PyPlate: a software package for processing digitized astronomical photographic plates

Taavi Tuvikene

The APPLAUSE Collaboration

Bamberg, March 11, 2019
What do we want to accomplish?

Raw digitized data
- High-resolution scans
- Low-resolution preview images
- Digitized logbook pages
- Transcribed metadata
- APPLAUSE: ~ 100 000 scans in 24 archives (~ 50 TB)

Processed data / publication
- Scans in FITS format
- Metadata in FITS headers and in a relational database
- Sources extracted from scans + calibrated coordinates and magnitudes
- Astrometric solution in FITS
Development started in 2013...

• Harvard DASCH pipeline was described in couple of papers
  • Laycock et al. (2010), Tang et al. (2013)
• SExtractor, Astrometry.net, SCAMP were available
• Python was coming into wide use in astronomy
• Astropy version was 0.2.x
**PyPlate versions**

<table>
<thead>
<tr>
<th>PyPlate</th>
<th>APPLAUSE</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>DR1</td>
<td>2015-02-14</td>
</tr>
<tr>
<td>2.0</td>
<td>DR2</td>
<td>2015-12-23</td>
</tr>
<tr>
<td>3.0</td>
<td>DR3</td>
<td>2017-10-23</td>
</tr>
<tr>
<td>3.1</td>
<td>—</td>
<td>2019-03-09</td>
</tr>
<tr>
<td>4.0</td>
<td>DR4</td>
<td>Summer 2019</td>
</tr>
</tbody>
</table>
PyPlate modules

- metadata
- solve
- image
- database
- pipeline
**metadata**
- Read CSV files
- Read WFPDB files
- Handle relations
- Calculate exposure times
- Create FITS headers

**solve**

**image**

**database**

**pipeline**

Metadata on plates, scans, logbooks, logpages

Wide-Field Plate Database (Tsvetkov et al. 1997)

Associations between plates, scans, previews, logbooks, logpages

From original (sidereal time, local time) to UT, JD, and HJD. Calculate mid-exposure times.

Following FITS standard and grouping keywords for better readability. Each keyword is documented with a comment.
Using the SExtractor software

Machine learning: following talk by Gal Matijevic

Currently using Tycho-2 and UCAC4. Soon: Gaia DR2
Initially with Astrometry.net, then with SCAMP in subfields. Getting rid of wave pattern caused by flatbed scanners.
metadata
- Read CSV files
- Read WFPDB files
- Handle relations
- Calculate exposure times
- Create FITS headers

solve
- Extract sources
- Flag artefacts
- Apply proper motions
- Solve astrometry
- Photometric calibration
- Crossmatch with ext. catalogs

image
database
pipeline
metadata
- Read CSV files
- Read WFPDB files
- Handle relations
- Calculate exposure times
- Create FITS headers

solve
- Extract sources
- Flag artefacts
- Apply proper motions
- Solve astrometry
- Photometric calibration
- Crossmatch with ext. catalogs

image
database
pipeline

Global calibration curve, corrections in subfields
**metadata**
- Read CSV files
- Read WFPDB files
- Handle relations
- Calculate exposure times
- Create FITS headers

**solve**
- Extract sources
- Flag artefacts
- Apply proper motions
- Solve astrometry
- Photometric calibration
- Crossmatch with ext. catalogs

**image**
- Using the SExtractor software

**database**
- Machine learning: following talk by Gal Matijevic

**pipeline**
- Currently using Tycho-2 and UCAC4. Soon: Gaia DR2
- Initially with Astrometry.net, then with SCAMP in subfields. Getting rid of wave pattern caused by flatbed scanners.
- Global calibration curve, corrections in subfields
- Currently: Tycho-2, UCAC4, APASS. Soon: Gaia DR2
metadata
- Read CSV files
- Read WFPDB files
- Handle relations
- Calculate exposure times
- Create FITS headers

solve
- Extract sources
- Flag artefacts
- Apply proper motions
- Solve astrometry
- Photometric calibration
- Crossmatch with ext. catalogs

image
- Convert TIFF to FITS
- Separate wedges

database
- Create table structure
- Write data to database

pipeline
- Process scans
- Parallel processing
Easy to install

Requires Python installation, then

    pip install pyplate
<table>
<thead>
<tr>
<th>Plate</th>
<th>Date</th>
<th>Time</th>
<th>Magnitude</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>plate1</td>
<td>1956-03-11</td>
<td>21:11:30</td>
<td>600</td>
<td>Observer Name, Europe/Berlin</td>
</tr>
<tr>
<td>plate2</td>
<td>1956-03-11</td>
<td>21:30:00</td>
<td>1800</td>
<td>Observer Name, UT</td>
</tr>
<tr>
<td>plate3</td>
<td>1956-03-12</td>
<td>23:05:00</td>
<td>1200</td>
<td>Observer Name, ST</td>
</tr>
</tbody>
</table>
Example: scan metadata (CSV file)

plate_0001.fits, plate1, Scan Author, 2018-10-12
plate_0002.fits, plate2, Scan Author, 2018-10-12
plate_0003.fits, plate3, Scan Author, 2018-10-13
## Example: configuration file

### [Files]
- `csv_dir = /path/to/csv/dir`
- `plate_csv = my_plates.csv`
- `scan_csv = my_scans.csv`

### [my_plates.csv]
- `plate_id = 1`
- `date_orig = 2`
- `tms_orig = 3`
- `exptime = 4`
- `observer = 5`
- `tz_orig = 6`

### [my_scans.csv]
- `filename = 1`
- `plate_id = 2`
- `scan_author = 3`
- `datescan = 4`
Example: reading metadata

```python
import pyplate
archive = pyplate.metadata.Archive()
archive.assign_conf('/path/to/my_archive.conf')
archive.read_csv()
```
Example: metadata calculations

plate_list = archive.get_platelist()

# Iterate over the plate list
for pid in plate_list:
    plate = archive.get_platemeta(plate_id=pid)
    plate.calculate()

    # Then do something with the metadata
    print(plate['jd_avg'])
Example: create FITS header

```python
header = pyplate.metadata.PlateHeader()
header.assign_conf(archive.conf)
header.populate()
plate1 = archive.get_platemeta(plate_id='plate1')
header.update_from_platemeta(plate1)
header.output_to_fits('plate_0001.fits')
```
SIMPLE = T / file conforms to FITS standard
BITPIX = 16 / number of bits per data pixel
NAXIS = 2 / number of data axes
NAXIS1 = 0 / length of data axis 1
NAXIS2 = 0 / length of data axis 2
BSCALE = 1.0 / physical_value = BZERO + BSCALE * array_value
BZERO = 32768 / physical_value = BZERO + BSCALE * array_value
MINVAL = / minimum image value
MAXVAL = / maximum image value
EXTEND = T / file may contain extensions

---------------------------------------

Original data of the observation

DATEORIG = '1956-03-11' / recorded date of the observation
TMS-ORIG = '21:11:30' / recorded time of the start of exposure 1
TME-ORIG = ' ' / recorded time of the end of exposure 1
JDA-ORIG = / recorded Julian date, mid-point of exposure 1
TIMEFLAG = ' ' / quality flag of recorded time
RA-ORIG = ' ' / recorded right ascension of exposure 1
DEC-ORIG = ' ' / recorded declination of exposure 1
COORFLAG = ' ' / quality flag of recorded coordinates
OBJECT = ' ' / observed object or field (exposure 1)
OBJTYPE = ' ' / object type
EXPTIME = 600 / [s] exposure time of exposure 1
NUMEXP = 1 / number of exposures of the plate
Computed data of the observation

DATE-OBS= '1956-03-11T20:11:30' / UT date of the start of exposure 1
DATE-AVG= '1956-03-11T20:16:30' / UT date of the mid-point of exposure 1
DATE-END= '        '           / UT date of the end of exposure 1

YEAR    =        1956.19258404 / decimal year of the start of exposure 1
YEAR-AVG=        1956.19259354 / decimal year of the mid-point of exposure 1

JD      =        2435544.34132 / Julian date at the start of exposure 1
JD-AVG  =        2435544.34479 / Julian date at the mid-point of exposure 1

HJD-AVG =                      / heliocentric JD at the mid-point of exposure 1
RA      = '        '           / right ascension of pointing (J2000) "h:m:s"
DEC     = '        '           / declination of pointing (J2000) "d:m:s"
RA_DEG  = '        '           / [deg] right ascension of pointing (J2000)
DEC_DEG = '        '           / [deg] declination of pointing (J2000)

Scan

SCANNER = 'Epson Expression 10000XL' / scanner name
SCANRES1=                 2400 / [dpi] scan resolution along axis 1
SCANRES2=                 2400 / [dpi] scan resolution along axis 2
PIXSIZE1=              10.5833 / [um] pixel size along axis 1
PIXSIZE2=              10.5833 / [um] pixel size along axis 2
DATESCAN= '2018-10-12'         / scan date and time
SCANAUTH= 'Scan Author'        / author of scan
FILENAME = 'plate_0001.fits'  / filename of the plate scan
FN-WEDGE = '        '       / filename of the wedge scan
FN-PRE   = '        '       / filename of the preview image
FN-COVER = '        '       / filename of the plate cover image
ORIGIN   = '        '
DATE     = '2019-03-09T20:44:27' / last change of this file

--- wcs ---

--- licence ---

--- acknowledgements ---

--- history ---
HISTORY Header created with PyPlate v3.1.0 at 2019-03-09T20:44:27
HISTORY Header updated with PyPlate v3.1.0 at 2019-03-09T20:44:27

--- checksums ---
CHECKSUM = '        '
DATASUM  = '        '
Example: parallel processing

```python
filenames = archive.get_scanlist()
pipeline = pyplate.pipeline.PlatePipeline()
pipeline.assign_conf(archive.conf)
pipeline.parallel_run(filenames)
```
Performance: APPLAUSE DR3

- Extracted ~3.5 billion sources from ~70 000 scans
- Robust: only 17 processes out of 70730 had problems
- Processing time: ~900 CPU core-days
- Used 20 processes in parallel
Results: V466 Cyg light curve

APPLAUSE DR2 (PyPlate 2.0)

Calibration in annular bins

APPLAUSE DR3 (PyPlate 3.0)

Calibration in sub-fields
PyPlate

• It is open source!
• Install: `pip install pyplate`
• Use it as a library for your needs
• Check documentation: `pyplate.readthedocs.io`
• Contribute: `www.github.com/astrotuvi/pyplate`
  Report bugs, feature requests, etc
Thank you!

E-mail: taavi.tuvikene@ut.ee

GitHub: astrotuvi