PRL-TN-2022-117

PRL Technical Note

Automated Pointing Data Acquisition For Multi-Application Solar Telescope (MAST)

By

Bireddy Ramya, Shibu K. Mathew & A Raja Bayanna

भौतिक अनुसंधान प्रयोगशाला , अहमदाबाद

Physical Research Laboratory, Ahmedabad

प्लेटिनम जयंती
1947 2022
PLATINUM JUBILEE

Disclaimer: *This technical report is based on the work carried out by the authors at PRL. It is assumed that due credit / references are provided by the authors. PRL assures no liability whatsoever for any acts of omissions and any of the issues arising due to the use of results.*

Published by
The Dean's office, PRL.

AUTOMATED POINTING DATA ACQUISITION FOR MULTI-APPLICATION SOLAR TELESCOPE (MAST)

Bireddy Ramya*, Shibu K. Mathew & Ankala Raja Bayanna

Received 27 August 2021; Revised 03 March 2022; Accepted 28 March 2022; Published 28 March 2022.

Abstract
Multi-Application Solar Telescope (MAST) is an off-axis 50 cm solar telescope mounted on an *alt-az* drive. The telescope is installed on an island observatory situated in the middle of Fatehsagar Lake at Udaipur, Rajasthan. After the installation, it is necessary to have a 'pointing model' for the telescope in order to get it precisely point to any object in the sky and also for the solar observations. The pointing model provides correction factors to the initial parameters used in the model or any change in these parameters during the long run. The model is computed by taking the difference between the issued and observed azimuth and elevation coordinates and needs to be done for as many as stars covering the entire sky. Pointing to each star manually and computing the corrections takes longer times, in this Technical Report we discuss a procedure that automated our entire observations for collecting the pointing model data. This allowed us to cover more number of stars and thus to obtain a better pointing model.

Keywords
Multi-Application Solar Telescope, Alt-Az mount, Telescope pointing, Solar Imaging

[1] *Udaipur Solar Observatory, Physical Research Laboratory, Udaipur, Rajasthan*
***Corresponding author**: ramyab@prl.res.in

Contents

Table 1. Important features of the MAST.

Mount	Alt-azimuth
Clear Aperture	50 cm
Type	off-axis Gregorian
Field-of-view	3 arc-min
Pointing accuracy	better than 10 arc-sec per 1 hour
Closed loop tracking	better than 0.1 arc-sec
Secondary mirror	on hexapod, active
Miscellaneous	Thermal control for main mirrors, collapsible open dome enclosure

1. Introduction

Multi-Application Solar Telescope (MAST) is an off-axis 50 cm solar telescope (Venkatakrishnan, et.al. [2017], Mathew, et.al. [2017]). The telescope is designed, fabricated, and installed by the Advanced Mechanical and Optical Systems (AMOS), Belgium (`https://www.amos.be/`). The telescope is situated on an island in Fateh Sagar Lake, Udaipur. The optical schematic of the telescope is shown in Figure 1. The important features of the telescope are listed in Table 1. Tracking of the telescope is done by using an alt-azimuth (alt-az) mount, which is similar to the mounts used in stellar telescopes. Precise, relative encoders are used for pointing the telescope in the sky. A 'pointing model' is computed, by moving the telescope to a large number of stars using computed *alt-az* coordinates.

The computation of the star coordinates is accomplished by obtaining the accurate time and location data (longitude and latitude of the telescope site) from the GPS attached to the Telescope Control System (TCS). In the ideal case, providing the coordinates to the TCS should bring the telescope to the exact location of the star in the sky. But in practice, several issues affect the pointing accuracy. Some of them which are used while deriving the pointing model are listed below;

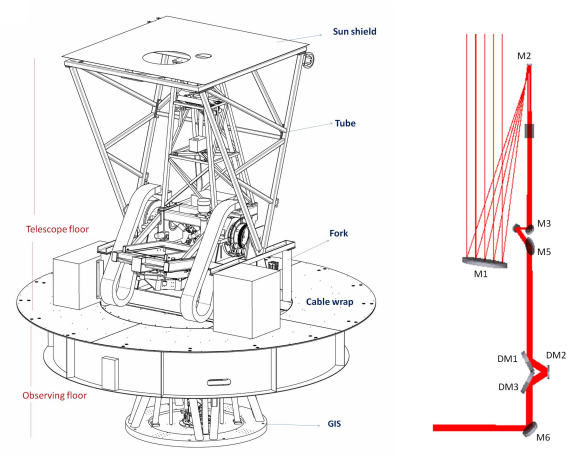

Figure 1. (Left) Schematic of the MAST structure, (right) the optical layout showing all the mirrors. M6 is used to fold the beam to the back-end instruments in the observing floor, which in our case is also used for the pointing tests

.

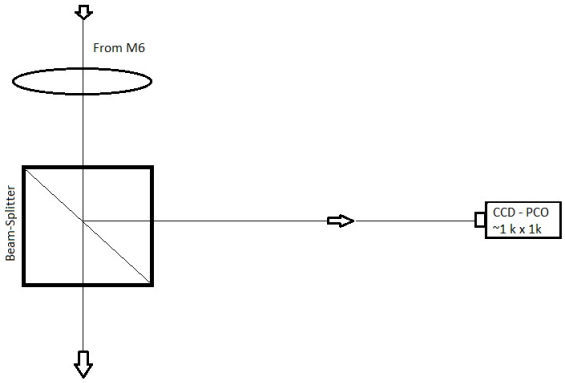

Figure 2. Optical set-up for pointing model data acquisition. The existing optical setup for the G-band solar imaging is used for this purpose. The G-band and neutral density filters are removed from the path to increase the photon flux on the CCD.

- alignment (tilt) of the telescope azimuth axis to earth's east-west and north-south directions
- non-perpendicularity of the elevation and azimuth axes
- indexing error, i.e., zero-point errors in azimuth and elevation readouts
- collimation error, non-perpendicularity of elevation axis to the optical axis
- tube flexure; varying bend in the tube assembly with telescope elevation

A pointing model for the telescope can be obtained by comparing the computed coordinates for a star issued to the telescope and the actual position to which the telescope is moved. TPOINT program provided along with the TCS is used for computing the model parameters (Wallace [1994]). The program fits all the above-mentioned parameters to the observed and computed data, retrieves the appropriate values for these parameters, and later on use those parameters in telescope tracking.

To achieve reliable pointing accuracy, the pointing model should be computed using data from a large number of stars scattered all over the sky in all possible telescope elevation and azimuth values (within the alt-az drive limits). This will allow the fit to properly calculate the parameters, especially the ones which varies with the telescope orientation when pointed to different sky quadrants, e.g., the one for the tube flexure.

The manual star pointing observations involve pointing the telescope to the star using the computed azimuth and elevation values, bringing the star to the center point (reference point) in the FOV (which is normally the optic-axis) for calculating the offsets in the elevation and azimuth and logging those values in the TPOINT data input file. Doing the above steps manually takes a long time and thus limits the observations to a few numbers of stars in a single observing run. To overcome this, we made a computer program in Interactive Data Language (IDL) software making use of the 'socket' programming available in this package. TCS support 'socket' programming through its TCP/IP port where appropriate commands for the telescope control can be issued. In this technical report, a detailed description of the procedure and the program for obtaining the data for the telescope pointing model are given. This procedure can be used for any telescope which is controlled by a similar TCP/IP socket and reduce the manual intervention to a minimum for obtaining the pointing observations. The software is written with a mixture of C++ and IDL, the camera acquisition programs are written in C++ and are called from IDL programs using 'spawn' syntax which is used for executing the program. The images are written in 'fits' format, read and processed online to get the measurements.

2. Steps involved in data acquisition for MAST pointing model

MAST is an off-axis, Gregorian telescope modified to steer the solar beam to a fixed location in the observing room using three mirrors Coudé arrangement. Doing so will produce a field rotation, a three-mirror derotator arrangement is used for compensating the field rotation. The alignment of optical elements of the telescope is done in a way such that the optic axis passes through the center of the derotator cage, and thus can be taken as the center point of the FOV (reference point). A simple optical set-up shown in Figure 2 is used for recording the images of the stars in the FOV. The afocal beam

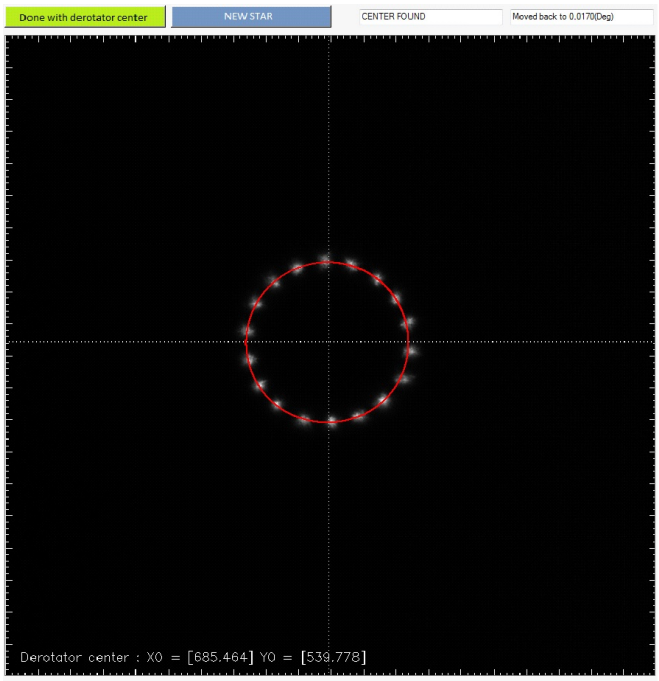

Figure 3. Derotator center finding widget in IDL. The bright points show the rotated location of the star in the FOV for every 10° rotations of the derotator, the red circle is the fit, with center reference points indicated in the bottom.

Figure 4. IDL widget showing the location of the tracked (image in the right side) and corrected position (image on the cross-wire) of the star on the CCD plane. Computed offsets are listed in the lower-left corner of the image.

(reflected by the M6) from the telescope is collected by a lens and is focused on to a CCD (PCO Imaging, SensiCam) having 1376×1040 pixels.

The procedure for collecting the data for the pointing model is listed below;

1. find the derotator center in FOV, and mark it as the reference point
2. point the telescope to a star with the computed co-ordinates
3. by applying collimation offsets in azimuth (CA) and elevation (CE) bring the star to the reference point
4. log the points in the TPOINT data file
5. go to the next star and repeat the steps listed above

In the following sections, we describe each step in detail.

2.1 Derotator center finding

The derotator mirror cage is installed on a rotating mechanism which is driven by a Newport make rotation stage. For finding the center reference point, first the telescope is pointed to a bright star; the coordinates are issued from TCS 'pointing test' widget. The derotator is moved in 10° steps by issuing TCP/IP command;

```
"do rotmove angle=x"
```

and the images of the star are acquired. If the star is not in the derotator center, the images of the star make a circular trace. By fitting a circle to the star track (obtained by adding up all images taken for every 10° rotations, the center point of the star is obtained by taking the centroid of the star image), the center points of the circle can be obtained. A screenshot of the IDL widget for computing the derotator center is shown in Figure 3. The center point (X0, Y0) of the derotator computed from the star track is indicated in the lower left of the figure.

2.2 Star pointing

Instead of manual star pointing, data from standard star catalogs are used for pointing the telescope to different stars. The RA and DEC values from the catalog are read and converted into telescope azimuth and elevation values (procedure RADECTOALTAZ in IDL, refer to Appendix). In order to reduce the travel time taken by the telescope to point to different stars, stars from the catalog are sorted in decreasing order in azimuth angles in 5° bins. The pointing is done by moving the telescope to all the stars in the 5° bins starting from the minimum to the maximum elevation angles within the bin. This procedure is repeated for all the azimuth bins. The maximum and minimum azimuth angles covered by the MAST telescope are 295.5° and 64.5°, respectively and for the elevation, it is 88.5° and 5°. An auto-exposure

Table 2. TPOINT sample input data file, logged by TCS. The first line indicates the telescope name and date, the second line the type of mount, the third for derotator, the fourth, first three numbers are of latitude of the place, and the remaining again the date. From the fifth line onwards, the star coordinates are stored. The first two numbers are the input azimuth and elevation values, the next two numbers are offset applied azimuth and elevation, and the last number is the derotator angle.

```
MAST 2014 05 21
: ALTAZ
: ROTCR
+24 36 15.6 2014 05 21
252.156825 36.562123 252.506044 36.575512 11.093611
247.761465 38.204820 248.113548 38.218794 16.483867
259.042016 37.180458 259.392999 37.192539 359.597069
263.401070 38.976381 263.755925 38.987693 350.713396
270.429921 36.016220 270.779231 36.026921 343.446747
```

time calculation for the CCD camera is also carried out in order to reject very faint stars; the maximum exposure time for 95% of the full well capacity of the CCD, if the required exposure time is above this value the star is rejected from the observations. By scheduling the observations in this way reduces the overall telescope travel time and thus allows us to cover much more stars than the random manual pointing.

2.3 Azimuth and Elevation corrections (ca and ce offsets)

Because of all the issues listed earlier with telescope tracking, applying the computed azimuth and elevation values may not bring the star to the reference point (X0, Y0) as defined by the derotator center. By applying offsets in azimuth (ca) and elevation (ce), the star image can be brought to the reference point. Since the derotator does a field rotation correction in our telescope, applying the elevation and azimuth offsets doesn't move the star's image exactly in the horizontal and vertical directions in the CCD plane. In order to find the required offsets to move the image of the star to the reference point we follow the following procedure;

1. issue the computed azimuth and elevation values for the star to bring it in CCD field-of-view
2. applying small offset steps in azimuth and elevation, record the images.

   For e.g., in our case, we apply -100.0,0.0, and +100.0 arc-secs in ca and ce by issuing the TCP/IP command;

   ```
   "do colloffset ca = az-offset
          ce = ele-offset"
   ```

3. compute the centroid pixel coordinates of the star in the images. From this calculation the slopes $m_{cax} = \Delta ca/\Delta x$, $m_{cay} = \Delta ca/\Delta y$ for ca offsets and $m_{cex} = \Delta ce/\Delta x$, $m_{cey} = \Delta ce/\Delta y$ for ce offsets
4. using the above constants compute the required ca and ce values to bring the star to the reference point as follows;

   $$ca = (m_{cax} \times \delta x) + (m_{cay} \times \delta y)$$
   $$ce = (m_{cex} \times \delta x) + (m_{cey} \times \delta y)$$

   where δx and δy are the difference in star coordinates in pixels between the reference point and the star location on the CCD when the telescope is pointed using the computed azimuth and elevation values.

   Figure 4 is an IDL widget showing one example of the above operation. The location of the star shown in the cross-wire is obtained after correcting the computed azimuth and elevation offsets.

2.4 Logging the star into the TPOINT input file

Once ca and ce are applied to bring the star to the reference point, the TCS allows the logging of the coordinates in a specific file by issuing the TCPI/IP command;

```
"do pointadd"
```

The file format is shown in Table 2. This file is used by the TPOINT to model various parameters. All the above steps are repeated automatically for the next star until all the stars in the sorted star catalogs are exhausted. There is also an option to repeat the whole session. Figure 7 describes the whole procedure in a flowchart.

With logging the pointing data, the automated pointing data acquisition part of the procedure is completed. The next step is to manually run the TPOINT for fitting the observed data.

3. TPOINT for the model parameter calculations

TPOINT software (`http://tpointsw.uk/pointing.htm`) is by TPOINT Consulting, copyrighted to P. T. Wallace, and is provided along with the TCS for MAST. TPOINT do mathematical modelling (the exact details of the mathematical model are not publicly available) of various systematic errors in telescope and the mounting, using the pointing data (as listed in Table 2), and providing the corrected parameters to the TCS. In our pointing runs we fitted the following parameters with TPOINT;

- **IE** - index error in elevation
- **IA** - index error in azimuth

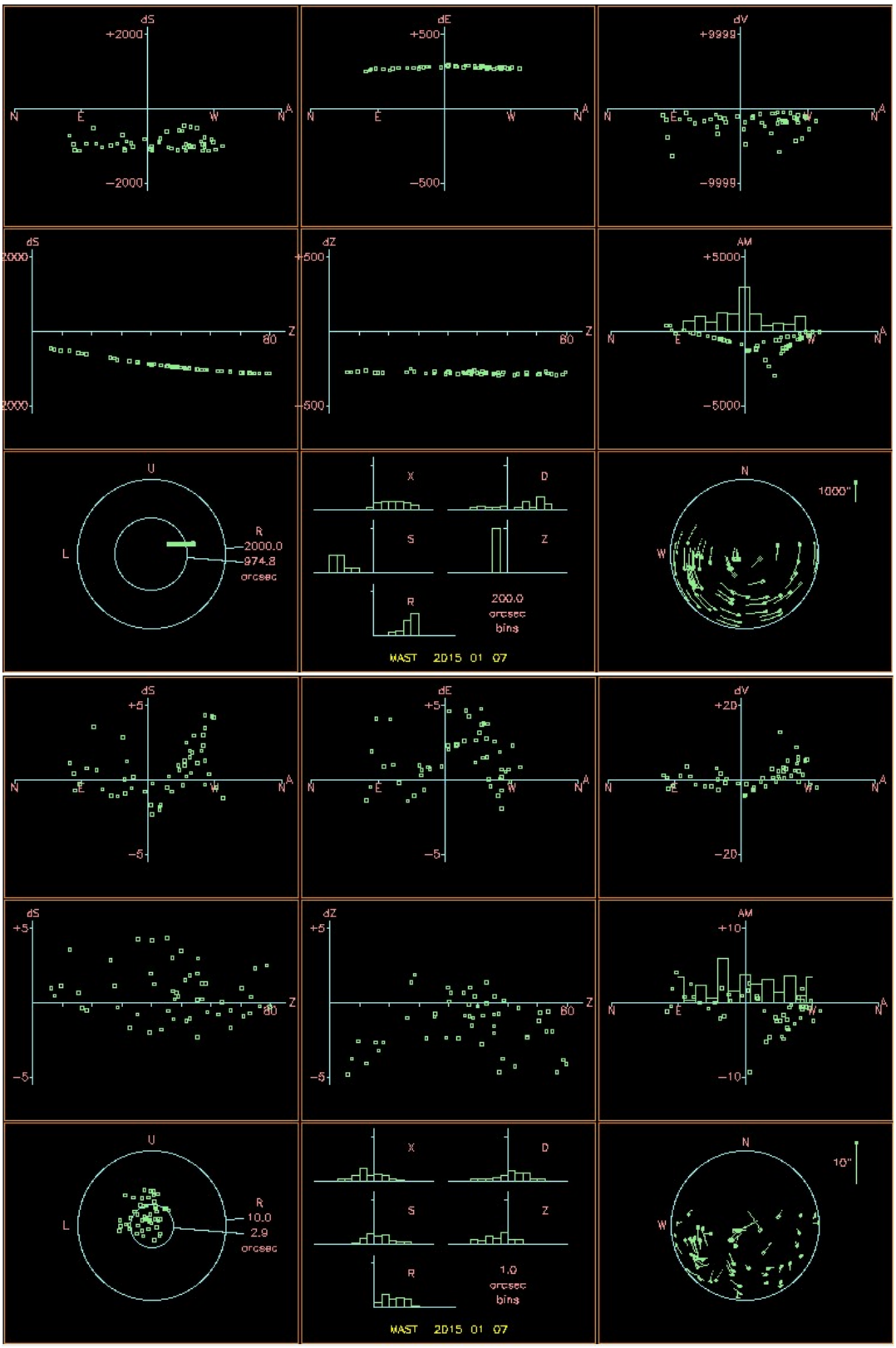

Figure 5. TPOINT windows show the scatter in the pointing error before (top) and after (bottom) fitting the observed data. In the bottom figure, lower left corner, the inner circle is for 2.5 arc-sec error. After fitting, the errors in pointing lies within 10 arc-sec for all observed stars. The various plots in the panel (top two panels, from left-top, clockwise), left-right shift on the sky (ds) with azimuth, elevation errors (dE) with azimuth, elevation non-perpendiculaity (dV) with azimuth, axis mis-alignment (AM) versus azimuth, zenith distance errors (dZ) versus zenith distance, lef-right distance on the sky (dS) versus zenith distance. The error distribution and the map of error vectors are also shown in the last two figures in the panel.

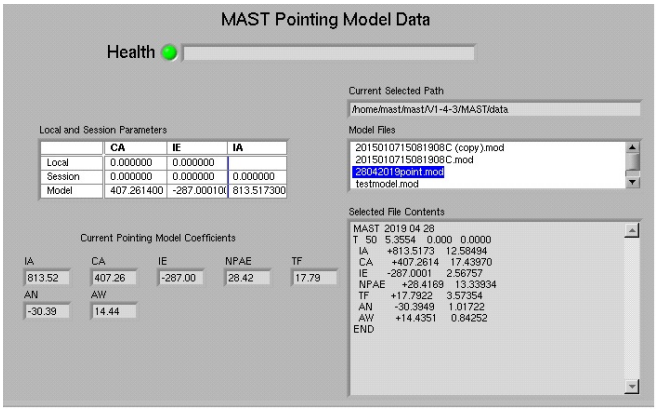

Figure 6. TCS window showing the new loaded pointing parameters. These parameters will be used by the TCS while pointing the telescope to a different parts of the sky.

- **CA** - non-perpendicularity of elevation and pointing axes
- **AN** - NS misalignment of the azimuth axis
- **AW** - EW misalignment of the azimuth axis
- **NPAE** - non-perpendicularity of azimuth and elevation axis
- **TF** - classical tube flexure

Figure 5 shows the TCS window where results from the TPOINT fitting procedure are plotted. Figure 6 shows the TCS window, where the new fitted parameters are loaded and ready for observations.

4. Acknowledgement

The editor, D. Pallamraju, thanks two anonymous reviewers for their assistance in the evaluation of this technical note.

References

Venkatakrishnan P., Mathew S. K., Srivastava N., Bayanna A. R., Kumar B., Ramya Bireddy, Jain N., Saradava M., The multi application solar telescope. *Current Science*, 113(4), 2017.

Mathew S. K., Bayanna A. R., Tiwary A. R., Ramya B., Venkatakrishnan P., First observations from the multi-application solar telescope (mast) narrow-band imager. *Solar Physics*, 292:106, 2017.

Wallace P. T., TPOINT – Telescope Pointing Analysis System *Starlink 1994*, User Note 100

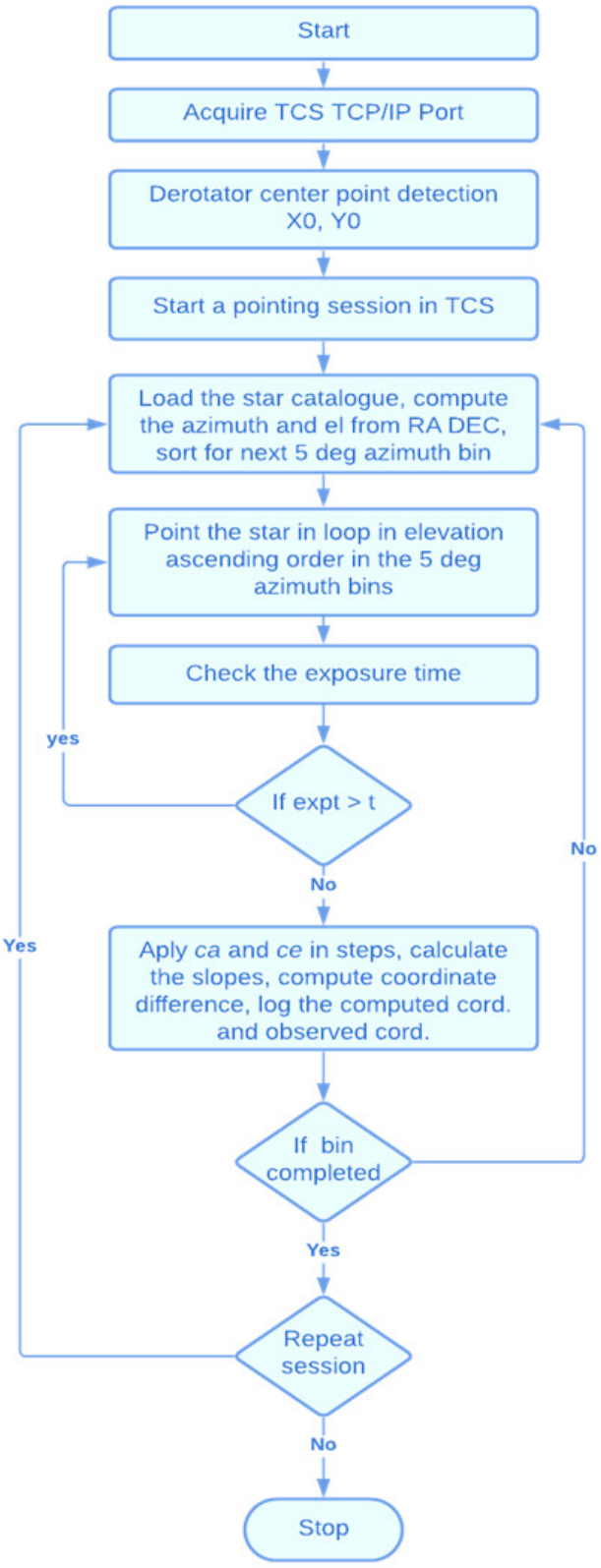

Figure 7. Flow chart of the automated pointing procedure. The real implementation can be understood from the program listed in the appendix.

Appendix

```
IDL program for automated data acquisition for pointing model in MAST
; ------------------------------------
; Program for Pointing data collection |
; Shibu K. Mathew & Ramya B, USO-PRL   |
;  ------------------------------------

common AVE_RAD,rtd,dtr    ; Common variables

;--- Main widget program starts here --
PRO pointing
rtd=180.d0/!dpi & dtr=!dpi/180.d0
LOC='F:\PROGRAMS\POINTINGMODEL\'
cd,LOC
yof=20
xof=110
wsize_xof=57
wsize_yof=0.0
img_y=860 & img_x=860

; ----- IDL widget for buttons, windows and tables -----

set_plot,'win'
base = widget_base(title='MAST Pointing test widget:17/06/2014',xsize=883,ysize=925,
bitmap='MAST.bmp')
BUT00= widget_button(base, value='Derotator_C.bmp',xsize=213,ysize=30,xoffset=10,yoffset=10,$
/bitmap,uvalue='start_D C',tooltip="Derotator center finding")
BUT11= widget_button(base, value='Center_star.bmp',xsize=213,ysize=30,xoffset=230,yoffset=10,$
/bitmap,uvalue='Center_star',tooltip="Centering star")
text1=widget_text(base,value='Current Exposure Time = 0.0 ms',xsize=30,ysize=1.0,xoffset=480,$
yoffset=15,uvalue='text1',/editable,unit=0)
text2=widget_text(base,value='Current rot. position =  0.0 (Deg)',xsize=30,ysize=1.0,xoffset=480+200,$
yoffset=15,uvalue='text2',/editable,unit=0)
IMAGE=widget_draw(base,frame=10,xsize=860,ysize=860,xoffset=10,yoffset=50,uvalue='image')
widget_control,IMAGE,get_value=index
wset,index
widget_control, base, /REALIZE
xmanager, 'pointing',base
END

;----- Event driven routines starts here ------------------
PRO pointing_event, ev
set_plot,'win'

; --------- Finding the center of the derotator --------------
; A bright star is brought into the FOV of the telescope      |
; the derotator is rotated with a step size of 10 degree      |
; steps, acquiring few images for each step. If the star is   |
; not at center, the derotator makes a circular star track    |
; a circle is fitted to this track and center of the circle   |
; is defined as the derotator center.                         |
;-------------------------------------------------------------
widget_control,ev.ID,get_uvalue=D0
if (D0 eq 'start_DC')then begin ;START_DC IF
widget_control,ev.ID,set_uvalue='running',set_value='Auto_expos.bmp',/bitmap
```

```
Parent  = 'F:\PROGRAMS\POINTINGMODEL\'
DRC_dir = 'F:\PROGRAMS\POINTINGMODEL\DRC_IMG\'
SC_dir  = 'F:\PROGRAMS\POINTINGMODEL\SC_IMG\'
cd,Parent

;-------------------------------------------------------------
; Pointing session file, all the useful data and parameters are |
; written out to this file and saved                           |
;-------------------------------------------------------------
sessiont=systime(/utc)
sessiont=strsplit(sessiont,/extract)
sessionf='RC'+sessiont(4)+sessiont(1)+sessiont(2)+sessiont(0)+strmid(sessiont(3),0,2)+$
strmid(sessiont(3),3,2)+strmid(sessiont(3),6,2)+'.ses'
tempf='RC'+sessiont(4)+sessiont(1)+sessiont(2)+sessiont(0)+strmid(sessiont(3),0,2)+$
strmid(sessiont(3),3,2)+strmid(sessiont(3),6,2)+'.ps'
get_lun,unit_ses
openw,unit_ses,DRC_dir+sessionf
spawn,"del "+DRC_dir+'Rot*.sav',/hide
printf,unit_ses,'        Pointing session     '
printf,unit_ses,'Date & time : ',sessiont
printf,unit_ses,'----------------------------'
printf,unit_ses,'Filename   -----   Angles   ------   Centroid points'

exposure,expt=expt,flagexp=flagexp   ; Auto-exposure routine
if flagexp eq -1 then begin
str = "Not enough light level, source check or faint star!"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
widget_control,ev.ID,set_uvalue='NotDone_exp',set_value='Derotator_start.bmp',/bitmap
goto,endwid1
endif

if flagexp eq 0 then begin
str= "Current exposure time = "+strcompress(string(expt),/remove_all)+" ms"
widget_control, ev.ID+2, SET_VALUE=str,set_uvalue='expose'
widget_control,ev.ID,set_uvalue='Done_exp',set_value='Derotator_start.bmp',/bitmap
endif

;---------------------------------------------
; Establishing the connection with the TCS PC   |
; IDL socket programming routines are used for  |
; this purpose                                  |
;---------------------------------------------
connect_TCS,flagTCS=flagTCS,unitTCS=unitTCS
if flagTCS eq -1 then begin ;FLAGTCS IF
str = "  Connection 1 Rejected!"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
goto,endwid1
endif

if flagTCS eq 0 then begin
str = "  Connection 1 Accepted!"
widget_control, ev.ID+2, set_value=str,set_uvalue='running'
endif
```

```
input="do rotmove angle=0.00000"
command_done,unitTCS=unitTCS,input=input,flagDone=flagDone
; Initializing the derotator to zero position
if flagDone eq -1 then begin
str= "Prob. to move to Zero!"
widget_control, ev.ID+3, set_value=str,set_uvalue='text2'
goto,endwid1
endif

if flagDone eq 0 then begin
str= "Rotator Moved to 0.0 Deg "
widget_control, ev.ID+3, set_value=str,set_uvalue='text2'
endif

str= "ROTATOR STARTED "
widget_control, ev.ID+2, set_value=str,set_uvalue='text2'

;-----------------------------------------------
;Stepping the derotator in 10 degree intervals |
;-----------------------------------------------
for i = 0.d0, 360.d0, 10.d0 do begin
input="do rotmove angle="+string(strtrim(string(double(i)),2))
command_done,unitTCS=unitTCS,input=input,flagDone=flagDone

if flagDone eq -1 then begin
str= "Prob. to move rotator!"
wiget_control, ev.ID+3, set_value=str,set_uvalue='text2'
goto,endwid1
endif

if (flagDone eq 0) then begin
str= "Rotator moving"
widget_control, ev.ID+3, set_value=str,set_uvalue='text2'
input='get rotator'
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
if outp(3) ne 'unknown' then RP=(outp(3)*1.d0)/(-2.0)
str= "Current rot. position =  "+strmid(strcompress(string(RP),/remove_all),0,6)+"(Deg)"
widget_control, ev.ID+3, set_value=str,set_uvalue='text2'
endif

;----------------------------------------|
; ---- Image acquisition starts here -----|
;----------------------------------------|
for j=0,4 do begin
str0="sensi_AE "+string(fix(expt)) ; Camera routine written in C is used
; for image acquisition, 5 images in
; each step for better S/N
spawn,str0,/hide
r=readfits("TEMP.fits",header,/silent)
if j eq 0 then begin
size_image=size(r)
Temp=fltarr(size_image(1),size_image(2))
```

```
cenx=size_image(1)/2. & ceny=size_image(2)/2.
endif
tvscl,reform(r(cenx-860/2.0:cenx+860/2.0,ceny-860/2.0:ceny+860/2.0))
Temp=Temp+r
spawn,"del TEMP.fits",/hide
endfor
; End of image acquisition, averaging 5 images
Image=Temp/5.0
caf='0000'+strcompress(string(fix(i)),/remove_all)
caf=strmid(caf,strlen(caf)-3,3)
fname=DRC_dir+'Rot'+caf+'.sav'
save,filename=fname,Image,/xdr ; save the averaged file
endfor

str= "ROTATOR MOVED "
widget_control, ev.ID+2, set_value=str,set_uvalue='text2'
input="do rotmove angle=0.00000"
command_done,unitTCS=unitTCS,input=input,flagDone=flagDone

if flagDone eq -1 then begin
str= "Prob. in moving Back"
widget_control, ev.ID+3, set_value=str,set_uvalue='text2'
goto,endwid1
endif

if (flagDone eq 0) then begin
str= "Rotator returning!"
widget_control, ev.ID+3, set_value=str,set_uvalue='text2'
input='get rotator'
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
if outp(3) ne 'unknown' then RP=(outp(3)*1.d0)/(-2.0)
str= "Moved back to "+strmid(strcompress(string(RP),/remove_all),0,6)+"(Deg)"
widget_control, ev.ID+3, set_value=str,set_uvalue='text2'
endif

cd,'F:\PROGRAMS\POINTINGMODEL\DRC_IMG\'
f=findfile("Rot*.sav")
n=n_elements(f)
angles=fltarr(n)
xy_cord=fltarr(2,n)

;-------------------------------------------------------------------------|
; Reading the saved files and Calculating the center of the de-rotator |
;-------------------------------------------------------------------------|
for i=0,n-1 do begin
angles(i)=strmid(strcompress(f(i),/remove_all),3,3)*1.d0
restore,f(i)
ss=size(Image)
if i eq 0 then ave_image=fltarr(ss(1),ss(2))
ave_image=ave_image+Image
Temp_columns=total(Image,1)/ss(1)
yc=max(Temp_columns,index_y)
```

```
yc=index_y
Temp_rows=total(Image,2)/ss(2)
xc=max(Temp_rows,index_x)
xc=index_x
xy_cord(0,i)=xc & xy_cord(1,i)=yc
printf,unit_ses,f(i),'  ',angles(i),'  ',xc, '  ',yc
endfor
ave_image=ave_image/(n*1.0)

radius=fltarr(12) & xc=fltarr(12) & yc=fltarr(12)
for i=0.0,11.0 do begin
x0=xy_cord(0,i) & y0=xy_cord(1,i)
x1=xy_cord(0,i+12)& y1=xy_cord(1,i+12)
x2=xy_cord(0,i+24)& y2=xy_cord(1,i+24)
cir_3pnt,[x0,x1,x2],[y0,y1,y2],rad,xx,yy
radius(i)=rad & xc(i)=xx & yc(i)=yy
endfor
DR_CEN=[mean(xc),mean(yc)]
AVE_RAD=mean(radius)
save,filename='DRC.sav',DR_CEN,AVE_RAD,/xdr

forplot=[DR_CEN(0)-(cenx-860/2.0)-1,DR_CEN(1)-(ceny-860/2.0)-1]
dis_image=reform(ave_image(cenx-860/2.0:(cenx+860/2.0)-1,ceny-860/2.0:(ceny+860/2.0)-1))
tvscl,dis_image
contour,dis_image,levels=[1000000],/xst,/yst,pos=[0,0,1,1],/noerase,xthick=1.0,$
ythick=1.0,xticks=20,yticks=20,xminor=5,yminor=5,ticklen=0.01
oplot,[DR_CEN(0)-(cenx-860/2.0)-1,DR_CEN(0)-(cenx-860/2.0)-1],[0,10000],linestyle=1
oplot,[0,10000],[DR_CEN(1)-(ceny-860/2.0)-1,DR_CEN(1)-(ceny-860/2.0)-1],linestyle=1
str1="Derotator center : X0 = ["+strcompress(string(DR_CEN(0)),/remove_all)+"] "+$
"Y0 = ["+strcompress(string(DR_CEN(1)),/remove_all)+"]"
xyouts,20,20,str1,charsize=1.5,charthick=1.2
draw_circle,forplot(0),forplot(1),AVE_RAD,/device,linestyle=0,thick=1.5,npts=3000
str= "CENTER FOUND "
widget_control, ev.ID+2, set_value= str, set_uvalue='text2'

;-------------------------------------------------------------------|
; Saving Derotator center image in PS file                          |
;-------------------------------------------------------------------|
set_plot,'ps'
device,filename=DRC_dir+tempf,xsize=6,ysize=6,/inches,bits=8,xoffset=1.0,yoffset=2.5
tvscl,max(dis_image)-dis_image
contour,dis_image,levels=[1000000],/xst,/yst,pos=[0,0,1,1],/noerase,xthick=1.0,ythick=1.0,$
xticks=20,yticks=20,xminor=5,yminor=5,ticklen=0.01
oplot,[DR_CEN(0)-(cenx-860/2.0)-1,DR_CEN(0)-(cenx-860/2.0)-1],[0,10000],linestyle=1
oplot,[0,10000],[DR_CEN(1)-(ceny-860/2.0)-1,DR_CEN(1)-(ceny-860/2.0)-1],linestyle=1
str1="Derotator center : X0 = ["+strcompress(string(DR_CEN(0)),/remove_all)+"] "+$
"Y0 = ["+strcompress(string(DR_CEN(1)),/remove_all)+"]"
xyouts,20,20,str1,charsize=1.5,charthick=1.2
draw_circle,forplot(0),forplot(1),AVE_RAD,/device,linestyle=0,thick=1.5,npts=3000,color=255
device,/close
set_plot,'win'
printf,unit_ses,'Derotator center on the CCD at XC = ',DR_CEN(0), ' YC = ',DR_CEN(1)
cd,Parent
widget_control,ev.ID,set_uvalue='Done',set_value='Done_DRC.bmp',/bitmap
printf,unit_ses,'------ end of session --------------'
```

```
;-------------------------------------------------------------------------|
; Preparing the telescope axis for tracking the stars                     |
; Track off, Initialization of the telescope                              |
;-------------------------------------------------------------------------|
input="do track state=off"
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
systemactivity,unit=unitTCS
input="do umacengage state=off"
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
systemactivity,unit=unitTCS

input="do datum"
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
wait,15
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)

systemactivity,unit=unitTCS
input="do umacengage state=on"
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)

systemactivity,unit=unitTCS
free_lun,unit_ses
free_lun,unitTCS
close,/all
endwid1:
endif

;-------------------------------------------------------------------------|
; Tracking each star starts here                                          |
; Centering the star by calculating the Delta Ca and Delta Ce values      |
; Moving the star to center found by applying ca and ce                   |
; Log the star by adding to pointing data                                 |
;-------------------------------------------------------------------------|
widget_control,ev.ID,get_uvalue=D0 if (D0 eq 'Center_star') then
widget_control,ev.ID,set_uvalue='running',set_value='Connecting.bmp',/bitmap

Parent  = 'F:\PROGRAMS\POINTINGMODEL\'
DRC_dir = 'F:\PROGRAMS\POINTINGMODEL\DRC_IMG\'
SC_dir  = 'F:\PROGRAMS\POINTINGMODEL\SC_IMG\'
cd,Parent
restore,DRC_dir+'DRC.sav'
```

```
connect_TCS,flagTCS=flagTCS,unitTCS=unitTCS
if flagTCS eq -1 then begin
str = "  Connection 2 Rejected!"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
goto,endwid2
endif
if flagTCS eq 0 then begin  str = "  Connection 2 Accepted!"

widget_control, ev.ID+2,
set_value=str,set_uvalue='running'
endif
;-----------------------------------------------------------------------------|
; Preparing text file to save all the starts information that are tracked |
;-----------------------------------------------------------------------------|
sessiont=systime(/utc)
sessiont=strsplit(sessiont,/extract)
sessionf='SC'+sessiont(4)+sessiont(1)+sessiont(2)+sessiont(0)+strmid(sessiont(3),0,2)+$
strmid(sessiont(3),3,2)+strmid(sessiont(3),6,2)+'.ses'
sessionstar_SES=sessionf+'.uso'
get_lun,unit_ses
openw,unit_ses,SC_dir+sessionf
printf,unit_ses,'        Pointing session     '
printf,unit_ses,'Date & time : ',sessiont
printf,unit_ses,'----------------------------'
printf,unit_ses,$
'Sr no. ---- Star name   -----   Star RA   ------   Star DEC ---- Star Mag --- Status'
printf,unit_ses,' ------------------------------------------'
printf,unit_ses,''
get_lun,unit_STAR
openw,unit_STAR,SC_dir+sessionstar_SES

input="do pointnew"
command_done,unitTCS=unitTCS,input=input,flagDone=flagDone
if flagDone eq -1 then begin
str = "Unable to open dat file"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
goto,endwid2
endif
if flagDone eq 0 then begin
str = "Opened dat file"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
endif

flags1=0
star_count=0
;-----------------------------------------------------------------------------|
; For loop for next session once the current session completes            |
;-----------------------------------------------------------------------------|
for jm=0,4 do begin
BIN_START=285.0 & BIN_SIZE=5.0
;-----------------------------------------------------------------------------|
; For loop for next Zone in Azimuth bins                                    |
;-----------------------------------------------------------------------------|
for kk=0,44 do begin
```

```
star_catalogue,BIN_START=BIN_START,BIN_SIZE=BIN_SIZE,STARNAME=STARNAME,STARRA=STARRA,$
STARDEC=STARDEC,STARPMRA=STARPMRA,STARPMDEC=STARPMDEC,EQUINOX=EQUINOX,STARMAG=STARMAG,$
r_index=r_index,flagCAT=flagCAT
if flagCAT eq -1 then begin
str ="No stars in "+strcompress(string(BIN_START),/remove_all)+"Next Bin!"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
endif
if flagCAT eq 0 then begin
n_stars=n_elements(STARNAME)
str ="Zone "+strcompress(string(BIN_START),/remove_all)+" : "+strcompress(string(n_stars),/remove_all)+$
" stars"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'

;-------------------------------------------------------------------------|
; Sorted Order for starts, Track each star accordingly                    |
;-------------------------------------------------------------------------|
for ij=0,n_stars-1 do begin
input="do target name="+STARNAME(r_index(ij))+" ra="+STARRA(r_index(ij))+" dec="+STARDEC(r_index(ij))+$
" pmra="+STARPMRA(r_index(ij))+" pmdec="+STARPMDEC(r_index(ij))+" equinox="+EQUINOX(r_index(ij))
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
if outp(2) eq -1 then begin
str = STARNAME(r_index(ij))+"   Not observable!"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
printf,unit_ses,STARNAME(r_index(ij)),'   ',STARRA(r_index(ij)),'   ',STARDEC(r_index(ij)),'    ',$
STARMAG(r_index(ij)),' Target not observable'
goto,endnext
endif

if outp(2) eq 0 then begin
str = STARNAME(r_index(ij))+"   Tracked"
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
widget_control,ev.ID,set_uvalue='running',set_value='StarRADEC.bmp',/bitmap
if flags1 eq 0 then begin
x=0
input="do track state=on"
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
flags1=1
endif
endif
if x  eq 0 and jm eq 0 then begin
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
x=x+1
endif
if (outp(0) eq 'done') then flagTRACK=0
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
if (outp(0) eq 'done') then flagTRACK=0
```

```
;-------------------------------------------------------------------------|
; Skip Faint Stars as they take longer time to calculate slopes and center|
;-------------------------------------------------------------------------|
exposure,expt=expt,flagexp=flagexp
if flagexp eq -1 then begin
str = STARNAME(r_index(ij))+"   Too faint!"
widget_control, ev.ID+1, set_value=str,set_uvalue='skipping'
printf,unit_ses,STARNAME(r_index(ij)),'   ',STARRA(r_index(ij)),'   ',STARDEC(r_index(ij)),'    ',$
STARMAG(r_index(ij)),' Target too faint'
goto,endnext
endif
if flagexp eq 0 then begin
str= "Current exposure time = "+strcompress(string(expt),/remove_all)+" ms"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
str = 'ID : '+STARNAME(r_index(ij))
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
endif
spawn,"del "+parent+'TEMP.fits',/hide
spawn,"del "+SC_dir+'Ca*.sav',/hide
spawn,"del "+SC_dir+'Ce*.sav',/hide

ca=-20.d0 & ce=0.0
docace,ca=ca,ce=ce,unitTCS=unitTCS,flagCACE=flagCACE
widget_control,ev.ID,set_uvalue='running',set_value='CASlope.bmp',/bitmap
if flagCACE eq -1 then begin
str= "Not moved to -20, 0"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
str = 'ID : '+STARNAME(r_index(ij))
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
goto,endwid2
endif

if flagCACE eq 0 then begin
str= "Moved to -20, 0"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
str = 'ID : '+STARNAME(r_index(ij))
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
endif

;-------------------------------------------------------------------------|
; Steps in CA & CE to calculate slope                                     |
;-------------------------------------------------------------------------|
for i = 0.d0,4.0 do begin
ca=10.d0 & ce=0.d0
if i eq 0 then ca=0.d0
docace,ca=ca,ce=ce,unitTCS=unitTCS,flagCACE=flagCACE
if flagCACE eq -1 then begin
str= "Not moved to "+strcompress(string((i*10)-20.0),/remove_all)+", 0.0 "
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
goto,endwid2
endif
if flagCACE eq 0 then begin
str= "Moved to "+strcompress(string((i*10)-20.0),/remove_all)+", 0.0 "
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
endif
```

```
;-------------------------------------------------------------------------|
; Move ca in steps of 10" from -20" to +20" to find the slope             |
;-------------------------------------------------------------------------|
for j=0,4 do begin
str0="sensi_AE "+string(fix(expt))
spawn,str0,/hide
r=readfits("TEMP.fits",header,/silent)
if j eq 0 then begin
size_image=size(r)
Temp=fltarr(size_image(1),size_image(2))
cenx=size_image(1)/2. & ceny=size_image(2)/2.
endif
tvscl,reform(r(cenx-860/2.0:cenx+860/2.0,ceny-860/2.0:ceny+860/2.0))
Temp=Temp+r
spawn,"del TEMP.fits",/hide
endfor
Image=Temp/5.0
tvscl,reform(Image(cenx-860/2.0:cenx+860/2.0,ceny-860/2.0:ceny+860/2.0))
caf='0000'+strcompress(string(fix(i)),/remove_all)
caf=strmid(caf,strlen(caf)-3,3)
fname=SC_dir+'Ca'+caf+'.sav'
save,filename=fname,Image,/xdr
endfor

ca=-20.0 & ce=0.d0
docace,ca=ca,ce=ce,unitTCS=unitTCS,flagCACE=flagCACE
if flagCACE eq -1 then begin
str= "Not moved to 0, 0"
widget_control, ev.ID+1, SET_VALUE=str,set_uvalue='expose'
str = 'ID : '+STARNAME(r_index(ij))
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
goto,endwid2
endif

if flagCACE eq 0 then begin
str= "Moved to 0, 0"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
str = 'ID : '+STARNAME(r_index(ij))
widget_control, ev.ID+2, set_value=str,set_uvalue='skipping'
endif
ca=0.d0 & ce=-20.0
docace,ca=ca,ce=ce,unitTCS=unitTCS,flagCACE=flagCACE
widget_control,ev.ID,set_uvalue='running',set_value='CESlope.bmp',/bitmap
if flagCACE eq -1 then begin
str= "Not moved to 0, -20"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
goto,endwid2
endif
if flagCACE eq 0 then begin
str= "Moved to 0, -20"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
endif

;-------------------------------------------------------------------------|
```

```
; Move ce in steps of 10" from -20" to +20" to find the slope             |
;-------------------------------------------------------------------------|
for i = 0.d0,4.0  do begin
ca=0.d0 & ce=10.d0
if i eq 0 then ce=0.d0
docace,ca=ca,ce=ce,unitTCS=unitTCS,flagCACE=flagCACE
if flagCACE eq -1 then begin
str= "Not moved to 0.0, "+strcompress(string((i*10)-20.0),/remove_all)
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
goto,endwid2
endif
if flagCACE eq 0 then begin
str= "Moved to 0.0, "+strcompress(string((i*10)-20.0),/remove_all)
widget_control, ev.ID+1,set_value=str,set_uvalue='expose'
endif
for j=0,4 do begin
str0="sensi_AE "+string(fix(expt))
spawn,str0,/hide
r=readfits("TEMP.fits",header,/silent)
if j eq 0 then begin
size_image=size(r)
Temp=fltarr(size_image(1),size_image(2))
cenx=size_image(1)/2. & ceny=size_image(2)/2.
endif
tvscl,reform(r(cenx-860/2.0:cenx+860/2.0,ceny-860/2.0:ceny+860/2.0))
Temp=Temp+r
spawn,"del TEMP.fits",/hide
endfor
Image=Temp/5.0
tvscl,reform(Image(cenx-860/2.0:cenx+860/2.0,ceny-860/2.0:ceny+860/2.0))
cef='0000'+strcompress(string(fix(i)),/remove_all)
cef=strmid(cef,strlen(cef)-3,3)
fname=SC_dir+'Ce'+cef+'.sav'
save,filename=fname,Image,/xdr
endfor
ca=0.d0 & ce=-20.d0
docace,ca=ca,ce=ce,unitTCS=unitTCS,flagCACE=flagCACE
if flagCACE eq -1 then begin
str= "Not moved to 0, 0"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
goto,endwid2
endif
if flagCACE eq 0 then begin
str= "Moved to 0, 0"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
endif

;-------------------------------------------------------------------------|
; Calculation of slopes from data obtained                                |
;-------------------------------------------------------------------------|
widget_control,ev.ID,set_uvalue='running',set_value='computing.bmp',/bitmap
cd,SC_dir
f=findfile("Ca*.sav")
n=n_elements(f)
xca=fltarr(n) & yca=fltarr(n) & del_caa=fltarr(n)
```

```
for i=0,n-1 do begin
del_ca=(i*10.0)-20.d0
restore,f(i)
find_centroid,Image,xc=xc,yc=yc
xca(i)=xc & yca(i)=yc & del_caa(i)=del_ca
endfor
coeff=poly_fit(xca,yca,1,yfit=yfit)
dxca=xca(n-1)-xca(0)
dyca=yca(n-1)-yca(0)
dca=del_caa(n-1)-del_caa(0)
dxca=dxca/dca & dyca=dyca/dca
f=findfile("Ce*.sav")
n=n_elements(f)
xce=fltarr(n) & yce=fltarr(n) & del_cae=fltarr(n)
for i=0,n-1 do begin
del_ce=(i*10.0)-20.d0
restore,f(i)
find_centroid,Image,xc=xc,yc=yc
xce(i)=xc & yce(i)=yc & del_cae(i)=del_ce
endfor
coeff=poly_fit(xce,yce,1,yfit=yfit)
dxce=xce(n-1)-xce(0)
dyce=yce(n-1)-yce(0)
dce=del_cae(n-1)-del_cae(0)
dxce=dxce/dce & dyce=dyce/dce
TRANS=fltarr(2,2)
TRANS(0,0)=dxca & TRANS(0,1)=dyca
TRANS(1,0)=dxce & TRANS(1,1)=dyce

;-------------------------------------------------------------------------|
; Put the star at the center by calculating offsets required              |
;-------------------------------------------------------------------------|
cd,parent
spawn,"del TEMP.fits",/hide
for j=0,4 do begin
if j eq 0 then
Temp=fltarr(size_image(1),size_image(2))
str0="sensi_AE "+string(fix(expt))
spawn,str0,/hide
r=readfits("TEMP.fits",header,/silent)
tvscl,reform(r(cenx-860/2.0:cenx+860/2.0,ceny-860/2.0:ceny+860/2.0))
Temp=Temp+r
spawn,"del TEMP.fits",/hide
endfor
Image=Temp/5.0
unshifted=Image
find_centroid,Image,xc=xc,yc=yc
dxc=DR_CEN(0)-xc & dyc=DR_CEN(1)-yc
TRANSINVERSE=IMSL_INV(TRANS)
dxdy=fltarr(1,2)
dxdy(0,0)=dxc & dxdy(0,1)=dyc
DCADCE=TRANSINVERSE##dxdy
delta_ca=reform(DCADCE(0,0)) & delta_ce=reform(DCADCE(0,1))
docace,ca=delta_ca,ce=delta_ce,unitTCS=unitTCS,flagCACE=flagCACE
widget_control,ev.ID,set_uvalue='running',set_value='Centering.bmp',/bitmap
```

```
if flagCACE eq -1 then begin
str= "Not moved to  "+strcompress(string(delta_ca),/remove_all)+', '+$
strcompress(string(delta_ce),/remove_all)
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
goto,endwid2
endif

if flagCACE eq 0 then begin str= "Moved to  "+strcompress(string(delta_ca),/remove_all)+', '+$
strcompress(string(delta_ce),/remove_all)
widget_control, ev.ID+1,
set_value=str,set_uvalue='expose'
endif
spawn,"del TEMP.fits",/hide
for j=0,4 do begin
if j eq 0 then Temp=fltarr(size_image(1),size_image(2))
str0="sensi_AE "+string(fix(expt))
spawn,str0,/hide
r=readfits("TEMP.fits",header,/silent)
tvscl,reform(r(cenx-860/2.0:cenx+860/2.0,ceny-860/2.0:ceny+860/2.0))
Temp=Temp+r
spawn,"del TEMP.fits",/hide
endfor
Image=Temp/5.0
shifted=Image
dis_image=(unshifted+shifted)/2.d0
find_centroid,Image,xc=xc,yc=yc
forplot=[DR_CEN(0)-(cenx-860/2.0)-1,DR_CEN(1)-(ceny-860/2.0)-1]
dis_image=reform(dis_image(cenx-860/2.0:(cenx+860/2.0)-1,ceny-860/2.0:(ceny+860/2.0)-1))
set_plot,'win'
tvscl,dis_image
contour,dis_image,levels=[1000000],/xst,/yst,pos=[0,0,1,1],/noerase,xthick=1.0,ythick=1.0,$
xticks=20,yticks=20,xminor=5,yminor=5,ticklen=0.01
oplot,[DR_CEN(0)-(cenx-860/2.0)-1,DR_CEN(0)-(cenx-860/2.0)-1],[0,10000],linestyle=1
oplot,[0,10000],[DR_CEN(1)-(ceny-860/2.0)-1,DR_CEN(1)-(ceny-860/2.0)-1],linestyle=1
str1="Applied ca = ["+strcompress(string(delta_ca),/remove_all)+"] "+$
"ce = ["+strcompress(string(delta_ce),/remove_all)+"]"
xyouts,650,700,'Star ID: '+strcompress(STARNAME(r_index(ij)),/remove_all),charsize=1.5,charthick=1.2
xyouts,20,20,str1,charsize=1.5,charthick=1.2
xyouts,20,40,'DX : '+strcompress(string(DR_CEN(0)-xc),/remove_all)+' , DY :'+$
strcompress(string(DR_CEN(1)-yc),/remove_all),charsize=1.5,charthick=1.2
draw_circle,forplot(0),forplot(1),5,/data,linestyle=1,thick=1.5,npts=300,color=255
set_plot,'ps'
device,filename=SC_Dir+sessionf+strcompress(STARNAME(r_index(ij)),/remove_all)+'.ps',xsize=6,$
ysize=6,/inches,bits=8,xoffset=1,yoffset=2.5
tvscl,max(dis_image)-dis_image
contour,dis_image,levels=[1000000],/xst,/yst,pos=[0,0,1,1],/noerase,xthick=1.0,ythick=1.0,$
xticks=20,yticks=20,xminor=5,yminor=5,ticklen=0.01
oplot,[DR_CEN(0)-(cenx-860/2.0)-1,DR_CEN(0)-(cenx-860/2.0)-1],[0,10000],linestyle=1
oplot,[0,10000],[DR_CEN(1)-(ceny-860/2.0)-1,DR_CEN(1)-(ceny-860/2.0)-1],linestyle=1
str1="Applied ca = ["+strcompress(string(delta_ca),/remove_all)+"] ce = ["+$
strcompress(string(delta_ce),/remove_all)+"]"
xyouts,650,700,'Star ID: '+strcompress(STARNAME(r_index(ij)),/remove_all),charsize=1.5,charthick=1.2
xyouts,20,20,str1,charsize=1.5,charthick=1.2
xyouts,20,50,'DX : '+strcompress(string(DR_CEN(0)-xc),/remove_all)+' , DY : '+$
strcompress(string(DR_CEN(1)-yc),/remove_all),charsize=1.5,charthick=1.2
```

```
draw_circle,forplot(0),forplot(1),5,/data,linestyle=1,thick=1.5,npts=300
device,/close
set_plot,'win'

;---------------------------------------------------------------------------|
; Star centered, add it to pointing data file                               |
;---------------------------------------------------------------------------|
input="do pointadd"
command_done,unitTCS=unitTCS,input=input,flagDone=flagDone
widget_control,ev.ID,set_uvalue='running',set_value='Adding.bmp',/bitmap
if flagDone eq -1 then begin
str= "Point not added"
widget_control, ev.ID+1,
set_value=str,set_uvalue='expose'
goto,endwid2
endif

if flagDone eq 0 then begin
sessiont_1=systime(/utc)
sessiont_1=strsplit(sessiont_1,/extract)
printf,unit_STAR,strcompress(sessiont_1,/remove_all),star_count,' ',STARNAME(r_index(ij)),$
(DR_CEN(0)-xc),(DR_CEN(1)-yc)
star_count=star_count+1
str= "Star "+strcompress(string(star_count),/remove_all)+"   Added"
widget_control, ev.ID+1, set_value=str,set_uvalue='expose'
printf,unit_ses,STARNAME(r_index(ij)),'   ',STARRA(r_index(ij)),'   ',STARDEC(r_index(ij)),$
'    ',STARMAG(r_index(ij)),' Point logged'
endif
endnext:
endfor
endif
BIN_START=BIN_START-BIN_SIZE
print,'Next bin is : ',BIN_START
endfor
yesno='y'
print,'--------- PRESENT SESSION ENDED --------------'
print,'PROCEED TO ANOTHER SESSION (y/n)?'
read,yesno
if yesno eq 'n' then goto,endwid2
endfor
endwid2:
close,/all
endif
end

;---------------------------------------------------------------------------|
; All the Sub Routines Starts Here                                          |
;---------------------------------------------------------------------------|

;---------------------------------------------------------------------------|
; Find centroid of star                                                     |
;---------------------------------------------------------------------------|
pro find_centroid,img,xc=xc,yc=yc
ss=size(img)
temp1=fltarr(ss(1))
```

```
temp2=fltarr(ss(2))
temp1=total(img,2)/ss(2)
temp2=total(img,1)/ss(1)
r=max(temp1,index)
xc=index
r=max(temp2,index)
yc=index
end

;-------------------------------------------------------------------------|
; Sort the star catalogue according to bins                               |
;-------------------------------------------------------------------------|
pro star_catalogue, BIN_START=BIN_START, BIN_SIZE=BIN_SIZE,STARNAME=STARNAME,STARRA=STARRA,STARDEC=STARDEC
STARPMRA=STARPMRA,STARPMDEC=STARPMDEC,$ EQUINOX=EQUINOX,STARMAG=STARMAG,r_index=r_index,flagCAT=flagCAT
get_lun,unit
f='F:\PROGRAMS\POINTINGMODEL\starcatalog_origi.txt'
openr,unit,f
s=''
count=0
while not eof(unit) do begin
readf,unit,s
count=count+1
endwhile
STARNAME=strarr(count) & STARRA=strarr(count) & STARDEC=strarr(count) & STARPMRA=strarr(count)
STARPMDEC=strarr(count) & EQUINOX=strarr(count) & STARMAG=strarr(count)
AZI=fltarr(count) & ELE=fltarr(count)
free_lun,unit
get_lun,unit
f='F:\PROGRAMS\POINTINGMODEL\starcatalog_origi.txt'
openr,unit,f
for i=0,count-1 do begin
s=''
readf,unit,s
s=strsplit(s,/extract)
STARNAME(i)=strcompress(s(0)) & STARPMRA(i)=strcompress(s(7))& STARPMDEC(i)=strcompress(s(8))
EQUINOX(i)=strcompress(s(9))
STARRA(i)=s(1)+' '+s(2)+' '+s(3)
STARDEC(i)=s(4)+' '+s(5)+' '+s(6)
radeg=s(1)*1.d0 & ramin=s(2)*1.d0 & rasec=s(3)*1.d0
decdeg=s(4)*1.d0 & decmin=s(5)*1.d0 & decsec=s(6)*1.d0

RADECTOALTAZ,radeg,ramin,rasec,decdeg,decmin,decsec,azimuth=azimuth,elevation=elevation
AZI(i)=azimuth & ELE(i)=elevation
if n_elements(s) lt 11 then STARMAG(i)=strcompress(string(6.0),/remove_all)$
else STARMAG(i)=strcompress(s(10))
endfor
starmag_temp=AZI*1.d0
sort,starmag_temp,result=result,direction='up',r_index=r_index
z=where(((result le BIN_START) and (result gt BIN_START-BIN_SIZE)) and ((result ge 65.0 )$
and (result le 285.0)))
if z(0) eq -1 then flagCAT=-1 else flagCAT=0
if flagCAT eq 0 then begin
ELE=reform(ELE(z))
AZI=reform(AZI(z))
STARRA=reform(STARRA(r_index(z)))
```

```
STARDEC=reform(STARDEC(r_index(z)))
STARNAME=reform(STARNAME(r_index(z)))
STARPMRA=reform(STARPMRA(r_index(z)))
STARPMDEC=reform(STARPMDEC(r_index(z)))
STARMAG=reform(STARMAG(r_index(z)))
EQUINOX=reform(EQUINOX(r_index(z)))
ELESORT=ELE*1.d0
check='up'
IF (((285-BIN_START)/BIN_SIZE) mod 2) eq 0  then check='up' else check='down'
sort,ELESORT,result=result,direction=check,r_index=r_index
endif
free_lun,unit
end

;-------------------------------------------------------------------------|
; Calculation of AZ, EL from RA,DEC                                       |
;-------------------------------------------------------------------------|
pro RADECTOALTAZ,radeg,ramin,rasec,decdeg, decmin,decsec,azimuth=azimuth,elevation=elevation
longitude_uso=73.673888888
latitude_uso=24.60433333
dtr=!dpi/180.d0
rtd=180.d0/!dpi
filename_year='F:\PROGRAMS\POINTINGMODEL\tabB.dat'
filename_month='F:\PROGRAMS\POINTINGMODEL\tabA.dat'
ra=((radeg*1.d0)+(ramin/60.d0)+(rasec/(60.d0*60.d0)))*15.d0
dec=(decdeg*1.d0)+(decmin/60.d0)+(decsec/(60.d0*60.d0))
jul_day=systime(/julian,/utc)
CALDAT, jul_day, Month , Day , Year , Hour , Minute , Second
fday=(Hour+Minute/60.d0+Second/(60.d0*60.d0))/24.d0
years_days=fltarr(2,12)
month_days=fltarr(3,12)
get_lun,unit
openr,unit,filename_year
readf,unit,years_days
free_lun,unit
list_years=reform(years_days(0,*))
list_ydays=reform(years_days(1,*))
get_lun,unit
openr,unit,filename_month
readf,unit,month_days
free_lun,unit
list_months=reform(month_days(0,*))
list_nodays=reform(month_days(1,*))
list_lpdays=reform(month_days(2,*))
flag_ly=(Year/4.d0)-fix(Year/4.d0)
zy=where(list_years eq Year)
ydays=list_ydays(zy)*1.d0
zm=where(list_months eq Month)
if (flag_ly ne 0.0) then mdays=list_nodays(zm)
if (flag_ly eq 0.0) then mdays=list_lpdays(zm)
time_elapsed=fday+mdays*1.d0+Day*1.d0+ydays*1.d0
LST = 100.46+(0.985647*time_elapsed)+longitude_uso+(15.d0*(Hour+Minute/60.d0+Second/(60.d0*60.d0)))
LST = LST-fix(LST/360.d0)*360.d0
LST=LST(0)
HA=(LST-ra)
```

```
if HA lt 0.0 then HA=HA+360.d0
HA=HA(0)
SINALT=(sin(dec*dtr)*sin(latitude_uso*dtr))+(cos(dec*dtr)*cos(latitude_uso*dtr)*cos(HA*dtr))
ALT=asin(SINALT)*rtd
COSAZ=(sin(dec*dtr)-(sin(ALT*dtr)*sin(latitude_uso*dtr)))/(cos(ALT*dtr)*cos(latitude_uso*dtr))
AZ=acos(COSAZ)*rtd
if sin(HA*dtr) gt 0.0 then AZ=360.d0-AZ
elevation=ALT & azimuth=AZ
end

;-------------------------------------------------------------------------|
; Auto Exposure for image                                                  |
;-------------------------------------------------------------------------|
pro exposure,expt=expt,flagexp=flagexp
flagexp=0
expt=1.0
dyn_range=2500.0
for i=0,50 do begin
str0="sensi_AE "+string(fix(expt))
spawn,str0,/hide
r=readfits("TEMP.fits",header,/silent)
size_image=size(r)
cenx=size_image(1)/2.0 & ceny=size_image(2)/2.0
tvscl,reform(r(cenx-860/2.0:(cenx+860/2.0)-1,ceny-860/2.0:(ceny+860/2.0)-1))
spawn,"del TEMP.fits",/hide
max_int=max(r)
r=r*1.d0
expt=expt*dyn_range/max_int
if max_int gt dyn_range then goto,skip
if ((expt eq !VALUES.F_INFINITY) or (expt gt 1000.0)) then begin
flagexp=-1
goto, skip_end
endif
endfor
skip:
expt=0.95*expt
skip_end:
end

;-------------------------------------------------------------------------|
; Connection to TCS                                                       |
;-------------------------------------------------------------------------|
pro connect_TCS,flagTCS=flagTCS, unitTCS=unitTCS
s=''
flagTCS=0
get_lun,unitTCS
host='xxx.xx.xx.xxx'
port = xxxx
password='xxxxxxx'
SOCKET, unitTCS, host, port
readf,unitTCS,s
if s eq "Connect: OK" then print,"Connection Sucessful"
if s ne "Connect: OK" then begin
print,"Not connected, skipping the session"
flagTCS=-1
```

```
endif
readf,unitTCS,s
printf,unitTCS,password
readf,unitTCS,s
if s eq "Password accepted" then print,"Password accepted"
if s ne "Password accepted" then begin
print,"Wrong password"
flagTCS=-1
endif
end

;-------------------------------------------------------------------------|
; Check System Busy/ Idle                                                 |
;-------------------------------------------------------------------------|
pro systemactivity,unit=unit
z:
input="get systemactivity"
outp_1=''
printf,unit,input
readf,unit,outp_1
outp=strsplit(outp_1,/extract)
if outp(3) ne 'Idle' then goto,z
end

;-------------------------------------------------------------------------|
; Execute Command in TCS                                                  |
;-------------------------------------------------------------------------|
pro command_done,unitTCS=unitTCS,input=input, flagDone=flagDone
flagDone=0
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
if (outp(0) eq 'done') then flagDone=0 else flagDone=-1
end

;-------------------------------------------------------------------------|
; Move in CA and CE                                                       |
;-------------------------------------------------------------------------|
pro docace,ca=ca,ce=ce,unitTCS=unitTCS, flagCACE=flagCACE
flagCACE=0
input='do colloffset ca='+strtrim(string(ca),2)+'  ce='+strtrim(string(ce),2)
outp_1=''
printf,unitTCS,input
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
readf,unitTCS,outp_1
outp=strsplit(outp_1,/extract)
if (outp(0) eq 'done') then flagCACE=0 else flagCACE=-1
wait,5.0
end
```

PRL research encompasses the earth the sun immersed in the fields and radiations reaching from and to infinity, all that man's curiosity and intellect can reveal

पीआरएल के अनुसंधान क्षेत्र में समविष्ट हैं पृथ्वी एवं सूर्य जो निमीलित हैं चुंबकीय क्षेत्र एवं विकिरण में अनंत से अनंत तक जिन्हे प्रकट कर सकती है मानव की जिज्ञासा एवं विचारशक्ति