Winter 2001

Multi -mission Attitude Determination System for balloon flight

Liping Mo
Louisiana Tech University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

MULTI-MISSION ATTITUDE DETERMINATION SYSTEM

FOR BALLOON FLIGHT

by

Liping Mo, M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Computational and Analysis Modeling

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

February 2001

UMI Number: 3000446

UMI Microform 3000446

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

2-9-01
Date

We hereby recommend that the dissertation prepared under our supervision by Liping Mo

entitled Multi-mission Attitude Determination System for Balloon Flight

be accepted in partial fulfillment of the requirements for the Degree of PhD in Computational Analysis and Modeling

Natalia V. Zotov
Supervisor of Dissertation Research

Raja Nassar
Head of Department

CAM
Department

Recommendation concurred in:

Advisory Committee

Approved:
Director of Graduate Studies

Approved:
Director of the Graduate School

Dean of the College

GS Form 13
2/97

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author Liping Liu

Date 2-9-2001

GS Form 14
2/97

ABSTRACT

MADS (Multi-mission Attitude Determination System) is a new software package used to determine the attitude of instruments on a high-altitude balloon employed for scientific experiments. There is no existing system for the automated determination of the attitude of instruments in balloon experiments, so we have developed MADS to do the data analysis for balloon experiments to find the location of astrophysical sources such as gamma-ray or x-ray sources.

The two areas that required most work were modeling star trackers and modeling the motion of the balloon. Star trackers are used on satellites, but are far too expensive and sophisticated to use on balloons. Their processes have to be modeled using only the data from simple CCD cameras. The motion of a three-axis-stabilized satellite moving in a prescribed orbit is very much simpler than the motion of a balloon, which is carried by stratospheric winds and always retains to some degree its initial spinning and pendular motions.

Another area that had to be addressed was interpolation when the balloon is out of range of a sufficient number of GPS satellites to determine it position (This may happen in the Arctic or Antarctic).

The software package, with documentation written to NASA standards, is being made available to NASA at their request.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

It would be impossible to acknowledge adequately all the people who have been helpful and supportive in forming this dissertation. My dissertation committee particularly deserves to be acknowledged. Dr. Natalia Zotov, the committee chair, is the first person who accepted me in a research project that led to my dissertation. Since then I have worked with her for several years. I have experienced many challenges in completion of my dissertation. Her encouragement was so important to me, particularly when I felt frustrated by my research project. She has always been available whenever I needed her help and guidance. I owe special thanks to her, and I have also very much appreciated her friendship.

I also want to acknowledge special debts to my other committee members: Dr.Weizhong Dai, Dr. Ben Choi, and Dr. Raj Nassar were also my committee for the thesis for the master's degree. Dr. Nassar is one of the most helpful persons I have ever met. He has given me much help in my work both in class and in office hours. I have greatly benefited from him in statistics and SAS. Dr. Dai is the first helping me enter Dr. Zotov's research team and the ACAM program. He has been completely helpful and supportive in my dissertation writing. Dr. Ben Choi has help me in research project and in class .

I would like to extend a deep sense of gratitude and thanks to Dr. Richard J. Greechie. He interviewed me on the telephone when I was in Albany, New York, and

made it possible for me to become a graduate student here at Louisiana Tech. He always stimulated and inspires me during my graduate studies career here. I would also like to thank Dr. Johnston, who financially supported me for several quarters. I would also like to thank Mr. Rick Harman who is in Guidance, Navigation, and Control center Flight Dynamics Analysis Branch at NASA. He has given me much help in my research.

I want to thank my parents, my brother, and parents-in-law for their love, understanding, and support. My parents came to the USA twice to take care of my two sons. Their love and support are invaluable. No words can express my thanks to them.

Finally, I thank my two lovely sons and my husband. My older son, Yang, always wants me spend more time with him and his brother. But he understands and supports my study. My younger son, Starr, was born during my graduate studies career at Louisiana Tech. He seems to understand and support me since he came to this beautiful world. I feel sorry for not spending more time with them. I am also very grateful for their love and support. My husband, Shanhe, has given me deep love, understanding, encouragement, and support. Without his special efforts I could not have finished my doctoral study. In the past several years he has spent much time taking care of our two sons and cooking, even though he has been busy in teaching, research, and other university duties. This dissertation is dedicated to Shanhe, Yang, and Starr.

CHAPTER 1

INTRODUCTION

The Multi-mission Attitude Determination System (MADS) is a software package that analyzes data from balloon experiments to find the attitude of a telescope on a balloon. All instruments intended for use on astrophysical satellites must first be tested on high-altitude balloons. Until now, no automated system for determining the attitude of a detector on a balloon has been available. This dissertation presents a system for determining the pointing and rotation of a telescope at any time during a balloon flight. This attitude determination system has been made available to NASA and will be used by NASA and university balloon groups.

1.1 BALLOON EXPERIMENTS

Many important observations in fields such as hard X-ray/gamma-ray and infra red astronomy, cosmic rays, and atmospheric studies have been made from balloons because balloons offer a low-cost, quick-response method for doing scientific investigations. Balloons have important advantages over satellites. They can be launched where a scientist needs to conduct the experiment. Balloon can be readied for flight in as little as six months. Balloon payloads provide information on the atmosphere, the universe, the Sun, and the near-Earth and space environments. A balloon flight mission is relatively simple. The balloon is partially filled with helium and launched with the

payload suspended beneath it. As the balloon rises, the helium expands and fills out the balloon until it reaches its peak altitude two to three hours after launch. After the scientist has concluded the experiment, a radio command is sent from a ground station to separate the payload from the balloon. The parachute opens and floats the payload back to the ground so it can be reused. The payload reaches the ground about 45 minutes after it has been separated from the balloon. Payload separation creates a tear in the balloon, which falls to the ground, where it is retrieved and discarded.

Ballooning gives scientists an inexpensive way of getting telescopes above most Earth's atmosphere. From altitudes of 38km altitude (126,000 feet or 23.8 miles), where the remaining atmospheric pressure is only 3 millibar (about 1/3 of 1 percent of sea level pressure), instruments can see the universe almost as clearly as the Hubble Space Telescope and other orbiting observations. And while the ride usually lasts only 30 or 40 hours, it is far cheaper than a space launch (or satellite), so many research projects used balloon instruments to achieve their goals.

The newly developed capability for long duration ballooning has greatly expanded the opportunities for scientific studies from balloons. The long duration balloon project is to develop balloon systems capable of supporting scientific observation above 99% of the Earth's atmosphere for a duration approaching 100 days. The design goal is to support to a scientific payload of 2200 pounds and to be able to deliver 800 watts of continuous power to the scientific instrument. Using this kind of balloon, a scientist can command his instrument and receive scientific data at his home institution via the internet. This long duration balloon will have global flight capability. Using balloons is a major technical challenge in the research of the atmospheric and astrophysics.

1.2 DATA ANALYSIS IN BALLOON EXPERIMENTS

As we have seen, the balloon has a great advantage for rapid, low-cost access to space. Balloons have been used for research on cosmic ray studies, gamma-ray and X-ray astronomy, optical and ultraviolet astronomy, infrared astronomy, atmospheric sciences, magnetospherics, and micrometeorite particles. For these kinds of research, the balloon has a telescope or other primary detector to detect the sources.

The purpose of a telescope, regardless of type, is to gather light in some form that allows one to construct a picture of the sky. For the part of the electromagnetic spectrum ranging from radio through ultraviolet, telescopes operate in roughly the same manner using the basic optical principles of reflection or reflection. However, for photon energies higher than ultraviolet, light begins to interact with matter in different ways, and in general, phenomena such as reflection no longer exist. The telescope will be a different type detector. Telescopes which observe high-energy photons are therefore based on rather different principles. Low-energy X-rays (0.1-5keV) can be made to reflect from certain metals provided they are incident at very shallow angles. In this energy range, one can construct a grazing incidence telescope, which uses a series of nested hyperbolic mirrors to focus low energy X-ray onto an X-ray detector. Hard-energy X-rays (5-100keV) do not reflect at all, so the hard X-ray telescope uses a coded-aperture mask where a special pattern of holes is machined into some dense material. In the gamma-ray region (above a few hundred keV), photons have so much energy that they begin to penetrate even the densest materials, generally creating secondary background radiation in the process. The Compton telescope makes use of this interaction to observe photons in this energy range.

Determining the pointing and rotation angle of a telescope at any given time turns out to be the one of most important projects in a balloon experiment. How to analyze the data from the secondary instruments carried on a balloon to determinate the pointing and rotation angle of telescope is itself a research area in balloon experiments. Based on the pointing and rotation angle of telescope, a scientist can find the position of, for example, X-ray and gamma-ray sources in the sky. So the main goal of data analysis for balloon experiment is to find the pointing and rotation angle of telescope, which is the called attitude of telescope.

There is no existing system for the automated determination of attitude in balloon experiments; however, we have developed an attitude determination system suitable for a variety of balloon missions. This dissertation will describe this attitude determination system.

1.3 MADS SUPPORT OF DATA ANALYSIS FOR BALLOON EXPERIMENTS

The Multi-mission Attitude Determination System (MADS) was originally designed to do the data analysis to locate gamma-rays sources in the sky. It determines where the X- ray or gamma-ray source detector is pointing in the sky and the detector angle of rotation with respect to the pointing vector.

MADS accepts information from secondary instruments to determine the attitude of the primary balloon detector. These data-- such as (1) Global Positioning System (GPS) data, which will be discussed in chapter 4; (2) Charged Coupled Device (CCD) cameras, which will be discussed in chapter 5; and (3) gyroscope data, which will be

discusses in chapter 3 -- are basic information for the attitude determination system. These data are obtained from the balloon experiment to determine the attitude of the balloon's primary detector from MADS. MADS will use these data to calculate the pointing and rotation angle of the telescope as an attitude quaternion and represent it as a vector with a timetag at each time step during the balloon flight. The quaternion will be discussed in chapter 6.

MADS has developed a determination system for balloon experiments based on the Multimission Three-Axis Stabilized Spacecraft (MTASS) package, which provides the star identification procedure. MTASS is a software system that supports attitude determination and analysis for satellites, but not for balloons. A balloon flight is different in many ways from a satellite's. A satellite moves with a predictable direction, but balloon fly in random directions. A balloon also carries different instruments from a satellite's thereofre, MADS operates differently to determine the attitude of balloon instruments. It provides a full system to determine the attitude of a balloon instrument with user-provided data such as GPS data, CCD camera data, gyro data, and a ground-level initial quaternion, which will be discussed in chapter 3. MADS will pre-process the CCD camera data and GPS data first, then use these processed data and a ground-level initial quaternion to calculate the later attitude quaternions.

MADS is a very useful software package for balloon experiments. It can do the data analysis and find the attitude of balloon instruments as a function of time, which means that it provides the vector of telescope pointing and rotation angle at each time step during the balloon flight time. It allows a scientist to analyze the pointing and the rotation angle of telescope not only at end of a balloon experiment but at any

intermediate time. MADS will be used in NASA's scientific balloon program and by other university balloon groups for their experiments.

CHAPTER 2

SATELLITE ATTITUDE AND MTASS

The Multimission Three-Axis Stabilized Spacecraft (MTASS) package is a software system developed by Flight Dynamics Support System of NASA. Its original function was to support attitude determination and analysis for satellites.

2.1 OVERVIEW OF MTASS

The MTASS software system grew out of the earlier satellite missions such as the Hubble Telescope, the Solar Maximum Mission, and the Gamma Ray Observatory. The motivation was to reduce errors and development, testing, and maintenance costs by having the functions common to many users contained within a single system.

MTASS system generally cannot satisfy all the software requirements for supported mission. Besides orbit determination and control, MTASS does not provide for other mission–specific functions such as telemetry processing or science and mission planning and generation. It concentrates on features generic to attitude support. Specifically, it provides the following:

- An attitude determination system that adjusts telemetry, identifies, and computes attitude using either differential corrector or quaternion estimation methods. These functions are often used with a mission–specific telemetry processor to build a real-time attitude determination system.

- Validation of attitude and ephemerides computed onboard the spacecraft
- On-orbit calibration of spacecraft attitude sensors and gimballed platforms.
- Prediction of attitude, antenna contacts, and guide star occultations.

MTASS tends to be generic, but the MTASS system does contain some mission-specific elements. Missions supported by MTASS have included:

- Upper Atmosphere Research Satellite (UARS)

The Upper Atmosphere Research Satellite is the first major flight element of NASA's mission to Planet Earth. It is designed to help scientists learn more about the fragile mixture of gases protecting Earth from the harsh environment of space.

- Extreme Ultraviolet Explorer (EUVE)
- Solar, Anomalous, and Magnetospheric Particle Explorer
- Solar and Heliospheric Observatory (SOHO)

Solar and Heliospheric Observatory SOHO is designed to study the internal structure of the Sun. It will help scientist to understand the interactions between the Sun and the Earth's environment.

- Tropical Rainfall Measuring Mission (TRMM)
- Rossi X-Ray Timing Explorer (RXTE)

The RXTE is maneuverable (6 deg/minute) so that it can be made to point to a chosen source rapidly. This flexibility allows the instrument to respond to short-lived new phenomena as they are discovered. We are actually using the RXTE version of MTASS to as a basis for the MADS package for balloon experiments.

2.2 DETERMINATION OF ATTITUDE FOR A SATELLITE

The attitude of a satellite is usually represented by the pointing and rotation angle of the primary detector, e.g., a telescope, at any moment in time. The determination system of attitude for a satellite includes the following functions: adjust ment of processed data, assembling reference data, determination of coarse attitude, determination of fine modular attitude (which computes the attitude of the operational modular frame using a least squares differential correction algorithm), determination of definitive modular attitude, and determination of a gimbaled platform attitude. The basic ideas of determination are using star tracker data to find the observation vectors of stars with respect to the satellite, then using ephemeris data and gyroscope data to calculate the reference vectors (explained in chapter 6) of stars with respect to Earth center. Comparison of the reference and observation vectors of stars enables the stars to be identified and the attitude of star trackers to be determined. Finally one calculates the attitude of the satellite's primary detector.

1. Representation of Attitude

There are many different ways to represent satellite attitude. In MTASS, attitudes are represented in three ways:

- Attitude quaternions

The attitude quaternion is a four-dimensional vector whose format is expressed as (q_1, q_2, q_3, q_4), defined as

$$q_1 = e_x \sin\frac{\theta}{2} \qquad \text{(Eq. 2.2.1)}$$

$$q_2 = e_y \sin\frac{\theta}{2} \qquad \text{(Eq. 2.2.2)}$$

$$q_3 = e_z \sin\frac{\theta}{2} \qquad \text{(Eq. 2.2.3)}$$

$$q_4 = \cos\frac{\theta}{2} \qquad \text{(Eq. 2.2.4)}$$

where (e_x, e_y, e_z) is a unit vector in the geocentric coordinate system. The geocentric coordinate axes originate in the Earth's center. The +Z-axis points north along the Earth's spin axis, the +X-axis points to the vernal equinox direction in the Earth's equatorial plane, and the +Y-axis completes the orthogonal triad. Because the Earth's spin axis precesses about the ecliptic pole with a period of approximately 26,000 years, the geocentric axes move slowly in inertial space at a rate of approximately 50 arc-sec per year. Therefore, a reference time (epoch) has to be attached to the definition of geocentric coordinate to make them truly fixed inertially.

θ is a rotation angle about the pointing reference vector (e_x, e_y, e_z). Since a quaternion presents the attitude of the balloon telescope and in order to understand the quaternion, note Figure 2.2.1:

Let X, Y, Z, be the axes in geocentric coordinate system, (e_x, e_y, e_z) be a unit attitude vector, and (q_1, q_2, q_3, q_4) an attitude quaternion.

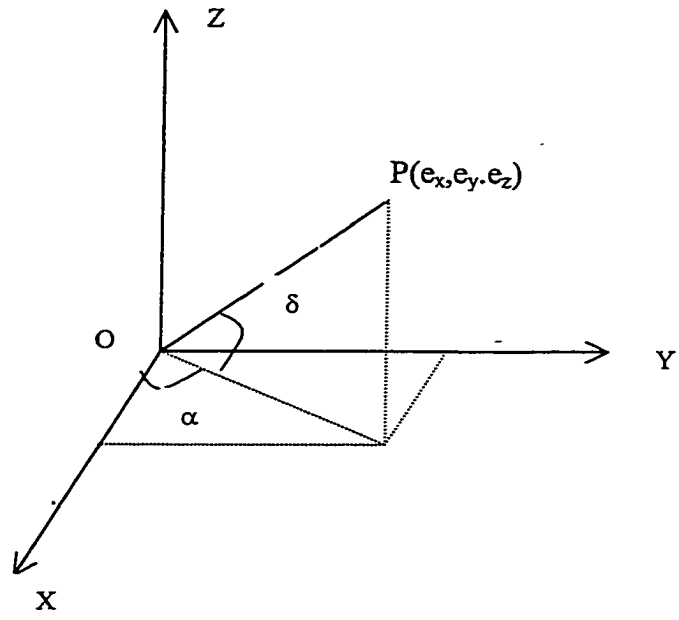

Figure 2.2.1 Attitude quaternion in geocentric coordinate

α is right ascension, and $0 \leq \alpha < 2\pi$,

δ is declination, and $-\pi/2 \leq \delta \leq \pi/2$,

θ is the angle of rotation about OP and $0 \leq \theta < 2\pi$.

Since

$$e_x = \cos\delta\cos\alpha \qquad \text{(Eq. 2.2.5)}$$

$$e_y = \cos\delta\sin\alpha \qquad \text{(Eq. 2.2.6)}$$

$$e_z = \sin\delta \qquad \text{(Eq. 2.2.7)}$$

from Eq. 2.2.1 and Eq. 2.2.2,

$$\frac{q_2}{q_1} = \frac{e_{yx}\sin\frac{\theta}{2}}{e_x\sin\frac{\theta}{2}} = \frac{e_y}{e_x}. \qquad \text{(Eq. 2.2.8)}$$

From Eq. 2.2.5.and Eq. 2.2.6

$$\frac{e_y}{e_x} = \frac{\cos\delta c\sin\alpha}{\cos\delta c\cos\alpha} = \tan\alpha \qquad \text{(Eq.2.2.9)}$$

Using Eq.2.2.8 and Eq. 2.2.9, we get $\tan\alpha = \frac{q_2}{q_1}$, so $\alpha = \tan^{-1}\frac{q_2}{q_1}$, where

the quadrant is determined by the signs of q_1 and q_2.

To calculate δ, use Eq. 2.2.3, and Eq. 2.2.7, we have

$$\sin\delta = e_z = \frac{q_3}{\sin\frac{\theta}{2}} \qquad \text{(Eq. 2.2.10)}$$

Also since

$$\cos\delta = \sqrt{e_x^{\,2} + e_y^{\,2}} \qquad \text{(Eq. 2.2.11)}$$

From Eq. 2.2.1 and Eq. 2.2.2,

$$q^{\,2} + q_2^{\,2} = (e_x^{\,2} + e_y^{\,2})\sin^2\frac{\theta}{2} \qquad \text{(Eq. 2.2.12)}$$

Substituting Eq. 2.2.12 into Eq. 2.2.11,

$$\cos\delta = \frac{\sqrt{q_1^{\,2} + q_2^{\,2}}}{\sin\frac{\theta}{2}} \qquad \text{(Eq. 2.213)}$$

Then from Eq. 2.2 11 and Eq. 2.2.13, we get:

$$\tan\delta = \frac{q_3}{\sqrt{q_1^{\,2} + q_2^{\,2}}}, \text{ that is}$$

$$\delta = \tan^{-1}\frac{q_3}{\sqrt{q_1^{\,2} + q_2^{\,2}}} \qquad \text{(Eq. 2.2.14)}$$

Where the quadrant is determined by the sign of q_3.

For the rotation angle θ, from Eq. 2.2.4, we get:

$$\theta = 2\cos^{-1} q_4 \qquad \text{(Eq.2.2.15)}$$

Since the attitude can be represented by a quaternion, and α, δ, θ can be represented by quaternion, the attitude can be represented as a vector which incorporates pointing and rotation using geocetric coordinates.

The attitude also can be represented in two other ways:

- Euler angle rotations with respect to an attitude reference frame, which may be either the geocentric reference frame or an orbital coordinate system.

The Euler angle are represented as (y, r, p), where y is a yaw angle, i.e the rotation about the Z- axis, r is the roll angle, i.e the rotation about the X-axis, and p is pitch angle, i.e rotation about the Y-axis.

- Attitude matrix

The attitude matrix is a 3 × 3 matrix. Its elements are related to the y, r, p and also can represented by the quaternion.

2. Relation between the three types of attitudes

The attitude matrix is related to the y, r, p as the following:

$$
\begin{aligned}
A_{11} &= \cos p \cos y - \sin r \sin p \sin y \\
A_{12} &= \cos p \sin y + \sin r \sin p \cos y \\
A_{13} &= -\sin p \cos y \\
A_{21} &= -\cos r \sin y \\
A_{22} &= \cos r \cos y \\
A_{23} &= \sin r \cdot \\
A_{31} &= \sin p \cos y + \cos p \sin r \sin y \\
A_{32} &= \sin p \sin y - \cos p \sin r \cos y \\
A_{33} &= \cos r \cos p
\end{aligned}
\qquad \text{(Eq. 2.2.16)}
$$

The Euler angles also can be represented by elements of the attitude matrix as following:

$$
\begin{aligned}
y &= \arctan\left[\frac{-A_{21}}{A_{22}}\right] \qquad 0 \le y \le 2\pi \\
\gamma &= \arcsin[A_{23}] \qquad -\pi/2 \le \gamma \le \pi/2 \\
\rho &= \arctan\left[\frac{-A_{13}}{A_{33}}\right] \qquad 0 \le \rho \le 2\pi
\end{aligned}
\qquad \text{(Eq. 2.2.17)}
$$

The attitude can also be expressed in quaternion form. There are four possible ways to compute the quaternion from the attitude matrix (Shuster 1993). Here the quaternion are designed the denominators are not equal to zero. One of four way is presented as:

$$\left.\begin{aligned} q_1 &= \frac{1}{4q_4}(A_{23} - A_{32}) \\ q_2 &= \frac{1}{4q_4}(A_{31} - A_{13}) \\ q_3 &= \frac{1}{4q_4}(A_{12} - A_{21}) \\ q_4 &= \pm\frac{1}{2}(1 + A_{11} + A_{22} + A_{33}) \end{aligned}\right\} \qquad \text{(Eq.2.2.18)}$$

The relation between the quaternion components and attitude matrix can be presented as below:

$$\left.\begin{aligned} A_{11}(q) &= q_1^{\,2} - q_2^{\,2} - q_3^{\,2} + q_4^{\,2} \\ A_{12}(q) &= 2(q_1q_2 + q_3q_4) \\ A_{13}(q) &= 2(q_1q_3 - q_2q_4) \\ A_{21}(q) &= 2(q_1q_2 - q_3q_4) \\ A_{22}(q) &= -q_1^{\,2} + q_2^{\,2} - q_3^{\,2} + q_4^{\,2} \\ A_{23}(q) &= 2(q_2q_3 + q_1q_4) \\ A_{31}(q) &= 2(q_1q_3 + q_2q_4) \\ A_{32}(q) &= 2(q_2q_3 - q_1q_4) \\ A_{22}(q) &= -q_1^{\,2} - q_2^{\,2} + q_3^{\,2} + q_4^{\,2} \end{aligned}\right\} \qquad \text{(Eq. 2.2.19)}$$

2.3 THE BASIC DATA IN MTASS

The RXTE version of MTASS uses spacecraft ephemeris data, fixed-head star tracker data, inertial reference unit or gyroscope data, and an attitude history file for the attitude determination system.

The spacecraft ephemeris gives the position of spacecraft at time t_s. An ephemeris is in principle a table whose entries are function of time. The entries may be calculated from formulas as need. For time t_s, MTASS first generates references Earth-to-spacecraft vector, and a spacecraft velocity vector, both in geocentric coordinates, by either reading the spacecraft ephemeris file or by using an analytic orbit generator. Then MTASS uses the spacecraft position vector and velocity vector for processes such as star identification.

The fixed head star tracker data are for attitude determination and control. In general, a fixed head star tracker measures star position vectors relative to itself. Typically, its output consists of three parameters: H, the horizontal coordinate of the star in the filed of view; V, the vertical coordinate of the star in the filed of view; and I, the star-generated signal intensity. In principle I can be used to determine the star magnitude, but since magnitude measurement error typically is quite high, and a majority of stars are variable, it gives only limited help in identifying the star. MTASS uses fixed head star tracker data to calculate the star observation vector and star reference vector in geocentric coordinates. Both vectors then are used in the star identification process and determinationof attitude processes.

The gyro data in MTASS provide angular rate information about the spacecraft. MTASS uses gyro data in star identification and in determining an attitude by propagation of an attitude quaternion.

The attitude history file contains the attitude quaternion from the attitude determination functions described, or from the onboard computer solutions as extracted by the telemetry processor. MTASS allows to use an attitude history file to find the final quaternion for the attitude spacecraft if user choice to use attitude history file in attitude determination system.

Since MTASS is an attitude determination system for multiple missions, it also use other kinds of data such as a three-axis magnetometer, an Earth sensor assembly, and coarse and fine Sun sensor for determination of attitude.

2.4 THE PROCESSES OF MTASS

The basic processes includes telemetry processing, data adjustment, star identification, attitude determination, attitude and ephemeris validation, and sensor calibration. These process will now be described in more detail.

1. Telemetry Processing: This process includes importing the raw sensor, actuator, and control system data from an external file source and placing it in internal array for later processing. The telemetry processor converts files in an arbitrary format to arrays in the MATLAB memory used by MTASS. In RXTE version of MTASS, it loads data from the gyro, two star trackers, and or an on-board computer attitude history file.

2. Data Adjustment: This process includes converting raw sensor and actuator data to observed vectors in the spacecraft body coordinates and attaching reference vectors for each time point, except that the fixed head star trackers are not matched up with reference vector in this process. Data adjusted in this process include fixed head star tracker data and gyroscope data.

The raw fixed head star tracker data, includes information: (1) time, (2) phi, a angle present the horizontal coordinate of star in the field of view, (3) theta, an angle present the vertical coordinate of star in the field of view, (4) temperature, which is measured from the camera, (5) intensity, the star-generated signal intensity, and (6) flag. The data adjustment process calculates the star's observation vector in spacecraft body coordinates and its magnitude.

The raw gyro data include information on time and values of rotation about the x-axis, y-axis, and z-axis of the gyro. The data adjustment process computes a corrected rate vector in spacecraft body coordinate system.

3. Star Identification: This process matches up stars with their reference vectors and is performed by use of a direct match and /or a pattern match method based on an angular separation matching technique. The star identification process uses ephemeris data, fixed head star track data, gyro data, and SKYMAP data. The SKYMAP is a star catalog that provides a star's identification, reference vector, and magnitude. The observed star vectors and candidate star vectors are used in the direct match method and doublet match method for the star identification. The direct match algorithm tests the angles θ and magnitude difference Δm

etween each observed star vector $\hat{W}$ and the reference star vector $\hat{W}^R$ against limits θ_{max} and m_{max}, where

$$\theta = \cos^{-1}(\hat{W} \bullet \hat{W}^R)$$

and $$\Delta m = \left| m - m^R \right|$$

The vectors $\hat{W}$ and $\hat{W}^R$ are in geocentric coordinates. The direct match process first calculates observed star vector $\hat{W}$, then chooses candidate stars from SKYMAP to have their reference vector $\hat{W}^R$. The geometry for direct match is shown in figure 2.4.1 (page 28). For every set of observed and reference star vectors, if $\theta < \theta_{max}$ and $\Delta m < m_{max}$, the observed star is matched with the candidate star. When a star is matched, it has the same information such as star ID and position as candidate star, so the observed star has the same reference vector with respect to the Earth center as candidate star. This method is the basis of star identification.

The doublet match computes the angle θ between each two of observation star vectors and compares it to the angle θ_R between their corresponding candidate stars' reference vectors (found in the direct match); that is, for the match pairs $(\hat{W}_1, \hat{W}^R{}_1)$ and $(\hat{W}_2, \hat{W}^R{}_2)$, the angles θ and θ^R are computed from:

$$\theta = \cos^{-1}(\hat{W}_1 \bullet \hat{W}_2)$$

and

$$\theta^R = \cos^{-1}(\hat{W}_1^R \bullet \hat{W}_2^R)$$

The doublet match assumption is that if $\hat{W}_1$ and $\hat{W}_2$ are properly matched to $\hat{W}^R{}_1$ and $\hat{W}^R{}_2$ respectively, then θ should be close to θ^R. Closeness tested against a limit:

$$\left|\theta - \theta^R\right| < \rho_{\max}$$

All pairs $(\hat{W}, \hat{W}^R)$ which satisfy this condition pass the doublet match. The geometry for the doublet match is in figure 2.4.2.

4. Attitude Determination: The RXTE version of MTASS uses an Extended Kalman Filter to compute to computer the spacecraft attitude. The Extended Kalman Filter solves for the attitude and gyro bias. It uses a gyro, fixed head star tracker observed vector and reference vectors, and attitude history data to calculate the attitude quaternion from the previous time step and propagate it using the latest gyro rates compensated by the latest gyro bias.

5. Attitude and Ephemeris Validation: This process allows the user to compare any pair of attitude history files. Comparson provides a check on the ground-based or on-board computer ephemeris, by comparing their predictions with later actual positions.

The Figure 2.4.3 is a data flow diagram for processes in MTASS.

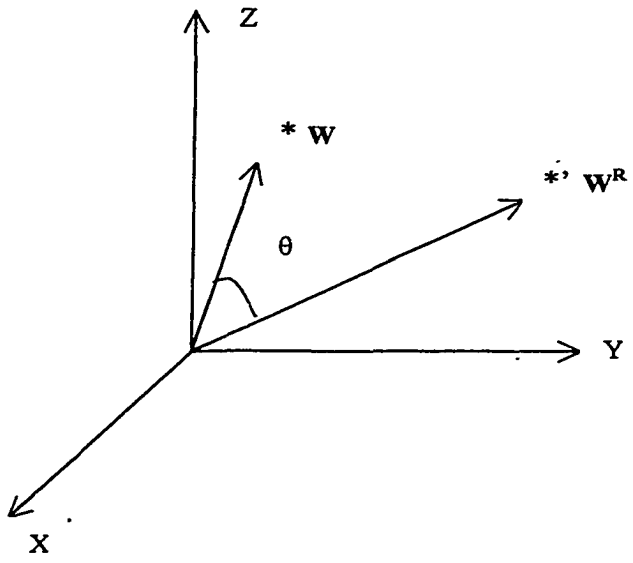

Figure 2.4.1 Geometry for direct match

In Figure 2.4.1, X, Y, Z are axes in geocentric coordinate system, * is a observed star, $\mathbf{W}$ is observed star reference vector. *' is a candidate star for observed star. $\mathbf{W}^R$ is candidate star reference vector.

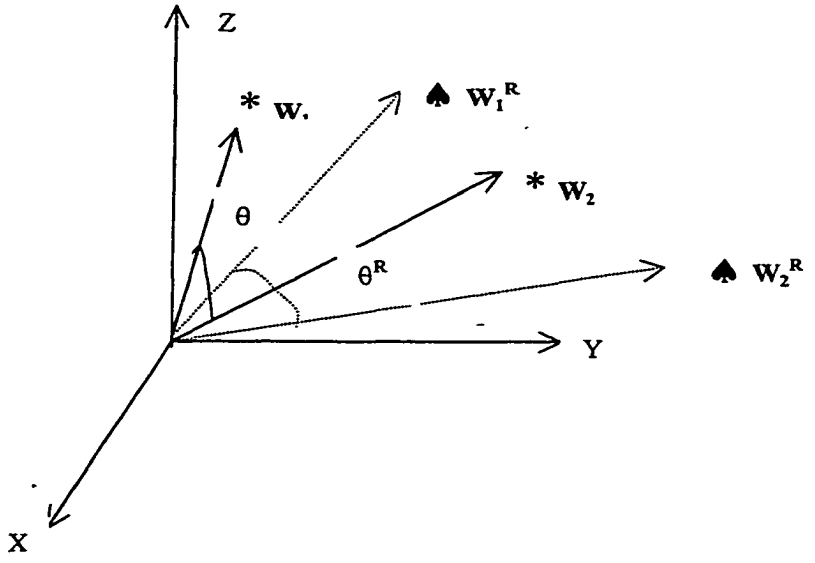

Figure 2.4.2 Geometry for doublet match

In figure 2.4.2, X, Y, Z are axes in geocentric coordinate system, * is a observed star, W_1 , W_2 are observed star reference vectors. ♠ is a candidate star for observed star. W_1^R , W_2^R are candidate star reference vectors.

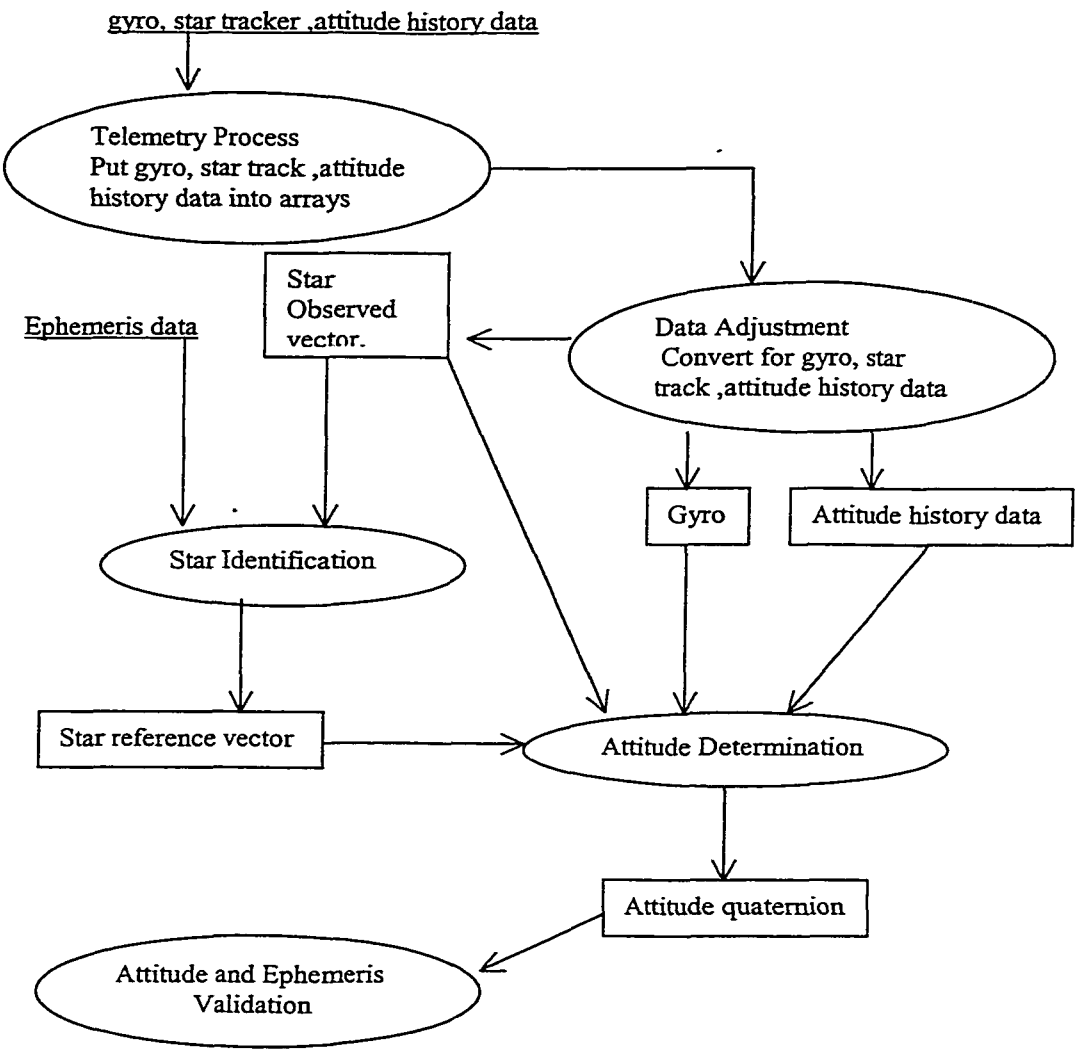

Figure 2.4.3 Data flow diagram for processes in MTASS

Note: ▭ : output data, ——— : input data, ⬭ : process

2.5 THE RESULTS FROM MTASS FOR SATLLITE

The attitude determination system has been described for satellites. In the RXTE version of MTASS, we input the gyro data, fixed head star track data, on-board computer attitude history file, and we use the spacecraft ephemeris that we got from NASA to run the MTASS. We used the Data Adjuster, Star Identification, and Extended Kalman Filter to determinate the attitude quaternions of satellite. The results are in the Table 2.5.1 and Table 2.5.2. Table 2.5.1 shows the mean and stand deviation of observation stars' reference vectors. Table 2.5.2 shows the final attitude quaternion of the telescope on the satellite at the time.

Table 2.5.1 Sensor result in Extended Kalman Filter

Sensor	#obs.	mean-x	mean-y	mean-z	std-x	Std-y	std-z
Star tracker#1	4797	0.00001	-0.00001	0.00068	0.00008	0.00211	0.00198
Star tracker#2	3849	-0.00013	0.00019	-0.00060	0.00018	0.00100	0.00091

Table 2.5.2 Quaternion result

Epoch Time			960823.021814128	
Quaternions	q_1	q_2	q_3	q_4
	-0.45707327	-0.58050044	0.33566653	0.58552252
Gyro Bias(deg/sec)	x	y	z	
	-0.00000470	-0.000001057	0.00001329	

CHAPTER 3

AN ATTITUDE DETERMINATION SYSTEM FOR BALLOONS

MTASS is a attitude determination system for satellites. For balloon experiments, an attitude determination system called Multi-mission Attitude Determination System (MADS) has been developed to determine attitude quaternions for balloon detectors.

3.1 A NEW SOFTWARE PACKAGE: MAD

As explained in chapter 2, MTASS is a software package used by NASA to determine the attitude of a satellite. MADS is a software package that has been developed using MTASS as a basis, and the main function of MADS is to support an attitude determination system for balloon missions carrying instruments such as gamma-ray or x-ray telescopes. This software package has been presented to NASA for NASA itself and university balloon experiments to determine the attitude of balloon detectors. It has same user interface as MTASS, but it uses very different kinds of data to determine the attitude of the balloon detectors. The balloon offer a low-cost, quick- response method for doing science investigations, so the balloon is widely used in atmospheric studies and in astronomy.

MADS is intended for off-line attitude determination of a balloon. The essential purpose of MADS is to determine the position and orientation of a telescope or detector

on a balloon with respect to the sky. The balloon experiments will measure and store the instruments' orientation data during a balloon flight. MADS will analyze this information to determine the balloon's motion and the attitude of the primary detector at any time.

3.2 ATTITUDE OF BALLOON DETECTOR

By the attitude of a balloon, we mean the pointing direction and rotation of the telescope with respect to the sky. MADS uses the same basic principle as MTASS in determining the quaternion for the attitude of balloon. There are again three ways to represent the attitude of balloon detector: 1. Attitude quaternion, which is a four-dimension vector in geocentric coordinate system. It is defined as ($e_x \sin\theta/2$, $e_y \sin\theta/2$, $e_z \sin\theta/2$, $\cos\theta/2$). 2. Three Euler angles (y, r , p). 3. Attitude matrix, which is a 3-by-3 matrix. The explanation of each parameter is the same as in chapter 2.2, but here the calculation of attitude quaternions are dependent on the different conditions and different initial quaternions. One of the first difference is that the balloon cannot move likes satellite on a certain (or predicted) orbital track. The balloon flies in random directions carried by atmospheric currents. We need two initial quaternions at two different time in order to find the attitude of balloon detector. The first initial quaternion is found at time when the balloon is launched. We call this initial quaternion the ground-level initial quaternion. The second initial quaternion is associated with the time when the balloon has become stable and the CCD camera begins to take the useful data. We call this one the initial quaternion. This initial quaternion is found by propagation of the ground-level initial quaternion using gyro data only in the time interval between the time of balloon launch and the time when the CCD camera begins to take the useful data. We have to use

this second initial quaternion at time when the attitude determination system starts to use the CCD data, GPS data and gyro data to calculate the attitude quaternions and propagate them time by time. The calculation for ground-level initial quaternion is based on the CCD camera's angles represented in the geocentric coordinate system and can be measured when the balloon experiment is being prepared for launch.

Suppose the camera is set up with a box. The box is forward to +Z direction in the geocentric coordinate system. Figure 3.3.1 gives the view of box in the geocentric coordinate system. From this set up, suppose (e_x, e_y, e_z) is unit vector of ground-level initial quaternion in geocentric coordinate system and α, δ, θ can be measured when the camera is set up.

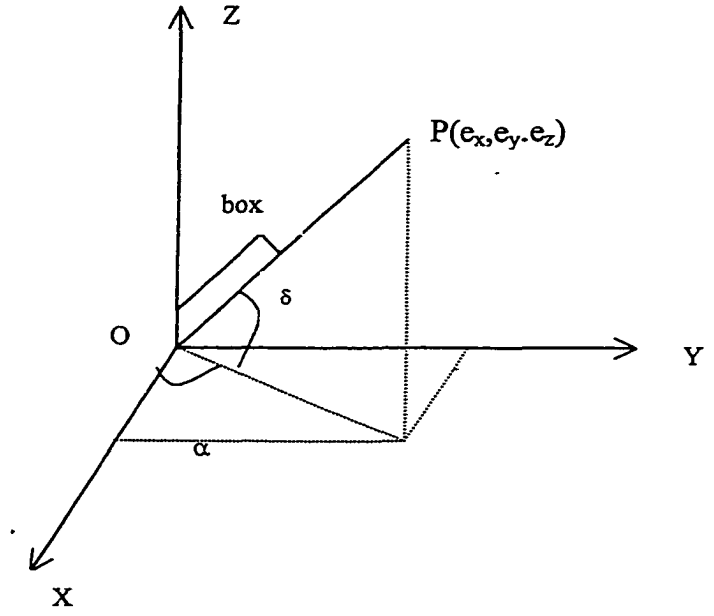

Figure 3.3.1 Ground-level initial quaternion in geocentric coordinate system

α is right ascension, and $0 \leq \alpha < 2\pi$,

δ is declination, and $-\pi/2 \leq \delta \leq \pi/2$,

θ is the angle of rotation about OP and $0 \leq \theta < 2\pi$.

The ground-level initial quaternion in geocentric coordinate system is calculated from following steps:

1. Find the unit vector (e_x, e_y, e_z) of ground-level initial quaternion

   Since the α, δ are the angles for the unit vector, so

$$e_x = \cos\delta\cos\alpha \qquad \text{(Eq.3.3.1)}$$

$$e_y = \cos\delta\sin\alpha \qquad \text{(Eq.3.3.2)}$$

$$e_z = \sin\delta \qquad \text{(Eq.3.3.3)}$$

2. Find the ground-level initial quaternion (q_1, q_2, q_3, q_4)

   Since θ is known from the measurement for camera, use the definition of attitude quaternion in chapter 2 (Eq 2.3.1) to (Eq 2.3.4), the primary quaternion can be solved as:

$$q_1 = e_x \sin\frac{\theta}{2} = \cos\delta\cos\alpha\sin\frac{\theta}{2} \qquad \text{(Eq.3.3.4)}$$

$$q_2 = e_y \sin\frac{\theta}{2} = \cos\delta\sin\alpha\sin\frac{\theta}{2} \qquad \text{(Eq.3.3.5)}$$

$$q_3 = e_z \sin\frac{\theta}{2} = \sin\delta\sin\frac{\theta}{2} \qquad \text{(Eq.3.3.6)}$$

$$q_4 = \cos\frac{\theta}{2} \qquad \text{(Eq.3.3.7)}$$

The (q_1, q_2, q_3, q_4) here is the ground-level initial quaternion at the time before the balloon start to move. Using this ground-level initial quaternion, MADS will create the initial quaternion for the attitude determination system. This will be discussed in chapter 3. 5.

3.3 BASIC DATA FOR MADS

There are three basic types of data for MADS to process in the attitude determination system for balloon:

1. Data from two CCD cameras

The CCD data, measure the stars in the star fields. The data are taken from CCD camera once every predetermined time interval so that we can have pictures of stars at each time interval during the balloon flight. Each CCD camera frame includes all the stars in a fields within m × n degrees, the numbers depending on the lens and the chip. For the Apogee AP1 camera, a frame consists of 512×768 pixels. When a star is focused, its light falls on one pixel, but the camera is then defocused slightly, so that the light falls on roughly 2×2 pixels. A window of about 8×8 pixels centered on the star is used to obtain its centroid and magnitude. MADS uses the CCD camera data and the GPS data to calculate the star's observation vector and its reference vector with respect to Earth's center in order to do the star identification for the attitude determination system. The CCD data flow diagram is shown in Figure 3.3.2 and will be explained in detail in chapter 5.

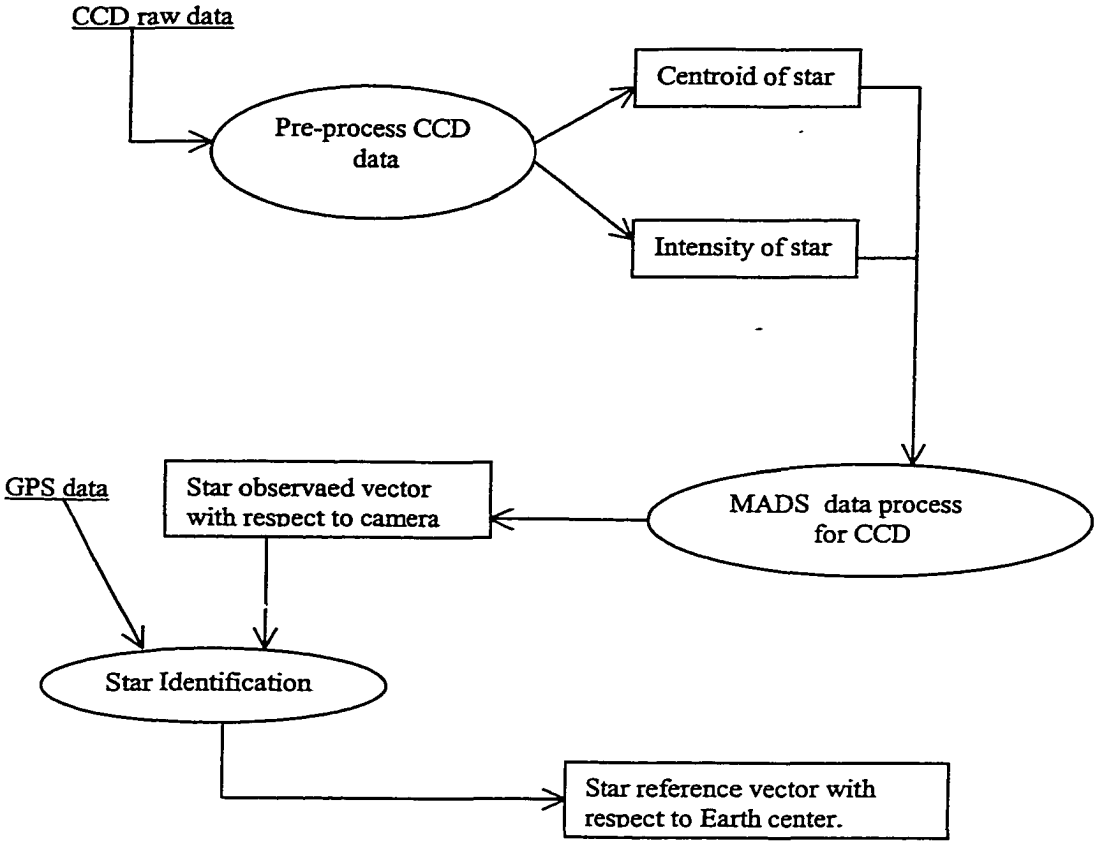

Figure 3.3.2 CCD data in MADS

Note: ▭ : output data, ——— : input data, ⬭ : process

2. Three-axis Gyro Data

The gyro data measure the rotation of the balloon while in flight. they also measures the motion of the balloon between camera pictures and provide an estimate of the next camera position. The gyro gives information about the rate of rotation about its x-axis, y-axis and z-axis in radians per unit time. MADS uses gyro data to calculate and propagate attitude quaternions from one to the next time step. The gyro data are also used to predict the attitude quaternion for balloon within a time period.

3. Global Positioning System (GPS) Data

The GPS data give the location of the balloon during its flight. The location is represented in the three geodetic coordinates: latitude, longitude, and altitude at universal time. The GPS data only provide information about the location of balloon's center of mass. The altitude is measured perpendicular to the tangent plane of the Earth. MADS use GPS information to calculate the position vector and velocity vector of the balloon with respect to Earth's center. Then with the star observed vector calculated from CCD data and the reference star vector obtained from SKYMAP, MADS calculates the observed star's reference vector with respect to Earth's center using the star identification processes. The GPS data flow diagram is shown in Figure 3.3.3 and will be discussed detail in chapter 4.

The attitude determination system for balloon uses these data and the initial quaternion to determine the attitude quaternion that gives the direction and rotation of the telescope.

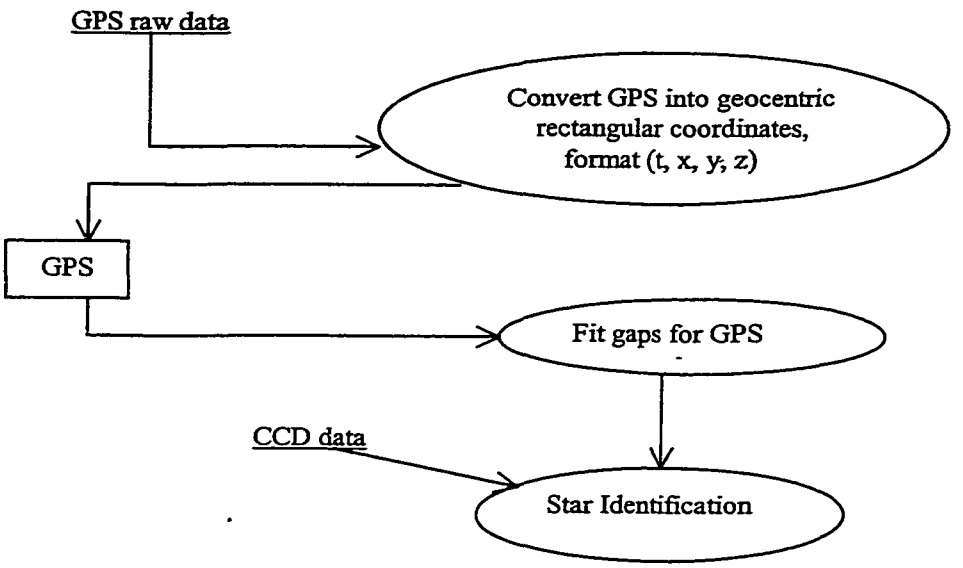

Figure 3.3.3 GPS data in MADS

Note: ▭ : output data, ——— : input data, ◯ : process

3.4 CONSTRUCTION OF MADS

MADS is an attitude determination system. The raw data used in MADS first need to be pre-processed and additional parameters calculated to get the data into the format that the attitude determination system uses. For off-line data analysis, we divide the post-flight data processing into two parts: part A, which handles pre-processing of raw data; and part B, which deals with mission-independent data analysis.

Part A pre-processes the raw data so that they can match the format expected in the main attitude determination system, and be ready as input data for MADS. In part A, the CCD data and GPS data are processed, and additional parameters that would be supplied by a star tracker are calculated here.

Since the GPS data are obtained as latitude, longitude, and altitude at universal time, we first convert them into geocentric rectangular coordinates and put them into the format as (t, x, y, z). The GPS data also need to be checked for gaps which may occur when the GPS receiver is out of range of a sufficient number of satellites. In this case, we will use polynomial interpolation to interpolate the across gaps. The detail about pre-processing GPS data will be discussed in chapter 4.

In processing the CCD camera data, first select up to the five brightest stars from each frame. For each star, determine the apparent magnitude and the location in the frame of its centroid. Then correct these positions for such effects as distortion by a spherical lens, and distortion due to temperature, which affects the focal length of the lens. The format of CCD camera data in MADS is presented as:

(time, phi, theta, temperature, intensity, flag). The details of CCD camera data are explained in chapter 5.

Part B is the mission independent analysis, essentially the attitude determination system. It uses data from part A, which include CCD data and GPS data, to calculate the observed vector of a star in balloon body coordinates using the data adjuster process, and the reference vector of the star with respect to Earth's center using the star identification process. To calculate the attitude quaternion of balloon telescope, we must follow two steps. First, we generate the initial quaternion at the time the CCD camera starts to take the useful pictures of star. This initial quaternion is found by propagation of the ground-level initial quaternion using gyro data only in the time interval between the time of balloon launch and the time when the CCD camera begins to take the useful data (i.e., the balloon has become stable). The ground-level initial quaternion is measured from the CCD camera set up on the balloon. In the second step, we find the attitude quaternion of balloon telescope with camera data, initial quaternion, and gyro data over a second time interval that is the same as the CCD camera data time interval.

Based on the discuss above, MADS has:

1. Calculation of parameters provided automatically by star trackers.
2. Algorithms for (i) locating stars, (ii) finding centroids of star with respect to camera, and (iii) calculating apparent magnitudes of star.
3. Algorithms for filling gaps in GPS data
4. Determination of the ground-level initial quaternion at the time balloon starts launch and initial quaternion at the time the balloon has became to be stable.
5. Algorithms for finding quaternion at any specified time using gyro data.

The construction of MADS is showed in Figure 3.4.1, and details about calculations of the observaed vector, reference vector, star identification, and attitude quaternions will discusses in chapter 6.

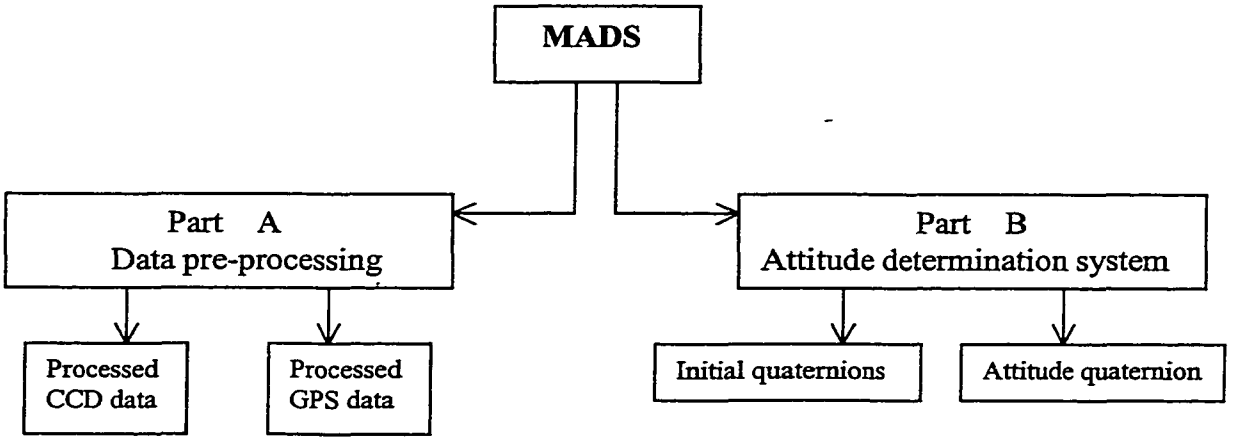

Figure 3.4.1 Construction of MADS

3.5 SUMMARY OF THE PROCESSES OF MADS

For a balloon experiment, the off-line data analysis is performed by MADS in two parts. The processes of MADS are shown in Figure 3.5.1. The raw data measured from balloon experiment include CCD data, Gyro data, and GPS data. The ground-level initial quaternion is calculated from angle measurements of the camera when it fixed on the balloon before launch.

The CCD raw data and GPS raw data are first pre-processed in part A, which has one section for CCD raw data and one for GPS raw data. The CCD raw data give the counts of each pixel in a CCD camera frame. From the counts in each pixel, the part A process for CCD data calculate the centroids of stars with respect to the center of frame, and also find the intensity of stars. The centroids and intensity are put in an array along with the time for each observed star. The GPS raw data represent the position of the balloon in latitude, longitude, and altitude. The pre-processing for the GPS data in part A converts this information into geocentric coordinates. The GPS data will be checked for gaps, and if necessary these will be filled by interpolation. The GPS data are put in an array ordered chronologically by time.

Part B of MADS uses the processed CCD data and GPS data from part A, Gyro data, and a ground-level initial quaternion of the telescope to calculate the attitude quaternions of the balloon telescope during the flight. Part B has two main processes: creation of initial quaternions, and calculation of attitude quaternions.

Part B of MADS first does data adjustments for processed CCD data and gyro data. The gyro data are the rate of rotation about its x-axis, y-axis, and z-axis at time. The adjustment of gyro data computes a corrected rate. The adjustment for processed CCD

data involves taking CCD data from part A and convert it to observed vectors in balloon body coordinates.

Creation of initial quaternions is done in two stages. The ground-level initial quaternion is a pre-flight quaternion. It is determined from the attitude of the camera, rotated to give the telescope's attitude using the carefully measured angles between their axes. The measurement of CCD camera axes include (1) deviation from North, and (2) deviation of perpendicular axes from horizontal latitude and longitude. The angles between these axes and the axes of the primary detector must be measured accurately to set up a rotation matrix.

The balloon will initially be very unstable, spinning and swinging like a pendulum as it rises. Since camera exposure times are typically on the order of 1/10 second, the camera frames will be unusable until the balloon stabilizes, several hours after launch. For a particular flight, this time will have to be determined by inspecting the camera frames. We need an initial quaternion at the time the balloon has stabilized.

The initial quaternion is determined by running MADS with only the ground-level initial quaternion and gyro data. Process propagate the ground-level initial quaternion to the time when camera data begins.

The calculation of attitude quaternions process areas follows first, the star's reference vectors are calculated, using GPS data druing the star identification process. Then based on the reference vectors, initial quaternion, and gyro data, we use the Extended Kalman Filter to calculates the attitude quaternions. Details will be discussed in chapter 6.

The MADS software package is different from MTASS in the following ways: (1) the preprocessing of CCD data to give parameters supplied automatically by a star tracker, (2) establishment of a ground-level initial quaternion at the time balloon starts launch and an initial quaternion at the time balloon has became stabilized, (3) preprocessing of GPS data to identify and interpolate across gaps, (4) replacement of orbital ephemeris by GPS data, (5) preparation of a file containing an attitude history consisting of all of the quaternions calculated.

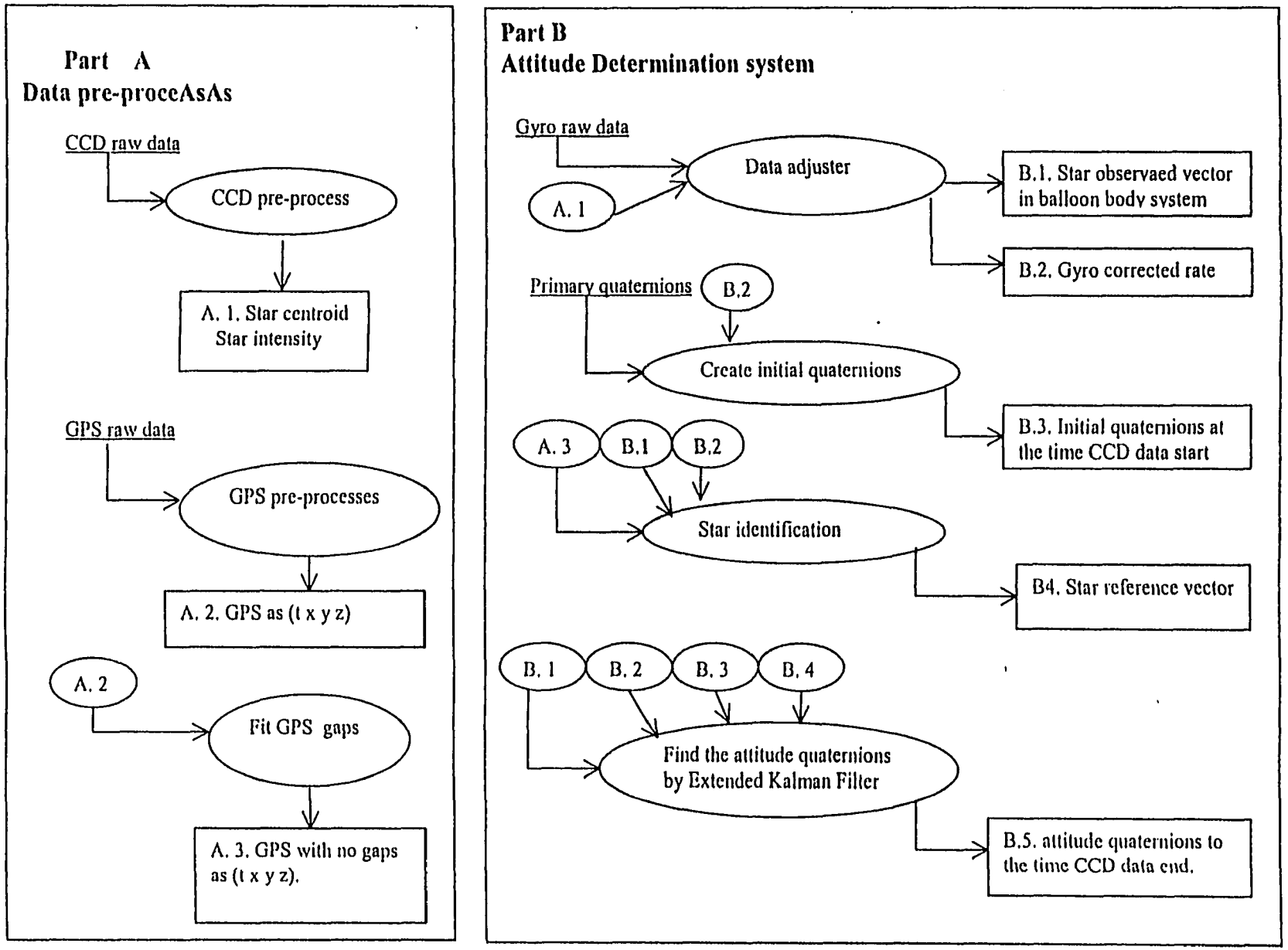

Figure 3.5.1 Processes of MADS

CHAPTER 4

GPS DATA PROCESSES IN PART A

This chapter describes the GPS data and the pre-process for the GPS data in MADS part A. There are two treatments for the row GPS data. The first is converting the raw GPS data from geodetic coordinates to geocentric rectangular coordinates. The second is fitting the GPS data if there are any missing GPS data within a required time interval during the balloon flight.

4.1 VIEW OF GPS DATA

The Global Positioning System, or GPS , is the only system today able to show an exact position on the Earth. GPS is a satellite-based radionavigation system developed and controlled by the U.S. Department of Defense (DOD) for over 20 years. While there are many thousands of civil users of GPS world-wide, the system was designed for and is operated by the U. S. military. The Global Positioning System consists of 21 satellites, as well as three back-up satellites in predictable orbits around the Earth.

GPS has three parts: the space segment, the user segment, and the control segment.

The space segment consists of 24 operational satellites, each in its own orbit 11,000 nautical miles above the Earth. The satellites continuously broadcast position and time data to users throughout the world. The GPS user segment consists of receivers and

the user community. The control segment consists of ground network of tracking stations, which monitor and control the GPS satellites in orbit to make sure the satellites are working properly.

Four satellites are required to compute the four dimensions of X, Y, Z (position) and time. GPS satellites send two signals: a carrier and a pseudo-random code. The signals are timed by an atomic clock in the satellite, and the GPS receiver generates a matching code timed by its own synchronized clock. To measure precise latitude, longitude, and altitude, the receiver measures the time it took for the signal from four separate satellites to get the receiver. By checking its time against the time of three satellites whose position are known, a receiver can pinpoint its longitude, latitude, and altitude.

The idea behind GPS is to use satellites in space as reference points for locations here on Earth. The satellites transmit signals that can be detected by any with GPS receiver. The GPS works in five logical steps:(1) Triangulating from satellites: this is the basis of GPS, (2) measuring distance from a satellite, (3) getting perfect timing, (4) knowing where a satellite is in space, (5) correcting errors.

The GPS system depends on two things to make it work:

- First, each satellite has an onboard atomic clock that gives it an extremely precise time base. The satellites send radio signals to the receiver, and the extremely precise time bases make it possible for the receiver to determine exactly how far away each satellite is. The receiver can to calculate exactly how long it took for the signal to travel from the satellite to the receiver, and from that time determine the exact distance between the receiver and the satellite.

- Second, each receiver has stored in memory an almanac that indicates exactly where each satellite is in its orbit at any moment. The almanac is possible because of the extremely precise orbit flown by the satellites.

The purpose of many balloon experiments is to search for or research energy sources in the sky. To find the pointing and rotation of telescope, the position and velocity of balloon are very important information, so the GPS receiver is used on a balloon to find the position and velocity of the balloon at universal time during its flight.

4.2 GPS DATA IN BALLOON EXPERIMENTS

A balloon experiment first uses the GPS to get the position of the balloon at launch and later uses the GPS data to calculate reference vectors of stars with respect to Earth's center. The GPS receiver gives the location of the balloon and universal time. The balloon's position is given in terms of its latitude, longitude, and altitude at a universal time. The format of the GPS data from the receiver is actually presented as (time, latitude, longitude, altitude). These track the motion of the center of mass of the balloon. The altitude is measured perpendicular to a tangent plane of the Earth. Since the GPS data are in geographical coordinates, we need to convert those coordinates to geocentric rectangular coordinates. The balloon position represented by GPS data is shown in Figure 4.2.1.

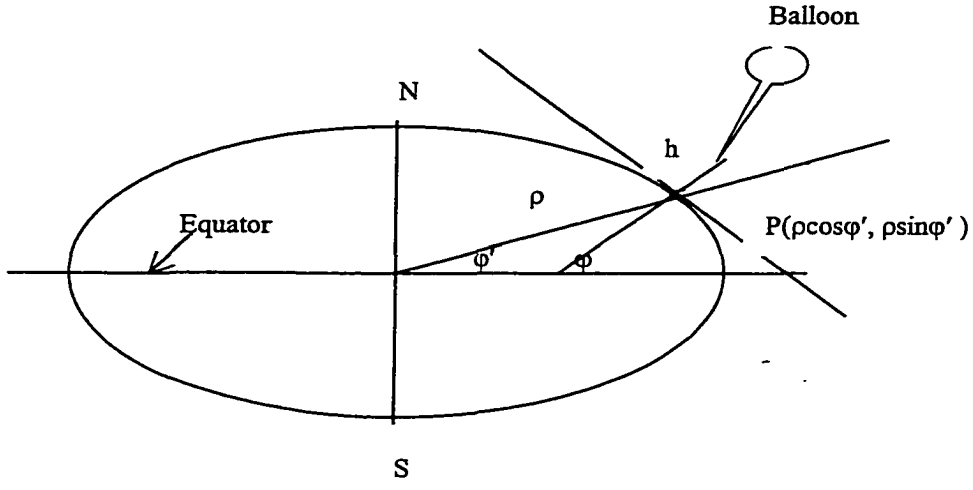

Figure 4.2.1 Balloon position data from GPS

φ is the latitude of the balloon from GPS.

φ′ is the geocentric latitude of balloon.

h is the altitude of the balloon from GPS.

ρ is the radius of Earth.

Figure 4.2.1 gives the position of the balloon respect to the Earth center. The angle φ is read out from the GPS data as the latitude of the balloon, and h is altitude of balloon. To calculate the balloon position in geocentric rectangular coordinates, we first need to know the quantities ρsinφ′ and ρcosφ′, where ρ is the radius of Earth at that latitude, and φ′ is the geocentric latitude of balloon.

As we know, the Earth is not quite spherical, but is instead more like a spheroid of revolution, being flattened along the line joining the north and south poles. A cross-section through the Earth along any line of longitude would be approximately elliptical, while a cross-section along any line of latitude would be circular. In the figure 4.2.1page 52), the Earth is drawn with its north and south poles, N and S. When the balloon at position P locates its zenith by means of a plumb line to be along the line PZ; the angle this makes with the equatorial plane defines the balloon's geographical latitude, φ. Since the Earth is not quite spherical, the balloon's geocentric latitude φ′ is slightly different than the balloon's geographical latitude, φ. The calculation of ρsinφ′ and ρcosφ′, are done as following:

$$u = \tan^{-1}\{0.996647 \tan\varphi\}$$

$$h' = (h / 6378140)$$

(Peter D. Smith 1981). Here, h is the altitude of the balloon, which is its height above sea-level. The constant 6378140 is the equatorial radius of the Earth in meters. The h′ is a correction of height h above sea-level. Then,

$$\rho \sin\varphi' = 0.96647 * \sin u + h' * \sin\varphi$$

$$\rho \cos\varphi' = \cos u + h' * \cos\varphi$$

and $\theta' = \theta$

where θ' is the geocentric longitude.

4.3 PRINCIPLES OF TRANSFORMATION OF GPS DATA

Figure 4.2.1 shows the balloon position with respect to Earth. We have known the quantities ρsinφ′ and ρcosφ′, so we can show the balloon's position in rectangular coordinates in Figure 4.3.1:

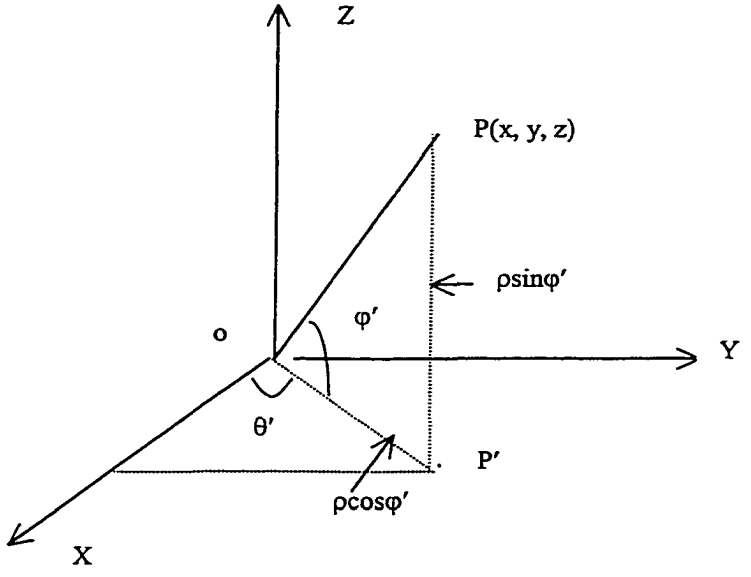

Figure 4.3.1 Balloon position in rectangular coordinate system

This is a geocentric rectangular coordinate system with $\mathbf{Z} = \mathbf{X} \times \mathbf{Y}$. The geocentric coordinate axes originate in the Earth's center. The +Z –axis points north along the Earth's spin axis, the +X-axis points to the vernal equinox direction in the Earth's equatorial plane, and the +Y-axis completes the orthogonal triad. P(x, y, z) represents a position of balloon in the system. P′ is point of projection from P on the XY-plane. The

φ' is an angle between the OP and OP′ in the plane that includes point P and is perpendicular to the XY-plane. The $\rho\sin\varphi'$ is the distance from P to XY- plane, and $\rho\cos\varphi'$ is distance from origin of system to P′. θ' is the angle between the X axis and OP′ in XY-plane. The angles φ' and θ' are in different planes.

In the OPP′-plane,

$$\sin\varphi' = z/\rho .$$

In the XY-plane,

$$\sin\theta' = \frac{y}{\rho\cos\varphi'} \quad \text{and} \quad \cos\theta' = \frac{x}{\rho\cos\varphi'}.$$

So the balloon position P(x, y, z) in geocentric rectangular coordinates system is represented as:

$$x = \rho\cos\varphi'\cos\theta'$$

$$y = \rho\cos\varphi'\sin\theta'$$

$$z = \rho\sin\varphi'$$

The GPS data are translated from geographical coordinates into geocentric rectangular coordinates. So now the GPS represent the balloon position as (t, x, y, z,) which are the x coordinate, y coordinate, and z coordinate with universal time at that position. MADS will use the balloon position (t, x, y, z,) to calculate the reference vector for star.

4.4 INTERPOLATIONS FOR MISSING GPS DATA

During a balloon flight, the GPS receiver may be out of the range of enough satellites to determine its position for some time interval, due to the random motion of the balloon. This is especially likely to happen in the Arctic and Antarctic which are used for long duration flights. In this case, GPS data for the balloon position will be missing for that time interval. The GPD receiver is fixed on the balloon. For a 10-day flight, there is no guarantee that GPS receiver will receive the signal from enough satellites at all times. Since the GPS determines the position of balloon, if GPS data miss for a too long a time interval, we need to fit the missing data for the time gap to predict the position of balloon.

Even though the flight balloon moves randomly in the sky, the track for a balloon will be a curve, so the fitting methods will use polynomial interpolation to interpolate the across time gaps. As we know, the GPS data are represented in four-components (time, x, y, z). For each spatial component x, y, and z, we use same method to do the curve fitting over a time gap. A function g(x) is said to be an interpolation function for a given set of data points (t_1, x_1), (t_2, x_2),..... (t_n, x_n) if its graph passes through (not just close to) selected accurately tabulated points $p_i(t_i, x_i)$ in the tx-plane, where $x_i = f(t_i)$ is rounded to the accuracy of the table. Geometrically, interpolation is the game of "follow the dots." The analytic challenge is to find functional descriptions for curves that do this. Once such a function is found, its value at t can be used to approximate f(t) when t is not a tabulated value t_i This geometric problem is usually solved by transforming it to the algebraic problem of solving equations.

The most natural curve to try to pass through (n + 1) tabulated p_i's is a polynomial having (n + 1) coefficients, that is of degree n. This is called polynomial interpolation.

There are several polynomial interpolation such as Lagrange, Newton, and piecewise. The piecewise interpolation also has two kinds of interpolation functions. One is linear interpolations, and other is piecewise cubic splines. We will use the piecewise cubic splines interpolation method to fit the missing data for the GPS because the balloon's motion will not be linear, and the GPS data will be an curve. If GPS data are missing for two hours during a balloon flight, we will need to find the interpolation function based on data before and after the missing data time interval. The method for the interpolation is as following:

1. Definition

Suppose the GPS data miss in time interval $[t_a\ t_b]$, so pick up 2m's GPS data points at time: $t_{a-m}, t_{a-(m-1)}, \ldots t_{a-1}, t_{b+1}, t_{b+2}, \ldots t_{b+m}$. The first m's GPS data points are in the time interval before the missing data time interval. The rest of m's GPS data points are picked up from the time interval after the missing data time interval. For convenience of explaining the methods, we write them as: $t_0, t_1, \ldots t_n$, where $n = 2m-1$ and $t_0 < t_1 < \ldots < t_n$. Let $p_k(t_k, x_k)$ represent the first component of GPS data at the time t_k, where $x_k = f(t_k)$. $k = 0,1,2,\ldots n$.

The function s(t) is called a piecewise cubic on $[t_0, t_n]$ if there exist cubic functions $q_0(t), \ldots q_{n-1}(t)$ such that

$$s(t) = q_k(t) \text{ on } [t_k, t_{k+1}] \quad \text{for } k = 0,1,2,\ldots n-1$$

must satisfy:

$$q_k(t_k) = x_k \quad \text{and} \quad q_k(t_{k+1}) = x_{k+1} \qquad \text{(Eq. 4.4.1)}$$

$$q'_{k-1}(t_k) = q'_k(t_k) \ [=s'(t_k)] \quad \text{for } k = 1,2,\ldots n-1 \qquad \text{(Eq. 4.4.2)}$$

$$q''_{k-1}(t_k) = q''_k(t_k) \ [=s''(t_k)] \quad \text{for } k = 1,2,\ldots n-1 \qquad \text{(Eq. 4.4.3)}$$

There are 2n conditions in Eq. 4.4.1, together with n-1 condition in each of Eq. 4.4.2 and Eq. 4.4.3, ensure that s(t) and both its first and second derivatives are continuous on $[t_0,t_n]$.

2. Find s(t) in each $[t_k, t_{k+1}]$

Since s(t) is a piecewise cubic function on the $[t_0,t_n]$, then its second derivative s''(t) is piecewise linear on the $[t_0,t_n]$; By Eq.4.4.3, $q''_k(t_k) = s''(t_k)$, $q''_k(t)$ is linear and interpolates $(t_k, s''(t_k))$ and $((t_{k+1}, s''(t_{k+1}))$ in $[t_k, t_{k+1}]$. By lagrange's piecewise linear interpolation form,

$$q''_k(t) = s''(t_k)\left(\frac{t-t_{k+1}}{t_k-t_{k+1}}\right) + s''(t_{k+1})\left(\frac{t-t_k}{t_{k+1}-t_k}\right), \qquad k=0,1,...n-1 \quad \text{(Eq. 4.4.4)}$$

let $h_k = t_{k+1} - t_k$ for $k=0,1,...n$

$\sigma_k = s''(t_k)$ for $k=0,1,...n$

Then Eq. 4.4.4 can be written as

$$q''_k(t) = \frac{\sigma_k}{h_k}(t_{k+1}-t) + \frac{\sigma_{k+1}}{h_k}(t_k - t), \qquad k=0,1,...n-1 \qquad \text{(Eq. 4.4.5)}$$

where the h_k's and σ'_k s are constants, with the σ'_k s to be determined. So,

$$q'_k(t) = \frac{-\sigma_k}{h_k}\frac{(t_{k+1}-t)^2}{2} - \frac{\sigma_{k+1}}{h_k}\frac{(t_k-t)^2}{2} + \lambda_k, \qquad k=0,1,...n-1 \qquad \text{(Eq. 4.4.6)}$$

$$q_k(t) = \frac{\sigma_k}{h_k}\frac{(t_{k+1}-t)^3}{6} + \frac{\sigma_{k+1}}{h_k}\frac{(t_k-t)^3}{6} + \lambda_k(t), \qquad k=0,1,...n-1 \qquad \text{(Eq. 4.4.7)}$$

here, $\lambda_k(t) = A_k(t-t_k) + B_k(t_{k+1}-t)$.

Solving A_k and B_k by using (Eq.4.4.1):

Let $t = t_k$, then,

$$q_k(t_k) = \frac{\sigma_k}{h_k}\frac{(t_{k+1} - t_k)^3}{6} + B_k(t_{k+1} - t_k)$$

i.e $\quad q_k(t_k) = \frac{\sigma_k}{6}h^2{}_k + B_k h_k = t_k$

So, $\quad B_k = \frac{t_k - \frac{\sigma_k}{6}h_k^2}{h_k}$

Let $t = t_{k+1}$, then,

$$q_k(t_{k+1}) = \frac{\sigma_{k+1}}{h_k}\frac{(t_k - t_{k+1})^3}{6} + A_k(t_{k+1} - t_k)$$

i.e $\quad q_k(t_{k+1}) = \frac{\sigma_{k+1}}{6}h^2{}_k + A_k h_k = t_{k+1}$

So, $\quad A_k = \frac{t_{k+1} - \frac{\sigma_{k+1}}{6}h_k^2}{h_k}$

Put A_k and B_k into (Eq. 4.4.7), we have:

$$q_k(t) = \frac{\sigma_k}{h_h}\frac{(t_k - t)^3}{6} + \frac{\sigma_{k+1}}{6}\frac{(t - t_k)^3}{h_k} + \frac{t_{k+1} - \frac{\sigma_{k+1}}{6}h_k^2}{h_k}(t - t_k) + \frac{t_k - \frac{\sigma_k}{6}h_k^2}{h_k}(t_{k+1} - t)$$

i.e

$$q_k(t) = \frac{\sigma_k}{6}\{\frac{(t_k - t)^3}{h_k} - h_k(t_{k+1} - t)\} + \frac{\sigma_{k+1}}{6}\{\frac{(t - t_k)^3}{h_k} - h_k(t - t_k)\} + t_k\frac{(t_{k+1} - t)}{h_k} + t_{k+1}\frac{(t - t_k)}{h_k} \qquad \text{k} = 0,1,.....\text{n} - 1 \quad , \text{(Eq. 4.4.8)}$$

The $q_k(t)$ are the piecewise cubic functions on $[t_k, t_{k+1}]$ for $k = 0,1,\ldots n-1$.

They can be used to evaluate s(t) (as $q_k(t)$) for $t_k \leq t \leq t_{k+1}$ once we know the values

of σ_k and σ_{k+1}. There are n-1 piecewise functions on $[t_0,t_n]$, with σ_0 .....σ_n. The following step is finding the σ_0 .....σ_n for the piecewise function s(t).

3. Solving the system of σ_0 .....σ_n

In order to solve the σ_0 .....σ_n, we first find the equation system for σ_0 .....σ_n. Differentiating (Eq. 4.4.8) gives

$$q_k'(t) = \frac{\sigma_k}{6}\{\frac{-3(t_{k+1}-t)^2}{h_k} + h_k\} + \frac{\sigma_{k+1}}{6}\{\frac{3(t-t_k)^2}{h_k} - h_k\} + \frac{(x_{k+1}-x_k)}{h_k}$$

Let $t = t_k$, and $\Delta x_k = \frac{(x_{k+1}-x_k)}{h_k}$, then

$$q_k'(t_k) = \frac{\sigma_k}{6}\{\frac{-3(t_{k+1}-t_k)^2}{h_k} + h_k\} + \frac{\sigma_{k+1}}{6}\{-h_k\} + \frac{(x_{k+1}-x_k)}{h_k}$$

i.e

$$q_k'(t_k) = \frac{\sigma_k}{6}\{-2h_k\} + \frac{\sigma_{k+1}}{6}\{-h_k\} + \Delta x_k \qquad \text{(Eq.4.4.9)}$$

Let $t = t_{k+1}$, then

$$q_k'(t_{k+1}) = \frac{\sigma_k}{6}\{+h_k\} + \frac{\sigma_{k+1}}{6}\{\frac{3(t_{k+1}-t_k)^2}{h_k} - h_k\} + \frac{(x_{k+1}-x_k)}{h_k}$$

i.e

$$q_k'(t_{k+1}) = \frac{\sigma_k}{6}\{+h_k\} + \frac{\sigma_{k+1}}{6}\{2h_k\} + \Delta x_k \qquad \text{(Eq. 4.4.10)}$$

Let k-1 replace k in (Eq. 4.4_7), we have

$$q_{k-1}'(t_k) = \frac{\sigma_{k-1}}{6}\{+h_{k-1}\} + \frac{\sigma_k}{6}\{2h_{k-1}\} + \Delta x_{k-1} \qquad \text{(Eq. 4.4.11)}$$

By (Cond. 4.4.2)

$$\frac{\sigma_k}{6}\{-2h_k\}+\frac{\sigma_{k+1}}{6}\{-h_k\}+\Delta x_k=\frac{\sigma_{k-1}}{6}\{+h_{k-1}\}+\frac{\sigma_k}{6}\{2h_{k-1}\}+\Delta x_{k-1}$$

i.e

$$h_{k-1}\sigma_{k-1}+2(h_{k-1}+h_k)\sigma_k+h_k\sigma_{k+1}=6[\Delta x_k-\Delta x_{k-1}] \qquad k=1,2,.....n-1 \text{ (Eq. 4.4.12)}$$

In (Eq. 4.4_9) , there are n-1's equations with n+1 unknown σ_0 .....σ_n . If σ_0 and σ_n are given by boundary condition, this equation system includes n-1 equations and n-1 unknown variables σ_1 .....σ_{n-1}. This system is presented as

$$A\sigma = D \qquad \text{(Eq. 4.4.13)}$$

where

$$A=\begin{bmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix}$$

$$\sigma=\begin{bmatrix}\sigma_1\\ \sigma_2\\ .\\ .\\ \sigma_{n-1}\end{bmatrix}, \qquad D=\begin{bmatrix}6(\Delta x_1-\Delta x_0)-h_0\sigma_0\\ 6(\Delta x_2-\Delta x_1)\\ .\\ .\\ 6(\Delta x_{n-2}-\Delta x_{n-3})\\ 6(\Delta x_{n-1}-\Delta x_{n-2})-h_{n-1}\sigma_n\end{bmatrix}$$

This system is a (n-1) x (n-1) tridiagonal system. Since h_0, h_1, ...h_{n-1} are positive and coefficient matrix A is a diagonally dominant, so (Eq. 4.4.13) can always be solved uniquely for σ_1 .....σ_{n-1} (using Gaussian Elimination Algorithm). There are some strategy for calculating the boundary condition of σ_0 and σ_n. We

tried using them to find the best boundary conditions for the GPS data fitting problem.

4. Endpoints strategies

The equation system Eq.4.4.13 $A\sigma = D$ for σ_0 ..... σ_n has n-1 equations. The equation (E1) and equation (En-1) have σ_0 and σ_n that can be eliminating by endpoints of interval $[t_0,t_n]$. We have four case of eliminating for σ_0 and σ_n.

Case 1. $\sigma_0 = s''(t_0)$, and $\sigma_n = s''(t_n)$

In this case, the values of $s''(t_0)$ and $s''(t_n)$, are calculated base on the endpoints t_0, t_1, t_2 and t_n, t_{n-1}, t_{n-2}.

Suppose $s''(t_0) = A_0 s(t_0) + A_1 s(t_1) + A_2 s(t_2)$

Here

$$s(t_1) = s(t_0 + h) = s(t_0) + hs'(t_0) + \frac{h^2}{2}s''(t_0) + O(h^3)$$

$$s(t_2) = s(t_0 + 2h) = s(t_0) + 2hs'(t_0) + \frac{(2h)^2}{2}s''(t_0) + O(h^3)$$

So,

$$s''(t_0) = A_0 s(t_0) + A_1\{s(t_0) + hs'(t_0) + \frac{h^2}{2}s''(t_0)\} + A_2\{s(t_0) + 2hs'(t_0) + \frac{(2h)^2}{2}s''(t_0)\}$$

i.e

$$s''(t_0) = (A_0 + A_1 + A_2)s(t_0) + (A_1 + 2A_2)hs'(t_0) + (A_1 + 4A_2)\frac{h^2}{2}s''(t_0)$$

Then

$$\begin{cases} A_0 + A_1 + A_2 = 0 \\ (A_1 + 2A_2)h = 0 \\ (A_1 + 4A_2)\dfrac{h^2}{2} = 1 \end{cases} \qquad \text{So,} \quad \begin{cases} A_0 = \dfrac{1}{h^2} \\ A_1 = -\dfrac{2}{h^2} \\ A_2 = \dfrac{1}{h^2} \end{cases}$$

Hence,

$$s''(t_0) = \frac{1}{h^2}s(t_0) - \frac{2}{h^2}s(t_1) + \frac{1}{h^2}s(t_2)$$

and $$s''(t_n) = \frac{1}{h^2}s(t_n) - \frac{2}{h^2}s(t_{n-1}) + \frac{1}{h^2}s(t_{n-2})$$

since $s(t_i) = x_i$ for i = 0,1,2,.....n , so the $s''(t_0)$ and $s''(t_n)$ can be calculate as values. Put $\sigma_0 = s''(t_0)$, and $\sigma_n = s''(t_n)$ into (Eq. 4.4.13), We can get uniquely σ_0 ..... σ_n.

Case 2. $\sigma_0 = \sigma_1$, and $\sigma_n = \sigma_{n-1}$

In this case, assume that s''(t) is constant near the endpoint. Replace $\sigma_0 = \sigma_1$, and

$\sigma_n = \sigma_{n-1}$ into (Eq.4.4.13), we got:

$$A = \begin{bmatrix} 3h_0 + 2h_1 & h_1 & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & \\ & h_2 & 2(h_2 + h_3) & h_3 & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & h_{n-2} & 2h_{n-2} + 3h_{n-1} \end{bmatrix}$$

$$D=\begin{bmatrix} 6(\Delta x_1-\Delta x_0) \\ 6(\Delta x_2-\Delta x_1) \\ \cdot \\ \cdot \\ 6(\Delta x_{n-2}-\Delta x_{n-3}) \\ 6(\Delta x_{n-1}-\Delta x_{n-2}) \end{bmatrix} \qquad \text{(Eq. 4.4.14)}$$

Solve this equation system, we can get the unique value for σ's.

• Case 3. $\sigma_0=\frac{1}{h_1}\{(h_0+h_1)\sigma_1-h_0\sigma_2\}$ and

$$\sigma_n=\frac{1}{h_{n-2}}\{-h_{n-1}\sigma_{n-2}+(h_{n-2}+h_{n-1})\sigma_{n-1}\}$$

In this case, assume that s''(t) is linear near the endpoint. Replacing σ_0 and σ_n into (Eq.4.4.13), the system is became as:

$$A=\begin{bmatrix} (2+\frac{h_0}{h_1})(h_0+2h_1) & h_1-\frac{h_0^{\,2}}{h_1} & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & h_{n-2}-\frac{h_{n-1}^{\,2}}{h_{n-2}} & (2+\frac{h_{n-1}}{h_{n-2}})(h_{n-2}+h_{n-1}) & \end{bmatrix}$$

and D is same as (Eq.4.4.14). Solve this system, we can get the unique values for σ's.

Case 4. $\sigma_0=\frac{3}{h_0}[\Delta x_0-s'(t_0)]-\frac{1}{2}\sigma_1$ and $\sigma_n=\frac{3}{h_{n-1}}[s'(t_n)-\Delta x_{n-1}]-\frac{1}{2}\sigma_{n-1}$

In this case, specify the values of s'(t) at endpoint. The s'(t) calculated here are based on the points t_0, t_1, t_2 and t_n, t_{n-1}, t_{n-2}.

By Three-point formulas for differentiation, we have

$$s'(t_0) = \frac{1}{2h}[-3s(t_0) + 4s(t_1) - s(t_2)] \text{ and } s'(t_n) = \frac{1}{2h}[3s(t_n) - 4s(t_{n-1}) + s(t_{n-2})]$$

Replacing σ_0 and σ_n into (Eq.4.4.13), the system is becomes:

$$A = \begin{bmatrix} (2-\frac{1}{2})h_0 + 2h_1 & h_1 & & \\ h_1 & 2(h_1+h_2) & h_2 & \\ & h_2 & 2(h_2+h_3) & h_3 \\ \cdots & \cdots & \cdots & \cdots \\ & & h_{n-2} & 2h_{n-2} + (2-\frac{1}{2})h_{n-1} \end{bmatrix}$$

$$D = \begin{bmatrix} 6(\Delta x_1 - \Delta x_0) - (\Delta x_0 - s'(t_0)) \\ 6(\Delta x_2 - \Delta x_1) \\ \cdot \\ \cdot \\ 6(\Delta x_{n-2} - \Delta x_{n-3}) \\ 6(\Delta x_{n-1} - \Delta x_{n-2}) - (s'(t_n) - \Delta x_{n-1}) \end{bmatrix}$$

Solve this equations system, we also get other set of values for σ's.

Those four endpoint strategies treat the second derivative on endpoints t_0 and t_n in different ways. Which one is best is will depend on the data curve. Since the balloon moves slowly, the GPS data for balloon follow a very smooth curve, so those methods give us very similar results of interpolation of the missing data. The Figure 4.4. and

Figure 4.4.2 give the fitting curve on x component and y component by using third method of endpoint strategies respectively. For the better fitting, we choose 45 data points before the GPS data missing time interval and 45 data points after the GPS data missing time interval. We also tried to use only 20 data points before the GPS data missing time interval and 20 data points after the GPS data missing time interval, but the curve did not fit the data very well.

Once we solve the system of equations and get a set of values for σ's, the interpolation cubic function $s(t) = q_k(t)$ is determined on each interval $[t_k, t_{k+1}]$ for $k = 0, 1, \ldots n-1$. The length of interval $[t_k, t_{k+1}]$ are not same for each interval, because $[t_0, t_n]$ includes the interval $[t_a\ t_b]$ that missing the GPS data for balloon positions. Use interpolation cubic function $q_a(t)$ on $[t_a\ t_b]$ to calculate the x(ti) within $[t_a\ t_b]$.

Here $i = a + ih_0$ and $a < i < b$.

The GPS data have four components: (t, x, y, z). We use the same interpolation methods on each component x, y, and z respectively to get the interpolation values x, y, and z such that they fit the GPS data curve. Then put them into format (t, x, y, z) data file for MADS to use. We have separate programs for each endpoints strategies to interpolate the GPS missing data. Since we perform the third method which is using endpoints as $\sigma_0 = \frac{1}{h_1}\{(h_0 + h_1)\sigma_1 - h_0\sigma_2\}$ and $\sigma_n = \frac{1}{h_{n-2}}\{-h_{n-1}\sigma_{n-2} + (h_{n-2} + h_{n-1})\sigma_{n-1}\}$, the program for this method has been down for each component. Other programs are for one component.

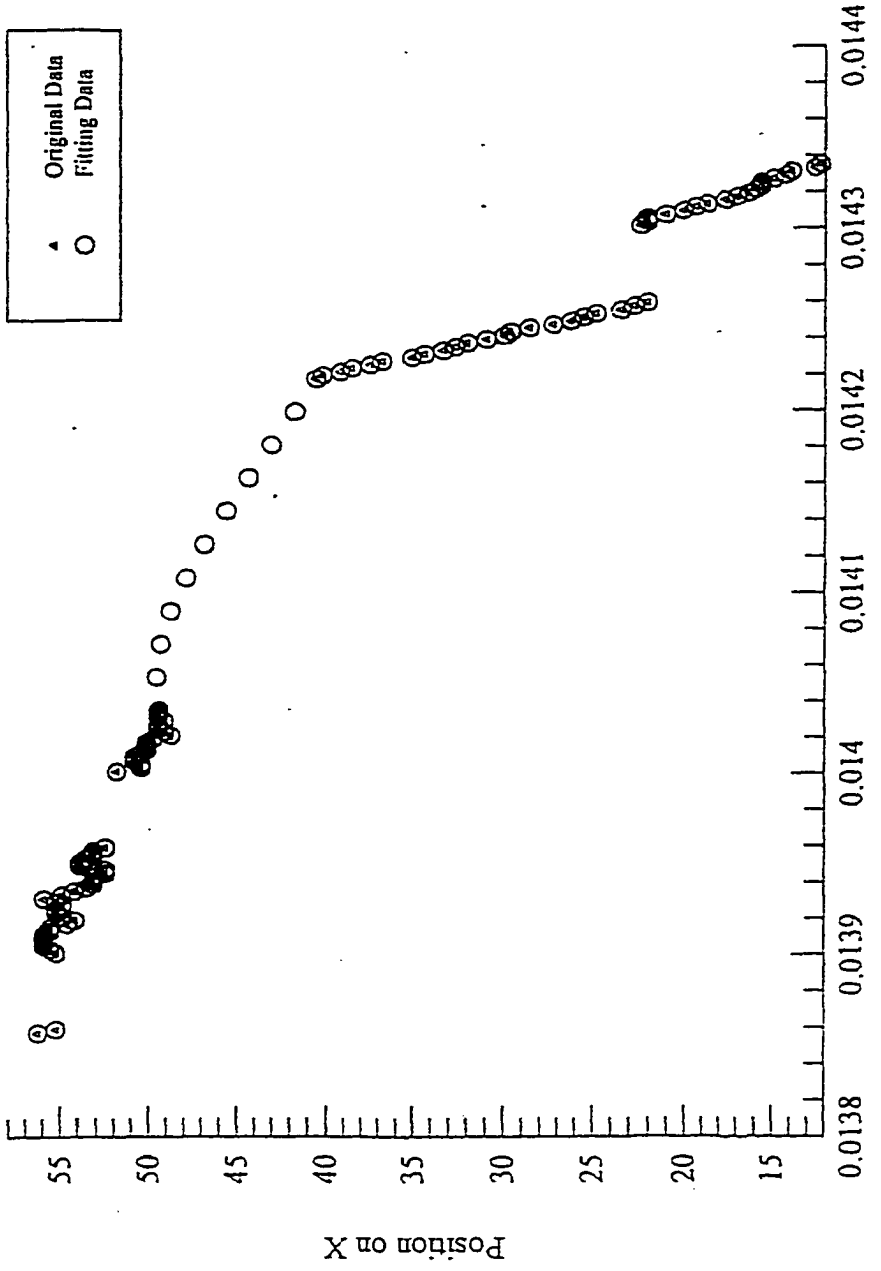

Figure 4.4.1. GPS fitting curve on the x component vs. time.

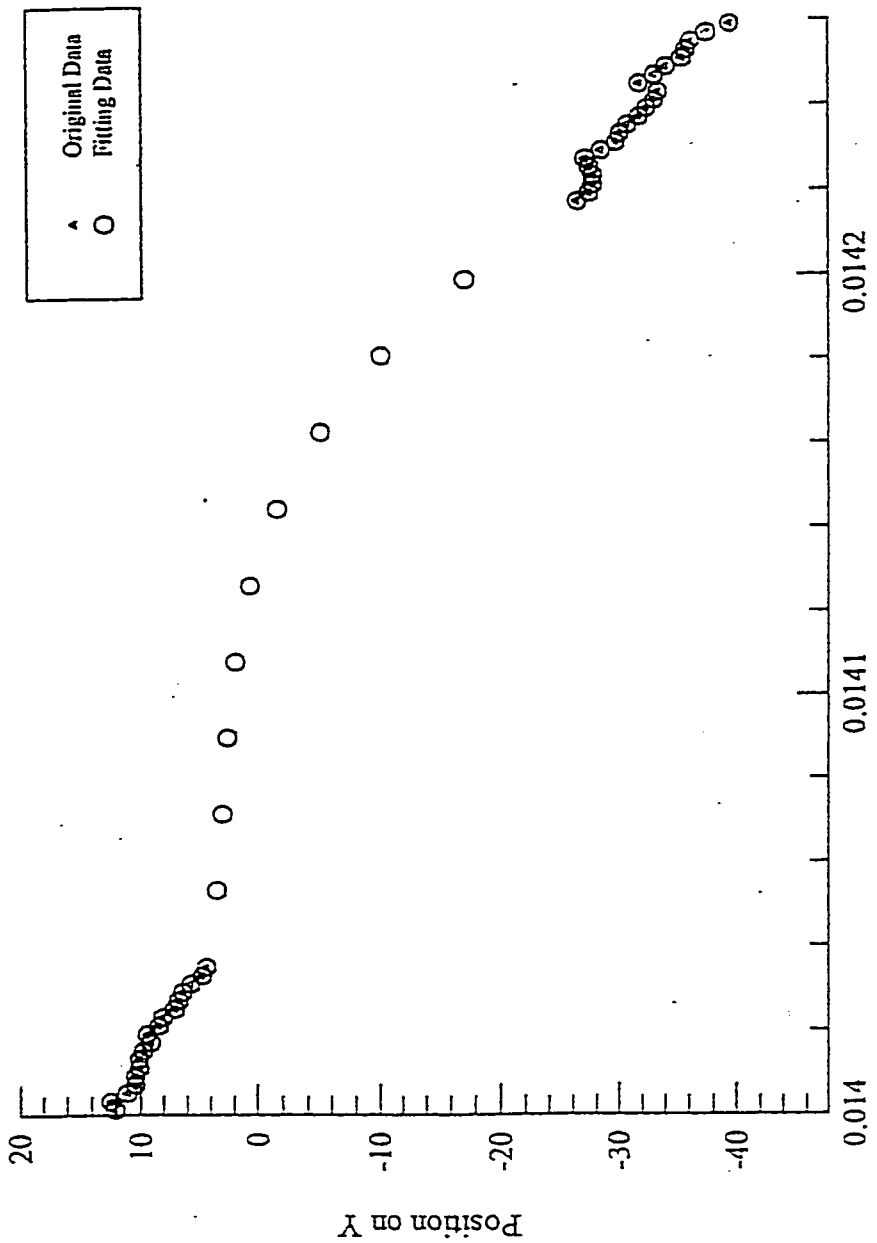

Figure 4.4.2. GPS fitting curve on the y component vs. time.

CHAPTER 5

REDUCTION OF CAMERA DATA

To identify stars, a balloon experiment uses CCD cameras to take pictures of stars and determines the orientation of the balloon instruments. This chapter discusses the appropriate reduction of CCD camera data.

5.1 MODELING A STAR TRACKER

1. Principles of operation of a star tracker

Satellites use fixed head star tracker information for identification of stars in MTASS. Most satellite attitude control systems or attitude determination systems today use star trackers. These will never view the Sun or Earth, since each has an associated bright object sensor and a stray light shield. A star tracker is designed to acquire and track stars in a predetermined magnitude interval and field of view. The trackers provides two-axis position information relative to its chip.

A fixed head star tracker consists of six basic subsystems: sensor head , signal processor, synchronous logic unit, power converter, digital deflection unit, and input/output interface unit.

When power is applied to the system, the fixed head star tracker will automatically start searching the total field of view. The search scan pattern is a

horizontal (H) direction sawtooth sweep with a staircase signal applied to the vertical (V) direction. The track pattern centers on the star image and a 'star present' bit is generated in the output words signifying that two-axis position and intensity data are available. If the star moves in the field of view due to balloon attitude changes, the track scan pattern will follow and remain centered on the star image. The star's position is identified by a positive or negative H coordinate and a positive or negative V coordinate.

As the star moves in the field of view due to satellite attitude changes, the track pattern remains locked onto the star image, and the position information is continuously updated. The fixes head star tracker sees only the acquired star; it is not influenced by and provides no information about other targets in the field of view. Tracking of the same star continues until the star leaves the field of view or the star intensity falls below the commanded level. The number of stars followed by a tracker is different for each model; the Ball star tracker used by RXTE tracks up to five stars.

In general, a fixed head star tracker determines star position vectors. The fixed head star tracker output consists of three parameters: H, the horizontal coordinate of the star in the field of view; V, the vertical coordinate of the star in the field of view; and I, the star-generated signal intensity. The intensity determines the star magnitude. The magnitude measurement error typically is quite high, so it gives only limited help in identifying the star. In MTASS, fixed head star tracker data are used to calculate observed vectors of stars with respect to the tracker in the satellite, and the ephemeris data are used to calculate

reference vectors of star with respect to Earth's center. Then comparison of observed star's vectors and candidate star's reference vectors is used to identify the stars. In MTASS, the fixed head star tracker data is input data with format: (time, phi, theta, temperature, intensity, flag).

2. Modeling a star tracker using a CCD camera

Balloon experiments use CCD cameras to take pictures of stars at predetermined intervals, for example, at once a minute. Storing the images is a problem if telemetry is not available. The images have to be either stored on board or sent down by telemetry.

In MADS, the CCD camera data are used to calculate observed vectors of stars with respect to the camera in the balloon, and GPS data are used to calculate the reference vectors of stars with respect to Earth's center (The star's observed vectors and reference vectors will be discusses in chapter 6). Then observed vectors and reference vectors are used to identify stars as in the case of the satellite. The CCD data are used one frame at a time, and each picture includes all stars in the field of view.

The raw data of the CCD camera are counts of pixels in the chip. To use the CCD data for MADS, the CCD camera data have to be pre-processed. In the pre-processes of CCD camera, we need to pick up between three and five of the brightest stars from each frame because MADS requires three to five stars in the frame for identifying stars. For each star, calculate the angles phi, and theta, and its intemsity (Figure 5.1.1). Finally, we put the data into the format (time, phi,

theta, camera temperature, intensity, flag) for each star in one frame at the time. With a different time tag, there are different camera frame data.

The CCD camera frame coordinate system is shown on the figure 5.1.1.

- *b* is distance along the boresight from the lens to the chip.
- **V** is vertical component in the plane of CCD chip.
- **H** is horizontal component in the plane of CCD chip and

perpendicular to **V**, so $\mathbf{H} = \mathbf{V} \times \mathbf{b}$, where **b** is a vector along the boresight, dircted from the chip to the lens.

- θ is angle associated with vertical component of image position.
- ϕ is angle between the boresight and the projection onto the plane

of the boresight and H of the line of sight to the star. It is associated with horizontal component of image position

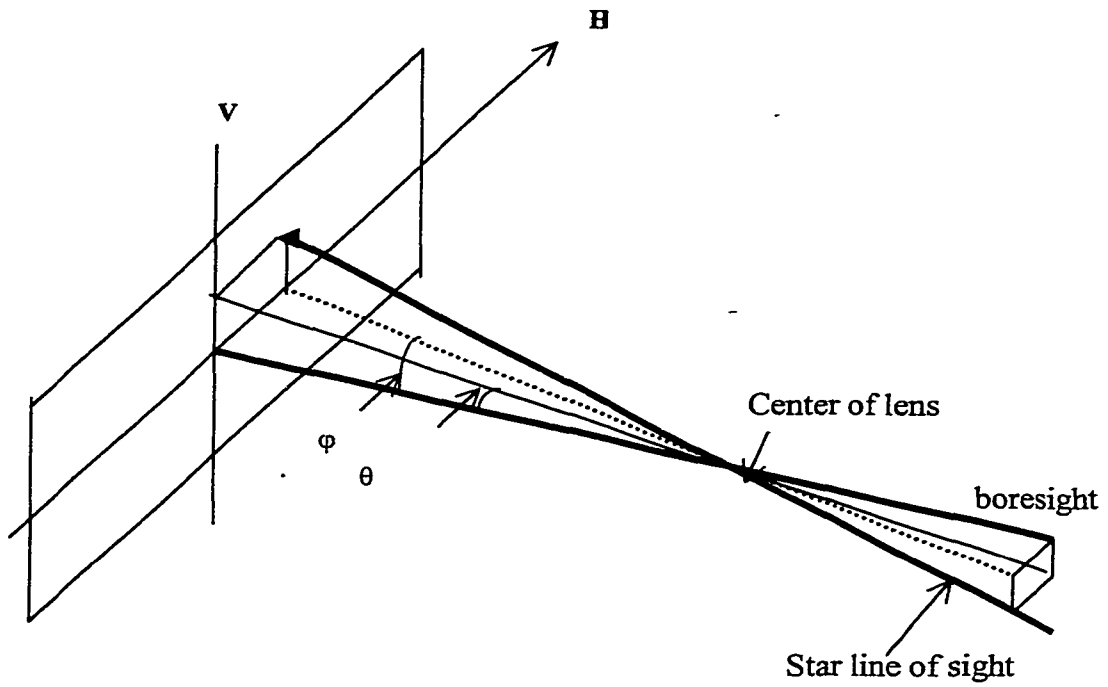

Figure 5.1.1 CCD camera coordinate system

In this figure, (H, V) are the coordinates of the centroid of the image with respect to the frame center. The angles φ and θ are the corresponding to centroid coordinates H and V of image. *b* is a length within the camera and takes a constant value. So

$$\tan\phi = \frac{H}{b} \qquad \text{(Eq. 5.1.1)}$$

and

$$\tan\theta = \frac{V}{b} \qquad \text{(Eq. 5.1.2)}$$

Suppose a balloon experiment uses a CCD cameras to measure a star's position vectors, and suppose the camera takes a picture of the star field every minute. That is, each picture frame gives a position vector of star at that time. To use the CCD camera's output data for MADS, we need to find the brightest stars in each picture frame, find the centroid of each bright star with respect to frame center, and the magnitude of each star.

5.2 FINDING BRIGHT SOURCES IN A CCD CAMERA FRAME

CCD chips are high quantum efficiency, linear detectors, and many balloon experiments use CCD cameras to record star fields. The data from a CCD frame are recorded in a one dimensional array with timetag. Each time is associated with a picture frame. The Kodak chip in the Apogoe camera records 512×768 +2 rows for each frame, and these can be represented in a 512×768 matrix from the first row of the first column to the last row of the last column. For each data frame, the first entry is the number of column in the matrix, and the second entry is number of row in the matrix. The 512×768 rows data are the elements of matrix.

Many stars are recorded in one frame. We need to find the 3-5 brightest stars from among them. Each star in the frame needs an associated star+sky window and an associated star box for the star image. When a star is focussed, its light falls on one pixel, but the camera is then defocused slightly so that the light falls on roughly 2×2 pixels. The motion of the balloon causes trailing of star images. The right choice of lens for the chip and chosen exposure time, usually ~1/10 second, allows the trail image to 2 pixels, giving an image size up to 4×4 pixels. Allow a margin of error for such problems as further

defocusing due to changes in the temperature of camera, and use a 6×6 star box for the star image. To include a sky 'annulus' around this star box, cut a star+sky window with final size 8×8 pixels.

For star identification, we don't use stars near an edge of the matrix, and since the size of star window is 8×8, we cut the data in first and last 8 rows; also we cut the data in first and last 8 columns. In a star+sky window, the count of pixel in the center with the biggest value.

To identify the brightest star, we first have to find the maximum value max_{1st} in the matrix with their row i_{1st} and column j_{1st} indexes. Then, from intersection of i_{1st} row and j_{1st} column as center we cut the 8×8 pixels from the frame chip as a star+sky window for the first brightest star. The star+sky window represents the region of the frame containing light from the brightest star. In the remaining frame matrix, we will find the second brightest star. To do this, we have to find the maximum value max_{2th} in the remaining matrix with their row i_{2th} and column j_{2th} indexes. Then, from intersection of i_{2th} row and j_{2th} column as center we also cut another 8×8 pixels as a star+sky window for the second star. In the remaining matrix, use the same way to find the third brightest star so on until we find the numbers of star required, for example, 3 to 5 brightest stars. It shows in Figure 5.2.1.

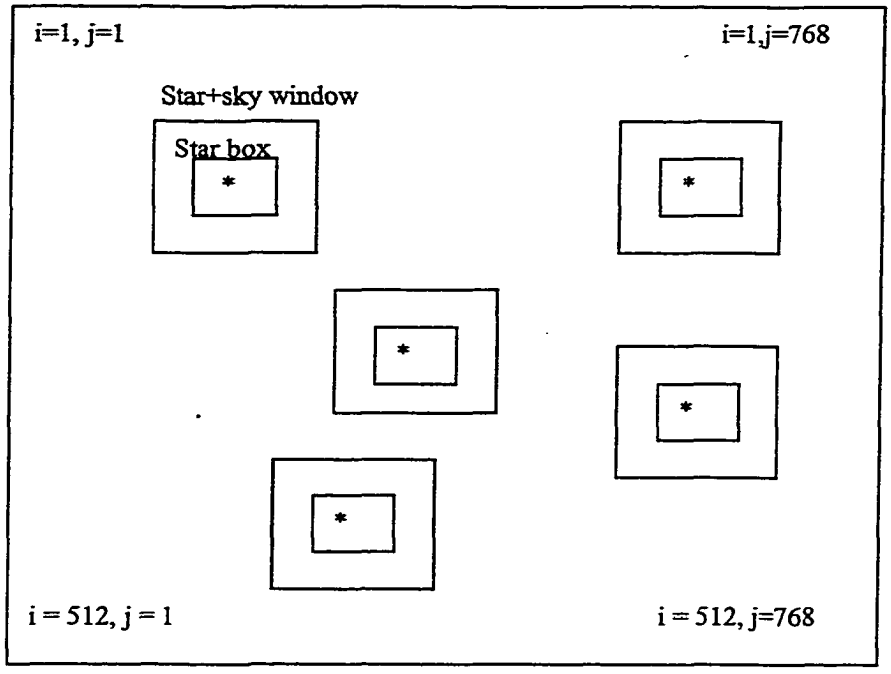

Figure 5.2.1 Stars+sky window and star box in a CCD frame

CCD frame has 512×768 pixels.
Star+sky window has 8×8 pixels.
Star box has 6×6 pixels.

To implement these idea, our program driveccd.m in part A of MADS for the camera data reads each camera frame data, then calls the program matrix.m to put them into the matrix. In this matrix, we do not use stars in the edge of the matrix, so we set the values that are in the edge of the matrix to be zero, since the values of element in the matrix are positive numbers. As we have discussed above, the size of a star+sky window is 8×8, so the number of rows and columns are to be the edge of the matrix are 8. In the matrix, the search for the maximum value of element with its index of row and column. It belongs to the brightest star in the frame. Then we choose 8×8 elements from the intersection of maximum value's row and column indexes. Once we identify the first brightest star, we set values in the star window of first brightest star to be zero, so when we do maximum value search for next brightest star in the frame, this star will not be selected again. The program will continue to select the brightest star in the frame until it reaches the given number of bright stars. For balloon experiments that use MADS to determine the attitude of balloon, we need to set the maximum number of brightest stars to be five. The program CCDsort.m does this task. The number of brightest stars in this program is a parameter that can be given by the user.

When the program finished the selection of the required number of brightest stars, we need to find the centroid of those brightest stars with respect to the center of frame.

5.3 FINDING THE CENTROID AND MAGNITUDE FOR A STAR

The technique for performing photometry using CCD frames makes the assumption that the frames have been "fully processed," which means they are assumed to be bias-subtracted, flattened, and cosmic-ray cleaned.

The principle of measuring the brightness of a star is to add up the counts in all the pixels that contain light from the star. Then we estimate the contribution to these pixels from the sky background using nearby pixels, and then subtract the sky contribution to get the net signal from the star. We must first determine the center of a stellar image in a CCD frame. This is of fundamental importance in astrometry. There are a variety of methods to perform this task. But for the balloon experiments, we are going to use an image centroiding method to find the centroid of the star since we know that the peak count is at the center of the star+sky window within the star+sky box. Hence, it makes sense to concentrate on these pixels nearest the center of the star box. This star box is in the star+sky window. The star+sky window is a submatrix of the frame matrix that only contains only one starwith size 8×8. So the size for the star box we choose is 6×6. The center of the box we initial as (i_b, j_b), where (i_b, j_b) is the row number and column number of brightest pixel.

The calculation of centroid for brightest star is described below.

(1) Calculation of marginal sums of box

We begin by computing the x and y marginal sums for the box. The x and y marginal sums, $\rho(x_i)$ and $\rho(y_j)$ are formed by the summing the pixels' counts at each columns in i rows or summing the pixels' counts at each rows in i column, respectively among the star box. Let 2a be the size

of star box, then a = 3 since the size of star box is 6×6. Suppose (i_b, j_b) represents the row number and column of the brightest star.

If I_{ij} is counts at pixel (i,j), then the marginal sums are defined as

$$\rho(x_i) = \sum_{j=-a}^{a} I_{ij} \quad \text{here i = –a : a,} \quad \text{(Eq. 5.3.1)}$$

$$\rho(y_j) = \sum_{i=-a}^{a} I_{ij} \quad \text{here j = –a : a.} \quad \text{(Eq. 5.3.2)}$$

Here, we can see that $\rho(x_i)$ is row marginal sums for row i, here i from –a to a, and $\rho(y_i)$ is column marginal sums for column j for j from –a to a, in the star box.

(2) Calculation of mean of counts for row and column marginal sums

Once we have the marginal sums for each row and each column, it is easy to computer the mean of the marginal sums over rows or columns. We define them as $\bar{x}$ and $\bar{y}$:

$$\bar{x} = \frac{1}{2a+1}\sum_{i=-a}^{a}\rho(x_i) \qquad \text{(Eq. 5.3.3)}$$

and

$$\bar{y} = \frac{1}{2a+1}\sum_{j=-a}^{a}\rho(y_j) \qquad \text{(Eq. 5.3.4)}$$

Here, the 2a is size of star box, , i from –a to a and j from –a to a, in the star box.

(3) Calculation of centroids of brightness in matrix frame

For the centroids of the brightest star, we will use the values calculated above:

$$x_c = \frac{\sum_{i=-a}^{a}(\rho(x_i)-\bar{x})x_i}{\sum_{i=-a}^{a}(\rho(x_i)-\bar{x}} \qquad \text{(Eq. 5.3.5)}$$

$$y_c = \frac{\sum_{j=-a}^{a}(\rho(y_j)-\bar{y})y_j}{\sum_{j=-a}^{a}(\rho(y_j)-\bar{y})} \qquad \text{(Eq. 5.3.6)}$$

(x_c , y_c) are the centroids of brightest star with respect to the center of CCD frame. They are the row and column numbers of the star's centroids in the frame. Here, $\rho(x_i)$, $\rho(y_j)$ are marginal sums in the star box. x_i and y_i take values from $-a : a$.

If (x_c , y_c), lies within two pixel of the (i_b , j_b), then we consider that this estimation is a good one. Otherwise, it is necessary to repeat the process with (x_c , y_c) as the new initial value (i_b , j_b). This method is an iterative process.

The centriod (x_c , y_c) represents in CCD frame now are row number and column number in frame matrix which has a size as 512 ×768. For the MADS to use, we need to represent the centroids in (H,V) coordinate.

(4) Present the centroids of brightest star in (H,V) coordinate

To use (H,V) coordinates known in terms of the row and column of frame matrix, we know that a frame matrix represents one CCD frame, so the center of HV coordinate system, or the origin (h_0,v_0) of coordinates (H, V), is at (786/2, 512/2). The matrix system and (H,V) coordinates are show in the Figure 5.3.1.

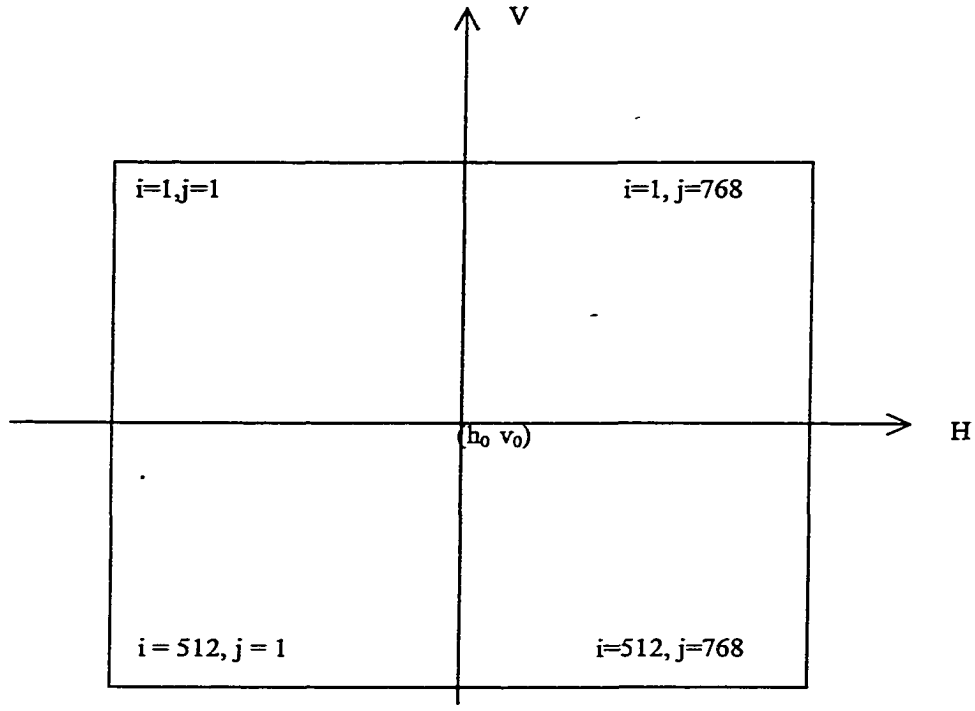

Figure 5.3.1 HV --Coordinate, in CCD frame

If (h_0,v_0) is the origin of the H, V-coordinates, then $(h_0,v_0) = (0,0)$ is in H, V coordinate system, but in the matrix of frame, h_0, is column number and v_0 row number for center of matrix frame. So $(h_0,v_0) = (768/2,512/2)$. Suppose (x_c , y_c) represents the centroids of brightest star in the frame matrix, (H_c , V_c) represents the centroids of brightest star in the H,V coordinate system. Then:

$$\text{Since} \quad h_0 = 768/2 \quad \text{and} \quad v_0 = 521/2$$

$$\text{So} \quad H_c = y_c - h_0 \quad \text{and} \quad V_c = v_0 - x_c \quad \text{(Eq. 5.3.7)}$$

Here (H_c , V_c) are the coordinates in the H,V system. The H_c is the horizontal coordinate and V_c is the vertical coordinate.

(5) Calculation of angles θ and φ

We have calculated the centroid of the brightest star and represent them by (H,V) coordinates. Since the format of CCD camera data for input to MADS use θ and φ to represent the centroid of the star, we need to calculate angles θ, φ from the HV- coordinates of the centroid of the star. The CCD camera coordinate system is shown in Figure 5.1.1. From it we can see θ and φ, corresponding to V and H respectively, are the angles from the star' line-of-sight, as explained in chapter 5.1

By Eq. 5.1.1 and Eq. 5.1.2, we calculate θ and φ as:

$$\theta = \arctan(\frac{V_c}{b}) \qquad \text{(Eq. 5.3.8)}$$

$$\phi = \arctan(\frac{H_c}{b}) \qquad \text{(Eq. 5.3.9)}$$

Here b is a distance between camera lens and camera chip. The H_c is horizontal coordinate of centroid for the star and V_c is vertical coordinate of centroid for the star.

(6) Calculation of the mode for sky background

The sky background is the signal that would be in the aperture if the star of interest were not there. To determine the sky background, the usual procedure is to look at the signal in an annular region centered on the star or a symmetrically placed region centered on the star. The size of star+sky window is 8×8. The annular region is between the star+sky window and star box. This region includes pixels in the star+sky window but not in the star box. These pixels are used to calculate the mode for the star+sky window. The star+sky window is shown in Figure 5.3.2.

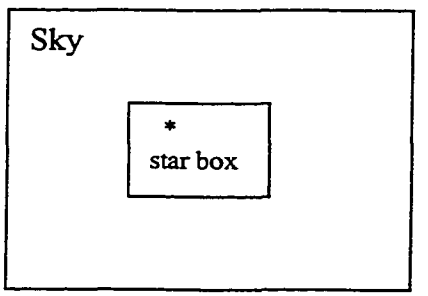

Figure 5.3.2 Star+sky window

Suppose the annulus includes the pixels in the star+sky window but not in the box. With the pixels in this annulus, we first calculate the mean and then the median of the values of these pixels. Let ANNU ={ counts of pixels in the annulus}. If the counts from pixels in the annulus

would have a Gaussian form, the appropriate value for the sky background would be the mean of the distribution. A mode is calculated to compensate for the fact that some light from another star or nebulosity might be included.The mode is estimated from the formula (Kendall, Stuart, 1977):

median = median(ANNU)

mean = mean(ANNU)

Then mode = 3 * median-2 * mean. (Eq. 5.3.10)

Here mode represents the sky background.

(7) Calculation of the background intensity

Since we have choose size of star+sky window is 8×8, we can calculate the background intensity for a star. In the star+sky window, there are 8x8 elements whose distance from the (x_b, y_b) is less than 4 pixels in the CCD frame matrix. These pixels represent the signal for this star. We calculate the intensity of the star, based on those pixels. The formula is showed as below:

$$I = \sum I_{ij} - n_{pix} * i_{sky} \qquad \text{(Eq. 5.3.11)}$$

here $\sum I_{ij}$ is summation that carried out over the pixels in star+sky window. So,

$\sum I_{ij} = \sum_{i=}^{8} \sum_{j=1}^{8} I_{ij}$ Here I_{ij} is counts of pixel or signal of star in its field.

The n_{pix} is number of pixels in the star+sky window. So

$$n_{pix} = 8 * 8 = 64$$

The i_{sky} here is the background sky value which is the mode that we calculated at step 8). So i_{sky} = mode or this star intensity, our final calculation is for the star magnitudes.

(8) Calculation of magnitudes of brightest star

The magnitudes of a star is used in the MADS as one parameter of the CCD data for the attitude determination of balloon. Its calculation is basis on the intensity of star and an zero point number `zpt` found by calibrating the camera. This value is typical in the rang 23.5 to 25.0.

The magnitude of a star we is calculated as

$$m = zpt - 2.5 * \log I$$

where m is the magnitude of a star, and I is intensity of star calculated in part 7).

We have discussed the method for the calculation of centroid, φ, θ, and the intensity of a star in the star window. A CCD camera frame includes many stars, and we have picked a maximum of five stars in the frame. We have to find a centroid, φ, θ, and intensity for each star that we picked up in the frame. The program for this algorithm repeat this processes to find the centroid, φ, θ, and intensity for each star we have picked up in the frame.

When CCD picture frames are taken over a time period, the program will repeat the calculations for each frame, then put the results into a data file with the format and time tag needed by MADS.

This CCD image center determination algorithm is used to find the centroid of the brightest stars in one CCD frame at one point in time. Each CCD frame is at different time. It can find as many of the brightest stars as many as the user requires for from each frame. This algorithm is different with others because it finds more than one brightest star in the CCD frame and calculate the centroid and magnitude for each star in the CCD frame. This algorithm also converts the star's centroid from HV-coordinates to angles φ

and θ. The final output from the CCD camera data processes will be in format (time, φ, θ, temperature, magnitude, flag). The temperature is the camera temperature, measured by a sensor attached to it. The flag is a value for data quality, zero indicating that the data are good, nonzero indicating a basic problem.

5.4 RELATED RESEARCH

Our method of calculating centroid and magnitudes are for unmagnified star images. There is also other research about the zero point parameter zpt, and calculation of magnitude and estimate of the size of star+sky window.

In the general case, the size of the star+sky window will be much bigger if the camera is attached to a telescope.

Writers on basic photometry techniques discuss this problem from a consideration of the errors involved in star+sky window for aperture photometry (e.g., Newberry, 1991). Assume that a star window with radius R contains light of total intensity I_{st} from a star. Then the number of photoelectrons generated is just $I_{st}D$, D is star sky. Photons follow a Poisson distribution, so the standard deviation associated with signal $I_{st}D$ photons in star +sky window is just $(I_{st}D)^{1/2}$. The magnitude error is then

$$m \pm \delta m_1 = zpt - 2.5\log(I_{st} \pm \sigma(I_{st}))$$

and $$\delta m_1 = 1.09\frac{\sigma(I_{st})}{I_{st}} = \frac{1.09}{I_{st}D}(I_{st}D + r^2 n_{pix})^{1/2}$$

where r is the readout noise, and n_{pix} is the number of pixels include in the star window. The error in the magnitude decreases with increasing star window size, since I_{st} increases with increasing star window radius. However, this proposition is not true

indefinitely because once the star window includes "almost all the pixels," δm_1 remains constant or may actually increase due to the readout noise contribution of the extra pixels.

Another source of noise arises, as follows: the sky background i_{sky} is presumably well determined since it is derived from a large number of pixels. But the actual sky signal in the star window is subject to Poisson statistics in the same way as the intensity. So this source also a ffects the magnitude as:

$$\delta m_2 = \frac{1.09}{I_{st}} \left(\frac{n_{pix} i_{sky}}{D}\right)^{1/2}$$

In this equation, the δm_2 increases with increasing star window size, and this error is clearly minimized for a small star window.

The combined error is then:

$$\delta m = \frac{1.09}{I_{st} D} \left(I_{st} D + n_{pix} (r^2 + i_{sky})\right)^{1/2}$$

This equation indicates that there are two limiting cases:

(a) "bright" stars for which $I_{st}D >> npix(r^2+i_{sky}D)$. In this situation, larger star+sky windows that contain "all" the light are acceptable.

(b) "faint" stars for which $I_{st}D$ does not dominate $n_{pix}(r^2+i_{sky})$. In this situation, small star+sky windows, which will not contain all the light, are preferred.

Other parameters, such as *a*, the size of star box, which is a constant, we have discussed when we calculated them.

For the image center determination, there is another technique called the marginal sum method (Auer & van Altena 1978, Stetson 1979). This method first extracts from the CCD frame a star window centered on an initial guess for the center of the star. The size

of star window has to be large enough so that it contains not only the star of interest but also enough pixels to allow an estimation of the sky background. Usually the size will be 5 times the full width at half maximum of the image. From this star box calculate the x and y marginal sums:

$$\rho(x_i) = \sum_j I_{ij} \qquad \text{and} \qquad \rho(y_j) = \sum_i I_{ij}$$

here i and j are row and column numbers of star window. If a one-dimensional function, such as the Gaussia, is then fitted to each of these marginal sums, a estimate of the image center in each coordinate will result. If the star of interest is isolated, then the center determined by this method will be perfectly adequate for placing at the center of the measurement aperture. But if the star has nearby companions that fall within the star window, then the derived center is likely to be invalid because background, center, width, and height fits are rather easily biased by the presence of nearby companions.

So in our algorithm, we used the image centroiding method which makes use of the fact that the star of interest is presumably near the center of the star window.

5.5 TESTING FOR ALGORITHM OF CENTROID OF STAR

The algorithm we use for the calculation of the centroid of star and intensity of star are used for pre-processing the CCD camera data. Then the CCD data can be used by MADS to determine the attitude of the telescope.

We have used data from an Apogee CCD camera to test the programs. This data set includes 18 frames. We pick up three brightest stars in each frame, and calculate their centroids, the angles φ, θ, and intensity, then put them in to a file with the format required for MADS. Table 5.5.1 shows the CCD camera data processed using the centroiding algorithm we have developed for the CCD camera data. This table has data for four frames of data, with the three brightest stars picked out in each frame. The data are in a format ready for MADS to use.

Figure 5.5.1. shows horizontal coordinate vs. vertical coordinate for stars in each frame at a point in time. (18 frames). From the graph, we can see the star has moved relative to the camera in 18 frames.

Figure 5.5.2 shows the three brightest stars in one frame we can see them in different positions in the frame.

Table 5.5.1 Processed CCD camera data

Time	phi	theta	Temp	intensity	flag
9.9120000104831904e+05	1.46727	-1.50915	2.0	1.22511	0.0
9.9120000104831904e+05	-1.53619	-1.52728	2.0	6.43600	0.0
9.9120000104831904e+05	-1.54269	-1.36164	2.0	9.77999	0.0
9.9120000115376594e+05	1.47882	-1.49800	2.0	1.46739	0.0
9.9120000115376594e+05	-1.53994	-1.38932	2.0	6.39199	0.0
9.9120000115376594e+05	1.52175	-7.29974	2.0	6.78800	0.0
9.9120000125932298e+05	1.48812	-1.48187	2.0	7.06159	0.0
9.9120000125932298e+05	1.54368	1.51215	2.0	-8.72799	0.0
9.9120000125932298e+05	-1.54212	-1.52258	2.0	-6.03600	0.0
9.9120000140498998e+05	1.49584	-1.45663	2.0	1.17479	0.0
9.9120000140498998e+05	1.54123	-1.52924	2.0	1.37279	0.0
9.9120000140498998e+05	-1.28617	-1.52246	2.0	-5.88800	0.0

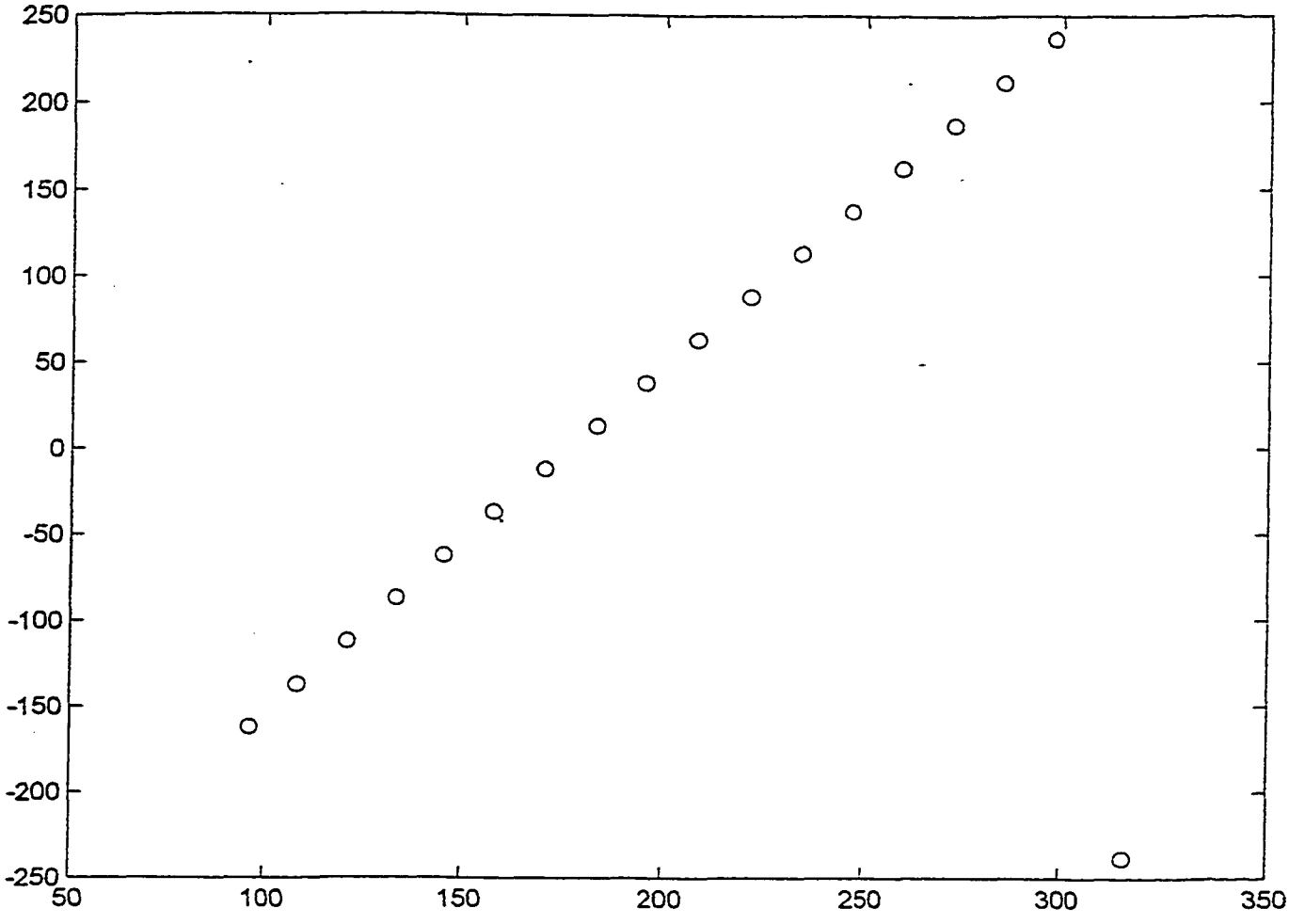

Figure 5.5.1 Star tracker : horizontal coordinates vs. vertical coordinates

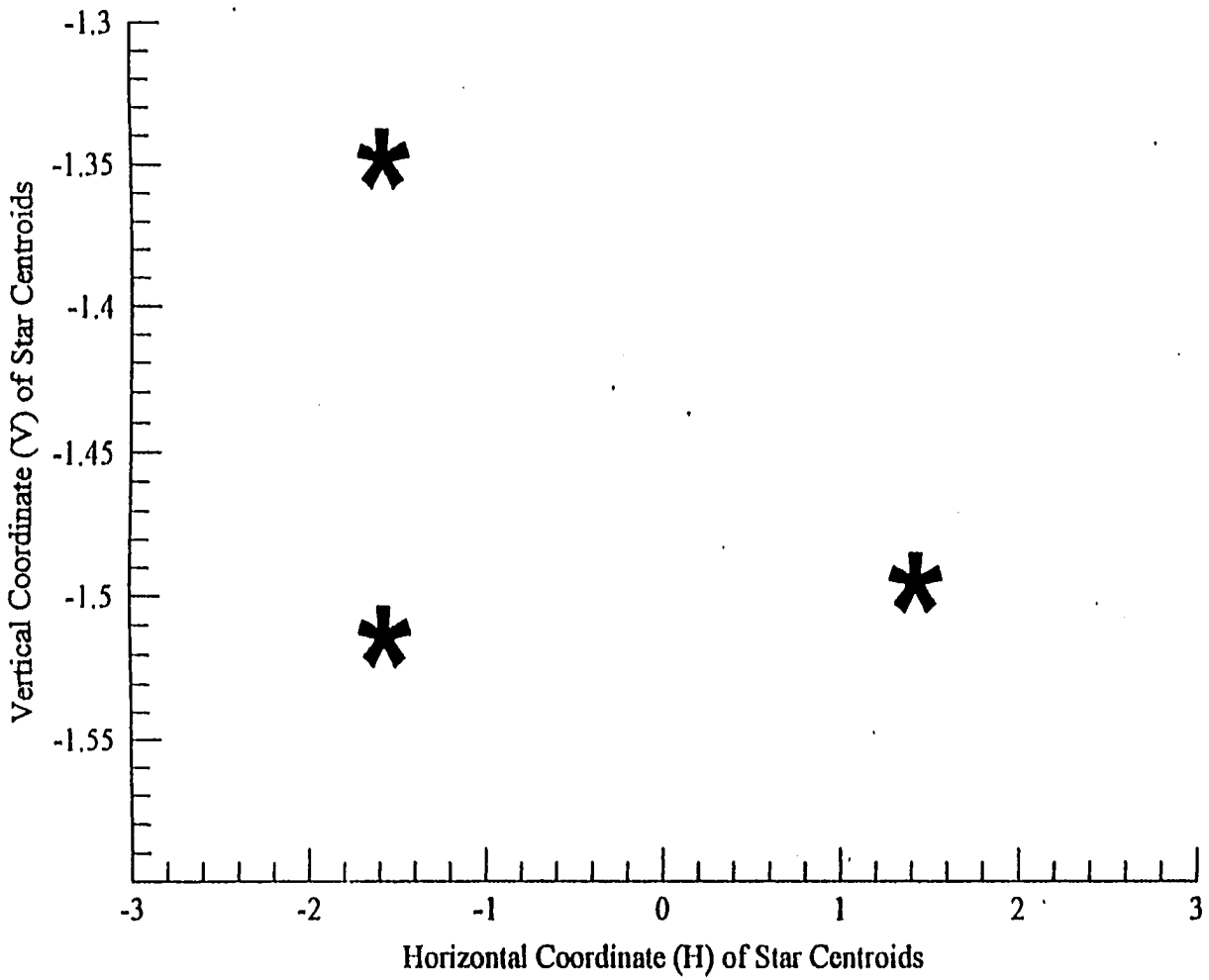

Figure 5.5.2. Brightest stars in a CCD frame.

CHAPTER 6

MODELING MOTION OF BALLOON IN MADS

MADS is used in data analysis for balloon experiments. It uses the star vectors found from the CCD data, and GPS data, gyro vectors, and initial attitude quaternions to find the attitude quaternions of balloon telescope during the balloon flight time. This chapter will discusses the modules that we used or modified in the MADS.

6.1 REPLACING EPHEMERIS BY GPS IN MADS

The GPS data are used to find the position and velocity of the balloon. The position and velocity are used in the calculation of the observed vector and reference vector of a star in preparation for identifying the star by comparison with a catalog. The position vectors and velocity vectors of balloon are in the geocentric coordinate system.

In part A of MADS, the GPS data have been translated into geocentric coordinates which is GCI coordinates system. In the part B of MADS, the velocity vector of the balloon is calculated from the position vectors at each time.

Let $(x_{i-1}, y_{i-1}, z_{i-1})$ and (x_i, y_i, z_i) be position vectors of balloon at times t_{i-1} and t_i. The position vectors of balloon have been calculated in part A of MADS, as discussed in chapter 4. Suppose (v_x, v_y, v_z) is a velocity vector of balloon at time t_i in geocentric coordinates. Then

$$v_x = \frac{x_i - x_{i-1}}{t_i - t_{i-1}} \qquad \text{(Eq.6.1.1)}$$

$$v_y = \frac{y_i - y_{i-1}}{t_i - t_{i-1}} \qquad \text{(Eq.6.1.2)}$$

$$v_z = \frac{z_i - z_{i-1}}{t_i - t_{i-1}} \qquad \text{(Eq.6.1.3)}$$

Since the balloon flight is for a period of time, and the velocity of balloon changes during the flight time, we have to calculate the velocity of balloon in each time interval during the flight. These velocity vectors of the balloon are used to correct the reference vectors of stars when we calculate these.

6.2 OBSERVED AND REFERENCE VECTORS OF STAR

The observation vectors and reference vectors of stars are used to determine the attitude quaternion for the balloon telescope. The attitude quaternion represents the pointing and rotation of the telescope in geocentric coordinates. The observation vector represents the star's position with respect to the camera, which has coordinates in a balloon body coordinate system, and the reference vector represents the star's position with respect to the Earth's center, which is the origin of the geocentric coordinate system. The figure 6.2.1 shows observation and reference vector of a star.

As discussed in the chapter 5, the CCD cameras take pictures of stars during the balloon flight. Using CCD data, we first calculated the centroids of star with respect to the frame centers and the intensity of each star in the frame for all the frames taken over the period of balloon flight time. This data analysis has been finished in part A of MADS using the centroid and intensity of each star, MADS calculates the observation vector of

the star with respect to the camera (which is the balloon body coordinate system) in the data adjuster' processes. Then it converts the observed vector from the balloon body coordinate system to the geocentric coordinate system. The reference vector of the star is generated during the star identification process. We discuss them in the following sections.

1. Calculate observation vector with respect to balloon body coordinate system

The observation unit vector W_s in the balloon coordinates system is a 1x3 vector. Since we have determined the centroid of a star and its representation by ϕ, θ in the chapter 5, the CCD-measured star unit vector $W_{s'}$ in CCD coordinate system is defined as (V. Johnson and M. Woodard, 1995):

$$\hat{W}_{s'} = \begin{bmatrix} -\tan\theta \\ \tan\phi \\ 1 \end{bmatrix} w_{s'}^{-1} \qquad \text{(Eq. 6.2.1)}$$

where

$$w_{s'} = (1 + \tan^2\theta + \tan^2\phi)^{\frac{1}{2}} \qquad \text{(Eq. 6.2.2)}$$

Let $m_{ss'}$ be the transformation matrix from the CCD camera coordinate system to balloon body system. It defined as (V. Johson/M. Woodard 1994):

$$m_{ss'} = \begin{bmatrix} \hat{X}\cdot\hat{X}' & \hat{Y}\cdot\hat{X}' & \hat{Z}\cdot\hat{X}' \\ \hat{X}\cdot\hat{Y}' & \hat{Y}\cdot\hat{Y}' & \hat{Z}\cdot\hat{Y}' \\ \hat{X}\cdot\hat{Z}' & \hat{Y}\cdot\hat{Z}' & \hat{Z}\cdot\hat{Z}' \end{bmatrix} \qquad \text{(Eq. 6.2.3)}$$

where the $\hat{X},\hat{Y},\hat{Z}$ are unit vectors in the balloon body coordinate system and $\hat{X}',\hat{Y}',\hat{Z}'$ are unit vectors in the CCD coordinate system. These unit vectors

are determined when the CCD is fixed in the balloon and balloon body coordinate system is fixed.

The star observation vector in balloon coordinate system is then:

$$\hat{W}_s = m_{ss'}\hat{W}_{s'} \qquad \text{(Eq. 6.2.4)}$$

2. Find the reference vectors of star by star identification.

MADS uses direct match and doublet match methods to do the star identification. These two methods have discussed in the chapter 3. The observed vector of a star is in the balloon body coordinate system and converted to a vector in the geocentric coordinate system during the star identification process. Figure 6.2.1 shows the observed and reference vectors of star. The basic idea of star identification is comparing a observed star's vector with a candidate reference star's vector. The candidate reference star is picked up from SKYMAP. If they match, then the vector of the candidate reference star will be assigned to the matching star as its reference vector. The velocity aberration correction will be performed on the position vector of the reference star, by using the velocity of the balloon calculated from the processed GPS data. The calculation of the velocity of balloon is performed in calling to rungps.m, rdgps.m and calvel.m programs in MADS. Again, this reference vector of star is in the geocentric coordinate system.

So far, we know the observed vector of the star in balloon body coordinate system and the reference vector of the star in geocentric coordinate system. MADS uses these vectors with gyro data, and a initial attitude quatenion at the time when the balloon has been stabilized, to calculate the attitude quaternion for the balloon during the flight time.

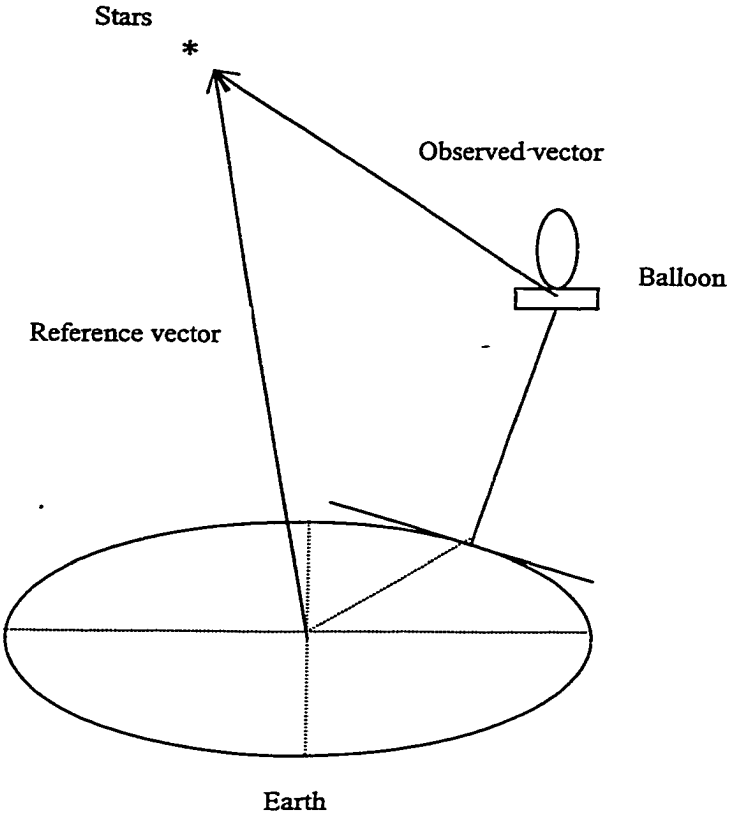

Figure 6.2.1 Observed and reference vectors of star

6.3 PRINCIPLE OF ATTITUDE DETERMINATION IN MADS

A frame of data is defined to be a timetag and a set of observation vectors and reference vectors associated with that time. The observation vectors are in the balloon body coordinate system and reference vectors are in the geocentric coordinate system. MADS uses a quaternion estimation algorithm (Shuster and Oh, 1981; Wertz, 1985) to estimate the attitude quaternion from observed and reference vectors for the stars in a single camera frame. The observed vectors and reference vectors are showed in the figure 6.2.1(page 98). The description of estimation of the attitude quaternion using identified stars is as follows:

For any given time t, $\hat{W}_i$ is an observation unit vector in the balloon body coordinate system, and $\hat{V}_i$ is a reference unit vector in geocentric coordinate system. A is a rotation matrix that transforms vectors from the geocentric to the body coordinate system. A is also an attitude matrix as we have discussed in chapter 2.2. A loss function $l(A)$ (M. Lambertson, J. Keat 1994) is defined as

$$l(A) = \frac{1}{2}\sum_{i=1}^{n} a_i \left| \hat{W}_i - A\hat{V}_i \right|^2 \qquad \text{(Eq. 6.3.1)}$$

where $$a_i = \frac{\sigma_{tot}^{\ 2}}{\sigma_i^{\ 2}} \qquad \text{(Eq. 6.3.2)}$$

$$\sigma_i^{\ 2} = \sigma_{W_i}^{\ 2} + \sigma_{V_i}^{\ 2} \qquad \text{(Eq. 6.3.3)}$$

$$\frac{1}{\sigma_{tot}^{\ 2}} = \sum_{i=1}^{n} \frac{1}{\sigma_i^{\ 2}} \qquad \text{(Eq. 6.3.4)}$$

$$\sum_{i=1}^{n} a_i = 1 \qquad \text{(Eq. 6.3.5)}$$

n is number of CCD canera. The loss function $l(A)$ is a function of attitude matrix. The optional attitude matrix A_0 is the matrix that minimizes $l(A)$. Alternatively, it maximizes the gain function $g(A)$, where

$$g(A) = 1 - l(A)$$

From the Eq. 6.3.1, the $g(A)$ is represented as

$$g(A) = \sum_{i=1}^{n} a_i \hat{W}_i^T A \hat{V}_i \qquad \text{(Eq. 6.3.6)}$$

Since A is the attitude matrix, as we have discussed in chapter 2, it can been represented in terms of attitude quaternions. From Eq. 2.2.19, expressing A in terms of quaternions:

$$A(\bar{q}) = (q^2 - \bar{Q} \bullet \bar{Q})[I]_3 + 2\bar{Q}\bar{Q}^T - 2q\bar{Q}^* \qquad \text{(Eq. 6.3.7)}$$

Where $\bar{q} = \begin{bmatrix} \bar{Q} \\ q \end{bmatrix}$, $\bar{Q} = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix}$, and $\bar{Q}^* = \begin{bmatrix} 0 & -Q_3 & Q_2 \\ Q_3 & 0 & -Q_1 \\ Q_2 & Q_1 & 0 \end{bmatrix}$ (Eq. 6.3.8)

Then $g(A)$ is expressed in terms of quaternions by substituting Eq. 6.3.7 into Eq.6.3.6

$$g(\bar{q}) = \sigma(q^2 - \bar{Q} \bullet \bar{Q}) + \bar{Q}^T S \bar{Q} + 2q\bar{Q} \bullet \bar{Z} \qquad \text{(Eq. 6.3.9)}$$

$$\sigma = \sum_{i=1}^{n} a_i \hat{W}_i \bullet \hat{V}_i \qquad \text{(Eq. 6.3.10)}$$

$$\bar{Z} = \sum_{i=1}^{n} a_i \hat{W}_i \times \hat{V}_i \qquad \text{(Eq. 6.3.11)}$$

$$S = \sum_{i=1}^{n} a_i (\hat{W}_i \hat{V}_i^T + \hat{V}_i \hat{W}_i^T) \qquad \text{(Eq. 6.3.12)}$$

For the optimal attitude quaternions, we write Eq. 6.3.9 in matrix form:

$$g(\bar{q}) = \bar{q}^T K \bar{q} \qquad \text{(Eq. 6.3.13)}$$

where $$K = \begin{bmatrix} S - \sigma[I]_3 & \bar{Z} \\ \bar{Z}^T & \sigma \end{bmatrix} \qquad \text{(Eq. 6.3.14)}$$

Then the problem of obtaining the optimal attitude estimate can be reduced to finding the quaternion, $\bar{q}_{opt}$, which maximizes $g(\bar{q})$ with the quaternion normalization constraint

$$\bar{q}^T \bar{q} = 1 \qquad \text{(Eq. 6.3.15)}$$

Use the method of Lagrange multipliers to derive a new gain function, $g'(\bar{q})$ which can be maximized without constraint, as

$$g'(\bar{q}) = q^T K q - \lambda q^T q \qquad \text{(Eq. 6.3.16)}$$

where λ is chosen to satisfy the constraint ($\bar{q}^T \bar{q} = 1$). A stationary value of $g'(\bar{q})$ is obtained when

$$\frac{\partial g'(\bar{q})}{\partial \bar{q}^T} = 0 \qquad \text{(Eq. 6.3.17)}$$

that is

$$K\bar{q} = \lambda \bar{q} \qquad \text{(Eq. 6.3.18)}$$

From the Eq. 6.3.13, we have

$$g(\bar{q}) = \bar{q}^T K \bar{q} = \bar{q}^T \lambda \bar{q} = \bar{q}^T \bar{q}\, \lambda = \lambda \qquad \text{(Eq. 6.3.19)}$$

Therefore, $g(\bar{q})$ is maximum when λ is maximum. The corresponding quaternion is the optimum quaternion $\bar{q}_{opt}$. In other words, the quaternion is the eigenvector corresponding to the maximum positive eigenvalue of K.

This optimum quaternion $\overline{q}_{opt}$ is the estimated quaternion obtained from the observation star vector and reference star vector at this time. MADS then uses gyro information to propagate the previous quaternion from the previous time to this time. Then it uses the estimated quaternion and the quaternion that has been propagated using gyro data and the least squares methods to find the best quaternion for the telescope at this time. This function is performed in the Extended Kalman Filter process.

6.4 PROPAGATION OF QUATERNIONS

The sensors for MADS consist of two CCD cameras and a three-axis gyro. These data are assumed to be independent and are individually timetagged. For the generation of the initial quaternion at the time when the balloon has been stabilized, we only use gyro data to propagate the ground-level initial quaternion at the time balloon is launched to the time the balloon has stabilized. After this time, MADS uses observed vectors and reference vectors to calculate an estimated attitude quaternion at time t_i for each camera, and propagates a previous quaternion at time t_{i-1} to this time t_i using gyro data. Then we use these three quaternions in the least square method to determine the attitude quaternion of each balloon telescope at the time t_i. The propagation of quaternion is based on the gyro data over the time internal (t_{i-1}, t_i). With continuous gyro data, the estimated quaternion at time t_i can be propagated to time t_n based on the following module:

$$\overline{q}(t_n) = D(t_n, t_{n-1})D(t_{n-1}, t_{n-2}).....D(t_2, t_1)\overline{q}(t_1) \qquad \text{(Eq. 6.4.1)}$$

where, in the time interval (t_j, t_{j-1}),

$$D(t_j,t_{j-1})=\begin{bmatrix} C_j & k_{jz}S_j & -k_{jy}S_j & k_{jx}S_j \\ -k_{jz}S_j & C_j & k_{jx}S_j & k_{jy}S_j \\ k_{jy}S_j & -k_{jx}S_j & C_j & k_{jz}S_j \\ -k_{jx}S_j & -k_{jy}S_j & -k_{jz}S_j & C_j \end{bmatrix} . \qquad \text{(Eq. 6.4.2)}$$

and

$$C_j=\cos[\frac{\omega_j}{2}(t_j-t_{j-1})] \qquad \text{(Eq. 6.3.3)}$$

$$S_j=\sin[\frac{\omega_j}{2}(t_j-t_{j-1})] \qquad \text{(Eq. 6.3.4)}$$

and $k_{jx}=[\frac{\omega_{jx}}{\omega_j}]$, $k_{jy}=[\frac{\omega_{jy}}{\omega_j}]$, $k_{jz}=[\frac{\omega_{jz}}{\omega_j}]$, and $\omega_j=|\overline{\omega}_j|$, the average angular rate $\overline{\omega}_j$ is in the interval (t_j,t_{j-1}) which is computed from gyro data.

Since the quaternions represent the attitude of telescope at time t_i, the quaternions are functions of time. In other words, at any time during the balloon flight, there is an attitude quaternion of the telescope. This quaternion at any particular time is found by the least-squares method in MADS.

With each CCD camera single frame, an estimated quaternion is determined from the observation vectors and reference vectors of identified stars and each CCD camera has a timetag t_i, which is the time for the frame. So this quaternion is estimated at time t_i. To get the best quaternion, MADS will use gyro data to propagate the previous quaternion at time t_{i-1} to t_i. This procedure provides a second quaternion at time t_i. If the CCD camera frame is the first frame, the propagation is done with an input initial quaternion. We assume there are only two kinds of sensors data; CCD camera data and gyro data, in a balloon experiment. So there are only three quaternions at the time t_i one

from each of the camera and one from the gyro. Then the least-squares method is used on these three quaternions to find the best quaternion at the time t_i. MADS also save this quaternion into an attitude history file with timetag t_i. It then becomes the previous quaternion for the time t_{i+1}.

The gyro data are used for propagating quaternion from the time to time by using the module Eq. 6.4.1. This propagation depends only on the previous quaternion and gyro data. In this attitude determination system, a ground-level initial quaternion is calculated before launch.

To use MADS, we first propagate the ground-level initial quaternion at the time when the balloon is launched to the time right before the balloon is steady and the cameras start to take the useful pictures of stars. This propagated quaternion becomes the initial quaternion for later use and will itself be propagated for the first frame.

We employ this propagation method to propagate the previous quaternion from frame to frame in order to use the least-squares method with the estimated quaternion at frame time, and find the best (or final) quaternion for each time within the time span of the CCD camera frames.

This propagation method also predicts the quaternion at a time between camera frames. If there is a previous quaternion is at the time t_i, and next frame is at time t_{i+1}, we can find a quaternion between the camera frame at t_i and the camera frame at t_{i+1}, i.e within the time interval (t_i , t_{i+1}). This quaternion is determined without using CCD camera data.

6.5 TESTING MADS

We have tested MADS using data from a satellite mission. NASA's requirement for a new software to be considered 'operational' is that it run using data files from another mission. We have been supplied with files from the Rosi X-ray Timing Explorer satellite. In our testing of MADS, we used fhst1_raw.data, fhst2_raw.data from the fixed head star trackers as preprocessed data from the two cameras. We also use the file gyro.data as three axis gyro data and a file mat1.data data from the RXTE ephemeris as the GPS data. For an initial quaternion, we pick up one from the onboard computer attitude history file within the gyro time interval. The processes for running MADS are described detail in the Readme file and RUN_mads file. The Readme file and RUN_mads are in the appendix A.

The output attached in Figure 6.5.1 gives us the attitude quaternion at the end time of the star trackerdata. The first part of this output is the mean and stand deviation for the observed star' s vectors taken from two cameras. The second part of the output gives us the attitude quaternion at the end time of star tracker data and also shows the bias of gyro.

Since the attitude quaternion of the balloon telescope is a function of time, it means that for any time during a balloon flight, there is an attitude quaternion for the telescope. MADS save these attitude quaternions into a separate quaternion file so that the user can late do the analysis of determining the pointing and rotation of the telescope at any time during a balloon flight.

This quaternion file includes five columns. Table 6.5.1, showa the quaternion file we got when we tested MADS. The data in the first column are the timetags. The second

through fifth columns contain the quaternion components q_1 through q_4. Since this data file is too big, we show only the first ten and the last timetags and quaternions in the file.

Table 6.5.1 Attitude quaternion of balloon telescope from MADS

Time	q_1	q_2	q_3	q_4
9.6082301000274997e+05	-4.5705646e-001	-5.8051145e-001	3.3357371e-001	5.8552065e-001
9.6082301000374998e+05	-4.5706180e-001	-5.8050559e-001	3.3357601e-001	5.8552098e-001
9.6082301000474999e+05	-4.5707254e-001	-5.8049480e-001	3.3358508e-001	5.8551812e-001
9.6082301000574999e+05	-4.5707164e-001	-5.8049586e-001	3.3357996e-001	5.8552069e-001
9.6082301000675000e+05	-4.5706775e-001	-5.8049997e-001	3.3357202e-001	5.8552418e-001
9.6082301000912802e+05	-4.5706438e-001	-5.8050341e-001	3.3356622e-001	5.8552671e-001
9.6082301001012803e+05	-4.5706205e-001	-5.8050534e-001	3.3356280e-001	5.8552856e-001
9.6082301001112803e+05	-4.5706007e-001	-5.8050708e-001	3.3356146e-001	5.8552915e-001
9.6082301001212804e+05	-4.5705859e-001	-5.8050647e-001	3.3355822e-001	5.8553275e-001
9.6082301001312805e+05	-4.5705825e-001	-5.8050629e-001	3.3355854e-001	5.8553301e-001
9.6082302180512797e+05	-4.5705590e-001	-5.8050268e-001	3.3354124e-001	5.8554827e-001
9.6082302180612797e+05	-4.5705442e-001	-5.8050417e-001	3.3354011e-001	5.8554860e-001
9.6082302180712798e+05	-4.5705415e-001	-5.8050423e-001	3.3354055e-001	5.8554850e-001
9.6082302180812799e+05	-4.5705079e-001	-5.8050785e-001	3.3353620e-001	5.8555000e-001
9.6082302180912800e+05	-4.5704898e-001	-5.8051008e-001	3.3353367e-001	5.8555064e-001
9.6082302181012800e+05	-4.5705422e-001	-5.8050564e-001	3.3354370e-001	5.8554525e-001
9.6082302181112801e+05	-4.5705817e-001	-5.8050307e-001	3.3355694e-001	5.8553717e-001
9.6082302181212802e+05	-4.5706466e-001	-5.8049672e-001	3.3356058e-001	5.8553633e-001
9.6082302181312803e+05	-4.5706950e-001	-5.8049212e-001	3.3356159e-001	5.8553653e-001
9.6082302181412803e+05	-4.5707220e-001	-5.8048971e-001	3.3356092e-001	5.8553720e-001

Sensor	#obs	mean-x	mean-y	mean-z	std-x	std-y	std-z
FHST#1 (deg)	752	-0.00000	0.00011	0.00092	0.00091	0.01348	0.00608
FHST#2 (deg)	830	-0.00026	0.00043	-0.00107	0.00025	0.00145	0.00122

Epoch Time: 960823.021814128

Quaternion: -0.45707220 -0.58048971 0.33356092 0.58553720

Gyro Bias (deg/sec): -0.00000431 -0.00001152 0.00001337

Figure 6.5.1 Quaternion output from MADS

6.6 CONCLUSIONS

MADS is a multimission attitude determination system, used for the data analysis for balloon experiments. The basic sensors for this attitude determination consist of two CCD camera and a three-axis gyro. Then with GPS data for the balloon position, MADS will calculate the attitude quaternion of the telescope at any time during the balloon flight.

The attitude quaternion is represented as a vector $(e_x \sin\frac{\theta}{2}, e_y \sin\frac{\theta}{2}, e_z \sin\frac{\theta}{2}, \cos\frac{\theta}{2})$. The vector (e_x, e_y, e_z) is a unit vector in the geocentric coordinate system, and θ is rotation angle about this vector. The attitude quaternions are a function of time, which means that for any timetag of a CCD frame, there is a quaternion at the same time. These attitude quaternions are saved as an attitude history file, and the user can find the attitude quaternion for the telescope at any time within the time span of CCD camera frames.

To find the attitude of telescope within the time from one camera frame to next, MADS provides a method of propagating the attitude quaternion using gyro information from the previous timetag to the desired time so that the user can find the attitude quaternion at any user required time within the gyro time span.

This attitude determination system will be useful to the NASA balloon group and other university balloon groups.

We have presented this attitude determination system to NASA with a set of documentation that includes documentation for MADS, a readme for MADS and a guide for using MADS. These are in Appendix A.

Appendix A Documentation for MADS as Supplied to NASA

A. 1 Documentation for balloon position and velocity

A. 2 MADS Read Me

A. 3. MADS Guide

A. 1 Documentation for balloon position and velocity

Adaptation of Satellite Attitude Determination Software for Balloon Flights

Prepared by: **Liping Mo**

Advisor: **Dr. Natalia Zotov**

Abstract

MTASS (Multimission Three-Axis Stabilized Spaceraft) is a package used by NASA to determine satellite attitude. It has been adapted for use by balloon missions carrying instruments such as gamma-ray telescopes. The new package, which we have named MADS (Multi-mission Attitude Determination System) first uses pre-launch data and gyroscope data to determine the initial in-flight quaternion at the time when useful camera data becomes available, and then replaces the satellite ephemeris using GPS data.

Additional sections have added for the convenience of balloon users on interpolating across gaps in GPS data, extracting information equivalent to star tracker data from ordinary CCD camera data, interpolating later between camera frames if these are taken infrequently because of storage limitations, and decoding the quaternions into time-tagged pointing direction and rotation angle.

Table of Contents

1.0 Introduction

Following the example of MTASS, we divide the post-flight data processing into two parts: part A handles pre-processing of raw data, and part B deals with mission-independent data analysis.

Data to be processed in part A:

Gyro data: The Gyro data formats in MADS are x-axis, y-axis z-axis angles in radians or counts. They are rotation angles about the x-axis, y-axis, z-axis.

GPS data: MADS uses a GPS(Global Positioning System) data file which gives the location of the balloon with respect to Earth at universal time, instead of ephemeris files which are used for a satellite. The GPS data are read as latitude, longitude, and altitude at universal time, We need to convert those into geocentric rectangular coordinates, and put them in the format (t,x,y,z) used by MADS. The GPS raw data also need to be checked for gaps which happen when the GPS receiver is out of range of the satellites. In this case, we use polynomial interpolation to interpolate the across gaps.

CCD data: For the CCD camera data, first select up to five brightest stars from each frame. For each star, determine the apparent magnitude and location in the frame of its centroid. Then correct these for such effects as distortion by spherical lens, and distortion due to temperature. The format of CCD camera in MADS is (time, phi, theta, temperature, intensity, flag), where the angles are explained in the documentation.

Part B: Mission Independent Analysis.

Use GPS data to calculate the velocity of balloon at time t. Then we have data for balloon position and velocity at the time t.

This document includes a description of the functions that prepare GPS data for use in the main body of MADS,

corresponding to part B of MTASS, where they will replace the satellite's ephemeris.

Major difference between MADS and MTASS:

1. A balloon group must determine two initial quaternions a pre-launch quaternion, and another at a time when the balloon has became to be stabilized sufficiently for the cameras to provide useful data.
2. A balloon has no predetermined orbit, so ephemerides cannot be used. These are replaced by parameters calculated from the GPS data.
3. Camera frames are taken much less frequently, e.g. one per minute on long flights, because of data storage problems. MADS will initially calculate a new quaternion for a time interval containing a camera frame. Eventually, additional quaternions will have to be found by interpolation using the gyro data at times corresponding to triggers from the primary detector.

In addition, since balloon groups will typically be unfamiliar with the conversion of quaternions to astronomical coordinates, a routine for this will be supplied.

2.0 Determination of initial Attitude

Determination of the initial attitude of the primary detector or telescope is facilitated by the use of an orientation cube. This will be attached to the assembly consisting of the telescope and thattitude determination instruments: the CCD cameras and the gyroscope. Since the cube itself will probably not be machined very accurately, grooves are milled into the faces of this cube that set accurately at right angles, or their deviations measured. The angles between these grooves and the primary axis of the telescope, the boresights of the cameras, and the axes of the gyroscope must also be measured as accurately as possible, at least to within 1/10 degree.

Before launch, the alignment of this cube relative to magnetic north must be measured, and also the deviation from horizontal in two perpendicular directions. (The cube should be attached to the assembly in a position which will facilitate sighting along the grooves.) The other quantity needed is the exact time of launch. From this, the hour angle, or celestial longitude, of the zenith at the time of launch can be found from the web site of the U.S. Naval Observatory. These angles allow the telescope's right ascension and declination to be calculated, and the components of its initial quaternion to be found.

3.0 Extract Equivalent of Star Tracker Information from CCD Camera Files

In order to identify the stars, satellites use star tracker information in MTASS. Most satellites today used fixed head star trackers. The star trackers used on the RXTE satellite are designed to acquire and track stars whose magnitude is between +5.7 and +2.0 in an 8 degree square field-of-view. In general, a fixed head star tracker measures star position vectors. The fixed head star tracker's output consists of three parameters: H, the horizontal coordinate of the star in the field-of-view, or an equivalent angle; V, the vertical coordinate of the star in the field-of-view, or an equivalent angle; and I, the star-generated signal intensity. The intensity is used to determine the star magnitude. In MTASS, the fixed head star tracker data is input data with format:
(time, phi, theta, temperature, intensity flag). MTASS uses fixed head star tracker data to calculate an observed vector for each star with respect to the camera, and use this with ephemeris data to calculate a reference vector with respect to Earth's center. Then it compares the observed star's reference vector with a catalog star's reference vector to identify the star. Balloon experiments use CCD cameras to take pictures of stars, to measure the stars in the background star fields. The images taken by the camera will be processed and the magnitude of the background stars in the field-of-view will be calculated. The images are very large and to store them on-board the balloon system would require massive amounts of the storage media. In MADS, the CCD camera data are used to calculate observed vectors for stars with respect to camera, and GPS data are used to calculate reference vector of star with respect to Earth's center. Then comparison of observed vectors and reference vectors is used to identify the stars. The CCD camera data has to be in the format:
(time, phi, theta, temperature, intensity flag). Corresponding to each time, there is one camera frame. In order to be useful, each camera frame has to be included at least three stars.

The programs analyzing CCD camera data include: Drivecam.m, matrix.m, CCDsort.m, imgcent.m.

3.1 CCD camera coordinate system

The CCD camera coordinate system is shown in following:

- b distance from center of lens to CCD chip.
- v is vertical component relative to chip.
- H is horizontal component relative to chip.
- ϕ is angle associated with horizontal component of image component.
- θ is angle associated with horizontal component of image component.

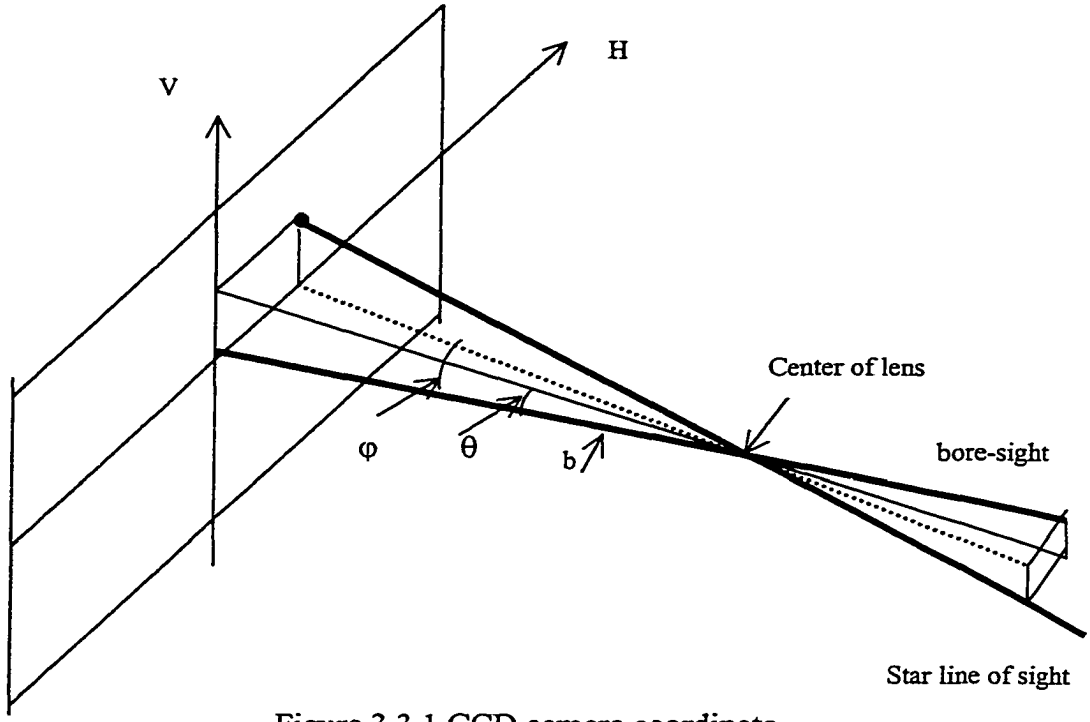

Figure 3.3.1 CCD camera coordinate

From this graph, (H, V) are centroid of image with respect to image centre. b is a constant value and is given by CCD camera. So

$$\tan \varphi = H/b; \text{ and } \tan\theta = V/b;$$

The φ and θ are represented by the centroid of image. These φ and θ are the variables that used in MADS as camera data for star position.

3.2 CCD camera frame is as a matrix

For the Surpose of illustration in this document, let us suppose that the CCD camera chip has 512×768 pixels, as for the Kodak chip. Assume that the camera data are stored in a file as one column, and there are 512×768+2 rows of data for each frame. The one datum in the first row is the number of row of pixels and the datum in second row is the number of column of pixels. Based on those two numbers, the program called mtatix.m will put the data into a 512×768 matrix from the first row of the first column to the last row of the last column.

3.3 Finding the brightest star in the CCD camera frame

A CCD frame will contain the images of many stars. In MADS, we need at least the three brightest stars to do the star identification but no more than five. So we need to find 3-5 brightest stars from the CCD frame, and those stars must not be at the edge of frame. Each star in the frame has a star+sky window. The size of this window is dependent the on the choice of lens and exposure time for a camera. The best choice of lens for the Apogee Ap1 are a 55 mm lens (Rossi XTE) or a 50 mm lens (standard) (N. Zotov, 1998). Normally the point source of light is focused on one pixel. For any given CCD chip, the lens is chosen so that the starlight, which has been defocused to fall on 2×2 pixels, will trail by no more than two pixels during an exposure. Ball allows 6×6 pixels for star box, making allowance for changes in temperature and 8×8 pixels for a star+sky window. So for a CCD chip, the 8 fixels from the chip edge are not using for calculation of centroid of the stars. Figure 3.3.2 shows the star box and star+sky window in a CCD chip.

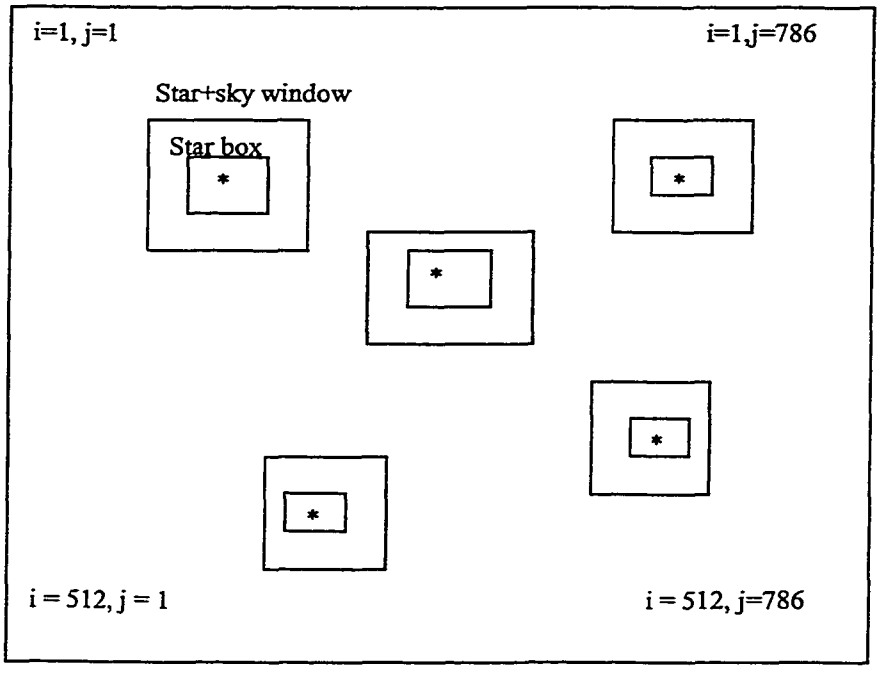

Figure 5.2.1 Stars+sky window and star box in the frame

3.3.1 Finding the brightest star in frame INPUT

M = matrix of CCD camera frame. The element in the matrix is the value of pixel in the frame.

groundbval = the lower bound value, such that the value bigger then will be consider to represent light from the star.

winsize = 8 is size of of star+sky window.

egsize = 8 is the size of row or column of edge in frame matrix.

n = number of rows for matrix.

m = number of columns for matrix.

max = maximum value of pixels in the frame.

3.3.2 Finding the brightest star in frame PROCESSES

1. Cut the edge of matrix
   let $M^{cutedge}$ be a remain matrix that have cut first 8 rows and columns pixels, also cut last 8 rows and 8 columns from from the CCD frame. Only picks up brightest from $M^{cutedge}$ so that star won't be in the edge of the chip.

2. Finding the maximum counts max_{1st} from $M^{cutedge}$ with its row and column number (i_{1st}, j_{1st}).

3. Cut the star+sky window
   The brightest star in the frame has max_{1st} at (i_{1st}, j_{1st}), The star+sky window has 8×8 pixels and its center is at (i_{1st}, j_{1st}), and set the counts in this window to be zero in order to find the second brightest star in remain matrix, so that it does't pick up the same counts .
4. Repeat the processes
   Repeat the processes from step 1 to step 3 in the remaining submatrix to find second, third brightest stars in the frame until has been found maximum five stars in the frame even the frame includes many stars.

3.3.3 Finding the brightest star in frame OUTPUT

st = k x 3 matrix for k stars with the maximum pixel value of each, and its row and its number. The values of pixels are sorted in increasing order.

3.4 Calculation of centroid and intensity or brightness of a star

For each star in the frame, find the star's centroid with respect to the frame's center. This will be used later for calculation of φ and θ. The intensity of the star is a measure of its brightness above the sky background background.

3.4.1 Calculation of centroid and intensity INPUT

M = matrix of CCD camera frame. The elements in the matrix are values of the pixels in the frame.

st = k x 3 matrix for k stars. The first column of matrix is value of the brightest for star. The 2nd and 3rd columns are the row number and column number of the brightest pixels.

3.4.2 Calculation of centroid and intensity PROCESSES

1. Size of star box

The size of star box is 6×6.

2. Calculation of marginal sums of star box

$$\rho(x_i) = \sum_{j=-3}^{3} I_{ij}$$

and

$$\rho(x_j) = \sum_{i=-3}^{3} I_{ij}$$

Here the I_{ij} are the value of pixels in box.

3. Calculation mean intensities

The mean intensities in the box for each marginal is:

$$\bar{x} = \frac{1}{2a+1} \sum_{i=-3}^{3} \rho(x_i)$$

and

$$\bar{y} = \frac{1}{2a+1} \sum_{j=-3}^{3} \rho(x_j)$$

4. Calculation of centroids of brightness

Let $x_i = -3 : 3$, and $y_j = -3 : 3$

$$x_c = \frac{\sum_{i=-3}^{3}(\rho(x_i) - \bar{x})x_i}{\sum_{i=-3}^{3}(\rho(x_i) - \bar{x})}$$

$$y_c = \frac{\sum_{i=-3}^{3}(\rho(y_i) - \bar{y})y_i}{\sum_{i=-3}^{3}(\rho(y_i) - \bar{y})}$$

here (x_c, y_c) is centroids coordinate of brightness which represent the rows and columns in the frame matrix.

5. Calculation of centroids with respect to center of frame

Since one frame includes 512 rows and 768 columns, so the center of frame (H_0,V_0) in HV coordinate is at (512/2,768/2) (row and column) in the frame matrix.

Let

$$h_0 = 768/2$$

$$v_0 = 512/2$$

Then the centroid represent by the HV coordinate is:

$$H_c = y_c - h_0$$

and

$$V_c = x_c - v_0$$

(H_c,V_c) is a centroid of brightest with respect to the center of frame.

6. Calculation of the sky background intensity

The star+sky window is 8 by 8 matrix. The star box is a 6×6 matrix within this matrix. The 'annulus' are

between star+sky window and box includes the pixels that in the star matrix but not in the box. Those pixels are used to calculate the intensity of sky background.

Let ANNU = { all values of pixels in the annulus }

$$Median = median(ANNU),$$

$$Mean = mean(ANNU).$$

$$mode = 3 * Median - 2 * Mean$$

So, the intensity of sky background is as:

$$I = \sum I_{ij} - n_{pix} * i_{sky}$$

here, $\sum I_{ij}$ is summation that carried out over the pixels in star matrix,

$$n_{pix} = cutrow * cutcol$$

and

$$i_{sky} = mode$$

7. Calculation of magnitudes of brightest stars

Let *zpt* be the zero point determined by calibrating the camera. The typical values are around 23.5 to 25. Then the magnitude of background is:

$$m = zpt - 2.5 \log I$$

here I is intensity of background.

3.4.3 Calculation of centroid and intensity OUTPUT

H_c = horizontal coordinate of centroid of brightest star.
V_c = Vertical coordinate of centroid of brightest star in.
I = intensity of brightest background.

3.3 Calculation of φ and θ

In the Figure 3.1.1, the φ and θ are angles related to the horizontal and vertical coordinates (H,V) of a star in the plane of the CCD chip.

3.5.1 Calculation of φ and θ INPUT

H = horizontal coordinate of centroid.
V = Vertical coordinate of centroid.
b = distance from lens to camera frame.

3.5.2 Calculation of φ and θ PROCESSES

As shows in Figure 3.1.1, θ is an angle that related to the vertical coordinate of centroid, and φ is angle that related to the horizontal coordinate of centroid.
So

$$\tan\varphi = H/b,$$

and

$$\tan\theta = V/b\ .$$

Then,

$$\varphi = \arctan(H/b)$$

and

$$\theta = \arctan(V/b)$$

3.5.3 Calculation of φ and θ OUTPUT

φ = angle relate to the horizontal coordinate of centroid

θ = angle related to the vertical coordinate of centroid..

The final output for CCD camera data program is saved in file imgdat.dat. This is the CCD data file will be used in MADS. It is in the format: (time, phi, theta, temperature, intensity, flag), where

time = time when the picture was taken.
phi = φ, angle related to the horizontal coordinates of centroid.
theta = θ, angle that related to vertical coordinates of centroid.
temperature = temperature measured by a camera sensor. This allows the user to calculate loss of apparent intensity as camera defocuses due to temperature changes .
intensity = I, intensity of star in a star background.
flag = Boolean value, as star track for the flag In MTASS.

4.0 A review of GPS data

The Global Positioning System (GPS) will give three geodetic coordinates of the balloon gondola: latitude, longitude, and altitude at universal time. We have to convert the geodetic coordinates to geocentric rectangular coordinates and reformat the data records as (t, x, y, z) for MADS to use.

We assume that the following complement of instruments is available: a GPS receiver, two CCD cameras, and a gyroscope. All the instruments are fixed relative to the telescope or primary detector.

The position vectors with their associated time coordinates determined by the GPS are used to calculate velocity vectors for the balloon. Up to five stars have previously been selected from each camera frame. The GPS position and velocity vectors now allow the calculation of an "observed" vector and a reference vector for each star. The observed vector gives the position of a star relative to axes fixed in the balloon, and the reference vector gives its position relative to geocentric axes that are fixed in the Earth. The reference vectors for each frame are compared with catalog star vectors, using files from MTASS, to identify the stars. The pointing direction and rotation angle of each camera can then be determined, and this information is combined with gyro data to determine the attitude of the telescope or other primary detector.

4.1 Transformation for GPS data

The GPS data is assumed to be stored in the format (t, latitude, longitude, altitude) in a file called GPS.dat. They are in geocentric coordinates. We convert those coordinates to geocentric rectangular coordinates using the function gpstrans.m and save those into gpscvnew.dat.

4.1.0 Balloon position represented by GPS data.

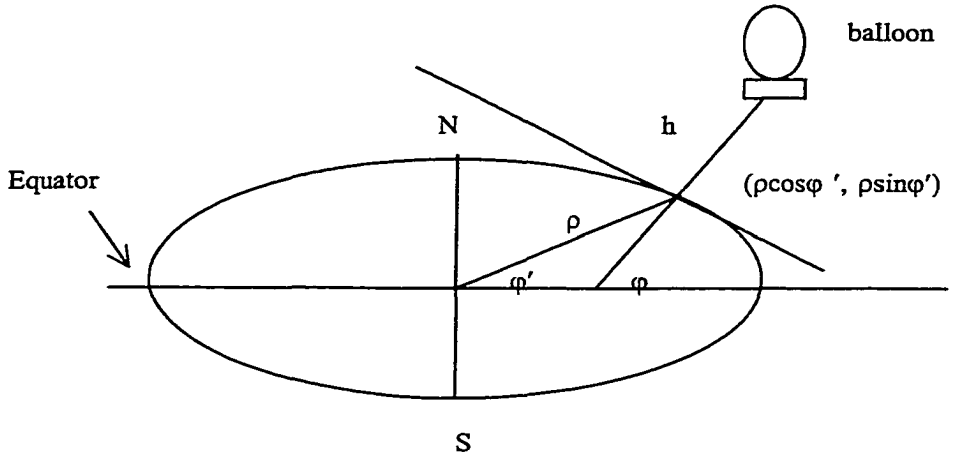

Figure 4.1.1.a Balloon position given by GPS data

This graph gives the position of the balloon with respect to Earth. The angle φ is read out from GPS data as the latitude of the balloon, ρ is the radius of Earth at that latitude, and h is the altitude of the balloon.

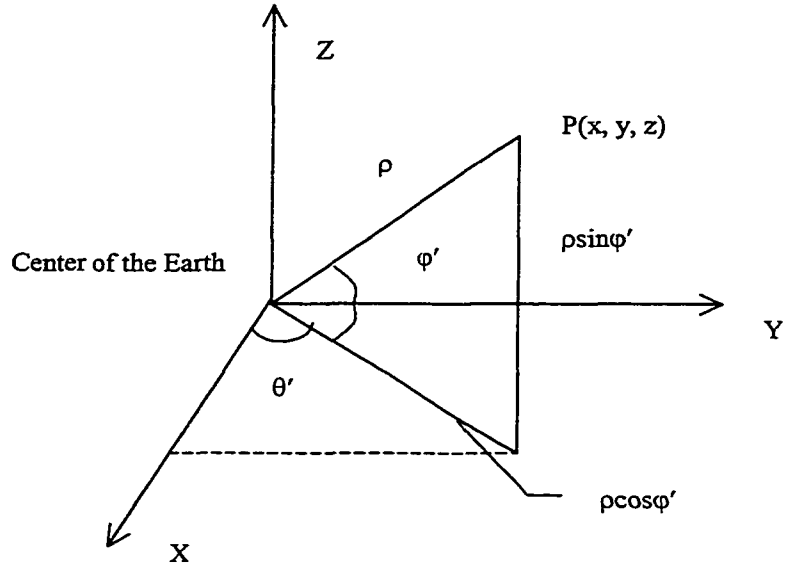

Figure 4.1.b

The figure shows the balloon position in rectangular coordinates.

4.1.1 Transformation for GPS data INPUT

t = The universal time of a position measurement for the balloon.

φ = The geographical latitude of the balloon, determined using a normal to the Earth's surface(which does not go through the center of the Earth).

θ = The geographical longitude of balloon which is measured from the longitude through Greenwich.

h = The altitude of the balloon, which is its height above sea-level.

4.1.2 Transformation for GPS data PROCESSES

1. Find the geocentric latitude φ' in terms of the function $\sin\varphi'$ and $\cos\varphi'$.

   Let

   $$u = \tan^{-1}\{0.996647 * \tan\phi\}.$$

   φ' must be reckoned as positive in the northern hemisphere and negative in the southern hemisphere. (See the reference (1))
   Let

   $$h' = (h/6378140).$$

   where h' is a correction of height h above sea-level. The equatorial radius of the Earth is 6378140m.

   Let

   $$\rho\sin\varphi' = 0.96647 * \sin u + h' * \sin\varphi.$$
   $$\rho\cos\varphi' = \cos u + h' * \cos\varphi.$$

2. The geocentric longitude θ':

   $$\theta' = \theta$$

3. Find the coordinates in the geocentric coordinate system,

using

$$x = \rho \cos \varphi' \cos \theta'.$$

$$y = \rho \cos \varphi' \sin \theta'.$$

$$z = \rho \sin \varphi'.$$

The coordinates (x, y, z) give the balloon's position with respect to Earth's center.

4. Put data into processed GPS format as (t, x, y, z), where
   t is the same in the new GPS data file,
   and x, y, z have be calculated by process 2.

   We save them into a file gpscvnew.dat. It is ready for MADS to use in functions that require the balloon's position and velocity.

References:

(1) Practical Astronomy with Your Calculator. Duffe Smith, Peter 1980.
(2) Explanatory Supplement to the Astronomical Almanac, Ed. P. Kenneth Seidelmann (U.S. Naval Observatory), University Science Books, 1992.

4.1.3 Transformation for GPS data OUTPUT

t = Universal time for position of balloon.
x = x-coordinate value of balloon position in the geocentric coordinate system.
y = y-axis value of balloon position in

the geocentric coordinate system.

z = z-axis value of balloon position in the geocentric coordinate system.

4.2 Files for using GPS data

The GPS data have been translated into geocentric coordinates and MADS will use them to find the position and velocity of the balloon.

The determination of balloon position and velocity is done by three programs in MADS:

rungps.m, rdgps.m, calvel.m.

The file rungps.m is a main program for finding balloon position and velocity. First, it reads a GPS file that has been reformatted by the program rdgps.m. Then it calculates a velocity vector of the balloon using the program calvel.m, and passes back position and velocity vectors.

The program rdgps.m reads gps data using a given initial reading time or a line number and reading one line of data at each calling, so that it can start and end the read time as we want.

The program calvel.m is used to calculate the velocity vector of balloon movement over each one second interval, assuming that is the interval between camera frames.

The program named predict.m can be used to fit the missing data when the balloon is out of range of GPS satellites over a time interval. We have decided to interpolate for missing data in part A of MADS, so at this stage the GPS data file is without gaps. We use a natual cubic spline interpolation method to fitting missing data.

These functions perform the task of using GPS data files for balloon position and velocity in part B of MADS instead of using ephemeris files as in MTASS for satellite position and velocity.

4.3 Balloon position and velocity

4.3.1 Balloon position and velocity INPUT

inipos = initial value for position of balloon.
iniTim = initial value for time.
iniLine = gps file start line of reading Gps data.

- time = column array of input times.
- gps_name = GPS data file name.
- gps_parm = GPS parameters.
- unit_number = unit number to use gps access.

4.3.2 Balloon position and velocity PROCESSES

1. Find size of input time column array.
2. If size of time is less then zero, input time is invalid.
3. Initialize r_1, v_1, r, v, as zero matrixes,
   Where r_1 is 1x3 matrix for one position of balloon
   v_1 is 1x3 matrix for one velocity of balloon
   r is 3xn matrix for balloon position
   v is 3xn matrix for balloon velocity
4. Get j^{th} position of balloon at line j in GPSdata file.
   j is the line number of GPS data to be read from GPS file.
   Calling rdgps.m to get the balloon position at j^{th} line in pgs file.
   $r_1 = (r_j(1), r_j(2), r_j(3))$ read from GPS data file at j^{th} line.
   t_1 = time where the balloon at position r_1.
5. Put r_1 into matrix **r** as j^{th} column of matrix **r** for output format.
6. Get the velocity of balloon at j^{th} position from **calvel.m** .
   Using $r_1, t_1, r_0, t_0,$ as input variables to call calvel.m to get the Velocity of balloon v_1 at r_1.

$$v_j = \frac{\Delta s}{\Delta t}$$

7. Put v_1 into matrix **v** as j^{th} column of matrix **v** for output format.
8. Put r_1, t_1 as initial value r_0 ,t_0 , iterate to do the processes 4,5,6, 7 to get $r_1, r_2 \ldots\ldots r_p$ and $v_1, v_2 \ldots\ldots v_p$ for balloon at required time.

9. The **r** is a 3xn matrix, n is dependent on input time or dependent on how many GPS data points we need to read.
10. The **r** is a 3xn matrix, n is dependent on input time or dependent on how many GPS data points we need to read.

4.3.3 Balloon position and velocity OUTPUT

- position of balloon during a time period.
  This is a 3 × n matrix.

- Velocity of balloon during a time period.
  This is a 3 × n matrix.

- quality is a variable that indicates the quality flag.
  The quality is zero if reading the GPS file was successful and is not zero if there was a problem on reading the GPS file.

4.3.4 Balloon position and velocity

A rdgps.m

This function is used to read the GPS data from the GPS file. It reads one line data by given line number, so that we can get the data as we need it.

1. **rdgps.m** **INPUT**

- The first line number.
- The processed GPS file name for reading the data.
- The unit number for using GPS data.
- The GPS parameters. In our case, we only use parm(1,1) = 2;

2. **rdgps.m** **PROCESS**

- Open the input GPS file
- Read the data from the specified line.
- Put data value into GPS position vector as gpspos (1x3) matrix for output format.
- Put data value into GPS time vector as gpstim (1x1) matrix for output format.

3. **rdgps.m** **OUTPUT**

- Balloon position vector (x,y,z,t) at universal time t.
- Time of balloon position.
- Message for reading the GPS data file.

B **calvel . m**

This function is used to calculate the velocity vector for balloon at a given point in space and time. The balloon moves with different velocities at different times.

1. calvel.m **INPUT**

   - Initial position vector for balloon.
   - Next position vector for balloon at end of time period.
   - Initial time for starting calculation of balloon velocity.
   - The end time for interval Δt.

2. calvel.m **PROCESS**

   The position vector of balloon with respect to center of Earth in geocentric coordinates. The position vector we use is from the GPS data file. The position vector is in (x, y, z) format.

   a) Find the change of position vector in the Period time interval.

$$\Delta x = x - x_o$$

$$\Delta y = y - y_o$$

$$\Delta z = z - z_o$$

   b) Find the change of time in the period time interval.

$$\Delta t = t - t_o$$

c) Find the velocity vector of balloon at position (x, y, z).

$$v = \left(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}, \frac{\Delta z}{\Delta t}\right)$$

The velocity vector is a (1x3) matrix.

3. calvel.m OUTPUT

The velocity vector of balloon during a time interval Δt at position (x, y, z).

5.0 Conclusion

The functions determining balloon position and velocity are used to find the position vector of balloon in the geocentric coordinate system and the velocity vector at that position using GPS data. MTASS uses ephemeris data to find the position and velocity vectors for satellite but no ephemeris are available for balloon experiments. The GPS data are received from a GPS receiver on the balloon. We use MATLAB to run the software package that processes the GPS data.

The files and documentation will enable a balloon group to use NASA's MTASS package to determine the attitude of balloon instruments at any point in time. Since the geocentric coordinate system is used, MADS can be used at any latitude or altitude.

A. 2 MADS Readme

Welcome to the MADS

(Multimission Attitude Determination System)

for Balloon Missions

This document contains information on:

1. View of MADS
2. Getting started (Creation of MADS)
3. A list of files for MADS

1. View of MADS

MADS (Multi-mission Attitude Determination System) is a new software package which is used to determine the attitude of instruments on a high-altitude balloon. It is an adaptation of NASA's MTASS (Multimission Three-Axis Stabilized Spacecraft), whose original function was to determine satellite attitude. Among the differences are MADS's use of the GPS, rather than ephemerides, to locate the balloon, and a suite of programs to process raw CCD data to obtain the information that is available from the star trackers carried on satellites.

MADS has two parts, corresponding to the division of MTASS: Part A includes the programs that are used to treat raw data including gyro data, GPS data, and CCD data. Part B is the mission-independent section, which includes locating the balloon, identifying the stars in the CCD frames, and determining the attitude of instruments such as a telescope.

2. Getting started (Creation of MADS)

(1) Request a copy of MTASS from Guidance, Navigation and Control center Flight Dynamics Analysis Branch. Contact Mr. Rick Horman at (rharman@pop500.gsfc.nasa.gov) to get a copy of the RXTE version of MTASS.

(2) Use Part A to treat the raw data.
In Part A, there are three folders: GPS, Gyro, and Camera.

1. The GPS folder includes the programs that transform GPS data from geodetic coordinates to geocentric rectangular coordinates, and put them into the file GPSnew. dat. Use the program drivegps.m to carry out this process.
2. The Camera folder includes the programs drivecam.m, matrix.m, CCDsort.m, and imgcent.m. Those programs read the camera data from the CCD frame data file, find the brightest stars in the frame, calculate the centroids of the brightest stars in the frame and put them into the file imgdat1.data, using the program drivecam.m to do this. In order to run this program, you need to submit the CCD camera data file name to the program imgdat.m for it to open and read the CCD raw data from the camera data file. If there are two CCD cameras, you will need to change the input file name and the output file name to imgdat2.dat in place of imgdat.m for processing data from the second CCD camera.

(3) Creating Part B of MADS

Part B includes the programs rungps.m, rdgps.m, and calvec.m.

1. In a copy of MTASS, rename the file runephem.m, so that you can return to the original file if you need to.
2. Rename rungps.m to runephem.m in Part B, which is in folder Part B.
3. Copy new runephem.m, rdgps.m, and calvec.m from Part B into the copy of MTASS. The new runephem.m has the same name as an MTASS file that is used for satellites, but its function is using GPS data to locate the balloon.
4. Copy GPSnew.data from mads_rawgps folder in Part A into the copy of MTASS
5. Copy imgdat1.data and imgdat2.data from mads_rawccd folder in Part A folder into the copy of MTASS.

MADS has now been created from MTASS by adding these new files.

(4) Use MADS to determine the attitude of the balloon

1. In MADS, modify file xte_nl.m which is the namelist for the RXTE MATLAB Ground System. In xte_nl.m, do the following set up:
   - reset ephem_name ='GPSnew.data';
   - reset fhst1_raw_dsn = 'imgdat1.data';
   - reset fhst2_raw_dsn = 'imgdat2.data';
   - reset ekf_state_parm(1,1)=1; (user input quaternion)
   - reset ekf_state_parm(2,1)=1; (user QUEST attitude history file)

- reset si_direct_parm(8,1)=1; (for using input quaternion)
- reset si_direct_parm(9,1)=0; (not using on-board attitude history file)
- reset si_doub_parm(2,1)=1; (min # of doublet match in star identification)

2. Run MADS to find the attitude of balloon instruments in two steps: (1) Generate initial quaternion at the time when the balloon stabilizes. (2) Find the attitude of balloon instruments during the flight time. Following the direction in file RUN_MADS.doc to use MADS.

3. A list of files for MADS

Part A folder includes two subfolders:

GPS folder: drivegps.m
gpstrans.m

Camera folder: drivecam.m
CCDsort.m
matrix.m
imgcent.m

Part B: rungps.m
rdgps.m
calvec.m

A. 3. MADS Guide

Direction for running attitude determination system in MADS

Since MADS is a adaptation from MTASS, the directions for MADS are modified from the directions of running MTASS for Attitude Determinate System by Michael Lee at Guidance, navigation and Control Center Flight Dynamics Analysis Branch.

Using MADS in the determination of attitude for balloon instruments includes two processes:

Process 1: Generate initial quaternion at the time the CCD camera starts to take useful pictures of stars. This initial quaternion is found by propagation of a ground-level initial quaternion using gyro data only in the time interval between the time of balloon launch and the time when the CCD camera begins to take the useful data (i.e the balloon has become stable). The ground-level initial quaternion is calculated from angles measured from the CCD camera case which is set up on the balloon.

The file gyro_raw1.data should include the gyro data in this time interval.

Process 2: Find the attitude quaternion of balloon instruments using camera data, the initial quaternion, and gyro data over a second time interval that is the same as the CCD camera data time interval.

The file gyro_raw2.data should include the gyro data in the second time interval.

When you open the MATLAB and go to the MADS directory, take the following steps:

Open up MATLAB. Go to the directory containing MADS.

Process1: Generate initial quaternion

In file xte_nl.m: set iru_raw_dsn='gyro_raw1.data'; Then in MATLAB:

(1) Type "runxte"

(2) Using the mouse, click on the "DATA ADJUSTER" button in the graphical interface.

(3) Click on the "LOAD RAW SENSOR DATA " button. In the display window, only Load IRU raw data set . Then click on "LOAD DATA AND EXIT " button.

(4) After data has been loaded, the timespans of the available data can be see by clicking the "START AND END TIMES" button.

(5) Click on the "EXECUTE DATA ADJUST" button. The gyro data for the balloon has now be ingested in the MATLAB environment in a usable format.

(6) Click on the "EXTENDED KALMAN FILTER" button, then click on the "USER INPUT ATTITUDE" button. In window for input attitude:

Initial attitude source, check for 'quaternion'.

Attitude reference, check on 'GCI'

Attitude time, input an attitude time within gyro time (i.e within gyro_raw1.data file timetag)

Then input the primary quaternion determined from ground measurements into attitude relative to reference frame

Quat. comp1,

Quat. comp2,

Quat. comp3,

Quat. comp4,

Then click "SAVE INPUT" button, and "EXIT".

(7) Click "EXECUTE EKF" button. This will display the quaternion with the epoch time. This quaternion is the initial quaternion at the time when the CCD camera starts to take the useful pictures of stars.

Exit the EKF MAIN MENU, and back to XTE MATLAB GROUND SYSTEM.

(8) Exit MADS.

Process2: Find attitude quaternion for balloon instruments

In file xte_nl.m: set iru_raw_dsn='gyro_raw2.data'; Then in MATLAB:

1. Type "runxte"

2. Using the mouse, click on the "DATA ADJUSTER " button

3. Click on the "LOAD RAW SENSORDATA" button then click "LOAD DATA AND EXIT" button

4. After the data have been loaded, the timespans of the available data can be see by clicking the " START AND END TIME " button (then EXIT the GUI)

5. Click on the "EXECUTE DATA ADJUST " button. The gyro, CCD camera and onboard attitude for the balloon have now been ingested in the MATLAB environment in a usable format.

6. Next, input the user attitude as follows. EXIT from the data adjuster("DA") executive display and click on The "EXTENDED KALMAN FILTER" button then, the "USER INPUT ATTITUDE ". In window for input attitude:

Initial attitude source, check for quaternion.

Attitude reference, check on GCI.

Attitude time, input an attitude time within gyro time (which is within gyro_raw2.data file timetag)

Then input the initial quaternion that was determined from process 1:

Quat. comp1,

Quat. comp2,

Quat. comp3,

Quat. comp4,

Then click "SAVE INPUT" button, and "EXIT".

7. Next, perform star identification. EXIT from the "EKF" executive display and click on the "STAR ID" button.

8. Read in the star catalog (already included for the example data in a *.mat file format). Click on the "GENERATE STAR CATALOG" button and then the "LOAD DATA AND EXIT" button.

9. The namelist parameters have been set up for the RXTE case (use the same set for balloon). Click on the "EXECUTE DIRECT MATCH" button (which does a preliminary star identification) and then "EXECUTE DOUBLET MATCH" button. The stars should now be identified and the blue buttons can be clicked for a graphical display for the results. The "POS-DA" plots show the right ascension and declination of all the processed tracked observations (represented as white circles) in the Geocentric inertial coordinates(GCI) of the mission (J2000 for RXTE) assuming that the a priori attitude is correct. The "PRE-SI" plots show the clump star observations and the reference stars resulting from the direct match algorithm. The observations are white circles and the yellow crosses are the candidate reference stars. Finally, the "POST-SI"

plot includes the reference stars identified with the observations along with the other items from the "PRE-SI" plot. EXIT from the star identification GUI when ready.

10. An attitude determination method can now be applied. The "EXTENDED KALMAN FILTER" options have already been preset for the example case. Click on this button,

then click on the "EXECUTE EKF" button. There will be plots of star tracker (FHST # 1 and #2) residuals if using satellite data, or CCD camera (CCD #1 and #2) residuals for the balloon case and the solved for quaternions will be available –use the labeled buttons.

11. In the satellite case, the performance of the onboard computations can be compared with the ground attitude. Click on the "ATTITUDE COMPARSION' and then the "execute compare" buttons, the system is present to compare the EKF attitude solution with the onboard solutions. Check the "ATTITUDE COMPARISON PLOTS" and about a 15 arcsecond rediual will be seen in the "Comparison Residuals" plot. This difference is due to not performing velocity abberration corrections on the star tracker data. For this correction, the Solar, Lunar and Planetary ephemeris information (SPL file) is needed. This file and the SKMAP star catalog are available on the internet at ???

12. The SLP data and the star catalogs can be obtained by following the directions on the Web page.

Appendix B Programs for MADS

B.1 MADS Part A programs

1. dirvecam.m
2. CCDsort.m
3. matrix.m
4. imgcent.m
5. drivegps.m
6. gpstrans.m
7. gpsfit1.f
8. gpsfit2.f
9. gpsfit3.f
10. gpsfit4.f

1. drivecam.m

```
% This function is used to find the brightness stars in CCD camera
% frame and put tem into CCD camera data file with the format as FHST
% data in MTASS by reading raw CCD data from camera.
% INPUT:     raw CCD data file.
% OUTPUT:    CCD camera data file in format:
%            (time phi theta temperature intensity flag)
% MODIFY:     Liping Mo

[fid,message] = fopen('rawccd.dat');
j = 1;
while(feof(fid))==0
position = ftell(fid);
stude = fseek(fid,position,-1);
u = fscanf(fid,'%g',[1,1]);
%disp(u);
position = ftell(fid);
k = feof(fid);
if k ==0
stude = fseek(fid,position,-1);
v = fscanf(fid,'%g %g',[1,1]);
v = v';
%disp(v);
position = ftell(fid);
stude = fseek(fid,position,-1);
w = fscanf(fid,'%g %d',[1,3]);
w = w';
disp(w);
position = ftell(fid);
A = fscanf(fid,'%d %d',[1,w(2,1)*w(3,1)]);
A = A';
 [a,b,M] = matrix(w(3,1),w(2,1),A); %put the frame into matrix.
 [B] = CCDsort(M);                  %find the brightness stars in
frame.

%% put the brightness into format
 [stnum,stinf] = size(B);
 len=10;       % This len has to choice by user.
 tempeture=2; % This tempture has to measure from CCD camera.
 flag=0;
 for i = 1:stnum
 img(i+(j-1)*5,1) = v(1,1);
 img(i+(j-1)*5,2) = atan(B(i,1)/10);
 img(i+(j-1)*5,3) = atan(-B(i,2)/10);
 img(i+(j-1)*5,4) = 2;
 img(i+(j-1)*5,5) = B(i,3);
 img(i+(j-1)*5,6) = flag;
 %cent(i+(j-1)*5,1)= B(i,1); %centroid of star in H_axis
 %cent(i+(j-1)*5,2)= -B(i,2); %centroid of star in V_axis
 %cent(i+(j-1)*5,3)= B(i,3);  %intensity of star in the frame.
 end;
 j = j+1;
 end %for k==0
end; %while()
fclose(fid);
```

```
%save centriod.dat cent -ascii;
save imgdat.dat img -ascii -double; %save the data into the file
```

2. CCDsort.m

```
% This function is used to sort the CCD data for one frame.
% We just use the data that great than the background value.
% THis program first find the count of data that great than
% background value. Then just sort for those data. Put the
% sort data with their index of the matrix into the file
% for stwin4.m to use to call imgcent13.m to find the centriod
% of star with respect to center of frame.
% INPUT:   Read a CCD data file which has been in matrix for a
%          frame from the orgin CCD data file. It come out from
%          matrix.m.
% OUTPUT:  The data file which has been sorted with the index
%          of matrix with respect to a frame.
% MODIFY:  Liping Mo      3-14-99
function[B] = CCD1(imgmatrix);
%filename = 1;
%load imgmatrix.dat          %Read the matrix of CCD data
%ty = [imgmatrix];
%load stt1.dat
%ty = [stt1];
ty = imgmatrix;  %imgmatrix is input 8-1-99
[n,m] = size(ty);
groundbval = 230;              %Set up the background value which
decied
cutrow = 10;
cutcol = 10;
egrow = 15;
egcol = 15;
starnum = 3;                                %by histgram of orgin CCD
data.
%% Sort for data that great than background.
                        % as bright stars. So it is also number of
bright
                        % stars.
for k = 1:starnum          % loop only runing number of data that
great
 MAX = groundbval;      % background value times.
 Rc = 0;                % use for index of row for sort data.
 Cc = 0;                % use for index of colcumn for sort data.
for i = egrow:n-egrow
   for j =egcol:m-egcol
    if ty(i,j)~=0      %This condition is for checking the element not
0.
     if ty(i,j)>MAX
        MAX = ty(i,j);
        Rc = i;
        Cc = j;
%        disp(i)
     end;
    end;
  end;
end;
 A(k,1) = MAX;
 A(k,2) = Rc;
 A(k,3) = Cc;
```

```
% ty(Rc,Cc) = 0;       % try(Rc,Cc) is the max in this serach time.
Sing
 for i = Rc-cutrow:Rc+cutrow
      for j = Cc-cutcol:Cc+cutcol
           ty(i,j)=0;
      end;
 end; % it as zero, so next serach time


end;                    % end for sort loop.
a = 1;
b = 1;
%save sort1.dat A -ascii;

st = [A];
  [n,m] = size(st);

 for i = 1:n
        [centx1,centy1,mag1] =
imgcent(st(i,1),st(i,2),st(i,3),imgmatrix);
             B(i,1) = centx1;
             B(i,2) = centy1;
             B(i,3) = mag1;
        end;
%disp(A);
%    save centriod2.dat A -ascii -double;
```

3. matrix.m

```
% This function is going to read CCD images star data file for
% one frame that come from ADSDAQ system. The data are in one
% colcumn. This function will read the data and put them into a
% matrix for function imgcent13.m to use for finding the centriod
% of bright source.
% This program use led6.dat data from Dr.Greedwood
%
%
%INPUT:
%       1. Read the data of CCD images which are pixes values in
%          one frame.
%       2. The first value from data file is the column number of
%          matrix.
%       3. The second value from data file is the row number of
%          matrix.       .
%OUTPUT:
%          a    row number of matrix.
%          b    column number of matrix.
%          Matrix of frame. It save as ascil file.
%MODIFICATION:   11/12/98       Liping
%                6-20-1999      Liping   ouput row and column number
%                                        for stwin4.m to use.
%function[radius,centerx,centery] = starcent(rownum,colnum)
      function[a,b,frame] = matrix(mcol,mrow,A);
      d = 1;
       % load star.dat;    %File from Dr. zotov hand working

%%Load the CCD image data as line data,then put them into a matrix
 %     load led6.dat;  %File from Dr.Greenwood/pub/margie/image
 %     try = [led6];
       ty = A;
       %  disp(try);
      c = mcol;    %Get the matrix column number from CCD data file
      r = mrow;    %Get the matrix row number from CCD data file

%%create the elements of matrix for one frame.
        for l = 1:r
          for s = 1:c
           I(l,s) = ty((l-1)*c+s);
          end
        end             %end for read line data code.
      frame = [I];      %Matrix for CCD data file
      [N,M] = size(frame);     %n is row number of matrix
       a = N;
       b = M;
%save matrix.dat frame -ascii;
%disp(frame(44,6));
%disp(frame(49,15));
%disp(frame(32,32));
%disp(frame(32,28));
%disp(frame(24,16));
```

4. imgcent.m

```
%This function is going to read CCD images star data, find the center
%ofthe bright source and the radius of bright source matrix.
%This program use LED.dat data from Dr.Greedwood
%
%
%INPUT:
%       MAX          :   The pixel value of brightest
%       rownum(1x1):   The row number of star in the frame.
%       colnum(1x1):   The colnum number of star in the frame.
%       imgmat       :   The CCD frame matrix.
%OUTPUT:
%       centerx(1x1):  The x-axis value of centroid of source.
%       centery(1x1):  The y-axis value of centroid of source.
%       intensity(1x1):  The intensity of star.
%MODIFICATION:              .  Liping MO       11/12/98
%function[centerH,centerV,intensity] = starcent(rownum,colnum)
      function [centerH,centerV,intensity] =
imgcent(MAX,rownum,colnum,imgmat);

      ty = imgmat;      %use matrix as input 8-1-99
      frame = [ty];       %Matrix for CCD data file
      [N,M] = size(frame);
       %  disp(ty);
%      c = ty(1);     %Get the matrix column number from CCD data  file
%      r = ty(2);     %Get the matrix row number from CCD data file



%%Cut a window matrix for one star. It is 12 by 12 matrix
       SR = rownum;     %SR is row number of a star in the frema
       SC = colnum;     %SC is colnum number of a star in the frema
       stars = frame(SR-4:SR+4,SC-4:SC+4);
       [n,m] = size(stars);



%%Calculate the marginal sums for one window.
      py = sum(stars');          %Marginal sums by columns in images
                                 %matrix
      px = sum(stars);           %Marginal sums by rows in images matri
      Hmax = MAX/2;              %Half maximum value

 % Box size is dependent onthe lens of camer
      a=3;

%% Create the boxsize matrix as (2ax2a) size from the CCD data matrix.
      stars2 = stars(n/2-a:n/2+a,m/2-a:m/2+a);
        % disp(stars2);
      py1 = sum(stars2');     %Marginal sums of boxsize matrix by
                              %columns
      px1 = sum(stars2);      %Marginal sums of boxsize matrix by  rows
```

```
      A = -a:a;
      avgy = (1/(2*a+1))*sum(py1);  %mean of columns
      avgx = (1/(2*a+1))*sum(px1);  %mean of rows


%%Calculate the centriod of bright source.
corx = (px1 - avgx);
for i = 1:2*a+1
            if corx(i) > 0       %choose the elements of corx, such
that
            corx(i) = corx(i);   %corx>0.
            else corx(i) = 0;
            end
         end

      corx1 = corx.*A;           %The elements of numerator. A is
given
                                 %as x(i) values from array(-a:a).


      sx1 = sum(corx1);          %numerator of centroid formula
      sx = sum(corx);            %denominator of centroid formula
      centerx1 = sum(corx1)/sum(corx); %X-axis value of centroid.

      cory = (py1 - avgy)
          for i = 1:2*a+1
               if cory(i) > 0        %chose the elements of cory, such
that
               cory(i) = cory(i);    %cory>0.
               else cory(i) = 0;
               end
          end
      cory1 = cory.*A;               %The elements of numerator. A is
                                     %given as x(i) values from array(-
a:a)
      sy = sum(cory);                %The numerator of centroid formula
      sy1 = sum(cory1);              %The demerator of centriod formula
      centery1 = sum(cory1)/sum(cory);     %y-axis value of centroid.

%%Find the centerx1,centery1 with respect to frame center.
%%frame will be 512x768 pixes. The origin will be (768/2,512/2)
%%here I use DR. greenwood data, it 50x50 frame. (50/2,50/2)
      frameorgV = 512/2;                %column in the matrix)
      frameorgH = 768/2;                %row in the matrix)
      cx1= centerx1+colnum;             %column in matrix.
      centerH = cx1-frameorgH;          %Horizontal coordinate
      cy1 = centery1+rownum;
      centerV = frameorgV-cy1;          %Vertical coordinate



%%didn't use r1,r2,c1,c2 in follow calculate. Just for fun!
       r1 = n/2 - a;         %Up row edge of box matrix
       r2 = n/2 + a;         %Down row edge of box matrix
       c1 = m/2 - a;         %Left column edge of box matrix
       c2 = m/2 + a;         %Right column edge of box matrix
```

```
%%Find four parts from outside of boxsize(5ax5a if the data is
enought)
%%matrix.
       A1 = stars(1:(n/2)-a,:);                    %Up part of boxsize
matrix

       A2 = stars((n/2)-a:(n/2)+a,1:(m/2)-a-1);    %Left part of
boxsize matrix

       A3 = stars((n/2)-a:(n/2)+a,(m/2)+a+1:m);    %Right part of
boxsize matrix

       A4 = stars((n/2)+a+1:n,:);                  %Down part of boxsize
matrix



%%Put each part into the a column vector. This goes by columns. Then
%%create row for all background data selected.
       a1 = A1(:);
       a2 = A2(:);
       a3 = A3(:);
       a4 = A4(:);
       a = [a1',a2',a3',a4'];    %Row vector of background data.
        %  disp(a);
       sorta = sort(a);          %sort the vector in increasing order
       md = median(sorta);       %The median value of background data
       me = mean(sorta);         %The mean value of background data
       mode = 3 * md - 2 * me;   %The mode value of background data

%%Put Starmag13.m here to calculate the magenitude of star.  11-12-98

%%calculate the Marginal sums from rows.
     px = sum(stars);          %Marginal sums by rows in images matrix


%%Find the new matrix after backgourd subtraction.
     stars3 = stars - mode;

%%Calculate intensity in background.
      npix = n * m;      %npix is the number of pixels in the aperture
      isky = mode;       %isky is the background sky value(per pixel)
      sumimg = sum(px); %summation is carried out over original image
                         %data.

      intensity = sumimg - npix * isky;

%%Calculate magenitude for one star.
      zpt = 23.5;        %zpt is an arbitray number used to produce
                         %reasonable output values for the magnitudes
                         %I choose the value as 23.5.

      mag = zpt - 2.5 * log10(intensity);
```

5. drivegps.m

```
% This function is used to call function gogcv.m. We use this function
%to set up the GPS data file name and the number of gps data for
reading.
%  INPUT:
%         gpsfile:    GPS data file name.
%         gpsnum :    Number of gps data for reading.
%
% MODIFY:     Liping Mo    7-30-99
  gpsfile = 'rawgps.dat';
  gpsnum  = 100;
  [a] = gpstrans(gpsfile,gpsnum);
  disp('The conversion of GPS data is in "gps.dat"');
```

6. gpstrans.m

```
%This function transformat the GPS data from the latitude, longitude,
%and altitude to GCI (t,x,y,z) coordeate system. Liping 1-10-99.
%This function read a GPS data file.
%This function convert the data as (time, latitude, longitude,
%altitude) and save them into gpscv.dat file.
%function [t,x,y,z] = gpstrans()
 function[a] = gpstrans(gps_name,gpsnum);
% gpsnum = 10;
 gps=zeros(gpsnum,4);
 [fid,message] = fopen(gps_name,'r');
 for j = 1:gpsnum
 position = ftell(fid);
 stude = fseek(fid,position,-1);
 A = fscanf(fid,'%g %g %g %g',[4,1]);
 A = A';

    u = atan(0.996647*tan(A(1,2)));
    h = A(1,4)/6378140;
    psin = 0.99467*sin(u)+h*sin(A(1,2));
    pcos = cos(u)+h*cos(A(1,2));
% disp('psin');
%disp(psin);
%disp('pcos');
%disp(pcos);
gps(j,2) = pcos*cos(A(1,3));
gps(j,3) = pcos*sin(A(1,3));
gps(j,4) = psin;
gps(j,1) = A(1,1);
end;
fclose(fid);
a = 1;
save gps.dat gps -ascii -double;
```

7. gpsfit1.f

```
c This is a correct program  6/28/2000 for case 1
C
C     NOTE:  qk(x) = q(k)/6.0*[(t(k+1)-t)**3/h(k)-h(k)*(t(k+1)-t)]
C                  +q(k+1)/6.0*[(t-t(k)**3/h(k)-h(k)*(t-t(k))]
C                  +y(k)*[(t(k+1)-t(k))/h(k)]+y(k+1)*[(t-t(k))/h(k)]
C             FOR t(k) < t < t(k+1), k=0,....n-1
C
           dimension h(150),t(150),y(150)
     dimension aa(150),bb(150),cc(150),dd(150),q(150)
     double precision t,h,y,aa,bb,cc,dd,q
     n=96
c    tt=t0
c       ss0=
c       ssn=
c input data              .
     open(unit=06,file='mtax.dat')
     do i=1,n+1
           read(6,*) t(i),y(i)
     enddo
     close(6)

c set up Aq=b
     do k=1,n
     h(k)=t(k+1)-t(k)
     enddo
     do k=3,n-1
     aa(k)=-h(k-1)
     bb(k)=2.0*(h(k-1)+h(k))
     cc(k)=-h(k)
     dd(k)=6.0*((y(k+1)-y(k))/h(k)-(y(k)-y(k-1))/h(k-1))
     enddo
c ******************************************
     aa(2)=0.0
     bb(2)=2.0*h(1)+2.0*h(2)
     cc(2)=-h(2)
     dd(2)=6.0*((y(3)-y(2))/h(2)-(y(2)-y(1))/h(1))
     $         -h(1)*ss0
     aa(n)=-h(n)
     bb(n)=2.0*h(n)+2.0*h(n-1)
     cc(n)=0.0
     dd(n)=6.0*((y(n+1)-y(n))/h(n)-(y(n)-y(n-1))/h(n-1))
     $         -h(n)*ssn
c ***********************************************
     call trid(aa,bb,cc,dd,q,n)
c ***********************************
     q(1)=ss0
     q(n+1)=ssn
c ***********************************
c    call calqk(tt,qk,t,y,h,q,n)
c output the data into a file
     open(unit=06,file='mtax_result1.data')
     do i=1,n+1
     write(6,1) i,q(i)
```

```
      enddo
    1 format(i2,e20.10)
c     write(6,2) tt,qk
c   2 format(f12.8,1x,f12.8)
      close(6)
      end

c Tridiagonal linear system
         subroutine trid(aa,bb,cc,dd,yy,n)
         dimension aa(150),bb(150),cc(150),dd(150),yy(150)
         dimension be(152),ar(152)
      double precision aa,bb,cc,dd,yy,be,ar
         be(1)=0.0
         ar(1)=0.0
         do 1 k=2,n
         be(k)=cc(k)/(bb(k)-aa(k)*be(k-1))
 1       ar(k)=(dd(k)+aa(k)*ar(k-1))/(bb(k)-aa(k)*be(k-1))
         yy(n+1)=0.0
         do 2 k=2,n
      j=n+2-k
 2       yy(j)=be(j)*yy(j+1)+ar(j)
         return
         end

      subroutine calqk(tt,qk,t,y,h,q,n)
      dimension h(150),t(150),y(150),q(150)
      double precision h,t,y,q,tt,qk
      do 1 k=1,n
      if(tt.ge.t(k).and.tt.lt.t(k+1))then
         qk = q(k)/6.0*((t(k+1)-tt)**3/h(k)-h(k)*(t(k+1)-tt))
     $        +q(k+1)/6.0*((tt-t(k))**3/h(k)-h(k)*(tt-t(k)))
     $        +y(k)*(t(k+1)-tt)/h(k)+y(k+1)*(tt-t(k))/h(k)
      endif
 1    continue
      return
      end
```

8. gpsfit2.f

```
c This is a correct program  5/4/2000 for case 2.
C
C     NOTE:  qk(x) = q(k)/6.0*[(t(k+1)-t)**3/h(k)-h(k)*(t(k+1)-t)]
C                  +q(k+1)/6.0*[(t-t(k)**3/h(k)-h(k)*(t-t(k))]
C                  +y(k)*[(t(k+1)-t(k))/h(k)]+y(k+1)*[(t-t(k))/h(k)]
C             FOR t(k) < t < t(k+1), k=0,....n-1
C
           dimension h(150),t(150),tt(150),qk(150),y(150)
     dimension aa(150),bb(150),cc(150),dd(150),q(150)
     double precision tt,t,qk,h,y,aa,bb,cc,dd,q,hx,kx,t0
     n=96
        m=10
        t0=0.014035148
        hx=0.0000182
        kx=50
     do i=1,m
     tt(i)=t0+i*hx
     enddo
c input data
     open(unit=06,file='mtaxnew.dat')
     do i=1,n+1
           read(6,*) t(i),y(i)
     enddo
     close(6)

c set up Aq=b
     do k=1,n
     h(k)=t(k+1)-t(k)
     enddo
     do k=3,n-1
     aa(k)=-h(k-1)
     bb(k)=2.0*(h(k-1)+h(k))
     cc(k)=-h(k)
     dd(k)=6.0*((y(k+1)-y(k))/h(k)-(y(k)-y(k-1))/h(k-1))
     enddo
     aa(2)=0.0
     bb(2)=3.0*h(1)+2.0*h(2)
     cc(2)=-h(2)
     dd(2)=6.0*((y(3)-y(2))/h(2)-(y(2)-y(1))/h(1))
     aa(n)=-h(n-1)
     bb(n)=3.0*h(n)+2.0*h(n-1)
     cc(n)=0.0
     dd(n)=6.0*((y(n+1)-y(n))/h(n)-(y(n)-y(n-1))/h(n-1))
     call trid(aa,bb,cc,dd,q,n)
     q(1)=q(2)
     q(n+1)=q(n)
     call calqk(tt,qk,t,y,h,q,n,m)
c output the data into a file
     open(unit=06,file='mtax_result1.data')
     do i=1,n+1
     write(6,1) i,q(i)
     enddo
        close(6)
```

```
        open(unit=06,file='mtax_fit.data')
  1   format(i2,e20.10)
      do k=1,kx
        write(6,2) t(k),y(k)
  2     format(f12.8,1x,f12.8)
      enddo
      do k=1,m-1
      write(6,21) tt(k),qk(k)
 21   format(f12.8,1x,f12.8)
      enddo
         do k=kx+1,n+1
         write(6,22) t(k),y(k)
 22      format(f12.8,1x,f12.8)
         enddo
      close(6)
      end

c Tridiagonal linear system
         subroutine trid(aa,bb,cc,dd,yy,n)
         dimension aa(150),bb(150),cc(150),dd(150),yy(150)
         dimension be(152),ar(152)
      double precision aa,bb,cc,dd,yy,be,ar
         be(1)=0.0
         ar(1)=0.0
         do 1 k=2,n
         be(k)=cc(k)/(bb(k)-aa(k)*be(k-1))
 1       ar(k)=(dd(k)+aa(k)*ar(k-1))/(bb(k)-aa(k)*be(k-1))
         yy(n+1)=0.0
         do 2 k=2,n
      j=n+2-k
 2       yy(j)=be(j)*yy(j+1)+ar(j)
         return
         end

      subroutine calqk(tt,qk,t,y,h,q,n,m)
      dimension h(150),tt(150),qk(150),t(150),y(150),q(150)
      double precision h,t,y,q,tt,qk
      do i=1,m
      do 1 k=1,n
      if(tt(i).ge.t(k).and.tt(i).lt.t(k+1))then
         qk(i)= q(k)/6.0*((t(k+1)-tt(i))**3/h(k)-h(k)*(t(k+1)-tt(i)))
     $        +q(k+1)/6.0*((tt(i)-t(k))**3/h(k)-h(k)*(tt(i)-t(k)))
     $        +y(k)*(t(k+1)-tt(i))/h(k)+y(k+1)*(tt(i)-t(k))/h(k)
      endif
 1    continue
      enddo
      return
      end
```

9. gpsfit3.f

```
c This is a correct program  12/21/2000 for case 3.
C
C     NOTE:  qk(x) = q(k)/6.0*[(t(k+1)-t)**3/h(k)-h(k)*(t(k+1)-t)]
C                  +q(k+1)/6.0*[(t-t(k)**3/h(k)-h(k)*(t-t(k))]
C                  +y(k)*[(t(k+1)-t(k))/h(k)]+y(k+1)*[(t-t(k))/h(k)]
C             FOR t(k) < t < t(k+1), k=0,....n-1
C
          dimension h(150),t(150),xx(150),yy(150),zz(150),tt(150)
     dimension qkx(150),qky(150),qkz(150),q(150)
     double precision tt,t,h,xx,yy,zz,qkx,qky,qkz,q,hx,kx,t0
     n=89
     m=10
        t0=0.014035148
        hx=0.0000182
        kx=50
        do i=1,m
        tt(i)=t0+i*hx
        enddo
c input data
     open(unit=05,file='mta1.dat')
     do i=1,n+1
           read(5,*) t(i),xx(i),yy(i),zz(i)
     enddo
     close(5)
     do k=1,n
     h(k)=t(k+1)-t(k)
     enddo
     call cubic_fit(t,xx,q,h,n)
     call calqk(tt,qkx,t,xx,h,q,n,m)
        call cubic_fit(t,yy,q,h,n)
        call calqk(tt,qky,t,yy,h,q,n,m)
        call cubic_fit(t,zz,q,h,n)
        call calqk(tt,qkz,t,zz,h,q,n,m)
c output the data into a file
c    open(unit=06,file='mtax_result3.data')
c    do i=1,n+1
c    write(6,1) i,q(i)
c    enddo
c         close(6)
        open(unit=06,file='mtax_fit_all.data')
        do k = 1,kx
     write(6,2) t(k),xx(k),yy(k),zz(k)
        enddo
        do k = 1,m-1
        write(6,2) tt(k),qkx(k),qky(k),qkz(k)
        enddo
        do k = kx+1,n+1
        write(6,2) t(k),xx(k),yy(k),zz(k)
        enddo
  2     format(f12.8,1x,f12.8,1x,f12.8,1x,f12.8)
     close(6)
        open(unit=06,file='mtax_fit_x.data')
        do k = 1,kx
```

```
      write(6,3) t(k),xx(k)
      enddo
      do k = 1,m-1
      write(6,2) tt(k),qkx(k)
      enddo
      do k = kx+1,n+1
      write(6,2) t(k),xx(k)
      enddo
3     format(f12.8,1x,f12.8)
      close(6)
      open(unit=06,file='mtax_fit_y.data')
      do k = 1,kx
      write(6,4) t(k),yy(k)
      enddo
      do k = 1,m-1
      write(6,4) tt(k),qky(k)
      enddo
      do k = kx+1,n+1
      write(6,4) t(k),yy(k)
      enddo
4     format(f12.8,1x,f12.8)
      close(6)
      open(unit=06,file='mtax_fit_z.data')
      do k = 1,kx
      write(6,5) t(k),zz(k)
      enddo
      do k = 1,m-1
      write(6,5) tt(k),qkz(k)
      enddo
      do k = kx+1,n+1
      write(6,5) t(k),zz(k)
      enddo
5     format(f12.8,1x,f12.8)
      close(6)
      open(unit=06,file='mtax_x.data')
      do k = 1,n+1
      write(6,6) t(k),xx(k)
      enddo
6     format(f12.8,1x,f12.8)
      close(6)
      open(unit=06,file='mtax_y.data')
      do k = 1,n+1
      write(6,7) t(k),yy(k)
      enddo
7     format(f12.8,1x,f12.8)
      close(6)
      open(unit=06,file='mtax_z.data')
      do k = 1,n+1
      write(6,8) t(k),zz(k)
      enddo
8     format(f12.8,1x,f12.8)
      close(6)
   end

   subroutine cubic_fit(t,y,q,h,n)
   dimension h(150),t(150),y(150)
      dimension aa(150),bb(150),cc(150),dd(150),q(150)
```

```
        double precision t,h,y,aa,bb,cc,dd,q
c set up Aq=b
        do k=3,n-1
        aa(k)=-h(k-1)
        bb(k)=2.0*(h(k-1)+h(k))
        cc(k)=-h(k)
        dd(k)=6.0*((y(k+1)-y(k))/h(k)-(y(k)-y(k-1))/h(k-1))
        enddo
c ****************************************
        aa(2)=0.0
        bb(2)=(2.0+h(1)/h(2))*(h(1)+h(2))
        cc(2)=-h(2)+h(1)*h(1)/h(2)
        dd(2)=6.0*((y(3)-y(2))/h(2)-(y(2)-y(1))/h(1))

        aa(n)=-h(n-1)+(h(n)*h(n)/h(n-1))
        bb(n)=2*(h(n-1)+h(n))+h(n)*(h(n-1)+h(n))/h(n-1)
        cc(n)=0.0
        dd(n)=6.0*((y(n+1)-y(n))/h(n)-(y(n)-y(n-1))/h(n-1))

c ************************************************
        call trid(aa,bb,cc,dd,q,n)
c ********************************
        q(1)=((h(1)+h(2))*q(2)-h(1)*q(3))/h(2)
        q(n+1)=((h(n-1)+h(n))*q(n)-h(n)*q(n-1))/h(n-1)
     return
     end


c Tridiagonal linear system
        subroutine trid(aa,bb,cc,dd,q1,n)
        dimension aa(150),bb(150),cc(150),dd(150),q1(150)
        dimension be(152),ar(152)
     double precision aa,bb,cc,dd,q1,be,ar
        be(1)=0.0
        ar(1)=0.0
        do 1 k=2,n
        be(k)=cc(k)/(bb(k)-aa(k)*be(k-1))
 1      ar(k)=(dd(k)+aa(k)*ar(k-1))/(bb(k)-aa(k)*be(k-1))
        q1(n+1)=0.0
        do 2 k=2,n
     j=n+2-k
 2      q1(j)=be(j)*q1(j+1)+ar(j)
        return
        end

     subroutine calqk(tt,qk,t,y,h,q,n,m)
     dimension h(150),tt(150),qk(150),t(150),y(150),q(150)
     double precision h,t,y,q,tt,qk
        do i=1,m
     do 1 k=1,n
     if(tt(i).ge.t(k).and.tt(i).lt.t(k+1))then
        qk(i) = q(k)/6.0*((t(k+1)-tt(i))**3/h(k)-h(k)*(t(k+1)-tt(i)))
     $       +q(k+1)/6.0*((tt(i)-t(k))**3/h(k)-h(k)*(tt(i)-t(k)))
     $       +y(k)*(t(k+1)-tt(i))/h(k)+y(k+1)*(tt(i)-t(k))/h(k)
     endif
 1   continue
        enddo
```

```
return
end
```

10. gpsfit4.f

```
c This is a correct program  5/4/2000 for case 2.
C
C     NOTE:  qk(x) = q(k)/6.0*[(t(k+1)-t)**3/h(k)-h(k)*(t(k+1)-t)]
C                  +q(k+1)/6.0*[(t-t(k)**3/h(k)-h(k)*(t-t(k))]
C                  +y(k)*[(t(k+1)-t(k))/h(k)]+y(k+1)*[(t-t(k))/h(k)]
C             FOR t(k) < t < t(k+1), k=0,....n-1
C
          dimension h(150),t(150),tt(150),qk(150),y(150)
     dimension aa(150),bb(150),cc(150),dd(150),q(150)
     double precision tt,t,qk,h,y,aa,bb,cc,dd,q,hx,kx,t0,s0,sn,h0
     n=96
        m=10
        t0=0.014035148
        hx=0.0000182
        kx=50
     do i=1,m
     tt(i)=t0+i*hx
     enddo
c input data
     open(unit=06,file='mtaxnew.dat')
     do i=1,n+1
           read(6,*) t(i),y(i)
     enddo
     close(6)

c calculate s0,sn
        h0=2.0*hx
        s0=(-3.0*y(1)+4.0*y(2)-y(3))/h0
        sn=(-3.0*y(n+1)+4.0*y(n)-y(n-1))/h0
c set up Aq=b
     do k=1,n
     h(k)=t(k+1)-t(k)
     enddo
     do k=3,n-1
     aa(k)=-h(k-1)
     bb(k)=2.0*(h(k-1)+h(k))
     cc(k)=-h(k)
     dd(k)=6.0*((y(k+1)-y(k))/h(k)-(y(k)-y(k-1))/h(k-1))
     enddo
     aa(2)=0.0
     bb(2)=2.0*h(1)+2.0*h(2)+h(1)/2
     cc(2)=-h(2)
     dd(2)=6.0*((y(3)-y(2))/h(2)-(y(2)-y(1))/h(1))
     $        -3.0*((y(2)-y(1))/h(1)-s0)
     aa(n)=-h(n-1)
     bb(n)=2.0*h(n)+2.0*h(n-1)-h(n)/2
     cc(n)=0.0
     dd(n)=6.0*((y(n+1)-y(n))/h(n)-(y(n)-y(n-1))/h(n-1))
     $        -3.0*(sn-(y(n)-y(n-1))/h(n-1))
     call trid(aa,bb,cc,dd,q,n)
     q(1)=3.0*((y(2)-y(1))/h(1)-s0)/h(1)-q(2)/2
     q(n+1)=3.0*(sn-(y(n+1)-y(n))/h(n))/h(n)-q(n)/2
     call calqk(tt,qk,t,y,h,q,n,m)
```

```
c output the data into a file
      open(unit=06,file='mtax_result4.data')
      do i=1,n+1
      write(6,1) i,q(i)
      enddo
         close(6)
         open(unit=06,file='mtax_fit4.data')
   1  format(i2,e20.10)
      do k=1,kx
        write(6,2) t(k),y(k)
   2    format(f12.8,1x,f12.8)
      enddo
      do k=1,m-1
      write(6,21) tt(k),qk(k)
   21 format(f12.8,1x,f12.8)
      enddo
         do k=kx+1,n+1
         write(6,22) t(k),y(k)
    22   format(f12.8,1x,f12.8)
         enddo
      close(6)
      end

c Tridiagonal linear system
         subroutine trid(aa,bb,cc,dd,yy,n)
         dimension aa(150),bb(150),cc(150),dd(150),yy(150)
         dimension be(152),ar(152)
      double precision aa,bb,cc,dd,yy,be,ar
         be(1)=0.0
         ar(1)=0.0
         do 1 k=2,n
         be(k)=cc(k)/(bb(k)-aa(k)*be(k-1))
 1       ar(k)=(dd(k)+aa(k)*ar(k-1))/(bb(k)-aa(k)*be(k-1))
         yy(n+1)=0.0
         do 2 k=2,n
      j=n+2-k
 2       yy(j)=be(j)*yy(j+1)+ar(j)
         return
         end

      subroutine calqk(tt,qk,t,y,h,q,n,m)
      dimension h(150),tt(150),qk(150),t(150),y(150),q(150)
      double precision h,t,y,q,tt,qk
      do i=1,m
      do 1 k=1,n
      if(tt(i).ge.t(k).and.tt(i).lt.t(k+1))then
         qk(i)= q(k)/6.0*((t(k+1)-tt(i))**3/h(k)-h(k)*(t(k+1)-tt(i)))
      $       +q(k+1)/6.0*((tt(i)-t(k))**3/h(k)-h(k)*(tt(i)-t(k)))
      $       +y(k)*(t(k+1)-tt(i))/h(k)+y(k+1)*(tt(i)-t(k))/h(k)
      endif
 1    continue
      enddo
      return
      end
```

B.2 MADS Part B programs

1. rungps.m
2. rdgps.m
3. calvec.m

1. rungps.m

```
%  This function reads a standard code GPS file by rdgps2.m,
%  uses a formula to calculate velocity vectors and passes
%  back the position at the required times...
%  This function is just like the runephem.m file in mtass.
%  For reading a gps file, only the time and gps_name are needed
%  as input. The unit_number and gps_parm do not need to be
%  included in your function call. For a testing, I used
%  gps_name = 'matx.mat' as a GPS file.
%
%
%  INPUT:
%         time(nx1)          Column array of input times.
%         format             yymmddhhmmss.
%         gps_name           GPS file name.
%
%         gps_parm(16,1)     GPS paramters:
%                            (1,1)=(1=use fitting function;
%             2=use GPS)
%           (2,1)= 0  %Elements Epoch time for GPS is 0
%           (3,1)=(1=use keplarian elements;
%                  2=use pos/vel). Only using 2 for balloon.
%           (4,1)= 0   sugges those values are zero.
%           (5,1)= 0    Those are used for ephemeirs for
%           (6,1)= 0    satellite. Don't use for GPS in
%           (7,1)= 0   balloon case.
%           (8,1)= 0
%           (9,1)= 0
%           (11:13,1)=position vector(km)
%           (14:16,1)=velocity vector(km/sec)
%
%  OUTPUT:
%          r(3xn)            Column array of position vector
%           at required times.
%          v(3xn)        Column array of velocity vector
%           at required times.
%          quality           Quality flag
%           0=ok
%                    >0  GPS read problem.
%           -1= invalid input times.
% MODIFY:    Liping Mo     07/00.
% function [r,v,quality] = rungps(time,gps_name,unit_number,gps_parm)
  function [r,v,quality] =
runephem(time,gps_name,unit_number,gps_parm);
%  gps_name = 'mta1.dat';
  inipos = [0, 0, 0];    %Substitute the initial value for position of
                         %balloon.
  iniTim = 960428.013857148;  %Initial time for a supplied satellite
data
                              %file

  if nargin==2           %input two arguments in function.
    unit_number=10; gps_parm(1,1)=2;
  end;
```

```
  if nargin==3            %input three arguments in function.
    gps_parm(1,1)=2;
  end;


  p=size(time,1); quality = 0;

%%Make sure there are input times available
  if p>0
     r1=zeros(1,3); %change row to p rows
     v1=zeros(1,3);
     r=zeros(3,p);  %Initial the r and v, so the size of r, v
     v=zeros(3,p);  %are same as runephem.m in MTASS.



% open GPS file here....
  [r1,t1,ierr] = rdgps(2,gps_name,10,gps_parm(1,1));
%% Ensure the GPS file can be opened sucessfully....
   if ierr == 0  %read GPS file here.....
     i = 2;    % i is a integer number for start to read the line of
file
     a = i;    %a is a integer number for while loop useing
     while i < 200+a, i = i+1;
      [r1,t1,ierr] = rdgps(i, gps_name, 10, gps_parm(1,1));
                                          %Using integer to call
rdgps2.m
                                          %The integer is number of
line in
                                          %gps data


      %%put r1(px3) to r(3xp)
              for j = 1:3
              r(j,i-a) = r1(1,j);
              end;
     delT = t1 - iniTim;
          [v1] =calvel(inipos,r1,iniTim,t1);
          %%put v1(px3) to v(3xp)
              for j = 1:3
              v(j,i-a) = v1(1,j);
              end;
          inipos = r1;
          iniTim = t1;
          quality = quality + ierr;
    end;            %end for while loop
  else
  quality = ierr;
  end;                 %end for if ierr==0

 else
    quality = -1;
 end;  %End for p>0 block if....
```

2. rdgps.m

```
%This function reads in gps data from gps.txt.
%  Input:
%           line(1x1):  line number for read GPS data.
%           gps_parm(16,1)     GPS paramters:
%           GPS_name:          gps files name. (like gps_txt)
%  output:
%           gpspos(nx3):       subset of gps data for position vector.
%           gpstim(nx3):       subset of gps data for time vector.
%           ierr       :       = 0 ok to read GPS file.
%                              >0  Problem to read GPS file.
%function [gpspos,gpstim,ierr] =
rdgps1(line,gps_name,unit_number,gps_parm)
 function [gpspos,gpstim,ierr] =
rdgps2(line,gps_name,unit_number,gps_parm);

 if nargin==2         %input two parmeters for function.
 unit_number=10; gps_parm(1,1)=2;
 end;
 if nargin==3         %input three parmeters for function.
   gps_parm(1,1)=2;
 end;
 if nargin==4         %input four parmeters for function.

      [fid,message] = fopen(gps_name);
      if fid ~=-1
         n = line;
         stude = fseek(fid,85*(n-1),-1);
         position=ftell(fid);
         a = fscanf(fid,'%g %g %g',[4,1]);
         b = a';
         %get gps position from gps file(2,3,4) colcumn
           for i = 1:3,
           gpspos(1,i) = b(1,i+1);
           end;
         %get gps time from gps flie(1) colcumn.
         gpstim(1,1) = b(1,1);
         ierr = 0;
         else
         disp('PROBLEM READING GPS DATA!');
         disp(message);
         ierr = 1;
      end;
end;
```

3. calvec.m

```
%THis function calculates the velocity vector using given
%initial values of position vector and time.
%    Input:
%                posit1(1x3):  initial position vector by user given.
%                posit2(1x3):  The position vector read from GPS data.
%                time1(1x1):   initial time by user given.
%                time2(1x1):   The time read from GPS data.
%    Output:
%                Vel(1x3):      velocity vector on time2.
%function [Vel] = calvel(posit1,posit2,time1,time2)
function [Vel] = Calve2(posit1,posit2,time1,time2);
  for i = 1:3,
  deltpos(1,i) = posit2(1,i) - posit1(1,i);
  deltTim = time2 - time1;
  Vel(1,i) = deltpos(1,i)./deltTim;
  end;
%  disp('The velocity is ');
%  disp([Vel]);
```

REFERENCE

1. *V. Johnson, M. Woodard,* Multimission(1994) Three-Axis stabilized Spacecraft (MTASS) Flight Dynamic Support System (FDSS) Mathematical Background GSFC/Code 553.

2. *V. Johnson, M. Woodard,* (1995)Multimission Three-Axis stabilized Spacecraft (MTASS) Flight Dynamic Support System (FDSS) Function Specifications GSFC/Code 553.

3. Matlab Attitude Determination System Function Specifications.1996 NASA.

4. Matlab Attitude Determination System User Guide. NASA 1996.

5. *D. Brasoveanu, J. Hashmall,* (1993) Fixed-Head Star Tracker Performance study GSFC/Code 553

6. *G.S.Da Costa,* (1992) Basic Photometry Techniques. ASP Conference Series, Vol. 23

7. *N. Zotov,* (1998) Selection of a Lens for Night and Day Balloon Observing and Automated Star Identification. Technical Report.

8. *M. Kendall, K. Stuart,*(1977) The Advanced Theory of Statistics, Vol.1: Distribution Theory, 4th edition.

9. *P.B. Stetson,* (1979) AJ,84,1056

10. *M.J. Maron,* (1982) Numerical Analysis A Practical Approach. Macmillan Publishing Company New York.

11. *Peter Duffett-smith* (1979) Practical Astronomy with Your Calculator, Cambridge University Press, Cambrige.

12. *Richard L. Burden, J. Douglas Faires* . (1995) Numerical Analysis PWS Publishing Company. Boston.

13. *Burden Faires* (1993) Numerical Analysis. IPT Publishing.

14. *Cont,S.D., C. DeBoor*. (1980) Elementary Numerical Analysis: An Algorithmic Approach,3rd ed. McGraw-Hill