Observation Scheduler for SAGE Sky Survey

Jie Zheng, 2016-07, Tucson (jiezheng(a)nao.cas.cn)

Files and directories structure

Basic File Rules:

All angle in degrees, except ra/dec in observation list, it depends on telescope, usually in sexagesimal hms/dms mode.

RunCode rule: in most case, use the month of the run starts, format yyyymm. If there are more runs in same month, add postfix A, B, or C.

Date rule: use extra modified Julian day, emjd = jd - 2450000.5, for observing, use the jd of local 18:00. In calling a program, use year, month, and day, but the day can be extended, not only 1 to 31.

Time rule: in most case, use hhmmss format time. In some case, will use decimal format, use `tttt` or `ttttt` to present.

Column seperator: if not specified, use space to seperate, First column left aligned, others right aligned for numerical and left or right for string.

- `/`
  - `programs`
- `/tel/` telescope name brief, currently we have `bok` and `xao`
- `/tel/conf/`
  - `basic.txt` site and telescope basic data, longitude, latitude, altitude, timezone, exposure accessory time, field of view.
  - `field.txt` list of all fields, col: field, ra, dec, gl, gb, block. Note: field id is **NOT** continual int, some will be skipped for future use. Block name consists of 7 digits, 3 for dec and 4 for ra., 10 times of mean ra/dec of the block.
  - `expplan.txt` plan of exposures, col: code name filter expt repeat factor active dither1 dither2
  - `expmode.txt` mode of exposures, col: filter expt code factor, use filter+expt to locate code and factor sum factor to check finished or not. Mode is a connection between real exposure and exposure plan.

- `/tel/obsed/yyyymm/` usually runcode is yyyymm, but can be others
    - `files.Jdddd.lst` daily file list, only good files, col: full filename
    - `check.Jdddd.lst` daily check list, col: filesn, imagetype, field/object, filter, exptime, ra, dec, filename
    - `obsed.Jdddd.lst` finished list of the day, col: field factorlist (by code)
- `/tel/schedule/runcode/Jjjjj/`
    - `plan.jjjj.yyyy.mm.dd.txt` merged plan of this day
    - `report.jjjj.yyyy.mm.dd.txt` report of generate script for this night. Including input parameters, moon info, and obs time. A list for blocks and obs time is also included. All warnings are listed.
    - `summary.jjjj.yyyy.mm.dd.txt` list only selected blocks. Can be read by code, for analysis use.
    - `scr.jjjj.nn.bbbbbbb.txt` plan of one block, nn means serial number of this night, bbbbbbb means block name.
    - `chk.jjjj.nn.bbbbbbb.txt` check file for choose a block from available blocks. Including candidates' name, ra, dec, airmass, rank.
    - `see.jjjj.nn.bbbbbbb.txt` sky map for each block chosen, besides footprint of past finished, it tells the zenith, sun pos, moon pos, candidates and chosen block.
    - `plan.jjjj.yyyy.mm.dd.eps` footprint of finished and planed fields.
    - `plan.jjjj.yyyy.mm.dd.png` png version of plan

Programs

list

Use `ls` to generage a file list: `/tel/obsed/runcode/files.xxx.lst`.

Not a python program but a shell operation

check

Check file header info, from `files.xxx.lst` to `/tel/obsed/runcode/check.xxx.lst`, and `tel/obsed/runcode/obsed.xxx.lst`.

param

- `tel` : string, telescope brief
- `run` : string, yyyymm
- `day` : 4-digit mjd of the night

collect

Collect file info from daily check list and generate daily obsed list.

param

- `tel` : string, telescope brief
- `run` : string, yyyymm
- `day` : 4-digit mjd of the night

Divide check and collect into 2 steps, so we can `check` on server and then `collect` in any machine without fits. If we altered plan or mode, we do not need to scan fits again.

headerinfo

Extract info from fits file header, use this to adapt to different telescope.

param

- `filename` : full filename
- `return` : instance of file header info: filename, filesn, image/object type, object, filter, exposure time, ra (deg), dec (deg)

footprint

Plot footprint map from obsed data, on a mollweide projected sky. It will generate a text report and two figure, in Equatorial and Galactic system. If specified run or day, then fields in the run or day will be red. If `before` set true, will obly draw fields before it.

param

- `tel` : telescope, must provide
- `reportfile` : text report filename. If not given, use current date and time. If given empty string, will not generate this report. Same rule for `equfile` and `galfile` .
- `equfile` : filename of footprint in Equatorial system
- `galfile` : filename of footprint in Galactic system
- `run` : optional, if specified, only draw this run, else draw all
- `day` : optional, if provided, only draw this day. If without runcode, this is ommitted
- `plan` : specified which plan to be drawn, if not, all plan
- `before` : optional, if set True, will only draw fields before specified day or run

takeoff

Make schedule.

param

- `tel` : telescope code
- `yr` : year of obs date, 4-digit year
- `mn` : year of obs date, 1 to 12
- `dy` : day of obs date, 0 to 31, or extended
- `run` : run code, default is `yyyymm`
- `obs_begin` : obs begin hour, float, default 1.25 hours after sunset
- `obs_end` , obs end hour, float, default, 1.25 hours before sunrise
- `moon_dis_limit` : limit of distance of good field to the moon, default 50 deg
- `airmass_limit` : limit of airmass, default 1.75, but for pole area, this should be greater
- `ha_limit` : limit of hour angle, default 3.0, should be greater for pole area
- `overwrite` : bool, when output dir already exists, overwrite or not
- `simulate` : bool, generate a obsed list or not

util

Utilities for common usage.

function

- dms2dec
- hmd2dec
- dec2hms
- dec2dms
- hour2str
- sxpar
- progress_bar
- read_conf
- msgbox

schdutil

Utilities for scheduler.

class

- baseinfo base class for xxxinfo classes
- plan_info
- mode_info
- check_info
- field_info
- block_info

- exposure_info

function

- load_basic
- load_expplan
- load_expmode
- load_field
- load_obsed
- ls_files
- dayofyear
- fmst
- mjd
- night_len
- night_time
- airmass

Data Structures

Site Info

An object returned from `load_basic`, direct instance of `base_info`. (Most objects are instances of derived classes.)

Read from `tel/conf/basic.txt`.

Property

- `lon` longitude of site, in degrees
- `lat` latitude of site, in degrees
- `alt` altitude of site, in meters from sea level
- `tz` timezone of site, -12 to +12
- `lons` longitude string, sexagesimal format
- `lats` latitude string, sexagesimal format
- `inter` average time between exposures, including readout and pointing, and even shared equally focus operation time. In seconds.
- `fov` field-of-view of telescope, in degrees
- `fmt` format of script lines, following info fields available:
  - obj: object name
  - filter: filter name
  - expti: int exposure time
  - exptf: float exposure time

- rad, ded: decimal degree of ra/dec
- ras, des: sexagesimal of ra/dec, use colon as delimiter
- rap, dep: packed sexagesimal of ra/dec, no delimiter

Exposure Plan

A dict returned from `load_expplan`, key is plan code, and value is instance of `plan_info`.

Property

- `code` plan code, int from 0 (in order to cooperate with other languages)
- `name` plan name, a short string to identify the plan
- `filter` filter name, must be same with name on telescope
- `expt` exposure time, in seconds, float or int
- `repeat` repeat times of this exposure, not used infact
- `factor` factor gained for each exposure
- `active` 0 or 1 to show the plan is active or not
- `dither1` ra dither from field center, in degrees
- `dither2` dec dither

About `repeat` and `factor`: this method of control comes from splitting long exposure. if we need a long exposure, we can split into more shorter ones. Each finish part of original plan. Once a field accumulate factor to 1.0 or more, means it is completed.

But in most case, we use short exposures, so repeat will be 1 and factor be 1.0.

Dither 1 and 2, is used when we have more exposures for one filter, we can set some exposures dithered a little.

For `active`, maybe we will have future plan, so we can write it in plan file ahead. Or, we will do different filter set in different nights, we can disable the plans we do not need temporarily.

Exposure Mode

A dict returned from `load_expmode`, key is mode code, and value is instance of `mode_info`.

Property

- `filter` filter name of exposure
- `expt` exposure time, in seconds
- `code` corresponding plan code, int
- `factor` factor for this mode

Method

- `mode` mode code, consists of filter and exposure time. This is unique between modes.

Exposure modes list all recognizable exposure mode, it is a connection between real exposure and plan. In some cases, one plan may have different practical exposure setting, for example, 300s of u, can be one u300, or two u150, another examples is 60s of strumu, we did 40s and 50s version, they are all acceptable.

File Info Check

An object to hold file info, returned from module `header_info`, and used by `check` and `collect`. `header_info` get file info from fits, this module is used to modulate difference of different telescope. `check` list the info into check file. And `collect` use check file, and cross with plan through mode.

Property

- `fitsfile` original fits filename
- `filesn` file serial number in the night, from filename
- `imgtyp` image type
- `object` object name
- `filter` filter name
- `expt` exposure time, in seconds
- `ra` ra, in degrees
- `de` dec, in degrees

Method

- `mode` get mode for the file
- `simulate` simulate a file from plan and field object

Field Info

A dict returned from `load_field`, key is field id, int, and value is instance of `field_info`. Field info will be updated by `load_obsed`, it loads obsed factors from historial obsed file, and accumulated by fields. Finally it set tag of each field.

Property

- `id` id of field, int, but not continuas because some fields are abondoned
- `ra`, `de` ra and dec of field center, in degrees
- `gl`, `gb` gl and gb, in degrees, low precious, just for check and plotting
- `bk` block name, set by module `makeblock`
- `factor` a dict of factor, from plancode to factor, all plans are here, despite of active or not
- `mark` another dict of factor, but only include factors come from marked files, for example, from a specified run

- `tag` most important property. Set according to factor, mark and scheduler. The following status depends only on active plan.
    - 0: total unobserved. And in scheduler, this is not planned in this night
    - 1: partly observed. Not all plans finished, but something has been done. Also, in scheduler, this is not used.
    - 2: finished. All factors are greater than or equal to 1.0.
    - 3: marked or planned. In footprint module, this tag means it is obsed in specified nights. In scheduler, it means chosen to observed in this night.
    - 4, 5: obs candidate. In selection the best block, we did an `bit-or` operation on original tag 0 and 1, so after plotting check figure, we can use `bit-and` to set it back to 0 and 1. This is a temperary status for plotting.
    - 7: curently selected obs field. This is also a temperary status, and after plotting, it will be 3.
    - 16, 17, 18: near moon or sun. These come from 0, 1, and 2, add 0x10 because it is too close to sun and moon. They will not be used in scheduler.

Tag Transfer

- 0: Original
- 1: after partly obsed, 0/1 --> 1, this is NOT an inevitable status.
- 2: after full obsed, 0/1 --> 2, final status of a field
- 3: chosen by scheduler, or marked. In scheduler, 0/1 be candidate --> 4/5 chosen --> 7 after potting --> 3.

Related ndarray

- afields: ndarray of field objects
- ara, ade, atag: extract ra, dec, and tag from fields dict/array
- newfield: ndarray of new fields (observable, not finished and not near to sun or moon)

Block Info

A dict of blocks, key is block name, and value is instance of `block_info`. Block is only used in scheduler, we set tasks by block , but not by field. Block array is extracted after new fields selected.

Property

- `bname`: block name
- `ra`, `de`: mean value of ra/dec of fields in this block
- `fields`: an ndarray holding fields in this block, fields sorted by ra

Related ndarray

- newblock: this is the block dict discussed above. Once a block is used, it will be removed.

- bra, bde: block ra/dec, extracted from newblock
- ha, airm: hour angle and airmass of each block at assumed time

Exposure Info

A class hold newly planned exposure. It generated by a plan and a field. The reason to have such a class, is to provide different format of output, using direct property and function property.

All available properties is discussed in site info.

Algorithm for scheduler

General structure of `takeoff`

1. Load configure file.
2. Calculate observation parameters for this night, mjd, lst, sun set and rise time, and etc.
3. Make day directory, and check priviledges.
4. Load fields and obsed info, set tags for fields. Remove fields near the sun or moon.
5. Extract blocks from fields, make a list of available block.
6. Set clock to obs start time.
7. If clock goes beyond the obs end time, finish work.
8. Select a best block from available ones, generate scripts. Then remove the selected block from available list. If no available block, we will make a warning.
9. Move clock to the end of the block, and jump back to step 7.
10. Close program.

Algorithms to select a block

1. Get hour angle and airmass of every available block.
2. Pick out candidates blocks, by airmass and hour angle.
3. Pick out blocks within the lowest 4 degrees of candidates as the final candidates.
4. If no good candidates, we need to skip some time.
5. Calculate key for each candidate block, the key is sum of ranks of block ra, dec, and airmass. That means, choose westest and southeat and highest.