Automated Image Quality Monitoring with IQMon

Josh Walawender[‡*]

https://www.youtube.com/watch?v=dGLkDOvYOHA

Abstract—Automated telescopes are capable of generating images more quickly than they can be inspected by a human, but detailed information on the performance of the telescope is valuable for monitoring and tuning of their operation. The IQMon (Image Quality Monitor) package[1] was developed to provide basic image quality metrics of automated telescopes in near real time.

Index Terms—astronomy, automated telescopes, image quality

Introduction

Using existing tools such as astropy [Astropy2013], astrometry.net [Lang2010], source extractor [Bertin1996], [Bertin2010a], SCAMP [Bertin2006], [Bertin2010b], and SWARP [Bertin2010c], IQMon analyzes images and provides the user with a quick way to determine whether the telescope is performing at the required level.

For projects which need to monitor the operation of an imaging telescope, IQMon is meant to provide a middle ground solution between simply examining the operations logs (e.g. those output by the control system) and a full data analysis pipeline. IQMon provides more information than typical operations logs while also giving a "ground truth" analysis since it looks at the actual data and not just what the system intended to do. While not as powerful as a full data pipeline, it is designed to provide operational information instead of scientific data products and thus its output is tuned to the task of examining the quality of the data and evaluating it for common problems.

IQMon can provide a determination of whether the telescope is focused (from the typical Full Width at Half Maximum, or FWHM, of stars in the image), whether it is pointing accurately (obtained from a comparison of the target coordinates with the astrometrically solved coordinates), whether the tracking or guiding is adequate (from the typical ellipticity of stars in the image), and whether the night is photometric (obtained from the typical photometric zero point of stars in the image). For wide field systems which detect many stars in each image, these metrics can be spatially resolved allowing for more detailed analysis such as differentiating between tracking error, focus error, and optical aberration or determining if the dome is partially obscuring the telescope aperture.

* *Corresponding author: joshwalawender@me.com*
‡ *Subaru Telescope, National Astronomical Observatory of Japan*

Copyright © 2015 Josh Walawender. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

1. Source code at https://github.com/joshwalawender/IQMon

To date, IQMon has been deployed on three disparate optical systems. Two for the VYSOS Project which performs photometric monitoring of young stars: a 735mm focal length wide field imager with a monochrome CCD camera which undersamples the point spread function (PSF) and an 0.5 meter f/8 telescope with a monochrome CCD camera with well sampled PSF. It has also been deployed on the prototype unit for the PANOPTES[2] Project: an 85mm focal length camera lens and DSLR camera (with Bayer color array) designed for very wide field photometry. PANOPTES aims to create a global network of low-cost, robotic observatories for citizen science projects. IQMon has provided valuable diagnostic information about system performance in all cases.

Structure and Example Use

IQMon operates by using `Telescope` and `Image` classes. The `Telescope` object contains basic information about the telescope which took the data. When a `Telescope` object is instantiated, a configuration file is read which contains information on the telescope and controls various user-configurable parameters and preferences for IQMon. The configuration file is a YAML document and is read using the pyyaml[3] module.

An `Image` object is instantiated with a path to a file with one of the supported image formats and with a reference to a `Telescope` object. The image analysis process is simply a series of calls to methods on the `Image` object.

At the most basic level, IQMon is a sequencing tool which calls other programs (e.g. SExtractor, Astronometry.net) and tracks their output. These calls are all made using the `subprocess32` module, so all of these dependencies need to be installed and visible in the path for IQMon to function properly.

The IQMon philosophy is to never operate on the raw file itself, but instead to create a "working file" (using the `read_image` method) and store it in a temporary directory. If the raw image file is a FITS file, then `read_image` simply copies the raw file to the temporary directory and records this file name and path in the `working_file` property. If the file is a raw image file from a DSLR (e.g. `.CR2` or `.dng` format), then `read_image` will call `dcraw`[4] using the subprocess32 module[5] to convert the file to `.ppm`. The file is then converted to FITS format using either `pamtofits` or `pnmtofits` tools from the `netpbm`[6] package. IQMon then operates on the green channel of that resulting FITS file. For full functionality, the user should populate the header of this FITS file with appropriate FITS

2. http://projectpanoptes.org/v1/
3. http://pyyaml.org

keywords (e.g. RA, DEC, EXPTIME, DATE-OBS, etc.). To date, IQMon has only been tested with FITS and .CR2 files, but should in principle work with numerous DSLR raw format images.

IQMon has been tested with Python 2.7.X, testing with Python 3.X is pending. Python 3.X compatibility notes will be posted to the readme file on the git repository. IQMon runs successfully on both Mac OS X and linux. Windows compatibility is untested, but will be limited by the availability of dependencies (astrometry.net, SExtractor, etc.).

Because the system is designed to do quick evaluations of image quality, the primary concept is an object representing a **single** image. IQMon does not do any image stacking or other processing which would be applied to more than one image at a time nor is it built around other organizational concepts such as targets or visits. It is not intended to supplant a full data reduction and analysis package. The output of IQMon, however, can be stored in a MongoDB[7] database making it potentially useful for collecting information on observing concepts which span multiple images such as targets, nights, or visits. It might also be useful as a preprocessing step for a more complex data pipeline.

The time to process an image varies depending on many factors. It has been well studied for two of the systems mentioned in the Introduction. Both of these systems are analyzed by the same computer (a 2.3GHz Quad-Core Intel Core i7 with 8GB of RAM), so they share the system resources during the night.

In both cases the full image analysis takes tens of seconds per image, but depends on the number of stars in the image. The total analysis time for these systems is dominated by the SCAMP solve (roughly one third of the total time) and the generation of two JPEG images (also roughly one third of the total time). IQMon itself is single threaded, but many of the programs it calls, such as SCAMP, are multi threaded and so will take advantage of multiple cores.

In the following sections, I will describe a simple example of evaluating image quality for a single image. A more complex example which is updated in concert with IQMon can be found in the measure_image.py script at the git repository for the VYSOS project[8]. That process can then be wrapped in a simple program to monitor a directory for images and analyze them as they are written to disk (see the watch_directory.py script in the same VYSOS repository for an example). This enables automatic near real time analysis.

Configuration and Reading the Image In

After importing IQMon, the first step would be to instantiate the Telescope object which takes a configuration file as its input. The next step is to instantiate an Image object with the path to the image file and the Telescope object representing the telescope which took that image.

```
tel = IQMon.Telescope('~/MyTelescope.yaml')
im = IQMon.Image('~/MyImage.fits', tel)
```

IQMon writes a log which is intended to provide useful information to the user (not just the developer) and shows the progress of the analysis. We can either pass in a logger object from Python's logging module, or ask IQMon to create one:

```
# create a new logger object
im.make_logger(verbose=False)
print('Logging to file {}'.format(im.logfile))
im.logger.info('This is a log entry')
```

The first step for any image analysis is likely to be to call the read_image method. After calling read_image, the FITS header is read and various Image object properties are populated by calling the read_header method.

```
# Generate working file copy of the raw image
im.read_image()
# Read the fits header
im.read_header()
```

Once the image has been read in and a working file created, IQMon uses various third party tools to perform image analysis. The following sections describe some of the analysis steps which are available.

PSF Size Measurements with Source Extractor

Source Extractor (SExtractor) [Bertin1996], [Bertin2010a] is a program which builds a catalog of sources (stars, galaxies, etc.) detected in an image. SExtractor is called using the run_SExtractor method which invokes the command using the subprocess32 module. Customization parameters can be passed to Source Extractor using the telescope configuration file.

The output file of SExtractor is read in and stored as an astropy table object. Stars with SExtractor generated flags are removed from the table and the table is stored as a property of the image object.

Determining the PSF size from the SExtractor results is done with the determine_FWHM method. The full width at half maximum (FWHM) and ellipticity values for the image are a weighted average of the FWHM and ellipticity values for the individual stars.

These steps not only provide the typical FWHM (which can indicate if the image is in focus), they can also be used to guess at whether the image is "blank" (i.e. very few stars are visible either because of cloud cover or other system failure). For example:

```
im.run_SExtractor()
# Consider the image to be blank if <10 stars
if im.n_stars_SExtracted < 10:
    im.logger.warning('Only {} stars found.'
                      .format(im.n_stars_SExtracted))
    im.logger.warning('Image may be blank.')
else:
    im.determine_FWHM()
```

Pointing Determination and Pointing Error

IQMon also contains a solve_astrometry method to invoke the solve-field command which is part of the astrometry.net software. The call to solve-field is only intended to determine basic pointing and orientation and so IQMon does not use the SIP polynomial fit of distortion in the image.

Once a world coordinate system (WCS) is present in the image header, then the determine_pointing_error method can be called which compares the right ascension (RA) and declination (DEC) values read from the RA and DEC keywords in the header (which are presumed to be the telescope's intended pointing) to the RA and DEC values of the center pixel which are calculated using the astropy.wcs module. The separation between the two

4. http://www.cybercom.net/~dcoffin/dcraw/

5. The subprocess32 module "is a backport of the subprocess standard library module from Python 3.2 & 3.3 for use on Python 2.4, 2.5, 2.6 and 2.7" (from https://pypi.python.org/pypi/subprocess32). It is used instead of the standard subprocess module due to its support for timeout functionality.

6. http://netpbm.sourceforge.net

7. http://www.mongodb.org

8. https://github.com/joshwalawender/VYSOStools

coordinates is determined using the `separation` method available in the `SkyCoord` object of the `astropy.coordinates` module. The magnitude of the separation between the two is reported as the pointing error.

```
# If WCS is not present, solve with astrometry.net,
if not im.image_WCS:
    im.solve_astrometry()
# Determine pointing error by comparing telescope
# pointing coordinates from the header with WCS.
im.determine_pointing_error()
```

Astrometric Distortion Correction

In order to make an accurate comparison of the photometry of stars detected in the image and stars present in a chosen stellar catalog, many optical systems require distortion coefficients to be fitted as part of the astrometric solution. IQMon uses the SCAMP software to fit distortions.

SCAMP is invoked with the `run_SCAMP` method. Once a SCAMP solution has been determined, the image can be remapped to new pixels without distortions using the SWARP tool with the `run_SWARP` method.

```
# If the image has a WCS and a SExtractor catalog,
# run SCAMP to determine a WCS with distortions.
if im.image_WCS and im.SExtractor_results:
    im.run_SCAMP()
    if im.SCAMP_successful:
        # Remap the pixels to a rectilinear grid
        im.run_SWarp()
```

A Note on Astrometry.net and SCAMP

In principle, Astrometry.net can solve for distortions. The `-t` option on `solve-field` allows the user to specify the order of the SIP polynomial which the program should fit. This is available in IQMon by calling the `solve_astrometry` method with the `SIP` keyword set to the polynomial order to pass to `solve-field`.

In my experience working with the first two systems IQMon was used on, I found that high order solves were not necessarily reliable or timely. The `solve-field` operation would sometimes fail to solve or would process for a very long time which would cause the analysis system to fail to keep up with the data rate from the two telescopes.

This is why SCAMP is also available in IQMon and is the recommended astrometric solution if you want full distortion correction. By defining a SCAMP "ahead" file, you can incorporate previous knowledge of the optical system's distortion characteristics rather then solving blindly. With a proper ahead file, SCAMP was a more reliable solution.

SWarp is used because (at the time) `astropy.wcs` did not handle the distortion coefficients as written by SCAMP. To solve this, SWarp remaps the pixels to de-distort the image which means that the WCS is properly described by a very basic set of header keywords (`CRPIXn`, `CRVALn`, `PCn_m`, etc.) which almost every analysis program supports.

Estimating the Photometric Zero Point

With a full astrometric solution, SExtractor photometry, and a catalog of stellar magnitude values, we can estimate the zero point for the image and use that as an indicator of clouds or other aperture obscurations.

The `get_catalog` method can be used to download a catalog of stars from VizieR using the `astroquery`[9] module. Alternatively, support for a local copy of the UCAC4 catalog is available using the `get_local_UCAC4` method.

Once a catalog is obtained, the `run_SExtractor` method is invoked again, this time with the `assoc` keyword set to `True`. This will limit the resulting catalog of detected stars to stars which **both** exist in the catalog and also are detected in the image. This may significantly decrease the number of stars used for the FWHM and ellipticity calculation, but may also remove spurious detections of image artifacts which would improve the reliability of the measured values.

```
# Retrieve catalog defined in config file
im.get_catalog()
im.run_SExtractor(assoc=True)
im.determine_FWHM()
im.measure_zero_point()
```

In the above example code, `determine_FWHM` is invoked again in order to use the new SExtractor catalog for the calculation.

The `measure_zero_point` method determines the zero point by taking the weighted average of the difference between the measured instrumental magnitude from SExtractor and the catalog magnitude in the same filter.

It should be noted that unless custom code is added to handle reduction steps such as dark/bias subtraction and flat fielding, the zero point result will be influenced by systematics due to those effects. In addition, the choice of catalog and the relative response curve of the filter in use and the filter defined by the catalog's photometric system will also introduce systematic offsets. For many systems (especially typical visible light CCDs), the zero point value from IQMon can be used to compare throughput from image to image, but should not be used to compare different equipment configurations.

Analysis Results and Mongo Database Integration

Results of the IQMon measurements for each image are stored as properties of the `Image` object as `astropy.units.Quantity`. For example, the `FWHM` value is in units of pixels, but can be converted to arcseconds using the equivalency which is automatically defined by the `Telescope` object (`tel.pixel_scale_equivalency`) for this purpose.

```
## Results are typically astropy.units quantities
## and can be manipulated as such.  For example:
print('Image FWHM = {:.1f}'.format(im.FWHM))
print('Image FWHM = {:.1f}'.format(\
      im.FWHM.to(u.arcsec, equivalencies=\
      im.tel.pixel_scale_equivalency)))
print('Zero Point = {:.2f}'.format(im.zero_point))
print('Pointing Error = {:.1f}'.format(\
      im.pointing_error.to(u.arcmin)))
```

These results can also be stored for later use. Methods exist to write them to an `astropy.Table` (the `add_summary_entry` method) and to a YAML document (the `add_yaml_entry` method), but the preferred storage solution is to use a mongo database as that is compatible with the `tornado` web application included with IQMon (see below).

The address, port number, database name, and collection name to use with `pyMongo` to add the results to an existing mongo database are set by the Telescope configuration file. The `add_mongo_entry` method adds a dictionary of values with the results of the IQMon analysis.

9. http://dx.doi.org/10.6084/m9.figshare.805208

Flags

For the four primary measurements (FWHM, ellipticity, pointing error, and zero point), the configuration file may contain a threshold value. If the measured value exceeds the threshold (or is below the threshold in the case of zero point), then the image is "flagged" as an indication that there may be a potential problem with the data. The flags property of an `Image` object stores a dictionary with the flag name and a boolean value as the dictionary elements.

This can be useful when summarizing results. For example, the Tornado web application provided with IQMon (see the Tornado Web Application section) lists images and will color code a field red if that field is flagged. In this way, a user can easily see when and where problems might have occurred.

Images and Plots

In addition to generating single values for FWHM, ellipticity, and zero point to represent the image, IQMon can also generate more detailed plots with additional information.

A plot with PSF quality information can be generated when `determine_FWHM` is called by setting the `plot=True` keyword. This generates a .png file (see Fig. 1) using matplotlib [matplotlib] which shows detailed information about the point spread function (FWHM and ellipticity metrics) including histograms of individual values, a spatial map of FWHM and ellipticity over the image, and plots showing the ellipticity vs. radius within the image (which can be used to show whether off axis aberrations influence the ellipticity measure) and the correlation between the measured PSF position angle and the position angle of the star within the image (which can be used to differentiate between tracking error and off axis aberrations).

In the example plot (Fig. 1), we can see several different effects. First, from the spatial distribution of FWHM and ellipticity, as well as the ellipticity vs. radius plot, we see that image quality is falling off at large radii. This image is from a wide field imaging system and we are seeing the signature of off axis aberrations. This is also suggested in the plot of the correlation between the measured PSF position angle and the position angle of the star within the image which shows strong diagonal components indicating that position within the image influences the PSF. There is also, however, a vertical component in that plot at $PA \sim 0$ which is suggestive of image drift perhaps due to slight polar misalignment or flexure.

A plot with additional information on the zero point can be generated when calling `measure_zero_point` by setting the `plot` keyword to `True`. This generates a .png file (see Fig. 2) using matplotlib which shows plots of instrumental magnitude vs. catalog magnitude, a histogram of zero point values, a plot of magnitude residuals vs. catalog magnitude, and a a spatial map of zero point over the image.

JPEG versions of the image can be generated using the `make_JPEG` method. The jpeg can be binned or cropped using the `binning` or `crop` keyword arguments and various overlays can be generated showing, for example, the pointing error and detected and catalog stars.

The JPEG overlays can be useful in evaluating the performance of SExtractor and SCAMP. In the example shown in Fig. 3, the stars marked as detected by SExtractor (which was run with the `assoc` keyword set to `True`) show that there are no stars detected in the very corners of the image. This indicates that the SCAMP distortion solution did not accurately fit the WCS in the corners and could be improved. Poor SCAMP solutions can also show up even more dramatically when entire radial zones of the image have no matched stars.

Tornado Web Application

IQMon comes with a tornado web application which, while it can be run stand alone, is intended to be used as a template for adding IQMon results to a more customized web page. The web application (`web_server.py`) contains two `tornado` web handlers: `ListOfNights` and `ListOfImages`. The first generates a page which lists UT dates and if there are image results associated with a date, then it provides a link to a page with the list of image results for that date. The second handler (see Fig. 4) produces the page which lists the images for a particular UT date (or target name) and provides a table formatted list of the IQMon measurement results for each image with flagged values color coded red, along with links to jpegs and plots generated for that image.

This web application is intended to be the primary interface for users. It provides three levels of interaction to the user. First, a custom plot of IQMon results over the course of a night is easy to generate from the mongo database entries and represents the highest level of interaction. Using such a plot, serious problems which affect many images can be detected at a glance. Users can then drill down to see a list of images for that UT date and see system performance as a table of IQMon results with flagged values highlighted in red. Finally an individual image can be examined as a jpeg with overlays or by using the PSF quality plots or zero point plots to examine detailed performance.

Conclusions

IQMon provides a way to evaluate the performance of automated telescopes. It allows the user to build a customized analysis for their particular application by assembling a script which includes only those steps which are required. Using the included tornado web application, a user can quickly and easily view the results and determine whether the observatory is performing acceptably or if it needs attention.

Over roughly two years of routine operation with two telescopes, it has enabled quick alerting of problems including stuck focus drives, poorly aligned dome rotation, and poor tracking model correction. Previously, some of these problems would have gone unnoticed until a spot check of the data downloaded from the site revealed them or they would have required a time consuming reading of the nightly system logs to reveal. Use of IQMon has resulted in greater uptime and improved data quality for both telescopes.

REFERENCES

[Astropy2013] Astropy Collaboration, Robitaille, T.~P., Tollerud, E.~J., et al. *Astropy: A community Python package for astronomy* 2013, A&A, 558, A33

[Bertin1996] Bertin, E., & Arnouts, S. *SExtractor: Software for source extraction*, 1996, A&AS, 117, 393

[Bertin2006] Bertin, E. *Automatic Astrometric and Photometric Calibration with SCAMP*, 2006, Astronomical Data Analysis Software and Systems XV, 351, 112

[Bertin2010b] Bertin, E. *SCAMP: Automatic Astrometric and Photometric Calibration*, 2010, Astrophysics Source Code Library, 1010.063

[Bertin2010a] Bertin, E., & Arnouts, S. *SExtractor: Source Extractor*, 2010, Astrophysics Source Code Library, 1010.064

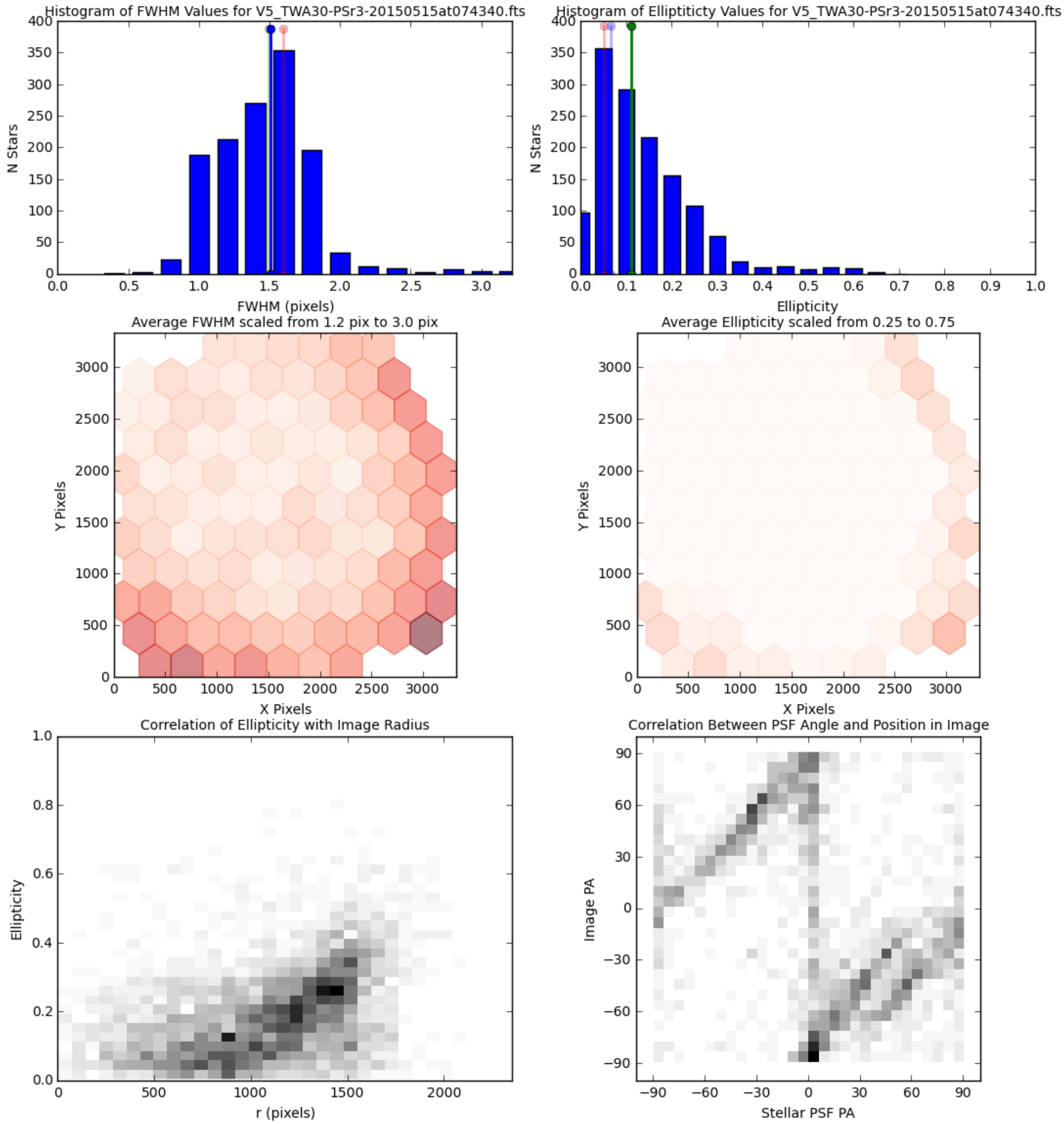

Fig. 1: *An example of the plot which can be produced using the* `determine_FWHM` *method. The plot shows histograms of the FWHM and ellipticity values (upper left and upper right respectively), the spatial distribution of FWHM and ellipticity values (middle left and middle right), ellipticity vs. radius within the image (lower left), and the correlation between the measured PSF position angle and the position angle of the star within the image (lower right).*

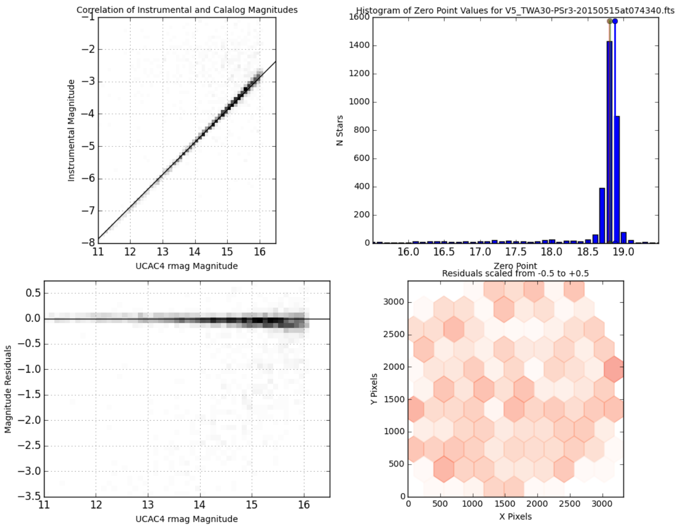

Fig. 2: *An example of the plot which can be produced using the* `measure_zero_point` *method. The plot shows the correlation between instrumental magnitude and catalog magnitude (upper left), a histogram of zero point values (upper right), a plot of the residuals vs. catalog magnitude (lower left), and a spatial distribution of the residuals (lower left).*

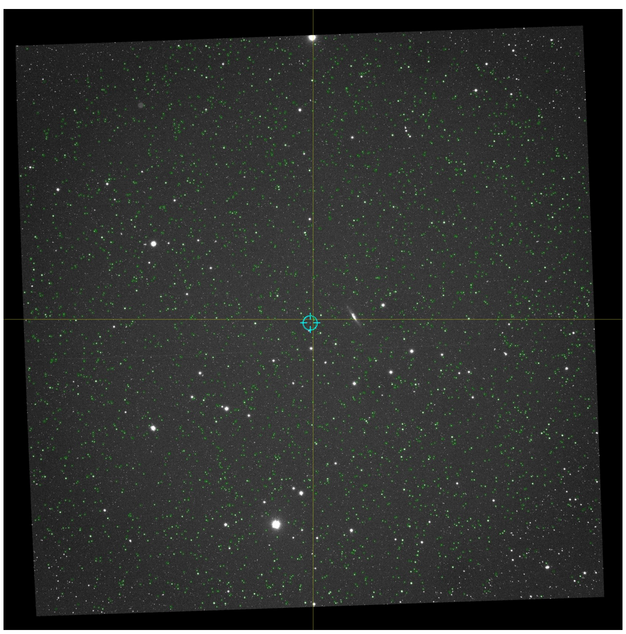

Fig. 3: *An example jpeg generated by the* `make_JPEG` *method using the* `mark_detected_stars` *and* `mark_pointing` *options. In this example, pointing error has placed the target (marked by the cyan crosshair) to the lower right (southwest) of the image center (marked by the yellow lines). Stars from the UCAC4 catalog which were detected in the image are marked with green circles.*

[Bertin2010c] Bertin, E. *SWarp: Resampling and Co-adding FITS Images Together* 2010, Astrophysics Source Code Library, 1010.068
[Lang2010] Lang, D., Hogg, D. W., Mierle, K., Blanton, M., & Roweis, S., *Astrometry.net: Blind astrometric calibration of arbitrary astronomical images* 2010, AJ, 137, 1782–1800
[matplotlib] Hunter, J. D., *Matplotlib: A 2D graphics environment* 2007, Computing In Science & Engineering, 9, 90-95

IQMon Results for 20150523UT for VYSOS-5

Exposure Start (Date and Time UT)	Image File Name	Alt (deg)	Az (deg)	Airmass	Moon Sep (deg)	Moon Illum. (%)	FWHM (pix)	Ellip.	Pointing Error (arcmin)	Zero Point (mag)	N Stars	Process Time (sec)
20150523UT 05:56:00	V5_TWA30-PSr3-20150523at055554.fts (JPEG) (PSF) (ZP)	40.1	180.8	1.56	64.8	27 %	3.62	0.61	0.39	16.03	321	63 s
20150523UT 05:58:19	V5_TWA30-PSr3-20150523at055812.fts (JPEG) (PSF) (ZP)	40.0	181.4	1.56	64.8	27 %	2.67	0.61	0.43	16.76	739	48 s
20150523UT 06:00:40	V5_TWA30-PSr3-20150523at060033.fts (JPEG) (PSF) (ZP)	40.0	182.1	1.56	64.8	27 %	2.32	0.56	0.64	17.28	536	52 s
20150523UT 06:17:30	V5_TWA30-PSr3-20150523at061724.fts (JPEG) (PSF) (ZP)	39.7	186.8	1.57	64.7	27 %	2.21	0.11	0.31	18.53	662	56 s
20150523UT 06:19:49	V5_TWA30-PSr3-20150523at061943.fts (JPEG) (PSF) (ZP)	39.7	187.5	1.57	64.6	27 %	2.10	0.10	0.49	18.66	647	55 s
20150523UT 06:22:09	V5_TWA30-PSr3-20150523at062202.fts (JPEG) (PSF) (ZP)	39.6	188.1	1.57	64.6	27 %	1.98	0.06	0.67	18.72	642	59 s
20150523UT 06:27:52	V5_TWA30-PSr3-20150523at062746.fts (JPEG) (PSF) (ZP)	39.4	189.7	1.58	64.6	27 %	1.73	0.09	2.85	18.84	630	57 s
20150523UT 06:30:18	V5_TWA30-PSr3-20150523at063011.fts (JPEG) (PSF) (ZP)	39.3	190.4	1.58	64.6	27 %	1.76	0.16	2.58	18.87	634	60 s
20150523UT 06:32:43	V5_TWA30-PSr3-20150523at063237.fts (JPEG) (PSF) (ZP)	39.2	191.0	1.59	64.6	27 %	1.91	0.10	0.31	18.88	684	59 s
20150523UT 06:36:08	V5_TWA30-PSr3-20150523at063602.fts (JPEG) (PSF) (ZP)	39.0	191.9	1.59	64.5	27 %	1.43	0.04	2.82	18.91	656	57 s
20150523UT 06:38:34	V5_TWA30-PSr3-20150523at063827.fts (JPEG) (PSF) (ZP)	38.9	192.6	1.60	64.5	27 %	1.56	0.08	0.47	18.89	704	58 s
20150523UT 06:40:53	V5_TWA30-PSr3-20150523at064047.fts (JPEG) (PSF) (ZP)	38.8	193.2	1.60	64.5	27 %	1.35	0.07	0.72	18.89	688	60 s
20150523UT 06:43:37	V5_TWA30-PSr3-20150523at064331.fts (JPEG) (PSF) (ZP)	38.6	194.0	1.61	64.5	27 %	1.67	0.08	1.00	18.88	693	58 s
20150523UT 06:45:57	V5_TWA30-PSr3-20150523at064551.fts (JPEG) (PSF) (ZP)	38.5	194.6	1.61	64.5	27 %	1.51	0.07	1.04	18.88	691	63 s
20150523UT 06:48:16	V5_TWA30-PSr3-20150523at064809.fts (JPEG) (PSF) (ZP)	38.3	195.2	1.62	64.5	27 %	1.83	0.11	1.29	18.88	693	67 s
20150523UT 06:51:00	V5_TWA30-PSr3-20150523at065054.fts (JPEG) (PSF) (ZP)	38.2	195.9	1.62	64.4	27 %	1.50	0.05	1.56	18.90	691	62 s

Fig. 4: *An example of the `ListOfImages` handler of the tornado web application. In this example, a user can easily determine that the first few images of the night had a problem (indicated by the red flagged values). Based on examination of the JPEGs, this turns out to have been due to the dome rotation being misaligned and partially blocking the telescope aperture leading to large FWHM and ellipticity values (image elongation due to "glints" of the dome edge) and low zero point values (due to aperture obscuration). The problem resolved itself without human intervention as can be seen by the green, un-flagged images which follow and which continued for the rest of the night.*