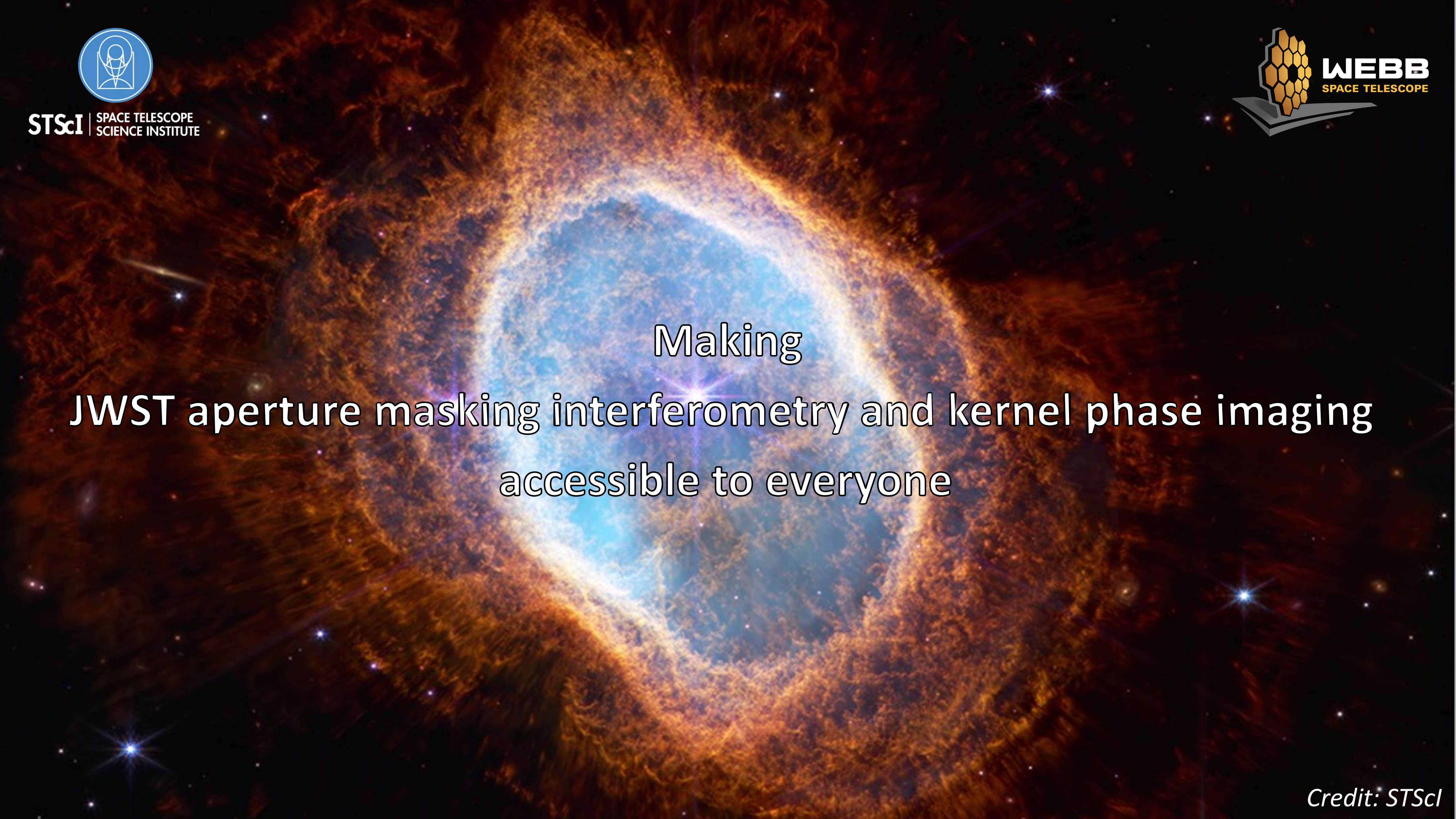
# JWST aperture masking interferometry and kernel phase imaging

**Jens Kammerer**

*Credit: STScI*

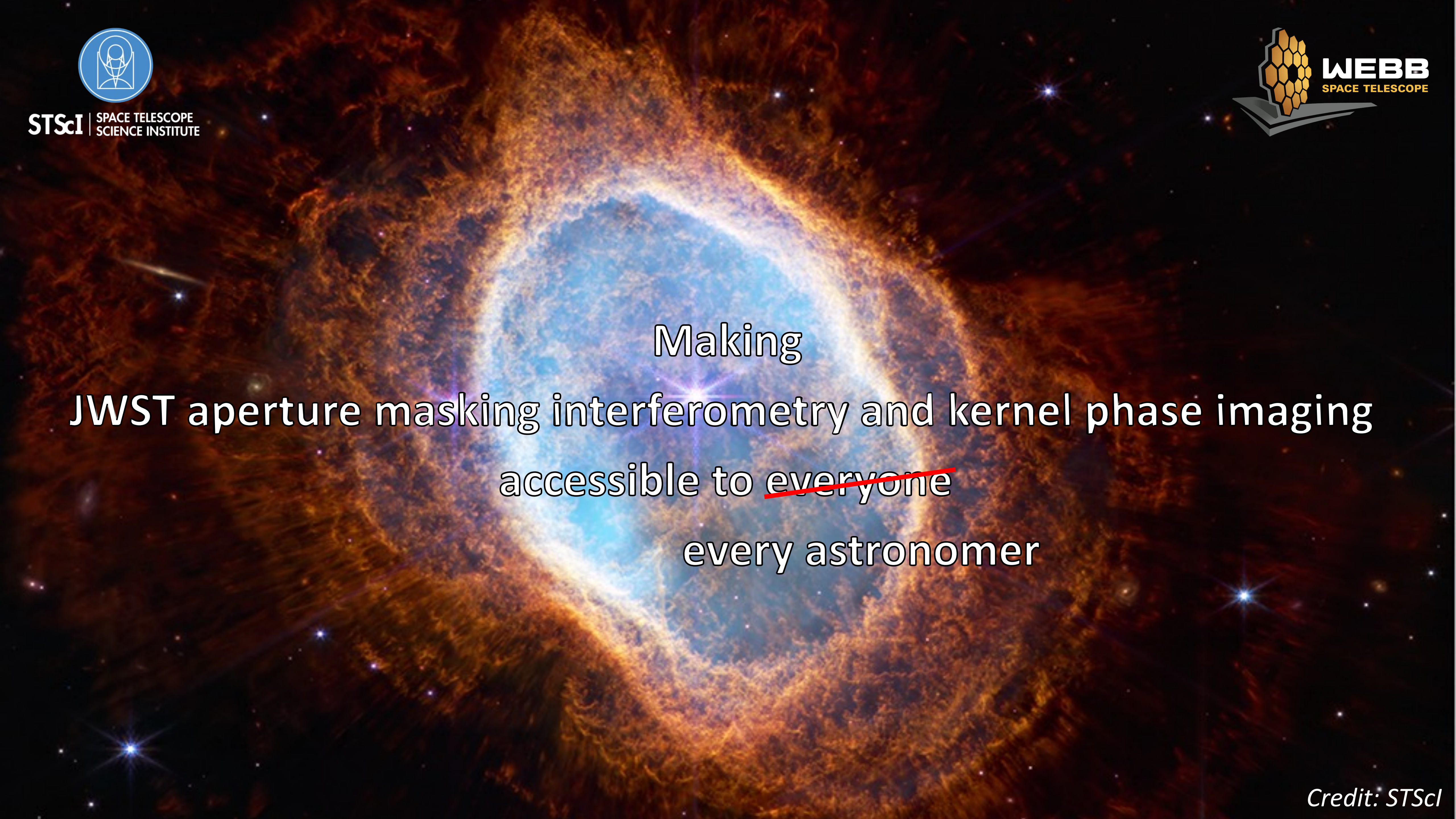


STScI | SPACE TELESCOPE SCIENCE INSTITUTE

WEBB SPACE TELESCOPE

# Making
# JWST aperture masking interferometry and kernel phase imaging
# accessible to everyone

*Credit: STScI*



# Making JWST aperture masking interferometry and kernel phase imaging accessible to ~~everyone~~ every astronomer

*Credit: STScI*

# The JWST opportunity


Credit: SPIE Astro 2022

# What do astronomers want?


Credit: SPIE Astro 2022

# What do astronomers want?

1. **Data reduction**
   From raw data to OIFITS/KPFITS


*Credit: SPIE Astro 2022*

# What do astronomers want?

1. **Data reduction**
   From raw data to OIFITS/KPFITS

2. **Model fitting**
   From OIFITS/KPFITS to companion detection map/contrast curve


*Credit: SPIE Astro 2022*

# What do astronomers want?

1. **Data reduction**
   From raw data to OIFITS/KPFITS

2. **Model fitting**
   From OIFITS/KPFITS to companion detection map/contrast curve

3. **Image reconstruction**
   From OIFITS/KPFITS to scene image


*Credit: SPIE Astro 2022*

# What do astronomers want?

1. **Data reduction**
   From raw data to OIFITS/KPFITS

2. **Model fitting**
   From OIFITS/KPFITS to companion detection map/contrast curve

3. **Image reconstruction**
   From OIFITS/KPFITS to scene image

*Credit: Shutterstock*

# 1. Data reduction

**AMI**

**KPI**

# 1. Data reduction

## AMI

STScI stage 1 & 2 pipelines

+

STScI stage 3 pipeline based on ImPlaneIA

https://github.com/spacetelescope/jwst

## KPI

# 1. Data reduction

## AMI

STScI stage 1 & 2 pipelines

+

STScI stage 3 pipeline based on ImPlaneIA

https://github.com/spacetelescope/jwst

## KPI

STScI stage 1 & 2 pipelines

+

Custom stage 3 pipeline based on XARA

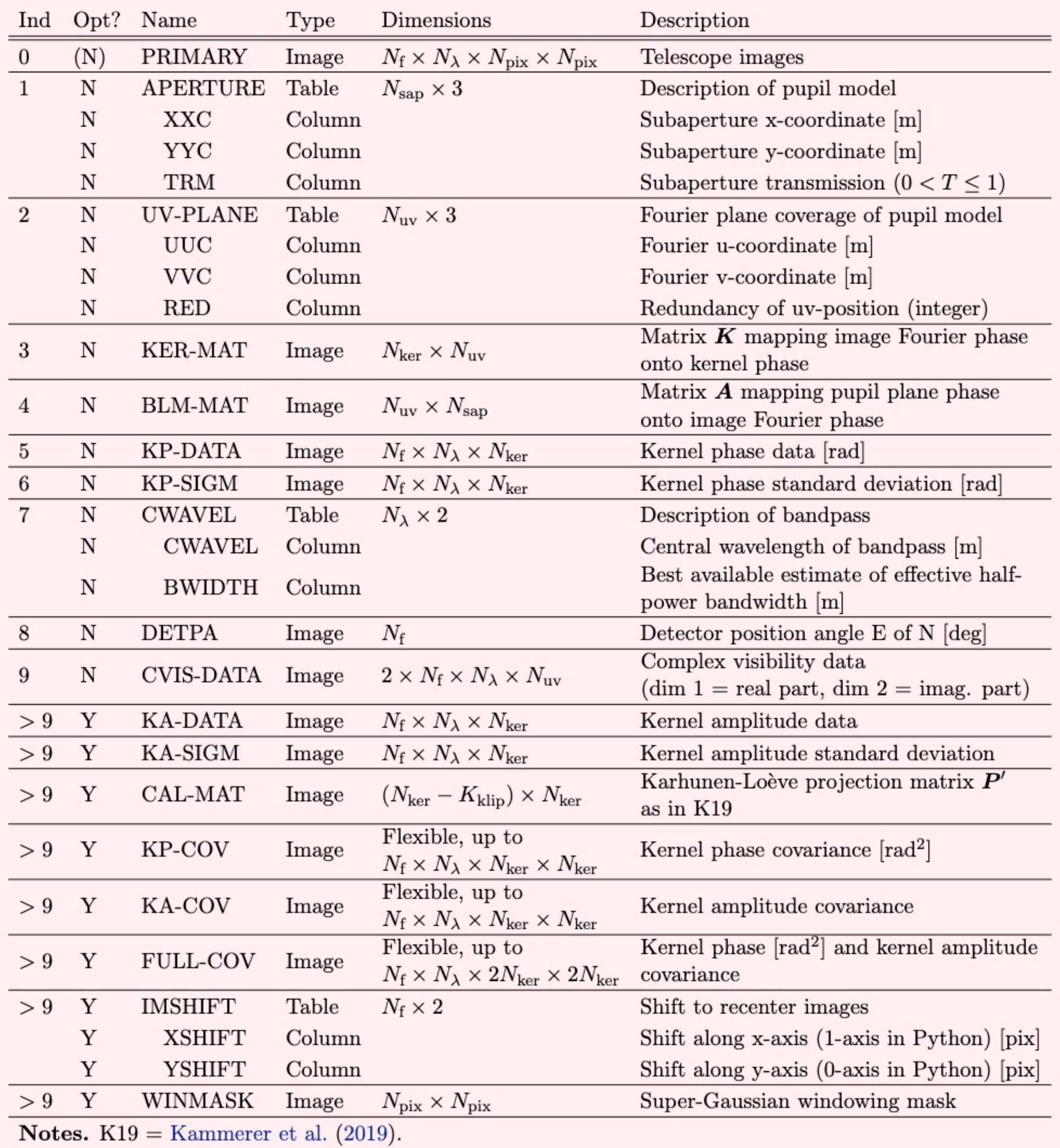
https://github.com/kammerje/xara/tree/develop

# KPI3Pipeline

```python
import os

# Set up CRDS environment variables (required for JWST data reduction pipeline).
os.environ['CRDS_PATH'] = 'crds_cache'
os.environ['CRDS_SERVER_URL'] = 'https://jwst-crds.stsci.edu'

# Import kernel phase pipeline from XARA.
from xara.calwebb_kpi3 import KPI3Pipeline

# Get calints files.
idirs = ['kerphase_realdata/NIRISS/comm_run2_s2/']
files = []
for idir in idirs:
    files += sorted([idir+f for f in os.listdir(idir) if f.endswith('_calints.fits')])
```

```python
# Run each file through pipeline.
for file in files:

    # Stage 3.
    result3 = KPI3Pipeline()
    result3.output_dir = 'kerphase_realdata/NIRISS/comm_run1_s3/' # output directory; if 'None', uses same directory as
    # Bad pixel fixing step.
    result3.fix_bad_pixels.skip = False # skip step?
    result3.fix_bad_pixels.plot = True # make and save plots?
    result3.fix_bad_pixels.bad_bits = ['DO_NOT_USE'] # DQ flags to be considered as bad pixels (see https://jwst-reffi
    result3.fix_bad_pixels.method = 'medfilt' # method to fix bad pixels; 'medfilt' or 'KI'
    # Re-centering step.
    result3.recenter_frames.skip = False # skip step?
    result3.recenter_frames.plot = True # make and save plots?
    result3.recenter_frames.method = 'FPNM' # XARA re-centering method; 'BCEN', 'COGI', or 'FPNM'
    result3.recenter_frames.crop = True # crop NIRISS reference pixels
    result3.recenter_frames.bmax = 4. # m; maximum baseline length for FPNM re-centering method
    result3.recenter_frames.pupil_path = '/Users/jkammerer/Documents/Code/xara/jwst/niriss_clear_pupil.fits' # path to
    # Windowing step.
    result3.window_frames.skip = False # skip step?
    result3.window_frames.plot = True # make and save plots?
    result3.window_frames.wrad = 15 # pix; radius of super-Gaussian window mask; if 'None', makes automatic guess
    # Kernel-phase extraction step.
    result3.extract_kerphase.skip = False # skip step?
    result3.extract_kerphase.plot = True # make and save plots?
    result3.extract_kerphase.bmax = None # m; maximum baseline length for kernel-phase extraction; if 'None', uses enti
    result3.extract_kerphase.pupil_path = '/Users/jkammerer/Documents/Code/xara/jwst/niriss_clear_pupil.fits' # path to
    # Empirical uncertainties step.
    result3.empirical_uncertainties.skip = False # skip step?
    result3.empirical_uncertainties.plot = True # make and save plots?
    result3.empirical_uncertainties.get_emp_err = True # estimate uncertainties empirically from standard deviation ove
    result3.empirical_uncertainties.get_emp_cor = False # estimate correlations empirically from standard deviation ove
    # Run pipeline.
    result3.run(file.replace('uncal', 'calints'))
```

- User interface similar to STScI pipelines
  → create familiarity
- No parameter tweaking necessary
  → plug & play
- 5 step process based on XARA
- Currently working with NIRISS & NIRCam (MIRI can be added)
  → versatility
- Need to provide pupil models for every instrument



| Ind | Opt? | Name | Type | Dimensions | Description |
|---|---|---|---|---|---|
| 0 | (N) | PRIMARY | Image | $N_\mathrm{f} \times N_\lambda \times N_\mathrm{pix} \times N_\mathrm{pix}$ | Telescope images |
| 1 | N | APERTURE | Table | $N_\mathrm{sap} \times 3$ | Description of pupil model |
|  | N | XXC | Column |  | Subaperture x-coordinate [m] |
|  | N | YYC | Column |  | Subaperture y-coordinate [m] |
|  | N | TRM | Column |  | Subaperture transmission ($0 < T \leq 1$) |
| 2 | N | UV-PLANE | Table | $N_\mathrm{uv} \times 3$ | Fourier plane coverage of pupil model |
|  | N | UUC | Column |  | Fourier u-coordinate [m] |
|  | N | VVC | Column |  | Fourier v-coordinate [m] |
|  | N | RED | Column |  | Redundancy of uv-position (integer) |
| 3 | N | KER-MAT | Image | $N_\mathrm{ker} \times N_\mathrm{uv}$ | Matrix $\boldsymbol{K}$ mapping image Fourier phase onto kernel phase |
| 4 | N | BLM-MAT | Image | $N_\mathrm{uv} \times N_\mathrm{sap}$ | Matrix $\boldsymbol{A}$ mapping pupil plane phase onto image Fourier phase |
| 5 | N | KP-DATA | Image | $N_\mathrm{f} \times N_\lambda \times N_\mathrm{ker}$ | Kernel phase data [rad] |
| 6 | N | KP-SIGM | Image | $N_\mathrm{f} \times N_\lambda \times N_\mathrm{ker}$ | Kernel phase standard deviation [rad] |
| 7 | N | CWAVEL | Table | $N_\lambda \times 2$ | Description of bandpass |
|  | N | CWAVEL | Column |  | Central wavelength of bandpass [m] |
|  | N | BWIDTH | Column |  | Best available estimate of effective half-power bandwidth [m] |
| 8 | N | DETPA | Image | $N_\mathrm{f}$ | Detector position angle E of N [deg] |
| 9 | N | CVIS-DATA | Image | $2 \times N_\mathrm{f} \times N_\lambda \times N_\mathrm{uv}$ | Complex visibility data (dim 1 = real part, dim 2 = imag. part) |
| $> 9$ | Y | KA-DATA | Image | $N_\mathrm{f} \times N_\lambda \times N_\mathrm{ker}$ | Kernel amplitude data |
| $> 9$ | Y | KA-SIGM | Image | $N_\mathrm{f} \times N_\lambda \times N_\mathrm{ker}$ | Kernel amplitude standard deviation |
| $> 9$ | Y | CAL-MAT | Image | $(N_\mathrm{ker} - K_\mathrm{klip}) \times N_\mathrm{ker}$ | Karhunen-Loève projection matrix $\boldsymbol{P}'$ as in K19 |
| $> 9$ | Y | KP-COV | Image | Flexible, up to $N_\mathrm{f} \times N_\lambda \times N_\mathrm{ker} \times N_\mathrm{ker}$ | Kernel phase covariance [rad$^2$] |
| $> 9$ | Y | KA-COV | Image | Flexible, up to $N_\mathrm{f} \times N_\lambda \times N_\mathrm{ker} \times N_\mathrm{ker}$ | Kernel amplitude covariance |
| $> 9$ | Y | FULL-COV | Image | Flexible, up to $N_\mathrm{f} \times N_\lambda \times 2N_\mathrm{ker} \times 2N_\mathrm{ker}$ | Kernel phase [rad$^2$] and kernel amplitude covariance |
| $> 9$ | Y | IMSHIFT | Table | $N_\mathrm{f} \times 2$ | Shift to recenter images |
|  | Y | XSHIFT | Column |  | Shift along x-axis (1-axis in Python) [pix] |
|  | Y | YSHIFT | Column |  | Shift along y-axis (0-axis in Python) [pix] |
| $> 9$ | Y | WINMASK | Image | $N_\mathrm{pix} \times N_\mathrm{pix}$ | Super-Gaussian windowing mask |

**Notes.** K19 = Kammerer et al. (2019).

# KPFITS

- Based on FITS file format from XARA
- Supports gray apertures
- Easy access to kernel phase projection matrix for model fitting/image re-construction purposes
- Supports IFS data
- Supports correlated errors

# 2. Model fitting

**AMI**

**KPI**

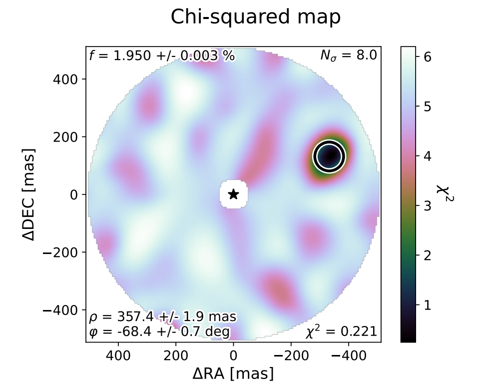
# 2. Model fitting

**AMI**

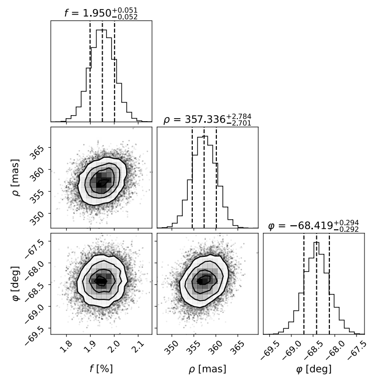
**KPI**

fouriever (understands OIFITS & KPFITS)

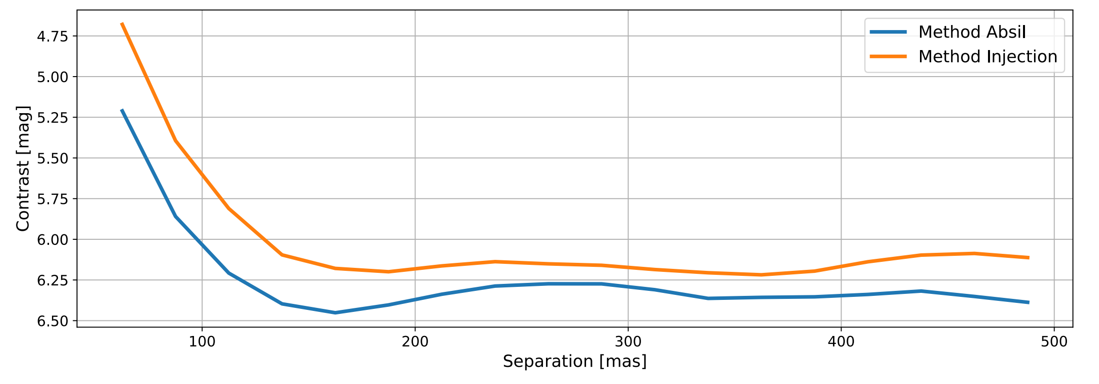
https://github.com/kammerje/fouriever



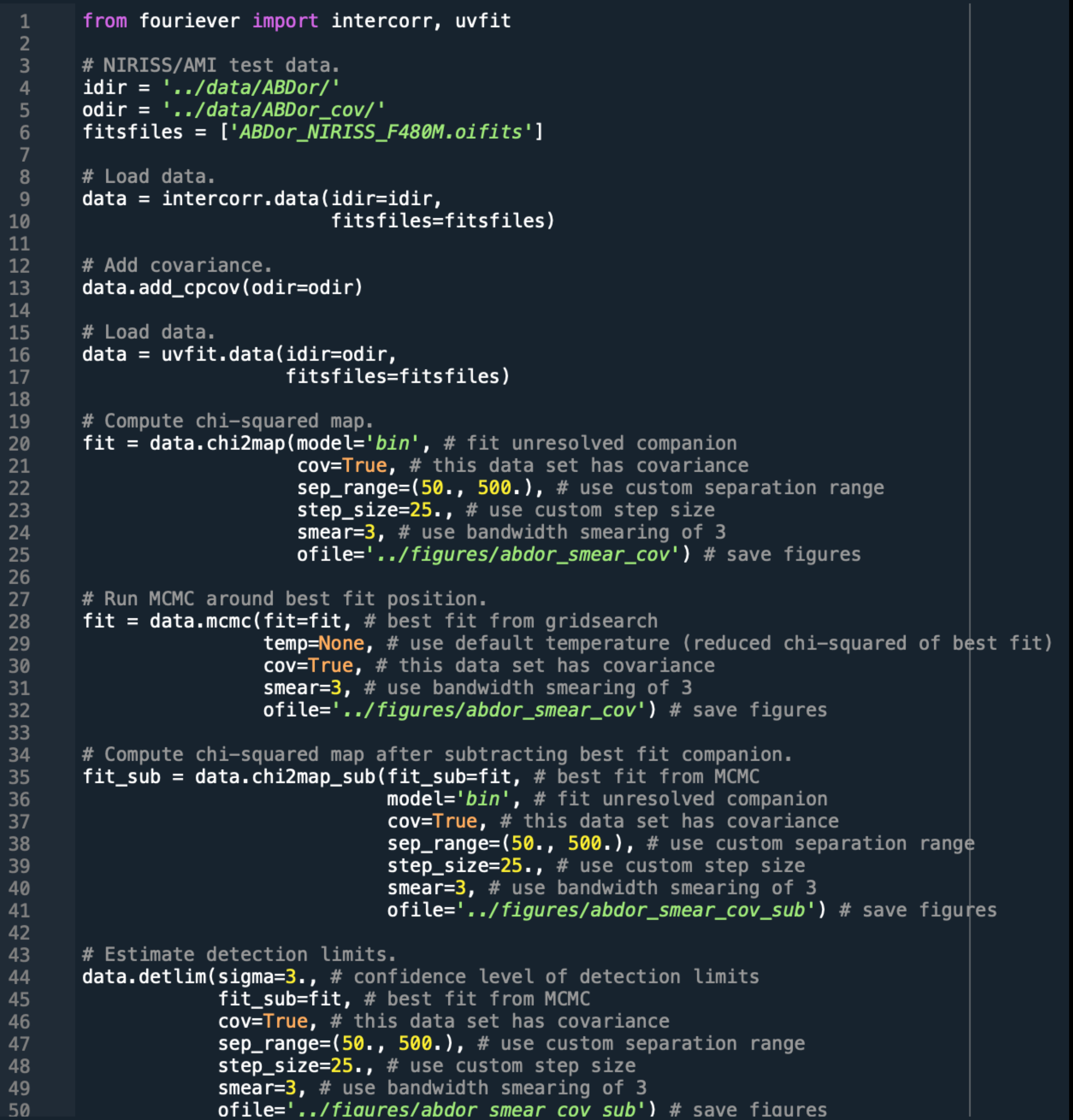
Detection map



MCMC fit



Contrast curve



```python
from fouriever import intercorr, uvfit

# NIRISS/AMI test data.
idir = '../data/ABDor/'
odir = '../data/ABDor_cov/'
fitsfiles = ['ABDor_NIRISS_F480M.oifits']

# Load data.
data = intercorr.data(idir=idir,
                      fitsfiles=fitsfiles)

# Add covariance.
data.add_cpcov(odir=odir)

# Load data.
data = uvfit.data(idir=odir,
                  fitsfiles=fitsfiles)

# Compute chi-squared map.
fit = data.chi2map(model='bin', # fit unresolved companion
                   cov=True, # this data set has covariance
                   sep_range=(50., 500.), # use custom separation range
                   step_size=25., # use custom step size
                   smear=3, # use bandwidth smearing of 3
                   ofile='../figures/abdor_smear_cov') # save figures

# Run MCMC around best fit position.
fit = data.mcmc(fit=fit, # best fit from gridsearch
                temp=None, # use default temperature (reduced chi-squared of best fit)
                cov=True, # this data set has covariance
                smear=3, # use bandwidth smearing of 3
                ofile='../figures/abdor_smear_cov') # save figures

# Compute chi-squared map after subtracting best fit companion.
fit_sub = data.chi2map_sub(fit_sub=fit, # best fit from MCMC
                           model='bin', # fit unresolved companion
                           cov=True, # this data set has covariance
                           sep_range=(50., 500.), # use custom separation range
                           step_size=25., # use custom step size
                           smear=3, # use bandwidth smearing of 3
                           ofile='../figures/abdor_smear_cov_sub') # save figures

# Estimate detection limits.
data.detlim(sigma=3., # confidence level of detection limits
            fit_sub=fit, # best fit from MCMC
            cov=True, # this data set has covariance
            sep_range=(50., 500.), # use custom separation range
            step_size=25., # use custom step size
            smear=3, # use bandwidth smearing of 3
            ofile='../figures/abdor_smear_cov_sub') # save figures
```

# fouriever

- Error correlation model (Kammerer+ 2020)
- Karhunen-Loève calibration (Kammerer+ 2019)
- Bandwidth smearing (Gallenne+ 2015)
- MCMC companion parameters (Wallace+ 2020)
- Detection limits (Gallenne+ 2015)

# 3. Image reconstruction

**AMI**

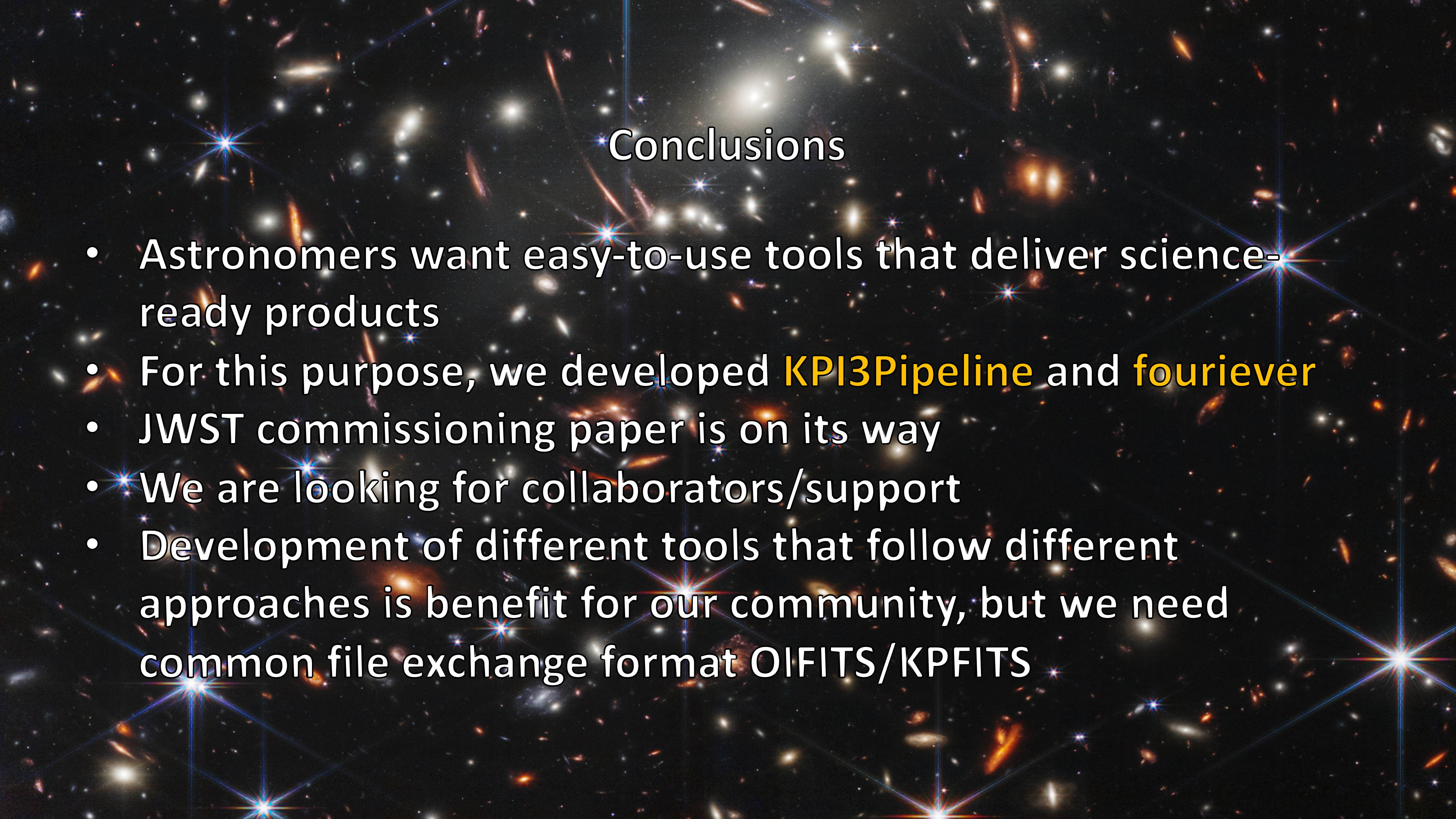
**KPI**

# 3. Image reconstruction

**AMI**

**KPI**

Your input is needed!



# Conclusions

- Astronomers want easy-to-use tools that deliver science-ready products
- For this purpose, we developed **KPI3Pipeline** and **fouriever**
- JWST commissioning paper is on its way
- We are looking for collaborators/support
- Development of different tools that follow different approaches is benefit for our community, but we need common file exchange format OIFITS/KPFITS