buf1,filext);
strcat(filealb,buf1);
```

```
if( (ptralb = fopen(filealb, "wb" )) == NULL )
   {
            printf("cannot open the file %s\n",filealb );
            exit(1);
   }
strcpy(filec1c2r,"");
strcpy(filext,"");
strcpy(filec1c2r,"d:\\msc\\q");
strcpy(filext,".mp#");
strcpy(buf1,"");
itoa(k,buf1,10);
strcat(buf1,filext);
strcat(filec1c2r,buf1);
if( (c1c2r = fopen(filec1c2r, "wb" )) == NULL )
   {
            printf("cannot open the file %s\n",filetemp );
            exit(1);
   }
   for(i=0;i<lin;i++)
              for(j=0;j<col;j++)
                        {
      if(cover[i][j]==0)
                                    stemp=-32767;
     else
                         stemp=(short)(ptemp[i][j][k]/0.01);
                          fwrite(&stemp,sizeof(stemp),1,ptrtemp);
                        }
                for(i=0;i<lin;i++)
                 for(j=0;j<col;j++)
                        {
                      if(cover[i][j]==0)
                                    rc1c2=-32767;
                      else
                                 rc1c2=(short)(c1c2ratio[i][j][k]/0.01);
                         fwrite(&rc1c2,sizeof(rc1c2),1,c1c2r);
                        }
                        for(i=0;i<lin;i++)
                 for(j=0;j<col;j++)
                        {
                     if(cover[i][j]==0)
                                     albedo=-32767;
                         else
                                     albedo=(short)(palb[i][j][k]/0.1);
                         fwrite(&albedo,sizeof(albedo),1,ptralb);
                        }
                        for(i=0;i<lin;i++)
                 for(j=0;j<col;j++)
                        {
                      if((short)flist[i][j][k]==-32)
                                    nd=-32767;
                         else
                                    nd=(short)(ndvi[i][j][k]/0.001);
                         fwrite(&nd,sizeof(nd),1,ptrndvi);
                        }
}
for(i=6;i<7;i++)
for(j=15;j<35;j++)
printf("%.3f",c1c2ratio[i][j][0]);
}
void get_max_monthly(float maxmon[][128],float maxmontemp[][128], short flag)
{
            short i,j,k;
            switch(flag)
            {
            case 1:
            for(i=1;i<13;i++)
            {
                        if(maxmon[i][0]>maxmon[i][1])
                        {
                                    mondvi[i][0]=maxmon[i][0];
                                    amontemp[i][0]=maxmontemp[i][0];
                        }
                        else
                        {
                                    mondvi[i][0]=maxmon[i][1];
                                    amontemp[i][0]=maxmontemp[i][1];
                        }
                        if(maxmontemp[i][0]>maxmontemp[i][1])
                                    montemp[i][0]=maxmontemp[i][0];
                        else
                                    montemp[i][0]=maxmontemp[i][1];

                        j=1;
                        for(k=3;k<strint;k+=3)
                        {
                                    mondvi[i][j]=maxmon[i][k];
                                    montemp[i][j]=maxmontemp[i][k];
                                    amontemp[i][j]=maxmontemp[i][k];
                                    if(mondvi[i][j]<maxmon[i][k-1])
                                    {
                                                mondvi[i][j]=maxmon[i][k-1];
                                                amontemp[i][j]=maxmontemp[i][k-1];
                                    }
                                    if(montemp[i][j]<maxmontemp[i][k-1])
                                    {
                                                montemp[i][j]=maxmontemp[i][k-1];
```

```
                                        }
                                        if(mondvi[i][j]<maxmon[i][k+1])
                                        {
                                                mondvi[i][j]=maxmon[i][k+1];
                                                amontemp[i][j]=maxmontemp[i][k+1];
                                        }
                                        if(montemp[i][j]<maxmontemp[i][k+1])
                                        {
                                                montemp[i][j]=maxmontemp[i][k+1];
                                        }

                                        j+=1;
                                }
                        }
                        break;
                        case 2:
                        for(i=1;i<13;i++)
                        {
                                if(maxmon[i][0]>maxmon[i][1])
                                {
                                        fmondvi[i][0]=maxmon[i][0];
                                        famontemp[i][0]=maxmontemp[i][0];
                                }
                                else
                                {
                                        fmondvi[i][0]=maxmon[i][1];
                                        famontemp[i][0]=maxmontemp[i][1];
                                }
                                if(maxmontemp[i][0]>maxmontemp[i][1])
                                        fmontemp[i][0]=maxmontemp[i][0];
                                else
                                        fmontemp[i][0]=maxmontemp[i][1];
                                j=1;
                                for(k=3;k<strint;k+=3)
                                {
                                        fmondvi[i][j]=maxmon[i][k];
                                        fmontemp[i][j]=maxmontemp[i][k];
                                        famontemp[i][j]=maxmontemp[i][k];
                                        if(fmondvi[i][j]<maxmon[i][k-1])
                                        {
                                                fmondvi[i][j]=maxmon[i][k-1];
                                                famontemp[i][j]=maxmontemp[i][k-1];
                                        }
                                        if(fmontemp[i][j]<maxmontemp[i][k-1])
                                        {
                                                fmontemp[i][j]=maxmontemp[i][k-1];
                                        }
                                        if(fmondvi[i][j]<maxmon[i][k+1])
                                        {
                                                fmondvi[i][j]=maxmon[i][k+1];
                                                famontemp[i][j]=maxmontemp[i][k+1];
                                        }
                                        if(fmontemp[i][j]<maxmontemp[i][k+1])
                                        {
                                                fmontemp[i][j]=maxmontemp[i][k+1];
                                        }
                                        j+=1;
                                }
                        }
                        break;
                        }
}
void get_max_temp_alb()
{
                        short i,j,k,x,y;
                        float maxtemp,maxalb;
                        for(k=2;k<3;k++)
                        {
                                maxtemp=ptemp[1][1][k];
                                maxalb=palb[1][1][k];
                                for(i=1;i<lin-1;i++)
                                        for(j=1;j<col-1;j++)
                                        {
                                                if((short)flist[i][j][k]!=-32)
                                                {
                                                        if(ptemp[i][j][k]>maxtemp)
                                                        {
                                                                maxtemp=ptemp[i][j][k];
                                                                x=i;
                                                                y=j;
                                                        }
                                                }
                                                if((short)flist[i][j][k]!=-32)
                                                {
                                                        if(palb[i][j][k]>maxalb)
                                                                maxalb=palb[i][j][k];

                                                }
                                        }
                                        mtemp[k]=maxtemp;
                                        malb[k]=maxalb;
                                        printf("%f %d %d\n",maxtemp,x,y);
                        }
}
float get_max_value(float arval[], short num)
{
```

```
	float max;
	short i;
	max=arval[0];
	for(i=1;i<num;i++)
		if(max<arval[i])
			max=arval[i];
		return(max);
}
void calc_evapo()
{
	short i,j;
	float max1,max2,s,rtemp,ralb;
	for(i=0;i<strint;i++)
	{
		max1=maxtemp[1][i];
		max2=maxalb[1][i];
		for(j=2;j<13;j++)
		if(j!=11)
		{
			if(max1<maxtemp[j][i])
				max1=maxtemp[j][i];
			if(max2<maxalb[j][i])
				max2=maxalb[j][i];
		}
		maxfalb[i]=max2;
		maxftemp[i]=max1;
	}
	for(i=0;i<strint;i++)
		for(j=1;j<13;j++)
			if(j!=11)
			{
				rtemp=avertemp[j][i]/maxftemp[i];
				ralb=averef[j][i]/maxfalb[i];
				s=sqrt(ralb);
				evapo[j][i]=(459.63-656.03*rtemp*s*(1-av[j][i]))/28.35;
			}


}
float get_min_value(float marval[], short no)
{
	float min;
	short i;
	min=marval[0];
	for(i=1;i<no;i++)
		if(min>marval[i])
			min=marval[i];
		return(min);
}
void four_for_arctndvi(float four_arr[][40], short flg)
{
	float ang;
	short x,y,z,freq,i,l;
	switch(flg)
	{
	case 1:
	for(i=1;i<13;i++)
		{
			l=0;
			for(x=0,y=0,z=1;x<2*32,y<32,z<2*32;x+=2,y++,z+=2)
				{
					tdata1[x]=four_arr[i][y];
					tdata1[z]=0;
				}
				four1(tdata1-1,32,1);
				for(x=0,freq=1;x<2*32,freq<2*32;x+=2,freq+=2)
				{
					iarct[i][l]=tdata1[freq];
					rarct[i][l]=tdata1[x];
					arctamp[i][l]=sqrt(pow(tdata1[x],2)+pow(tdata1[freq],2));
					ang=-tdata1[freq]/tdata1[x];
					arctphase[i][l]=atan((double)ang);
					l+=1;
				}
		}
	break;
	case 2:
	for(i=1;i<13;i++)
		{
			l=0;
			for(x=0,y=0,z=1;x<2*32,y<32,z<2*32;x+=2,y++,z+=2)
				{
					tdata1[x]=four_arr[i][y];
					tdata1[z]=0;
				}
				four1(tdata1-1,32,1);
				for(x=0,freq=1;x<2*32,freq<2*32;x+=2,freq+=2)
				{
					fiarct[i][l]=tdata1[freq];
					frarct[i][l]=tdata1[x];
					farctamp[i][l]=sqrt(pow(tdata1[x],2)+pow(tdata1[freq],2));
					ang=-tdata1[freq]/tdata1[x];
					farctphase[i][l]=atan((double)ang);
					l+=1;
				}
		}
```

```
                break;
                }
}
void calc_arct()
{
                short i,j;
                for(i=1;i<13;i++)
                                if(i!=11)
                                                for(j=0;j<32;j++)
                                                {
                                                                arctndvi[i][j]=atan(fmontemp[i][j]/fmondvi[i][j])*180*7/22;
                                                                farctndvi[i][j]=atan(famontemp[i][j]/fmondvi[i][j])*180*7/22;
                                                }
                                                arctndvi[1][23]=89;
                                                farctndvi[1][23]=89;
}

void time_series()
{
                FILE *t_series;
                short i,j;
                short x;
                if( (t_series = fopen("d:\\msc\\ndvitser.mp#", "wb" )) == NULL )
   {
                                printf("cannot open the file %s\n");
                exit(1);
                }
   for(i=0;i<strint;i++)
                   for(j=0;j<col;j++)
                                {
      if(cover[57][j]==0)
                                                x=-32767;
                                 else
                                 {
                                                switch(cover[57][j])
                                                {
                                                case 1:
                                                                x=(short)(finalf[1][i]/0.001);
                                                                break;
                                                case 2:
                                                                x=(short)(finalf[2][i]/0.001);
                                                                break;
                                                case 3:
                                                                x=(short)(finalf[3][i]/0.001);
                                                                break;
                                                case 4:
                                                                x=(short)(finalf[4][i]/0.001);
                                                                break;
                                                case 5:
                                                                x=(short)(finalf[5][i]/0.001);
                                                                break;
                                                case 6:
                                                                x=(short)(finalf[6][i]/0.001);
                                                                break;
                                                case 7:
                                                                x=(short)(finalf[7][i]/0.001);
                                                                break;
                                                case 8:
                                                                x=(short)(finalf[8][i]/0.001);
                                                                break;
                                                case 9:
                                                                x=(short)(finalf[9][i]/0.001);
                                                                break;
                                                case 10:
                                                                x=(short)(finalf[10][i]/0.001);
                                                                break;
                                                case 11:
                                                                x=(short)(finalf[11][i]/0.001);
                                                                break;
                                                case 12:
                                                                x=(short)(finalf[12][i]/0.001);
                                                                break;
                                                }
                                 }
                                fwrite(&x,sizeof(x),1,t_series);
                                }
}
void evapo_map()
{
FILE *t_series;
                short i,j;
                short x;
                if( (t_series = fopen("d:\\msc\\evapo.mp#", "wb" )) == NULL )
   {
                                printf("cannot open the file %s\n");
                exit(1);
                }
   for(i=0;i<lin;i++)
                   for(j=0;j<col;j++)
                                {
      if(cover[i][j]==0)
                                                x=-32767;
                                 else
                                 {
                                                switch(cover[i][j])
                                                {
```

```
                    case 1:
                            x=(short)(3.8/0.001);
                            break;
                    case 2:
                            x=(short)(2.2/.001);
                            break;
                    case 3:
                            x=(short)(3.7/0.001);
                            break;
                    case 4:
                            x=(short)(4.2/0.001);
                            break;
                    case 5:
                            x=(short)(2.6/0.001);
                            break;
                    case 6:
                            x=(short)(2.5/0.001);
                            break;
                    case 7:
                            x=(short)(2.4/0.001);
                            break;
                    case 8:
                            x=(short)(2.5/0.001);
                            break;
                    case 9:
                            x=(short)(1.7/0.001);
                            break;
                    case 10:
                            x=(short)(3.8/0.001);
                            break;
                    case 11:
                            x=(short)(4.6/0.001);
                            break;
                    case 12:
                            x=(short)(1.8/0.001);
                            break;
                    }
            }
            fwrite(&x,sizeof(x),1,t_series);
            }
}
```

References

Bakker, W. M., Parodi, G. N., and Timmermans, W. J., (1997). NOAA AVHRR Preprocessing: An Application of a Cloud-Detection Technique. *ITC Journal 1997-1,pp 64-73.*

Bartholome E., (1991). Remote Sensing and Agricultural Production Monitoring in Sahelian Countries. In: Belward and Valenzuela eds. Remote Sensing Applications and GIS for Resources Management in Developing Countries. *(Kluwer Acad. Publ. Dordrecht) Eurocourses, 189-214.*

Bastiaanssen, W. G. M. (1995). Regionalization of surface flux densities and moisture indicators in composite terrain, A remote sensing approach under clear skies in Mediterranean climates. *PhD Thesis, Agricultural University of Wageningen, The Netherlands, pp. 273.*

Bastiaanssen, W. G. M., and Roebeling, R. A. (1993). Analysis of Land Surface Processes in two Agricultural Regions in Spain using Thematic Mapper Simulator data. In: Bolle, Feddes and Kalma (Ed.), Exchange processes at the Land Surface for a Range of space and Time scales. *JAHS Publ., 212, 407-416.*

Caselles, C. C., Sobrino J. A., and Valor, E. (1994). On the Atmospheric Dependence of the Split Window Equation for Land Surface Temperature. *International Journal of Remote Sensing, 15: 1, 105-122.*

Cihlar J., St. Laurent L., and Dyer J. A. (1991). Relation between the Normalized Difference Vegetation Index and Ecological variables. *Remote Sensing for Environment , 35: 279-289.*

Dingman, S. L. (1994). Physical Hydrology. *Printice-Hall, Inc., Simon and Schuster / A Viacom Company, New Jersey, pp. 575.*

Eidenshink, J. C., and Faundeen, J. L. (1998). The 1-KM AVHRR Global Land Data Set: First Stages in Implementation. Internet web site "http://edcwww.cr.usgs.gov/ Landdaac/1km"

Grayson R.B., Blosch, G., Barling and Moore, I. D. (1993). Process, Scale and Constrains to Hydrologic Modeling in GIS. HydroGIS. Application of Hydrology and Water Resources Management. Kovar Ed. K. and Nachthbel, H. P. *IAHS publ. No.211 pp 83-92.*

Gutman, G. G., (1991). Vegetation Indices from AVHRR: An Update and Future Prospects. *Remote Sensing of Environment. 35:121-136*

Hamududu, B. H., (1998). Erosion Assessment for Large Basins using Remote Sensing and GIS: a Case Study of Lake Naivasha Basin, Kenya. *ITC's MSc. Thesis.*

Holben, B. N., Kaufman, Y. J., and Kendall, J. D., (1990). NOAA-11 AVHRR visible and near infrared Inflight Calibration*: The International Journal Of Remote Sensing, 11, 1511-1519.*

Holben,B. N. (1986). Characteristics of Maximum Value Composite Images From Temporal AVHRR DATA. *International Journal Of Remote Sensing, 7: NO. 11, 1417-1434.*

Iqbal, M. (1983). An Introduction to Solar Radiation. *Academic Press, a Subsidiary of Harcourt Brace Javanovich, Publishers.*

Kerr. Y. H., Imbernon, J., Dedieu.G, Hautecoeur, O., Lagouarde, J, and Seguin,B. (1989). NOAA-AVHRR and its Uses for Rainfall and Evepotranspiration Monitoring. *International Journal of Remote Sensing. 10:847-854.*

Kidwell, K. B., (1991). NOAA Polar Orbiter Data Users' Guide, National Oceanic and Atmospheric Adminstration, World Weather Building, Room 100, Washington D.C.

Kilham, p. (1971). Biochemistry of African Lakes and Rivers, *PhD thesis, Duke University, Durham. NC. pp 199.*

KSS (1980). Soil condition at Kulia farm (Naivasha). KSS (Kenya Soil Survey) report

Lambin E. F., and Ehrlich D. (1996). The Surface Temperature-Vegetation Index Space for Land Cover and Land Cover Change Analysis. *International Journal of Remote Sensing, 17: no. 3,463-487.*

Mavuti, K. M., (1983). Wetlands and Shallow Water Bodies in Kenya: Characteristics and General Status of Knowledge, *Promotion of Science and Tech. Kenya, 5.pp 48-58.*

Meijerink A. M. J., De Brower H. A. M., Mannerts C. M., and Valenzuela. C. R. (1994). Introduction to the use of Geographic Information Systems for practical Hydrology. *ITC, Publication Number 23.pp 243.*

Monteith, J.L. (1972). Vegetation and the atmosphere. *Vol. 1Principles. Vol. 2, Case studies. London, Acad.press.279 and 439 pp.*

NOAA Technical Report NESDIS 49, (1990). Implementation of Reflectance Models in Operational AVHRR Radiation Budget Processing.

Parazen, E., (1967). Time series Analysis Papers. *Holden-Day Inc., 500 Samsome St., San Francisco.*

Price, J. C., (1984). Land Surface Temperature Measurements from the Split Window Channels of the NOAA-7 Advanced very High resolution Radiometers, *J. Geophys. Res.89: 17-22.*

Price, J. C., (1987). Calibration of Satellite Radiometers and The Comparison of Vegetation Indices. *Remote Sensing of the Environment, 21, 15-27.*

Rao, N. C. R. (1987). Pre-Launch Calibration of channels 1 and 2 of Advanced Very High Resolution Radiometer: *NOAA Technical Report NESDIS 36, Satellite Research Laboratory, Natonal Environmental Satellite, Data, and Information Service, Wash*ington, *D.C.*

Richardson J. and Richardson A., (1972). History of an African lake and its Climate Implications. *Ecol.Monogr. 42: 499-534.*

Saunders R. W. (1990). The Determination of Broadband Surface Albedo from AVHRR Visible and Near-Infrared Radiances. *International Journal Of Remote Sensing, 11, No. 1, 49-67.*

Saunders, R. W. and Kriebel, K. T. (1988). An Improved Method for Detecting Clear Sky and Cloudy Radiances from NOAA-AVHRR Data. *International Journal of Remote sensing 9: 1,123-150.*

Schott, J. R., Salvaggio C., and William J. V. (1988). Radiometric Scene Normalization Using Pseudoinvariant Features. *Remote Sensing of Environment 26:1-16.*

Seguin, B., Assad,E., Freteaud,J. P. Imbernon, J., Kerr,Y., and Lagouarde,J.P. (1989).Use of Meteorological Satellites for Water Balance Monitoring in Sahelian regions. *International Journal of Remote Sensing.10:1101-1117.*

Smith, M. O., Ustin, S. L., Adams, T. B., and Gillespie, A. R. (1990). Vegetation in Deserts: II. Environmental influences on Regional abundance. *Remote Sensing of Environment. 31:27-52.*

Teillet, P. M., (1990). Rayleigh optical depth comparisons from various sources: *Applied Optics, 28, 1897-1900.*

Teillet, P. M., (1991). Radiometric and atmospheric correction procedures for AVHRR preprocessing in the solar reflective channels. *Proceedings of the Fifth International Colloquium on Spectral Signatures of Objects in Remote Sensing, Courchevel, France, 1991, 511-516.*

Teillet, P. M., and Holben, B. N., (1994). Towards Operational Radiometric Calibration of NOAA Imagery in the Visible and Near-Infrared Channels: *Canadian Journal of Remote Sensing, 20, 1.*

Tukey, J. W., 1977. Exploratory Dataanalysis. *Addison-Wesley publishing company, Reading, Massachusettes.*

Van de Griend A. A., and Owe M. (1993). On the Relationship between Thermal Emissivity and the Normalized Difference Vegetation Index for Natural Syrfaces. *International Journal of Remote Sensing, 14, no. 6, 1119-1131.*

Van Dijk, A., Susan L. C., Clarence M. S. and Vayne L. D., (1987). Smoothing Vegetation Index Profiles: An Alternative Method for Reducing Radiometric Disturbance in NOAA AVHRR Data. *Photogrametric Engineering and Remote Sensing, 53:8, 1059-1067.*

Velleman, P., and Hoaglin, D. C., (1981). Applications, Basics and Computing of Exploratory Data Analysis. *Duxbury Press, Boston, Mass.*