# Python在天文观测和数据处理中的应用

张天萌@兴隆

2023.06.14

- 天文学研究中的python应用
  - FITS图像相关
  - 天文观测和数据处理

- 软件系统工程化中的python应用
  - CSST数据系统开发

# CCD（Charge-Coupled Devices）的诞生和发展

## The Nobel Prize in Physics 1921

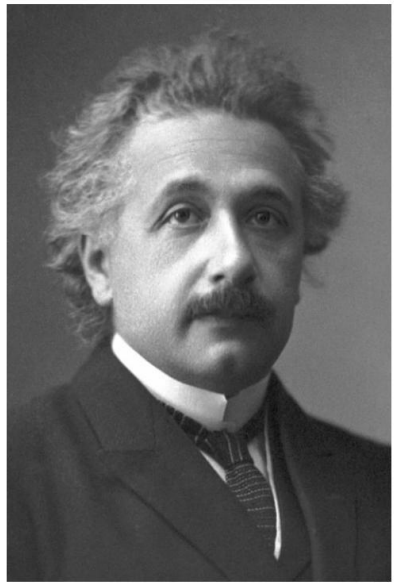


Photo from the Nobel Foundation archive.

Albert Einstein

Prize share: 1/1

"for his services to Theoretical Physics, and especially for his discovery of the law of the photoelectric effect."

- 基本原理来源于爱因斯坦的光电效应

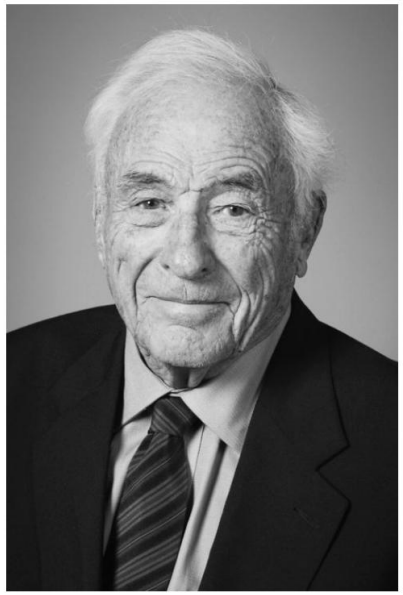
# CCD（Charge-Coupled Devices）的诞生和发展

## The Nobel Prize in Physics 2009

© The Nobel Foundation. Photo: U. Montan

Charles Kuen Kao

Prize share: 1/2



© The Nobel Foundation. Photo: U. Montan

Willard S. Boyle

Prize share: 1/4

© The Nobel Foundation. Photo: U. Montan

George E. Smith

Prize share: 1/4

- 基本原理来源于爱因斯坦的光电效应
- 1969年10月17日诞生于贝尔实验室的黑板上
- 1970年4月首次发表文章描述了新设备的概念和可能的用途（包括光学成像）

# CCD（Charge-Coupled Devices）的诞生和发展

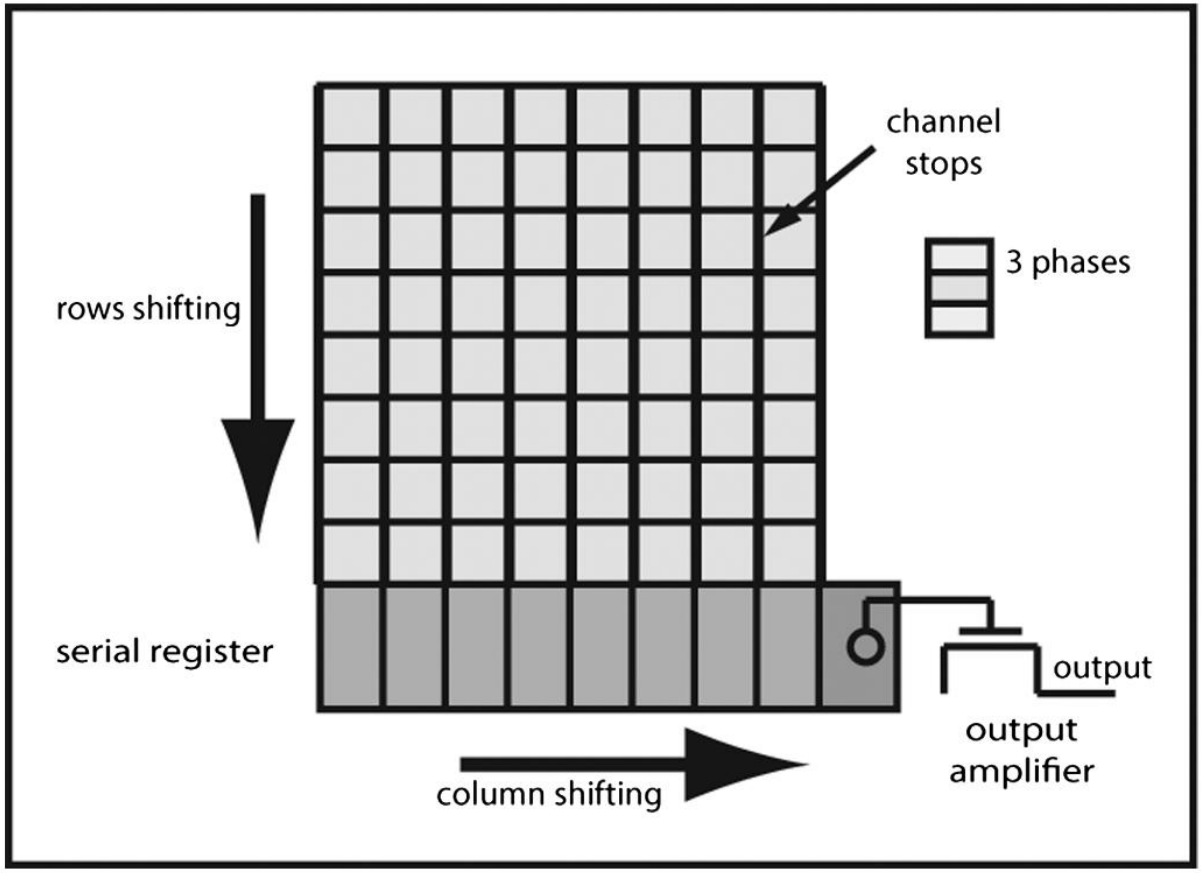
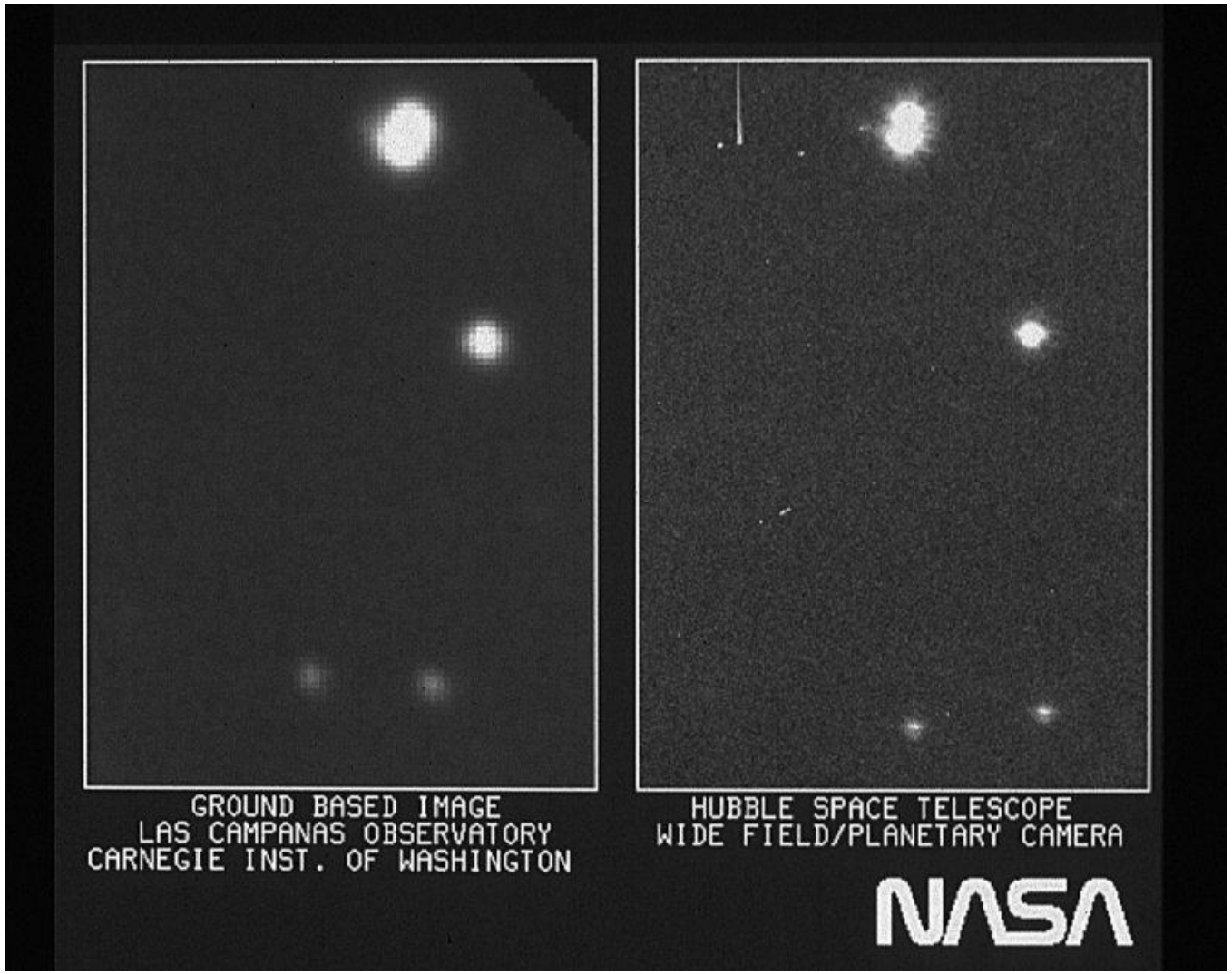



FIG. 1.—Physical layout of a typical three-phase CCD.

Credits: Michael Lesser 2015 PASP

- 基本原理来源于爱因斯坦的光电效应
- 1969年10月17日诞生于贝尔实验室的黑板上
- 1970年4月首次发表文章描述了新设备的概念和可能的用途（包括光学成像）
- 70年代贝尔实验室对设计进行了修改，使得CCD可以读出一幅图像，并将其存储在设备的另一部分，让下一幅图像可以继续记录在CCD上

# CCD（Charge-Coupled Devices）的诞生和发展




- 基本原理来源于爱因斯坦的光电效应
- 1969年10月17日诞生于贝尔实验室的黑板上
- 1970年4月首次发表文章描述了新设备的概念和可能的用途（包括光学成像）
- 70年代贝尔实验室对设计进行了修改，使得CCD可以读出一幅图像，并将其存储在设备的另一部分，让下一幅图像可以继续记录在CCD上
- 1976年使用CCD首次获得了天文图像
- 从地面到天空，从红外到X射线，更多像素，更低读出噪声，更高效率，CCD在天文观测中起着不可替代的作用

# 天文图像格式

- 天文图像数据主要格式：FITS
  - Flexible Image Transport System
  - FITS图像文件协议建立（1979）
    - Primary header and data unit (HDU)
      - Header由多个block组成，每个block包括36*80=2880-byte 必须以“END”结束
      - Data矩阵支持1-999维（通过NAXIS关键词定义，图像通常是NAXIS1和NAXIS2定义二维大小）
      - BITPIX定义矩阵的类型和位数（8，16，-32，64等）
    - Multi-Extension FITS (MEF)

**Definition of the Flexible Image Transport System (*FITS*)**

The *FITS* Standard

Version 4.0: updated 2016 July 22 by the IAUFWG

Original document publication date: 2016 July 22
Language-edited document publication date: 2018 August 13

| Position | Keyword |
|---|---|
| 1 | SIMPLE = T |
| 2 | BITPIX |
| 3 | NAXIS |
| 4 | NAXIS$n, n = 1, \ldots,$ NAXIS |
| | ⋮ |
| | (other keywords) |
| | ⋮ |
| last | END |

| Position | Keyword |
|---|---|
| 1 | XTENSION |
| 2 | BITPIX |
| 3 | NAXIS |
| 4 | NAXIS$n, n = 1, \ldots,$ NAXIS |
| 5 | PCOUNT |
| 6 | GCOUNT |
| | ⋮ |
| | (other keywords) |
| | ⋮ |
| last | END |

# Python在天文图像处理中的基本操作

- FITS操作（astropy.io.fits）
  - 读取：open()，info()，getdata()，getheader()
  - 写文件：PrimaryHDU()，HDUList()，writeto()
  - Header操作
    - 增加/删除keywords
    - 读取keywords的value
  - Data操作

```
In [1]: from astropy.io import fits

In [2]: obj = fits.open('CSST_MSC_MS_CROSSTALK_25_0001.fits')

In [3]: obj.info()
Filename: CSST_MSC_MS_CROSSTALK_25_0001.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      28   ()
  1  SCI           1 ImageHDU         8   (16, 16)   float32
  2  ERR           1 ImageHDU         8   (16, 16)   float32
  3  DQ            1 ImageHDU        10   (16, 16)   int32 (rescales to uint32)


In [4]: header = obj[0].header

In [5]: header['USEAFTER']
Out[5]: '2025-01-01T00:00:00'

In [6]: data = fits.getdata('CSST_MSC_MS_CROSSTALK_25_0001.fits')

In [7]: data.shape
Out[7]: (16, 16)
```

# 多维图像处理基本操作

- 矩阵操作
  - 切割
    - array[100:200, 100:200]
  - 转置
    - array.flatten()
    - array.reshape(100, 100)
  - 镜像
    - numpy.flipud(array)
    - 等效于array[::−1]
  - 旋转
    - numpy.rot90(array, k=2)
    - 等效于np.flipud(np.fliplr(array))
  - 多维矩阵排序/平均：master bias/flat/dark
    - numpy.sort(ndarray, axis = 0)
    - numpy.mean(ndarray_sort, axis = 0)

```
In [1]: import numpy as np

In [2]: array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

In [3]: array.flatten()
Out[3]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [4]: np.flipud(array)
Out[4]:
array([[7, 8, 9],
       [4, 5, 6],
       [1, 2, 3]])

In [5]: array[::-1]
Out[5]:
array([[7, 8, 9],
       [4, 5, 6],
       [1, 2, 3]])

In [6]: a = np.random.random((3,3,3))

In [7]: a
Out[7]:
array([[[0.9610923 , 0.20989562, 0.69589999],
        [0.92906225, 0.38499607, 0.0057568 ],
        [0.14640188, 0.33291293, 0.29034816]],

       [[0.6287261 , 0.60961688, 0.19397544],
        [0.7457779 , 0.26134888, 0.8459206 ],
        [0.15901851, 0.56605089, 0.15058119]],

       [[0.82407722, 0.41190758, 0.21061846],
        [0.8756162 , 0.86174875, 0.00506587],
        [0.76732293, 0.63074179, 0.71960938]]])

In [8]: np.mean(a, axis=0)
Out[8]:
array([[0.80463187, 0.41047336, 0.3668313 ],
       [0.85015212, 0.5026979 , 0.28558109],
       [0.35758111, 0.50990187, 0.38684624]])
```

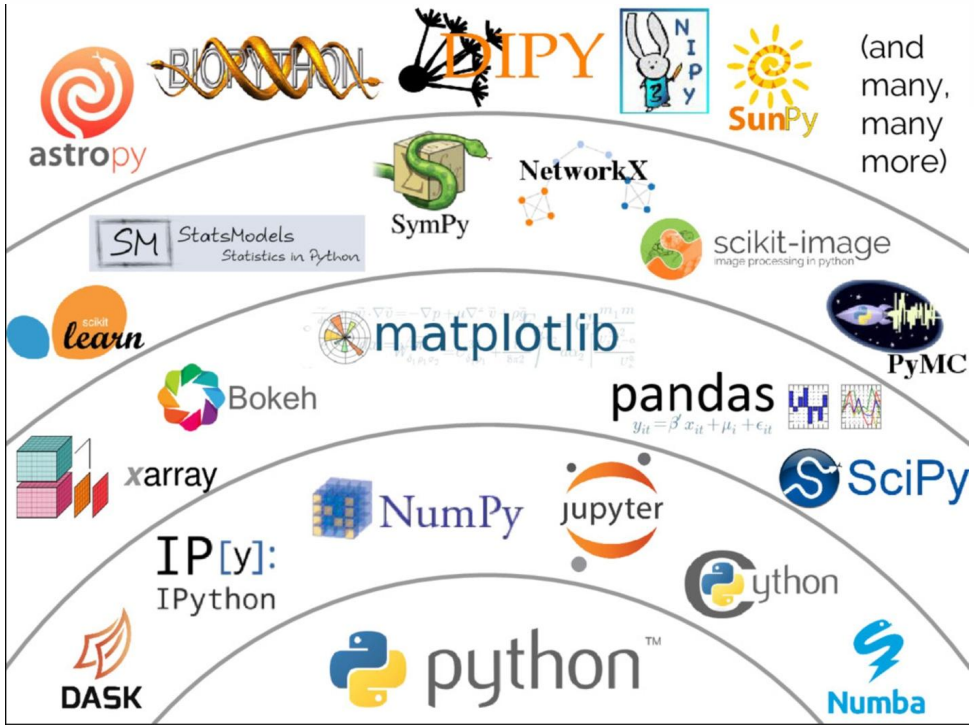
# 其它矩阵操作

- 寻找并定位矩阵中的特殊元素
- 用表达式快速定位矩阵元素
  - index = (array == 3)
  - array[index]
- 生成mask矩阵
  - mask = ((array1 > 0.5) & (array2 < 1.0)

```
In [21]: a = np.random.random((5,5))

In [22]: b = np.random.random((5,5))

In [23]: index = (a > 0.5)

In [24]: mask = ((a > 0.5) & (b < 0.7))

In [25]: mask
Out[25]:
array([[ True, False, False, False, False],
       [False, False, False, False, False],
       [False, False,  True, False,  True],
       [False,  True, False, False,  True],
       [ True, False,  True,  True,  True]])

In [26]: a[index]
Out[26]:
array([0.95859947, 0.89533501, 0.53021969, 0.82986006, 0.53884837,
       0.82154079, 0.91373615, 0.96696163, 0.77411869, 0.76758937,
       0.87019978])

In [27]: a[mask]
Out[27]:
array([0.95859947, 0.53021969, 0.82986006, 0.53884837, 0.91373615,
       0.96696163, 0.77411869, 0.76758937, 0.87019978])
```

# Python在天文中的优势

- 简单
  - 语法简化，可读性强，0基础易上手
- 兼容性强
  - 可调用其它语言代码/库
  - 天文历史上的Fortran和C代码较多
- 库（轮子）多
  - Astropy，NumPy ...
- 用途广泛
  - 图像矩阵计算，画图，获取在线星表 ...
- 学习教材样例多

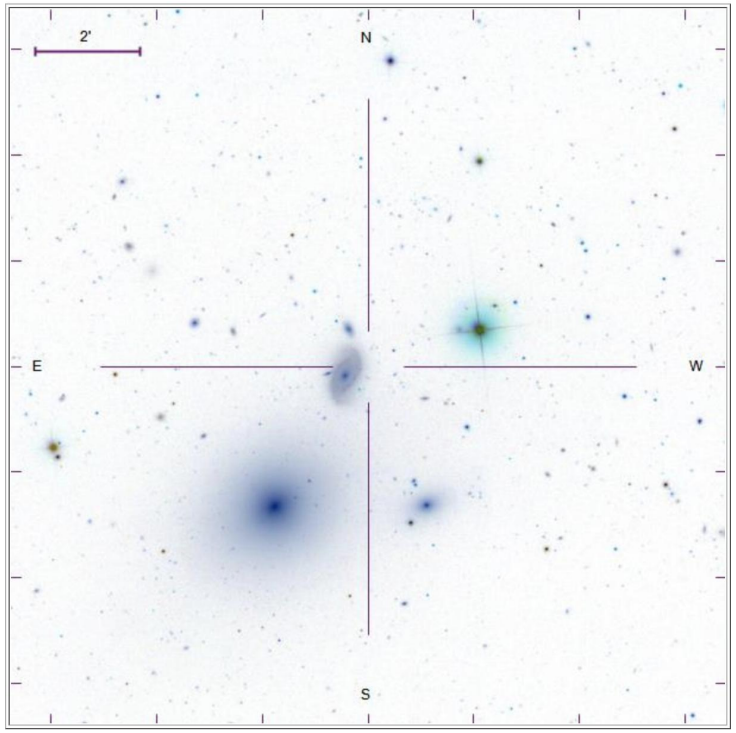
# Python用例：Vizier

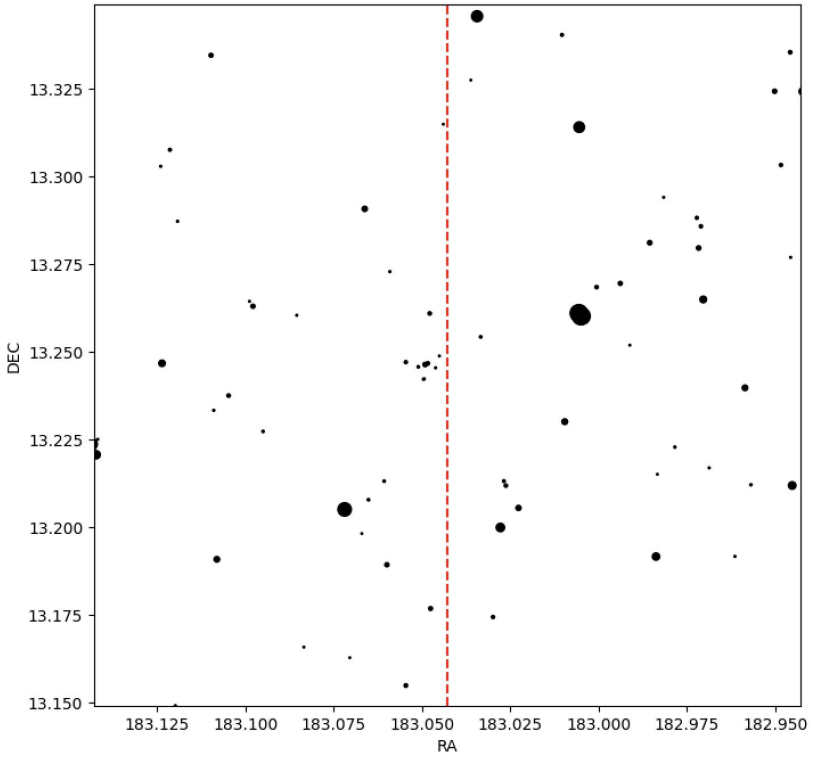
- **Finding chart plot**
  - **获取外部参考星表（Gaia，Pansstar...）**
    - **from astroquery.vizier import Vizier**
    - **from astropy.coordinates import SkyCoord**
  - 选择需要下载的星表和列
    - Gaia DR3：I/355/gaiadr3
    - Pansstar：II/349/ps1
  - 设置指向和半径

```
vquery = Vizier(columns=['RA_ICRS', 'DE_ICRS',
                         'pmRA', 'pmDE', 'Plx', 'RVDR2', 'Gmag'],
                row_limit=maxsources, column_filters={"Gmag": ("<%f" % maxmag), "Plx": ">0"})
coord = SkyCoord(ra=ra, dec=dec, unit=units.deg, frame="icrs")
r = vquery.query_region(coord, radius=rad * units.deg, catalog='I/355/gaiadr3')
```

# Python用例：Vizier

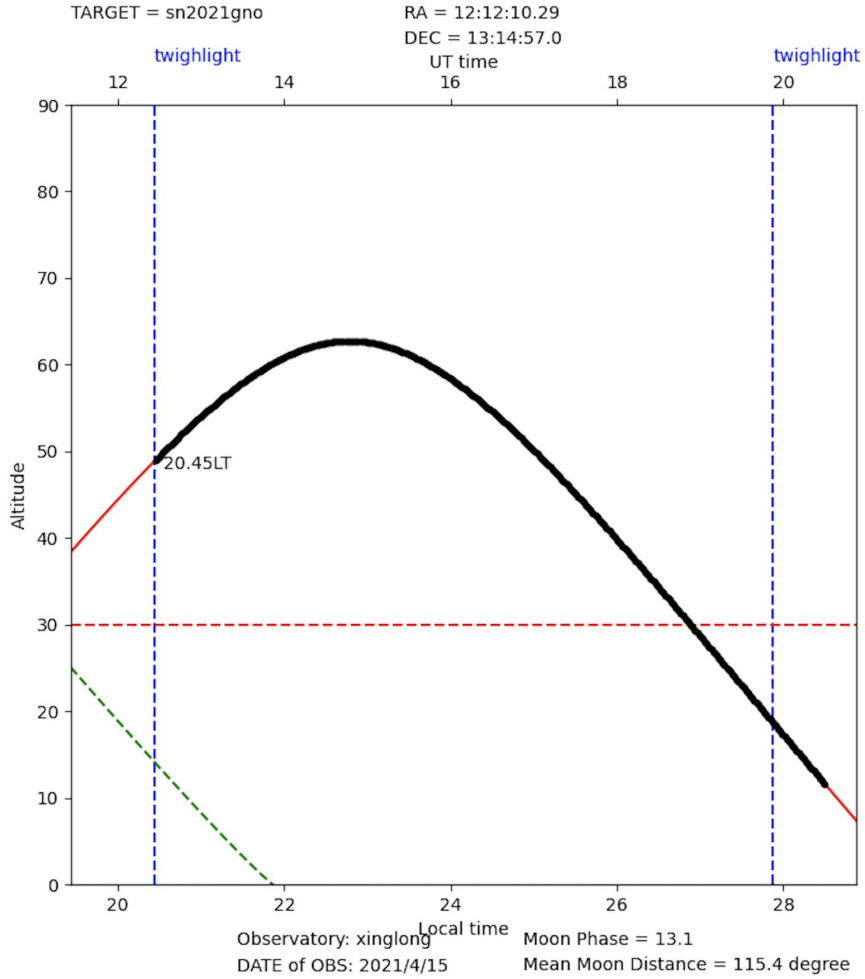
- **Finding chart plot**
  - 根据获得的星表和视场参数画证认图
  - 根据星等来设置圆点的半径大小
  - 旋转，加狭缝位置 ...



# Python用例： ephem

- **Observation plan**
  - **import ephem as ep (https://rhodesmill.org/pyephem/)**
    - 观测台站信息：
      - observatory = ep.Observer()
      - observatory.lon = '117.5750'
      - observatory.lat = '40.3933'
      - observatory.elevation = 950
    - 观测目标信息：
      - star = ep.FixedBody()
      - star._ra = ep.hours(eq.ra)
      - star._dec = ep.degrees(eq.dec)
    - 观测时间信息：
      - observatory.date = '2021-01-04'
    - Twighlight：
      - observatory.horizon = -18
      - twightlight_begin = observatory.next_setting(ep.Sun(), use_center=True)
      - twightlight_end = observatory.next_rising(ep.Sun(), use_center=True)
    - Moon phase/distance：
      - moon = ep.Moon()
      - moon.compute(observatory)
      - distance = ep.separation(star, moon)
      - moon.phase



# Python用例： photutils

- 相比于sextractor的优势
  - 部分功能的分解实现（找源、定心、天光背景估计...）
  - 输入为HDUList或者array，I/O具有优势
- **from photutils.detection import DAOStarFinder**

```
mean, median, std = sigma_clipped_stats(data, sigma=3.0)  # estimate global sky
data = data - median  # remove the sky background
daofind = DAOStarFinder(fwhm=fwhm, threshold=threshold * std, ratio=1, exclude_border=True, roundlo=roundlo,
                        roundhi=roundhi)
sources = daofind(data, mask=mask)  # detect the stars
```

- **from photutils.centroids import centroid_sources, centroid_1dg**

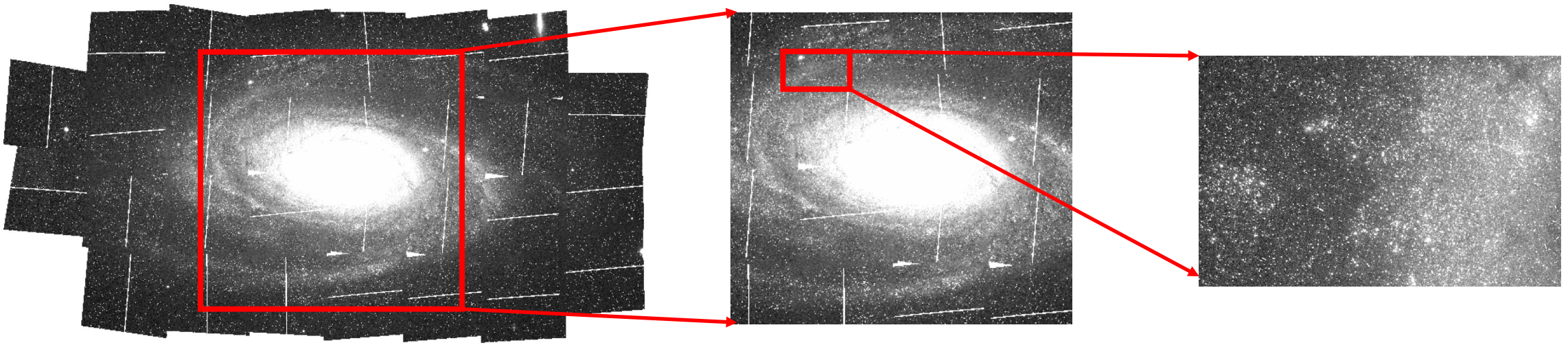
```
x_new, y_new = centroid_sources(data, sources['xcentroid'], sources['ycentroid'], box_size=11,
                                centroid_func=centroid_1dg)  # re-calculate the accurate positions
positions_new = list(zip(x_new, y_new))

radii = np.linspace(1, 20, 20)
apertures = [CircularAperture(positions_new, r=r) for r in radii]
phot_table = aperture_photometry(data, apertures)  # aperture photometry
```

# Python用例： SkyCoord

- **from astropy.coordinates import SkyCoord**
- **坐标系统转换**
  - c = SkyCoord(ra=ra * units.degree, dec=dec * units.degree, frame='icrs',
    pm_ra_cosdec=pmra * units.mas / units.yr,
    pm_dec=pmdec * units.mas / units.yr,
    obstime=Time(2016.0, format='jyear', scale='tcb'),
    distance=Distance(parallax=parallax * units.mas))
    - c.galactic.l.degree, c.galactic.b.degree
- **历元转换**
  - c_new = c.apply_space_motion(Time(60718.90593, format='mjd', scale='tcb'))
    - c_new.ra.degree, c_new.dec.degree

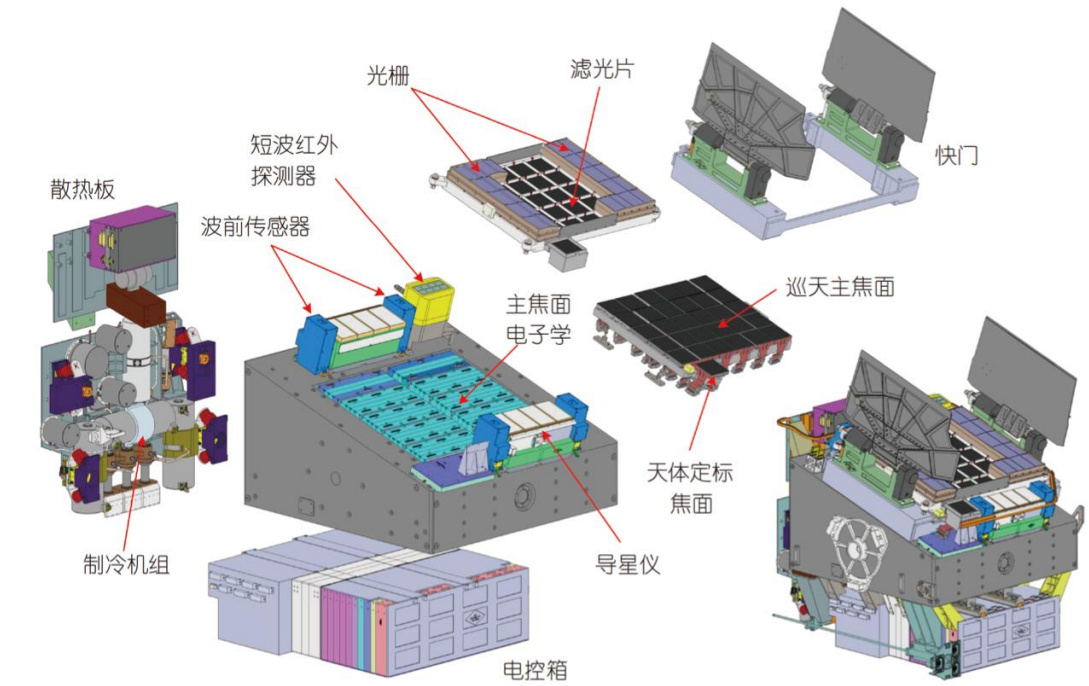
# Python用例： astrodrizzle

- **from drizzlepac import astrodrizzle**
- **HST图像合成**
  - 利用HST同一波段图像drizzle合成，并重采样到需要的分辨率下，合成后的图像的流量单位是e-/s，合成后产生sci图像（天光背景已经扣除）。

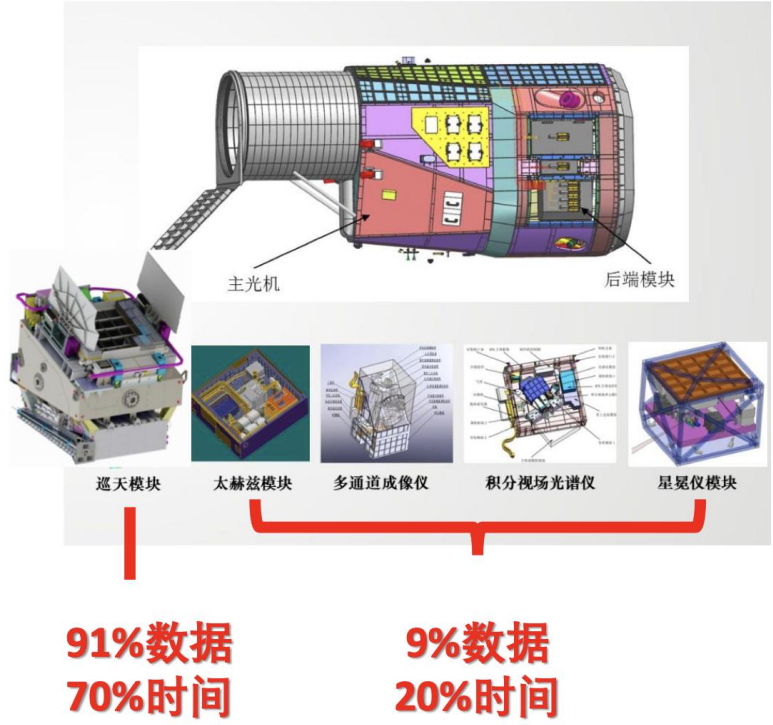


29幅F435W波段的HST图像合成出的M81中心区域图像

# 中国巡天空间望远镜（CSST）




- 载人空间站巡天光学望远镜（CSST）
  - 2米口径空间望远镜，是我国空间天文的旗舰级项目
  - 在轨运行十年间将对40%以上的天空进行高空间分辨率、高灵敏度巡天观测，获取百亿天体的多色图像及无缝光谱数据。



# 主巡天多色成像数据

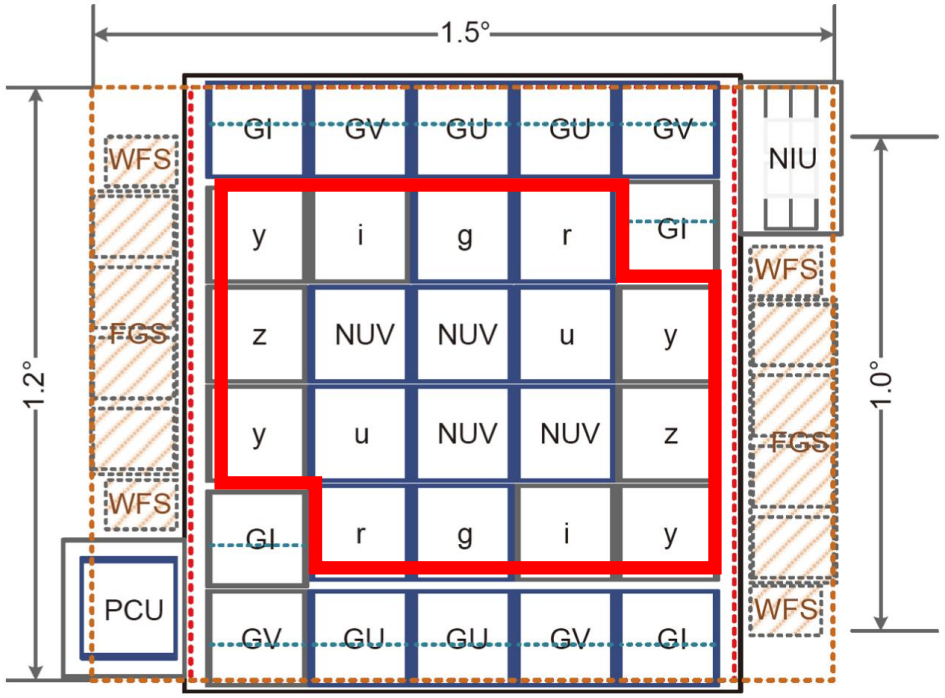



图 5　巡天模块焦面布局

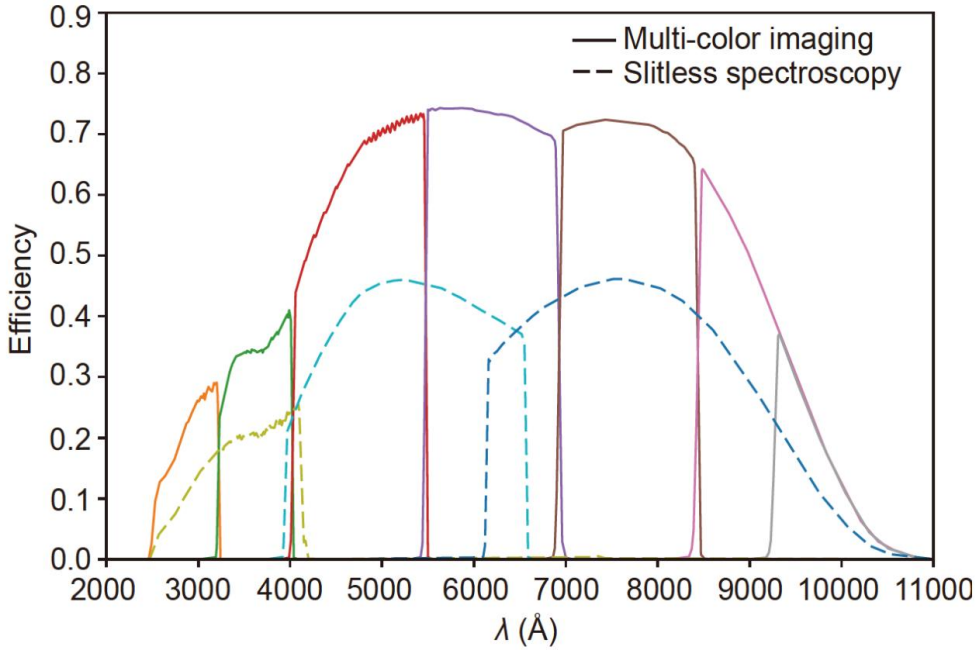


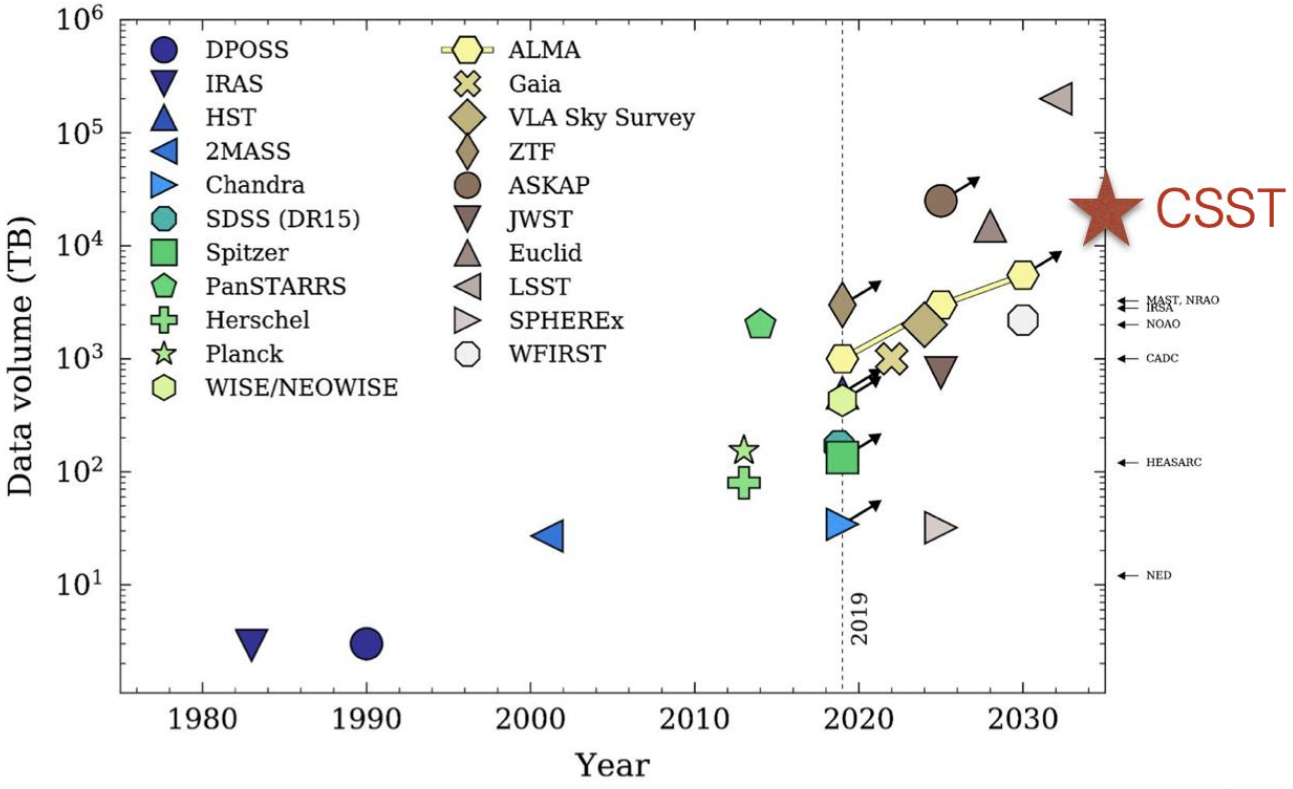

图 7　CSST巡天观测各波段系统通光效率

**Table 3**　Key parameters of the CSST survey

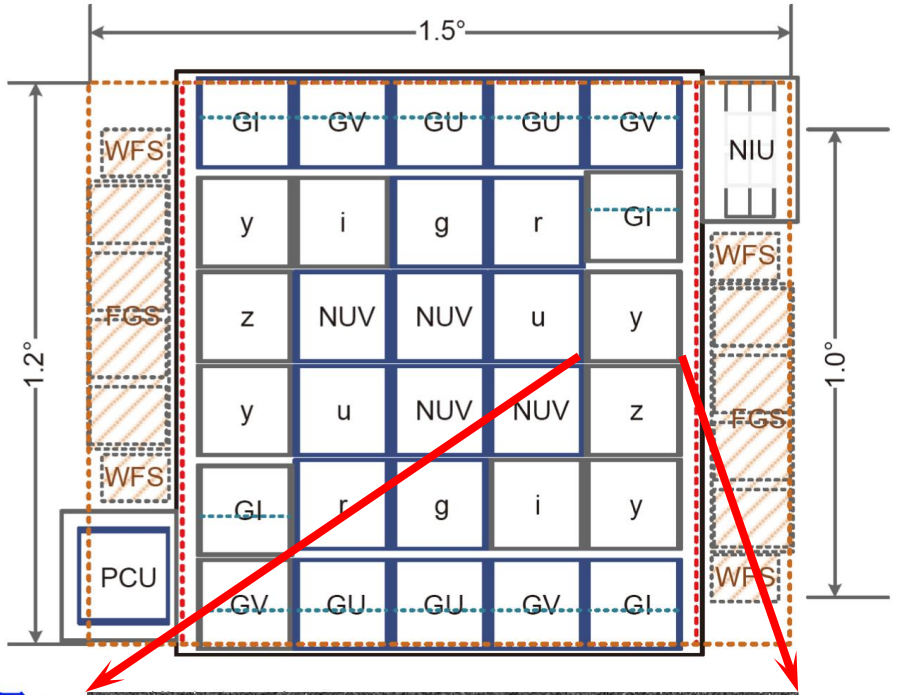
| 观测手段 | 天区面积(平方度) | 曝光时间(s) | 极限星等(点源, 5σ, AB星等) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | NUV[a)] | u | g | r | i | z | y[a)] |
| 多色成像 | 17500 | 150×2 | 25.4 | 25.4 | 26.3 | 26.0 | 25.9 | 25.2 | 24.4 |
| | 400 | 250×8 | 26.7 | 26.7 | 27.5 | 27.2 | 27.0 | 26.4 | 25.7 |

From Zhan 2021

# CSST多色成像数据量估计




- 单个探测器尺寸为9k*9k像素，像元张角0.074角秒，覆盖天空范围约11*11角分。原始图像大小（未压缩）约为170M
- 预计10年巡天观测中，科学观测与在轨定标会有>100万次的曝光。多色成像的原始数据量大约为：170*18*100万 = 3PB
- 按照传统巡天数据处理结果估计，最终需要保存并进入数据库的数据产品至少应为原始数据量的5—10倍以上
- 正常巡天观测需要在24小时内处理约300次曝光的数据，获得图像和高精度星表等产品


1.5°
1.2°
1.0°
WFS
FGS
WFS
PCU
GI GV GU GU GV
y i g r GI
z NUV NUV u y
y u NUV NUV z
GI r g i y
GV GU GU GV GI
NIU
WFS
FGS
WFS

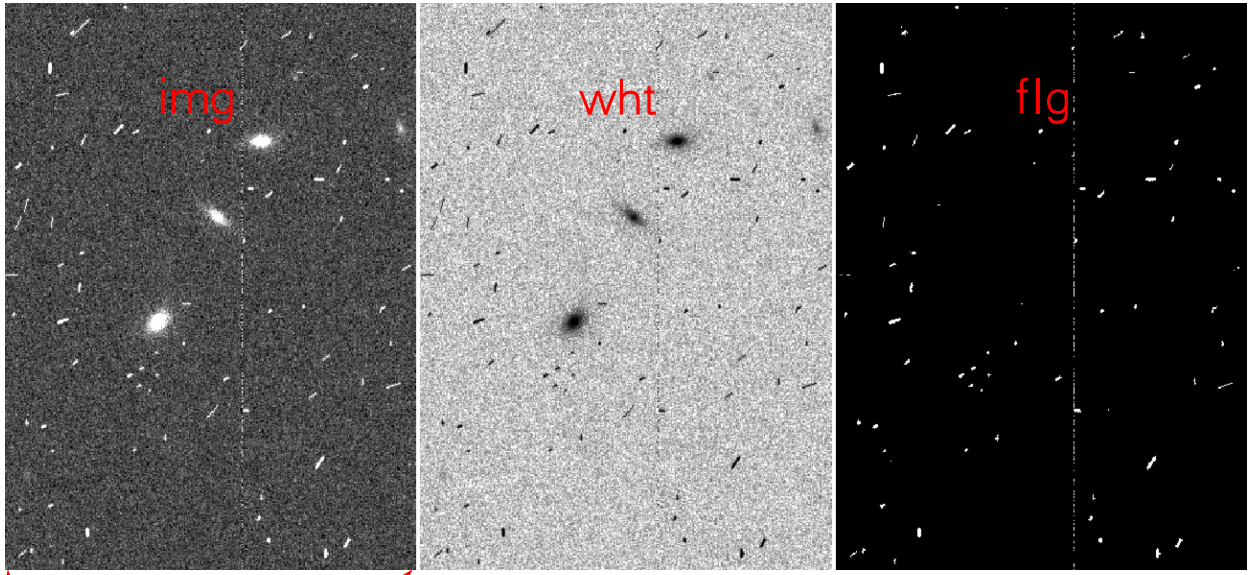
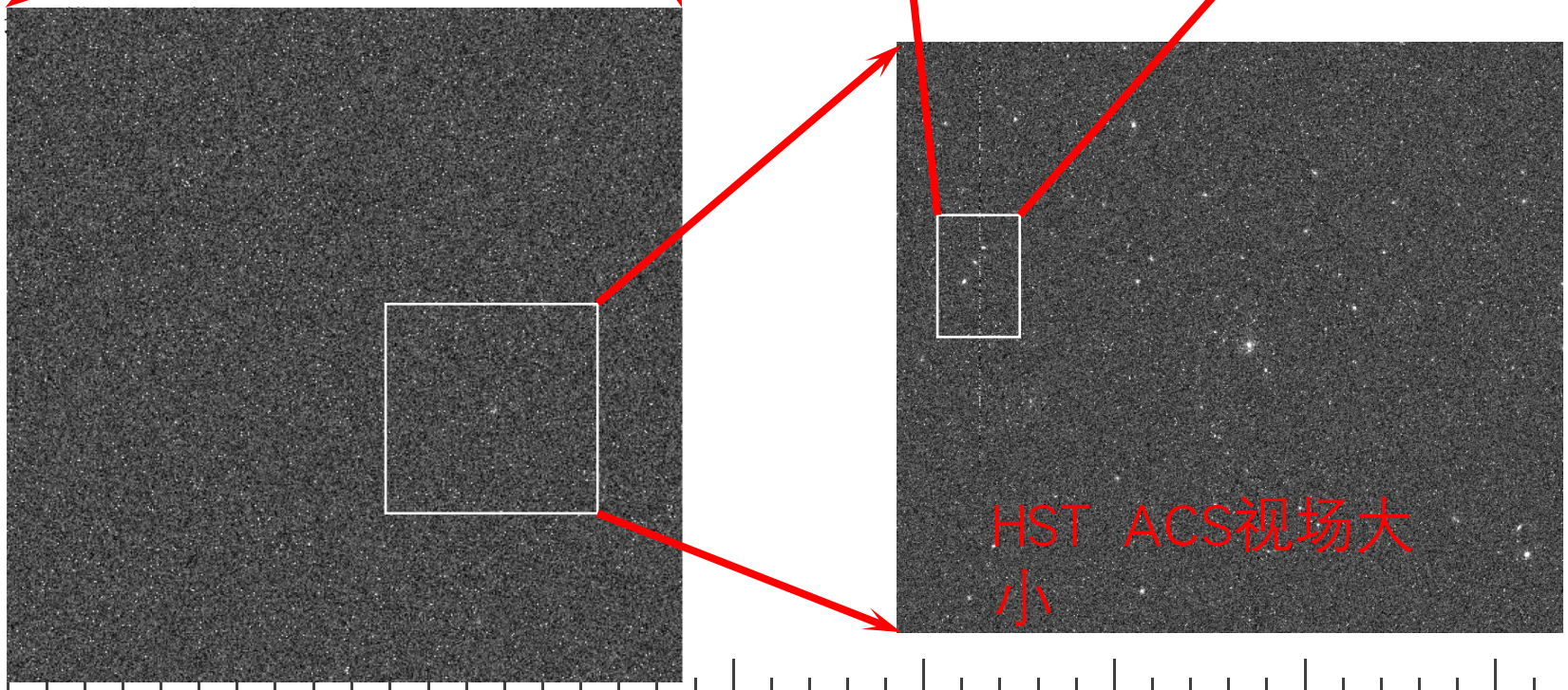
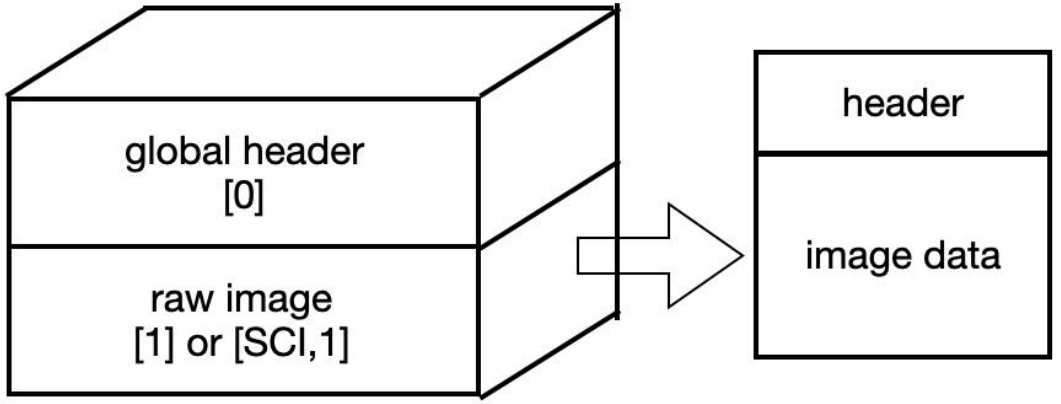

img
wht
flg

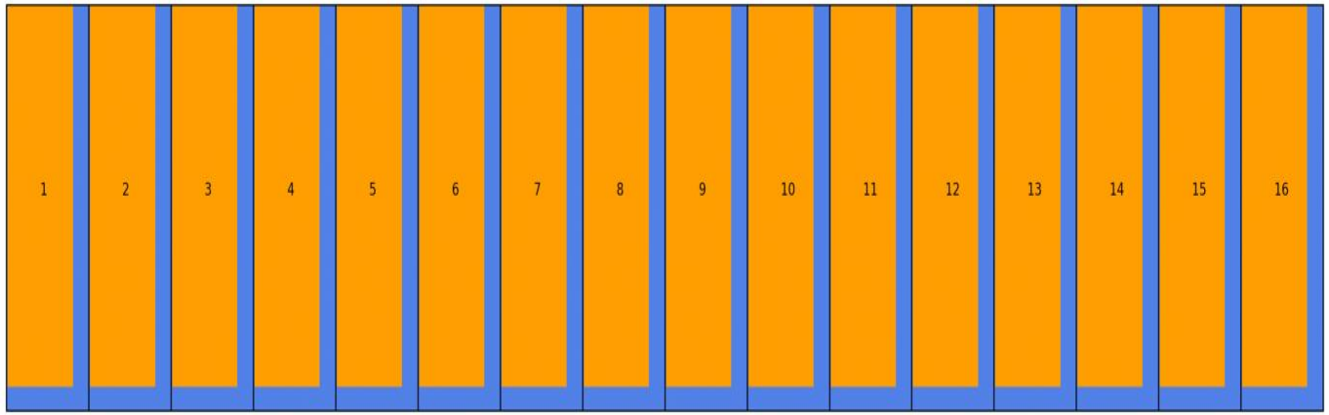
图 5


HST ACS视场大小

# 各级数据产品定义

- 0级数据定义（raw data）
  - HDU0：仅有一个header，包括望远镜基本观测信息，如目标，曝光，指向、平台信息等
  - HDU1：包括header和data两个部分，header部分包括探测器编号、滤光片、增益、初始WCS信息等。data部分包括从探测器读出后，未经修改过的数据矩阵
  - 仿真暂时以16通道读出的CCD为探测器选型
  - 对象数据格式遵循FITS标准，需要保证现有的主流FITS格式文件读写软件



# 各级数据产品定义

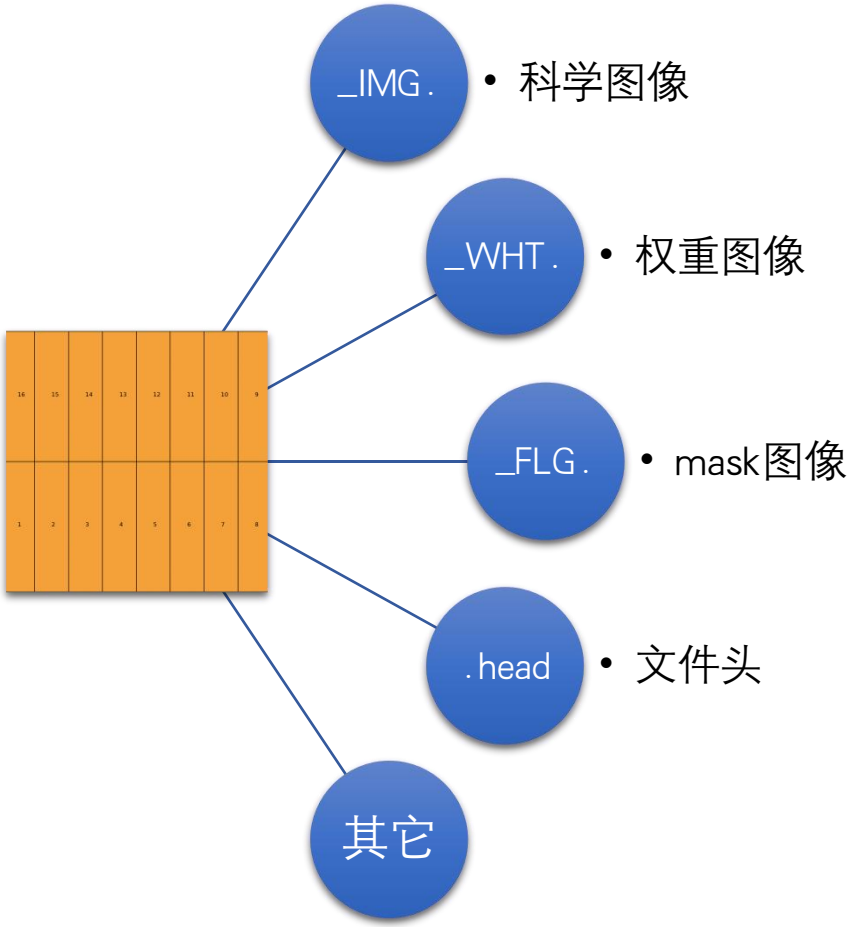
- 1级数据定义
  - data矩阵根据16通道0级数据进行切割和overscan，crosstalk改正后拼合成9232*9216尺寸的图像文件
  - 名称沿用0级数据前缀，前缀最后三位区分每个曝光的每幅图像对应的不同类型文件
  - 产生的数据中会记录数据处理操作的状态、时间、版本和计算结果等重要信息

### Reduced image

- **General Description**
  - Reduced image generated by `L1-MBI` pipeline.
- **Naming convention**
  - **Format**: `CSST_{facility}_{project}_{type}_{t_start}[14]_{t_end}[14]_{obs_id}[11]_{detector}[2]_L{level}[1]_V{version}[2]_IMG.fits`
  - **Example**: `CSST_MSC_MS_SCI_20270626203558_20270626203828_10100000066_06_L0_V01_IMG.fits`
  - **Tips**: Click here for regular expression.

HDU1

| key | value | comment | fill value | type | last modified |
|---|---|---|---|---|---|
| VER_INST | '0.0.1 ' | Version of instrument processing | '0.0.1 ' | str | C7 |
| STM_INST | '2022-12-30T10:18:53' | Time stamp of instrument processing | --- | str | C7 |
| STA_INST | 0 | 0=done 1=wrong | 1 | int | C7 |
| STA_BIAS | 0 | Status flag for bias frame correction | 1 | int | C7 |
| STA_DARK | 0 | Status flag for dark frame correction | 1 | int | C7 |
| STA_FLAT | 0 | Status flag for flat frame correction | 1 | int | C7 |
| SKY_BKG | 0.1 | Estimated sky background (e-/s per pixel) | -9999 | float | C7 |
| SKY_RMS | 10.0 | Standard dev of frame background (ADU) -> e-/s | -9999 | float | C7 |
| SATURATE | 1833.333333333333 | The flux limit of saturated pixel (e-/s) | -9999 | float | C7 |
| STA_CTE | 0 | Status flag for CTE correction | 1 | int | C7 |
| STA_SAT | 0 | Status flag for satellite correction | 1 | int | C7 |
| STA_CRS | 0 | Status flag for cosmic rays mask | 1 | int | C7 |
| CRCOUNT | 66791 | Cosmic rays counts | -9999 | int | C7 |



# 各级数据产品定义

Table Browser for 1: CSST_MSC_MS_SCI_20241219054224_20241219054454_10...

| | R20 | R50 | R90 | X_PSF | Y_PSF | RA_PSF | DEC_PSF | Chi2_PSF | Flux_PSF | FluxErr_PSF | Mag_PSF | MagErr_PSF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1878 | 0.047131 | 0.091372 | 0.206372 | 4189.91023 | 6530.17699 | 89.79943 | 24.30308 | 0.00012 | 237.32... | 0.736266 | 16.5617 | 0.003368 |
| 1064 | 0.049986 | 0.090532 | 0.202645 | 3000.05648 | 3884.5845 | 89.86445 | 24.30227 | 0.000195 | 178.22... | 0.622533 | 16.8726 | 0.003792 |
| 2129 | 0.04895 | 0.088252 | 0.199944 | 9125.08933 | 7361.62597 | 89.73825 | 24.38871 | 0.000155 | 120.84... | 0.501409 | 17.2944 | 0.004505 |
| 398 | 0.048555 | 0.087759 | 0.195062 | 4394.45777 | 1421.10977 | 89.90269 | 24.34838 | 0.00038 | 112.45... | 0.482108 | 17.3725 | 0.004655 |
| 1740 | 0.049792 | 0.087043 | 0.196243 | 605.2833 | 6028.61546 | 89.84172 | 24.24003 | 0.000449 | 107.34... | 0.471093 | 17.4231 | 0.004765 |
| 2348 | 0.047357 | 0.088581 | 0.198249 | 5039.28464 | 8136.83739 | 89.75881 | 24.30594 | 0.000462 | 86.1505 | 0.419992 | 17.6619 | 0.005293 |
| 379 | 0.050374 | 0.086474 | 0.201406 | 6414.62164 | 1370.29252 | 89.88571 | 24.38663 | 0.000844 | 75.019 | 0.389678 | 17.8121 | 0.00564 |
| 2377 | 0.047442 | 0.087115 | 0.177712 | 1676.3759 | 9229.18987 | 89.76638 | 24.23412 | 0.023273 | 71.6024 | 0.384572 | 17.8627 | 0.005831 |
| 2386 | 0.047774 | 0.088566 | 0.193649 | 2593.53767 | 9186.09074 | 89.75908 | 24.25164 | 0.001789 | 49.9129 | 0.315126 | 18.2545 | 0.006855 |
| 1434 | 0.048627 | 0.087492 | 0.197712 | 6069.59534 | 4992.85055 | 89.81427 | 24.35075 | 0.004705 | 43.8118 | 0.29646 | 18.396 | 0.007347 |
| 1190 | 0.046538 | 0.089603 | 0.193338 | 2698.01554 | 4207.81364 | 89.8605 | 24.29399 | 0.004061 | 42.5338 | 0.290915 | 18.4282 | 0.007426 |
| 95 | 0.048444 | 0.08812 | 0.195478 | 8802.49997 | 395.01362 | 89.88447 | 24.43927 | 0.000685 | 41.8331 | 0.285664 | 18.4462 | 0.007414 |
| 490 | 0.047868 | 0.088254 | 0.190462 | 5102.68553 | 1720.94817 | 89.8902 | 24.35921 | 0.005806 | 40.8447 | 0.285336 | 18.4722 | 0.007585 |
| 568 | 0.050619 | 0.086785 | 0.196993 | 6292.54774 | 2003.68913 | 89.87377 | 24.3792 | 0.006022 | 39.1157 | 0.279423 | 18.5191 | 0.007756 |
| 2694 | 0.045164 | 0.089972 | 0.190765 | 1860.04861 | 8799.15071 | 89.77358 | 24.24105 | 0.005557 | 38.9008 | 0.277791 | 18.5251 | 0.007753 |
| 2463 | 0.045821 | 0.089618 | 0.191316 | 1715.91339 | 8616.20963 | 89.77863 | 24.23983 | 0.005639 | 34.6025 | 0.262634 | 18.6522 | 0.008241 |
| 152 | 0.049623 | 0.085053 | 0.193512 | 4209.7095 | 573.39797 | 89.92178 | 24.3518 | 0.005927 | 33.2612 | 0.257446 | 18.6952 | 0.008404 |

- 2级数据定义
  - 单次曝光测光星表
    - FITS Table格式，采用后缀_CAT.fits，包括目标天体的位置、流量、星等和形态等125个参数
    - 对1级数据对象进行天光背景扣除、点扩散函数（PSF）构建、天体探测、测光和模型拟合等一系列处理，所产生的星表文件以及处理过程中生成的PSF轮廓、天光背景图像等文件
  - 多次曝光多波段合并星表
    - 把观测天区分成等面积的小天区（每个小天区称为一个BRICK，目前暂定为0.25*0.25度），把同属一个BRICK的不同探测器的单星表通过交叉匹配、去重、合并等步骤，得到合并星表
  - 其它二级数据产品

**L2 Data Model**

**Catalog**

### File contents

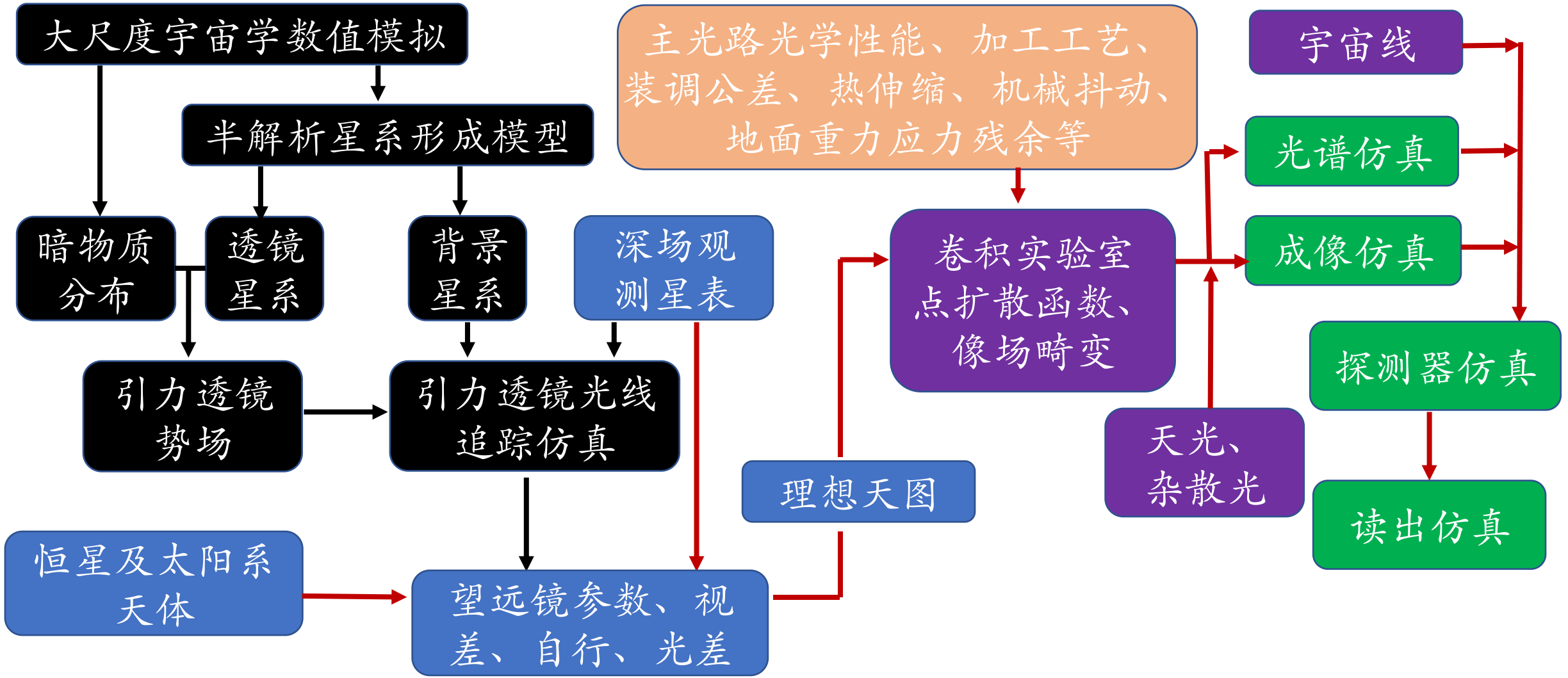
| HDU | data | note |
|---|---|---|
| HDU0 | None | PrimaryHDU |
| HDU1 | Table | ImageHDU |

### HDU0

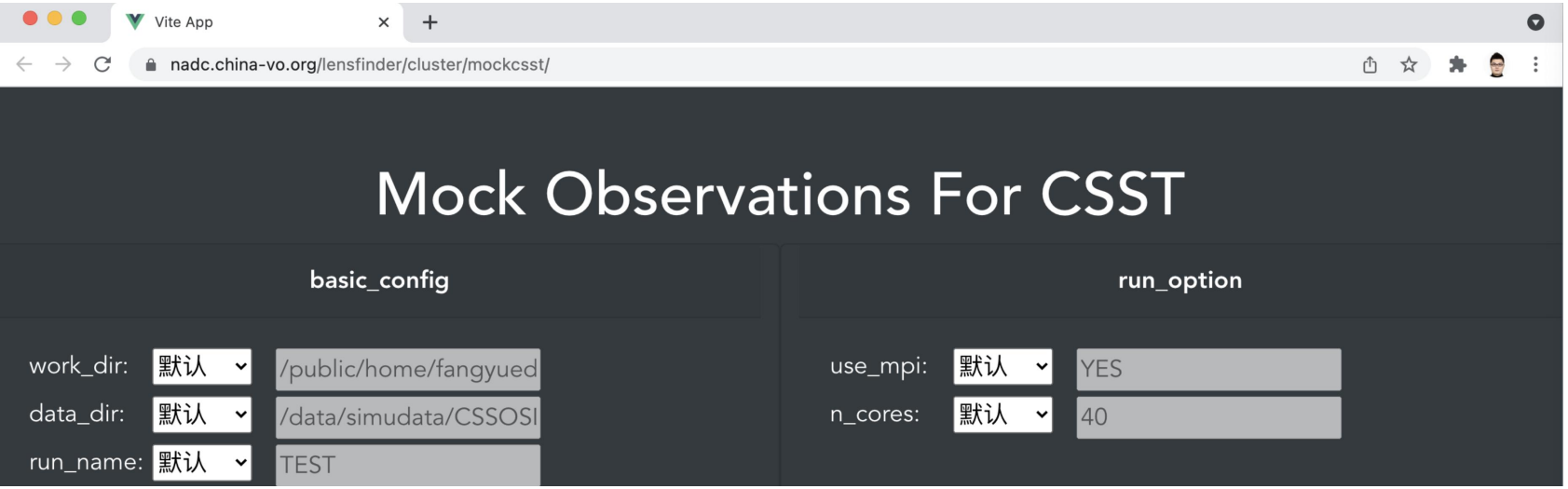
#### Header of photometry

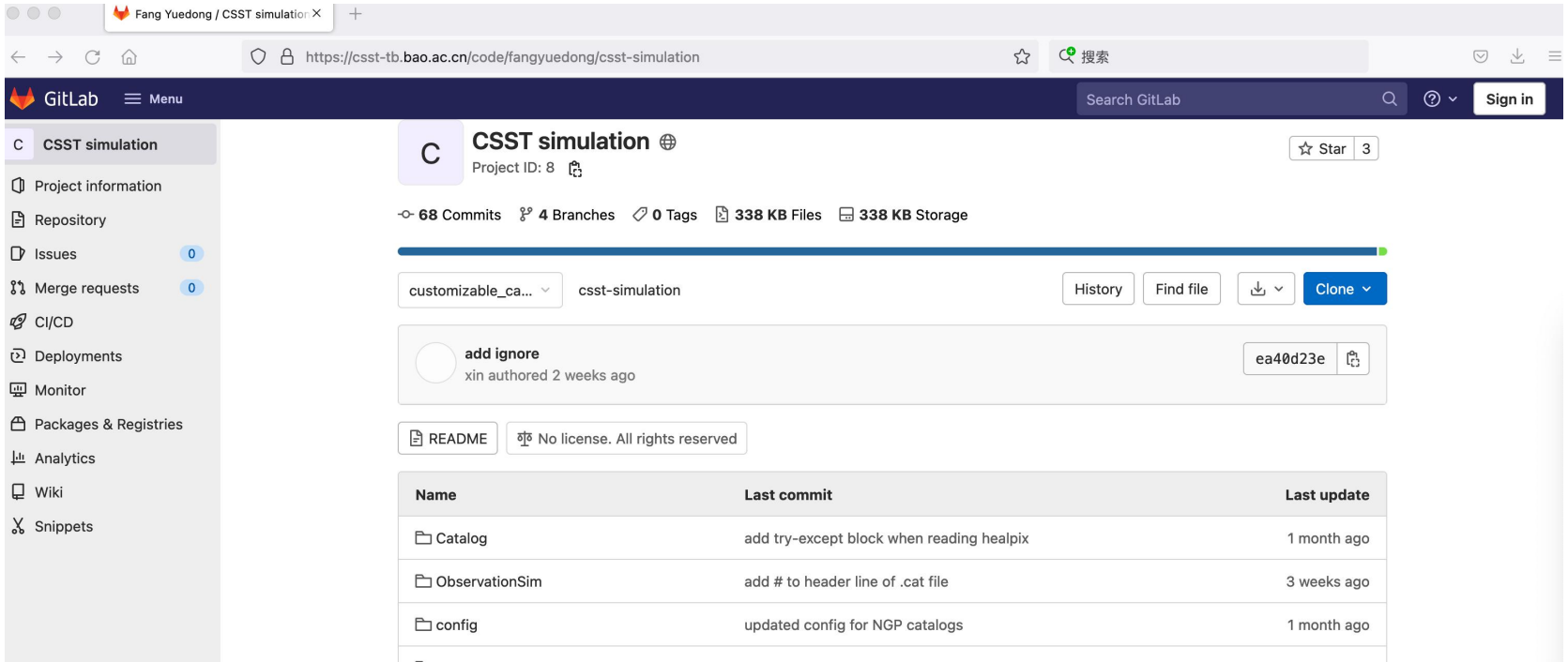
| keyword | value | comment | fill value | type | last modified |
|---|---|---|---|---|---|
| APERSIZE | '3,4,5,6,8,10,13,16,20,25,30,40,' | aperture radii in pixels | '3,4,5,6,8,10,13,16,20,25,30,40,' | str | C7 |
| NS_APER | 75 | number of stars used in aperture correction | 0 | i8 | C7 |
| APCOR0 | -0.06074262037873268 | mag correction for aperture #0 | 0.0 | f32 | C7 |
| APERR0 | 0.0 | mag correction error for aperture #0 | 0.0 | f32 | C7 |
| APCOR1 | -0.01975813694298267 | mag correction for aperture #1 | 0.0 | f32 | C7 |
| APERR1 | 0.0 | mag correction error for aperture #1 | 0.0 | f32 | C7 |
| APCOR2 | 0.0 | mag correction for aperture #2 | 0.0 | f32 | C7 |
| APERR2 | 0.0 | mag correction error for aperture #2 | 0.0 | f32 | C7 |
| APCOR3 | 0.01290098764002323 | mag correction for aperture #3 | 0.0 | f32 | C7 |
| APERR3 | 0.0 | mag correction error for aperture #3 | 0.0 | f32 | C7 |
| APCOR4 | 0.02804811112582684 | mag correction for aperture #4 | 0.0 | f32 | C7 |

# 仿真模拟工作流程


大尺度宇宙学数值模拟
半解析星系形成模型
暗物质分布
透镜星系
背景星系
深场观测星表
引力透镜势场
引力透镜光线追踪仿真
恒星及太阳系天体
望远镜参数、视差、自行、光差
主光路光学性能、加工工艺、装调公差、热伸缩、机械抖动、地面重力应力残余等
卷积实验室点扩散函数、像场畸变
理想天图
天光、杂散光
宇宙线
光谱仿真
成像仿真
探测器仿真
读出仿真

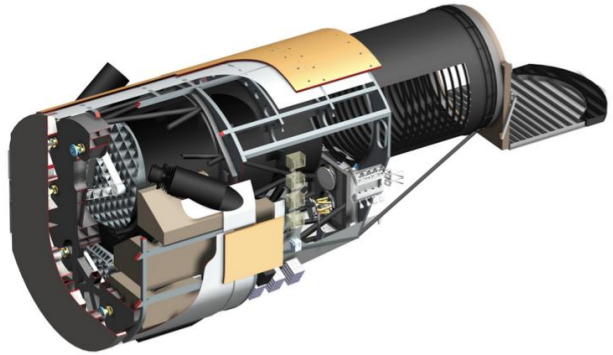
# CSST数据仿真




# CSST 主巡天仿真数据

# Cycle 6 数据产品说明



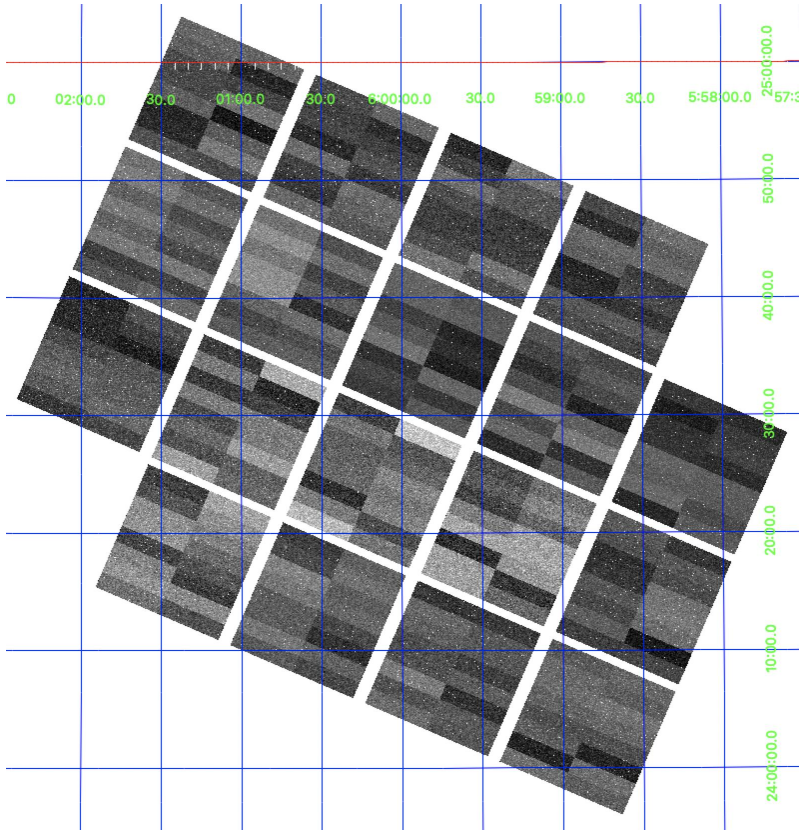

**编写：**

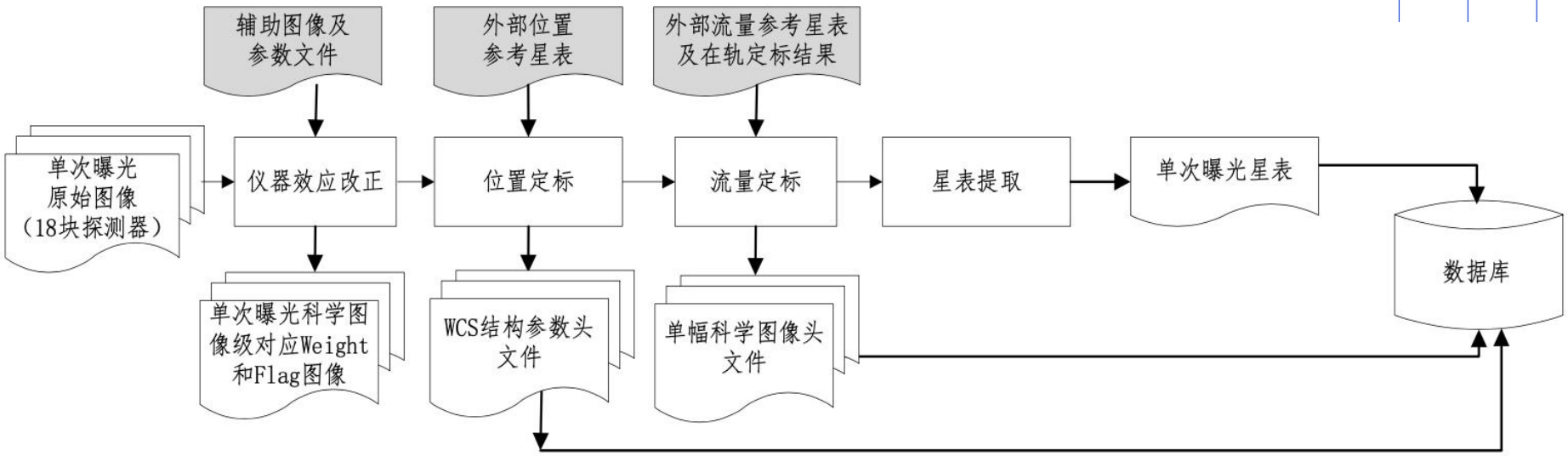
韦成亮，张鑫、刘德子、方越东、孟宪民、班章、罗煜、田浩、李佳东、李晓波、李然、李楠、齐朝祥、李国亮


2023 年 04 月 30 日

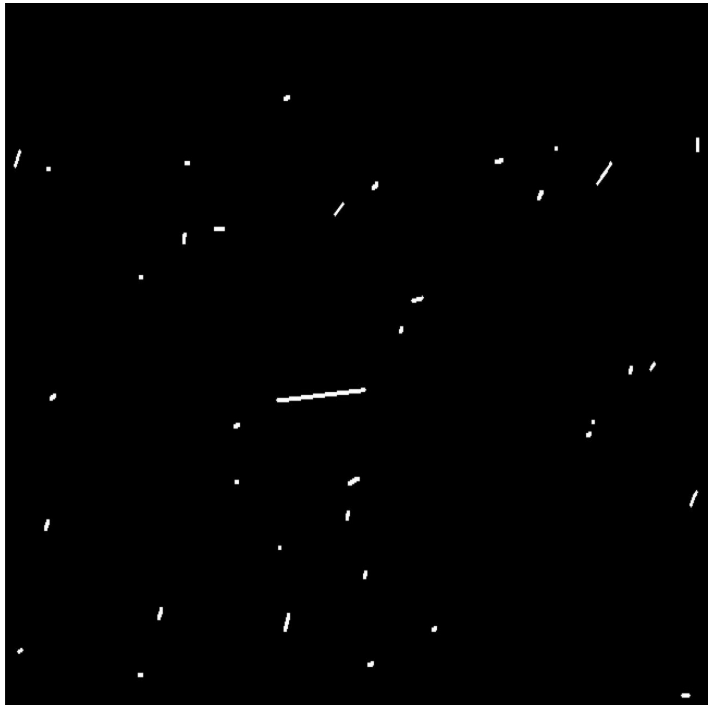
# 数据处理流水线

- 单次曝光数据处理流水线
  - 对从探测器获得的原始数字图像（0级数据）进行加工，最终形成可供天文学家使用的科学数据。产品包括定标后的科学图像及对应的权重和mask图（1级数据），以及单幅星表（2级数据）
    - 仪器效应改正：利用参考图像，改正望远镜和探测器本身特性所造成的图像数据不均匀性
    - 位置定标：利用已知的星表与科学图像中探测到的天体进行交叉匹配，计算图像畸变和指向校正
    - 流量定标：利用已知星表中的流量信息与科学图像中探测到的对应天体流量，计算图像零点、颜色项改正系数等信息
    - 星表提取：在已经完成定标的科学图像上进行天体探测，测量天体的位置、流量和形态参数

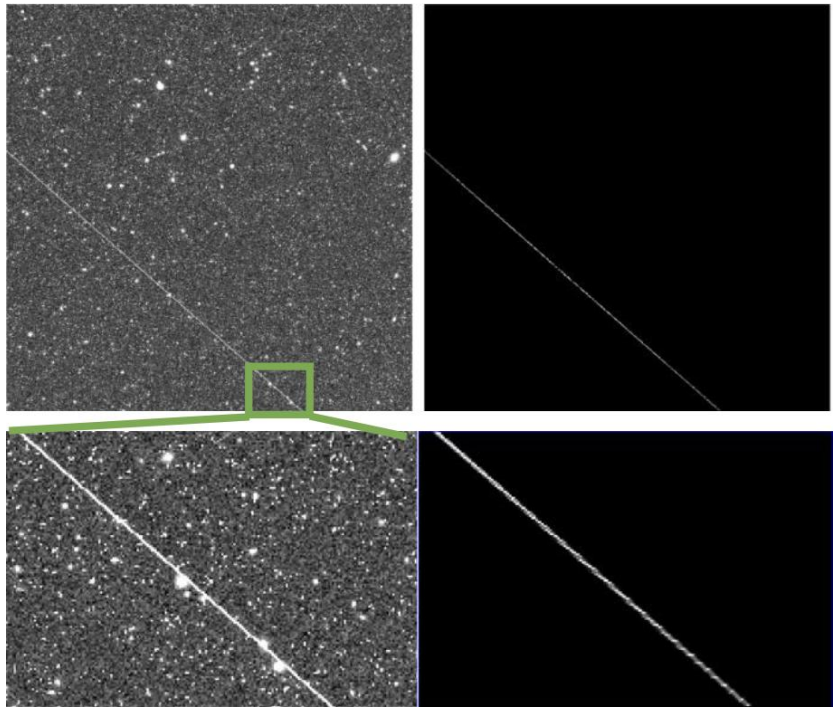




# 仪器效应改正

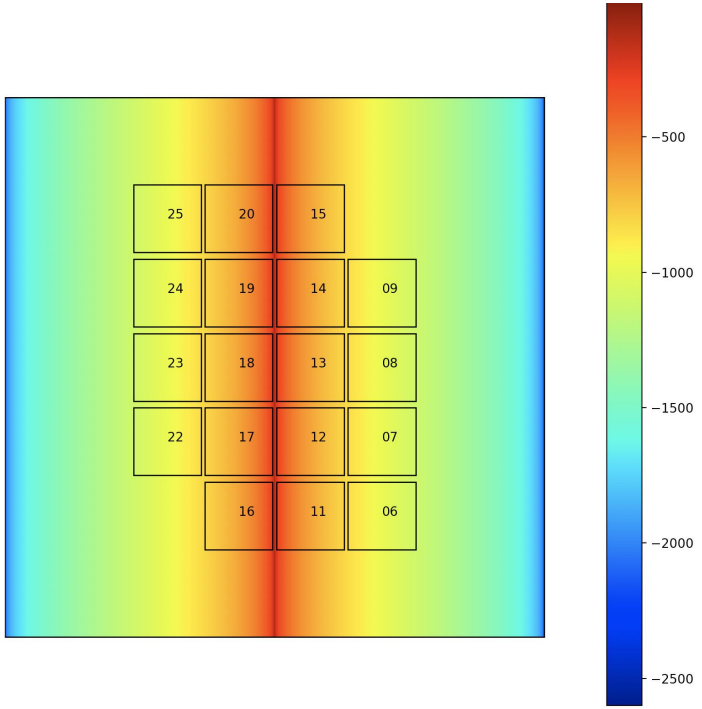


宇宙线探测



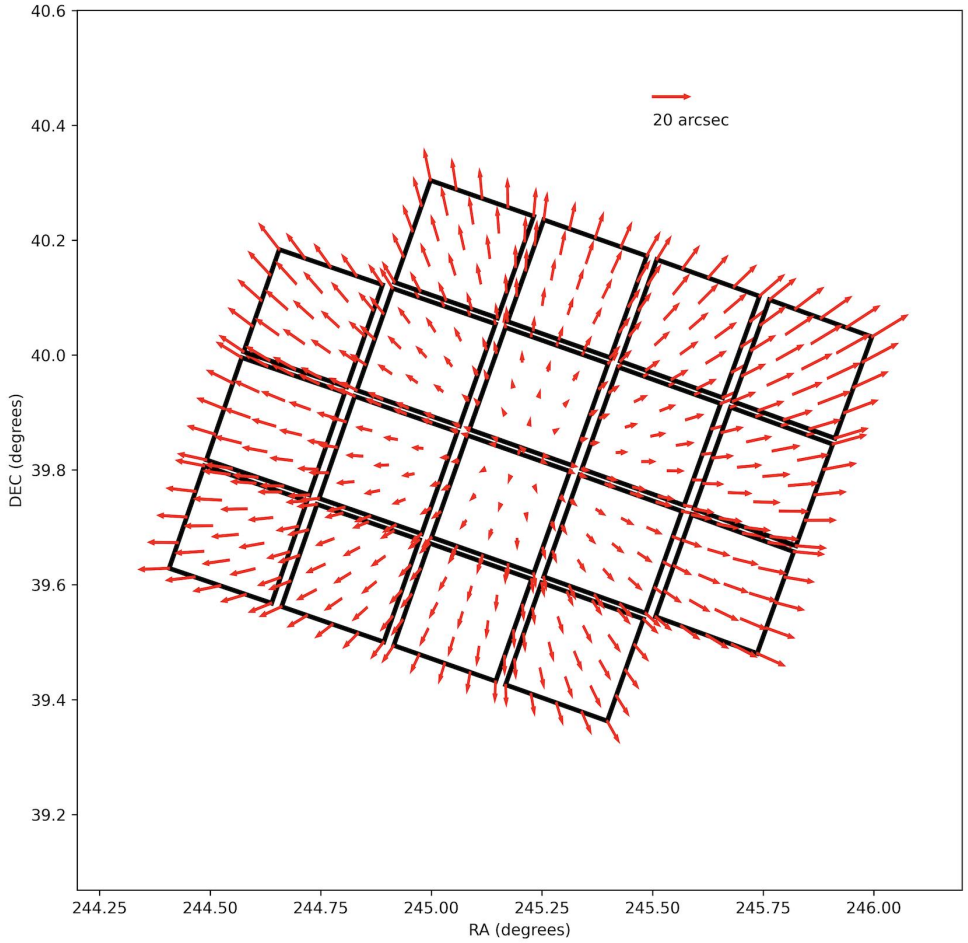
卫星轨迹探测

| flag数值 | 对应问题 |
| --- | --- |
| 0 | 正常 |
| 1 | 坏像素（对光无响应） |
| 2 | 热像素（积分时间内暗流的噪声大于读出噪声） |
| 4 | 温像素（积分时间内暗流的噪声在读出噪声的0.5倍到1倍之间） |
| 8 | 饱和像素 |
| 16 | 宇宙线污染像素 |
| 32 | 卫星轨迹污染像素 |
| 64 | 鬼像污染像素 |
| 128 | 杂散光污染像素 |

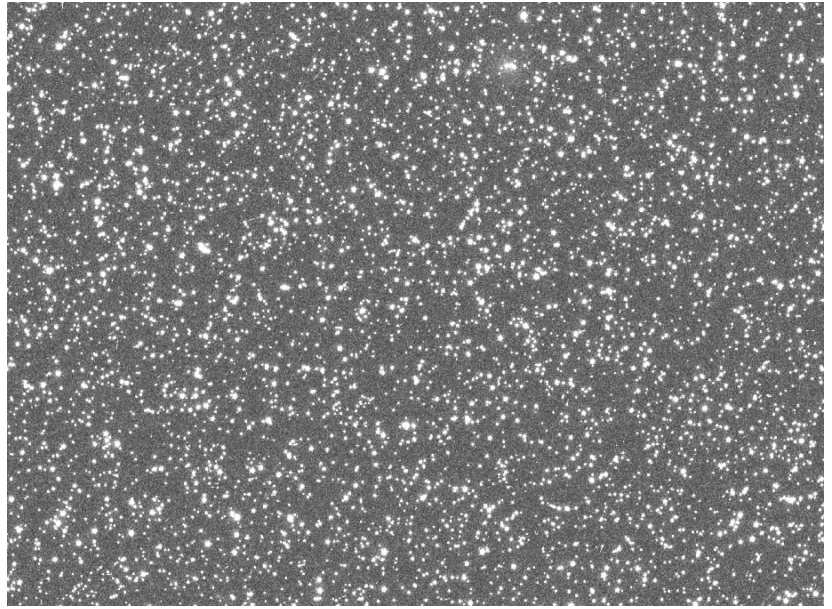



快门效应

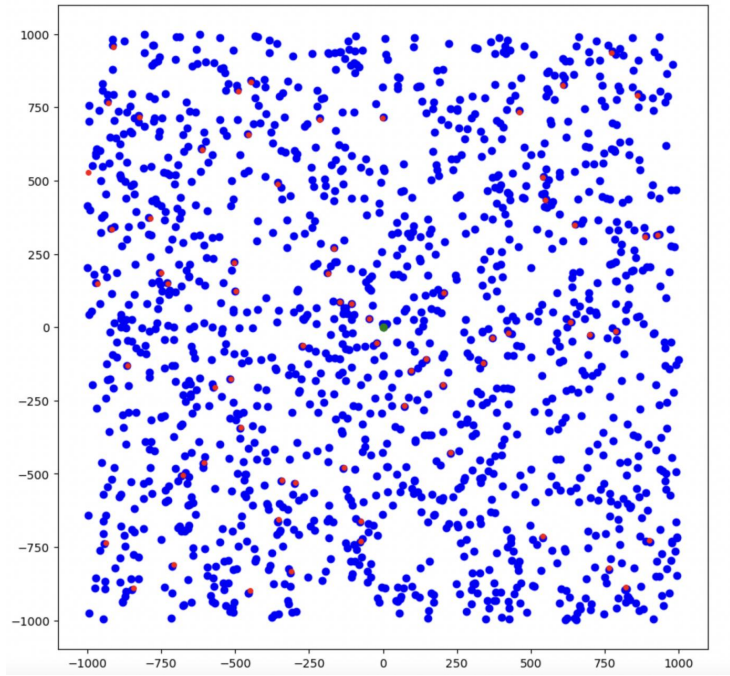
# 恒星位置匹配及畸变拟合



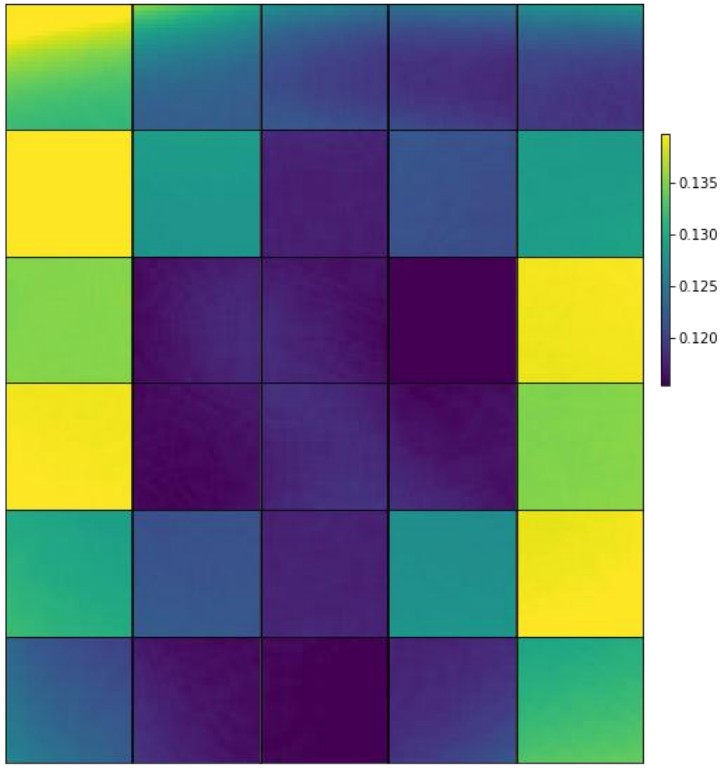

多色成像视场畸变



密集星场

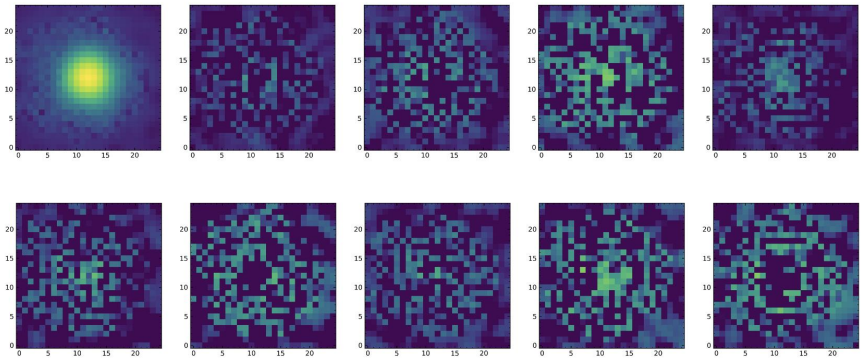


# 测光PSF重构（魏鹏）



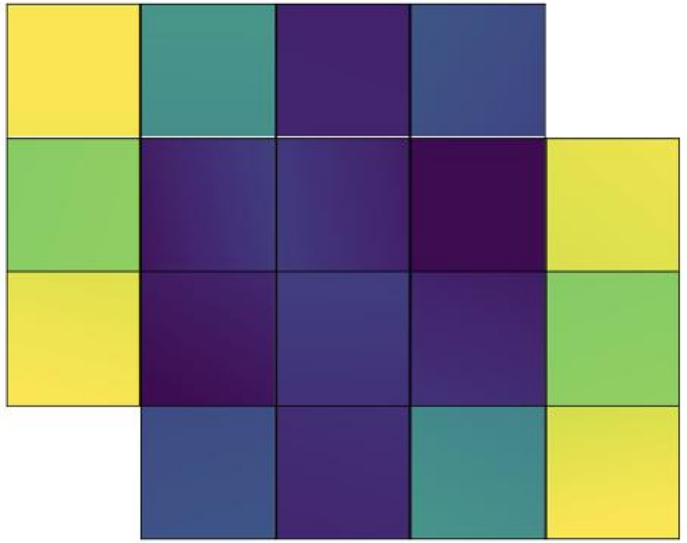

仿真输入PSF

数据仿真团队用软件进行整个焦面上
30 × 30 × 30个PSF仿真采样及内插

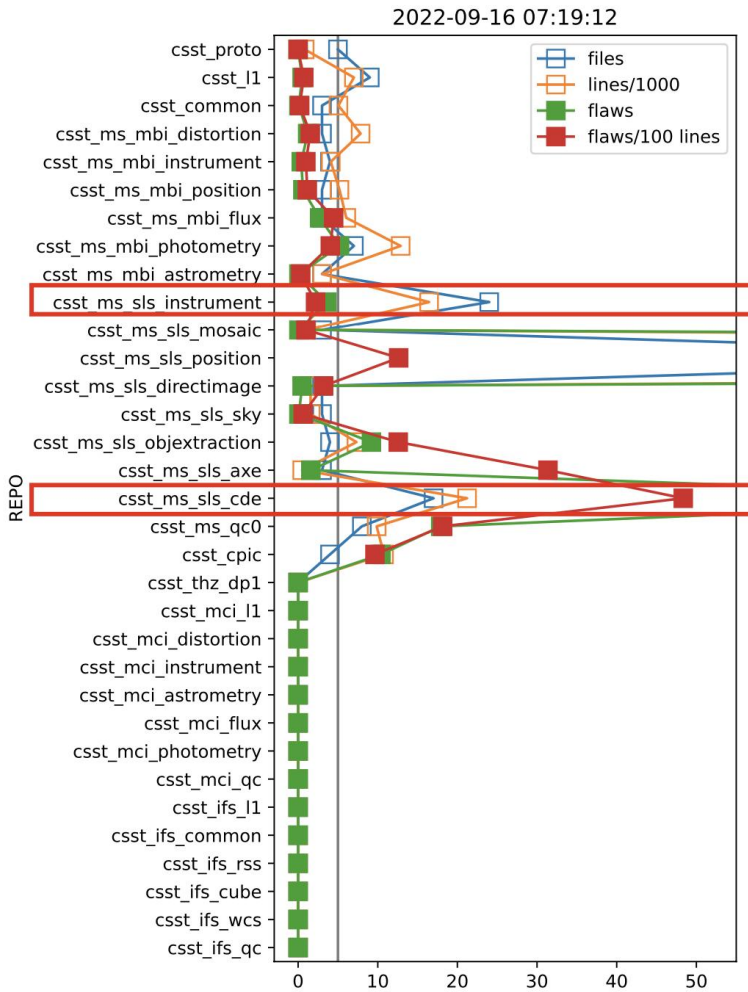


重构PSF

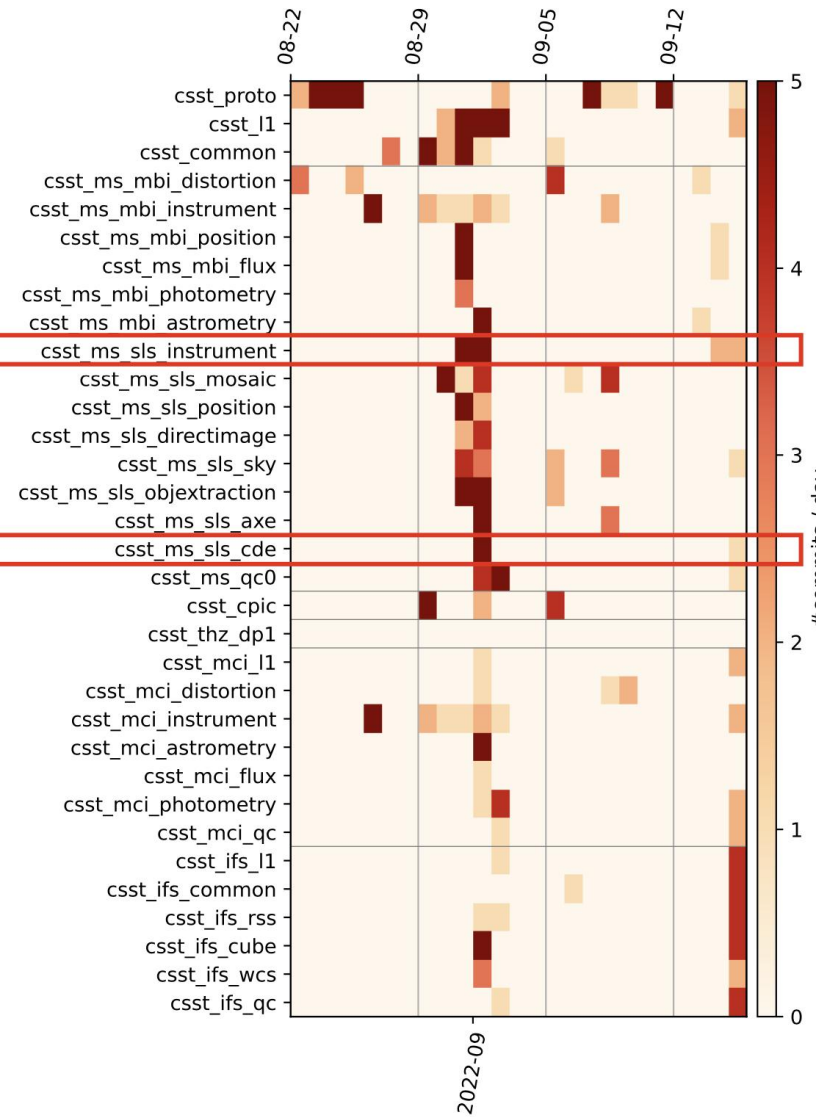
两平方度135次观测数据的多色成像
仿真图像进行了PSF重构，得到PSF
随视场中位置变化的多项式函数



## PyFlake8: PEP8代码缺陷检测

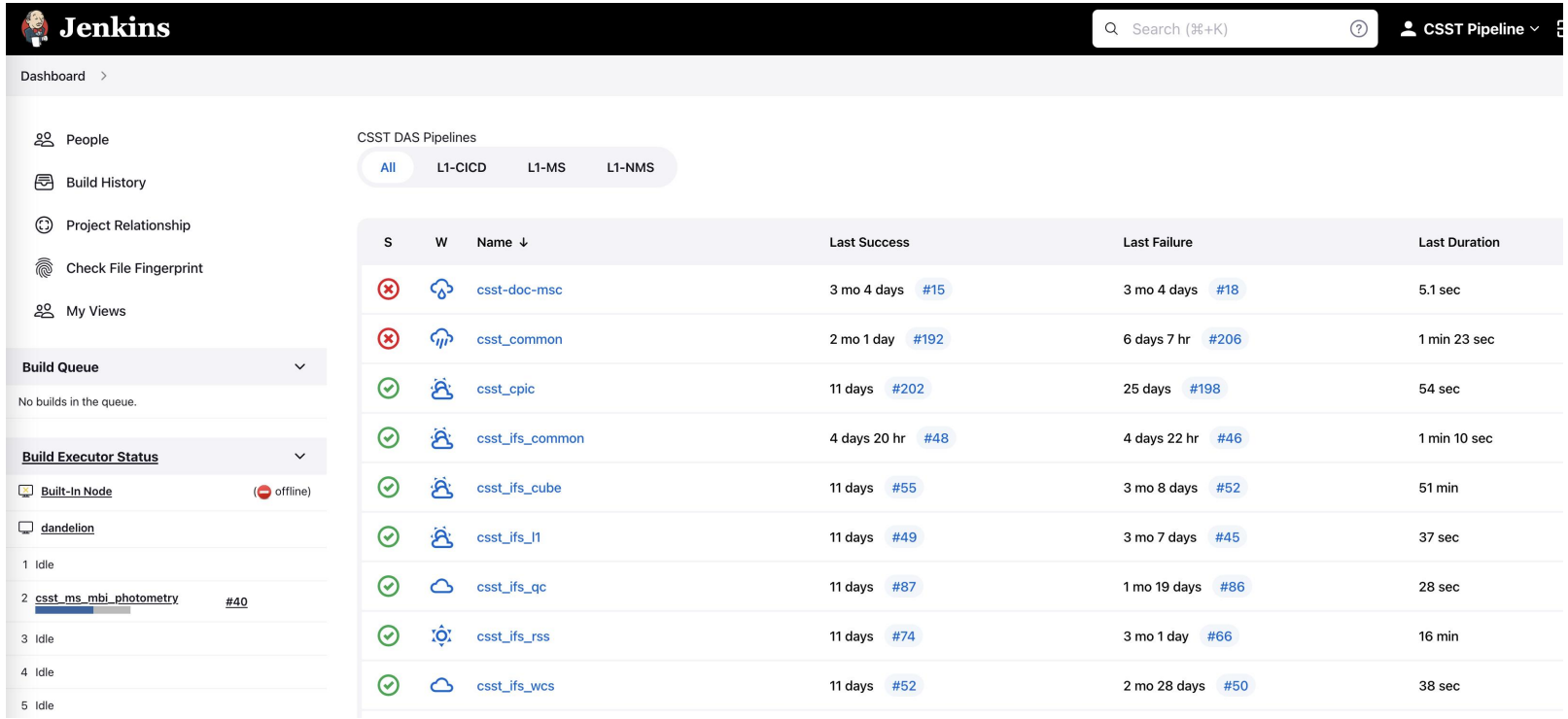



## Git commit 记录



# 基于Jenkins的代码单元测试平台

- 实时监测每个开发者的更新进度
- 检查代码风格：pycodestyle ./**/*.py
- 检查是否符合Numpydoc要求：python −m numpydoc −−validate
- 通过邮件系统自动发送测试结果



# 代码迭代和单元测试

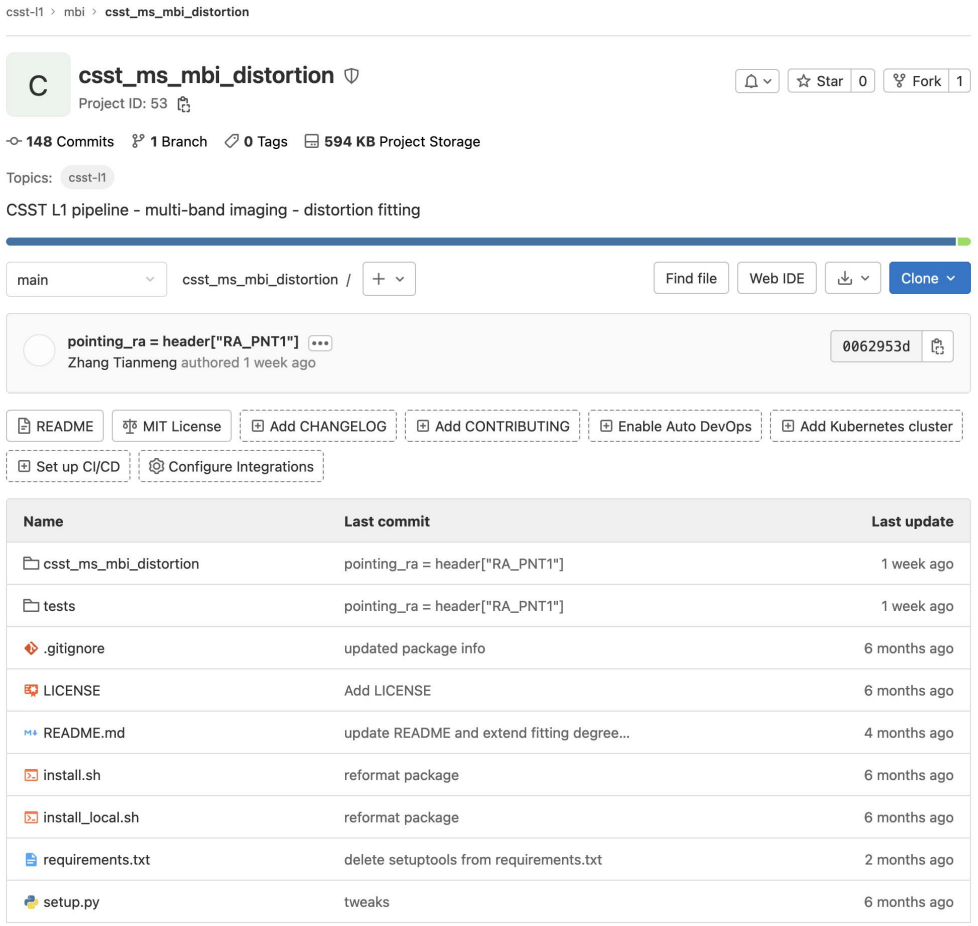
- 利用GitLab进行代码迭代管理
  - 规范化
  - 注释
- 对各模块提出单元测试需求
  - 不同运用场景下的测试
  - 容错、状态返回
  - 覆盖率测试

```
def test_multichips(self):
    """
    Aim
    ---
    Test the calculation of distortion for multiple chips.

    Criteria
    --------
    Get the correct results and all the header files created.

    Details
    -------
    This test is used to calculate for C6 version data.
    The test data are fields of NGP, which have less than 100 gaia stars in each chip.
    """
```

Coverage report: 93%

*coverage.py v7.2.1, created at 2023-03-12 09:46 +0800*

| *Module* | *statements* | *missing* | *excluded* | *coverage* |
|---|---|---|---|---|
| csst_ms_mbi_distortion/__init__.py | 3 | 1 | 0 | 67% |
| csst_ms_mbi_distortion/pvfit.py | 450 | 32 | 0 | 93% |
| test_codestyle.py | 16 | 4 | 0 | 75% |
| tests/test_distortion.py | 67 | 1 | 0 | 99% |
| **Total** | **536** | **38** | **0** | **93%** |



谢谢！