ALMA Memo 611

Atacama
Large
Millimeter
Array

Tools for using solar coordinate systems with ALMA

I. Skokić[1,2*], R. Brajša[1], M. Bárta[2], B. Chen[3], M. Shimojo[4], T. S. Bastian[5], A. S. Hales[6], K. Iwai[7], S. Kim[8], S. M. White[9], S. Wedemeyer[10] and S. Yu[3]

[1] Hvar Observatory, Faculty of Geodesy, University of Zagreb, Croatia
[2] Astronomical Institute, Czech Academy of Sciences, Ondřejov, Czech Republic
[3] Center For Solar-Terrestrial Research, New Jersey Institute of Technology, Newark, USA
[4] National Astronomical Observatory of Japan (NAOJ), Osawa, Mitaka, Japan
[5] National Radio Astronomy Observatory (NRAO), Charlottesville, USA
[6] Joint ALMA Observatory, Santiago, Chile
[7] Institute for Space–Earth Environmental Research, Nagoya University, Nagoya, Japan
[8] Korea Astronomy and Space Science Institute, Daejeon, Republic of Korea
[9] Space Vehicles Directorate, Air Force Research Laboratory, Kirtland AFB, USA
[10] Institute of Theoretical Astrophysics, University of Oslo, Norway

2018-06-21

Abstract

Solar observing with ALMA is very much different from other observing modes since it requires a number of special considerations. One of them is the difficulty of tracking a continuously moving target on the solar disc. The Sun itself is moving during observations in respect to the celestial background, while the position of a feature on the solar disc is affected by various motions such as differential rotation, meridional circulation and proper motion. In addition, the ALMA on-line control system uses a traditional celestial (equatorial) coordinate system, whereas solar physicists use heliocentric coordinates. Taking these considerations into account, methods and tools have been developed that enable easy preparation of ALMA solar observations and also conversion of the final images into the heliocentric coordinate frame to provide easy co-alignment and overlapping of ALMA solar images with images from other solar observatories.

[*] ivica.skokic@gmail.com

1 INTRODUCTION

The Sun is an extremely difficult target for interferometry. Following considerable work in commissioning and implementing solar observing modes, science observations of the Sun with ALMA have been possible since Cycle 4 (Bastian et al. 2018). However, there is still a lot of work ahead in implementing new methods, bands and improving calibration and imaging techniques. One of the problems is that the highly dynamic solar atmosphere makes it difficult to predict well in advance the occurrence and position of interesting features like sunspots, filaments or flares. Differential rotation, meridional circulation and surface motion change the position of solar features. For example, a feature at the solar disc center will move by almost 220 arcsec day^{-1}, assuming only Carrington rotation. While it is possible to include the mean rotation profile for the position prediction, proper motions relative to this mean profile are hard to predict. A feature with proper motion velocity of, e.g., 100 m s^{-1} will move another 11 arcsec per day, readily comparable with the field of view of ~26 arcsec of the 12 m ALMA antennas in Band 6. These facts make it very difficult to predict the position of a feature days in advance, so the target coordinates need to be specified as close to the actual observation as possible. On the other hand, scheduling constraints of (especially space-borne) solar observatories providing supporting data require specification of the target 1-2 days ahead. ALMA solar observations now require the PI to notify the Astronomer on Duty of the target position two days before the observation.

Another problem with solar observations is that solar physicists are accustomed to work in the heliocentric coordinate system, usually helioprojective or heliographic, while ALMA uses the celestial equatorial coordinate system. Therefore, it is necessary to have a tool for easy selection of solar features and simple generation of the corresponding ephemerides compatible with ALMA software (such as the ALMA Observing Tool, OT), taking into account differential rotation, coordinate systems and other effects. For this purpose, the ALMA Solar Ephemeris Generator has been developed.

A similar problem exists with the final FITS images provided to the PI by the project. These images are again in the equatorial coordinate system and most solar physicists prefer them to be converted into the heliocentric system. The python modules "helioimage2fits.py" and "equa2helio.py" have been developed exactly for this purpose. Also, an example script is given describing how to make a series of images from the ALMA data, which can be composed into a movie.

Finally, some solar physicists prefer radiation intensity to be expressed as the brightness temperature in units of K, rather than in Jy/beam. A short note on the conversion factor is given at the end.

2 COORDINATE CONVERSIONS FOR SOLAR OBSERVING

The problem of the conversion of coordinates can be put this way: given the equatorial coordinates of the target (specified as right ascension, α, and declination, δ, at a specific epoch), the equatorial coordinates of the solar disc center (α_0, δ_0) and the position angle of the solar north pole (P angle, measured from celestial North towards East), the aim is to find the helioprojective coordinates (θ_x, θ_y, sometimes also called solar x, y) of the target. Also, for the inverse problem, if (x, y) are given, to find the target (α, δ).

Approximate method

Since the Sun is only ~0.5 degrees in diameter when viewed from Earth, it is feasible to use the small angle approximation and planar geometry for targets near the Sun. In that case, the conversion between equatorial and helioprojective coordinates is just a rotation of the differences of equatorial coordinates of the Sun and the target by the negative solar *P* angle, corrected for the R.A. shrinkage with declination (cos δ_0 term):

$$\theta_x = -(\alpha - \alpha_0)\cos\delta_0 \cos P + (\delta - \delta_0)\sin P \tag{1}$$

$$\theta_y = (\alpha - \alpha_0)\cos\delta_0 \sin P + (\delta - \delta_0)\cos P \tag{2}$$

Helioprojective coordinates (θ_x, θ_y) are usually measured in arcseconds (positive towards solar west and north, hence the minus sign in the θ_x expression). Converting back to equatorial coordinates is straightforward:

$$\alpha = \frac{-\theta_x \cos P + \theta_y \sin P}{\cos\delta_0} + \alpha_0 \tag{3}$$

$$\delta = \theta_x \sin P + \theta_y \cos P + \delta_0 \tag{4}$$

This method works well for small angular distances from the solar center. Tests show that in extreme cases with extreme solar *P* angle values of ±26° and δ_0 values of ±23°, errors in both coordinates are expected to be less than 1 arcsec for targets on the solar limb. Errors due to the simplifications in (1) and (2) start to grow significantly at angular distances larger than 1 degree and at higher declinations. But, since this method is simple and has small errors for targets on the solar disk, it is frequently used (as in "helioimage2fits.py", see below). However, as new ALMA bands become available to the solar community, ALMA interferometric solar images will achieve resolutions well below 1 arcsec and then these small errors become unacceptable. In the next section, we develop an exact method which can be used in that case.

Exact method

By using spherical geometry it is possible to derive exact equations for the transformation (Figure 1):

$$\cos\rho = \cos\delta\cos\delta_0\cos(\alpha - \alpha_0) + \sin\delta\sin\delta_0 \tag{5}$$

$$\tan\varphi = \frac{\sin(\alpha - \alpha_0)}{\tan\delta\cos\delta_0 - \sin\delta_0\cos(\alpha - \alpha_0)} \tag{6}$$

$$\tan\theta_x = -\tan\rho\sin(\varphi - P) \tag{7}$$

$$\tan\theta_y = \tan\rho\cos(\varphi - P) \tag{8}$$

where ρ is the angular distance of the target from the solar center and φ is the position angle of the target measured from North towards East. The inverse equations are:

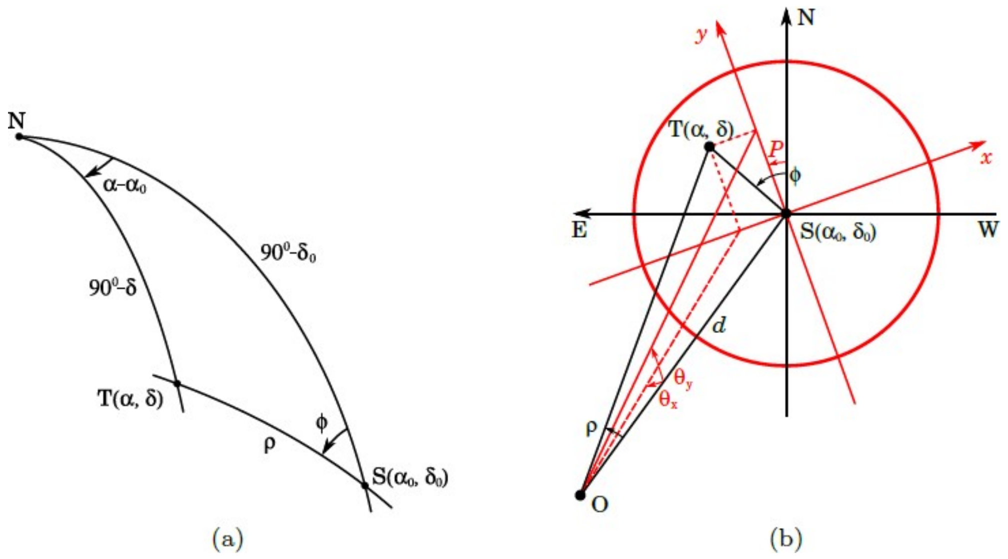


Figure 1: a) A spherical triangle on the celestial sphere defined by the Sun center (S), target location (T) and celestial north (N). b) Target (T) located on the solar disc (red circle) with helioprojective coordinate system (in red) and equatorial coordinate system (black). The observer (O) is located at a distance *d* from the Sun.

$$\tan(\varphi - P) = \frac{-\tan\theta_x}{\tan\theta_y} \tag{9}$$

$$\tan\rho = \frac{-\tan\theta_x}{\tan\theta_y} \tag{10}$$

$$\sin\delta = \sin\delta_0\cos\rho + \cos\delta_0\sin\rho\cos\varphi \tag{11}$$

$$\tan(\alpha-\alpha_0) = \frac{\sin\rho\sin\varphi}{\cos\rho\cos\delta_0 - \sin\rho\sin\delta_0\cos\varphi} \tag{12}$$

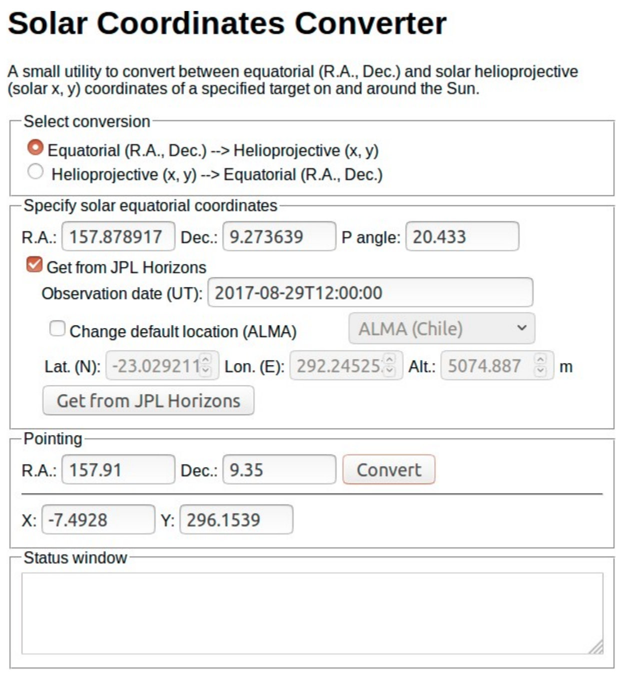
These equations are exact. However, computers have limited precision (rounding errors) and numerical problems can arise. The equation (5) for angular distance has problems for small angular distances because the cosine term changes slowly and is very close to unity. Probably the best way to avoid these problems is to use the equation for angular distance proposed by Vincenty (1975):

$$\tan\rho = \frac{\sqrt{[\cos\delta\sin(\alpha-\alpha_0)]^2 + [\cos\delta_0\sin\delta - \sin\delta_0\cos\delta\cos(\alpha-\alpha_0)]^2}}{\sin\delta\sin\delta_0 + \cos\delta\cos\delta_0\cos(\alpha-\alpha_0)} \tag{13}$$

Although more complicated, this expression should be precise and numerically stable for all angles and angular distances. That is the reason equation (13) is used in the ALMA Solar

Ephemeris Generator and the "equa2helio.py" script (see Appendix 3) for conversion of ALMA images to the helioprojective coordinate frame. Expressions for further conversion of the helioprojective coordinates to heliographic coordinates, if needed, can be found in, e.g., Thompson (2006) or in Appendix 2.

The Solar Coordinates Converter[1] (Figure 2) is a small javascript on-line utility for conversion between the equatorial and helioprojective coordinate systems that implements the exact method described above. It is a very useful tool to get the target coordinates either in equatorial or helioprojective coordinate system in a quick and easy way. Although it is a general tool, it was made with the intent for use with ALMA.



Solar Coordinates Converter

A small utility to convert between equatorial (R.A., Dec.) and solar helioprojective (solar x, y) coordinates of a specified target on and around the Sun.

Select conversion
[✓ Equatorial (R.A., Dec.) --> Helioprojective (x, y)]
Helioprojective (x, y) --> Equatorial (R.A., Dec.)

Specify solar equatorial coordinates
R.A.: 157.878917 Dec.: 9.273639 P angle: 20.433
[✓ Get from JPL Horizons]
Observation date (UT): 2017-08-29T12:00:00
Change default location (ALMA) ALMA (Chile)
Lat. (N): -23.029211 Lon. (E): 292.24525 Alt.: 5074.887 m
Get from JPL Horizons

Pointing
R.A.: 157.91 Dec.: 9.35 Convert
X: -7.4928 Y: 296.1539

Status window

Figure 2: User interface of the Solar Coordinates Converter

3 ALMA SOLAR EPHEMERIS GENERATOR

The ALMA Solar Ephemeris Generator[2] is a tool developed by the Czech ARC Node to be used together with the ALMA Observing Tool (OT) for preparation of solar observations. It was successfully tested and used during the December 2015 solar campaign and in regular ALMA solar observations since, by Principal Investigators (PIs) of the solar projects, ALMA contact scientists and astronomers on duty. It is a javascript-based application which runs in any modern browser and on many different operating systems, even mobile devices and tablets. The source and documentation can be found in SCIREQ-930 ticket and also on the hosting web site of the

[1]https://celestialscenes.com/alma/convert/
[2]https://celestialscenes.com/alma/coords/CoordTool.html

tool. The tool is based on the jsFITS javascript library[3] for FITS file manipulation with some background functions for accessing the data from external servers written in PHP.

The user interface consists of several panels (Figure 3). First, the user selects between the Graphical User Interface (GUI), where the target is specified by simple point-and-click method, and text interface, which enables the user to manually specify the target coordinates. In the input panel, it is possible to select one of the latest SDO/AIA images of the Sun in several wavelength channels, or the user can upload his own FITS file. Currently, only uncompressed FITS images with defined solar WCS keywords using the CROTA2 formalism are supported. Due to these constraints and to save bandwidth, AIA/SDO 1024x1024 pixel synoptic images (1/4 of the original resolution) are used, which are available at the Joint Science Operations Center (JSOC) web site[4]. Visualization and display panels enable the user to pan and zoom in/out of the region of interest or to show/hide coordinate grids and tweak the image display by false coloring and level scaling functions typical of other astronomical software packages and the ALMA Observing Tool. A useful feature is an overlay of the 12 m ALMA antenna field of view in the selected band. This gives the user a good perspective of the size of the selected feature and the coverage of the ALMA antennas.

The actual pointing in the GUI mode is selected by clicking on the desired feature, after which a green cross appears marking the chosen position. The coordinates of the pointing are displayed in several solar and image-based coordinate systems inside the pointing panel, where it is also possible to manually define the pointing. Here, it is also possible to specify the size and orientation of the field of view of the mosaic observation, if needed. A rough number of pointings with the 12 m ALMA antennas required for the specified mosaic size is automatically calculated (using the equation given in the ALMA Technical Handbook) and displayed in the GUI. The actual number of pointings used in the observation may differ slightly.

The “Location” panel enables the user to specify other observing sites besides the default (ALMA) one. This feature was implemented at the request of other observatories which may also use the tool for preparation of their solar observations. The user can select between several predefined observing sites or enter the geographic coordinates manually.

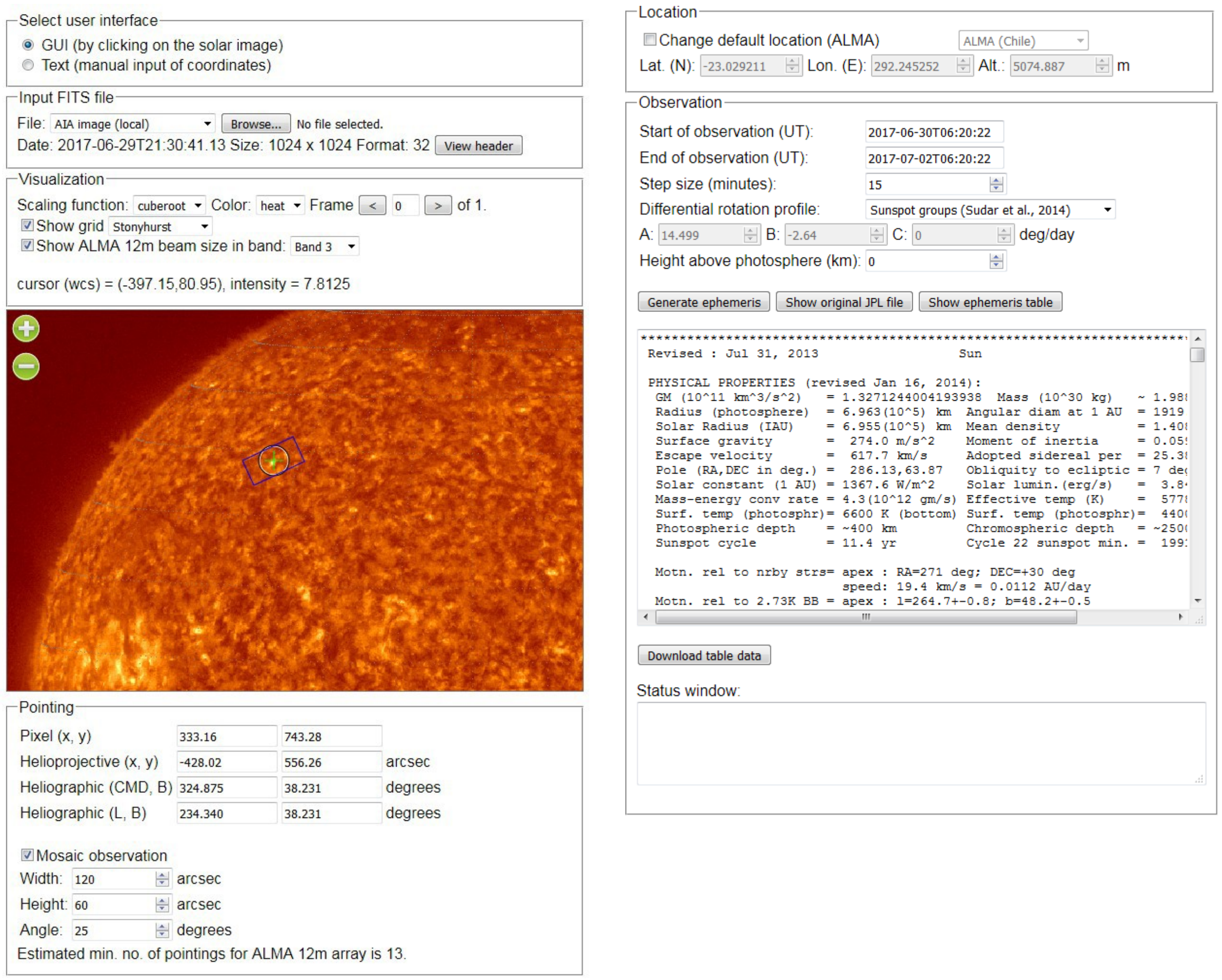
Finally, in the “Observation” panel, the user defines the start and end times of the observation and chooses a differential rotation profile which will be used for the generation of an OT-compatible ephemeris file. There are several rotation profiles to select from, or the user can define his/her own. Clicking the "Generate ephemeris file" will display the generated file which can be downloaded by clicking the "Download table data" button and then imported into the ALMA OT.

The ephemeris file is generated from a JPL Horizons file for the ephemeris of the center of the Sun which the tool queries and downloads directly from the JPL Horizons website[5]. However, during Cycle 4 regular observing sessions there was a period when the JPL Horizons site went offline and hence the Ephemeris Generator was unable to function properly. Work is currently

[3]https://github.com/slowe/jsFITS
[4]http://jsoc.stanford.edu/data/aia/synoptic/mostrecent/
[5]https://ssd.jpl.nasa.gov/horizons.cgi

underway to enable precise ephemeris calculations in the Ephemeris Generator even when the JPL site goes down.

The ALMA Solar Ephemeris Generator comes with a user manual and it is served on several sites for backup purposes. It is continuously improved and there are plans to move the development to a public server such as GitHub. Also, there are some discussions to include the tool functionality, at least partially, into the ALMA OT itself but a definite plan has not been set yet.


Select user interface
(● GUI (by clicking on the solar image)) ○ Text (manual input of coordinates)

Input FITS file
File: AIA image (local) Browse... No file selected.
Date: 2017-06-29T21:30:41.13 Size: 1024 x 1024 Format: 32 View header

Visualization
Scaling function: cuberoot Color: heat Frame 0 of 1.
☑ Show grid Stonyhurst
☑ Show ALMA 12m beam size in band: Band 3

cursor (wcs) = (-397.15,80.95), intensity = 7.8125

Pointing

Pixel (x, y)	333.16	743.28	
Helioprojective (x, y)	-428.02	556.26	arcsec
Heliographic (CMD, B)	324.875	38.231	degrees
Heliographic (L, B)	234.340	38.231	degrees

☑ Mosaic observation
Width: 120 arcsec
Height: 60 arcsec
Angle: 25 degrees
Estimated min. no. of pointings for ALMA 12m array is 13.

Location
☐ Change default location (ALMA) ALMA (Chile)
Lat. (N): -23.029211 Lon. (E): 292.245252 Alt.: 5074.887 m

Observation

Start of observation (UT):	2017-06-30T06:20:22
End of observation (UT):	2017-07-02T06:20:22
Step size (minutes):	15
Differential rotation profile:	Sunspot groups (Sudar et al., 2014)

A: 14.499 B: -2.64 C: 0 deg/day
Height above photosphere (km): 0

Generate ephemeris | Show original JPL file | Show ephemeris table

```
*********************************************************************
 Revised : Jul 31, 2013                  Sun

 PHYSICAL PROPERTIES (revised Jan 16, 2014):
  GM (10^11 km^3/s^2)   = 1.3271244004193938  Mass (10^30 kg)   ~ 1.98
  Radius (photosphere)  = 6.963(10^5) km  Angular diam at 1 AU  = 1919
  Solar Radius (IAU)    = 6.955(10^5) km  Mean density          = 1.40
  Surface gravity       =  274.0 m/s^2    Moment of inertia     = 0.05
  Escape velocity       =  617.7 km/s     Adopted sidereal per  = 25.3
  Pole (RA,DEC in deg.) =  286.13,63.87   Obliquity to ecliptic = 7 de
  Solar constant (1 AU) = 1367.6 W/m^2    Solar lumin.(erg/s)   =  3.8
  Mass-energy conv rate = 4.3(10^12 gm/s) Effective temp (K)    =  577
  Surf. temp (photosphr)= 6600 K (bottom) Surf. temp (photosphr)=  440
  Photospheric depth    = ~400 km         Chromospheric depth   = ~250
  Sunspot cycle         = 11.4 yr         Cycle 22 sunspot min. =  199

  Motn. rel to nrby strs= apex : RA=271 deg; DEC=+30 deg
                          speed: 19.4 km/s = 0.0112 AU/day
  Motn. rel to 2.73K BB = apex : l=264.7+-0.8; b=48.2+-0.5
```

Download table data

Status window:

Figure 3: Graphical User Interface of the ALMA Solar Ephemeris Generator. In this representation the interface is split up into two columns from the original one-column layout to fit onto the page.

4 COORDINATE SYSTEM SUPPORT FOR SOLAR IMAGES IN CASA

The Common Astronomy Software Applications (CASA) package[6] is the official software used for processing of ALMA data. While CASA has functions for defining different coordinate systems for the resulting images, there is no direct support for solar coordinate systems.

[6]https://casa.nrao.edu/

It was proposed that CASA should produce, for the solar case, standard FITS image files with coordinates familiar to the solar community and compatible with other tools (IDL/SolarSoft, Python/SunPy, etc.). The coordinate system proposed is the helioprojective Cartesian system with the gnomonic (TAN) projection. In this coordinate system, X and Y axes point to solar West and solar North, respectively (Figure 4). The units are described as angles:

$$\theta_x = x/d \cdot \pi/180 \cdot 3600,\ \theta_y = y/d \cdot \pi/180 \cdot 3600 \tag{14}$$

where d is the distance between the observer and the Sun center. FITS keywords should be specified so as to be compatible with the World Coordinate System (WCS; Thompson, 2006).

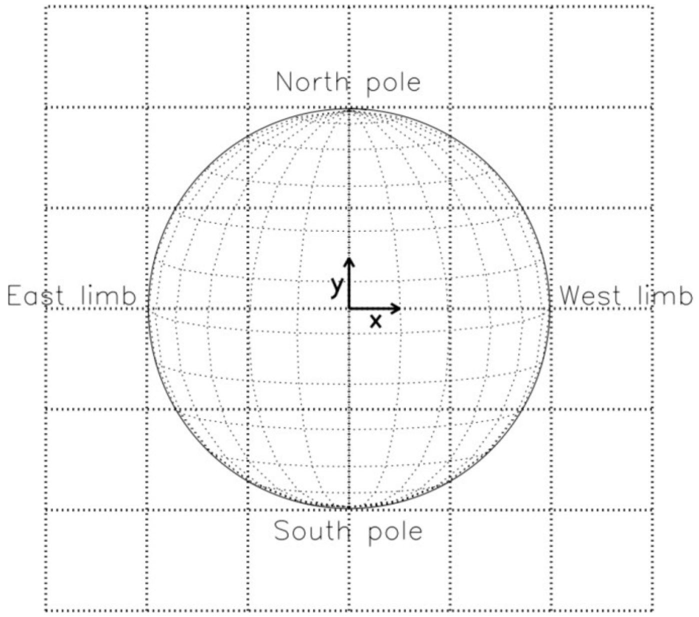


Figure 4: Heliographic (spherical grid) and helioprojective (rectangular grid) coordinate systems (from Thompson, 2006).

For transformation of the RA/DEC coordinates into the helioprojective coordinates (HPLN-TAN/HPLT-TAN) we need to calculate the image center with respect to the solar disc center, and rotate the image according to the solar *P* angle. Several other FITS keywords need to be modified and added in order for the FITS file to be compatible with solar mapping software.

For the purpose of converting ALMA data to the helioprojective coordinate system, the following information is needed:

- Reference time of observation
- Pointing coordinates at the reference time of observation in topocentric right ascension and declination (α and δ)
- Topocentric coordinates of the solar disc center (α_0, δ_0), solar *P* angle, and Sun-observer distance *d*, queried from JPL Horizons

A python module "helioimage2fits.py"[7] was developed by B. Chen and S. Yu that is able to convert VLA, EOVSA, and ALMA CASA images to FITS files with helioprojective coordinates. It does the following things:

1. Acquire the observatory and FIELD information from the measurement set visibility
2. Query JPL Horizons and obtain the solar disc center position in topocentric RA and DEC coordinates as well as the solar *P* angle as a function of time (for the duration of the measurement set; observatory aware)
3. Compare phase center and solar disc center (d_RA and d_DEC) according to the reference time of the image synthesis, and rotate according by the *P* angle to find d_solar_x and d_solar_y
4. Update the FITS header with helioprojective coordinates

It requires the following inputs:

- vis: the source ms visibility in order to derive the observatory and FIELD info
- timerange: time integration used for imaging
- imagefile: input CASA image

Specifically for ALMA, one also needs to set:

- reftime: the same time used in fixplanets

An example run in CASA:

```
> import helioimage2fits as hf
> hf.imreg(vis= your_visibility, timerange=your_timerange_for_clean,
imagefile= your_CASA_image, reftime= your_fixplanets_reftime, fitsfile =
name_of_your_output_fitsfile)
```

The output FITS files have been tested to be compatible with SSW/IDL's fits2map and sunpy.map routines. This routine has included things to be backward compatible with the previous VLA solar data in which each solar scan has a different FIELD ID, and for EOVSA which is always centered at the solar disc center.

While "helioimage2fits.py" module is a successful utility that can be used with data from other observatories besides ALMA, it needs visibility data (measurement sets) so it can deduce the needed (correct) values like DATE-OBS or EXPTIME. In order to decouple analysis of imaging data from the measurement sets, a much simpler Python/CASA script "equa2helio.py" was developed (listed in Appendix 3). It converts CASA exported FITS files from RA/Dec to helioprojective coordinates and doesn't need visibility data, but with some limitations since CASA will need to provide additional metadata in FITS files (e.g., to deduce the correct value of EXPTIME). The FITS keywords that are modified or added are listed in Table 1.

There are still some minor problems left regarding some FITS keywords. First, the DATE-OBS keyword that CASA produces defines the start of the the whole observation sequence (which begins with calibration scans) while in solar work the usual practice is to use it for start of the

[7]https://github.com/binchensun/suncasa/blob/master/utils/helioimage2fits.py

scientific observation of a solar target. “helioimage2fits.py” deals with this, but “equa2helio.py” doesn’t since it only has access to DATE-OBS already written by CASA.

The EXPTIME keyword is also problematic since CASA does not provide the duration of the integration directly. In addition, this keyword depends on the interval and the scans used in imaging of the data. It can be a source of misunderstanding when the integration time is larger than the duration of one scan, since it is unclear if it includes the duration of the phase/atmosphere calibrations and the overhead time for changing the direction of the antenna. A special subroutine needs to be developed to estimate EXPTIME but it would be helpful if CASA could have a method for this. "helioimage2fits.py" currently uses the time integration used for clean/tclean, but this may not be accurate enough. “equa2helio.py” doesn’t write EXPTIME keyword. To circumvent current problems with EXPTIME, OBS-END keyword is added that indicates the end time of the integration interval.

Table 1: List of FITS keywords that are modified and added to the original CASA ALMA FITS files during conversion to helioprojective coordinates.

FITS keyword	Value	Notes
CRVAL1	$\theta_x(t_{ref})$	helioprojective x (solar x) coordinate of the target in arcsec
CRVAL2	$\theta_y(t_{ref})$	helioprojective y (solar y) coordinate of the target in arcsec
CUNITx	‘arcsec ’	units used for CRVALx and CDELTx keywords
CDELT1	-cdelt1	just a negative value from the original image, converted to arcsec
CDELT2	cdelt2	original value converted to arcsec
CTYPE1	‘HPLN-TAN’	
CTYPE2	‘HPLT-TAN’	
DATE-OBS		CASA writes here start of the observation (it is not the start time of solar scans). Following solar convention, DATE-OBS should be the start time of the time integration in the image synthesis.
REF_TIME		Reference time used during calibration in fixplanets CASA task
DATE-END		end time of the integration
XCEN		helioprojective x coordinate of the image center
YCEN		helioprojective y coordinate of the image center
DSUN_OBS	d	Sun-observer distance in meters
P_ANGLE/SOLAR_P	P	
RSUN_OBS	$6.96e8$	Observed apparent radius of the Sun in arcsec
RSUN_REF	$R_\odot$	Reference solar radius used for calculation of RSUN_OBS, in meters
HGLN_OBS	0	Stonyhurst heliographic longitude of the observer, in degrees
HGLT_OBS	B_0	Stonyhurst heliographic latitude of the observer, in degrees
CRLN_OBS	L_0	Carrington heliographic longitude of the observer, in degrees
CRLT_OBS	B_0	Carrington heliographic latitude of the observer, in degrees

Finally, the RA and DEC values in the CASA exported FITS files may not always be correct, especially if the measurement set data contains ephemeris information and only includes one set of RA/DEC value for the target (or "FIELD"). To ensure a correct RA/DEC value in the FITS header, running "fixplanets" task with the actual RA/DEC of the phasecenter and reference time (before clean) is always recommended, and this is done in the currently used CASA scripts for calibration/imaging of the solar ALMA data. "helioimage2fits.py" does take into account and obtain the actual RA/DEC phase center by reading the information from the CASA measurement set while "equa2helio.py" uses RA/DEC values provided by the input FITS file.

Instead of using special scripts, CASA could add direct support for heliocentric coordinate systems. This is not necessary at this point since the scripts provided do the job just fine and are easier to modify and maintain. However, if an opportunity arises in the future, the following CASA tasks would need to be modified to provide such support:

- "**exportfits**": an option added to allow conversion to solar coordinates using the procedure as described above. Task parameters might be incorporated to query JPL Horizons for the solar disc center position or to import user-supplied target coordinates.
- "**viewer**" or "**imview**": add options to display solar FITS images in helioprojective Cartesian coordinates. This modification might be difficult. Moreover, the current CASA viewer will be replaced by the "CARTA" viewer in the near future. We recommend to forward this request to the CARTA development team.

In addition, a support for interpretation of time series datacubes to enable plotting (and overplotting) of light curves would be a useful feature. The ALMA archive currently does not support ALMA images with helioprojective coordinates as a QA2 product, because the registration program of the ALMA archive does not accept some "irregular" values in the FITS keywords (e.g., "HPLN_TAN"). This issue needs to be sorted with the ALMA Archive developers. In the meantime, a Python/CASA module or script, such as "helioimage2fits.py" or "equa2helio.py" could be provided to the solar PIs to do the conversion themselves.

5 ADDITIONAL UTILITIES FOR SOLAR PHYSICISTS

The final ALMA product that the PI gets is a calibrated FITS image of the target. However, the PI might want to make images of smaller time intervals to analyze the dynamics of the observed target. The CASA tasks clean/tclean do not support creation of movies as data cubes and it is necessary to export frames as individual images. In Appendix 4, a sample script prepared by M. Shimojo is provided that performs imaging/cleaning on every 20-second interval of the observation. As a result, you get a series of ~20-second integration images. However, movies produced in this way show variations of brightness intensities due to changes in amplitude and phase of the signal introduced by the Earth's atmosphere and other effects. There are ongoing efforts within the Solar Development Team and SSALMON[8] network (Wedemeyer et al. 2015) to further minimize the impact of these effects on the final image.

Finally, some solar physicists prefer using brightness temperature (expressed in K) instead of radiation intensity given in Jy/beam and usually used in the ALMA images. One good reason for this is that ALMA roughly operates as a linear thermometer of the solar chromosphere and brightness temperature can be almost directly mapped to the plasma temperature. For the conversion, we need the information about the beam shape which is defined by FITS keywords

[8]http://www.ssalmon.uio.no/

BMAJ(major axis), BMIN(minor axis) and BPA(position angle), specified in degrees. If intensity *I* is specified in W m^{-2} Hz^{-1} beam^{-1}, then its value in K will be (assuming a Gaussian beam shape):

$$T = \frac{4 \ln 2}{2 k_B} \frac{\lambda^2}{a b \pi} I \tag{15}$$

where k_B is Boltzmann constant (1.38×10^{-23} m^2 kg s^{-2} K^{-1}), λ is the wavelength in meters and *a, b* are major/minor axes of the beam in radians. Converting all constants to a pre-factor and changing units to Jy and arcseconds, we get:

$$T = 1.222 \times 10^6 \frac{I}{a\, b\, \nu^2} \tag{16}$$

where ν is the frequency in GHz. It is worthwhile to note that the conversion to brightness temperature has no meaning unless the total power (TP) data are included through feathering or some other means. The interferometric (INT) data only gives the variation in the brightness temperature relative to the large scale background. By combining TP and INT data, this background signal can be recovered and the data properly scaled.

B. Chen and S. Yu have written a preliminary CASA task, "ptclean"[9], which executes the CASA clean task in parallel using multiple CPUs (if available) at a series of time integrations, creating a sequence of CASA images, registering them in heliocentric coordinates and converting the unit from Jy/beam to brightness temperature in K. The final outputs of the task are a series of image FITS files. The syntax is similar to CASA clean tasks, but additionally one needs to provide the duration of the time integration of each image frame.

Acknowledgements

We express our thanks to Dragan Roša and Pascal Saint-Hilaire for their help with coordinate conversion. Special thanks goes to Andy Biggs, members of the ALMA Solar Development Team and the Czech ALMA Regional Center Node. Research leading to this work was performed within ESO Development Plan Study: Solar Research with ALMA (2014 - 2017). This work is supported by the SolarALMA project, which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 682462), and by the Research Council of Norway through its Centres of Excellence scheme, project number 262622.

6 REFERENCES

ALMA Technical Handbook, https://almascience.eso.org/documents-and-tools/latest/alma-technical-handbook

Bastian, T. S. et al. 2018, Exploring the Sun with ALMA, *The Messenger*, vol. 171, p. 25-30

Thompson, W. T. 2006, Coordinate systems for solar image data, *Astronomy and Astrophysics*, 449, 791

[9]https://github.com/binchensun/suncasa

Vincenty, T. 1975, Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations, *Survey Review*. XXIII (176): 88–93.
Wedemeyer, S., Bastian, T., Brajša, R. et al. 2015, Advances in Space Research, Vol. 56, Issue 12, p. 2679-2692.

APPENDIX 1

A comparison of the FITS file headers of the ALMA solar image before and after the conversion of the coordinate system from RA/Dec to helioprojective.

Original ALMA FITS header	ALMA FITS header modified for solar physicists
SIMPLE = T	SIMPLE = T
BITPIX = -32	BITPIX = -32
NAXIS = 4	NAXIS = 4
NAXIS1 = 512	NAXIS1 = 512
NAXIS2 = 512	NAXIS2 = 512
NAXIS3 = 1	NAXIS3 = 1
NAXIS4 = 1	NAXIS4 = 1
EXTEND = T	EXTEND = F
BSCALE = 1.00E+00	BSCALE = 1.00E+00
BZERO = 0.00E+00	BZERO = 0.00E+00
BMAJ = 1.29E-03	BMAJ = 1.29E-03
BMIN = 5.62E-04	BMIN = 5.62E-04
BPA = -7.76E+01	BPA = -7.76E+01
BTYPE = 'Intensity'	BTYPE = 'Intensity'
OBJECT = 'object'	OBJECT = 'object'
BUNIT = 'Jy/beam'	BUNIT = 'K'
EQUINOX = 2.00E+03	EQUINOX = 2.00E+03
RADESYS = 'FK5'	RADESYS 'FK5'
LONPOLE = 1.80E+02	LONPOLE = 1.80E+02
LATPOLE = -2.28E+01	LATPOLE = -2.28E+01
PC1_1 = 1.00E+00	PC1_1 = 1.00E+00
PC2_1 = 0.00E+00	PC2_1 = 0.00E+00
PC3_1 = 0.00E+00	PC3_1 = 0.00E+00
PC4_1 = 0.00E+00	PC4_1 = 0.00E+00
PC1_2 = 0.00E+00	PC1_2 = 0.00E+00
PC2_2 = 1.00E+00	PC2_2 = 1.00E+00
PC3_2 = 0.00E+00	PC3_2 = 0.00E+00
PC4_2 = 0.00E+00	PC4_2 = 0.00E+00
PC1_3 = 0.00E+00	PC1_3 = 0.00E+00
PC2_3 = 0.00E+00	PC2_3 = 0.00E+00
PC3_3 = 1.00E+00	PC3_3 = 1.00E+00
PC4_3 = 0.00E+00	PC4_3 = 0.00E+00
PC1_4 = 0.00E+00	PC1_4 = 0.00E+00
PC2_4 = 0.00E+00	PC2_4 = 0.00E+00
PC3_4 = 0.00E+00	PC3_4 = 0.00E+00
PC4_4 = 1.00E+00	PC4_4 = 1.00E+00
CTYPE1 = 'RA---SIN'	CTYPE1 = 'HPLN-TAN

CRVAL1	= 2.59E+02	CRVAL1	= 209.64875163918
CDELT1	= -8.33E-05	CDELT1	= 0.29999999999988
CRPIX1	= 2.57E+02	CRPIX1	= 2.57E+02
CUNIT1	= 'deg'	CUNIT1	= 'arcsec'
CTYPE2	= 'DEC--SIN'	CTYPE2	= 'HPLT-TAN'
CRVAL2	= -2.28E+01	CRVAL2	= 753.15624141307
CDELT2	= 8.33E-05	CDELT2	= 0.29999999999988
CRPIX2	= 2.57E+02	CRPIX2	= 2.57E+02
CUNIT2	= 'deg'	CUNIT2	= 'arcsec'
CTYPE3	= 'FREQ'	CTYPE3	= 'FREQ'
CRVAL3	= 1.00E+11	CRVAL3	= 1.00E+11
CDELT3	= 1.50E+10	CDELT3	= 1.50E+10
CRPIX3	= 1.00E+00	CRPIX3	= 1.00E+00
CUNIT3	= 'Hz'	CUNIT3	= 'Hz'
CTYPE4	= 'STOKES'	CTYPE4	= 'STOKES'
CRVAL4	= 1.00E+00	CRVAL4	= 1.00E+00
CDELT4	= 1.00E+00	CDELT4	= 1.00E+00
CRPIX4	= 1.00E+00	CRPIX4	= 1.00E+00
CUNIT4	= ''	CUNIT4	= ''
PV2_1	= 0.00E+00	PV2_1	= 0.00E+00
PV2_2	= 0.00E+00	PV2_2	= 0.00E+00
RESTFRQ	= 1.00E+11	RESTFRQ	= 1.00E+11
SPECSYS	= 'TOPOCENT'	SPECSYS	= 'TOPOCENT'
ALTRVAL	= -1.17E+04	ALTRVAL	= -1.17E+04
ALTRPIX	= 1.00E+00	ALTRPIX	= 1.00E+00
VELREF	= 259	VELREF	= 259
TELESCOP	= 'ALMA'	TELESCOP	= 'ALMA'
OBSERVER	= 'rsoto'	OBSERVER	= 'rsoto'
DATE-OBS	= '2014-12-11T18:44:39.072000'	DATE-OBS	= '2014-12-11T18:44:39.072000'
TIMESYS	= 'UTC'	TIMESYS	= 'UTC'
OBSRA	= 2.59E+02	OBSRA	= 2.59E+02
OBSDEC	= -2.28E+01	OBSDEC	= -2.28E+01
OBSGEO-X	= 2.23E+06	OBSGEO-X	= 2.23E+06
OBSGEO-Y	= -5.44E+06	OBSGEO-Y	= -5.44E+06
OBSGEO-Z	= -2.48E+06	OBSGEO-Z	= -2.48E+06
DATE	= '2016-12-26T04:05:44.741000'	DATE	= '2018-06-01T16:59:27.471000'
ORIGIN	= 'CASA 5.1.2-4'	ORIGIN	= 'CASA 5.1.2-4'
		DSUN_OBS	= 147296035413.001
		RSUN_REF	= 696000000.0
		RSUN_OBS	= 974.61
		HGLN_OBS	= 0.0
		HGLT_OBS	= -0.47
		CRLN_OBS	= 317.97
		CRLT_OBS	= -0.47
		SOLAR_P	= 11.6872
		XCEN	= 451.189110130716
		YCEN	= -0.397966279232493
		WCSNAME	= 'Helioprojective-cartesian'
		REF_TIME	= '2014-12-11T19:09:13'

APPENDIX 2

Conversion between various solar-based coordinate systems (adapted from Thompson, 2006).

Helioprojective-cartesian (θ_x, θ_y) to heliocentric-cartesian (x, y, z):

$$q = D_\odot \cos\vartheta_x \cos\vartheta_y$$
$$d = q - \sqrt{q^2 - D_\odot^2 + r^2}$$
$$x = d\cos\vartheta_y \sin\vartheta_x$$
$$y = d\sin\vartheta_y$$
$$z = D_\odot - d\cos\vartheta_y \cos\vartheta_x$$

where $D_\odot$ is the Sun-observer distance and r is the distance of the radial distance from Sun center (usually solar radius, $R_\odot$).

Heliocentric-cartesian (x, y, z) to helioprojective-cartesian (θ_x, θ_y):

$$\tan\vartheta_x = \frac{x}{D_\odot - z}$$
$$\sin\vartheta_y = \frac{y}{\sqrt{x^2 + y^2 + (D_\odot - z)^2}}$$

Stonyhurst heliographic (r, Φ, Θ) to heliocentric-cartesian (x, y, z):

$$x = r\cos\Theta\sin(\Phi - \Phi_0)$$
$$y = r(\sin\Theta\cos B_0 - \cos\Theta\cos(\Phi - \Phi_0)\sin B_0)$$
$$z = r(\sin\Theta\sin B_0 + \cos\Theta\cos(\Phi - \Phi_0)\cos B_0)$$

where B_0 and Φ_0 are the Stonyhurst heliographic latitude and longitude of the observer.

Heliocentric-cartesian (x, y, z) to Stonyhurst heliographic (r, Θ, Φ):

$$r = \sqrt{x^2 + y^2 + z^2}$$
$$\sin\Theta = \frac{y\cos B_0 + z\sin B_0}{r}$$
$$\tan(\Phi - L_0) = \frac{x}{z\cos B_0 - y\sin B_0}$$

Stonyhurst heliographic (r, Φ, Θ) to Carrington heliographic (r, Φ_C, Θ):

$$\Phi_C = \Phi + L_0$$

where L_0 is the Carrington longitude of the central meridian as seen from Earth.

APPENDIX 3

Python/CASA routines for conversion of the final ALMA solar image in FITS format from equatorial into helioprojective coordinate system.

```
import urllib
import re
import pyfits
import ssl
import sys

from taskinit import qa
#import numpy as np

# angular distance by Vincenty equation (in radians)
def angdist(ra1,de1,ra2,de2):
    num1 = cos(de2) * sin(ra2 - ra1)
    num2 = cos(de1) * sin(de2) - sin(de1) * cos(de2) * cos(ra2 - ra1)
    denominator = sin(de1) * sin(de2) + cos(de1) * cos(de2) * cos(ra2 - ra1)
    return atan2(hypot(num1, num2), denominator)

# convert ra,dec to helioprojective (all values in radians)
def radec2hpc(ra, de, sun_ra, sun_de, sun_P):
    rho = angdist(ra, de, sun_ra, sun_de)
    theta = atan2(sin(ra - sun_ra), tan(de) * cos(sun_de) - sin(sun_de) * cos(ra - sun_ra))
    hpc_x = atan(-tan(rho) * sin(theta - sun_P))
    hpc_y = atan(tan(rho) * cos(theta - sun_P))
    return rho, hpc_x, hpc_y


# updates fits header keys, inserts new if the key doesn't exist
def update_hdr(hdr, keyword, value):
    #pyfits changed header set/update methods in v3.1
    if hasattr(pyfits.Header, 'set') and callable(getattr(pyfits.Header, 'set')):
        hdr.set(keyword, value)
    else:
        hdr.update(keyword, value)


# retrieve from JPL Horizons coordinates of the Sun, distance and orientation angles [in AU and
radians]
def getJPLSunEphemForALMA(reftime='', verbose=False):
    if reftime=='': start_time = qa.quantity("now")
    else: start_time = qa.quantity(reftime)
    end_time = qa.add(start_time, "1min")
    querystring = "https://ssd.jpl.nasa.gov/horizons_batch.cgi?batch=1&COMMAND=%2710%27&CENTER=
%27-7@399%27&MAKE_EPHEM=%27YES%27&TABLE_TYPE=%27OBSERVER%27&START_TIME=%27"
    querystring = querystring + str(qa.time(start_time, form="fits")[0]) + "&STOP_TIME=%27" +
str(qa.time(end_time, form="fits")[0]) + "%27&STEP_SIZE=%272%20m%27&QUANTITIES=
%271,14,17,20%27&CSV_FORMAT=%27YES%27"
    # In CASA 5 which comes with new python version, urllib.urlopen requests ssl certificates,
this disables checking (ssl._create_unverified_context() introduced in python v2.7.9)
    try:
        context = ssl._create_unverified_context()
        response = urllib.urlopen(querystring, context=context)
    except:
        response = urllib.urlopen(querystring)
    the_page = response.read()
    result = re.search('\$\$SOE([\S\s]*?)\$\$EOE', the_page)
    _, _, _, ra_str, de_str, L0, B0, P, rapp, dsun, rrate, _ =result.group(1).split(',')
    rah, ram, ras = ra_str.split(' ')
    ded, dem, des = de_str.split(' ')
    ra = 15*(float(rah) + float(ram)/60.0 + float(ras)/3600.0)
    if float(ded)<0.0: m=-1
    else: m=1
    de = m*(abs(float(ded)) + float(dem)/60.0 +float(des)/3600.0)
    if verbose==True:
        print 'JPL Horizons - Solar Ephemeris'
        print 'Date: ', qa.time(start_time, form="fits")[0]
        print 'Location: ALMA'
        print 'RA/DEC Astrometric J2000.0: ', ra_str, de_str
        print 'Range, range rate [AU, AU day^-1]: ', dsun, rrate
        print 'Apparent radius [arcsec]: ', rapp
        print 'P, B0, L0 [deg]: ', P, B0, L0
    sun_eph = {'t': qa.time(start_time, form="fits")[0], 'ra': radians(ra), 'dec': radians(de),
'dsun': float(dsun),
    'rrate': float(rrate), 'rapp': radians(float(rapp)/3600.0),
    'P':radians(float(P)), 'B0':radians(float(B0)),'L0':radians(float(L0))}
    return sun_eph


#convert Jy/beam to K
def jy2K(in_image, out_image, dc_offset=None, verbose=False):
    if verbose: print 'Converting Jy/beam to K...'
    bunit = imhead(in_image, mode='get', hdkey='bunit')
    if bunit=='K':
        if verbose: print 'Image data already in Kelvin.'
        return

    bmaj = imhead(in_image, mode='get', hdkey='bmaj')
    bmin = imhead(in_image, mode='get', hdkey='bmin')
    restfreq = imhead(in_image, mode='get', hdkey='restfreq')
    a = bmaj.values()[0]
    b = bmin.values()[0]
    freq = restfreq.values()[0]/1.e9
    if verbose:
        print "Frequency: ",freq, " GHz"
        print "Beam: ",a," x ",b, "arcsec"

    convJyb2K = str(1.222e6/(a*b*freq**2))
    expr = 'IM0*' + convJyb2K

    os.system('rm -rf '+out_image)
    immath(imagename=in_image, expr=expr, outfile=out_image)
    ia.close()



def radec2helio(infits, outfits, reftime='', toKelvin=True, verbose=False):
    '''
    Converts FITS file with equatorial RA/Dec WCS to helioprojective WCS, rotating the image to
solar North.
    Function assumes that equatorial image was oriented North up, East left in J2000.0 frame.
    Inputs:
        infits - input FITS file with equatorial WCS
        outfits - output FITS file to be written with helioprojective WCS
        reftime - reference time used in fixplanets routine
        toKelvin - convert Jy/beam to K
        verbose - print additional info on screen
    '''
    # tmp files
    img = 'tmp_alma2solar.im'
    imgtmp = 'tmp2_alma2solar.im'

    # import fits file
    importfits(infits, img)

    try:
        # get reftime
        if reftime == '':
            print "REF_TIME not specified. Using DATE-OBS from input FITS as REF_TIME..."
            reftime = imhead(img, mode='get',hdkey='DATE-OBS')

        #convert Jy/beam to Kelvin if requested
        if toKelvin:
            os.system('cp -r '+img+'/ '+imgtmp+'/')
            jy2K(imgtmp, img, dc_offset=7300, verbose=verbose)
            os.system('rm -rf '+imgtmp)

        # get solar epphem from JPL Horizons
        if verbose: print "Retreiving solar data from JPL Horizons..."
        sun_ephem = getJPLSunEphemForALMA(reftime, verbose)

        if verbose: print "Rotating image by -solar_P angle..."
        # rotate image by -solar P angle to get solar north up
        ia.open(img)
        imr=ia.rotate(outfile=imgtmp, pa=str(-degrees(sun_ephem['P']))+'deg')
        imr.done()
        ia.close()

        # get sky coords and calc hpc coords
        hdu = pyfits.open(infits)
        hdr = hdu[0].header
        naxis1 = hdr['NAXIS1']
        naxis2 = hdr['NAXIS2']
        crval1 = qa.convert(qa.quantity(hdr['CRVAL1'], hdr['CUNIT1']), 'rad')['value']
        crval2 = qa.convert(qa.quantity(hdr['CRVAL2'], hdr['CUNIT2']), 'rad')['value']
        cdelt1 = qa.convert(qa.quantity(hdr['CDELT1'], hdr['CUNIT1']), 'rad')['value']
        cdelt2 = qa.convert(qa.quantity(hdr['CDELT2'], hdr['CUNIT2']), 'rad')['value']
        crpix1 = hdr['CRPIX1']
        crpix2 = hdr['CRPIX2']
        hdu.close()

        rho, crval1, crval2 = radec2hpc(crval1, crval2, sun_ra=sun_ephem['ra'],
sun_de=sun_ephem['dec'], sun_P=sun_ephem['P'])

        cdelt1 = -cdelt1 #correct for different direction of RA and HGLN axes
        hpc_x = degrees(crval1)*3600.0
        hpc_y = degrees(crval2)*3600.0
        crota2 = 0.0 #since we already rotated the image
        #xcen = 3600.0*degrees(crval1 + cdelt1*cos(crota2)*((naxis1+1)/2.0 - crpix1) -
cdelt2*sin(crota2)*((naxis2+1)/2.0 - crpix2))
        #ycen = 3600.0*degrees(crval2 + cdelt1*sin(crota2)*((naxis1+1)/2.0 - crpix1) +
cdelt2*cos(crota2)*((naxis2+1)/2.0 - crpix2))
        xcen = hpc_x
        ycen = hpc_y

        if verbose: print 'New reference pixel coordinates (helioprojective-x,y, arcsec):',
hpc_x, hpc_y

        # delete if file already exists
        os.system('rm -rf '+outfits)

        # save to fits
        if verbose: print "Exporting FITS..."
        exportfits(imagename=imgtmp, fitsimage=outfits)

    finally:
        #  clean-up
        os.system('rm -rf '+img)
        os.system('rm -rf '+imgtmp)


    # pyfits workaround for solar coordsys type (needs pyFits installed in CASA)
    if verbose: print 'Updating FITS keywords...'

    JPL_AU = 149597870700.0 #m
    JPL_RSUN = 696000000.0 #m

    hdu = pyfits.open(outfits, mode='update')
    hdr = hdu[0].header
    update_hdr(hdr, 'CRVAL1', hpc_x)
    update_hdr(hdr, 'CRVAL2', hpc_y)
    update_hdr(hdr, 'CUNIT1', 'arcsec')
    update_hdr(hdr, 'CUNIT2', 'arcsec')
    update_hdr(hdr, 'CDELT1', degrees(cdelt1)*3600.0)
    update_hdr(hdr, 'CDELT2', degrees(cdelt2)*3600.0)
    update_hdr(hdr, 'CTYPE1', 'HPLN-TAN')
    update_hdr(hdr, 'CTYPE2', 'HPLT-TAN')
    update_hdr(hdr, 'DSUN_REF', JPL_AU)
    update_hdr(hdr, 'DSUN_OBS', sun_ephem['dsun']*JPL_AU)
    update_hdr(hdr, 'RSUN_REF', JPL_RSUN)
    update_hdr(hdr, 'RSUN_OBS', abs(degrees(sun_ephem['rapp']))*3600.0)
    update_hdr(hdr, 'HGLN_OBS', 0.0)
    update_hdr(hdr, 'HGLT_OBS', degrees(sun_ephem['B0']))
    update_hdr(hdr, 'CRLN_OBS', degrees(sun_ephem['L0']))
    update_hdr(hdr, 'CRLT_OBS', degrees(sun_ephem['B0']))
    update_hdr(hdr, 'SOLAR_P', degrees(sun_ephem['P']))
    update_hdr(hdr, 'XCEN', xcen)
    update_hdr(hdr, 'YCEN', ycen)
    update_hdr(hdr, 'WCSNAME', 'Helioprojective-cartesian')
    update_hdr(hdr, 'PC1_1', cos(crota2))
    update_hdr(hdr, 'PC1_2', -sin(crota2)*cdelt2/cdelt1)
    update_hdr(hdr, 'PC2_1', sin(crota2)*cdelt1/cdelt2)
    update_hdr(hdr, 'PC2_2', cos(crota2))
    update_hdr(hdr, 'REF_TIME', qa.time(reftime, form='fits')[0])
    if toKelvin:
        update_hdr(hdr, 'BUNIT', 'K')
    #del hdr['RADESYS']
    hdu.flush()
    hdu.close()

    if verbose: print "radec2helio done."


#test on CSV data
radec2helio('QuietSun_Band3_allspw_I.fits', 'QuietSun_Band3_allspw_I_rot.fits',
reftime='2014/12/11/19:09:13.06', toKelvin=True, verbose=True)
```

APPENDIX 4

An example script for making a set of images from the ALMA observation data, one image every 20 seconds of the whole observing interval.

```
##########################
# Script for synthesizing images for a movie.
##########################
import numpy as np

#######
#ASDM Name
#######
asdm = 'uid___A002_Xbf9c0a_Xb0ab'
msc = asdm + '.ms.split.cal'

###############
## Integration time
################
IMinterval = 20.

tgt_scans = [11,14,17,20,23,26,29,32]
ref_tim = '2017-04-26T14:35:35'

################
##Select ScanID from above the list
################

for tgt_sc in range(0,8):

################
# Image Name
###############
    imgname_top = '00070.S_1st_20s_1sc'

#################
# Get time information and fix the number of created images
#################

    tb.open(msc)
    raw_data =tb.query('SCAN_NUMBER == %d && ANTENNA1 == 0 && ANTENNA2 == 2 && DATA_DESC_ID ==1'
% tgt_scans[tgt_sc])
    tt_data = raw_data.getcol('TIME')
    tb.close

    img_num = int(floor((tt_data[tt_data.shape[0]-3] - tt_data[2])/IMinterval))
    st_ind = np.array(range(0, img_num))*IMinterval/2.+2
    ed_ind = st_ind + IMinterval/2.-1

#################
# Image Synthesis
#################

    for i in range(0, img_num):
        st_tim = qa.time(qa.quantity(tt_data[st_ind[i]], 's'), form="ymd",prec=8)[0]
        ed_tim = qa.time(qa.quantity(tt_data[ed_ind[i]]+2, 's'), form="ymd",prec=8)[0]
        timerange = '%s~%s' % (st_tim, ed_tim)
        imagename = imgname_top + '_' + str(tgt_scans[tgt_sc]) + '_%03d' % i

        print ('Now processing:%03d/%03d ') % (i, img_num) + ' : ' + st_tim +' :'+imagename

        tclean(vis = msc,
            spw = '0,1,2,3',
            stokes = 'I',
            field='0',
            timerange = timerange,
            phasecenter=0,
            imagename=imagename,
            cell = '0.3 arcsec',
            imsize = [512, 512],
            interactive = False,
            usemask = 'pb',
            pbmask = 0.5,
            pblimit = 0.5,
            specmode = 'mfs',
            gridder = 'mosaicft',
            weighting = 'briggs',
            robust = 1.,
            niter = 500000000,
            deconvolver = 'multiscale',
            scales= [0,16,42,64],
            threshold='1.5Jy',
            gain=0.1)

    #Modify the value of the date_obs keyword
        imhead(imagename+'.image', mode='put', hdkey='date-obs', hdvalue=st_tim)
        imhead(imagename+'.image', mode='add', hdkey='date-end', hdvalue=ed_tim)
        imhead(imagename+'.image', mode='add', hdkey='ref_time', hdvalue=ref_tim)


    #Primary Beam Correction
        impbcor(imagename=imagename + '.image',
                pbimage=imagename+'.pb',
                outfile=imagename+'.pbcor', mode='divide')

    #make FITS files
        exportfits(imagename= imagename + '.image',
                fitsimage= imagename + '.fits')
```