pipeline

February 6, 2024

[ ]:
```
import sys
import os
import numpy as np
np.set_printoptions(threshold=sys.maxsize)
%matplotlib widget
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
from mpl_toolkits.mplot3d import Axes3D
import astropy.constants as c
import astropy.units as u
from astropy.io import fits
from IPython.display import display
import logging
from pathlib import Path
from datetime import datetime
# import pyregion
from astropy.wcs import WCS
from astropy.visualization.wcsaxes import WCSAxes
from astropy.coordinates import SkyCoord
import importlib
import pickle
import pandas as pd
try:
    logging.getLogger('matplotlib').disabled = True
except:
    pass
import pandas as pd
pd.pandas.set_option('display.max_columns', None)
pd.pandas.set_option('display.max_rows', None)
import shutil
from astropy.table import QTable, Table
from matplotlib.patches import (Ellipse, Rectangle)
import matplotlib.patches as mpatches
import itertools
import shapely
import shapely.plotting
```

```
from shapely.geometry.point import Point
from shapely import affinity
from scipy.integrate import dblquad
from uncertainties import ufloat
from scipy.optimize import curve_fit
from astropy.visualization import (
    MinMaxInterval,
    SqrtStretch,
    ImageNormalize,
    simple_norm
)
from astropy import visualization
import matplotlib.cm as cm
from pprint import pprint
import pylustrator
from astropy.stats import sigma_clipped_stats
from astropy.modeling import models
from astropy.convolution.kernels import CustomKernel
from astropy.stats import gaussian_fwhm_to_sigma
from photutils.utils._parameters import as_pair
from astropy.convolution import discretize_model
from pathlib import Path

# with open('imports.py','w') as file:
#     file.write(_ih[-1])
# !pipreqs .
```

[ ]:
```
ROOT_DIR = os.path.dirname(os.path.realpath('__file__'))
PROG_DIR = os.path.abspath('./mirar')
os.chdir(PROG_DIR)
```

[ ]:
```
class Fits:
    def __init__(self, file, print_=True):
        self.file = file
        self.data = {}
        self.header = {}
        open_file = fits.open(self.file)
        if print_:
            open_file.info()
        open_file.close()
        self.read()
        try:
            logging.getLogger("matplotlib").disabled = True
        except:
            pass

    def read(self, hdu=0):
```

```
        open_file = fits.open(self.file)
        try:
            self.data[hdu] = open_file[hdu].data.astype(float)
        except (AttributeError, TypeError, ValueError):
            print(rf"HDU {hdu} is not float")
            self.data[hdu] = open_file[hdu].data

        self.header[hdu] = open_file[hdu].header
        open_file.close()

    def wcs_plot(self, hdu=0):
        wcs = WCS(self.header[hdu])
        fig = plt.figure(clear=True)
        ax = plt.subplot(projection=wcs)
        fig.add_axes(ax)
        return fig, ax

    def create_copy(self):
        shutil.copy2(self.file, f"{self.file}.copy")

    def image(
        self,
        hdu=0,
        column=False,
        title=None,
        scale=[5, 95],
        save=False,
        tag=None,
        wcs=False,
        median=True,
        meanstd=True,
    ):
        data = self.data[hdu]
        if column:
            data = data[column]
        if not wcs:
            fig, ax = plt.subplots()
            ax.set_xlabel("x pixel")
            ax.set_ylabel("y pixel")
        else:
            wcs_ = WCS(self.header[hdu])
            fig = plt.figure(clear=True)
            ax = plt.subplot(projection=wcs_)
            fig.add_axes(ax)
        self.ax = ax
        if not meanstd and scale:
            try:
```

```
                if median:
                    img = np.nanmedian(data, axis=1)
                else:
                    img = data
            except np.AxisError:
                print("invalid shape", data.shape)
                return
            self.scale_low, self.scale_high = np.percentile(img, scale)
            im = ax.imshow(
                data, cmap="magma", vmin=self.scale_low, vmax=self.scale_high
            )
        elif meanstd:
            mean, std = np.nanmean(data), np.nanstd(data)
            vmin = mean - std
            vmax = mean + 10 * std
            im = ax.imshow(
                data,
                interpolation="nearest",
                cmap="grey",
                vmin=vmin,
                vmax=vmax,
                origin="lower",
                # norm=mcolors.Normalize(vmin=vmin,vmax=vmax)
            )
        else:
            im = ax.imshow(data, cmap="magma")
        fig.colorbar(im, ax=ax , pad=0.005)
        if not title:
            title = rf"{os.path.basename(os.path.dirname(self.file))}/{os.path.
↪basename(self.file)}"
        ax.set_title(title)
        if save:
            if tag:
                fig.savefig(
                    os.path.join(
                        os.path.dirname(self.file),
                        rf"{os.path.basename(self.file)}_{tag}.png",
                    ),
                    dpi=600,
                )
            else:
                fig.savefig(
                    os.path.join(
                        os.path.dirname(self.file),
                        rf"{os.path.basename(self.file)}.png",
                    ),
                    dpi=600,
```

```
                )
        fig.tight_layout()
        return ax

    def image_with_reg(
        self, reg, hdu=0, wcsaxis=[0.1, 0.1, 0.8, 0.8], v=[0, 100], save=False,␣
↪tag=None
    ):
        r = pyregion.open(reg).as_imagecoord(self.header[hdu])
        patch_list, artist_list = r.get_mpl_patches_texts()
        wcs = WCS(self.header[hdu])
        fig = plt.figure(clear=True)
        ax = plt.subplot(projection=wcs)
        # ax = WCSAxes(fig,wcsaxis,wcs=wcs)
        fig.add_axes(ax)
        for p in patch_list:
            p.set_color("red")
            p.set_facecolor("none")
            ax.add_patch(p)
        for t in artist_list:
            ax.add_artist(t)

        if v:
            im = ax.imshow(
                self.data[0], origin="lower", vmin=v[0], vmax=v[1], cmap="magma"
            )
        else:
            im = ax.imshow(self.data[0], origin="lower", cmap="magma")
        fig.colorbar(im, cmap="magma")
        if save:
            if tag:
                fig.savefig(
                    os.path.join(
                        os.path.dirname(self.file),
                        rf"{os.path.basename(self.file)}_{tag}.reg.png",
                    ),
                    dpi=600,
                )
            else:
                fig.savefig(
                    os.path.join(
                        os.path.dirname(self.file),
                        rf"{os.path.basename(self.file)}.reg.png",
                    ),
                    dpi=600,
                )
        # fig.show()
```

```
        return ax

    def bin_table(self, hdu=0, return_=True, save=False):
        df = pd.DataFrame(self.data[hdu])
        if save:
            df.to_csv(
                os.path.join(
                    os.path.dirname(self.file), rf"{os.path.basename(self.
↪file)}.csv"
                ),
                encoding="utf-8",
            )
        if return_:
            return df
        else:
            return None

    def bin_table2(self, hdu=0):
        table = QTable(self.data[hdu])
        return table

    def mark_from_cat(self, keys, cat_file=None, cat_table=None, hdu=2,␣
↪save=False, color='red'):
        x_offset = 2
        y_offset = 2
        if cat_file:
            catfits = Fits(cat_file)
            catfits.read(hdu)
            cat = catfits.bin_table2(hdu)
        elif isinstance(cat_table,QTable) or isinstance(cat_table,pd.DataFrame):
            cat = cat_table
        else:
            return
        shape = (self.ax.get_xlim()[-1], self.ax.get_ylim()[-1])
        for i, (x, y, a, b, theta) in enumerate(
            zip(
                cat[keys["x"]],
                cat[keys["y"]],
                cat[keys["a"]],
                cat[keys["b"]],
                cat[keys["angle"]],
            )
        ):
            marker = Ellipse(xy=(x,y), height=a, width=b, angle=theta-90,␣
↪color=color, fill=None) # theta-90 to rotate wrt. x
            self.ax.add_patch(marker)
            annotation = str(i)
```

```
            try:
                if 'NUMBER' in list(cat.columns):
                    annotation = cat['NUMBER'][i]
            except:
                print('could not access column names')
            if x + x_offset >= shape[0] - x_offset * 2:
                self.ax.annotate(
                    annotation, (x - x_offset*2, y + y_offset), color=color
                )
            else:
                self.ax.annotate(annotation, (x + x_offset, y + y_offset),␣
 ↪color=color)
        if save:
            plt.savefig(cat_file+'.png',dpi=300)

SEX_SRC_KEYS = {
    'x': 'X_IMAGE',
    'y': 'Y_IMAGE',
    'a': 'A_IMAGE',
    'b': 'B_IMAGE',
    'angle': 'THETA_IMAGE'
}

MIRAR_SRC_KEYS = {
    'x': 'xpos',
    'y': 'ypos',
    'a': 'aimage',
    'b': 'bimage',
    'angle': 'THETA_IMAGE'
}

PSF_PHOT_SRC_KEYS = {
    'x': 'x_fit',
    'y': 'y_fit',
    'a': 'a',
    'b': 'b',
    'angle': 'angle'
}

class MakeFits:
    def __init__(self, data, filename, obsclass="science", **kwargs):
        self.data = data
        self.filename = filename
        self.obsclass = obsclass

        hdu0 = fits.PrimaryHDU(data)
        self.hdul = fits.HDUList([hdu0])
```

```
        self.hdul.verify("fix")
        self.fix_header()
        self.save(**kwargs)

    def fix_header(self):
        header = self.hdul[0].header
        header[OBSCLASS_KEY] = self.obsclass
        header[TARGET_KEY] = self.obsclass
        header[TIME_KEY] = str(datetime.now())
        header[COADD_KEY] = 1
        header[GAIN_KEY] = 1
        header[PROC_HISTORY_KEY] = ""
        header[PROC_FAIL_KEY] = ""
        header[BASE_NAME_KEY] = Path(self.filename).name

    def save(self, **kwargs):
        self.hdul.writeto(self.filename, **kwargs)


def src_table(file, save=False, return_=True):
    df = pd.read_pickle(file).get_data()
    if save:
        df.to_csv(
            os.path.join(os.path.dirname(file), rf"{os.path.basename(file)}.
 ↪csv"),
            encoding="utf-8",
        )
    if return_:
        return df
    else:
        return None


class Convert:
    def __init__(self, ra, dec):
        self.coords = SkyCoord(ra=ra * u.deg, dec=dec * u.deg)

    def get_coords(self):
        return self.coords

    def to_hms(self):
        return (self.coords.ra.hms, self.coords.dec.hms)

    def to_wcs(self):
        return (self.coords.ra.hms, self.coords.dec.dms)
```

```
def command(config, py=""):
    return rf"python{py} -m mirar -p {PIPELINE} -n {NIGHT} -c {config} -m"


def run(config, *args, **kwargs):
    os.system(command(config, *args, **kwargs))

def get_prams():
    return _ih[-1]

def save_prams(path):
    with open(path,'w') as file:
        file.write(get_prams())

def test(py=""):
    os.system(rf"python{py} -m unittest discover tests/")

def plot_image(ax,data,cmap='grey',scale=True,colorbar=True):
    mean, std = np.nanmean(data), np.nanstd(data)
    if scale:
        vmin = mean - std
        vmax = mean + 10 * std
    else:
        vmin = vmax = None
    im = ax.imshow(
        data,
        interpolation="nearest",
        cmap=cmap,
        vmin=vmin,
        vmax=vmax,
        origin="lower",
    )
    if colorbar:
        ax.get_figure().colorbar(im, ax=ax , pad=0.005)
    return im

def peek(ax,coords,size):
    ax.set_xlim([coords[0]-size//2,coords[0]+size//2])
    ax.set_ylim([coords[1]-size//2,coords[1]+size//2])

def get_pos(table,obj,keys):
    row = table[obj-1]
    x = row[keys['x']]
    y = row[keys['y']]
    return np.array([float(x),float(y)])
```

```
def mark_from_cat(ax, keys, cat_file=None, cat_table=None, hdu=2, save=False,␣
 ↪color='red', condition=True,bound=False,bounds=None,annotate=True):
    x_offset = 2
    y_offset = 2
    if cat_file:
        catfits = Fits(cat_file)
        catfits.read(hdu)
        cat = catfits.bin_table2(hdu)
    elif isinstance(cat_table,QTable) or isinstance(cat_table,pd.DataFrame):
        cat = cat_table
    else:
        return
    shape = (ax.get_xlim()[-1], ax.get_ylim()[-1])

    pos_all = []
    for i, (x, y, a, b, theta) in enumerate(
        zip(
            cat[keys["x"]],
            cat[keys["y"]],
            cat[keys["a"]],
            cat[keys["b"]],
            cat[keys["angle"]],
        )
    ):
        if condition:
            if not bound or bound and (x>=bounds[0][0] and x<=bounds[0][1] and␣
 ↪y>=bounds[1][0] and y<=bounds[1][1]):
                marker = Ellipse(xy=(x,y), height=a, width=b, angle=theta-90,␣
 ↪color=color, fill=None) # theta-90 to rotate wrt. x
                ax.add_patch(marker)
                annotation = str(i)
                if annotate:
                    try:
                        if 'NUMBER' in list(cat.columns):
                            annotation = cat['NUMBER'][i]
                    except:
                        print('could not access column names')
                    if x + x_offset >= shape[0] - x_offset * 2:
                        ax.annotate(
                            annotation, (x - x_offset*2, y + y_offset),␣
 ↪color=color
                        )
                    else:
                        ax.annotate(annotation, (x + x_offset, y + y_offset),␣
 ↪color=color)
                pos_all.append([x,y])
    return pos_all
```

```
def none_mask(lst):
    for i in range(len(lst)):
        if lst[i] == None:
            lst[i] = np.nan
    return lst
```

[ ]:
```
def test_path(name):
    if isinstance(name,tuple):
        return Path(os.path.abspath(os.path.join(TEST_DIR,*name)))
    else:
        return Path(os.path.abspath(os.path.join(TEST_DIR,name)))

def make_test_dirs():
    for dir in OUTPUT_DIRS.values():
        os.makedirs(dir, exist_ok=True)

def move_to_raw(path):
    path = Path(path)
    dst = os.path.join(RAW_DIR,path.name)
    shutil.copyfile(path,dst)

def prepare(src_dir,src_basenames):
    for file in os.listdir(src_dir):
        for basename in src_basenames:
            if basename in file:
                move_to_raw(os.path.join(src_dir,file))

def save_params(path):
    with open(path,'w') as file:
        file.write(_ih[-1])

DATA_DIR = os.path.join(ROOT_DIR,'SampleData')

TEST_ID = '0'
TEST_DIR = os.path.abspath(os.path.join(ROOT_DIR,'test_'+TEST_ID))
OUTPUT_DIRS = {
    'BKG': test_path('background'),
    'DET': test_path('detection'),
    'PSF_MODEL': test_path('psf_model'),
    'PHOTCAL': test_path('photcal'),
    'PSF_PHOT': test_path('psf_phot'),
    'APER_PHOT': test_path('aper_phot'),
    'LOG': test_path('log'),
    'CONF': test_path('config'),
    'RES': test_path('results'),
```

```
    'CAT': test_path('catalogs'),
}
make_test_dirs()

RAW_DIR = os.path.join(TEST_DIR,'raw')
os.makedirs(RAW_DIR, exist_ok=True)

LOGGING = True
LOG_FILE = os.path.abspath(os.path.
 ↪join(OUTPUT_DIRS['LOG'],f"test{TEST_ID}_{datetime.now()}"))
if LOGGING:
    logger = logging.getLogger(__name__)
    logging.basicConfig(filename=LOG_FILE,level=logging.DEBUG)
    logging.getLogger('matplotlib').disabled = True

ENV = {
    'RAW_DATA_DIR': RAW_DIR,
    'OUTPUT_DATA_DIR': TEST_DIR,
    'REF_IMG_DIR': '',
    'USE_WINTER_CACHE': 'false',
    'FRITZ_TOKEN': 'test',
    'KOWALSKI_TOKEN': 'test',
    'DB_USER': 'postgres',
    'DB_PWD': '',
}
for var in ENV.keys():
    os.environ[var] = ENV[var]
```

[ ]:
```
# mirar
from mirar.pipelines.wifes_autoguider.wifes_autoguider_pipeline import␣
 ↪WifesAutoguiderPipeline
from mirar.processors.astromatic.sextractor.background_subtractor import (
    SextractorBkgSubtractor,
)
from mirar.processors.astromatic.sextractor.sextractor import Sextractor
from mirar.processors.utils import (
    CustomImageBatchModifier,
    HeaderAnnotator,
    ImageBatcher,
    ImageDebatcher,
    ImageLoader,
    ImageSaver,
    ImageSelector,
    MEFLoader,
)
from mirar.processors.utils.header_annotate import (
    HeaderEditor,
```

```
    # SextractorHeaderCorrector,
)
from mirar.data import (
    Image,
    Dataset,
    ImageBatch,
    SourceBatch,
    SourceTable
)
from mirar.io import open_raw_image
from mirar.paths import (
    BASE_NAME_KEY,
    COADD_KEY,
    GAIN_KEY,
    LATEST_SAVE_KEY,
    LATEST_WEIGHT_SAVE_KEY,
    OBSCLASS_KEY,
    PROC_FAIL_KEY,
    PROC_HISTORY_KEY,
    RAW_IMG_KEY,
    SATURATE_KEY,
    TARGET_KEY,
    TIME_KEY,
    DIFF_IMG_KEY,
    REF_IMG_KEY,
    SCI_IMG_KEY,
    XPOS_KEY,
    YPOS_KEY,
    NORM_PSFEX_KEY,
    core_fields,
    get_output_dir
)
from mirar.processors.astromatic import PSFex, Scamp
# from mirar.processors.photometry.psf_photometry import SourcePSFPhotometry
# from mirar.processors.photometry.aperture_photometry import␣
 ↪SourceAperturePhotometry
from mirar.processors.sources import (
    SourceWriter
)
# from mirar.processors.sources.source_detector import (
#     SourceGenerator
# )
from mirar.processors.base_processor import (
    BaseImageProcessor,
    BaseSourceProcessor,
    BaseSourceGenerator,
)
```

```
# from mirar.processors.photometry.base_photometry import (
#     BaseSourcePhotometry,
# )
from mirar.utils.pipeline_visualisation import flowify
from mirar.io import (
    open_fits,
    save_to_path
)
from mirar.processors.base_processor import PrerequisiteError
from mirar.processors.utils.image_selector import select_from_images
from mirar.processors.photcal import PhotCalibrator
from mirar.utils.ldac_tools import (
    save_table_as_ldac,
    get_table_from_ldac
)
from mirar.processors.astromatic.sextractor.sextractor import␣
 ↪SEXTRACTOR_HEADER_KEY
from mirar.catalog.base_catalog import CatalogFromFile
from mirar.catalog.vizier.gaia import Gaia
from mirar.catalog.vizier.base_vizier_catalog import VizierCatalog
from mirar.processors.astromatic.sextractor.sextractor import␣
 ↪sextractor_checkimg_map
from mirar.processors.photometry.utils import get_mags_from_fluxes

# photutils
from photutils.background import Background2D
from photutils.detection import DAOStarFinder
from photutils.psf import extract_stars
from photutils.psf import EPSFBuilder

import astropy
from photutils.background.interpolators import BkgZoomInterpolator
from photutils.background import Background2D
from photutils.background.core import (
    SExtractorBackground,
    StdBackgroundRMS
)
from astropy.nddata import NDData
from astropy.stats import SigmaClip
from typing import Callable
from astropy.convolution import convolve
from astropy.convolution import (
    Kernel,
    CustomKernel,
)
from photutils.segmentation import (
    make_2dgaussian_kernel,
```

```
    SourceFinder,
    SourceCatalog
)
from astropy.wcs import WCS
```

[ ]:
```
ACQ_KEY = 'acq'
BGMED_KEY = 'BGMED'
BGRMSMED_KEY = 'BGRMSMED'
# SEGMPATH_KEY = 'SEGMPATH'
SEGMPATH_KEY = sextractor_checkimg_map['SEGMENTATION']
CONVPATH_KEY = 'CONVPATH'
# BGPATH_KEY = 'BGPATH'
BGPATH_KEY = sextractor_checkimg_map['BACKGROUND']
BGRMSPATH_KEY = sextractor_checkimg_map['BACKGROUND_RMS']
SEGMOBJ_KEY = 'SEGMOBJ'
PSF_CUTOUTS_PATH_KEY = 'PCUTPATH'
PSF_CUTOUTS_SIZE_KEY = 'PCUTSIZE'
NPSFPATH_KEY = 'NPSFPATH'
PIXSCALE_KEY = 'PIXSCALE'

sex_all_ground = CustomKernel(np.array([
    [1,2,1],
    [2,4,2],
    [1,2,1]
]))

def default_select_acquisition(
    images: ImageBatch,
) -> ImageBatch:
    """
    Returns images in a batch with are tagged as error

    :param images: set of images
    :return: subset of bias images
    """
    return select_from_images(images, key=OBSCLASS_KEY, target_values=ACQ_KEY)

def load_object(path):
    with open(path,'rb') as file:
        return pickle.load(file)

def dump_object(data,path):
    with open(path,'wb') as file:
        return pickle.dump(data,file)

class WifesAutoguiderVisier:
```

```
    def __init__(
        self,
        visier_catalog: VizierCatalog,
        cache: bool = False,
    ):
        self.visier_catalog = visier_catalog
        self.cache = cache

    def generator(
        self,
        image: Image,
    ) -> VizierCatalog | CatalogFromFile:

        logger.debug(image)
        filter_name = image["FILTER"]

        search_radius_arcmin = (
            np.max([image["NAXIS1"], image["NAXIS2"]])
            * np.max([np.abs(image["CD1_1"]), np.abs(image["CD1_2"])])
            * 60
        ) / 2.0

        # TODO: match closest
        if filter_name == 'I':
            filter_name = 'RP'

        return self.visier_catalog(
            min_mag=10,
            max_mag=20,
            search_radius_arcmin=search_radius_arcmin,
            filter_name=filter_name,
            cache_catalog_locally=self.cache,
        )

def wifes_autoguider_photometric_catalog_purifier(
    catalog: Table,
    image: Image
) -> Table:
    logger.debug('Using filter: wifes_autoguider_photometric_catalog_purifier')
    logger.debug(catalog)
    # TODO: filter
    # clean_mask = np.ones(catalog.to_pandas().shape,dtype=bool)
    # return catalog[clean_mask]


    return catalog
```

```
class PhotutilsBkgSubtractor(BaseImageProcessor):

    base_key = "photutilsbkgsubtractor"

    def __init__(
        self,
        box_size = 40,
        mask=None,
        coverage_mask=None,
        fill_value=0.0,
        exclude_percentile=10.0,
        filter_size=(3, 3),
        filter_threshold=None,
        edge_method='pad',
        sigma_clip=SigmaClip(
            sigma=3.0,
            sigma_lower=3.0,
            sigma_upper=3.0,
            maxiters=10,
            cenfunc='median',
            stdfunc='std',
            grow=False
        ),
        bkg_estimator=SExtractorBackground(sigma_clip=None),
        bkgrms_estimator=StdBackgroundRMS(sigma_clip=None),
        interpolator=BkgZoomInterpolator(),
        output_sub_dir = 'background',
        select_images: Callable[[ImageBatch], ImageBatch] =␣
 ↪default_select_acquisition, #change
        dev: bool = False,
        cache: bool = False,
        save_bkg: bool = True,
        save_bkg_rms: bool = True,
    ):
        super().__init__()
        self.box_size = box_size
        self.mask = mask
        self.coverage_mask = coverage_mask
        self.fill_value = fill_value
        self.exclude_percentile = exclude_percentile
        self.filter_size = filter_size
        self.filter_threshold = filter_threshold
        self.edge_method = edge_method
        self.sigma_clip = sigma_clip
        self.bkg_estimator = bkg_estimator
        self.bkgrms_estimator = bkgrms_estimator
        self.interpolator = interpolator
```

```
        self.cache = cache
        self.output_sub_dir = output_sub_dir
        self.dev = dev
        self.select_images = select_images
        self.save_bkg = save_bkg
        self.save_bkg_rms = save_bkg_rms

    def _apply_to_images(
        self,
        batch: ImageBatch,
    ) -> ImageBatch:

        images = self.select_images(batch)

        for image in images:
            data = image.get_data()
            header = image.get_header()
            background = Background2D(
                data=data,
                box_size=self.box_size,
                mask = self.mask,
                coverage_mask = self.coverage_mask,
                fill_value = self.fill_value,
                exclude_percentile = self.exclude_percentile,
                filter_size = self.filter_size,
                filter_threshold = self.filter_threshold,
                edge_method = self.edge_method,
                sigma_clip = self.sigma_clip,
                bkg_estimator = self.bkg_estimator,
                bkgrms_estimator = self.bkgrms_estimator,
                interpolator = self.interpolator
            )
            background_map = background.background

            header[BGMED_KEY] = background.background_median
            header[BGRMSMED_KEY] = background.background_rms_median

            bkgsub = data - background_map
            image.set_data(bkgsub)

            save_images = {}
            output_dir = get_output_dir(self.output_sub_dir, self.night_sub_dir)

            if self.save_bkg:
                save_images[BGPATH_KEY] = 'background'

            if self.save_bkg_rms:
```

```
                save_images[BGRMSPATH_KEY] = 'background_rms'

            # TODO:
            if self.cache:
                save_images += ['background']
                # bkg_image_name = image[BASE_NAME_KEY].
↪replace('fits','background.fits')
                bkg_image_name = image[BASE_NAME_KEY]+'.background'
                header[BGPATH_KEY] = str(output_dir.joinpath(bkg_image_name))

            if self.dev:
                save_images[sextractor_checkimg_map['MINIBACKGROUND']] =␣
↪'background_mesh'
                save_images[sextractor_checkimg_map['MINIBACK_RMS']] =␣
↪'background_rms_mesh'
                save_name = image[BASE_NAME_KEY].replace('fits','background.
↪pkl')
                dump_object(
                    data=background,
                    path=output_dir.joinpath(save_name)
                )

            for im in save_images.keys():
                # save_name = image[BASE_NAME_KEY].replace('fits',im+'.fits')
                save_name = image[BASE_NAME_KEY]+f".{save_images[im]}"
                save_path = output_dir.joinpath(save_name)
                save_to_path(
                    data=eval('background.'+save_images[im]),
                    header=image.header,
                    path=save_path,
                    overwrite=True
                )
                image[im] = str(save_path)

            image.set_header(header)

        return batch

class PhotutilsSourceFinder(BaseImageProcessor):
    """
    Processor to detect sources using photutils.segmentation.SourceFinder

    Args
        convolution_fwhm: FWHM of convolution mask
        convolution_kernel_size: size of convolution kernel
        npixels: number of connected pixels for detection
        threshold_factor: threshold factor of background RMS median
```

```
        connectivity: {4,8} source pixel grouping
        deblend: Whether to deblend overlapping sources.
        nlevels: The number of multi-thresholding levels to use for deblending.
        contrast: The fraction of the total source flux that a local peak must␣
↪have
            (at any one of the multi-thresholds) to be deblended as a separate␣
↪object.
        mode: The mode used in defining the spacing between the␣
↪multi-thresholding levels.
        relabel: If True (default), then the segmentation image will be␣
↪relabeled after deblending
        nproc: The number of processes to use for multiprocessing (deblending)
        progress_bar: Whether to display a progress bar.

    Returns
        ImageBatch
    """

    base_key = 'photutilssourcedetection'

    def __init__(
        self,
        output_sub_dir: Path | str = 'detection',
        convolve: bool = False,
        convolution_kernel: Kernel | None = None,
        convolution_fwhm: float | None = 2,
        convolution_kernel_size: int | None = 3,
        npixels: int = 10, # default from SE config
        threshold_factor: float = 1.5,
        connectivity: int = 8, # default from SE config
        deblend: bool = True,
        nlevels: int = 32, # default from SE config
        contrast: float = 0.001,
        mode: str = 'exponential', # default from SE config
        relabel: bool = True,
        nproc: int = 1,
        progress_bar: bool = False,
        dev: bool = False,
        cache: bool = False,
    ):
        super().__init__()
        self.cache = cache
        self.output_sub_dir = output_sub_dir
        self.npixels = npixels
        self.threshold_factor = threshold_factor
        self.connectivity = connectivity
        self.convolution_fwhm = convolution_fwhm
```

```
        self.convolution_kernel_size = convolution_kernel_size
        self.deblend = deblend
        self.contrast = contrast
        self.nlevels = nlevels
        self.mode = mode
        self.relabel = relabel
        self.nproc = nproc
        self.progress_bar = progress_bar
        self.dev = dev
        self.convolve = convolve
        self.convolution_kernel = convolution_kernel

    def _apply_to_images(
        self,
        batch: ImageBatch,
    ) -> ImageBatch:

        for image in batch:
            data = image.get_data()
            header = image.get_header()

            # convolve the data
            if self.convolve:
                if self.convolution_kernel is not None:
                    kernel = self.convolution_kernel
                else:
                    kernel = make_2dgaussian_kernel(
                        self.convolution_fwhm,
                        size=self.convolution_kernel_size
                    )
                convolved_data = convolve(data, kernel)
            else:
                convolved_data = data

            # detect the sources
            if BGRMSMED_KEY not in header.keys():
                raise PrerequisiteError(
                    f"{BGRMSMED_KEY} key not found in image. "
                    f"PhotutilsBkgSubtractor must be run before running this␣
↪processor"
                )
            threshold = self.threshold_factor * image[BGRMSMED_KEY]  #␣
↪per-pixel threshold
            finder = SourceFinder(
                npixels=self.npixels,
                connectivity=self.connectivity,
                deblend = self.deblend,
```

```
                nlevels = self.nlevels,
                contrast = self.contrast,
                mode = self.mode,
                relabel = self.relabel,
                nproc=self.nproc,
                progress_bar=self.progress_bar,
            )
            segm = finder(convolved_data, threshold)

            if segm is None:
                # TODO: add logger message
                continue

            output_dir = get_output_dir(self.output_sub_dir, self.night_sub_dir)
            save_name = image[BASE_NAME_KEY]+'.segm'
            save_path = output_dir.joinpath(save_name)
            save_path_obj = str(save_path)+'.pkl' # maybe a different way?

            header[SEGMOBJ_KEY] = str(save_path_obj)
            header[SEGMPATH_KEY] = str(save_path)

            if self.convolve:
                save_path_conv = str(save_path).replace('segm','conv')
                header[CONVPATH_KEY] = str(save_path_conv)
            else:
                header[CONVPATH_KEY] = str(None)

            with open(save_path_obj, 'wb') as file:
                pickle.dump(segm, file) # better format?
            save_to_path(         # maybe move to dev as need only object not␣
↪image
                data=segm.data,
                header=header,
                path=save_path,
                overwrite=True
            )
            save_to_path(
                data=convolved_data,
                header=header,
                path=save_path_conv,
                overwrite=True
            )

            if self.dev:
                params = ['cmap','polygons','segments','areas']
                for param in params:
                    with open(f"{save_path}.{param}.pkl", 'wb') as file:
```

```
                        pickle.dump(eval(f"segm.{param}"),file)

            image.set_header(header)

        return batch

#TODO: multiple aperture
class PhotutilsSourceCatalog(BaseSourceGenerator):
    """
    Args
        localbkg_width: The width of the rectangular annulus used to
            compute a local background around each source.
        detection_cat: A SourceCatalog object for the detection image.
        make_cutouts: Whether to make cutouts for psf modeling
        cutout_size: size of cutout (if make_cutouts)

    Returns

    """

    base_key = "photutilssourcecatalog"

    def __init__(
        self,
        calc_total_error: bool = False,
        error = None, # TODO: [somewhat done] maybe␣
 ↪PhotutilsTotalErrorCalculator or calc_total_error internally
        mask = None,
        wcs = None,
        localbkg_width = 15, # default: 0, mirar: 15
        background = None,
        use_background = False,
        apermask_method = 'correct', # default from SE config
        kron_params = [2.5, 3.5], # default from SE config
        detection_cat = None,
        progress_bar: bool = False,
        make_psf_cutouts: bool = True,
        psf_cutout_size: int = 21,
        output_sub_dir: str = "detection",
        copy_image_keywords: str | list[str] = None,
        cache: bool = False,
    ):
        super().__init__()
        self.output_sub_dir = output_sub_dir
        self.copy_image_keywords = copy_image_keywords
        if isinstance(copy_image_keywords, str):
            self.copy_image_keywords = [self.copy_image_keywords]
```

```
        self.cache=cache
        self.error = error
        self.mask = mask
        self.wcs = wcs
        self.localbkg_width = localbkg_width
        self.apermask_method = apermask_method
        self.kron_params = kron_params
        self.detection_cat = detection_cat
        self.progress_bar = progress_bar
        self.make_psf_cutouts = make_psf_cutouts
        self.psf_cutout_size = psf_cutout_size
        self.calc_total_error = calc_total_error
        self.background = background
        self.use_background = use_background

    def _apply_to_images(
        self,
        batch: ImageBatch,
    ) -> SourceBatch:

        src_batch = SourceBatch()

        for image in batch:

            data = image.get_data()
            header = image.get_header()

            # TODO: check pre-rec or as a processor function
            with open(header[SEGMOBJ_KEY],'rb') as file:
                segment_img = pickle.load(file)

            if self.calc_total_error:
                if LATEST_WEIGHT_SAVE_KEY in header:
                    error = fits.getdata(image[LATEST_WEIGHT_SAVE_KEY])
                else:
                    error = None
            else:
                error = self.error

            if self.use_background:
                if self.background is not None:
                    background = fits.getdata(header[BGPATH_KEY]) # [!imp]␣
↪restimate?
                else:
                    background = self.background
            else:
                background = None
```

```
        if self.wcs is None:
            wcs = WCS(header=header)
        else:
            wcs = self.wcs

        srccat = SourceCatalog(
            data=data,
            segment_img=segment_img,
            convolved_data=fits.getdata(header[CONVPATH_KEY]),
            error=error,
            mask=self.mask,
            background=background,
            wcs=wcs,
            localbkg_width=self.localbkg_width,
            apermask_method=self.apermask_method,
            kron_params = self.kron_params,
            detection_cat = self.detection_cat,
            progress_bar = self.progress_bar,
        )
        srccat_table = srccat.to_table()

        pix_scale = np.sqrt(np.abs(np.linalg.det(wcs.pixel_scale_matrix)))
        # TODO: maybe rename col instead of duplicate
        #       or make a table from scratch with only necessary columns
        srccat_table['NUMBER'] = srccat.label
        srccat_table['fwhm'] = srccat.fwhm
        srccat_table['ellipticity'] = srccat.ellipticity
        srccat_table['elong'] = srccat.elongation
        srccat_table[XPOS_KEY] = srccat.xcentroid
        srccat_table[YPOS_KEY] = srccat.ycentroid
        srccat_table['xcentroid_win'] = srccat.xcentroid_win
        srccat_table['ycentroid_win'] = srccat.ycentroid_win
        srccat_table['sky_centroid_icrs'] = srccat.sky_centroid_icrs
        srccat_table['sky_centroid_win'] = srccat.sky_centroid_win
        # logger.debug(srccat.sky_centroid_win)
        # cov_eigvals = srccat.covariance_eigvals
        # srccat_table['cov_eigvals'] = cov_eigvals
        # srccat_table['aimage'] = cov_eigvals[0]
        # srccat_table['bimage'] = cov_eigvals[1]
        srccat_table['aimage'] = srccat.semimajor_sigma
        srccat_table['bimage'] = srccat.semiminor_sigma
        srccat_table['THETA_IMAGE'] = srccat.orientation
        srccat_table['kron_aperture'] = srccat.kron_aperture

        # sex compatability
```

```
            srccat_table['ALPHAWIN_J2000'] = [coords.ra for coords in srccat.
↪sky_centroid_win]
            srccat_table['DELTAWIN_J2000'] = [coords.dec for coords in srccat.
↪sky_centroid_win]
            srccat_table['X_IMAGE'] = srccat.xcentroid_win
            srccat_table['Y_IMAGE'] = srccat.ycentroid_win
            srccat_table['FWHM_IMAGE'] = srccat.fwhm
            srccat_table['FWHM_WORLD'] = srccat.fwhm * pix_scale

            mag, mag_unc = get_mags_from_fluxes(
                flux_list = srccat.kron_flux,
                fluxunc_list = np.zeros(len(srccat_table)),
                zeropoint = 0.0,
                zeropoint_unc = 0.0,
            )
            srccat_table['MAG_AUTO'] = np.array(mag,dtype=float)

            # TODO: compute this somehow
            aper_mags = {

            }

            if len(aper_mags) > 0:
                aper_fluxes = np.array(list(aper_mags.values()))
                mag, mag_unc = get_mags_from_fluxes(
                    flux_list = aper_fluxes,
                    fluxunc_list = np.zeros(aper_fluxes.shape),
                    zeropoint = 0.0,
                    zeropoint_unc = 0.0,
                )
                srccat_table['MAG_APER'] = np.array(mag,dtype=float)

            # TODO: CHANGE!!
            srccat_table['FLAGS'] = 0

            # mirar compatability
            srccat_table[DIFF_IMG_KEY] = image[LATEST_SAVE_KEY] # has to be
            srccat_table[SCI_IMG_KEY] = '' #image[LATEST_SAVE_KEY]
            srccat_table[REF_IMG_KEY] = ''
            srccat_table[PIXSCALE_KEY] = pix_scale

            if self.make_psf_cutouts:
                # TODO: check if/what different for dithers with wcs

                # twiddle -->
                twiddle_keys = {
                    'label': 'id',
```

```
                    'xcentroid': 'x',
                    'ycentroid': 'y'
                }
                for key in twiddle_keys.keys():
                    srccat_table.rename_column(key,twiddle_keys[key])

                stars = extract_stars(
                    data=NDData(data=data),
                    catalogs=srccat_table[*twiddle_keys.values()],
                    size=self.psf_cutout_size
                )

                # twiddle <--
                for key in twiddle_keys.keys():
                    srccat_table.rename_column(twiddle_keys[key],key)

                save_name = image[BASE_NAME_KEY]+'.cutouts.pkl'
                output_dir = get_output_dir(self.output_sub_dir, self.
↪night_sub_dir)
                save_path = output_dir.joinpath(save_name)
                with open(save_path,'wb') as file:
                    pickle.dump(stars,file)

                header[PSF_CUTOUTS_PATH_KEY] = str(save_path)
                header[PSF_CUTOUTS_SIZE_KEY] = self.psf_cutout_size

            output_dir = get_output_dir(self.output_sub_dir, self.night_sub_dir)
            output_cat = output_dir.joinpath(
                image[BASE_NAME_KEY].replace(".fits", ".cat")
            )

            header[SEXTRACTOR_HEADER_KEY] = str(output_cat)
            # header[CALSTEPS] += Sextractor.base_key
            save_table_as_ldac(
                tbl = srccat_table,
                file_path = output_cat
            )

            image.set_header(header)

            metadata = {}
            for key in image.keys():
                if key != "COMMENT":
                    metadata[key] = image[key]

            if len(aper_mags) > 0:
                srccat_table['MAG_APER'] = Table(aper_mags)
```

```
            src_batch.append(SourceTable(srccat_table.
 ↪to_pandas(),metadata=metadata))

        return src_batch

class SourcePhotCalibrator(PhotCalibrator):

    def __init__(
        self,
        *args,
        **kwargs
    ):
        super().__init__(*args,**kwargs)

    def get_sextractor_apertures(self) -> list[float]:
        # TODO: do it!
        return []

    def check_prerequisites(self):
        return True

    def table_to_fake_image(
        self,
        table
    ):
        return Image(data=np.zeros([1,1]),header=table.get_metadata())

    def _apply_to_images(
        self,
        batch: SourceBatch,
    ) -> SourceBatch:

        batch_updated = SourceBatch()

        batch_fake_images = ImageBatch([self.table_to_fake_image(table) for␣
 ↪table in batch])
        batch_photcal = super()._apply_to_images(batch_fake_images)

        for table, table_updated in zip(batch,batch_photcal):
            table.metadata = table_updated.get_header()
            batch_updated.append(table)

        return batch_updated

class SeeingCalculator(BaseSourceProcessor):

    def __init__(
```

```
        self,
    ):
        super().__init__()

    def _apply_to_sources(
        self,
        batch: SourceBatch,
    ) -> SourceBatch:
        raise NotImplementedError

def load_wifes_guider_fits(
    path: str | Path
) -> tuple[np.array, astropy.io.fits.Header]:
    data, header = open_fits(path)
    header[OBSCLASS_KEY] = ACQ_KEY
    header[TARGET_KEY] = header['OBJECT']
    header[COADD_KEY] = 1
    header[GAIN_KEY] = 1
    header['CALSTEPS'] = ''
    header[PROC_FAIL_KEY] = ''
    if 'RADECSYS' in header:
        sys = header.pop('RADECSYS')
        header['RADESYSa'] = sys
        header['RADECSYS'] = sys
    if 'FILTER' not in header:
        header['FILTER'] = header['TVFILT']
    return data, header

def load_wifes_guider_image(path: str | Path) -> Image:
    return open_raw_image(path, load_wifes_guider_fits)

load = [
    ImageLoader(input_sub_dir=RAW_DIR, input_img_dir=TEST_DIR,␣
 ↪load_image=load_wifes_guider_image)
]

bkg_sub = [
    PhotutilsBkgSubtractor(
        box_size=(10,10),
        select_images=default_select_acquisition,
        output_sub_dir=OUTPUT_DIRS['BKG'],
        dev=False,
        save_bkg=False,
    ),
    ImageSaver(output_dir_name=OUTPUT_DIRS['BKG'])
]
```

```
src_det = [
    PhotutilsSourceFinder(
        convolve=True,
        convolution_kernel=sex_all_ground,
        output_sub_dir=OUTPUT_DIRS['DET'],
        dev=True
    ),
    PhotutilsSourceCatalog(
        make_psf_cutouts=False,
        use_background=False,
        output_sub_dir=OUTPUT_DIRS['DET']
    ),
    SourceWriter(output_dir_name=OUTPUT_DIRS['DET'])
]

photcal = [
    SourcePhotCalibrator(
        ref_catalog_generator=WifesAutoguiderVisier(Gaia).generator,
        temp_output_sub_dir="phot",
        crossmatch_radius_arcsec=3.0, # or 2 TODO: test
        write_regions=True,
        cache=True,
        outlier_rejection_threshold=[1.5, 2.0, 3.0],
        ␣
  ↪image_photometric_catalog_purifier=wifes_autoguider_photometric_catalog_purifier,
        num_matches_threshold=0, # TODO: Change! (testing only ? maybe)
    ),
    SourceWriter(output_dir_name=OUTPUT_DIRS['PHOTCAL'])
]

test_config = list(itertools.chain(
    load,
    bkg_sub,
    src_det,
    photcal
))

pipeline = WifesAutoguiderPipeline(night=f"test_{TEST_ID}")
pipeline.night_sub_dir = TEST_DIR
pipeline.add_configuration(configuration_name="test_config",␣
  ↪configuration=test_config)

save_params(Path(TEST_DIR).joinpath('all.param'))
```

[ ]:
```
# !! clear outputs !!
clear_dirs = [
    OUTPUT_DIRS['BKG'],
```

```
    OUTPUT_DIRS['DET'],
    OUTPUT_DIRS['PSF_MODEL'],
    OUTPUT_DIRS['PSF_PHOT'],
    OUTPUT_DIRS['APER_PHOT'],
    OUTPUT_DIRS['PHOTCAL'],
]
for dir in clear_dirs:
    for root, dirs, files in os.walk(dir, topdown=False):
        for file in files:
            os.remove(os.path.join(root, file))
        for dir in dirs:
            os.rmdir(os.path.join(root, dir))
```

[ ]:
```
logger.debug(f"\n\n{datetime.now()}\n\n")

for file in os.listdir(DATA_DIR):
    if file.endswith('.fits'):
        move_to_raw(os.path.join(DATA_DIR,file))
configuration = "test_config"
flowify(processor_list=eval(configuration), output_path=Path(TEST_DIR).
 ↪joinpath(configuration))
pipeline.reduce_images(Dataset([ImageBatch()]), catch_all_errors=False,␣
 ↪selected_configurations=configuration)
```

```
  0%|          | 0/1 [00:00<?, ?it/s]
100%|      | 4/4 [00:00<00:00, 121.21it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/mirar/mirar/processors/photometry/utils.
py:316: RuntimeWarning: invalid value encountered in log10
  magnitudes = zeropoint - 2.5 * np.log10(flux_list)
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
```

```
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: RADECSYS= 'FK5 '
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
  0%|          | 0/1 [00:00<?, ?it/s]
```

[ ]:
```
(<mirar.data.base_data.Dataset at 0x38689bb10>,
 <mirar.errors.error_stack.ErrorStack at 0x3750dafd0>)
```

[ ]:
```
img = Fits('/Users/astqx/Desktop/WiFeS/WiFeS_seeing/SampleData/
 ↪OBK-530784-WiFeS-Acq--UT20231008T093800-5.fits')
wcs = WCS(img.header[0])
```

```
Filename: /Users/astqx/Desktop/WiFeS/WiFeS_seeing/SampleData/OBK-530784-WiFeS-
Acq--UT20231008T093800-5.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      60   (1072, 1027)   int16 (rescales to
uint16)
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Coordinate reference frame
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
```

[ ]:
```
np.sqrt(np.abs(np.linalg.det(wcs.pixel_scale_matrix)))*3600
```

[ ]:
```
0.2661384728756299
```

[ ]:
```
img.header[0]
```

[ ]:
```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                   16 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 1072 / length of data axis 1
NAXIS2  =                 1027 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A…376..359H
BZERO   =                32768 / offset data range to that of unsigned short
BSCALE  =                    1 / default scaling factor
OBSBLKID=               530784 / Observation Block ID
PROPID  =              2370179 / Proposal ID
DATE-OBS= '2023-10-08T09:38:58.1' / UT at start of exposure
RA      = '20:26:42.690'       / Right ascension (hours)
DEC     = '+10:41:41.90'       / Declination (degrees)
```

```
HA      = '+00:14:36.0'        / Hour angle at start (hours)
ROTREF  = 'POSITION_ANGLE'     / Rotator reference
DETSEC  = '[1:1072,1:1027]'    / Region of detector
CCDSUM  = '1 1     '           / CCD on-chip summing
RADECSYS= 'FK5     '           / Coordinate reference frame
CTYPE1  = 'RA---TAN'           / Coordinate system of x-axis
CTYPE2  = 'DEC--TAN'           / Coordinate system of y-axis
CUNIT1  = 'deg     '           / Unit of coordinate transformation
CUNIT2  = 'deg     '           / Unit of coordinate transformation
OBJECT  = '2023uda '           / User defined ID
FILTER  = 'I       '           / Acq/Guide filter
TVFILT  = 'I       '           / Guide system filter
OBSERVAT= 'SSO     '           / Observatory
TELESCOP= 'ANU 2.3m'           / Telescope
TIMESYS = 'UTC     '           / Time System
EXPTIME =              7.92447 / Exposure time (s)
MJD-OBS =     60225.4020601852 / MJD at start of exposure
TELPAN  =                   0. / Position angle (degrees)
TELVAN  =             184.1664 / Vertical angle (degrees)
MECHROTA=              52.0038 / Rotator mechanical angle (degrees)
APERX   =                  1.5 / Focal plane X position of aperture (arcsec)
APERY   =                 -1. / Focal plane Y position of aperture (arcsec)
TELFOCUS=             -84.8595 / Focus position (mm)
ZD      =              42.1594 / Zenith distance at start (degrees)
AIRMASS =                1.349 / Airmass at start
LTM1_1  =                   1. / Image transform matrix
LTM1_2  =                   0. / Image transform matrix
LTM2_1  =                   0. / Image transform matrix
LTM2_2  =                   1. / Image transform matrix
LTV1    =                   0. / Image transform vector
LTV2    =                   0. / Image transform vector
APERPIXX=      530.30854088513 / Aperture X position (0-Indexed pixels)
APERPIXY=     509.586299147722 / Aperture Y position (0-Indexed pixels)
CRPIX1  =      530.80854088513 / Reference pixel X
CRPIX2  =     510.086299147722 / Reference pixel Y
CD1_1   = 2.36767754033145E-07 / Matrix component
CD1_2   = -7.45775588821833E-05 / Matrix component
CD2_1   = -7.32820784640372E-05 / Matrix component
CD2_2   = -2.3265488156849E-07 / Matrix component
CRVAL1  =           306.677875 / Reference RA
CRVAL2  =     10.6949722222222 / Reference Dec
EQUINOX =                2000. / WCS coordinate equinox
LAT-OBS =             -31.27336 / Observatory latitude (deg)
LONG-OBS=              149.0612 / Observatory longitude (deg)
ALT-OBS =                 1149. / Observatory altitude (m)
```

[ ]:
```
cat = get_table_from_ldac('/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/
 ↪detection/OBK-530784-WiFeS-Acq-0-0-UT20231008T093612-0.cat')
cat['MAG_AUTO']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[69], line 2
      1 cat = get_table_from_ldac('/Users/astqx/Desktop/WiFeS/WiFeS_seeing/
 ↪test_0/detection/OBK-530784-WiFeS-Acq-0-0-UT20231008T093612-0.cat')
----> 2 cat['MAG_AUTO']

File ~/miniconda3/envs/mirar/lib/python3.11/site-packages/astropy/table/table.p
 ↪2055, in Table.__getitem__(self, item)
   2053 def __getitem__(self, item):
   2054     if isinstance(item, str):
-> 2055         return self.columns[item]
   2056     elif isinstance(item, (int, np.integer)):
   2057         return self.Row(self, item)

File ~/miniconda3/envs/mirar/lib/python3.11/site-packages/astropy/table/table.p
 ↪264, in TableColumns.__getitem__(self, item)
    253 """Get items from a TableColumns object.
    254
    255 ::
   (…)
    261   tc[1:3] # <TableColumns names=('b', 'c')>
    262 """
    263 if isinstance(item, str):
--> 264     return OrderedDict.__getitem__(self, item)
    265 elif isinstance(item, (int, np.integer)):
    266     return list(self.values())[item]

KeyError: 'MAG_AUTO'
```

[ ]:
```
wcs.proj_plane_pixel_scales()[0].to(u.arcsec)
```

[ ]: $0.26381686\ ''$

[ ]:
```
np.matmul(np.array([[306,10]]),wcs.pixel_scale_matrix)
```

[ ]:
```
array([[-0.00066037, -0.02282306]])
```

[ ]:
```
wcs
```

[ ]:
```
WCS Keywords

Number of WCS axes: 2
```

```
CTYPE : 'RA---TAN'  'DEC--TAN'
CRVAL : 306.677875  10.6949722222222
CRPIX : 530.80854088513  510.086299147722
CD1_1 CD1_2  : 2.36767754033145e-07  -7.45775588821833e-05
CD2_1 CD2_2  : -7.32820784640372e-05  -2.3265488156849e-07
NAXIS : 1072  1027
```

[ ]:
```
a = wcs.pixel_to_world(20,20)
b = wcs.pixel_to_world(20,21)
print(a,b)
```

```
<SkyCoord (FK5: equinox=2000.0): (ra, dec) in deg
    (306.71487639, 10.73244365)> <SkyCoord (FK5: equinox=2000.0): (ra, dec) in
deg
    (306.71480048, 10.73244342)>
```

[ ]:
```
for file in os.listdir(OUTPUT_DIRS['BKG']):
    if file.endswith('.fits'):
        file = Fits(os.path.join(OUTPUT_DIRS['BKG'],file))
        display(file.header[0])
```

```
Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-530784-WiFeS-
Acq-0-1-UT20231008T093616-7.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      76   (1072, 1027)   float64

SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                  -64 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 1072 / length of data axis 1
NAXIS2  =                 1027 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A…376..359H
OBSBLKID=               530784 / Observation Block ID
PROPID  =              2370179 / Proposal ID
DATE-OBS= '2023-10-08T09:37:19.7' / UT at start of exposure
RA      = '20:26:46.000'       / Right ascension (hours)
DEC     = '+10:42:08.71'       / Declination (degrees)
HA      = '+00:12:48.9'        / Hour angle at start (hours)
ROTREF  = 'POSITION_ANGLE'     / Rotator reference
DETSEC  = '[1:1072,1:1027]'    / Region of detector
CCDSUM  = '1 1     '           / CCD on-chip summing
CTYPE1  = 'RA---TAN'           / Coordinate system of x-axis
CTYPE2  = 'DEC--TAN'           / Coordinate system of y-axis
CUNIT1  = 'deg     '           / Unit of coordinate transformation
CUNIT2  = 'deg     '           / Unit of coordinate transformation
```

```
OBJECT  = '2023uda '           / User defined ID
TVFILT  = 'I       '           / Guide system filter
OBSERVAT= 'SSO     '           / Observatory
TELESCOP= 'ANU 2.3m'           / Telescope
TIMESYS = 'UTC     '           / Time System
EXPTIME =                0.796 / Exposure time (s)
MJD-OBS =     60225.4009143518 / MJD at start of exposure
TELPAN  =                   0. / Position angle (degrees)
TELVAN  =              183.6108 / Vertical angle (degrees)
MECHROTA=                 51.46 / Rotator mechanical angle (degrees)
APERX   =                  41.6 / Focal plane X position of aperture (arcsec)
APERY   =                -41.85 / Focal plane Y position of aperture (arcsec)
TELFOCUS=               -84.855 / Focus position (mm)
ZD      =               42.1363 / Zenith distance at start (degrees)
AIRMASS =                1.3485 / Airmass at start
LTM1_1  =                    1. / Image transform matrix
LTM1_2  =                    0. / Image transform matrix
LTM2_1  =                    0. / Image transform matrix
LTM2_2  =                    1. / Image transform matrix
LTV1    =                    0. / Image transform vector
LTV2    =                    0. / Image transform vector
APERPIXX=     685.632616252871 / Aperture X position (0-Indexed pixels)
APERPIXY=     358.079242890045 / Aperture Y position (0-Indexed pixels)
CRPIX1  =     686.132616252871 / Reference pixel X
CRPIX2  =     358.579242890045 / Reference pixel Y
CD1_1   = 2.36773568324427E-07 / Matrix component
CD1_2   = -7.45793902787442E-05 / Matrix component
CD2_1   = -7.32820784640372E-05 / Matrix component
CD2_2   = -2.3265488156849E-07 / Matrix component
CRVAL1  =     306.691666666667 / Reference RA
CRVAL2  =     10.7024194444444 / Reference Dec
EQUINOX =                2000. / WCS coordinate equinox
LAT-OBS =             -31.27336 / Observatory latitude (deg)
LONG-OBS=              149.0612 / Observatory longitude (deg)
ALT-OBS =                 1149. / Observatory altitude (m)
BASENAME= 'OBK-530784-WiFeS-Acq-0-1-UT20231008T093616-7.fits'
RAWPATH = '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/raw/OBK-530784-WiFeS&'
CONTINUE  '-Acq-0-1-UT20231008T093616-7.fits'
OBSCLASS= 'acq     '
TARGNAME= '2023uda '
COADDS  =                    1
GAIN    =                    1
CALSTEPS= 'load,photutilsbkgsubtractor,'
PROCFAIL= ''
RADESYSA= 'FK5     '
RADECSYS= 'FK5     '
REDUCER = 'astqx   '
REDMACH = 'Adityas-MacBook-Pro.local'
```

```
REDTIME = '2024-02-02 17:24:04.046936'
REDSOFT = 'mirar   '
BGMED   =    945.1675170068028
BGRMSMED=   10.908523220060083
SAVEPATH= '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-53078&'
CONTINUE  '4-WiFeS-Acq-0-1-UT20231008T093616-7.fits'
DATE    = '2024-02-02T06:24:04.069'



Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-530784-WiFeS-
Acq-0-0-UT20231008T093612-0.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      76   (1072, 1027)   float64

SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                  -64 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 1072 / length of data axis 1
NAXIS2  =                 1027 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A…376..359H
OBSBLKID=               530784 / Observation Block ID
PROPID  =              2370179 / Proposal ID
DATE-OBS= '2023-10-08T09:37:11.9' / UT at start of exposure
RA      = '20:26:46.000'       / Right ascension (hours)
DEC     = '+10:42:08.63'       / Declination (degrees)
HA      = '+00:12:43.9'        / Hour angle at start (hours)
ROTREF  = 'POSITION_ANGLE'     / Rotator reference
DETSEC  = '[1:1072,1:1027]'    / Region of detector
CCDSUM  = '1 1     '           / CCD on-chip summing
CTYPE1  = 'RA---TAN'           / Coordinate system of x-axis
CTYPE2  = 'DEC--TAN'           / Coordinate system of y-axis
CUNIT1  = 'deg     '           / Unit of coordinate transformation
CUNIT2  = 'deg     '           / Unit of coordinate transformation
OBJECT  = '2023uda '           / User defined ID
TVFILT  = 'I       '           / Guide system filter
OBSERVAT= 'SSO     '           / Observatory
TELESCOP= 'ANU 2.3m'           / Telescope
TIMESYS = 'UTC     '           / Time System
EXPTIME =              3.83911 / Exposure time (s)
MJD-OBS =     60225.4008217593 / MJD at start of exposure
TELPAN  =                   0. / Position angle (degrees)
TELVAN  =              183.582 / Vertical angle (degrees)
MECHROTA=              51.4325 / Rotator mechanical angle (degrees)
APERX   =                 41.6 / Focal plane X position of aperture (arcsec)
APERY   =               -41.85 / Focal plane Y position of aperture (arcsec)
```

```
TELFOCUS=             -84.8096 / Focus position (mm)
ZD      =              42.1349 / Zenith distance at start (degrees)
AIRMASS =               1.3485 / Airmass at start
LTM1_1  =                   1. / Image transform matrix
LTM1_2  =                   0. / Image transform matrix
LTM2_1  =                   0. / Image transform matrix
LTM2_2  =                   1. / Image transform matrix
LTV1    =                   0. / Image transform vector
LTV2    =                   0. / Image transform vector
APERPIXX=     685.632616252871 / Aperture X position (0-Indexed pixels)
APERPIXY=     358.079242890045 / Aperture Y position (0-Indexed pixels)
CRPIX1  =     686.132616252871 / Reference pixel X
CRPIX2  =     358.579242890045 / Reference pixel Y
CD1_1   = 2.36773550968432E-07 / Matrix component
CD1_2   = -7.45793848119198E-05 / Matrix component
CD2_1   = -7.32820784640372E-05 / Matrix component
CD2_2   = -2.3265488156849E-07 / Matrix component
CRVAL1  =     306.691666666667 / Reference RA
CRVAL2  =     10.7023972222222 / Reference Dec
EQUINOX =                2000. / WCS coordinate equinox
LAT-OBS =            -31.27336 / Observatory latitude (deg)
LONG-OBS=             149.0612 / Observatory longitude (deg)
ALT-OBS =                1149. / Observatory altitude (m)
BASENAME= 'OBK-530784-WiFeS-Acq-0-0-UT20231008T093612-0.fits'
RAWPATH = '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/raw/OBK-530784-WiFeS&'
CONTINUE  '-Acq-0-0-UT20231008T093612-0.fits'
OBSCLASS= 'acq     '
TARGNAME= '2023uda '
COADDS  =                    1
GAIN    =                    1
CALSTEPS= 'load,photutilsbkgsubtractor,'
PROCFAIL= ''
RADESYSA= 'FK5     '
RADECSYS= 'FK5     '
REDUCER = 'astqx   '
REDMACH = 'Adityas-MacBook-Pro.local'
REDTIME = '2024-02-02 17:24:04.046904'
REDSOFT = 'mirar   '
BGMED   =    947.5350000000001
BGRMSMED=   11.084449186770835
SAVEPATH= '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-53078&'
CONTINUE  '4-WiFeS-Acq-0-0-UT20231008T093612-0.fits'
DATE    = '2024-02-02T06:24:04.063'


Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-530784-WiFeS-Acq--
```

```
UT20231008T093800-5.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      77   (1072, 1027)   float64

SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                  -64 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 1072 / length of data axis 1
NAXIS2  =                 1027 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A…376..359H
OBSBLKID=               530784 / Observation Block ID
PROPID  =              2370179 / Proposal ID
DATE-OBS= '2023-10-08T09:38:58.1' / UT at start of exposure
RA      = '20:26:42.690'       / Right ascension (hours)
DEC     = '+10:41:41.90'       / Declination (degrees)
HA      = '+00:14:36.0'        / Hour angle at start (hours)
ROTREF  = 'POSITION_ANGLE'     / Rotator reference
DETSEC  = '[1:1072,1:1027]'    / Region of detector
CCDSUM  = '1 1     '           / CCD on-chip summing
CTYPE1  = 'RA---TAN'           / Coordinate system of x-axis
CTYPE2  = 'DEC--TAN'           / Coordinate system of y-axis
CUNIT1  = 'deg     '           / Unit of coordinate transformation
CUNIT2  = 'deg     '           / Unit of coordinate transformation
OBJECT  = '2023uda '           / User defined ID
FILTER  = 'I       '           / Acq/Guide filter
TVFILT  = 'I       '           / Guide system filter
OBSERVAT= 'SSO     '           / Observatory
TELESCOP= 'ANU 2.3m'           / Telescope
TIMESYS = 'UTC     '           / Time System
EXPTIME =              7.92447 / Exposure time (s)
MJD-OBS =     60225.4020601852 / MJD at start of exposure
TELPAN  =                   0. / Position angle (degrees)
TELVAN  =             184.1664 / Vertical angle (degrees)
MECHROTA=              52.0038 / Rotator mechanical angle (degrees)
APERX   =                  1.5 / Focal plane X position of aperture (arcsec)
APERY   =                   -1. / Focal plane Y position of aperture (arcsec)
TELFOCUS=             -84.8595 / Focus position (mm)
ZD      =              42.1594 / Zenith distance at start (degrees)
AIRMASS =                1.349 / Airmass at start
LTM1_1  =                   1. / Image transform matrix
LTM1_2  =                   0. / Image transform matrix
LTM2_1  =                   0. / Image transform matrix
LTM2_2  =                   1. / Image transform matrix
LTV1    =                   0. / Image transform vector
LTV2    =                   0. / Image transform vector
APERPIXX=      530.30854088513 / Aperture X position (0-Indexed pixels)
```

```
APERPIXY=     509.586299147722 / Aperture Y position (0-Indexed pixels)
CRPIX1  =      530.80854088513 / Reference pixel X
CRPIX2  =     510.086299147722 / Reference pixel Y
CD1_1   = 2.36767754033145E-07 / Matrix component
CD1_2   = -7.45775588821833E-05 / Matrix component
CD2_1   = -7.32820784640372E-05 / Matrix component
CD2_2   = -2.3265488156849E-07 / Matrix component
CRVAL1  =           306.677875 / Reference RA
CRVAL2  =     10.6949722222222 / Reference Dec
EQUINOX =                2000. / WCS coordinate equinox
LAT-OBS =            -31.27336 / Observatory latitude (deg)
LONG-OBS=             149.0612 / Observatory longitude (deg)
ALT-OBS =                1149. / Observatory altitude (m)
BASENAME= 'OBK-530784-WiFeS-Acq--UT20231008T093800-5.fits'
RAWPATH = '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/raw/OBK-530784-WiFeS&'
CONTINUE  '-Acq--UT20231008T093800-5.fits'
OBSCLASS= 'acq     '
TARGNAME= '2023uda '
COADDS  =                    1
GAIN    =                    1
CALSTEPS= 'load,photutilsbkgsubtractor,'
PROCFAIL= ''
RADESYSA= 'FK5     '
RADECSYS= 'FK5     '
REDUCER = 'astqx   '
REDMACH = 'Adityas-MacBook-Pro.local'
REDTIME = '2024-02-02 17:24:04.046862'
REDSOFT = 'mirar   '
BGMED   =               949.03
BGRMSMED=   11.273863579093016
SAVEPATH= '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-53078&'
CONTINUE  '4-WiFeS-Acq--UT20231008T093800-5.fits'
DATE    = '2024-02-02T06:24:04.053'



Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-530784-WiFeS-Acq-
CM-0-UT20231008T093601-3.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      76   (1072, 1027)   float64

SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                  -64 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 1072 / length of data axis 1
NAXIS2  =                 1027 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
```

```
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A…376..359H
OBSBLKID=               530784 / Observation Block ID
PROPID  =              2370179 / Proposal ID
DATE-OBS= '2023-10-08T09:37:04.4' / UT at start of exposure
RA      = '20:26:46.330'       / Right ascension (hours)
DEC     = '+10:42:06.62'       / Declination (degrees)
HA      = '+00:12:33.0'        / Hour angle at start (hours)
ROTREF  = 'POSITION_ANGLE'     / Rotator reference
DETSEC  = '[1:1072,1:1027]'    / Region of detector
CCDSUM  = '1 1     '           / CCD on-chip summing
CTYPE1  = 'RA---TAN'           / Coordinate system of x-axis
CTYPE2  = 'DEC--TAN'           / Coordinate system of y-axis
CUNIT1  = 'deg     '           / Unit of coordinate transformation
CUNIT2  = 'deg     '           / Unit of coordinate transformation
OBJECT  = '2023uda '           / User defined ID
TVFILT  = 'I       '           / Guide system filter
OBSERVAT= 'SSO     '           / Observatory
TELESCOP= 'ANU 2.3m'           / Telescope
TIMESYS = 'UTC     '           / Time System
EXPTIME =              3.16979 / Exposure time (s)
MJD-OBS =     60225.4007407407 / MJD at start of exposure
TELPAN  =                   0. / Position angle (degrees)
TELVAN  =             183.5237 / Vertical angle (degrees)
MECHROTA=              51.3776 / Rotator mechanical angle (degrees)
APERX   =                 41.6 / Focal plane X position of aperture (arcsec)
APERY   =               -41.85 / Focal plane Y position of aperture (arcsec)
TELFOCUS=              -84.855 / Focus position (mm)
ZD      =              42.1315 / Zenith distance at start (degrees)
AIRMASS =               1.3484 / Airmass at start
LTM1_1  =                   1. / Image transform matrix
LTM1_2  =                   0. / Image transform matrix
LTM2_1  =                   0. / Image transform matrix
LTM2_2  =                   1. / Image transform matrix
LTV1    =                   0. / Image transform vector
LTV2    =                   0. / Image transform vector
APERPIXX=     685.632616252871 / Aperture X position (0-Indexed pixels)
APERPIXY=     358.079242890045 / Aperture Y position (0-Indexed pixels)
CRPIX1  =     686.132616252871 / Reference pixel X
CRPIX2  =     358.579242890045 / Reference pixel Y
CD1_1   = 2.36773114911587E-07 / Matrix component
CD1_2   = -7.45792474619028E-05 / Matrix component
CD2_1   = -7.32820784640372E-05 / Matrix component
CD2_2   = -2.3265488156849E-07 / Matrix component
CRVAL1  =     306.693041666667 / Reference RA
CRVAL2  =     10.7018388888889 / Reference Dec
EQUINOX =                2000. / WCS coordinate equinox
LAT-OBS =            -31.27336 / Observatory latitude (deg)
LONG-OBS=             149.0612 / Observatory longitude (deg)
```

```
ALT-OBS =                1149. / Observatory altitude (m)
BASENAME= 'OBK-530784-WiFeS-Acq-CM-0-UT20231008T093601-3.fits'
RAWPATH = '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/raw/OBK-530784-WiFeS&'
CONTINUE  '-Acq-CM-0-UT20231008T093601-3.fits'
OBSCLASS= 'acq     '
TARGNAME= '2023uda '
COADDS  =                    1
GAIN    =                    1
CALSTEPS= 'load,photutilsbkgsubtractor,'
PROCFAIL= ''
RADESYSA= 'FK5     '
RADECSYS= 'FK5     '
REDUCER = 'astqx   '
REDMACH = 'Adityas-MacBook-Pro.local'
REDTIME = '2024-02-02 17:24:04.046965'
REDSOFT = 'mirar   '
BGMED   =    946.8793939393938
BGRMSMED=   11.046089666542073
SAVEPATH= '/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-53078&'
CONTINUE  '4-WiFeS-Acq-CM-0-UT20231008T093601-3.fits'
DATE    = '2024-02-02T06:24:04.074'
```

[ ]:
```
outdir = Path(OUTPUT_DIRS['DET'])
for file in os.listdir(outdir):
    if file.endswith('.segm'):
        segm = Fits(outdir.joinpath(file))
        cmap = load_object(outdir.joinpath(file+'.cmap.pkl'))
        plt.figure(clear=True)
        plt.imshow(segm.data[0], origin='lower',␣
 ↪cmap=cmap,interpolation='nearest')
        plt.show()
```

```
Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-
Acq-0-0-UT20231008T093612-0.fits.segm
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      79   (1072, 1027)   int64 (rescales to
float64)
Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-Acq-
CM-0-UT20231008T093601-3.fits.segm
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      79   (1072, 1027)   int64 (rescales to
float64)
Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-Acq--
```

```
UT20231008T093800-5.fits.segm
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      80   (1072, 1027)   int64 (rescales to
float64)
Filename:
/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-
Acq-0-1-UT20231008T093616-7.fits.segm
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      79   (1072, 1027)   int64 (rescales to
float64)

/var/folders/n1/nl_dwlrx4v9_6j5wdwq5xnwh0000gn/T/ipykernel_65675/858614285.py:8:
UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  plt.show()
```

[ ]:
```
load_object('/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/
↪OBK-530784-WiFeS-Acq--UT20231008T093800-5_sources.pkl').
↪get_data()['kron_aperture'][0]
```

[ ]:
```
9.916381467479884
```

[ ]:
```
from astroquery.vizier import Vizier
```

[ ]:
```
catalog = Vizier()
```

[ ]:
```
tables = catalog.query_object('sirius',catalog='I/355/gaiadr3')
```

[ ]:
```
tables[0]
```

[ ]:
```
<Table length=50>
    RA_ICRS         DE_ICRS     …     RAJ2000         DEJ2000
      deg             deg       …       deg             deg
    float64         float64     …     float64         float64
--------------- --------------- … --------------- ---------------
101.29264109428 -16.74746500884 … 101.29264109428 -16.74746500884
101.30841658134 -16.74029759782 … 101.30841658134 -16.74029759782
101.30585400236 -16.73843681955 … 101.30585400236 -16.73843681955
101.29751227495 -16.74425044166 … 101.29750574897 -16.74425380959
101.30903506763 -16.73977481430 … 101.30903506763 -16.73977481430
101.28893444726 -16.74596584258 … 101.28893444726 -16.74596584258
101.29298121878 -16.73173035325 … 101.29298122825 -16.73173550173
101.29363786986 -16.74053735100 … 101.29363786986 -16.74053735100
101.31496875086 -16.73451419657 … 101.31505654331 -16.73454691511
            …               … …             …               …
101.29091246209 -16.68991406360 … 101.29089644524 -16.68990058514
101.29663198483 -16.69502728362 … 101.29658518510 -16.69504665819
101.29497173019 -16.69685782605 … 101.29498028678 -16.69686706854
101.25263478810 -16.71659640724 … 101.25263478810 -16.71659640724
```

```
101.25252135203 -16.71678853149 … 101.25252135203 -16.71678853149
101.26072630956 -16.69647067631 … 101.26072630956 -16.69647067631
101.25651330308 -16.70193879340 … 101.25651330308 -16.70193879340
101.26101350472 -16.70102626104 … 101.26101350472 -16.70102626104
101.25720756232 -16.70426576966 … 101.25724720631 -16.70429870909
101.26784885815 -16.70320528060 … 101.26784885815 -16.70320528060
```

[ ]:
```
load_object('/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/photcal/
↪OBK-530784-WiFeS-Acq--UT20231008T093800-5_sources.pkl').get_metadata()
```

[ ]:
```
{'SIMPLE': True,
 'BITPIX': -64,
 'NAXIS': 2,
 'NAXIS1': 1072,
 'NAXIS2': 1027,
 'EXTEND': True,
 'BZERO': 32768,
 'BSCALE': 1,
 'OBSBLKID': 530784,
 'PROPID': 2370179,
 'DATE-OBS': '2023-10-08T09:38:58.1',
 'RA': '20:26:42.690',
 'DEC': '+10:41:41.90',
 'HA': '+00:14:36.0',
 'ROTREF': 'POSITION_ANGLE',
 'DETSEC': '[1:1072,1:1027]',
 'CCDSUM': '1 1',
 'CTYPE1': 'RA---TAN',
 'CTYPE2': 'DEC--TAN',
 'CUNIT1': 'deg',
 'CUNIT2': 'deg',
 'OBJECT': '2023uda',
 'FILTER': 'I',
 'TVFILT': 'I',
 'OBSERVAT': 'SSO',
 'TELESCOP': 'ANU 2.3m',
 'TIMESYS': 'UTC',
 'EXPTIME': 7.92447,
 'MJD-OBS': 60225.4020601852,
 'TELPAN': 0.0,
 'TELVAN': 184.1664,
 'MECHROTA': 52.0038,
 'APERX': 1.5,
 'APERY': -1.0,
 'TELFOCUS': -84.8595,
 'ZD': 42.1594,
 'AIRMASS': 1.349,
```

```
 'LTM1_1': 1.0,
 'LTM1_2': 0.0,
 'LTM2_1': 0.0,
 'LTM2_2': 1.0,
 'LTV1': 0.0,
 'LTV2': 0.0,
 'APERPIXX': 530.30854088513,
 'APERPIXY': 509.586299147722,
 'CRPIX1': 530.80854088513,
 'CRPIX2': 510.086299147722,
 'CD1_1': 2.36767754033145e-07,
 'CD1_2': -7.45775588821833e-05,
 'CD2_1': -7.32820784640372e-05,
 'CD2_2': -2.3265488156849e-07,
 'CRVAL1': 306.677875,
 'CRVAL2': 10.6949722222222,
 'EQUINOX': 2000.0,
 'LAT-OBS': -31.27336,
 'LONG-OBS': 149.0612,
 'ALT-OBS': 1149.0,
 'BASENAME': 'OBK-530784-WiFeS-Acq--UT20231008T093800-5.fits',
 'RAWPATH':
'/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/raw/OBK-530784-WiFeS-Acq--
UT20231008T093800-5.fits',
 'OBSCLASS': 'acq',
 'TARGNAME': '2023uda',
 'COADDS': 1,
 'GAIN': 1,
 'CALSTEPS': 'load,photutilsbkgsubtractor,save,photutilssourcedetection,photutil
ssourcecatalog,SRCWRITE,photcalibrator,',
 'PROCFAIL': '',
 'RADESYSA': 'FK5',
 'RADECSYS': 'FK5',
 'REDUCER': 'astqx',
 'REDMACH': 'Adityas-MacBook-Pro.local',
 'REDTIME': '2024-02-05 16:39:00.091631',
 'REDSOFT': 'mirar',
 'BGMED': 949.03,
 'BGRMSMED': 11.273863579093016,
 'BKGRMS':
'/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-530784-WiFeS-Acq
--UT20231008T093800-5.fits.background_rms',
 'SAVEPATH':
'/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/background/OBK-530784-WiFeS-Acq
--UT20231008T093800-5.fits',
 'DATE': '2024-02-05T05:38:51.687',
 'SEGMOBJ':
```

```
'/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-Acq--
UT20231008T093800-5.fits.segm.pkl',
 'SEGMAP':
'/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-Acq--
UT20231008T093800-5.fits.segm',
 'CONVPATH':
'/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-Acq--
UT20231008T093800-5.fits.conv',
 'SRCCAT':
'/Users/astqx/Desktop/WiFeS/WiFeS_seeing/test_0/detection/OBK-530784-WiFeS-Acq--
UT20231008T093800-5.cat',
 'FWHM_MED': 0.00036506673463802,
 'FWHM_STD': 0.00013976367762040588,
 'FWHM_PIX': 4.938182106842681,
 'ZP_AUTO': 25.48361467108078,
 'ZP_AUTO_std': 0.08858372991190502,
 'ZP_AUTO_nstars': 15,
 'MAGLIM': 10.091836110835873,
 'ZP': 25.48361467108078,
 'ZPSTD': 0.08858372991190502,
 'ZPNSTARS': 15,
 'MAGSYS': 'AB'}
```

[ ]:
```
0.00036506673463802*3600
```

[ ]:
```
1.314240244696872
```

[ ]:
```

```