

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

UMI[®]



ROBOTIC TELESCOPE STAR ACQUISITION SYSTEM

by

Robert William Jara Vélez

A thesis Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCES

in

PHYSICS

UNIVERSITY OF PUERTO RICO

MAYAGUEZ CAMPUS

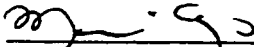
December, 2000

Approved by:



Luis M Quiñones, Ph.D.
Member, Graduate Committee

14 Dec 2000
Date



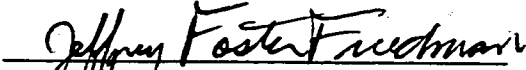
Moisés Orengo, Ph.D.
Member, Graduate Committee

14/Dec/2000
Date



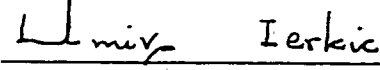
Leszek Nowakoski, Ph.D.
Member, Graduate Committee

13 Dec. 2000
Date



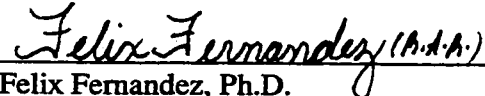
Jeffrey Friedman, Ph.D.
President, Graduate Committee

14 Dec. 2000
Date



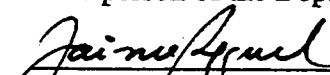
Mario Lerkic, Ph.D.
Representative of Graduate Studies

12/13/00
Date



Felix Fernandez, Ph.D.
Chairperson of the Department

14/12/00
Date



Jaime Seguel, Ph.D.
Director of Graduate Studies

18/Dec/2000
Date

UMI Number: 1403048

UMI[®]

UMI Microform 1403048

Copyright 2001 by Bell & Howell Information and Learning Company.

All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

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

ABSTRACT

An algorithm that reads data from the USNO-SA2.0 Star Catalog (USC) for any particular star field of the sky was developed. These data was used like input data, as much as the output data from the telescope CCD camera, in a pattern-recognition algorithm (PRA). The output data from PRA is the deviation error between current astronomical coordinates, to where the telescope is currently pointing, and the input astronomical coordinates (in theory both are the same). The CCD data was simulated by the computer from the USC by introducing noise and offsets. The algorithms are in the C++ language.

RESUMEN

Se desarrolló un algoritmo que lee la información almacenada en USNO-SA2.0 Star Catalog (USC), para cierta porción particular del cielo. Esta información, conjuntamente con la información de la cámara CCD del telescopio, es usada en el algoritmo de reconocimiento de patrón (PRA) con la finalidad de calcular el error de desviación que hay entre las coordenadas astronómicas hacia donde el telescopio apunta y las coordenadas hacia donde éste realmente debería apuntar (en teoría ambas son iguales). La información de la cámara CCD fue simulada por computador a partir de la información real proveniente del USC, introduciéndole ruidos y desviaciones. Los algoritmos están programados en lenguaje C++.

To
my **FAMILY** and **FRIENDS**
who are always close to me
even though the wild distance and the wild sea
separate us

To
my **ANCESTORS**
whose cultural legacy is even
infinite question
to genius and erudites

To
my **PERU**
who although is falling
it is always towering to shoulder of my ancient race

To
BORINQUEN
land of the coquí
white pillow of my dreams

To
the **HUNDREDS OF STRANGER SMILES**
that I kidnapped
and took to me...
because they are my only inspiration in my loneliness

*Thank you to
everybody
who gave me just a little bit of their time*

*An specially
Thanks to
DR. Jeffry Friedman*

TABLE OF CONTENTS

LIST OF TABLES.....	viii
LIST OF FIGURES.....	ix
LIST OF APPENDICES.....	x
CHAPTER I. INTRODUCTION.....	1
CHAPTER II. LITERATURE REVIEW.....	3
2.1 Star Acquisition from Star Catalog.....	3
2.2 Star Acquisition from the CCD camera.....	5
2.3 Pointing the telescope.....	5
CHAPTER III. THE REAL PROBLEM.....	7
3.1 Formulation.....	7
3.2 Solution.....	9
CHAPTER IV. THE MATHEMATICAL PROBLEM.....	13
4.1 Formulation.....	13
4.2 Numerical Solution.....	15
CHAPTER V. THE GENERAL SYSTEM.....	18
5.1 Structured Data.....	18
5.2 Design of Windows.....	21
5.3 The Source Program Code.....	24
CHAPTER VI. THE PATTERN RECOGNITION ALGORITHM.....	25
6.1 Data from USC	25
6.2 Data from FITS.....	28
6.3 The Pattern Recognition Algorithm.....	29
6.3.1 The Fist Solution.....	29
6.3.2 The Possible Solutions.....	30
6.3.3 The Solutions and the Best One.....	33

CHAPTER VII. RESULTS.....	36
CHAPTER VIII. DISCUSSION.....	39
CHAPTER IX. CONCLUSIONS AND PERSPECTIVES.....	40
BIBLIOGRAPHY.....	107

LIST OF TABLES

Table 1. Corrections on RA and DEC to several cases.....36

LIST OF FIGURES

Figure 1. CCD's screen.....	8
Figure 2. USC's screen	9
Figure 3. USC's screen-1.....	10
Figure 4. CCD's screen-1.....	11
Figure 5. CCD and USC screens superimposed.....	11
Figure 6. R and T superimposed.....	14
Figure 7. The best and worst solution of the problem.....	15
Figure 8. System structured diagram.....	18
Figure 9. System flow char.....	19
Figure 10. Input data window.....	21
Figure 11. Menu window.....	21
Figure 12. Window of the List of stars from USC in normal units.....	22
Figure 13. Window of the list of star from USC in pixel units.....	22
Figure 14. Window of the list of stars from CCD in pixel units.....	23
Figure 15. Window of the corrections on RA and DEC.....	23
Figure 16. Graphical representation of zonexxx.cat files and possible situations of screen's catalog.....	25
Figure 17. Relation between first solution and exact solution.....	30
Figure 18. Real comparison made for pattern-recognition().....	31
Figure 19. Vicinity of possible solutions.....	32

LIST OF APPENDICES

APPENDIX A.....	41
The Right Ascension and Declination.....	41
The USNO-SA2.0 Star Catalog (USC).....	42
The Flexible Image Transport System (FITS).....	44
APPENDIX B.....	45
The C++ Program.....	45

CHAPTER I.

INTRODUCTION

The first telescope was pointed to the sky by hand. This was very difficult. Let's imagine moving a telescope by hand at all times in order to keep a star within of field of view (FOV). At the same time astronomers had to locate with their own eyes the desired star on the sky, to then be able to point the telescope at it. There was no other way. This meant it was necessary to know the sky very well.

Many years later the first motor was coupled to a telescope. The motor automatically moved the telescope and tracked the stars, right after it was pointed. This mechanism avoided the moving star problem only in part, because the motors had low accuracy.

The next advance happened when the computer was introduced to solve the telescope positioning problem. The initial positioning of the telescope was completely automated. It was just necessary to input the astronomical coordinates of the position wanted in the computer, so the computer and the motors would do the rest.

Nevertheless, astronomers were not satisfied. Tracking became as exact as the data from the star catalog could be, but the struggle between the astronomer and accuracy had not ended.

At the same time, the video camera had been introduced to the systems. A portion of the sky embraced by the telescope viewer was displayed on a screen. In this way the astronomer could comfortably see the sky. All the information about the stars on the screen was saved digitally.

This information would later be compared with the data from the star catalog so the position of the telescope could be verified.

Many astronomical observatories in the world that have automated their telescopes have made a program similar to the one that has been developed for this project. But, at the same time, these programs are quite different because each observatory adjusts its programs to its own needs, and uses also the programming language more convenient for them. The problem consist in each programmer designing their source-programs in a different way, such that each source-program

looks original compared to another even when the entries and output data are quite similar. On the other hand, to get the source program from someone else would be very difficult and expensive, given that it is hard to understand the logic of other programmers, even when the program becomes very short and the documentation rather extensive.

This research deals with the development of software to calculate the deviation error between real astronomical coordinates (where the telescope is pointing) and theoretical astronomical coordinates (where the telescope should be pointing).

Before doing this it is necessary to read data from USNO-SA2.0 Star Catalog (USC) for some portion of the sky. This portion is centered on some desired astronomical coordinates. The data obtained from USC is used to get the simulated data of the CCD. After this, one is ready to try to find the deviation error because these data are the input data to the Pattern Recognition Algorithm (PRA). The data from USC is also important because one can find all the information about any stellar objects. (right ascension, declination, red magnitude, blue magnitude, etc.)

To use the calculated corrections in a real application it is necessary to create another program that would send a command to move the telescope, and check the astronomical coordinates. All this is only possible if the telescope has a computer system, a CCD camera system, and motors included.

In theory, the PRA can be used for any telescope system. It is important to keep in mind that each telescope system has its own specifications, like the focal length of the telescope, the aperture and the dimensions of pixels per degree of the CCD camera. Keep in mind the PRA was made for reading data from the USC.

The PRA does not work for sky sections which are too close to the celestial poles (1 degree). On the other hand, the sky section should not be greater than one degree of width and height to get good accuracy. The problem is that if the portion of the sky is bigger than this then the portion of the sky departs from the plane, and therefore the subroutine to get the coordinates in pixels is not appropriate.

CHAPTER II.

LITERATURE REVIEW

There has been a lot of work on specialized software to reduce the pointing error deviation of telescopes, as well as to retrieve stellar data from Star Catalogs. Discussing all of them would be very time consuming and generally impossible, because almost all major (and many minor) observatories around the world have some software to do this, each crafted to their specific environment and needs. There are several companies developing special software for their needs but most are done in house.

2.1 Star Acquisition from Star Catalog.

The Navy Observatory Flagstaff station (NOFS) was established in 1955, in Arizona. Among other things the NOFS is dedicated to conduct a research program to improve the observational methods and the accuracy of astronomical data required by the Navy and other components of the Department of Defense.

The NOFS has developed the software to get data from the USNO-A2.0 for any portion of the sky. This program is working on line on the Internet; so one can get the stellar data whenever one wants and from any geographical location where one is.

This program is flexible in the following aspects

- To access the Star Catalogs: USNO-SA2.0, ACT, and USNO-A2.0 + ACT, together.
- To get the pixel information about the position of the stellar objects on the plates.
- To choose the magnitude wanted: red or blue.
- To input the size of the portion of the sky in arcseconds, arcminutes or degrees.
- To choose the sort of listing (increasing): it could be by RA, DEC, Magnitude (order), Color (red or blue) or Distance to center (of the field of view).
- There are other options (e.g: get data in JPEG format, get data in FITS format, etc).

The program gets data of a rectangular field no greater than 2 degrees (7200 arcsec) in RA and DEC.

The Lowell Observatory (LO) was the first astronomical observatory in Arizona. It was founded in 1894 by Dr. Percival Lowell a mathematician and amateur astronomer from Massachusetts. The place was chosen because of its high elevation and its dark skies. Although Lowell Observatory was founded primarily to explore the possibility that intelligent life might exist on Mars (and was the site of the discovery of Pluto), the Observatory's research quickly expanded into other areas. One can say: the mission of Lowell Observatory is to pursue the study of astronomy, especially the study of our Solar System and its evolution; to conduct pure research in astronomical phenomena; and to maintain quality public education and outreach programs to bring the results of astronomical research to the general public.

The Lowell Observatory's software gets data from one of three Star Catalogs for any portion of the sky. This program is also working on line on the Internet, like the one NOFS. It is flexible in the following aspects:

- To access three Star Catalogs: USNO-SA2.0, USNO-A2.0 and PMM
- To choose the limits to the red magnitude and the blue magnitude at the same time.
- To choose the limits of color index (blue magnitude - red magnitude)
- To choose the shape of the sky portion: rectangular or circular
- To choose the maximum surface density (number of objects per square degree)
- To sort the list by: RA, DEC, Magnitude (Blue or Red), Angle, Radius, and Color Index

The greatest difference between LO program and NOFS program is that we can input the focal distance of the telescope and the pixel size of the CCD in the first one. So, the data request from USC corresponds automatically to FITS data. In PRA this is also taken into account.

The programs developed by the NOFS and Lowell Observatory to get the list of stars are discussed here because these allows to achieve our main objective: to calculate the error deviation of the coordinates of the telescope from the input coordinates.

2.2 Star Acquisition from the CCD camera.

There is a lot of software developed in order to manipulate data from FITS (Flexible Image Transport System). These were and are written to provide a powerful yet simple interface for accessing the FITS files that insulates the programmer from having to deal directly with the complicated internal details of the FITS file.

The High Energy Astrophysics Science Archive Research Center (HEASARC)

It has developed software for analyzing FITS files, but is specialized within the domain of high energy astrophysics.

The software called Fitsio can perform various operations on FITS files, such as open and close files; read, write, modify or delete the values of header keywords; and read or write any element of the associated data array or table. It can check if the FITS files given have a valid FITS format. It was originally written in Fortran-77 in 1991, but in 1996 was developed in ANSI-C as well.

The software called Fv is a general-purpose FITS file editor (formerly just a file viewer) able to manipulate virtually all aspects of a FITS file and perform basic data analysis of its content.

These software packages do not create files in formats required by this project. In order to get the main objective of this research it is necessary to get specific data from the FITS: (RA, DEC, MAG) where both RA and DEC are in pixel units. So, it is necessary to develop some program to do that. This program will have to be integrated into the PRA, in order to replace the subroutine getccddata() which simulates the CCD data.

2.3 Pointing the telescope

At first it is necessary to know the RMS (root-mean-square) pointing accuracy of the telescope. The RMS pointing is a statistical measurement of how accurately a telescope points to *any location* in the sky from *any other* location. It can be determined through a mapping process, where the telescope is pointed to a number of known positions (stars) and the difference

between where the telescope lands and the actual position of the star is recorded. This information is useful because allows us to put a precision limit when the error calculation is carried out.

-Tpoint is a software package developed by Bisque Software Company. It's a professional product and has been used in many telescopes around the world.

-Teledyne Brown Engineering, developed the StarView software package to improve the telescope pointing. It enables faster target acquisition and with greater precision for Space Shuttle astronomy missions.

It works by comparing positions of observed stars with those of "ideal" guide stars. It has a star catalog database used as reference. The telescope pointing is adjusted manually until the observed and cataloged guide stars coincide. The telescope therefore just needs to have a fine control adjustment. That allows super-fine pointing accuracy: it was proven in operation on NASA's Astro-2 Space Shuttle Mission.

It would be interesting if that adjustment could be made automatically. The PRA calculates the error deviation, and therefore the corrections. These corrections will be done automatically in the near future by including a command code generating subroutine.

The software packages described above can point with 1-2 arcsec RMS for large professional telescopes and, 30 arcsec for amateurs.

To talk about the source code and analyze it and compare them with PRA is not possible because they are not available. The software works like black box. One can only know what they do in general.

CHAPTER III.

THE REAL PROBLEM

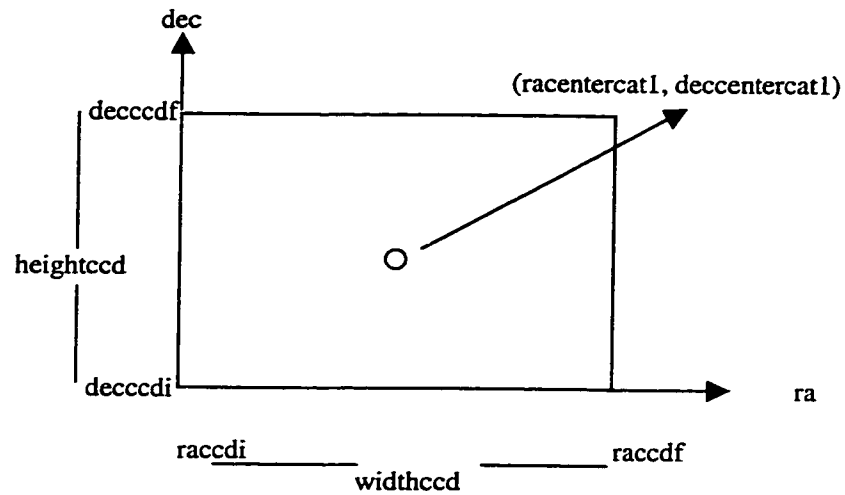
3.1 Formulation

When the telescope is pointing to some point in the sky ($racentercat1$, $deccentercat1$) (see figure 1) the CCD camera catches all stellar objects (SO) that are in its FOV. The information about each SO is saved in the FITS. For this, the CCD camera is considered like a two-dimensional array, whose horizontal axis represents the RA and vertical axis represents the DEC. So each pixel on the CCD camera can be represented by its position ($raccd$, $decccd$) and its magnitude ($magccd$), which is calculated from its intensity.

Given that telescope motor is not accurate, the point where the telescope is pointing ($racentercat1$, $deccentercat1$) at differs a little bit from the desired point ($racentercat$, $deccentercat$).

$$\begin{aligned} racentercat1 &= racentercat + desvra \\ deccentercat1 &= deccentercat + desvdec \end{aligned}$$

Therefore, the problem is:
find the deviations ($desvra$, $desvdec$)



heightccd : height of ccd camera field
 widthccd : width of ccd camera field
 raccdi : inicial right ascension of ccd camera
 raccdf : final right ascension of ccd camera
 decccdi : initial declination of ccd camera
 decccdf : final declination of ccd camera
 (racentercat1, deccentercat1): astronomical coordinates
 where the CCD's screen is centered

Figure 1. CCD's screen

Observations:

1. Take into account that each (raccd, decccd) has been affected by deviations in RA and DEC.

$$\text{raccd} = \text{raccd} + \text{desvra}$$

$$\text{decccd} = \text{decccd} + \text{desvdec}$$

2. Take into account that each (raccd, decccd) include a little perturbation in RA, DEC and MAG because of limitations and physical defects of CCD camera itself.

That is:

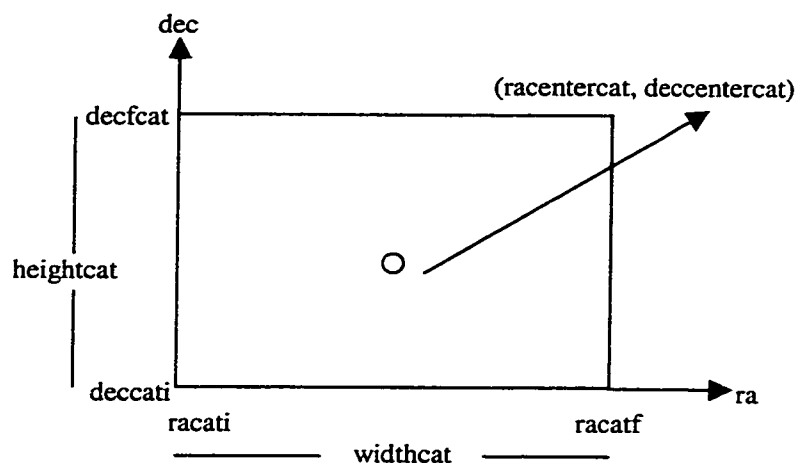
$$\text{raccd} = \text{raccd} + \text{perturba}$$

$$\begin{aligned} \text{decccd} &= \text{decccd} + \text{perturbdec} \\ \text{magccd} &= \text{magccd} + \text{perturbmag} \end{aligned}$$

3. Take into account that the position of the SO obtained from the CCD camera is the actual position (apparent position) while in the USC is the J2000 position.

3.2 Solution

At first, to get the solution to this problem it is necessary to have some reference pattern (theoretical values); otherwise, it would not be possible to get the solution. These references are called "star catalogs"; which are always useful in solving astrometric problems like this. Here the USC is used.



heightcat : height of catalog portion
 widthcat : width of catalog portion
 racati : initial right ascension of catalog portion
 racatf : final right ascension of catalog portion
 deccati : initial declination of catalog portion
 deccatf : final declination of catalog portion
 (racentercat, deccentercat): astronomical coordinates
 where the USC's screen is centered

Figure 2. USC's screen

1. To read SO's information from USC, taking into account observation (3), It means upgrade data from USC for:

$$\text{racati} \leq \text{racat} \leq \text{racatf}$$

$$\text{deccati} \leq \text{decat} \leq \text{deccatf}$$

From Figure 2:

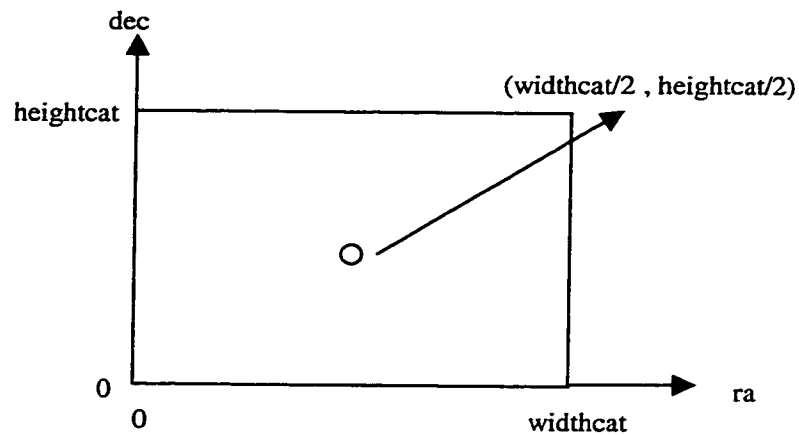
$$\text{racati} = \text{racentercat} - \text{widthcat}/2$$

$$\text{racatf} = \text{racentercat} + \text{widthcat}/2$$

$$\text{deccati} = \text{deccentercat} - \text{heightcat}/2$$

$$\text{deccatf} = \text{deccentercat} + \text{heightcat}/2$$

2. Taking into account the CCD camera scale (pixel/degree) and considering as origin (0, 0) of both CCD's screen and USC's screen at the left lower corner, transform its own SO's information to pixel units with respect to this origin (see figure 3 and 4).



heightcat : height of catalog portion
widthcat : width of catalog portion

Figure 3. USC's screen-1

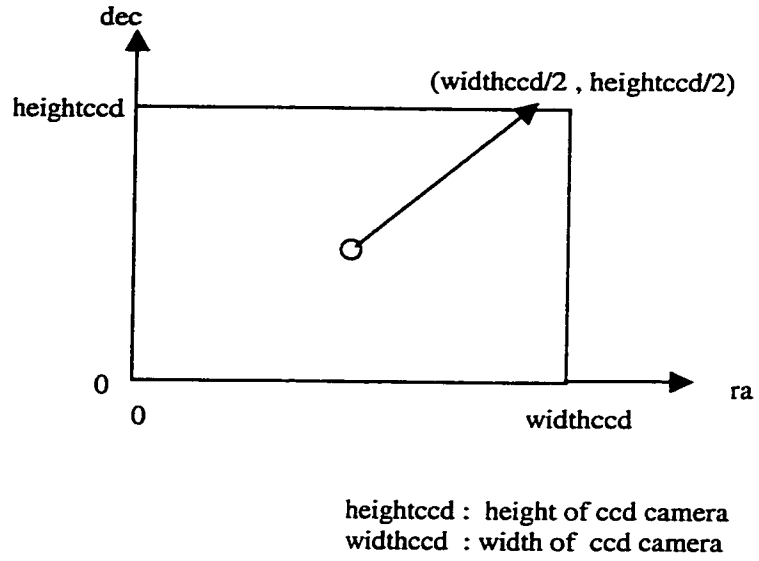


Figure 4. CCD's screen-1

3. So when both screens are in the same units (pixel) find where is the CCD's screen inside of USC's screen (see figure 5).

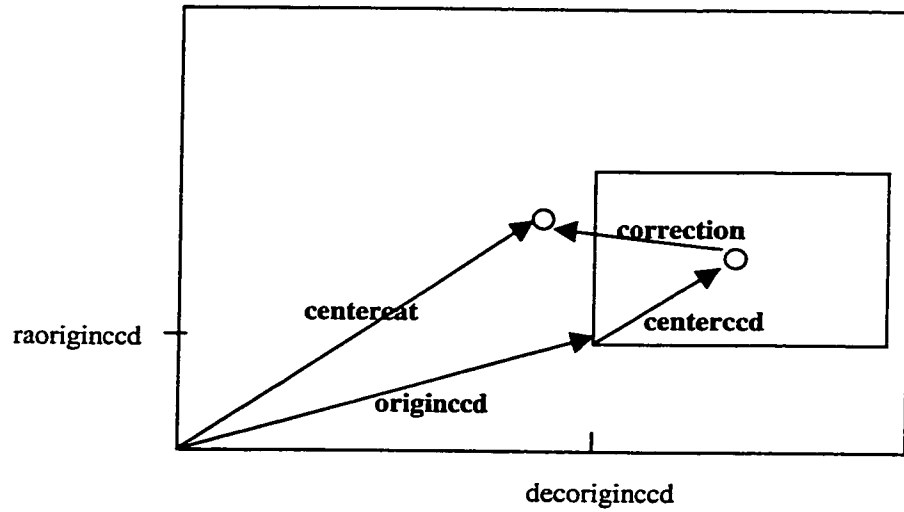


Figure 5. CCD and USC screens superimposed

Therefore:

$$\mathbf{originccd} = (\mathbf{raoriginccd}, \mathbf{deoriginccd})$$

4. Finally calculate the correction (see figure 5).

$$\mathbf{correction} = \mathbf{centercat} - (\mathbf{centerccd} + \mathbf{originccd})$$

and since

$$\mathbf{correction} = (\mathbf{racorrec}, \mathbf{deccorrec})$$

$$\mathbf{originccd} = (\mathbf{raoriginccd}, \mathbf{deoriginccd})$$

$$\mathbf{centercat} = (\mathbf{widthcat}/2, \mathbf{heightcat}/2)$$

$$\mathbf{centerccd} = (\mathbf{widthccd}/2, \mathbf{widthccd}/2)$$

then

$$\mathbf{racorrec} = (\mathbf{widthcat} - \mathbf{widthccd})/2 - \mathbf{raoriginccd}$$

$$\mathbf{deccorrec} = (\mathbf{widthcat} - \mathbf{widthccd})/2 - \mathbf{deoriginccd}$$

5. To convert the correction to degrees, just use the appropriate scale (pixel/degree) from CCD camera.

CHAPTER IV.

MATHEMATICAL PROBLEM

4.1 Formulation

Let the set $R = \{ r \text{ coordinate pairs } (x, y) /$

$$0 \leq x \leq lx$$

$$0 \leq y \leq ly$$

$$x, y, lx, ly \in N)$$

Let the set $S = \{ (x, y) \in R /$

$$x_i \leq x \leq x_f, \quad x_i < x_f, \quad 0 \leq x_i \text{ and } x_f \leq lx$$

$$y_i \leq y \leq y_f, \quad y_i < y_f, \quad 0 \leq y_i \text{ and } y_f \leq ly \}$$

Let the set $T = \{ t \text{ coordinate pairs } (u, v) /$

$$u = x - x_i$$

$$v = y - y_i \quad \forall (x, y) \in S \}$$

therefore

$$0 \leq u \leq lu$$

$$0 \leq v \leq lv \quad \text{where} \quad lu = x_f - x_i$$

$$lv = y_f - y_i$$

So, the *problem* is (see figure 6):

Given R, r, lx, ly

and T, t, lu, lv ; calculate x_i and y_i

where (x_i, y_i) is the origin of T relative to R

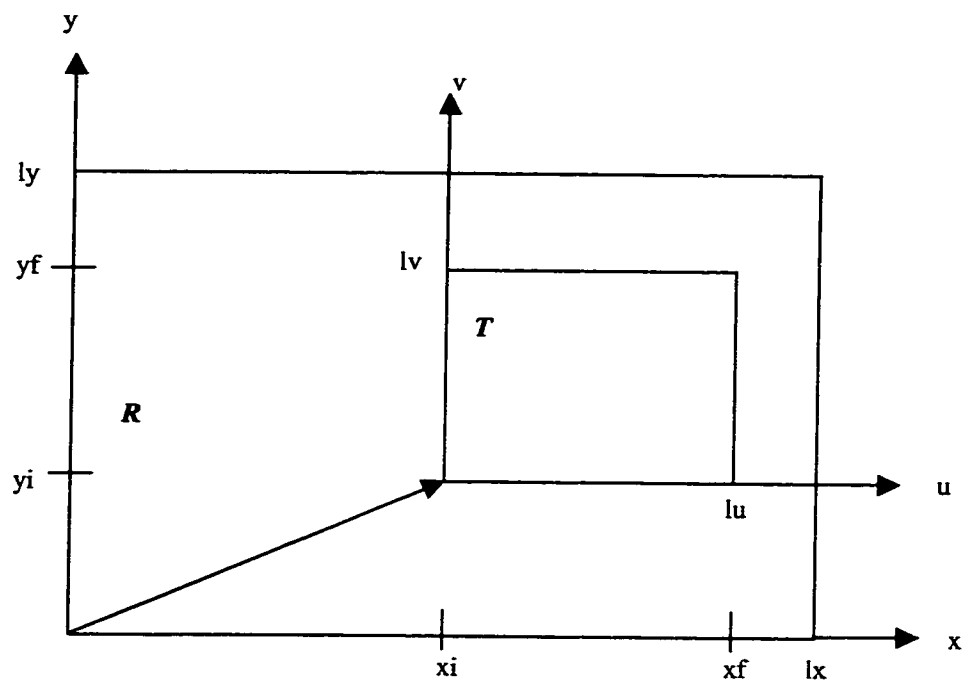


Figure 6. R and T superimposed

Assumptions:

1. It is unknown that $(x_j, y_j) \in R$ correspond to $(u_k, v_k) \in T$
2. The $(u_j, v_j) \in T$ has been effected or distorted by little perturbations p_u and p_v . That is:

$$u_j = u_j + p_u$$

$$v_j = v_j + p_v$$

4.2 Numerical Solution.

If assumptions 1 and 2 are not made the analytic solution would be obvious. To get it would be enough to know just one coordinate $(x, y) \in R$ and its corresponding $(u, v) \in T$

The numerical solution is the alternative due to assumptions 1 and 2.

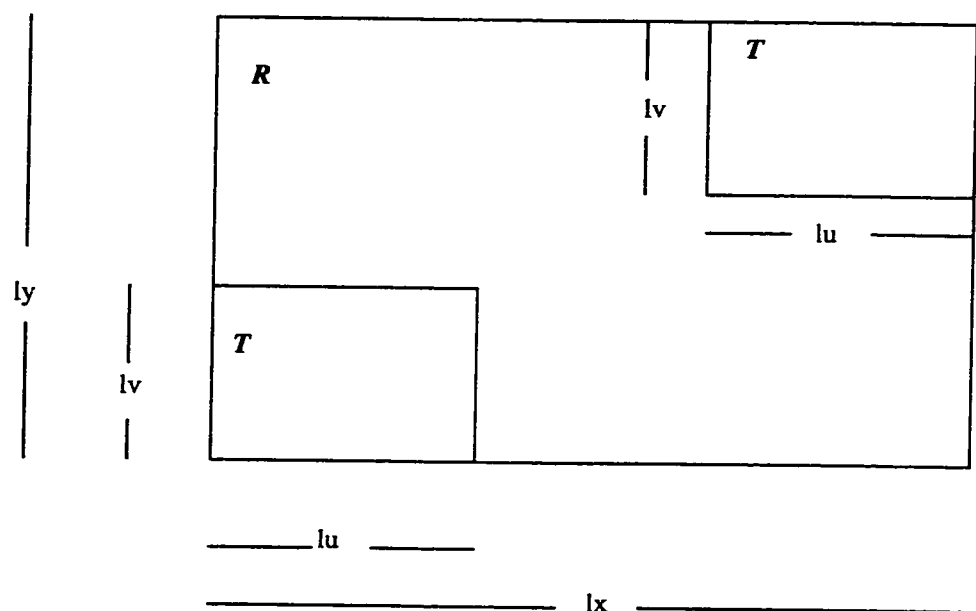


Figure 7. The best and worst solution of the *problem*

Looking at Figure 7, how many possible solutions (N) are there? That is, how many places of R could T possibly be in?

Since T is required to be within R (see **formulation** section) then:

$$x_i : [0, lx - lu]$$

$$y_i : [0, ly - lv]$$

Therefore

$$N = (lx - lu + 1) (ly - lv + 1)$$

There is no a priori judgment to take some particular (x_i, y_i) as solution over others one. Each possible solution has the same probability to be the solution. So, one is free to choose the order to try each possible solution.

Here is the *chosen order*:

$$\begin{aligned} x_i &= 0 \\ y_i &= 0, 1, 2, \dots, ly - lv \end{aligned}$$

$$\begin{aligned} x_i &= 1 \\ y_i &= 0, 1, 2, \dots, ly - lv \end{aligned}$$

$$\begin{aligned} x_i &= 2 \\ y_i &= 0, 1, 2, \dots, ly - lv \end{aligned}$$

...

...

$$\begin{aligned} x_i &= lx - lu \\ y_i &= 0, 1, 2, \dots, ly - lv \end{aligned}$$

From Figure-7, considering the time of calculation, we can see that the best and worst solutions are respectively

$$(x_i, y_i) = (0, 0)$$

$$(x_i, y_i) = (lx - lu, ly - lv)$$

The next steps are:

1. To make

$$T = \{ (u, v) / \begin{array}{l} p_u \leq u \leq lu - p_u \\ p_v \leq v \leq lv - p_v \end{array} \}$$

This redefinition of T avoids the coordinates pair which get out of T because of assumption 2.

2. To guess which (x_i, y_i) is the solution

3. To built

$$S_I = \{ \text{all } (x, y) \in R / \begin{array}{l} x_i \leq x \leq x_f, x_f = x_i + l_u \\ y_i \leq y \leq y_f, y_f = y_i + l_v \end{array} \}$$

4. To built

$$T^l = \{ (u^l, v^l) / u^l = x - x_i, v^l = y - y_i, \forall (x, y) \in S_I \}$$

number of elements of T^l is t^l

Because of step 1 t^l must be greater or equal than t in order for (x_i, y_i) to still be a possible solution.

5. To compare

each $(u, v) \in T$ to each $(u^l, v^l) \in T^l$

taking into consideration assumption 2

And if

$$\text{for each } (u, v) \in T \text{ there exist some } (u^l, v^l) \in T^l / \begin{array}{l} |u - u^l| \leq p_u \\ |v - v^l| \leq p_v \end{array}$$

then (x_i, y_i) is the first solution. (there could be several solutions which meet the same requirements)

6. If the first (x_i, y_i) is found then try from steps 1 to 5 for all possible solutions which are in the vicinity of (x_i, y_i) . The vicinity is defined by " p_u " and " p_v ". Let "n" is the number of solutions found in the vicinity.
7. To choose the best one from the "n" solutions; that is, the solution with a lesser standard deviation.
8. If N possible solutions were tried and "n" is zero, it could be because:
 - The perturbations p_u and p_v considered in step 5 are not appropriate: they are smaller than the real perturbation on the CCD camera.
 - Part or whole T is out of R
 - T really is not related to R

5.1 Structured System

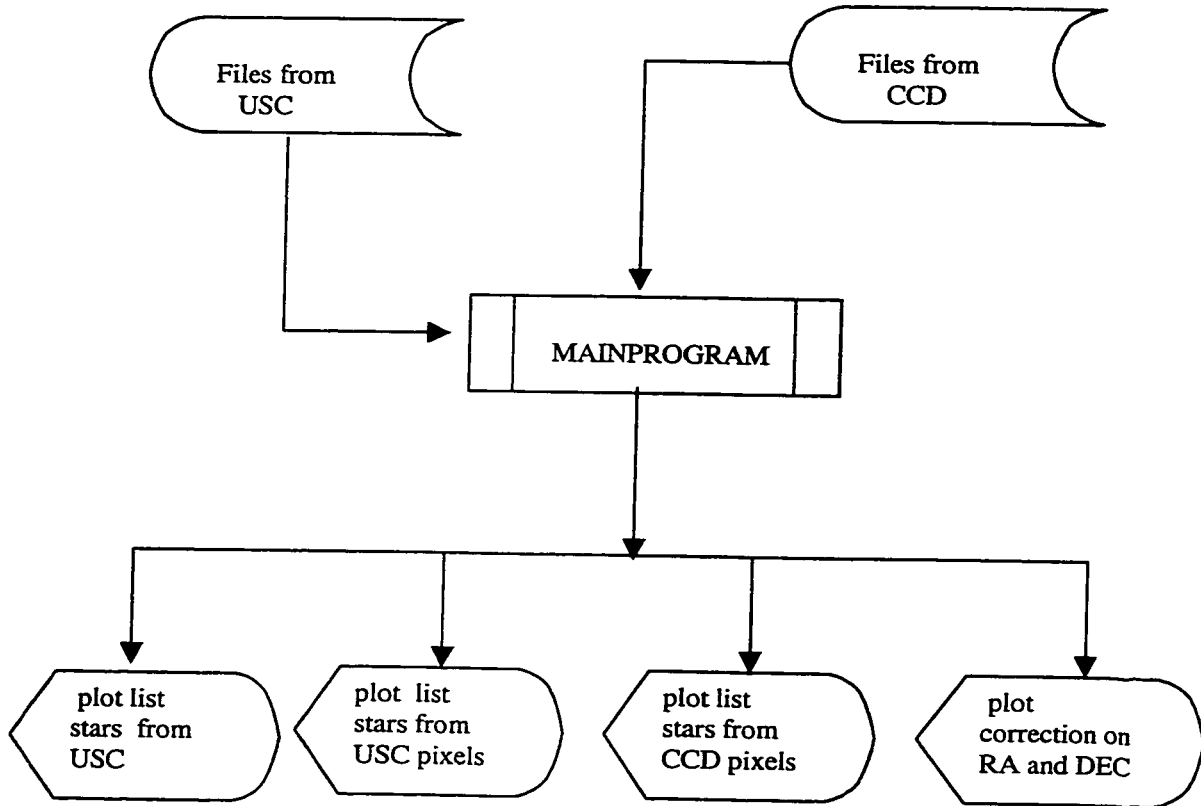
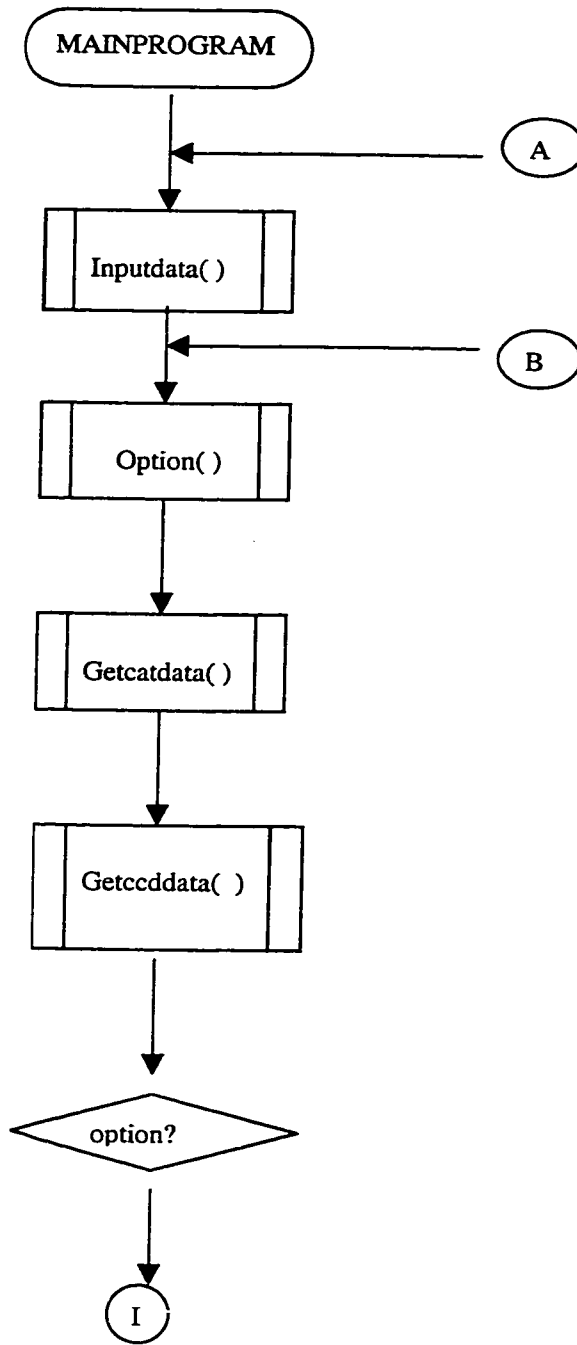


Figure 8. System structured diagram



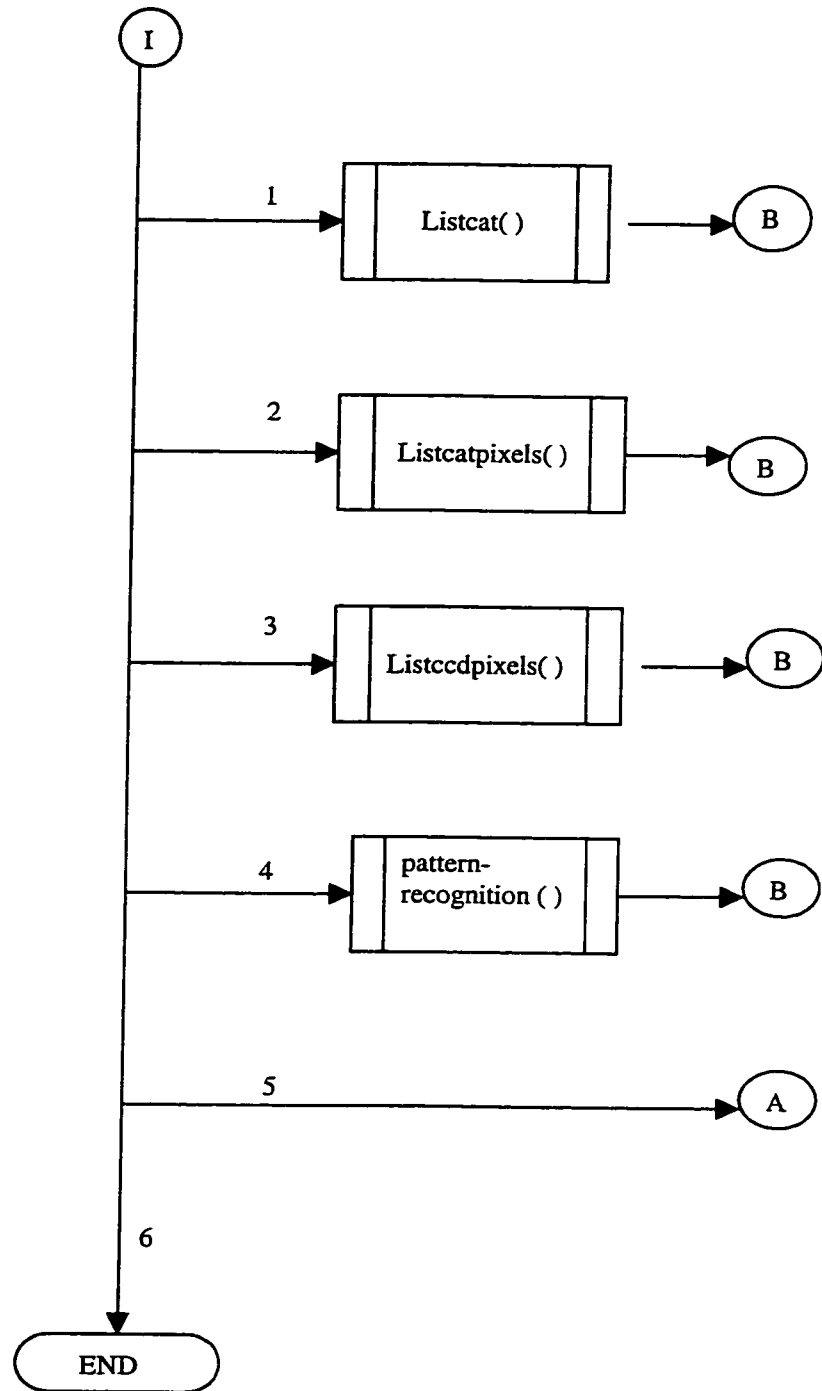


Figure 9. System flow char

5.2 Design of Windows

Our program communicates with the user by displaying the following windows:

pixel per degree	:	
width of CCD camera (arcsec)	:	
height of CCD camera (arcsec)	:	
right ascension (hr min sec)	:	----
declination (deg min sec)	:	----
minima red mag	:	--
maxima red mag	:	--
minima blue mag	:	--
maxima blue mag	:	--
width of catalog (arcsec)	:	--
height of catalog (arcsec)	:	--

Figure 10. Input data window

[1]: List of the stars from catalog	
[2]: List from catalog in pixels	
[3]: List from ccd in pixels	
[4]: Calculate deviation	
[5]: Begin again	
[6]: Quit	
enter your option: --	

Figure 11. Menu window

field of view :			GMT :					
width :								
height :								
centered in : (,)								
limits of ra : (,)								
limits of dec : (,)								
limits of mag :								
red : (,)								
blue : (,)								
ra	dec	s	q	ffr	mag	rad	ang	
(hr min sec)	(deg min sec)				red blue	(arcsec)	(deg)	
---	---	-	-	---	-	-	---	
---	---	-	-	---	-	-	---	
total objects found : ---			total objects listed: ---					

Figure 12. Window of the list of stars from USC in normal units

List from star catalog (pixels)		
ra	dec	mag
---	---	---
---	---	---
....		
total objects found : ---		total objects listed: ---

Figure 13. Window of the list of star from USC in pixel units

List from ccd camera (pixels)		
ra	dec	mag
---	---	---
---	---	---
	...	
total objects found : ---		total objects listed: ---

Figure 14. Window of the list of stars from CCD in pixel units

duration (sec) : ---	screen cat (pixels)			
	width : ---			
	height : ---			
	screen ccd (pixels)			
	width : ---			
	height : ---			
	ccd origin		corrections	
	generated	calculated	(pixels)	(arcsec)
right ascension :	----	----	----	----
declination :	----	----	----	----
rad perturbation : ---				
mag perturbation : ---				
possible solutions : [---, ---]				
solutions found : ---				

Figure 15. Window of the corrections on RA and DEC

5.3. The Source Program Code

The source program code has been made using C++ programming language.

At this stage it is necessary to take into account some considerations:

1. The USC was created for the Power PC. So, at first, it was necessary to create a subroutine `bigtolittle()` to read USC data using PC correctly.
2. The data from FITS was simulated from USC. For this purpose the subroutine `getccddata()` was created. It simulates the FITS data from USC data, just simulating the little noise (see note of section 6.2) due to the CCD camera.

The principal subroutine, `pattern-recognition()`, is in accord to "numerical solution": USC data would be R , and FITS data would be T. Therefore the numerical solution (x_i, y_i) would be the real solution (`raorigenccd, decorigenccd`). (See figures 5 and 6).

The C++ code source program is in appendix B.

CHAPTER VI.

THE PATTERN-RECOGNITION ALGORITHM

6.1 Data from USC

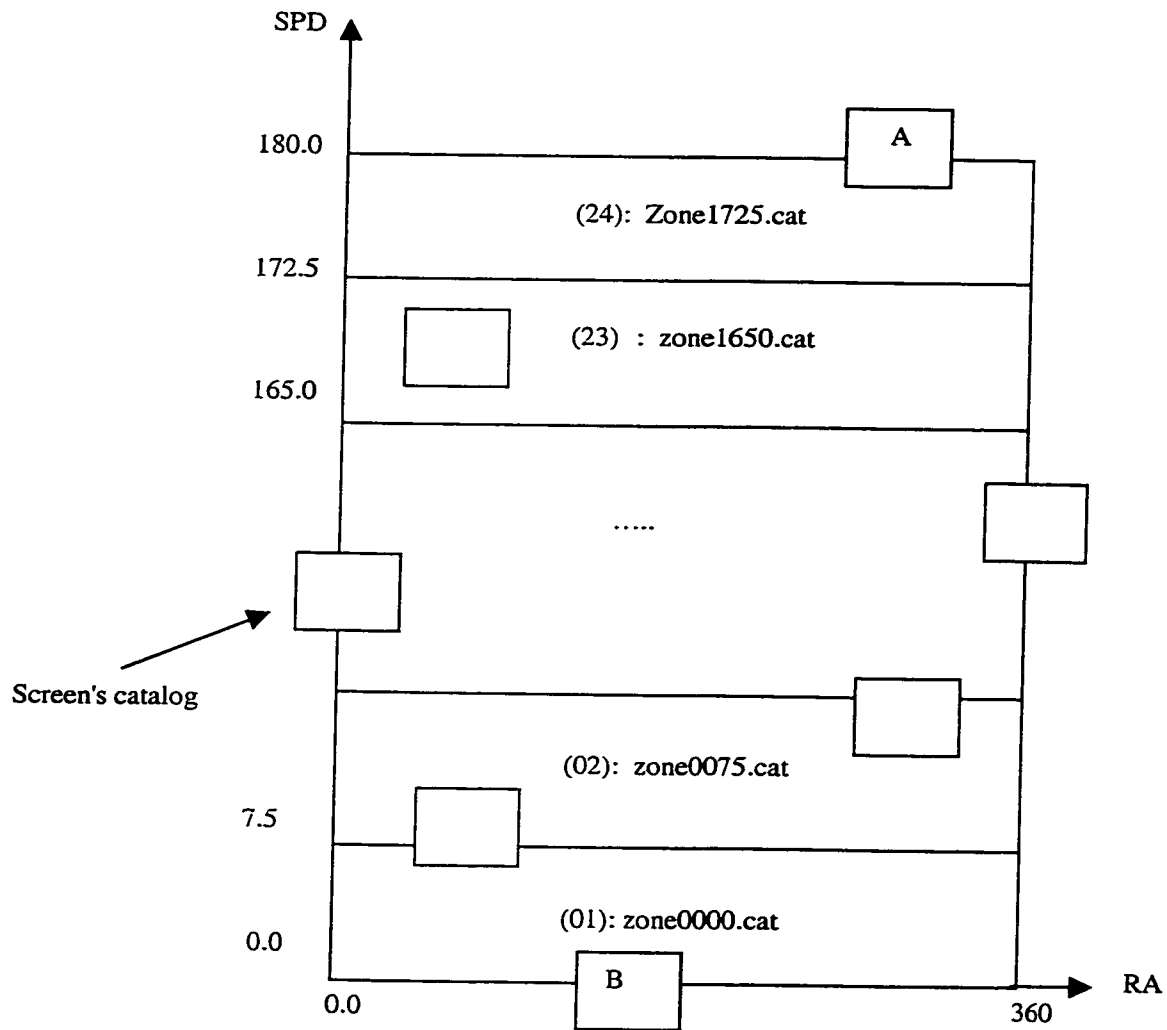


Figure 16. Graphical representation of zonexxx.cat files and possible situations of screen's catalog

The reading of USC data is accelerated using the auxiliary files zonexxxx.acc; without this the reading would take more time.

For each (racentercat, decentercat) given there are several possibilities (look at figure 16):

Heightcat does not cross any SPD (South Polar Distance) multiple of 7.5 degrees.
In this case open the file zonexxxx.cat which the (racentercat, decentercat) belong.

Heightcat crosses the SPD multiple of 7.5 degrees.

In this case open two files zonexxxx.cat:

- zonexxxx.cat which the (racentercat, decentercat) belong
- Zonexxx.cat which precede or antecede it, depend what case is.

Widthcat does not cross the limits of RA ("0" or "360" degrees).

In this case read the zonexxxx.acc corresponding to each zonexxxx.cat opened.

Widthcat crosses the limits of RA ("0" or "360" degrees)

In this case it is necessary to open two zonexxxx.acc files for each corresponding zonexxxx.cat opened.

These possibilities were solved introducing three switches:

ffcat: { 1, 2, 3, 24 } ; it gives the file number where the (racentercat, decentercat) is located

sscat : { -1, 0, +1 } ;

- if (-1) then read file number "ffcat" and "ffcat-1"
- if (0) then just read file number "ffcat"
- if (+1) then read file number "ffcat" and "ffcat+1"

rrcat: { 1, 0 };

- (1) if the widthcat crosses the limits of ra
- (0) if the widthcat does not cross the limits of ra.

This work is made by the following commands:

```

inputdata(pwg, widthccd, heightccd, racentercat, deccentercat, magcatr, magcatrf, magcatbi,
          magcatbf, sizeracat, sizedeccat);
widthheightcat(sizeracat, sizedeccat, widthcat, heightcat, pwg);
limites(racentercat, deccentercat, sizeracat, sizedeccat, racati, racatf, deccati, deccatf, spdcati,
         spdcentercat, spdcatf);
searchfiles(racati, racentercat, racatf, spdcati, spdcentercat, spdcatf,
            racati1, racatf1, racati2, racatf2,
            spdcati1, spdcatf1, spdcati2, spdcatf2,
            ffcats, sscats, rrcats);
createchunk(ffcats, sscats, rrcats, racati, racatf, spdcati, spdcatf,
            magcatr, magcatrf, magcatbi, magcatbf,
            racati1, racatf1, racati2, racatf2,
            spdcati1, spdcatf1, spdcati2, spdcatf2);
chunkvector("chunk", racat, deccat, scat, qcat, ffcats, bbbscat, rrrcat, ncat);
centroradec00(racat, deccat, rrrcat, racentercat, deccentercat, racati1, deccati1, rrrcat1,
              racatk1, deccatk1, ncat, rrcats, sizeracat);
radang(racatk1, deccatk1, racati1, deccati1, radcat, angcat, ncat);
trasladar(racati1, deccati1, rrrcat1, ncat, racati2, deccati2, rrrcat2, ncatp, racati, deccati);
chunkpixel(racati2, deccati2, rrrcat2, racatp, deccatp, rrrcatp, pwg, ncatp);

```

Finally one obtains:

- USC data in normal units from `chunkvector()`, and
- USC data in pixel units from `chunkpixel()`.

Note:

The PRA does not work for case "A" and "B" (see figure 16). Even when screen's catalog is completely inside of `zonexxxx.cat` but is very close to limits of SPD ("0" and "180").

6.2. Data from FITS

The FITS data is generated from CCD camera. The program to do that could be a subject of another project. There are already many programs to do that.

The FITS data could be considered like a set of data from USC but with noise (perturbations) and deviation; so it is possible to get it beginning from USC but introducing some random noises and deviation (see observations of section 3.1). That is:

1. Translate data USC (it would simulate the deviation because the motors of the telescope).
2. Made a little perturbation (it would simulate the noises introduce by the CCD camera itself).

This work is made by the following commands:

```
getorigccdteo(raorigccdteo, decorigccdteo, (widthcat-widthccd), (heightcat-heightccd));
getccddata(racatp, deccatp, rrrcatp, ncatp, raccdp, decccdp, rrrccdp, nccd, widthccd,
heightccd, raorigccdteo, decorigccdteo, perturbccdrad, perturbccdmag);
```

Finally one obtains:

- FITS data in pixels from `getccddata()`

Note:

The CCD camera is not perfect: It is susceptible to the following sources of noise:

Photon noise - Random fluctuations in the photon signal of the source. The rate at which photons are received is not constant.

Thermal noise - Statistical fluctuations in the generation of thermal signal. The rate at which electrons are produced in the semiconductor substrate due to thermal effects is not constant.

Readout noise - Errors in reading the signal; generally dominated by the on-chip amplifier.

Quantization noise - Errors introduced in the A/D (analog/digital) conversion process.

Sensitivity Variation - Sensitivity variations from photosite to photosite on the CCD detector or across the detector. Modern CCD's are uniform to better than 1% between neighboring photosites and uniform to better than 10% across the entire surface.

6.3. The pattern recognition algorithm

6.3.1 The first solution

The first solution is the first (ra, dec) in the region

ra : [0, (widthcat-widthccd)]

dec : [0, (heightcat-heightccd)]

which meets with

perturbrad condition:

$$| \text{perturbccdrad} + \text{desvorigen} | \leq \text{perturbrad}$$

where *perturbrad* is the maximum perturbation of the coordinates of stars.

and

magnitude condition :

$$| \text{perturbccdmag} | \leq \text{perturbmag}$$

where *perturbmag* is the maximum perturbation of the magnitude of the stars

perturbccdrad and **desvorigen** are define in Figure 17.

The first solution (raorigenccd1, decorigenccd1) is calculated by:

pattern(racatp, deccatp, rrcatp, ncatp, widthcat, heightcat, raccdp, decccdp, rrrccd, nccd, widthccd, heightccd, 0, (widthcat-widthccd), 0, (heightcat-heightccd), perturbrad, perturbmag, **raorigenccd1**, **decorigenccd1**, variance, swini);

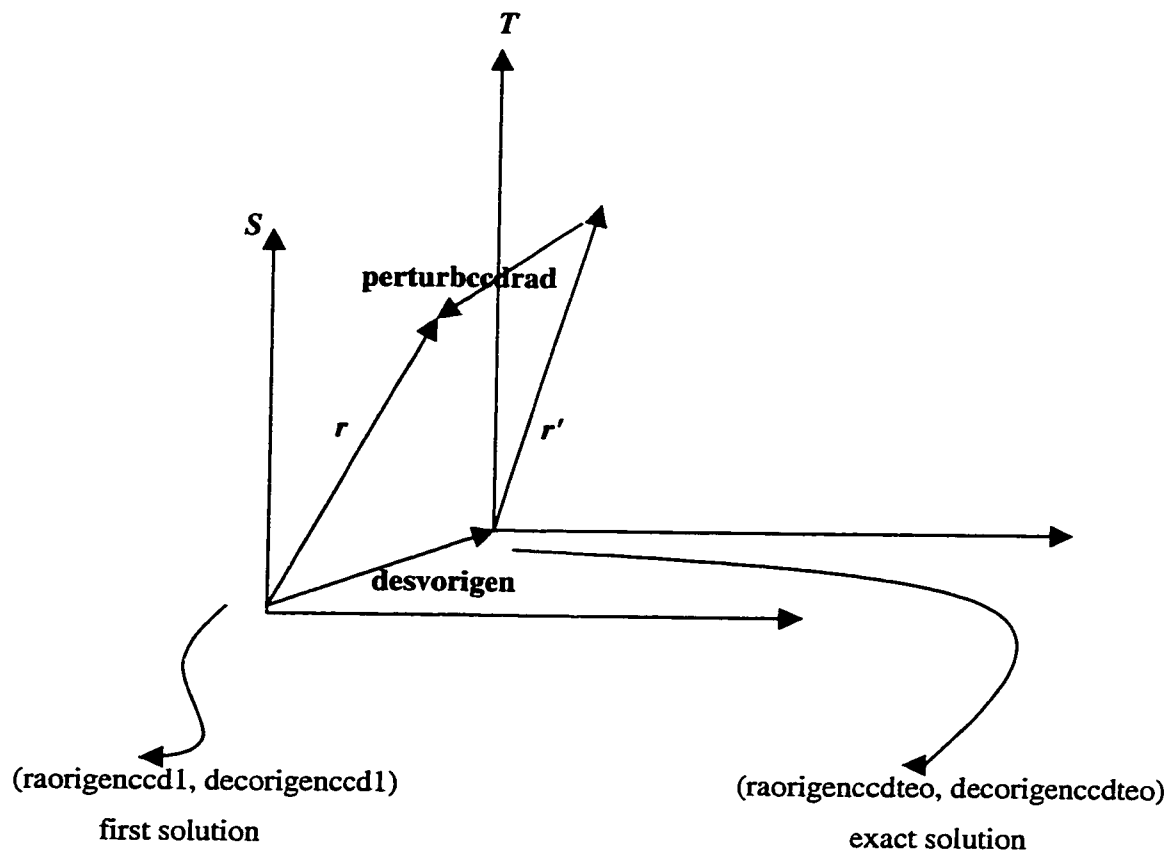


Figure 17. Relation between first solution and exact solution

6.3.2. The possible solutions

What the pattern-recognition() is really doing is to compare each $(x, y) \in S$ to each $(u, v) \in T$, guessing if both have the same coordinate origin (look at figure 18):

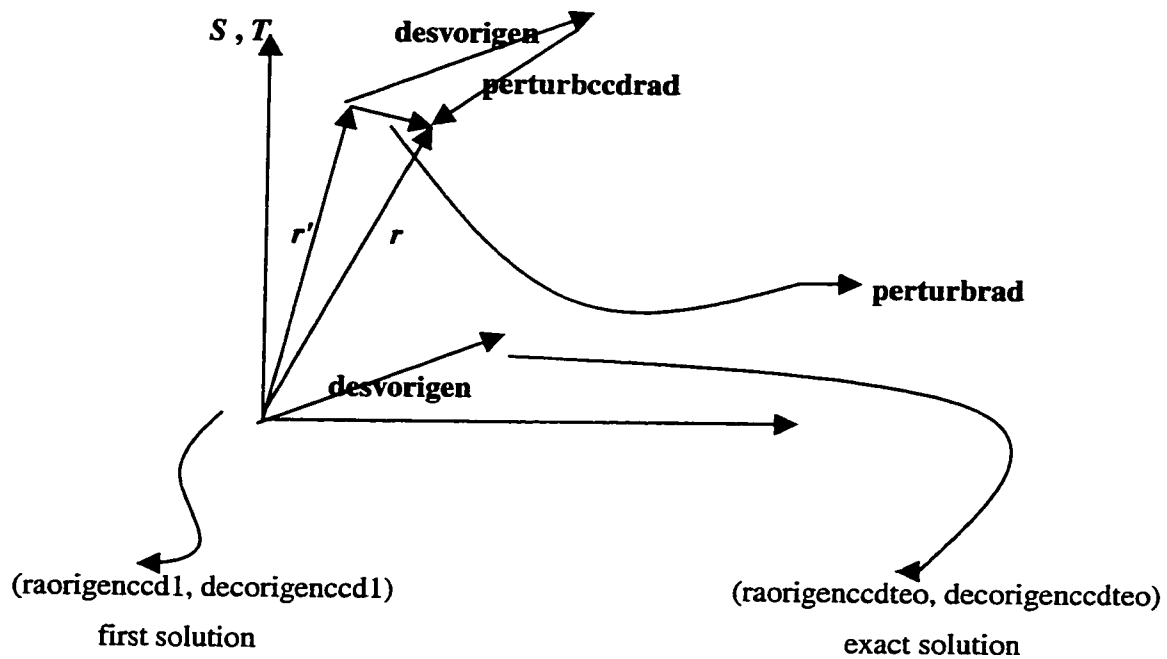


Figure 18. Real comparison made for pattern-recognition()

Therefore

$$r = r' + \text{perturbrad}$$

and

$$\text{perturbrad} = \text{perturbccdrad} + \text{desvorigen}.$$

Where **perturbccdrad** and **desvorigen** are unknown. That is the pattern-recognition() manipulate the *perturbrad* parameter.

Given that the first solution (raorigencdteo, decorigencdteo) should be in a vicinity whose radius is equal to *perturbrad*, then all possible solutions should be within the same vicinity.

If the exact solution is in the vicinity boundary (extreme case) then one can see there are possible solutions out of vicinity. But given these possible solutions should meet with "perturbrad condition", then the radius of the vicinity should be equal to $2 * \text{perturbrad}$ (look at figure 19).

$$|\text{perturbccdrad} + \text{desvorigen}| \leq 2 * \text{perturbrad}$$

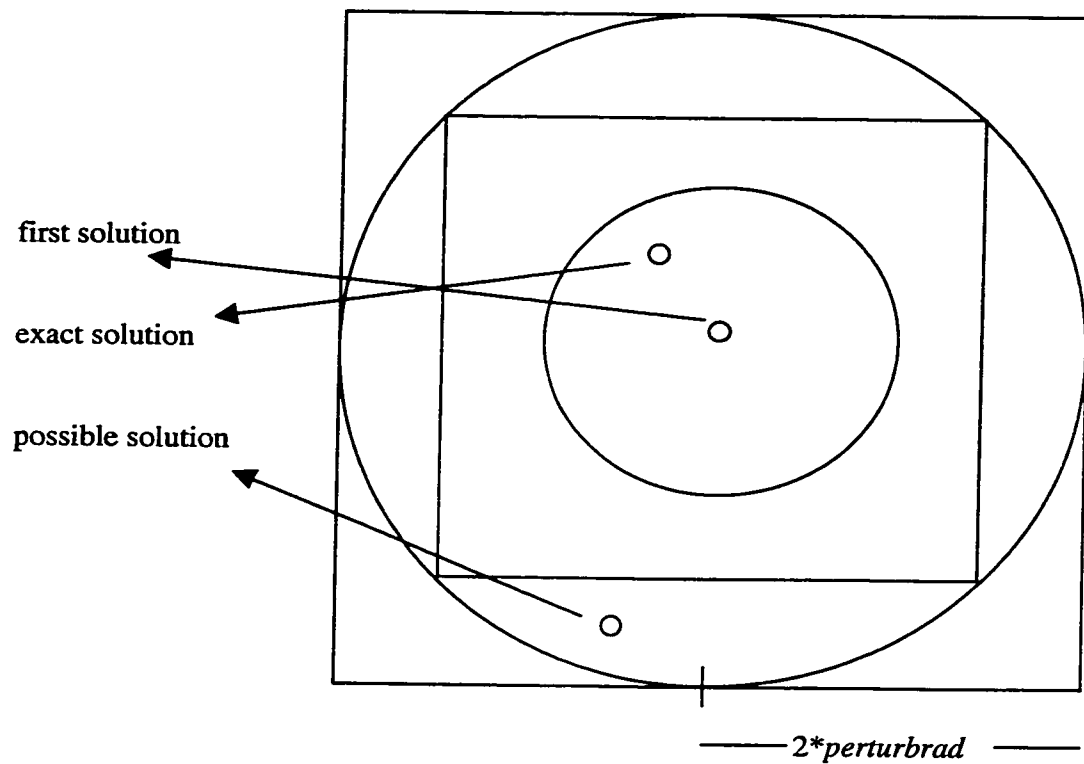


Figure 19. Vicinity of possible solutions

To estimate the number of possible solutions (*numsolposs*) keep in mind

- 1- the *perturbrad* is an integer because it is in pixels
- 2- The III quadrant of the vicinity was already tried because of the *chosen order* (see the section 4.2)

Let $numsolpossmax$ be the number of points in the outer square and $numsolpossmin$ be the number of points in the inner square (look at Figure 19).

Lets $p_2 = perturbrad$

$p_1 = \text{integer part of } (\sqrt{2}) * p_2$

$$\begin{aligned} numsolpossmin &= (2 * p_1 + 1)^2 - (p_1 + 1)^2 + 1 \\ &= 3 * p_1^2 + 2 * p_1 + 1 \end{aligned}$$

$$\begin{aligned} numsolpossmax &= (4 * p_2 + 1)^2 - (2 * p_2 + 1)^2 + 1 \\ &= 12 * p_2^2 + 4 * p_2 + 1 \end{aligned}$$

Therefore

$$numsolpossmin \leq numsolposs \leq numsolpossmax$$

Which depends just on $perturbrad$. Notice that if $perturbrad$ is small then $n \ll N$; and that $numsolposs$ is never zero.

This is calculated by:

$numsolvicinity(perturbrad, numsolpossmin, numsolpossmax);$

6.3.3. The Solutions and the Best One

If all possible solutions are required to meet the “magnitude condition”, it will obtain the right solutions; so always

$$0 \leq numsol(\text{number of right solutions}) \leq numsolposs$$

The right solutions and the best one of them are found by:

```

if (swini==1)
{
    i1 = raorigenccdini-2*perturbrad;
    j1 = decorigenccdini-2*perturbrad;
    i2 = raorigenccdini+2*perturbrad;
    j2 = decorigenccdini+2*perturbrad;

    if(i1<0)
    {
        i1 = 0;
    }
    if(j1<0)
    {
        j1 = 0;
    }
    if(i2>(widthcat-widthccd))
    {
        i2 = widthcat-widthccd;
    }
    if(j2>(heightcat-heightccd))
    {
        j2 = heightcat-heightccd;
    }

    for(j=j1; j<=j2; j++)
        for(i=i1; i<=i2; i++)
            {
                if((i>=raorigenccdini)||(j>=decorigenccdini))
                {
                    pattern(racatp, deccatp, rrrcatp, ncatp, widthcat, heightcat,
                        rccd, decccd, rrrccd, nccd, widthccd, heightccd,
                        i, i, j, j,
                        perturbrad, perturbmag, raorigenccd, decorigenccd,
                        variance, sw);
                }
            }
}

```

```

if(sw==1)
{
    numsol++;
    solutionsra[numsol] = raorigencd;
    solutionsdec[numsol] = decorigencd;
    solutionsvar[numsol] = variance;
}
}
}
ordenar2(solutionsra, solutionsdec, solutionsvar, numsol);
raorigencd = solutionsra[1];
decorigencd = solutionsdec[1];
}

```

where:

- The best solution is (raorigencd, decorigencd), and
- "numsol" is the number of solutions founded

Note:

It is important to notice that the real problem has a third coordinate which does not depend on the translation, it is the "magnitude" (it means it is the same in the FITS and in the USC). This coordinate allows us to get "numsol" because each solution which meets the "perturbed condition" should meet the magnitude condition, at the same time. And given that magnitude coordinate does not depend on translation, it gives us a strong condition, which allows us choose the "right solutions".

Notice that standard deviation of *perturbed* and *perturbmag* is calculated by `compare()` subroutine(Its is inside of `pattern-recognition()`). This result allows us to choose the best solution: the best one is that with minimum standard deviation (`solutionsvar[1]`).

CHAPTER VII.

RESULTS

To get CCD data it was assumed

perturbccdrad = 1pixel
 perturccdmag = 0.5

in subroutine **getccddata()**

To calculate corrections it was assumed

perturbrad = 2pixels
 perturbmag = 0.6

in subroutine **pattern-recognition()**

Pixel per degree = 4000

A.

	Screen cat (pixels)	Screen ccd (pixels)
Width	778	766
Hight	667	510

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	8.0	8.0	-2.0	-1.8	1.0
Declination	16.0	16.0	63.0	56.7	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	8.0	8.0	-2.0	-1.8	1.0
Declination	30.0	30.0	-7.0	-6.3	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	5.0	5.0	1.0	0.9	1.0
Declination	40.0	40.0	39.0	35.1	

B.

	screen cat (pixels)	screen ccd (pixels)
Width	667	556
Hight	667	556

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	81.0	81.0	-25.0	-22.5	4.0
Declination	64.0	64.0	-8.0	-7.2	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	37	37	19	17.1	9.0
Declination	101	101	-45	-40.5	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	64.0	64.0	-8.0	-7.2	< 1.0
Declination	16.0	16.0	40.0	36.0	

C.

	screen cat (pixels)	screen ccd (pixels)
Width	778	667
Hight	889	722

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	43	43	13.0	11.7	9.0
Declination	52	52	32.0	28.8	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	58	58	-2	-1.8	1.0
Declination	5	5	79	71.1	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	71	71	-15.0	-13.5	31.0
Declination	164	164	-80.0	-72.0	

D.

	screen cat (pixels)	screen ccd (pixels)
Width	556	444
Hight	667	556

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	80	80	-24.0	-21.6	3.0
Declination	64	64	-8.0	-7.2	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	14	14	42.0	37.8	<1.0
Declination	53	53	3.0	2.7	

	CCD origin		corrections		duration
	Generated	Calculated	(pixels)	(arcsec)	(sec)
Right ascension	89	89	-33.0	-29.7	3.0
Declination	104	104	-48.0	-43.2	

Table 1. Corrections on RA and DEC to several cases

CHAPTER VIII.

DISCUSSION

1). The list of the stars from USC generated here was compared with the one generated by United State Naval Observatory (USNO), for several entries. The results are in agreement, even though the short version of USNO (USNO-A2.0) was used. The USNO-A2.0 contains information about 500 hundred million stars, while the USC just contains a tenth of these.

The list was also compared with the one generated by the Lowell observatory, which uses the same catalog. The lists are in agreement. But there are some little differences between them. There is a couple more stars; this could be because each was using a different approximate methods and different criteria to try the problem boundary.

In general the list looks very well compared to the two lists mentioned above.

2). The data obtained from the USC and FITS is in pixels. This data was generated thinking to plot the sky in the future. Beside, it could be used to show the calculated "corrections" by the pattern-recognition() graphically.

3). The pattern-recognition() subroutine works not just for a little "originccdteo". And its result depends on "perturbrad" and "perturbmag" chosen, which are the unique variables one can manipulate.

4) The corrections of astronomical coordinates depend on the accuracy of "originccd" found. Because, "originccd" is really what the pattern-recognition() finds.

CHAPTER IX.

CONCLUSIONS AND PERSPECTIVES

The program lists the star of any portion of the sky centered in (racentercat, decentercat) astronomical coordinates .

The program can find the correction on RA and DEC astronomical coordinates with about 1pixel of accuracy considering "perturbrad" ≤ 2 pixels and "perturbmag" ≤ 0.6 .

The program can list in pixel units the data from FITS as much as from USC. This data could be used to get show the corrections to do in graphically.

The corrections will become quite useful to send a fine adjustment command to computer controlling the telescope, and then check the current position where the telescope is pointing.

APPENDIX A

The Right Ascension and Declination

One needs to locate the positions of stellar objects on the sky. Each point of the celestial sphere is identified by using a two numbers. These numbers depend on coordinate system used and could be constant or variable in time. There are several coordinate systems in use but the most used by astronomers is the equatorial coordinate system, the one that was used in this project.

The equatorial coordinate system uses the equatorial plane of the Earth as the fundamental plane, which does not depend on earth motion. Moreover it could be considered an inertial system or fixed coordinate system; in which the calculations are best carried out.

This coordinate system has its origin in earth's center and its principal axis pointing to the vernal equinox.

The two angles are right ascension and declination.

Right ascension (RA):

is measured in the equatorial plane. It is the angle between the vernal equinox and the point where meridian which contains the object crosses the equator. It varies between 0 and 24 hours. It is measured counterclockwise.

Declination (DEC):

is measured in the plane normal to the equator, which contains the object. It is the angle between equator and position vector of an object. It varies between -90 and +90 degrees. It is positive in the northern hemisphere.

South polar distance (SPD): It is measured in the same plane as DEC. It is the angle between the South Pole and the object. That is: $SPD = DEC + 90$. It varies between 0 and 180 degrees.

The USNO-SA2.0 Star Catalog (USC)

The USC is a digital file which contains information about five hundred million stellar objects. This information was obtained by the Precision Measuring Machine (PMM) at the US Naval Observatory. It was built and operated by the US. Naval Observatory Flagstaff Station during the scanning and processing of the Palomar Observatory Sky Survey I (POSS-I) O and E plates, the UK Science Research Council SCR-J Survey plates, and the European Southern Observatory ESO-R survey plates.

The USC uses the RA and SPD coordinate system to locate each stellar objects; based on the system of J2000 at the epoch of the survey blue plate. SPD was used instead DEC because SPD is far easier to manipulate than DEC since it is positive and for compatibility with the ESA Hipparcos and Tycho mission and catalogs.

The information about stellar objects has been stored in USC in such a way as to minimize the storage requirements as well as for easy access:

-The coordinates were converted to integers in the following, manner (An integer number is saved using less memory than a real number):

$$RA = RA * 15 * 3600 * 100$$

$$SPD = (DEC + 90) * 3600 * 100$$

(original RA is in decimal hours and
original DEC is in decimal degrees).

-The entire sky is partitioned into 24 zones of SPD, each of 7.5 degrees.

-In each zone, the catalog is sorted by increasing value of RA.

-Each of the 24 zones of the catalog contains 3 different kinds of files, and the naming convention is: zonexxxx.yyy

$xxx = 10 * SPD$

$yyy = acc$ (ASCII accelerator file)

= cat (binary catalog file)

= lut (binary lookup table for GSC stars).

-Each zonexxxx.cat file is a binary file containing 3 32-bit integer for each entry.

(it means 32-bit for RA, 32-bit for DEC and 32-bit for MAG(magnitude)).

-The byte order is BIG-ENDIAN, which is the default for machines like Silicon Graphics (LITTLE-ENDIAN is the default byte order for machines like DEC).

-Since the catalog files can be quite long, it was convenient to refer to zonexxxx.acc file, and use a combination of the direct and sequential access.

-The zonexxxx.acc file contains the first index for the first object every each 15 minutes of RA and the number of objects in that chunk of RA

-The MAG has been packed according to the following format: $MAG = SQFFFBBBRRR$

S = sign is (-) if there is a correlated GSC entry, (+) if not.

Q = 1 if magnitude(s) might be in error, or is 0 if looked ok.

FFF = field on which this object was detected.

BBB = 10 times the blue magnitude

RRR = 10 times the red magnitude

Flexible Image Transport System (FITS)

When CCD cameras were integrated into a telescope system the digital images became useful. The problem was that these images were not always useful to other astronomers because, each astronomer or astronomical observatory recorded the images according with their own computer facilities. This was a great inconvenience, because the astronomers often wish to compare their images with others in order to verify their own results.

It was necessary to quickly solve the image interchange problem. The solution was the adoption of an "unique image interchange format" called FITS (Flexible Images Transported System). Everyone who needed to interchange images adopted it or was left out.

The FITS needed to be very flexible (It should be so flexible that it can adapt to any changes, and so that it even can be used outside of astronomy). This would be the best feature for its evolution.

So one can say:

FITS is an unique data format developed for interchanging astronomical data between installations conveniently.

APPENDIX B

The C++ Program

```

/*
*****
*****
Main Program
*****
*****
*/

#include <iostream.h>
#include <fstream.h>
#include <time.h>
#include <dos.h>
#include <math.h>
#include <conio.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <iomanip.h>
#include <io.h>
#include <time.h>

const int ne = 2000;
const double pi = 3.1416;

const double perturbccdrad = 1;
const double perturbccdmag = 0.5;
const double perturbrad = 2;
const double perturbmag = 0.6;

```

```

struct record{long rasc; long spdsc; long magsc;};

void menu(char &option);
void centroradec00(double ra[ne], double dec[ne], double rrr[ne],
    double rak, double deck,
    double ra1[ne], double dec1[ne], double rrr1[ne],
    double &rak1, double &deck1,
    int n, int rr, double sizera);
void searchfiles(double &rai, double rak, double &raf,
    double spd, double spdi, double spdf,
    double &spdi1, double &spdf1, double &spdi2, double &spdf2,
    double &rai1, double &raf1, double &rai2, double &raf2,
    int &f, int &s, int &r);
void limites(double ra, double dec, double sizera, double sizedec,
    double &rai, double &raf, double &deci, double &decf,
    double &spdi, double &spd, double &spdf);
void print_limites(double ra, double dec, double sizera, double sizedec,
    double rai, double raf, double deci, double decf,
    double magri, double magrf, double magbi, double magbf);
void readzone(int ff, double rai, double raf, double spdi, double spdf,
    double magri, double magrf,
    double magbi, double magbf);
void createchunk(int ff, int ss, int rr,
    double rai, double raf, double spdi, double spdf,
    double magri, double magrf,
    double magbi, double magbf,
    double rai1, double raf1, double rai2, double raf2,
    double spdi1, double spdf1, double spdi2, double spdf2);
long bigtolit(long n1, int b, int v);
void namefiles(int ff, char namezone1[20], char namezone2[20]);
void nrfcat(char filename[20], long nrecordacc, long &nrecordcat);
void nrfacc( double raip, long &nrecordacc);

```

```

void print_ra (double rasc);
double convert_ra(long rasc);
void print_dec (double dec);
double convert_dec(long spdsc);
void convert_mag(long magsc, char &sig, int &q, int &fff, double &bbb, double &rrr);
void print_mag(char s, int q, int fff, double bbb, double rrr);
void chunkvector(char filename[20], double ra[ne], double dec[ne],
    char s[ne], int q[ne], int fff[ne], double bbb[ne], double rrr[ne], int &n);
void inputdata(double &pxg, double &widthccd, double &heightccd, double &raks, double
&decks, double &magri, double &magrf,
    double &magbi, double &magbf, double &sizera,
    double &sizedec);
void escrivector(double ra[ne], double dec[ne],
    char s[ne], int q[ne], int fff[ne], double bbb[ne],
    double rrr[ne], double rad[ne], double ang[ne], int ne1);
void escrivector1(double ux[ne], double uy[ne], double uz[ne], int ne1);
void trasladar(double ux[ne], double uy[ne], double uz[ne], int nu,
    double vx[ne], double vy[ne], double vz[ne], int &nv,
    double despx, double despy);
void chunkpixel(double ux[ne], double uy[ne], double uz[ne],
    double vx[ne], double vy[ne], double vz[ne], double pxg, int n);
void ordenar(double ra[ne], double dec[ne],
    char s[ne], int q[ne], int fff[ne], double bbb[ne],
    double rrr[ne], double rad[ne], double ang[ne], int ne1);
void ordenar2(double x[200], double y[200], double z[200], int ne1);
void cabecera(int x, int y);
void cabecera1(char titulo[80], char justi, int fila);
void eraselines(int ini, int fin);
void radang(double x0, double y0, double x1[ne], double y1[ne],
    double rad[ne], double ang[ne], int ne1);
void fillvector(double x[ne], double y[ne], double z[ne], int n,
    double x1[ne], double y1[ne], double z1[ne], int n1);
void inputdata(double &ra, double &dec, int &year, int &month, double &day);

```

```

double julianday(int year, int month, double day);
void propermotion(double jd, double &ra, double &dec);
void aberration(double jd, double &ra, double &dec);
void precession(double jd, double &ra, double &dec);
void nutation(double jd, double &ra, double &dec);
void print_dec (double dec);
void print_ra(double rasc);
void utnow( int &year, int &month, double &day);
void ra360dec90(double &ra, double &dec);
void corrections(double &ra, double &dec);
void print_gmt( );

double roundpos(double a);
double difvect(double ux, double uy, double vx, double vy);
void getorigenccdeo(double &dx1, double &dy1, double limx, double limy);
void getccddata(double ux[ne], double uy[ne], double uz[ne], int nu,
                double vx[ne], double vy[ne], double vz[ne], int &nv,
                double lxccd, double lyccd, double dx1, double dy1, double desvrad, double
desvmag);
void perturberccd(double x1[ne], double y1[ne], double z1[ne], int n1, double desvrad, double
desvmag);
void pattern-recognition(double rx[ne], double ry[ne], double rz[ne], int r, double lxr, double
lyr,
                        double tx[ne], double ty[ne], double tz[ne], int t, double lxt, double lyt,
                        double dxi, double dxl, double dyi, double dyf,
                        double desvrad, double desvmag, double &sdx, double &sdyl, double &variance, int
&sw );
void printresult(double dx, double dy, double dx1, double dy1, double dx11, double dy11,
                double lxcat, double lycat, double lxccd, double lyccd, double pxg,
                double perturbccdradp, double perturbccdmagp,
                double duration,
                int numsolut, int numsolmin, int numsolmax, char &accept, int sw);
void rangestars(double ux[ne], double uy[ne], double uz[ne], int nu,

```

```

        double vx[ne], double vy[ne], double vz[ne], int &nv,
        double lix, double liy, double lfx, double lfy);
void comparar(double ux[ne], double uy[ne], double uz[ne],
        double vx[ne], double vy[ne], double vz[ne],
        int nu, int nv, double desvrad, double desvmag, double &variance, int &sw);
void getresult(double lxr, double lyr, double lxr1, double lyr1,
        double dx11, double dy11, double &dx, double &dy);
void widthheightcat(double sizera, double sizedec, double &lracat, double &ldeccat, double
pxg);
void numsolvicinity(double p, int &min, int &max);

void main()
{
    char op, accept;

    double pxg, widthcat, heightcat, widthccd, heightccd;

    double racat[ne]; double deccat[ne]; char scat[ne]; int qcat[ne];
        int fffcat[ne]; double bbccat[ne]; double rrrcat[ne];
        double radcat[ne]; double angcat[ne];
    double racat1[ne], deccat1[ne], rrrcat1[ne];
    double racat2[ne], deccat2[ne], rrrcat2[ne];
    double racatp[ne], deccatp[ne], rrrcatp[ne];

    double variance;

    double racentercat, deccentercat, spdcentercat, sizeracat, sizedeccat, racatk1, deccatk1;
    double racati, racatf, deccati, deccatf, spdcati, spdcatf, magcatri, magcatrf, magcatbi,
magcatbf;
    double racati1, racatf1, racati2, racatf2;
    double spdcati1, spdcatf1, spdcati2, spdcatf2;
    int sscat, ffcac, rrcat;
    int ncat, ncatp, nccdp;

```

```
time_t timei, timef;
int sw, swini, num;
double duration;
double raorigenccdteo, decorigenccdteo, raorigenccdini, decorigenccdini, raorigenccd,
decorigenccd, racorrec, deccorrec;
double raccdp[ne], decccdp[ne], rrrccd[ne];

double solutionsra[200], solutionsdec[200], solutionsvar[200];
int numsolpossmin, numsolpossmax, numsol;

int i, i1, i2, j, j1, j2;

a2:
FILE *point2;

cout.setf(ios::fixed);
cout.setf(ios::showpoint);

point2 = fopen("chunk", "w+b");
fclose(point2);

inputdata(pwg, widthccd, heightccd, racentercat, deccentercat, magcatr, magcatf, magcatb,
magcatbf, sizeracat, sizedecat);
widthheightcat(sizeracat, sizedecat, widthcat, heightcat, pxg);
limites(racentercat, deccentercat, sizeracat, sizedecat, racati, racatf, deccati, deccatf, spdcati,
spdcatercat, spdcatf);
searchfiles(racati, racentercat, racatf,
            spdcati, spdcatercat, spdcatf,
            racati1, racatf1, racati2, racatf2,
            spdcati1, spdcatf1, spdcati2, spdcatf2,
            ffc, sscat, rrcat);
```

```

createchunk(ffcat, sscat, rrcat,
            racati, racatf, spdcati, spdcatf, magcatri, magcatrf, magcatbi, magcatbf,
            racati1, racatf1, racati2, racatf2,
            spdcati1, spdcatf1, spdcati2, spdcatf2);
chunkvector("chunk", racat, deccat, scat, qcat, fffcat, bbbcat, rrrcat, ncat);
centroradec00(racat, deccat, rrrcat, racentercat, deccentercat, racat1, deccat1, rrrcat1, racatk1,
deccatk1,
            ncat, rrcat, sizeracat);
radang(racatk1, deccatk1, racat1, deccat1, radcat, angcat, ncat);
trasladar(racat1, deccat1, rrrcat1, ncat, racat2, deccat2, rrrcat2, ncatp, racati, deccati);
chunkpixel(racat2, deccat2, rrrcat2, racatp, deccatp, rrrcatp, pxg, ncatp);
getorigenccdteo(raorigenccdteo, decorigenccdteo, (widthcat-widthccd), (heightcat-
heightccd));
getccddata(racatp, deccatp, rrrcatp, ncatp, raccdp, decccdp, rrrcdp, nccd, widthccd,
heightccd, raorigenccdteo, decorigenccdteo, perturbccdrad, perturbccdmag);

a1:
clrscr();
menu(op);
switch(op)
{
case '1':
    clrscr();
    print_limites(racentercat, deccentercat, sizeracat, sizedeccat, racati, racatf, deccati,
deccatf,
    magcatri, magcatrf, magcatbi, magcatbf);
    ordenar(racat, deccat, scat, qcat, fffcat, bbbcat, rrrcat, radcat, angcat, ncat);
    escrivector(racat, deccat, scat, qcat, fffcat, rrrcat, bbbcat, radcat, angcat, ncat);
    getchar();
    goto a1;
case '2':
    clrscr();
    cabecera1("list from catalog in pixels", 'c', 1);

```



```
    esrivector1(racatp, deccatp, rrrcatp, ncatp);
    getchar();
    goto a1;
case '3':
    clrscr();
    cabecera1("list from ccd in pixels", 'c', 1);
    esrivector1(raccdp, decccdp, rrrcdp, nccd);
    getchar();
    goto a1;
case '4':
    timei = time(NULL);
    numsol = 0;
    pattern-recognitionracatp, deccatp, rrrcatp, ncatp, widthcat, heightcat, raccdp,
decccdp, rrrcdp, nccd,
        widthccd, heightccd, 0, (widthcat-widthccd), 0, (heightcat-heightccd),
        perturbrad, perturbmag, raorigenccдини, decorigenccдини, variance, swini);

if (swini==1)
{
    i1 = raorigenccдини-2*perturbrad;
    j1 = decorigenccдини-2*perturbrad;
    i2 = raorigenccдини+2*perturbrad;
    j2 = decorigenccдини+2*perturbrad;

    if(i1<0)
    {
        i1 = 0;
    }
    if(j1<0)
    {
        j1 = 0;
    }
}
```

```

if(i2>(widthcat-widthccd))
{
    i2 = widthcat-widthccd;
}
if(j2>(heightcat-heightccd))
{
    j2 = heightcat-heightccd;
}

for(j=j1; j<=j2; j++)
for(i=i1; i<=i2; i++)
{
    if((i>=raorigenccdni)||(j>=decorigenccdni))
    {
        pattern-recognition(racatp, deccatp, rrrcatp, ncatp, widthcat, heightcat,
            raccdp, decccdp, rrrccd, nccd, widthccd, heightccd,
            i, i, j, j,
            perturbrad, perturbmag, raorigenccd, decorigenccd,
            variance, sw);

        if(sw==1)
        {
            numsol++;
            solutionsra[numsol] = raorigenccd;
            solutionsdec[numsol] = decorigenccd;
            solutionsvar[numsol] = variance;
        }
    }
}

ordenar2(solutionsra, solutionsdec, solutionsvar, numsol);
raorigenccd = solutionsra[1];
decorigenccd = solutionsdec[1];
}

```

```

timef = time(NULL); duration = difftime(timef, timei);
numsolvicinity(perturbrad, numsolpossmin, numsolpossmax);
getresult(widthcat, heightcat, widthccd, heightccd, raorigenccd, decorigenccd,
racorrec, deccorrec);
    printresult(raorigenccdteo, decorigenccdteo, raorigenccd, decorigenccd, racorrec,
deccorrec,
        widthcat, heightcat, widthccd, heightccd, pxg,
        perturbrad, perturbmag, duration, numsol, numsolpossmin, numsolpossmax,
accept, swini);
    getchar();
    goto a1;

case '5':
    goto a2;
case '6':
    clrscr();
    return;
}
}

```

```

void chunkvector(char filename[20], double ra[ne], double dec[ne],
    char s[ne], int q[ne], int fff[ne], double bbb[ne],
    double rrr[ne], int &n)
{
    int f, nn, q1, fff1 ;
    double bbb1, rrr1, mag1, ra1, dec1;
    long rale, decl, sqfbrle;
    char s1;

    FILE *point;
    struct record raspdmag;

    point = fopen (filename , "rb");

```

```

n = 0;
while ((f=fread(&raspdmag, sizeof(struct record), 1, point))== 1)
{
    n++;

    rale = bigtolit(raspdmag.rasc, 32, 8);
    decl = bigtolit(raspdmag.spdsc, 32, 8);
    sqfbrle = bigtolit(raspdmag.magsc, 32, 8);

    ral = convert_ra(rale);
    decl = convert_dec(decl);

    corrections(ral, decl);

    convert_mag(sqfbrle, s1, q1, fff1, bbb1, rrr1);
    ra[n] = ral;
    dec[n] = decl;
    s[n] = s1;
    q[n] = q1;
    fff[n] = fff1;
    bbb[n] = bbb1;
    rrr[n] = rrr1;
}
fclose(point);
}

double convert_ra(long rasc)
{
    return(rasc/3600/100.0000);
}

void print_ra(double rasc)

```

```
{
    char sgra;
    int rah, ram;
    double rah1, ram1, ras;

    if(rasc<0)
    {
        sgra = '-';
    }
    else
    {
        sgra = '+';
    }

    rah1 = fabs(rasc/15.00);

    rah = rah1;

    ram1 = (rah1 - rah)*60.00;
    ram = ram1;

    ras = (ram1 - ram)*60.00;
    printf("%c %02d %02d %05.2f", sgra, rah, ram, ras );
}

double convert_dec(long spdsc)
{
    return( spdsc/3600/100.00 - 90.00);
}

void print_dec (double dec)
{
    char sgdec;
```

```
int decg, decm;
double decs, dec1, decg1, decm1;

if(dec<0)
{
    sgdec = '-';
}
else
{
    sgdec = '+';
}

dec1 = fabs(dec);

decg = dec1;

decm1 = (dec1 - decg)*60.00;

decm = decm1;
decs = (decm1 - decm)*60.00;

decm = fabs(decm);
decs = fabs(decs);
printf("%c %02d %02d %05.2f", sgdec, decg, decm, decs );
}

void convert_mag(long magsc, char &sig, int &q, int &fff, double &bbb, double &rrr)
{

    unsigned long mag1, mag2, mag3;

    mag1 = magsc/1000.0;
    mag1 = mag1*1000.0;
```

```
    rrr = labs(mag1-magsc)/10.0;

    mag2 = mag1/1000000;
    mag2 = mag2*1000000;
    bbb = labs(mag2-mag1)/1000/10.0;

    mag3 = mag2/1000000000;
    mag3 = mag3*1000000000;
    fff = labs(mag3-mag2)/1000000;

    q = labs(mag3/1000000000);

    if(magsc<0)
    {
        sig = '-';
    }
    else
    {
        sig = '+';
    }
}

void print_mag(char s, int q, int fff, double bbb, double rrr)
{
    printf("%c %3d %5d %7.2f %7.2f", s, q, fff, bbb, rrr);
}

void escrivector(double ra[ne], double dec[ne],
                char s[ne], int q[ne], int fff[ne], double bbb[ne],
                double rrr[ne], double rad[ne], double ang[ne], int ne1)
{
    int n, nn, nnn;
```

```

nn = 14;
nnn = nn;
n=nnn;
cabecera(2, nn-3);
for(int i=1; i<=ne1; i++)
{
    gotoxy(2, n); n++, nn++;
    print_ra(ra[i]);
    printf(" ");
    print_dec(dec[i]);
    printf(" ");
    print_mag(s[i], q[i], fff[i], bbb[i], rrr[i]);
    printf(" ");
    printf("%7.2f %8.2f\n", rad[i], ang[i]);

    if((n==24)&&(nn!=(ne1+nnn)))
    {
        gotoxy(1,25);cout<<"total objects found : "<<ne1;
        gotoxy(28,25);cout<<"total objects listed : "<<(nn-nnn);
        gotoxy(57,25); cout<<"press ENTER to continue";
        getchar();
        eraselines(nnn, 24);
        n=nnn;
    }
}
eraselines(25,25);
gotoxy(1,25);cout<<"total objects found : "<<ne1;
gotoxy(28,25);cout<<"total objects listed : "<<(nn-nnn);
}

void chunkpixel(double ux[ne], double uy[ne], double uz[ne],
                double vx[ne], double vy[ne], double vz[ne], double pxg, int n)
{

```



```
for(int i=1; i<=n; i++)
{
    vx[i] = roundpos(ux[i]*pxg);
    vy[i] = roundpos(uy[i]*pxg);
    vz[i] = uz[i];
}
}

long bigtolit(long n1, int b, int v)
{
    char source1[50], source2[50], source3[50], *endptr;
    int j, k, len1, len2, mi1, mf1, mi2, mf2;

    k = b/v;

    ltoa(n1, source1, 2);
    len1 = strlen(source1);
    len2 = b-len1;

    for(int i = 0; i<= (len2-1); i++)
    {
        source2[i] = '0';
    }

    j = 0;

    for(int i=len2; i<=(b-1); i++)
    {
        source2[i] = source1[j];
        j++;
    }

    for(j = 0; j<=k-1; j++)
```

```
{
    mi1 = v*j;
    mf1 = mi1 + (v - 1);

    mi2 = v*(k-1-j);
    mf2 = mi2 + (v-1);

    for(int i=0; i<=(v-1); i++)
    {
        source3[mi2+i] = source2[mi1+i];
    }
}
return(strtol(source3, &endptr, 2));
}

void nrfcat(char filename[20], long nrecordacc, long &nrecordcat)
{
    char rah1[10], index1[10], nobje1[10];
    char *endptr;
    double rah, lar[96];
    long nobjecat;

    FILE *point;

    point = fopen(filename, "rt");

    fseek(point, 30*(nrecordacc-1), 0);
    fread(&rah1, sizeof(rah1), 1, point);
    fread(&index1, sizeof(index1), 1, point);
    fread(&nobje1, sizeof(nobje1), 1, point);

    rah = strtod(rah1, &endptr);
    nrecordcat = strtol(index1, &endptr, 10);
```

```
nobjecat = strtol(nobje1, &endptr, 10);
fclose(point);
}

void nrfacc( double raip, long &nrecordacc)
{
    int i;
    double lra[97];

    for(i=0; i<=96; i++)
    {
        lra[i] = .25*i;
    }

    for(i = 1; i<=96; i++)
    {
        if(((raip/15.00)>=lra[i-1]) && ((raip/15.00)<=lra[i]))
        {
            if((raip/15.00)==lra[i])
            {
                nrecordacc = i+1 ;
            }
            else
            {
                nrecordacc = i;
            }
            break;
        }
    }
}

void searchfiles(double &rai, double rak, double &raf,
                double spdi, double spdk, double spdf,
```

```
double &rai1, double &raf1, double &rai2, double &raf2,  
double &spdi1, double &spdf1, double &spdi2, double &spdf2,  
int &f, int &s, int &r)  
{  
    int i;  
    double lspd[25];  
  
    for(i=0; i<=24; i++)  
    {  
        lspd[i] = 7.5*i;  
    }  
  
    for (i = 1; i<=24; i++)  
    {  
        if ( (spdk >= lspd[i-1]) && (spdk <= lspd[i]))  
        {  
            f = i;  
            break;  
        }  
    }  
  
    if (spdi < lspd[f-1])  
    {  
        s = -1;  
    }  
    else  
    {  
        if (spdf >= lspd[f])  
        {  
            s = 1;  
        }  
        else  
        {
```

```
        s = 0;
    }
}

if (s == -1)
{
    spdi1 = spdi;
    spdf1 = lspd[f-1];
    spdi2 = lspd[f-1];
    spdf2 = spdf;
}
else
{
    spdi1 = spdi;
    spdf1 = lspd[f];
    spdi2 = lspd[f];
    spdf2 = spdf;
}

r = 0;
if (rai < 0)
{
    r = 1;
    rai1 = 0;
    raf1 = raf;
    rai2 = 360 + rai;
    raf2 = 360;

    rai = rai;
    raf = raf;
}

if (raf > 360.0)
```

```

    {
        r = 1;
        rai1 = 0;
        raf1 = raf - 360;
        rai2 = rai;
        raf2 = 360;

        rai = rai - 360.0;
        raf = raf - 360.0;
    }
}

void centroradec00(double ra[ne], double dec[ne], double rrr[ne],
    double rak, double deck,
    double ral[ne], double decl[ne], double rrr1[ne],
    double &rak1, double &deck1,
    int n, int rr, double sizera)
{ int n1;
  double lim;

  if(rr==1)
  {
    if(rak>sizera)
    {
      rak1 = rak-360;
    }
    else
    {
      rak1 = rak;
    }

    deck1 = deck;

```

```
for(int i=1; i<=n; i++)
{
    if(ra[i] >= sizera)
    {
        ral[i] = ra[i]-360;
    }
    else
    {
        ral[i] = ra[i];
    }

    decl[i] = dec[i];
    rrr1[i] = rrr[i];
}
}
else
{
    rak1 = rak;
    deck1 = deck;
    fillvector(ra, dec, rrr, n, ral, decl, rrr1, n1);
}
}

void escrivector1(double ux[ne], double uy[ne], double uz[ne], int ne1)
{
    int n, nn, nnn;

    nn = 5;
    nnn = nn;
    n=nnn;

    gotoxy(2, nn-2);
    cout << setw(20)<< "ra" << setw(20)<< "dec" << setw(20)<< "mag";
```

```

for(int i=1; i<=ne1; i++)
{
    gotoxy(2, n++); nn++;

    cout << setw(20) << ux[i] <<setw(20)<< uy[i] << setw(20)<< uz[i]<<endl;
    if((n==24)&&(nn!=(ne1+nnn)))
    {
        gotoxy(1,25);cout<<"total objects found : "<<ne1;
        gotoxy(28,25);cout<<"total objects listed : "<<(nn-nnn);
        gotoxy(57,25); cout<<"press ENTER to continue";
        getch();
        eraseslines(nnn, 24);
        n=nnn;
    }
}
eraseslines(25,25);
gotoxy(2,25);cout<<"total objects found : "<<ne1;
gotoxy(28,25);cout<<"total objects listed : "<<(nn-nnn);
}

void trasladar(double ux[ne], double uy[ne], double uz[ne], int nu,
               double vx[ne], double vy[ne], double vz[ne], int &nv,
               double despx, double despy)
{
    for(int i=1; i<=nu; i++)
    {
        vx[i] = ux[i]- despx;
        vy[i] = uy[i]- despy;
        vz[i] = uz[i];
    }
    nv = nu;
}

```



```
double roundpos(double a)
{
    double diference;

    diference = a-floor(a);

    if (diference < 0.5)
    {
        return(floor(a));
    }
    else
    {
        return(ceil(a));
    }
}

void readzone(int ff, double rai, double raf, double spdi, double spdf,
             double magri, double magrf,
             double magbi, double magbf)
{
    double rara, dede;

    long nrecordacc, nrecordcat;

    char namezone1[20], namezone2[20];

    int j;
    double ra1; double dec1; char s1; int q1;
    int fff1; double bbb1; double rrr1;
```

```
long double a;

unsigned long sqfbrle;
long rale, decle;

struct record raspdmag;
FILE *point1, *point2;

namefiles(ff, namezone1, namezone2);

point1 = fopen(namezone1, "rb");
point2 = fopen("chunk", "a+b");

nrfacct(rai, nrecordacc);
nrfcatt(namezone2, nrecordacc, nrecordcat);

fseek(point1, 12*(nrecordcat-1), 0);

do
{
    j = fread(&raspdmag, sizeof(struct record), 1, point1);
    rale = bigtolit(raspdmag.rasc, 32, 8);
    ral = convert_ra(rale);
}
while(ral < rai);

while(ral <= raf)
{
    if(j != 1) break;
    decle = bigtolit(raspdmag.spdsc, 32, 8);
    sqfbrle = bigtolit(raspdmag.magsc, 32, 8);
    decl = convert_dec(decle);
    convert_mag(sqfbrle, s1, q1, fff1, bbb1, rrr1);
}
```

```

    if( ((dec1+90.00)>=spdi)&&((dec1+90)<=spdf)
        &&(rrr1>=magri)&&(rrr1<=magrf)
        &&(bbb1>=magbi)&&(bbb1<=magbf) )
    {
        fwrite(&raspdmag, sizeof(struct record), 1, point2);
    }

    j = fread(&raspdmag, sizeof(struct record), 1, point1);
    rale = bigtolit(raspdmag.rasc, 32, 8);
    ral = convert_ra(rale);
}

fclose(point1);
fclose(point2);
}

void namefiles(int ff, char namezone1[20], char namezone2[20])
{
    switch(ff)
    {
        case 1:
            strcpy(namezone1, "e:\zone0000.cat");
            strcpy(namezone2, "e:\zone0000.acc");
            break;
        case 2:
            strcpy(namezone1, "e:\zone0075.cat");
            strcpy(namezone2, "e:\zone0075.acc");
            break;
        case 3:
            strcpy(namezone1, "e:\zone0150.cat");
            strcpy(namezone2, "e:\zone0150.acc");
            break;
    }
}

```

case 4:

```
strcpy(namezone1, "e:\zone0225.cat");  
strcpy(namezone2, "e:\zone0225.acc");  
break;
```

case 5:

```
strcpy(namezone1, "e:\zone0300.cat");  
strcpy(namezone2, "e:\zone0300.acc");  
break;
```

case 6:

```
strcpy(namezone1, "e:\zone0375.cat");  
strcpy(namezone2, "e:\zone0375.acc");  
break;
```

case 7:

```
strcpy(namezone1, "e:\zone0450.cat");  
strcpy(namezone2, "e:\zone0450.acc");  
break;
```

case 8:

```
strcpy(namezone1, "e:\zone0525.cat");  
strcpy(namezone2, "e:\zone0525.acc");  
break;
```

case 9:

```
strcpy(namezone1, "e:\zone0600.cat");  
strcpy(namezone2, "e:\zone0600.acc");  
break;
```

case 10:

```
strcpy(namezone1, "e:\zone0675.cat");  
strcpy(namezone2, "e:\zone0675.acc");  
break;
```

case 11:

```
strcpy(namezone1, "e:\zone0750.cat");  
strcpy(namezone2, "e:\zone0750.acc");  
break;
```

case 12:

```
strcpy(namezone1, "e:\zone0825.cat");  
strcpy(namezone2, "e:\zone0825.acc");  
break;
```

case 13:

```
strcpy(namezone1, "e:\zone0900.cat");  
strcpy(namezone2, "e:\zone0900.acc");  
break;
```

case 14:

```
strcpy(namezone1, "e:\zone0975.cat");  
strcpy(namezone2, "e:\zone0975.acc");  
break;
```

case 15:

```
strcpy(namezone1, "e:\zone1050.cat");  
strcpy(namezone2, "e:\zone1050.acc");  
break;
```

case 16:

```
strcpy(namezone1, "e:\zone1125.cat");  
strcpy(namezone2, "e:\zone1125.acc");  
break;
```

case 17:

```
strcpy(namezone1, "e:\zone1200.cat");  
strcpy(namezone2, "e:\zone1200.acc");  
break;
```

case 18:

```
strcpy(namezone1, "e:\zone1275.cat");  
strcpy(namezone2, "e:\zone1275.acc");  
break;
```

case 19:

```
strcpy(namezone1, "e:\zone1350.cat");  
strcpy(namezone2, "e:\zone1350.acc");  
break;
```

case 20:

```
strcpy(namezone1, "e:\zone1425.cat");
```

```
        strcpy(namezone2, "e:\\zone1425.acc");
        break;
    case 21:
        strcpy(namezone1, "e:\\zone1500.cat");
        strcpy(namezone2, "e:\\zone1500.acc");
        break;
    case 22:
        strcpy(namezone1, "e:\\zone1575.cat");
        strcpy(namezone2, "e:\\zone1575.acc");
        break;
    case 23:
        strcpy(namezone1, "e:\\zone1650.cat");
        strcpy(namezone2, "e:\\zone1650.acc");
        break;
    case 24:
        strcpy(namezone1, "e:\\zone1725.cat");
        strcpy(namezone2, "e:\\zone1725.acc");
        break;
    }
}

void limites(double ra, double dec, double sizera, double sizedec,
            double &rai, double &raf, double &deci, double &decf,
            double &spdi, double &spd, double &spdf)
{
    rai = ra - (sizera/2);
    raf = ra + (sizera/2);

    deci = dec - (sizedec/2);
    decf = dec + (sizedec/2);

    spdi = deci + 90.0;
    spd = dec + 90.0;
```

```

    spdf = decf + 90.0;
}

void createchunk(int ff, int ss, int rr,
    double rai, double raf, double spdi, double spdf,
    double magri, double magrf,
    double magbi, double magbf,
    double rai1, double raf1, double rai2, double raf2,
    double spdi1, double spdf1, double spdi2, double spdf2)
{
    if(((ff>1)&&(ff<24))||((ff==1)&&(ss!=-1))||((ff==24)&&(ss!=1)))
    {
        if(ss!=0)
        {
            if(ss==-1)
            {
                if(rr==1)
                {
                    readzone(ff-1, rai1, raf1, spdi1, spdf1, magri, magrf, magbi, magbf);
                    readzone(ff, rai1, raf1, spdi2, spdf2, magri, magrf, magbi, magbf);
                    readzone(ff-1,rai2, raf2, spdi1, spdf1, magri, magrf, magbi, magbf);
                    readzone(ff,rai2, raf2, spdi2, spdf2, magri, magrf, magbi, magbf);
                }
            }
            else
            {
                readzone(ff-1, rai, raf, spdi1, spdf1, magri, magrf, magbi, magbf);
                readzone(ff, rai, raf, spdi2, spdf2, magri, magrf, magbi, magbf);
            }
        }
    }
    else
    {
        if(rr==1)
        {

```

```

    readzone(ff, rai1, raf1, spdi1, spdf1, magri, magrf, magbi, magbf);
    readzone(ff+1, rai1, raf1, spdi2, spdf2, magri, magrf, magbi, magbf);
    readzone(ff, rai2, raf2, spdi1, spdf1, magri, magrf, magbi, magbf);
    readzone(ff+1,rai2, raf2, spdi2, spdf2, magri, magrf, magbi, magbf);
}
else
{
    readzone(ff, rai, raf, spdi1, spdf1, magri, magrf, magbi, magbf);
    readzone(ff+1, rai, raf, spdi2, spdf2, magri, magrf, magbi, magbf);
}
}
else
{
    if(rr==1)
    {
        readzone(ff, rai1, raf1, spdi, spdf, magri, magrf, magbi, magbf);
        readzone(ff, rai2, raf2, spdi, spdf, magri, magrf, magbi, magbf);
    }
    else
    {
        readzone(ff, rai, raf, spdi, spdf, magri, magrf, magbi, magbf);
    }
}
}
}

```

```

void ordenar(double ra[ne], double dec[ne],
             char s[ne], int q[ne], int fff[ne], double bbb[ne],
             double rrr[ne], double rad[ne], double ang[ne], int ne1)
{
    double aux;
    char aux1;

```



```
int aux2;

int i, j, k, l;

for(i=1; i<=(ne1-1); i++)
  for(j= (i+1); j<=ne1; j++)
    if (ra[i] > ra[j])
      {
        aux = ra[i];
        ra[i] = ra[j];
        ra[j] = aux;

        aux = dec[i];
        dec[i] = dec[j];
        dec[j] = aux;

        aux1 = s[i];
        s[i] = s[j];
        s[j] = aux1;

        aux2 = q[i];
        q[i] = q[j];
        q[j] = aux2;

        aux = rrr[i];
        rrr[i] = rrr[j];
        rrr[j] = aux;

        aux = bbb[i];
        bbb[i] = bbb[j];
        bbb[j] = aux;

        aux = rad[i];
```

```

        rad[i] = rad[j];
        rad[j] = aux;

        aux = ang[i];
        ang[i] = ang[j];
        ang[j] = aux;
    }
}

void cabecera(int x, int y)
{
    gotoxy(x, y);
    printf("%13s", "ra  ");
    printf(" ");
    printf("%13s", "dec  ");
    printf(" ");
    printf("%s %3s %5s %15s", "s", "q", "fff", "mag  ");
    printf("%7s %8s\n", "rad", "ang");
    printf("%14s", "hr min sec ");
    printf(" ");
    printf("%13s", "deg min sec");
    printf(" ");
    printf("%18s %7s", "red", "blue");
    printf(" ");
    printf("%8s %6s\n", "(arcsec)", "(deg)");
}

void print_limites(double ra, double dec, double sizera, double sizedec,
                 double rai, double raf, double deci, double decf,
                 double magri, double magrf, double magbi, double magbf)
{
    gotoxy(1, 1); printf("field of view :"); gotoxy(51, 1); print_gmt();
    gotoxy(1, 2); printf("    width : "); print_ra(sizera); printf("\n");

```

```

gotoxy(1, 3); printf("    hight : "); print_dec(sizedec); printf("\n");

gotoxy(1, 4); printf("centered in  : (");
print_ra(ra); printf(" , "); print_dec(dec); printf( ")");

gotoxy(1, 5); printf("limits of ra  : (");
print_ra(rai); printf(" , "); print_ra(raf); printf( ")");

gotoxy(1, 6); printf("limits of dec : (");
print_dec(deci); printf(" , "); print_dec(decf); printf( ")");

gotoxy(1, 7); printf("limits of mag :");
gotoxy(1, 8); printf("    red : (%05.2f , %05.2f)", magri, magrf);
gotoxy(1, 9); printf("    blue : (%05.2f , %05.2f)", magbi, magbf);
}

void eraselines(int ini, int fin)
{
for(int i= ini; i<=fin; i++)
{
gotoxy(1, i); clreol();
}
}

void inputdata(double &pxg, double &widthccd, double &heightccd, double &raks, double
&decks, double &magri, double &magrf,
double &magbi, double &magbf, double &sizera, double &sizedec)
{
int rah, ram, decg, decm;
double ras, decs;
double width, height;

clrscr();

```

```

gotoxy(1, 1); cout << "pixel per degree      :";
gotoxy(1, 2); cout << "width of ccd camera (arcsec):";
gotoxy(1, 3); cout << "height of ccd camera (arcsec):";

gotoxy(24, 6); cout << "right ascension (hr min sec):";
gotoxy(24, 7); cout << "declination  (deg min sec) :";
gotoxy(24, 8); cout << "minima red mag      :";
gotoxy(24, 9); cout << "maxima red mag      :";
gotoxy(24, 10); cout << "minima blue mag     :";
gotoxy(24, 11); cout << "maxima blue mag     :";
gotoxy(24, 12); cout << "width of catalog  (arcsec) :";
gotoxy(24, 13); cout << "height of catalog (arcsec) :";

do
{
    gotoxy(32, 1); clrhol();
    gotoxy(32, 1); scanf("%lf", &pxg);
}
while(pxg<3600);

do
{
    gotoxy(32, 2); clrhol();
    gotoxy(32, 2); scanf("%lf", &width);
}
while((width<400)||width>3600);

do
{
    gotoxy(32, 3); clrhol();
    gotoxy(32, 3); scanf("%lf", &height);
}

```

```
while((height<400)||((height>3600)));

do
{
  gotoxy(55, 6); clreol();
  gotoxy(55, 6); scanf("%d %d %lf", &rah, &ram, &ras);
}
while( (rah<0)||((rah>24)
  ||((ram<0)||((ram>59)
  ||(ras<0.0)||((ras>=60.0)
  ||((rah==24)&&((ram!=0.0)||((ras!=0.0))))
  ));

do
{
  gotoxy(55, 7); clreol();
  gotoxy(55, 7); scanf("%d %d %lf", &decg, &decm, &decs);
}
while( (decg<-90)||((decg>90)
  ||((decm<0)||((decm>59)
  ||(decs<0.0)||((decs>=60.0)
  ||((decg==90)&&((decm!=0.0)||((decs!=0.0))))
  ||((decg== -90)&&((decm!=0.0)||((decs!=0.0))))
  ));

do
{
  gotoxy(55, 8); clreol();
  gotoxy(55, 8); scanf("%lf", &magri);
}
while(magri<=-6);
```

```
do
{
    gotoxy(55, 9); clreol();
    gotoxy(55, 9); scanf("%lf", &magrf);
}
while(magrf<= magri);

do
{
    gotoxy(55, 10); clreol();
    gotoxy(55, 10); scanf("%lf", &magbi);
}
while(magbi<=-6);

do
{
    gotoxy(55, 11); clreol();
    gotoxy(55, 11); scanf("%lf", &magbf);
}
while(magbf<=magbi);

do
{
    gotoxy(55, 12); clreol();
    gotoxy(55, 12); scanf("%lf", &sizera);
}
while((sizera<=0)|| (sizera>3600));

do
{
    gotoxy(55, 13); clreol();
    gotoxy(55, 13); scanf("%lf", &sizedec);
}
}
```

```

while((sizedec<=0)||((sizedec>3600)));

raks = (rah + ram/60.0 + ras/3600.0)*15;
decks = decg + decm/60.0 + decs/3600.0;
sizera = sizera/3600.0;
sizedec = sizedec/3600.0;

widthccd = roundpos(width*pxg/3600);
heightccd =roundpos( height*pxg/3600);

getchar();
clrscr();
}

void radang(double x0, double y0, double x1[ne], double y1[ne],
           double rad[ne], double ang[ne], int ne1)
{
double dx, dy;

for(int i=1; i<=ne1; i++)
{

dx = (x1[i]-x0)*3600;
dy = (y1[i]-y0)*3600;

rad[i] = sqrt(pow(dx, 2)+(pow(dy, 2)));

if((dx!=0.0)&&(dy!=0.0))
{
ang[i] = atan(fabs(dy/dx))*180/pi;

if(dx>0.0)
{

```

```
    if(dy>0.0)
    {
        ang[i] = ang[i];
    }
    else
    {
        ang[i] = 360-ang[i];
    }
}
else
{
    if(dy>0.0)
    {
        ang[i] = 180-ang[i];
    }
    else
    {
        ang[i] = 180+ang[i];
    }
}
}
else
{
    if((dx!=0)&&(dy!=0))
    {
        ang[i] = -1;
    }
    else
    {
        if(dx==0.0)
        {
            if(dy>0.0)
            {
```



```
        ang[i] = 90.0;
    }
    else
    {
        ang[i] = 270.0;
    }
}
else
{
    if(dx>0.0)
    {
        ang[i] = 0.0;
    }
    else
    {
        ang[i] = 180.0;
    }
}
}
}
}
```

```
void fillvector(double x[ne], double y[ne], double z[ne], int n,
               double x1[ne], double y1[ne], double z1[ne], int n1)
{
    for(int i = 1; i<=n; i++)
    {
        x1[i] = x[i];
        y1[i] = y[i];
        z1[i] = z[i];
    }
    n1 = n;
}
```

```
}

double julianday(int year, int month, double day)
{

double a, b, c, d, e;

if( month>2)
{
}
else
{
year = year-1;
month = month+12;
}

modf(year/100.0, &a);
modf(a/4.0, &c);

b = 2 - a + c;
modf((365.25*(year+4716)), &d);
modf(30.6*(month+1), &e);

return(d + e + day + b - 1524.5);
}

void propermotion(double jd, double &ra, double &dec)
{
double t1, dra, ddec;

t1 = (jd - 2451545.0)/365.25;

dra = 0.0*15;
```

```
ddec = 0.0;

dra = dra*t1;
ddec = ddec*t1;

dra = dra/3600.0;
ddec = dra/3600.0;

ra = ra + dra;
dec = dec + ddec;
}

void aberration(double jd, double &ra, double &dec)
{
double L2, L3, L4, L5, L6, L7, L8;
double L, D, M, F;
double a[40], x1[40], x2[40], y1[40], y2[40], z1[40], z2[40];
double dra, ddec;
double T, f, c;
double x, y, z;

c = 17314463350;
f = 3.1416/180.0;

T = (jd - 2451545.0)/36525;

L2 = 3.1761467 + 1021.3285547*T;
L3 = 1.7534703 + 628.3075849*T;
L4 = 6.2034809 + 334.0612431*T;
L5 = 0.5995465 + 52.9690965*T;
L6 = 0.8740168 + 21.3299095*T;
L7 = 5.4812939 + 7.4781599*T;
L8 = 5.3118863 + 3.8133036*T;
```

$$L = 3.8103444 + 8399.6847337*T;$$

$$D = 5.1984667 + 7771.3771486*T;$$

$$M = 2.3555559 + 8328.6914289*T;$$

$$F = 1.6279052 + 8433.4661601*T;$$

$$a[1] = L3;$$

$$a[2] = 2*L3;$$

$$a[3] = L5;$$

$$a[4] = L;$$

$$a[5] = 3*L3;$$

$$a[6] = L6;$$

$$a[7] = F;$$

$$a[8] = L + M;$$

$$a[9] = 2*L5;$$

$$a[10] = 2*L3 - L5;$$

$$a[11] = 3*L3 - 8*L4 + 3*L5;$$

$$a[12] = 5*L3 - 8*L4 + 3*L5;$$

$$a[13] = 2*L2 - L3;$$

$$a[14] = L2;$$

$$a[15] = L7;$$

$$a[16] = L3 - 2*L5;$$

$$a[17] = L8;$$

$$a[18] = L3 + L5;$$

$$a[19] = 2*L2 - 2*L3;$$

$$a[20] = L3 - L5;$$

$$a[21] = 4*L3;$$

$$a[22] = 3*L3 - 2*L5;$$

$$a[23] = L2 - 2*L3;$$

$$a[24] = 2*L2 - 3*L3;$$

$$a[25] = 2*L6;$$

$$a[26] = 2*L2 - 4*L3;$$

$$a[27] = 3*L3 - 2*L4;$$

$$a[28] = L + 2*D - M;$$

$$a[29] = 8*L2 - 12*L3;$$

$$a[30] = 8*L2 - 14*L3;$$

$$a[31] = 2*L4;$$

$$a[32] = 3*L2 - 4*L3;$$

$$a[33] = 2*L3 - 2*L5;$$

$$a[34] = 3*L2 - 3*L3;$$

$$a[35] = 2*L3 - 2*L4;$$

$$a[36] = L - 2*D;$$

$$x1[1] = -1719914 - 2*T; \quad x2[1] = -25;$$

$$x1[2] = 6434 + 141*T; \quad x2[2] = 28007 - 107*T;$$

$$x1[3] = 715; \quad x2[3] = 0;$$

$$x1[4] = 715; \quad x2[4] = 0;$$

$$x1[5] = 486 - 5*T; \quad x2[5] = -236 - 4*T;$$

$$x1[6] = 159; \quad x2[6] = 0;$$

$$x1[7] = 0; \quad x2[7] = 0;$$

$$x1[8] = 39; \quad x2[8] = 0;$$

$$x1[9] = 33; \quad x2[9] = -10;$$

$$x1[10] = 31; \quad x2[10] = 1;$$

$$x1[11] = 8; \quad x2[11] = -28;$$

$$x1[12] = 8; \quad x2[12] = -28;$$

$$x1[13] = 21; \quad x2[13] = 0;$$

$$x1[14] = -19; \quad x2[14] = 0;$$

$$x1[15] = 17; \quad x2[15] = 0;$$

$$x1[16] = 16; \quad x2[16] = 0;$$

$$x1[17] = 16; \quad x2[17] = 0;$$

$$x1[18] = 11; \quad x2[18] = -1;$$

$$x1[19] = 0; \quad x2[19] = -11;$$

$$x1[20] = -11; \quad x2[20] = -2;$$

$$x1[21] = -7; \quad x2[21] = -8;$$

$$x1[22] = -10; \quad x2[22] = 0;$$

$$x1[23] = -9; \quad x2[23] = 0;$$

$$x1[24] = -9; \quad x2[24] = 0;$$

x1[25] = 0; x2[25] = -9;
x1[26] = 0; x2[26] = -9;
x1[27] = 8; x2[27] = 0;
x1[28] = 8; x2[28] = 0;
x1[29] = -4; x2[29] = -7;
x1[30] = -4; x2[30] = -7;
x1[31] = -6; x2[31] = -5;
x1[32] = -1; x2[32] = -1;
x1[33] = 4; x2[33] = -6;
x1[34] = 0; x2[34] = -7;
x1[35] = 5; x2[35] = -5;
x1[36] = 5; x2[36] = 0;

y1[1] = 25 - 13*T; y2[1] = 1578089 + 156*T;
y1[2] = 25697 - 95*T; y2[2] = -5904 - 130*T;
y1[3] = 6; y2[3] = -657;
y1[4] = 0; y2[4] = -656;
y1[5] = -216 - 4*T; y2[5] = -446 + 5*T;
y1[6] = 2; y2[6] = -147;
y1[7] = 0; y2[7] = 26;
y1[8] = 0; y2[8] = -36;
y1[9] = -9; y2[9] = -30;
y1[10] = 1; y2[10] = -28;
y1[11] = 25; y2[11] = 8;
y1[12] = -25; y2[12] = -8;
y1[13] = 0; y2[13] = -19;
y1[14] = 0; y2[14] = 17;
y1[15] = 0; y2[15] = -16;
y1[16] = 0; y2[16] = 15;
y1[17] = 1; y2[17] = -15;
y1[18] = -1; y2[18] = -10;
y1[19] = -10; y2[19] = 0;
y1[20] = -2; y2[20] = 9;

y1[21] = -8;	y2[21] = 6;
y1[22] = 0;	y2[22] = 9;
y1[23] = 0;	y2[23] = -9;
y1[24] = 0;	y2[24] = -8;
y1[25] = -8;	y2[25] = 0;
y1[26] = 8;	y2[26] = 0;
y1[27] = 0;	y2[27] = -8;
y1[28] = 0;	y2[28] = -7;
y1[29] = -6;	y2[29] = 4;
y1[30] = 6;	y2[30] = -4;
y1[31] = -4;	y2[31] = 5;
y1[32] = -2;	y2[32] = -7;
y1[33] = -5;	y2[33] = -4;
y1[34] = -6;	y2[34] = 0;
y1[35] = -4;	y2[35] = -5;
y1[36] = 0;	y2[36] = -5;

z1[1] = 10 + 32*T;	z2[1] = 684185 - 358*T;
z1[2] = 11141 - 48*T;	z2[2] = -2559 - 55*T;
z1[3] = -15;	z2[3] = -282;
z1[4] = 0;	z2[4] = -285;
z1[5] = -94;	z2[5] = -193;
z1[6] = -6;	z2[6] = -61;
z1[7] = 0;	z2[7] = -59;
z1[8] = 0;	z2[8] = -16;
z1[9] = -5;	z2[9] = -13;
z1[10] = 0;	z2[10] = -12;
z1[11] = 11;	z2[11] = 3;
z1[12] = -11;	z2[12] = -3;
z1[13] = 0;	z2[13] = -8;
z1[14] = 0;	z2[14] = 8;
z1[15] = 0;	z2[15] = -7;
z1[16] = 1;	z2[16] = 7;

```

z1[17] = -3;      z2[17] = -6;
z1[18] = -1;      z2[18] = -5;
z1[19] = -4;      z2[19] = 0;
z1[20] = -1;      z2[20] = 4;
z1[21] = -3;      z2[21] = 3;
z1[22] = 0;       z2[22] = 4;
z1[23] = 0;       z2[23] = -4;
z1[24] = 0;       z2[24] = -4;
z1[25] = -3;      z2[25] = 0;
z1[26] = 3;       z2[26] = 0;
z1[27] = 0;       z2[27] = -3;
z1[28] = 0;       z2[28] = -3;
z1[29] = -3;      z2[29] = 2;
z1[30] = 3;       z2[30] = -2;
z1[31] = -2;      z2[31] = 2;
z1[32] = 1;       z2[32] = -4;
z1[33] = -2;      z2[33] = -2;
z1[34] = -3;      z2[34] = 0;
z1[35] = -2;      z2[35] = -2;
z1[36] = 0;       z2[36] = -2;

```

```
x =0; y=0; z =0;
```

```
for(int i=1; i<=36; i++)
```

```
{
    x = x + (x1[i]*sin(a[i]) + x2[i]*cos(a[i]));
    y = y + (y1[i]*sin(a[i]) + y2[i]*cos(a[i]));
    z = z + (z1[i]*sin(a[i]) + z2[i]*cos(a[i]));
}
```

```
dra = (y*cos(ra*f) - x*sin(ra*f))/(c*cos(dec*f));
```

```
ddec = -((x*cos(ra*f) + y*sin(ra*f))*sin(dec*f) - z*cos(dec*f))/c;
```

```
ra = ra + dra/f;
```



```

    dec = dec + ddec/f;
}

void precession(double jd, double &ra, double&ddec)
{
    double t1, t2, m ,o , n;
    double dra, ddec;

    t1 = (jd - 2451545.0)/365.25;
    t2 = t1/100.0;

    m = (3.07496 + 0.00186*t2);
    n = (1.33621 - 0.00057*t2);
    o = (20.0431 - 0.0085*t2);

    dra = m + n*sin(ra*pi/180.0)*tan(dec*pi/180.0);
    dra = 15*dra/3600 ;

    ddec = o*cos(ra*3.1416/180);
    ddec = ddec/3600;

    dra = dra*t1;
    ddec = ddec*t1;

    ra = ra + dra;
    dec = dec + ddec;
}

void nutation(double jd, double &ra, double &ddec)
{
    double d, m, m1, ff, a;
    double T, L, L1;
    double w[60], u[60], v[60];

```

```

double f, dra, ddec, e, de, da;

f = 3.1416/180;

T = (jd - 2451545)/36525;

d = 297.85036 + 445267.111480*T - 0.0019142*pow(T, 2) + pow(T,3)/189474.0;
m = 357.52772 + 35999.050340*T - 0.0001603*pow(T, 2) - pow(T, 3)/300000.0;
ml = 134.96298 + 477198.867398*T + 0.0086972*pow(T,2) + pow(T,3)/56250.0;
ff = 93.27191 + 483202.017538*T - 0.0036825*pow(T,2) + pow(T,3)/327270.0;
a = 125.04452 - 1934.136261*T + 0.0020708*pow(T,2) + pow(T, 3)/450000.0;

L = 280.4665 + 36000.7698*T;
L1 = 218.3165 + 481267.8813*T;

e = 23.439302 - (46.8150*T + 0.00059*pow(T,2) - 0.001813*pow(T,3))/3600.0;

da = -17.20*sin(a*f) - 1.32*sin(2*L*f) - 0.23*sin(2*L1*f) + 0.21*sin(2*a*f);
de = 9.20*cos(a*f) + 0.57*cos(2*L*f) + 0.10*cos(2*L1*f) -0.09*cos(2*a*f);

dra = (cos(e*f) + sin(e*f)*sin(ra*f)*tan(dec*f))*da - cos(ra*f)*tan(dec*f)*de;
ddec = sin(e*f)*cos(ra*f)*da + sin(ra*f)*de;

dra = dra/3600.0;
ddec = ddec/3600.0;

ra = ra + dra;
dec = dec + ddec;
}

void utnow( int &year, int &month, double &day)
{
char *tzstr = "TZ=PST8PDT";//Pacific Standard Time & Daylight Savings

```

```
time_t t;

struct tm *gmt, gmt1;

putenv(tzstr);
tzset();

t = time(NULL);
gmt = gmtime(&t);
gmt1 = *gmt;

year = gmt1.tm_year + 1900;
month = gmt1.tm_mon + 1;
day = gmt1.tm_mday + (gmt1.tm_sec/3600.0 + gmt1.tm_min/60.0 + gmt1.tm_hour)/24.0;
}

void ra360dec90(double &ra, double &dec)
{
if(ra>360.0)
{
ra = ra - 360;
}

if(ra<0)
{
ra = ra + 360;
}

if(dec>90.0)
{
dec = 180.0 - dec;
}
}
```

```
    if(dec<-90)
    {
        dec = -180.0 - dec;
    }
}

void corrections(double &ra, double &dec)
{
    double jd, day;
    int year, month;

    utnow(year, month, day);
    jd = julianday(year, month, day);
    propermotion(jd, ra, dec);
    aberration(jd, ra, dec);
    precession(jd, ra, dec);
    nutation(jd, ra, dec);
    ra360dec90(ra, dec);
}

void print_gmt( )
{
    char *tzstr = "TZ=PST8PDT";//Pacific Standard Time & Daylight Savings
    time_t t;

    struct tm *gmt;

    putenv(tzstr);
    tzset();

    t = time(NULL);
    gmt = gmtime(&t);
    printf("GMT : %s", asctime(gmt));
}
```

```

}

void menu(char &option)
{
    gotoxy(22, 9); cout << "[1]: List of the stars" ;
    gotoxy(22, 11); cout << "[2]: List from catalog in pixels";
    gotoxy(22, 13); cout << "[3]: List from ccd in pixels";
    gotoxy(22, 15); cout << "[4]: Calculate deviation";
    gotoxy(22, 17); cout << "[5]: Begin again";
    gotoxy(22, 19); cout << "[6]: Quit";
    gotoxy(22, 21); cout << "enter your option: ";

    option = '0';
    while( (option != '1') && (option != '2') && (option != '3') && (option != '4')
        && (option != '5') && (option != '6'))
    {
        gotoxy(42, 21); cout << " "; //clreol();
        gotoxy(42, 21); cin >> option;
    }
}

double difvect(double ux, double uy, double vx, double vy)
{
    return(sqrt(pow((ux-vx), 2) + pow((uy-vy), 2)));
}

void getccdata(double ux[ne], double uy[ne], double uz[ne], int nu,
               double vx[ne], double vy[ne], double vz[ne], int &nv,
               double lxccd, double lyccd, double dx1, double dy1, double desvrad, double desvmag)
{
    int u, m, nu1;

    double lix, lfx, liy, lfy, aux;

```

```

double ux1[ne], uy1[ne], uz1[ne];

lix = 0; liy = 0; lfx = lxccd; lfy = lyccd;

trasladar(ux, uy, uz, nu, ux1, uy1, uz1, nul, dx1, dy1);
perturberccd(ux1, uy1, uz1, nul, desvrad, desvmag);
rangestars(ux1, uy1, uz1, nul, vx, vy, vz, nv, lix+perturbccdrad, lfx-perturbccdrad,
liy+perturbccdrad, lfy-perturbccdrad);
}

void perturberccd(double x1[ne], double y1[ne], double z1[ne], int n1, double perturbccdrad,
double perturbccdmag)
{
int i, ix, iy;
double iz;

for(int i = 1; i<=n1; i++)
{
ix = random(perturbccdrad+1);
iy = random(perturbccdrad+1);

if(random(2)==0)
{
x1[i] = x1[i] - ix;
}
else
{
x1[i] = x1[i] + ix;
}

if(random(2)==0)
{
y1[i] = y1[i] - iy;
}
}
}

```

```

    }
else
    {
        y1[i] = y1[i] + iy;
    }

if(random(2)==0)
    {
        z1[i] = z1[i] - perturbccdmag;
    }
else
    {
        z1[i] = z1[i] + perturbccdmag;
    }
}
}

void pattern-recognition(double rx[ne], double ry[ne], double rz[ne], int r, double lxr, double
lyr,
    double tx[ne], double ty[ne], double tz[ne], int t, double lxt, double lyt,
    double dxi, double dxf, double dyi, double dyf,
    double desvrad, double desvmag, double &sdx, double &sdyl, double &variance, int
&sw )
{
    int s, ns, nn;
    double dx, dy;
    double difx, dify;
    double sx[ne], sy[ne], sz[ne];

    nn = 0;

    if((t>3)&&(r>3))
    {

```

```
for (dy = dyi; dy<=dyf; dy++)
{
  clrscr();
  gotoxy(30, 12); cout<< "calculating...";
  for (dx = dxi; dx<= dxf; dx++)
  {
    trasladar(rx, ry, rz, r, sx, sy, sz, s, dx, dy);
    rangestars(sx, sy, sz, s, sx, sy, sz, s, 0, lxt, 0, lyt);

    if(s>=t)
    {
      comparar(sx, sy, sz, tx, ty, tz, s, t, desvrad, desvmag, variance, ns);

      if(ns==t)
      {
        nn = 1;
        sdx = dx;
        sdy = dy;
        dx = dxf;
        dy = dyf;
        variance = variance/ns;
      }
    }
  }
}
if(nn==1)
{
  sw = 1;
}
else
{
  sw = 0;
}
```



```

    }
    else
    {
        sw = 0;
    }
}

void getresult(double lxr, double lyr, double lxr1, double lyr1,
              double dx11, double dy11, double &dx, double &dy)
{
    dx = roundpos((lxr-lxr1)/2 - dx11);
    dy = roundpos((lyr-lyr1)/2 - dy11);
}

void printresult(double dx, double dy, double dx1, double dy1, double dx11, double dy11,
                double lxcac, double lycac, double lxcac1, double lycac1, double pxcg,
                double perturbccdradp, double perturbccdmagp,
                double duration, int numsolut, int numsolmin, int numsolmax, char &accept, int sw)
{
    clrscr();
    gotoxy(1, 1); cout<< "duration (sec) : " << duration;

    gotoxy(46, 1); cout<< "screen cat (pixels) ";
    gotoxy(46, 2); cout<< "      width : "<<lxcac;
    gotoxy(46, 3); cout<< "      hight : "<<lycac;

    gotoxy(46, 4); cout<< "screen ccd (pixels) ";
    gotoxy(46, 5); cout<< "      width : "<<lxcac1;
    gotoxy(46, 6); cout<< "      hight : "<<lycac1;

    gotoxy(27, 9); cout<< "ccd origin"; gotoxy(58, 9); cout<< "corrections";
    gotoxy(20, 10); cout<< "generated"; gotoxy(35, 10); cout<< "calculated";
    gotoxy(53, 10); cout<< "pixels"; gotoxy(67, 10); cout<< "arcsec";

```

```
gotoxy(1, 12); cout<< "righth ascension :";  
gotoxy(1, 13); cout<< "declination  :";  
gotoxy(1, 16); cout<<"rad perturbation  :";  
gotoxy(1, 17); cout<<"mag perturbation  :";  
gotoxy(1, 18); cout<<"possible solutions  :";  
gotoxy(1, 19); cout<<"solutions founded  :";
```

```
switch(sw)
```

```
{
```

```
case 1:
```

```
{
```

```
gotoxy(20, 12);cout<<dx;
```

```
gotoxy(20, 13);cout<<dy;
```

```
gotoxy(35, 12);cout<<dx1;
```

```
gotoxy(35, 13);cout<<dy1;
```

```
gotoxy(53, 12); cout<<dx11;
```

```
gotoxy(53, 13); cout<<dy11;
```

```
gotoxy(67, 12); cout<<(dx11/pxg*3600);
```

```
gotoxy(67, 13); cout<<(dy11/pxg*3600);
```

```
break;
```

```
}
```

```
case 0:
```

```
{
```

```
gotoxy(20, 12);cout<<dx;
```

```
gotoxy(20, 13);cout<<dy;
```

```

        gotoxy(35, 12);cout<<" x ";
        gotoxy(35, 13);cout<<" x ";

        gotoxy(53, 12);cout<<" x ";
        gotoxy(53, 13);cout<<" x ";

        gotoxy(67, 12);cout<<" x ";
        gotoxy(67, 13);cout<<" x ";

        break;
    }
}
gotoxy(22, 16); cout<<perturbccdradp;
gotoxy(22, 17); cout<<perturbccdmagp;
gotoxy(22, 18); cout<<"["<<numsolmin<<","<<numsolmax<<"]";
gotoxy(22, 19); cout<<numsolut;

}

void rangestars(double ux[ne], double uy[ne], double uz[ne], int nu,
               double vx[ne], double vy[ne], double vz[ne], int &nv,
               double lix, double lfx, double liy, double lfy)
{
    nv = 0;
    for(int s=1; s<=nu; s++)
    {
        if((ux[s]>=lix)&&(uy[s]>=liy)&&(ux[s]<=lfx)&&(uy[s]<=lfy)&&(uz[s]>=1)&&(uz[s]<=30))
        {
            nv++;
            vx[nv] = ux[s];
            vy[nv] = uy[s];

```

```

        vz[nv] = uz[s];
    }
}
}

```

```

void comparar(double ux[ne], double uy[ne], double uz[ne],
             double vx[ne], double vy[ne], double vz[ne],
             int nu, int nv, double desvrad, double desvmag, double &variance, int &ns)
{
    double difr, difm;
    double variancevect, variancemag;

    ns = 0;
    variancevect = 0;
    variancemag = 0;
    for(int i = 1; i<= nu; i++)
    {
        for(int j=1; j<=nv; j++)
        {
            difr = difvect(ux[i], uy[i], vx[j], vy[j]);
            difm = fabs(uz[i]-vz[j]);

            if((difr<=desvrad*1.4143)&&(difm<=desvmag))
            {
                ns++;
                j = nv;
                variancevect = variancevect + pow(difr, 2);
                variancemag = variancemag + pow(difm, 2);
            }
        }
    }

    variancevect = pow(variancevect, 0.5);
}

```

```
    variancemag = pow(variancemag, 0.5);
    variance = (variancevect+variancemag)/2;
}
```

```
void escrivector(double rx[ne], double ry[ne], double rz[ne], int ne1)
{
    for(int s=1; s<=ne1; s++)
    {
        cout <<"rx[" << s<< "]=" << " ";
        cout << setprecision(2) << rx[s]; cout << " ";
        cout <<"ry[" << s<< "]=" << " ";
        cout << setprecision(2) << ry[s]; cout<< " ";
        cout <<"rz[" << s<< "]=" << " ";
        cout << setprecision(2) << rz[s] << endl;
    }
}
```

```
void getorigencdteo(double &dx1, double &dy1, double limx, double limy)
{
    dx1 = random(limx+1);
    dy1 = random(limy+1);
}
```

```
void widthheightcat(double sizeracat, double sizedecat, double &lracat, double &ldeccat,
double pxg)
{
    lracat = roundpos(sizeracat*pxg);
    ldeccat = roundpos(sizedecat*pxg);
}
```

```
void cabecera1(char titulo[80], char justi, int fila)
{
```

```
int columna;

columna = (80 - strlen(titulo))/2;

switch (justi)
{
    case 'l':
        break;
    case 'r':
        break;
    case 'c':
        gotoxy(columna, fila); cout<<titulo;
        break;
}
}

void ordenar2(double x[200], double y[200], double z[200], int ne1)
{
    double aux;

    int i, j, k;

    for(i=1; i<=(ne1-1); i++)
        for(j= (i+1); j<=ne1; j++)
            if(z[i]>z[j])
                {
                    aux = x[i];
                    x[i] = x[j];
                    x[j] = aux;

                    aux = y[i];
                    y[i] = y[j];
                    y[j] = aux;
```

```
        aux = z[i];
        z[i] = z[j];
        z[j] = aux;
    }
}

void numsolvicinity(double p, int &min, int &max)
{
    double p1;

    double fraction, integer;
    double number = 100000.567;

    modf((sqrt(2)*p), &p1);

    min = 3*pow(p1, 2) + 2*p1 + 1;
    max = 12*pow(p, 2) + 4*p + 1;
}
```

BIBLIOGRAPHY

Water Savich, 1960,
C++ The object of the programming
Addison- Wesley Publishing Company, inc.

Steven Holzner, 1991,
C Programming
Brady Publishing.

Leendert Ammeraal, 1991,
C for programmers
Wiley Professional Computing.

Jean Meeus, 1991,
Astronomical Algorithms
Library of Congress Cataloging -in-Publication Data.

Aubrey Jones FRAS, 1978,
Mathematical Astronomy with a Pocket calculator
John Wiley and Sons.