CHEOPS PIPE manual

Alexis Brandeker and Jayshil Patel

February 2024

1 Introduction

PIPE (*PSF Imagette Photometric Extraction*) is developed to extract photometry from CHEOPS (Benz et al. 2021) data using PSF (point-spread function) photometry. CHEOPS data typically consists of a cube of 200×200 pixel *subarray* frames that come from a region of the full 1024×1024 pixel detector. For bright sources ($G < 8$ mag), each subarray frame may be co-added from several exposures to save data rate on the downlink. In those cases, also smaller *imagettes* frames of 60×60 pixel diameter are downloaded that are not co-added, or at least less co-added for the brightest stars. PIPE was originally developed to extract photometry from the imagettes, where PSF extraction is particularly useful since the background varying in time, can be fitted simultaneously with the PSF. The small size of the imagettes otherwise makes it difficult to estimate the background with aperture photometry. Advantages in using the imagettes for photometry is the higher time resolution from shorter cadence, but also the reduced sensitivity to bad pixels, cosmic rays, and satellite transits.

1.1 When to use PIPE

The first question to ask is *Do I really need PIPE?* All CHEOPS data sets are processed by the CHEOPS Data Reduction Pipeline (DRP, Hoyer et al. 2020), which uses aperture photometry on the subarrays to provide extracted photometry. The DRP products are robust and optimal for intermediately bright stars ($G = 8 - 11$ mag) without complicated background and no imagettes available; in those cases there is usually no benefit to using PIPE, so the simplest approach is to use the extracted photometry already provided by the DRP. There are however some circumstances when PIPE can improve the quality of the extracted photometry:

- For bright stars ($G < 8$ mag) where imagettes exist and when higher time resolution is desired, e.g. to resolve stellar activity or improve the time constraint on transits. There is usually no noise penalty in doing photometry on imagettes, binned to the same time resolution as the subarray photometry, imagette photometry is as good or better (due to more efficient filtering out of satellite transits and cosmic rays).
- Whenever there are nearby background stars that are suspected to contaminate the photometric aperture; PSF photometry disentangles the contribution from the background stars, significantly reducing the roll modulation.
- For faint stars ($G > 11$ mag) where hot and telegraphic pixels (pixels with a strong variable dark current) contribute significantly to the noise. Since PSF photometry can effectively mask any contributions from bad pixels, this reduces their impact on the photometry and improves the quality of extraction. As CHEOPS ages, the number of bad pixels increases with time, moving the limit where PSF photometry becomes beneficial to increasingly brighter targets.
- Whenever the DRP-extracted light curve shows unexpected behaviour, the independent PIPE extraction can be used for verification. In particular, PIPE has a different sensitivity to background stars contaminating the DRP aperture.

Usually there is no penalty for using PIPE, except for the extra time it may take to repeat the reduction and extraction.

2 Installation

The easiest way to install PIPE is using `pip`:

```
pip install pipe-cheops
```

This should install PIPE and all its dependencies and make them available from your python installation. Alternatively, the latest development version can be downloaded from GitHub:

```
git clone https://github.com/alphapsa/PIPE.git
cd PIPE
python setup.py install
```

Note: To use PIPE after installation, you also need to set the paths to data and calibration files, as well as download the proper calibration files. See subsections below.

2.1 Dependencies

PIPE is developed for python 3 and has been tested for python 3.7 and later, but earlier versions might also work. PIPE makes use of, and is thus dependent on, the following four external modules: NumPy, SciPy, Astropy, and scikit-image (an image processing package). You probably already have the first three installed. If anything is missing, `pip` should automatically install them for you as part of the installation process of PIPE. If you install directly from the GitHub repository on the other hand you will need to ensure yourself that the required modules are installed and available on your system. In general we recommend Anaconda as a good way of installing and maintaining most of your scientific python needs.

> **Bug note:** Astropy up to 5.1 has a known incompatibility with NumPy 1.24 and later. Best update to Astropy 5.2 or later, in which case there is no problem. Should you be unable to upgrade, a workaround is to set the PIPE parameter `mjd2bjd = False` (see §A.1); this bypasses the MJD to BJD computation where the problem appears but produces tables where no barycentric correction has been made.

2.2 Setting paths

After calling PIPE for the first time after installation, e.g. by executing

```
import pipe
```

from the python prompt, it will ask for root paths to data files (`data_root`) and reference/calibration files directories (`ref_lib_data`). These are the locations where CHEOPS data are expected to be found in subdirectories, and where all calibration files are expected to be found, respectively. If the paths are not explicitly set, the default paths `cheops-pipe` directory inside the `home` directory, will be created and used. It is, however, possible to change the paths afterwards, which can be done by executing,

```
from pipe.config import get_conf_paths
get_conf_paths(overwrite=True)
```

2.3 Calibration files

In the `ref_lib_data` directory, there are a number of calibration files expected in order for PIPE to be able to reduce data from CHEOPS and extract photometry. Most files are optional in that they will not be required if their use is switched off by setting the corresponding PIPE parameter to `False`; the exception is the `gain` parameter that has to be set if not calculated, and the the PSF library that always needs to be present.

Table 1 lists the various calibration files used by PIPE. The non-linearity definition file and the PSF library are for now separately provided from a dropbox; the rest can be downloaded from the CHEOPS archive. Use the *Reference Data Query* tab and select the desired files to query from the drop list. The bad pixel maps and dark frames are updated on a monthly bases, so you will want to add calibration files that are close in time to the observations to be reduced.

3 Using PIPE

3.1 Data files

Best practice is to download all files relevant for a visit and put it in a subdirectory. PIPE assumes all data is under the previously configured data path `data_root` organised in subdirectories according to target and visit as `data_root/target/visit`, where `target` and `visit` are arbitrary strings and the `visit` directory contains all data as downloaded from DACE or the CHEOPS archive. From DACE, choose to download "All data products", where each visit is unpacked to the `target` directory and optionally rename the visit directory. Instead of using the name from the full file key from, e.g., `PR100006_TG00313_V0300`, it can be renamed to `313` so that the full data path becomes `data_root/55Cnc/313/` (naming the target `55Cnc`, but could be named anything). Data can also be downloaded from the CHEOPS archive, though care has to be taken that all required files listed in Table 2 are downloaded.

Table 1: Calibration files used by PIPE.

Calibration file	Function
`REF_APP_GainCorrection`	File with gain correction parameters, expected directly in the `ref_lib_data` path. Can be downloaded from the CHEOPS archive. Optional, overridden if `gain` PIPE parameter is set (to e.g. $1.95\,e^-/\mathrm{ADU}$); see §A.2.
`nonlin.npy`	Non-linearity definition file, expected directly in the `ref_lib_data` path. Optional, can be ignored (with no non-linear correction) if the PIPE parameter `non_lin = False`. See §A.2.
`REF_APP_FlatFieldTeff`	Contains the pre-launch determined flat field as a function of efefctive temperature of the spectral energy distribution of the target. Huge file ($\sim$1 GB) that can be downloaded from the CHEOPS archive. Optional, disabled if PIPE parameter `flatfield = False`.
`psf_lib/`	The PSF library containing sub-directories with PSF model parameters. The library must be re-populated if the target is relocated to a new position on the CHEOPS detector, see §5.
`BadPixels/`	Directory containing bad pixel maps. These can be downloaded from the CHEOPS archive as `REF_APP_BadPixelMap` and put into this directory. Choose files that are from near the date of the target observation. When PIPE runs it selects the bad pixel map file nearest the date of the observation. Taking bad pixels into account can be switched off by setting the PIPE parameter `mask_badpix = False`. See §A.4.
`DarkFrames/`	Directory containing dark current frames. These can be downloaded from the CHEOPS archive as `REF_APP_DarkFrame` and put into this directory. Choose files that are from near the date of the target observation. When PIPE runs it interpolates the dark frames bracketing the date of the observation (or picks the nearest if not bracketed). bad pixel map file nearest the date of the observation. Dark current subtraction can be switched off by setting the PIPE parameter `darksub = False`. See §A.2.

Table 2: Data files used by PIPE.

Data file	Function
`SCI_RAW_Attitude`	Stores attitude information, such as time, pointing, and roll angle.
`SCI_RAW_HkExtended`	Contains housekeeping parameters used for gain correction. Optional, overridden if `gain` PIPE parameter is set.
`EXT_PRE_StarCatalogue`	A catalogue of stars in the field, retrieved from the Gaia DR2. Used to model the star background; optional, set PIPE parameter `bgstars = False` if not available.
`SCI_RAW_SubArray`	A datacube with the raw subarray data from the visit.
`SCI_RAW_Imagette`	A datacube with the raw imagette data from the visit; optional. PIPE checks for availability and extracts only subarray photometry if imagettes are not available.

Not all downloaded data files are used by PIPE, and some are optional (e.g. the star catalogue). In particular, none of the DRP generated files are used by PIPE, but can be good for reference, in particular the DRP report. Table 2 lists the data files used by PIPE.

3.2 Running PIPE

The PIPE workflow is controlled by a class called `PipeControl` that accepts a parameter object generated by `PipeParam`. When creating the `PipeParam` object, all parameters relevant for PIPE are set to default values that can then be modified before supplying it to `PipeControl`.

```
from pipe import PipeParam, PipeControl

# Name of target and visit. The data is assumed to be organised as
# data_root/target/visit where the visit directory contains all data
```

```
target = 'TOI-2085'
visit = '3801'

# Generate PIPE parameter file
pps = PipeParam(target, visit)

# Default parameters in pps can be modified, here are some examples:
pps.nthreads = 8     # Number of threads to use; default is
                     # number of virtual CPU cores - 1.
pps.klip = 5         # Number of principal components for PSF fit
pps.fit_bgstars = True       # This enables PSF photometry of
                             # bright stars in the field that
                             # are not the target

# Generate a PIPE control object using the parameter file
pc = PipeControl(pps)

# This is the time-consuming step where PIPE is executed
pc.process_eigen()
```

Execution time depends on many factors but a very rough estimate can be found from the following scaling law:

$$T_{\rm sa} = 10\,{\rm min} \times \left(\frac{10}{n_{\rm threads}}\right)\left(\frac{n_{\rm sa}}{400}\right)\left(\frac{\kappa}{5}\right)\left(\frac{R_{\rm fit}}{40}\right)^2 \tag{1}$$

where $n_{\rm threads}$ is the number of threads assigned, $n_{\rm sa}$ is the number of subarray frames in the visit, κ is the number of principal components used for the PSF fit, and $R_{\rm fit}$ is the fitting radius for the PSF. This scaling relation merely gives the order of magnitude as it depends on many details, including the CPU. A similar scaling relation holds for the imagettes, that typically take more time if they are significantly more numerous. If background stars are additionally to be fitted, the time scales almost linearly with the number of background stars fitted $n_{\rm BG}$ so that $T_{\rm sa} \propto (1 + n_{\rm BG})$.

PIPE is not optimised for memory, as it loads all subarrays and imagettes and also maintains intermediate processing copies in memory. The memory footprint is about 5 GB for 1000 subarray frames, and about twice that if there are imagettes available. For faint stars (defined as having 60 s per exposure), this means $\sim$ 5 GB for a 10-orbit visit. For one of the absolute longest CHEOPS visits, a 4-day visit of the bright HD 172555 ($G = 4.7$ mag) with 4552 subarray frames and 54 624 imagettes, the peak RAM usage of PIPE was observed to be 30 GB, slightly less than the given rule-of-thumb estimate.

Multiprocessing note: Because multiprocessing for python is more of an afterthought than being designed into the language, some peculiarities may surface. One such is related to how sub-processes are spawned that can result in instantiation errors on some systems. This is easily avoided by including this harmless if-statement in your python script before anything else is executed:

```
if __name__ == '__main__':
```

and then continue with the code.

4 PIPE output files

For a given visit, the extracted photometry and other optional diagnostic files are saved in the directory `data_root/target/visit/Output/version/`, where `Output` is a generated sub-directory if not already existing, and `version` is a zero-leading number of five digits. By default, PIPE checks for the lowest available version number not already existing, starting with `00000` and continuing with `00001`, `00002`, etc. for subsequent runs. The version of the run can also be explicitly defined by giving an argument when constructing the `PipeParam` object, as in `pps = PipeParam('TOI-2085', '3801', version=11)`. Output files will then be saved in '`data_root/TOI-2085/3801/Output/00011/`'. Below follows a description of the most common output files. See §A.6 for other optional diagnostic files.

4.1 Logfile

A text file `logfile.txt` is constructed saving all parameters for the run and being updated with processing log information as the the run progresses.

4.2 Extracted photometry

The extracted photometry for subarrays and imagettes (if existing) will be saved in fits table files called `target_visit_sa.fits` and `target_visit_im.fits`, respectively. The header of the fits file inherits keywords

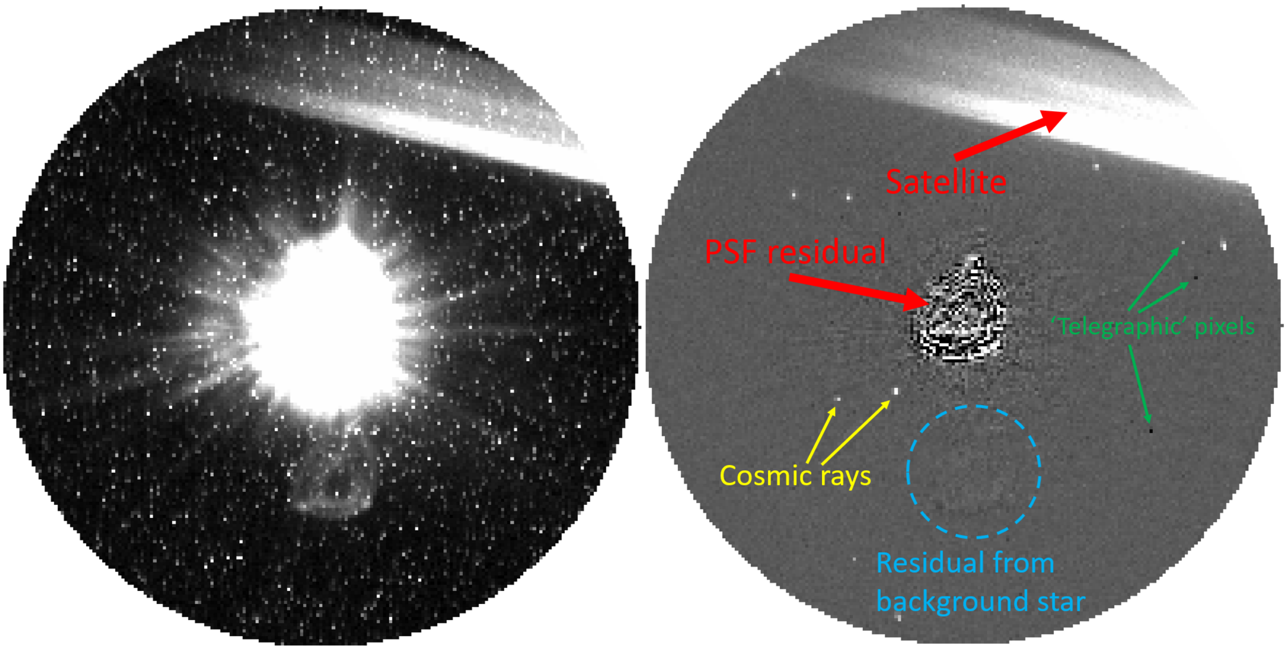

Figure 1: An observed raw subarray frame to the left, and the residuals after reduction and subtraction of target and background to the right. Artefacts in the residuals are annotated in the right image. Satellite tracks are visible in only a few % of the subarray frames. They are not modelled/removed but merely masked from the fit. Only in cases where the satellite passes very close to the target do they significantly affect the quality of the photometry. Telegraphic pixels are pixels with significant and variable dark current. They are normally masked from the PSF fit.

from the corresponding CHEOPS raw data files, and the columns for both files are of the following format:

```
'MJD_TIME' - modified Julian Day number
'BJD_TIME' - barycentric corrected Julian Day number
'FLUX' - in electrons per (co-added) exposure
'FLUXERR' - in electrons per flux value
'BG' - electrons per (co-added) exposure and pixel
'ROLL' - roll angle of field in degrees
'XC' - X-coordinate of target relative to corner of image
'YC'  Y-coordinate of target relative to corner of image
'FLAG' - integer sum of the following indicators:
   0 - no recognised issue with data point
   1 - deviating centroid (far off the median centroid)
   2 - abnormally high level of bad pixels in frame (could be satellite or CR)
   4 - strongly deviating flux in frame (could be strong background or satellite)
   8 - source not found in subarray
'U0' to 'U4' are the relative weights of the first 5 PSF principal components
'thermFront_2' - the value of the thermFront_2 sensor, sometimes useful for de-trending
```

4.3 Residuals

To see how well the PSF fit matched the data, it can be useful to look at the residuals of the fit. For each frame, the fitted PSF is subtracted from the data. If background stars are defined, then those are also subtracted. The resulting residual data cubes are saved in `resduals_sa.fits` and `resduals_im.fits`.

4.4 List of PSF files used

PIPE normally automatically selects suitable PSFs from the PSF library that are closely matched to the target, given information about the position on the detector, the target star effective temperature, and the range of telescope tube temperatures (thermFront_2 sensor readings) during the visit. The PSFs are then analysed using principal component analysis, forming a basis that is used to fit the target PSF in each frame. A list of the filenames of the PSFs used is saved in the text file `psf_filenames.txt`. Apart from being a record, this list can be used as input to PIPE, to ensure that those specific PSFs are selected. This can be useful when e.g. comparing photometry between visits, as different PSFs might derive slightly different absolute levels of the photometry. To use this list as input for a subsequent PIPE run, define the parameter as in

```
pps.psf_filenames_file = 'fullpath/psf_filenames.txt'
```

where `fullpath` is indeed the full path to the location of the directory holding the file.

4.5 Background star photometry

If any background star is selected for PSF photometric extraction, its photometry can be saved in the fits table `bg_star_photometry_sa.fits` (and `bg_star_ photometry_im.fits`). The columns in the table are

```
'MJD_TIME' - modified Julian Day number
'BJD_TIME' - barycentric corrected Julian Day number
'f0' - in electrons per (co-added) exposure for first bg star
'f1' - in electrons per (co-added) exposure for second bg star
...
'fn' - in electrons per (co-added) exposure for n:th bg star
```

The flux columns are ordered in distance from the target so that the nearest extracted background stars is the first column, and so on. The Gaia DR2 ID is listed as a text comment to the corresponding FITS-column. The tabulated flux for the background stars are relative to the visit-median of the target star flux.

5 Making custom PSF models

The existing PSF library is usually sufficient to match the target PSF well enough for good PSF photometry. How well the PSF matches can be checked in the residual output file, where systematic differences from white noise are easily seen. A PSF mis-match can result in PSF noise being added to the extracted photometry, deteriorating the photometric quality. This is particularly relevant for bright sources ($G \ll 8$ mag), where small deviations of the PSF can be the dominant source of noise. In those cases it can be beneficial to produce a PSF model from the data itself. Another instance when new PSF models are useful is if the observations have been centred on a location of the detector not previously archived in the library.

To produce high-quality PSF models, it is preferable to use visits with exposure times so short that imagettes are available. This is to reduce the motion smearing due to jitter that is inevitable during longer exposures. PIPE produces PDF models by fitting a single 2D spline to the normalised PSF of multiple frames coming from a full CHEOPS orbit. Together with the pre-processing that removes background stars this limits the PSF contamination coming from background stars, since the field makes a full rotation during one orbit. Combining data from multiple frames also reduces the effects of noise, outliers (like cosmic rays), and bad pixels (thanks to jitter). To avoid biasing the PSFs it is best to not remove the static image, i.e. set `remove_static = False` (see §A.4). PSF library producing code can look something like:

```
from pipe import PipeParam, PipeControl

# Define run parameters
pps = PipeParam(name='TOI-2085', visit='3801')
pps.remove_static = False

# Process the data before extracting PSF models
pc = PipeControl(pps)
pc.process_eigen()
pc.make_psf_lib()
```

Running `make_psf_lib` is computational intensive. It uses one thread per orbit of the visit (up to the number of available virtual cores), and each thread takes 20–60 min to complete, depending on the number of subarrays/imagettes per orbit and on the CPU. A resulting PSF model for each orbit is put into the `calibpath/psf_lib/XXXxYYY/` directory with the file format `psf_Teff_TF2_MJD_Texp_serial.npy`, where `XXX` and `YYY` give the location on the detector of the PSF in integer pixel coordinates, `Teff` is the effective temperature of the star used, `TF2` is the median (absolute) value of the thermFront_2 sensor during the orbit, `MJD` is themodified Julian date of the observation, `Texp` is the exposure time, and `serial` is just a serial number counting up in case there is already a file of the same name. An example of a PSF model filename that illustrates the format in more detail is

'`calibpath/psf_lib/291x830/psf_05920K_12.20C_59821_11.7_0000.npy`'

meaning that this is a PSF model determined for a visit at MJD = 59821 with a $T_{\rm eff} = 5920$ K star when thermFront_2$= -12.20°$ C and at exposure times of 11.7 s.

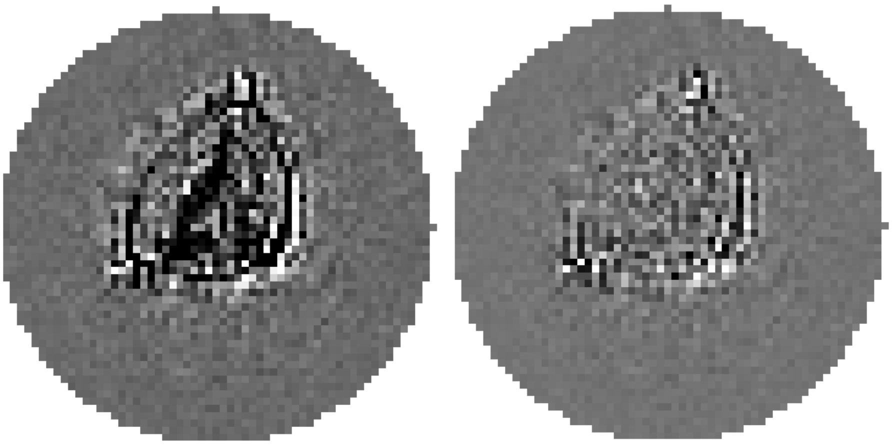

Figure 2: Residuals from the same imagette frame with different PSF models. To the left a poor match with a lot of systematic structure left in the residuals; to the right a good match with mostly white noise. A fits-cube with residual images is saved in the output directory by default.

6 Various tips

Accurate PSF photometry requires accurate PSF modelling. PIPE searches its PSF library for the best PSF matches according to criteria such as detector coordinate of the target, target radiation temperature, and CHEOPS telescope temperature (`thermFront_2` sensor values). The set of chosen PSFs are then analysed for their principal components (PCA) that then serve as a basis for matching the PSFs of individual frames.

One trade off is how many bases to use when matching PSFs. More bases can model more complex PSF behaviour but also become more sensitive to noise. Usually it is therefore better to have few bases for exposures of lower signal to noise, and more bases to model the PSFs of really bright stars. The number of bases are set by the `klip` value. Typically, `klip = 1` is optimal for anything requiring 60 s exposure, while for the brightest and shortest exposures up to `klip = 10` can give better results. Sometimes it is not easy to guess in advance what gives the best results, and in those cases it can make sense to empirically try out different approaches. This can be done using the "`optimise`" option in PIPE, see §A.5.

How well the model PSF matches the data can be best seen in the residual data cubes that are provided in the output directory by default. This residual data cube shows the data subtracted by the model for each frame, organised in a data cube. A successful match leaves only white noise without systematic structures. The presence of systematic patterns are an indication of sub-optimal PSF fits. The PSF match can sometimes be improved by increasing the number of bases used, or change the criteria for selecting model PSFs from the library (e.g. select more or fewer for the PCA).

The PSF tends to evolve during a visit, and this evolution is often well captured by the coefficients for the principal components. These coefficients are saved in the fits-table together with the extracted photometry (§4.2) and can be used to de-trend the light curve from e.g. a ramp.

Another important extraction parameter is the size of the region to be used for fitting the PSF. The PSF fit is noise-weighted, so in theory a larger region should always be better. In practice, larger regions exposes the fit to systematic deviations such as contaminating stars not perfectly taken into account, flat field effects and a less well-determined PSF in the far wings. Fitting regions typically range from radius $R = 25$ pix for the smallest to $R = 60$ pix for the largest.

If a contaminant star is significantly affecting the photometry, an option is to attempt to fit it in the data (rather than just compute its assumed brightness and position from Gaia DR2). For background stars at very small separations (5–30 pix), fitting the star can introduce noise into the extracted photometry of the target since the fits of the target and the background star are not entirely independent but can interfere with each other.

When the background is strong, CHEOPS photometry shows a significant non-linearity that is difficult to get rid during the data reduction. In general it is therefore recommended to mask any exposure with a background of more than $300\,e^-\,\mathrm{pix}^{-1}$, *per exposure*. If a subarray consists of e.g. 10 co-added exposures, this means that the recommended limit is $> 3000\,e^-\,\mathrm{pix}^{-1}$. One can also attempt to correct for the background non-linearity by calibrating for it, i.e. de-trend the flux against background. This requires a long visit (ideally > 5 orbits) with likely limited photometric quality for those data points, but can be warranted if the data points are important enough (e.g. right at the ingress/egress for a time-sensitive measurement),

References

Benz, W., Broeg, C., Fortier, A., et al. 2021, Experimental Astronomy, 51, 109

Hoyer, S., Guterman, P., Demangeon, O., et al. 2020, , 635, A24

Appendices

A PIPE parameters

Here are listed the parameters most relevant for extraction; for a complete listing, see `pipe_param.py` of the source code. In general, the prefix `sa` is for subarray and `im` for imagettes. The default value for each parameter is indicated after the '=' sign. When generating a `PipeParam` object, all parameters except `name` and `visit` are given default values that can subsequently be changed (as shown in the example of §3.2).

A.1 General

Table 3: General PIPE parameters.

PIPE parameter	Function
`datapath`	The default location for the data is defined by the configured `data_root` path, but this can be re-defined by this parameter. The directory with data to be extracted is then expected to be in `datapath/target/name`.
`calibpath`	just like for the `datapath`, the default location for the calibration data is `ref_lib_data` but can be changed with this `calibpath` parameter.
`nthreads`	PIPE is parallelised to make more efficient use of modern CPUs. This parameter defines the number of threads to be used by PIPE. Not all steps uses multiple threads, only the most computationally heavy work loads do. The division of work into threads is usually trivial by giving the threads one frame each to work on at the time. This also means that the number of frames sets the limit to how many threads can efficiently be used, but this is usually > 100 for even the shortest visits so no practical limitation. The default value for `nthreads` is the number of virtual CPU cores on the host machine, minus one. For CPUs with hyperthreading the number of virtual cores is usually twice the number of physical cores, typically ranging from 8 to 64 virtual cores for modern desktops. Workstations and servers can have many more cores than that. The speedup with number of threads is significant, close to linear in the number of threads.
`sa_range = None`	Some visits are very long, and then it can be useful to only work on a subset of the timeline to e.g. try out different extraction parameters. `sa_range` is a tuple of two integers describing the subarray indices range to work on. E.g., if there are 7092 subrray frames in the data set, setting `sa_range = (200,400)` limits the extraction to frames between indices 200 and 400, speeding up the extraction by 35×. Default is `sa_range = None`, in which case the full range is used.
`Teff = None`	Normally the effective temperature from the star is retrieved from the provided data files, but this parameter can be used to override it or simply define it if not available.
`mjd2bjd = True`	This switch encodes whether or not to make a barycentric correction when computing BJD from MJD. A reason to not do it is to work around an incompatibility between Astropy 5.1 or earlier and NumPy 1.24 or later. Normally this parameter should be kept at its default `True` value.

Table 3: General PIPE parameters.

PIPE parameter	Function
`resample_im_times = False`	This switch encodes whether or not to replace the time stamps listed for the imagettes with times derived from the subarrays. The reason for this option is that the imagette timing was (sometimes) off in early phases of the CHEOPS mission, and this was a way to correct for it. This has been fixed since 2021, so this parameter is only relevant for old data files with imagettes and should normally be kept at its default `False` value.

A.2 Data reduction

Table 4: Data reduction parameters.

PIPE parameter	Function
`flatfield = True`	Switch set to `True` if the data is to be corrected for the flat field. Since the flat field is wavelength-dependent, the flat field is weighted by the assumed spectral energy distribution for the target. This flat field was determined before launch, so it is not clear how valid it still is. If set to `False`, not flat field file is required for processing. Also see §2.3.
`darksub = True`	Switch set to `True` if the data is to be corrected for dark current. The dark frame is searched for in the calibration path, and the dark is interpolated over the nearest two dark frames bracketing the visit in time. If there is no dark frame after the visit available, then the latest dark frame is used (with no extrapolation). Also see §2.3.
`dark_min_snr = 15`	The dark current for various pixels is of variable quality. To ensure that dark subtraction does not add too much noise to image, only pixels with dark current determined to a signal to noise of this parameter is included.
`dark_min_level = 3.0`	Only pixels with dark current of at least this level (in e^- per second) are to be used for the dark subtraction. This is to avoid the low-level influence from residual stars that can affect the dark current determination, and to avoid adding noise.
`mask_bad_dark = True`	If `True`, identify pixels with bad dark correction and mask them.
`gain = None`	Gain is normally computed from the house-keeping data, but can be overridden if defined here (in e^- per ADU) to e.g. $1.95\,e^-/\mathrm{ADU}$.
`non_lin = True`	If `True`, apply a non-linear correction to the data.
`cti_corr = True`	As the detector ages, radiation damage produces charge traps that causes a charge transfer inefficiency (CTI). The CTI effect thus increases with time. With this parameter set to `True`, a simplified CTI model (devised by Göran Olofsson) is applied to correct the data.
`cti_t0 = 58800.0`	This is the zero epoch (in MJD) for start of CTI deterioration.
`cti_scale = 0.0016`	This is a CTI scaling factor.
`cti_expo = -0.65`	The CTI exponent.
`cti_lim = 0.0333`	An empiric CTI limit.

A.3 Background correction

Table 5: Background correction parameters.

PIPE parameter	Function

Table 5: Background correction parameters.

PIPE parameter	Function
`bg_fit = 0`	This integer parameter decides if and how the diffuse background (due to e.g. the zodiac or scattered light from the Earth limb) should be estimated. -1 means that the background will not be fitted simultaneously with the PSF (there will still be background correction for the subarray estimated from pixels far outside the target PSF. The background for the imagettes will then be interpolated from the subarray background estimates). **0** means that a constant will be simultaneously fitted with the PSF and assumed to be due to the background. **1** and **2** are supposed to be simultaneously fitted bi-linear and parabolic background fits, but are not yet implemented.
`bgstars = True`	This switch indicates if background stars (retrieved from the `EXT_PRE_StarCatalogue` file) are to be modelled and subtracted from the data cube. Their model brightness use the catalogued Gaia magnitudes directly scaled to the median brightness of the target, so this does not fit the brightness of the background stars; see `fit_bgstars` below for that.
`limflux = 1e-5`	Only stars bright down to this fraction will be modelled for the background. Particularly useful in fields densely populated by very faint stars that do not make a difference but take a very long time to model. The default is `limflux = 1e-5` that means that only stars brighter than 10^{-5} of the target will be taken into account when modelling the star background.
`star_rad_scale = 1.0`	When modelling the background stars, there is a heuristic that determines how far the PSF of each background star should extend, depending on star brightness. The PSF of brighter stars extend much further, up to 100 pixels from the background star, while fainter stars extend much less, down to 25 pixels radius. The default behaviour can be tweaked by changing this parameter, > 1.0 for larger PSFs, and < 1.0 for smaller PSFs. This can be used if it is suspected that a nearby background star contaminates the extracted photometry more than expected.
`fit_bgstars = True`	Normally the background stars are blindly modelled after the Gaia DR2 provided by the `EXT_PRE_StarCatalogue` file, using the tabulated brightness and positions relative to the target (see `bgstars` above). If a background star is significantly variable, however, or if it is bright and closely separated to the target, then it can be beneficial to make a PSF fit also for the background star. Default is `fit_bgstars = True`, i.e. some background stars will be fit rather than merely blindly assigned a brightness.
`lim_fit = 0.01`	It makes sense to fit only the brighter background stars, as attempting to fit stars hidden in the noise may introduce noise into the extraction, so better to use tabulated fluxes in those cases. The `lim_fit` parameter sets the brightness fraction limit where stars brighter than the limit will be fit. Note that fitting background stars significantly prolongs the execution time, as the work required to fit each background star is similar to the work required to fit the target.
`bg_star_inrad = 5.0`	For background stars or stellar companions at very close separation to the target , attempting to fit their brightness may interfere with the fit of the target itself, increasing the systematic noise; in those cases it is often better to just use the tabulated flux from Gaia. This parameter gives a limit (in pixels $\approx$ arcsec) of how close the background star is allowed to be for fitting, even if bright. Defaulted to `bg_star_inrad = 5.0` pixels.
`blur_res = 0.5`	Because the CHEOPS field is rotating with time due to space craft orbital roll, background stars will show rotational motion blur, with longer arcs for stars more separated form the star. To model this behaviour, the PSF for the background is added along the arc at steps defined by this parameter, defaulted to `blur_res = 0.5` pixels.

Table 5: Background correction parameters.

PIPE parameter	Function
`mask_bg_stars = False`	As an alternative to subtract the background stars they can be completely masked. A reason to do this could be that they are suspected variable while fitting being unpractical. Any masked pixels in a frame will not contribute to the PSF extraction of the target.
`mask_bg_star_sep = 30`	Since masking stars too close to the target star will also mask a large fraction of pixels from the target, this gives a minimum separation (in pixels) for background stars to be masked.
`mask_bg_stars_circle = True`	The default is to use a circular mask, but if set to `False` a PSF-shaped mask is used instead, with size determined by `mask_bg_level`.
`mask_bg_radius = 20`	This is the radius of the circular aperture (in pixels).
`mask_bg_level = 0.1`	If the mask is not circular but PSF shaped, then this value is used to determine the mask size. Any pixel of the PSF that is brighter than this parameter (as a fraction of the peak brightness) is within the mask.

A.4 Extraction

Table 6: Photometric extraction parameters.

PIPE parameter	Function
`klip = 5`	This is the number of principal components to use when fitting the target PSF in each frame. The more components, the more closely the observed PSF can be modelled. Too many components results in over fitting, however, introducing more noise to the photometry. To few components is also not optimal, since the observed PSF is not accurately modelled. In general, bright stars are better served with more components (up to 10 for the brightest stars, $G < 7$ mag), while faint stars ($G > 10$ mag) are generally better extracted with a single component.
`sigma_clip = 15`	This is the factor used for clipping bad pixels in a first iteration. Since the noise estimates can be biased at this point, this clipping factor is rather high to avoid masking out signal.
`sigma_iter = 2`	This is the number of iterations to use in the sigma-clipping loop.
`empiric_noise = True`	Instead of using a theoretical noise estimate based on Poisson statistics of the detected signal, background and dark noise, readout noise etc., estimate the empiric noise from the statistics of the residuals of each pixel. Importantly, this also takes care of noise due to PSF mis-match ("PSF noise"), which is difficult to asses by other means.
`empiric_sigma_clip = 4`	This is the sigma-clipping factor when using the more accurate empiric noise to identify and mask outliers.
`centfit_rad = 23`	Find target and fit centroid within this radius.
`centfit_subrad = 3`	Compute flux centroid within this radius in deconvolved image.
`fitrad = 30`	Fit the PSF inside this radius. In principal the photometry should not be sensitive to this parameter since the fit in the PSF wings are increasingly less significant. In practice a larger fit domain can be useful to better constrain the (diffuse) background. This also depends on if there are complications with contaminating stars. It can be worth trying out different radii (between, say, 25 and 60 pixels). This can be automated (see §A.5).
`normrad = 25`	To put PSFs derived from different stars and different circumstances onto a somewhat similar footing, they are normalised by adding up all flux within the radius defined by this parameter.
`centre = True`	Switch to decide whether to compute the centre of the target in each frame. The pointing jitter is typically less than a pixel, so an alternative is to not centring at all, if e.g. there is a problem due to a nearby companion biasing the centring computation.

Table 6: Photometric extraction parameters.

PIPE parameter	**Function**
`centre_psf_filename = None`	Centring is done using a PSF to de-convolve the image and then finding the centre through centre of flux of the de-convolved image. This parameter defines the filename of a specific PSF to use for de-convolution. Normally the PSF is automatically selected to match the star, but this allows to have a fixed PSF to be used for different visits and targets, potentially improving astrometric consistency (since the position is a weak function of PSF). Defaulted to `centre_psf_filename = None` meaning that this PSF is automatically assigned.
`source_window_radius = 30`	The target is assumed to be within this radius from the centre of the frame in order for the frame to not be flagged as source less. This can happen if the pointing is wrong, if there is a very bright transiting satellite, or if there is a deep occultation (in particular from solar system objects, such as Quaoar). Default is `source_window_radius = 30` pixels.
`centre_off_x = -0.710`	This is the typical x-coordinate offset from the exact centre of the frame, used as search window centre but also useful as a default value when not determining the centre. Default `centre_off_x = -0.710` pixels.
`centre_off_y = 1.055`	Same as for x, but in the y-direction. Default `centre_off_y = 1.055` pixels.
`mask_badpix = True`	Set to `True` if bad pixels as defined in the bad pixel map and by sigma-clipping are to be masked.
`mask_level = 2`	The bad pixel maps are not binary but have pixel behaviour encoded. This parameter defines what pixel behaviour should be masked out, with values to the right filter out all before: **0** no pixels are masked, **-2** filters only dead pixels, **-1** filters also half-dead pixels, **3** filters also telegraphic pixels, **2** filters also saturated pixels, **1** filters also hot non-saturated pixels. Default is `mask_level = 2`, i.e. all pixels defined to be bad except regular hot pixels are masked.
`smear_corr = True`	Since there is no shutter on CHEOPS, the detector is exposed during read out. For bright stars this results in vertical streaks in the read-out direction. This switch is set to `True` to correct this smearing from both the target and bright background stars. The smearing is computed from model stars, where all stars in the vertical column of the detector (even outside subarray) are considered.
`smear_resid_sa = False`	For bright stars, the smearing model may not remove the observed smearing completely. This can be because the smearing is determined by the instantaneous position of the star during readout, while the subarray image shows the superposition of the PSF positions during 30–60 seconds. If this parameter is set to True, there is an attempt to remove the residuals by vertical median filtering. The default `smear_resid_sa = False` since this rarely helps the photometric precision in practice.
`smear_resid_im = False`	The same as previous, but for imagettes. Since each imagette typically is of shorter duration than the subarrays, residual smearing is less of a problem. In addition, the small size makes it difficult to properly measure the vertical smearing. Thus it will almost certainly never be well advised to use residual smearing correction on imagettes.
`remove_static = True`	Some dark current and flat field may not be fully corrected for, and this can show up as a static pattern in the frame residuals, i.e. when removing the fitted model av the target and the background stars from the frame. By median filtering over all frames, the static residual is determined. If `remove_static = True`, this static image is then subtracted from the data cube before extraction in a second iteration. Doing this often improves the extracted photometric precision.
`pos_static = False`	Since the static image correction is the median of the residual frames, it can be both positive and negative. By setting this parameter to `True`, the static image is enforced to be non-negative.

A.5 Optimisation

Some parameters affect the quality of extraction while at the same time being difficult to constrain *a priori*. A solution is to extract photometry for a range of different parameters and pick the best. This can be time consuming, so to do this in a systematic way PIPE gives the option to optimise over a range of selected parameters that have proven to be of this nature. They are the number of principal components in fit (`klip`), the radius of the fitting region (`fitrad`), whether or not to simultaneously fit for background (`bg_fit`), correct for dark current (`darksub`), and remove the static image (`remove_static`).

To test all possible combinations of parameters would potentially be extremely time consuming; instead a simple gradient-like search algorithm is implemented. The metric used for the quality of extraction is reduced point-to-point scatter as measured by MAD. It starts with extraction using the nominal values, as defined by the parameters. It then continues to vary each of the parameters as defined by the optimisation parameters below, and picks the best value for each parameter in turn. As an example, assume that the nominal `klip` is 5, and we want to test the values 1, 3, 5, and 10, and similarly the nominal `fitrad` is 30 and we want to test the values 25, 30, 40, 50, 60. then the search will start by varying only `klip` keeping `fitrad` constant at 30, and then continue with keeping `klip` constant at 30 varying `fitrad` instead. The best value for each parameter is then saved as the new nominal to be again varied in the same way. E.g. let us assume `klip = 3` and `fitrad = 50` were found best. Then the full range of `klip` values will be tested for a `fitrad` kept contant at 50, and next `klip` will be kept constant at 3 for the full range of `fitrad` values. All evaluations are saved so that no parameter combination needs to be repeated. This procedure continues until no improvement is found or until `optimise_tree_iter` have been reached. The procedure then restarts with the best combination found so far as the new nominal. This is repeated `optimise_restarts` times.

Some steps like data reduction is identical for extraction with different parameters, so they are not repeated. This results in a shorter total execution time than if PIPE would be run repeatedly from scratch with the different set of parameters.

As the parameter ranges can be different for subarrays and imagettes, they have their own set of optimisation parameters.

Table 7: Optimisation parameters.

PIPE parameter	**Function**
`sa_optimise = False`	if set to `True`, this initiates a search for optimal parameters for subarray extraction.
`im_optimise = False`	if set to `True`, this initiates a search for optimal parameters for imagette extraction.
`optimise_tree_iter = 5`	The number of iterations to use in search before giving up. If no improvement is made by varying parameters, the search is concluded early.
`optimise_restarts = 3`	The number of search restarts made with the best parameter set so far set as nominal value.
`sa_test_klips = [1,3,5,10]`	`klip` values to be tested for subarray extraction.
`sa_test_fitrads = [25,30,40,50,60]`	`fitrad` values to be tested for subarray extraction.
`sa_test_BG = True`	Test to fit background or not?
`sa_test_Dark = True`	Test to correct for dark or not?
`sa_test_Stat = True`	Test to remove static image or not?
`im_test_klips = [1,3,5,10]`	`klip` values to be tested for imagette extraction.
`im_test_fitrads = [25,30]`	`fitrad` values to be tested for imagette extraction (no point in testing radii larger than the imagette).
`im_test_BG = True`	Test to fit background or not?
`im_test_Dark = True`	Test to correct for dark or not?
`im_test_Stat = True`	Test to remove static image or not?

A.6 Save switches

These parameters decide whether to save specific diagnostic output files, typically as fits image cubes or tables in the output directory. The log text file and extracted photometry fits tables are always saved and have no switches.

Table 8: Save switches.

PIPE parameter	**Function**
`save_mask_cube = True`	Saves a binary cube where all pixels in each frame that has been masked is marked.
`save_resid_cube = True`	Saves an image cube of residuals where the models for the target, background stars and their smearing have been subtracted from the reduced data. If defined, the static image is also subtracted.
`save_bg_mask_cube = True`	If a background star mask is defined and this oparameter is `True`, then a binary image cube with all pixels masked from background stars is saved.
`save_bg_cube = False`	Save an image cube with a model for the target star removed, thus keeping all the background stars in the cube.
`save_bg_models = False`	Saves an image cube with the modelled background, including stars, smearing and the static image.
`save_static = False`	Saves the static image.
`save_psfmodel = False`	Saves an image cube with the fitted PSF model for each frame.
`save_psf_list = True`	Saves a text file with a list of the PSF models used in extraction. This file can be used as input to ensure that the same PSF models are used for extraction. This can be useful when extracting photometry from multiple visits and inter-visit photometry is of interest, since it removes the photometric effect from switching PSF models. This, however, may also result in less than optimal PSF models being used for extraction. See §4.4 for more details on how to use it.
`save_psf_pc = False`	This saves an image cube of all derived PSF principal components with one component centred in each frame.
`save_motion_mat = False`	Because the star moves during an exposure, the PSF will be slightly blurred due to jitter. To match the observed PSF, the model PSF is therefore blurred. The motion matrix is a 2D matrix where each side corresponds to a sub-pixel offset in x and y due to motion.
`save_noise_cubes = False`	The noise (std) estimated for every pixel and frame is saved in an image cube.
`save_gain = False`	Save a table with estimated gain (with columns MJD and gain).
`save_bg_star_phot = True`	If photometry is extracted from background stars, save their photometry in a table.