LVV-T2229

March 8, 2022

1 Closed Loop ComCam Image Ingestion and Application of Correction

This notebook is used to execute the [LVV-2229 (2.0)] test script during System Spread Integration Tests on Level 3.
It is part of the plan LVV-P81 and of the test cylce LVV-C176.
Execution steps are separated by horizontal lines.
Upon completion, save the notebook and its output as a pdf file to be attached to the test execution in JIRA.

LVV-T2229 (2.0) simply repeats the LVV-T2228 (1.0) test case twice, but with different targets.
This simulates two visits and tell us how the MTAOS behaves on sky.
The idea is that, depending on the angular distance between the two targets,
the MTAOS should use or not the corrections applied from the previous target.

[1]:
```
from lsst.ts import utils

# Extract your name from the Jupyter Hub
__executed_by__ = os.environ["JUPYTERHUB_USER"]

# Extract execution date
__executed_on__ = utils.astropy_time_from_tai_unix(utils.current_tai())
__executed_on__.format = "isot"

# This is used later to define where Butler stores the images
summit = os.environ["LSST_DDS_PARTITION_PREFIX"] == "summit"

print(f"\nExecuted by {__executed_by__} on {__executed_on__}."
      f"\n  At the summit? {summit}")
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Executed by b1quint on 2022-03-08T15:37:16.694.
  At the summit? True
```

1.1 Initial Setup

log onto the summit nublado
https://summit-lsp.lsst.codes/
git clone the ts_notebook repo

There will be a series of procedures to set up, "slew" and track the telescope before we get an image.
This is similar to test case LVV-T2189.

1.2 Check ComCam Playback Mode

Verify that ComCam can be use the playback option and that the required images are stored in the right place **TBD**.

1.3 Load all the needed libraries

Using the setup procedure, get the remotes and the components ready.

This includes simulators as well as real hardware when available (this will depend on when the test is conducted at NCSA or on level 3 or on the telescope):

- pointing
- mount (with the CCW)
- rotator
- ready M1M3: raise mirror, turn on FB, clear forces. Note that if used at level 3, we need to have M1M3 LUT use mount telemetry
- ready M2: turn on FB, clear forces. Note that if used at level 3, we need to have M2 LUT use mount telemetry
- Get cam hex Ready: check config; make sure LUT is on and has valid inputs; make sure hex is at LUT position
- Get M2 hex (simulator) Ready: check config; make sure LUT is on and has valid inputs; make sure hex is at LUT position
- Finally, get the MTAOS CSC ready

[2]:
```
%load_ext autoreload
%autoreload 2
```

[3]:
```
import rubin_jupyter_utils.lab.notebook as nb
nb.utils.get_node()
```

```
/tmp/ipykernel_41042/1665379685.py:2: DeprecationWarning: Call to deprecated
function (or staticmethod) get_node. (Please use lsst.rsp.get_node())
  nb.utils.get_node()
```

[3]:
```
'yagan06'
```

[4]:
```
import os
import sys
import asyncio
import logging

import pandas as pd
import numpy as np

from matplotlib import pyplot as plt

import lsst.daf.butler as dafButler

from lsst.ts import salobj
from lsst.ts.observatory.control.maintel import MTCS, ComCam
from lsst.ts.observatory.control import RotType
```

[5]:
```
logging.basicConfig(format="%(name)s:%(message)s", level=logging.DEBUG)
```

[6]:
```
log = logging.getLogger("setup")
log.level = logging.DEBUG
```

[7]:
```
domain = salobj.Domain()
```

[8]:
```
mtcs = MTCS(domain=domain, log=log)
mtcs.set_rem_loglevel(40)
```

```
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
```

<IPython.core.display.HTML object>

[9]:
```
await mtcs.start_task
```

[9]: [None, None, None, None, None, None, None, None, None, None]

[10]:
```
comcam = ComCam(domain=domain, log=log)
comcam.set_rem_loglevel(40)
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

[11]:
```
await comcam.start_task
```

[11]: [None, None, None]

[12]:
```
await comcam.standby()
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

[13]:
```
await comcam.enable()
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

1.4 Slew and Track

Using the slew procedure, slew the systems to a specific elevation, azimuth and rotator angle. Verify that the telemetry is generated.

Slew to **RA 20:28:18.74** and **DEC -87:28:19.9** with **rot_type=RotType.Physical** and **Rotator Angle of 0º**. We use this field because it is the field that was simulated and that is a field that is visible the whole year.

RotType Physical Ensures that the Rotator will not move. This is necessary because the CCW is not running (MTmount in simulation mode).

Slew to target:

[14]:
```
await mtcs.slew_icrs(ra="20:28:18.74", dec="-87:28:19.9", rot_type=RotType.
↪Physical, rot=0)
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
```

```
[14]: (<ICRS Coordinate: (ra, dec) in deg
          (307.07808333, -87.47219444)>,
       <Angle 0. deg>)
```

1.5 Take in-focus image

Once the different components are ready (M1M3, M2, rotator and CCW, hexapods) and tracking, take an image using the take_image command in playback mode.
This second image should be the one that uses the correction calculated with the first slew.

[ ]:
```
exp_focus = await comcam.take_object(15)
print(f"Target exposure: {exp_focus}")
```

1.6 Intra Focus Position

Using the Camera Hexapod, piston ComCam +1mm

[15]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

[15]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f4cec8139d0>
```

1.7 Intra Focus Image

While tracking, take an image with ComCam and check that the header is containing the right telemetry

[16]:
```
exp_intra = await comcam.take_object(15)
print(f"Target 1 exposure: {exp_intra}")
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Target 1 exposure: [2022030800001]
```

1.8 Extra Focus Position

Using the Camera Hexapod, piston ComCam to -1mm

[17]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=-2000.)
```

[17]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f4c89ec2d30>
```

1.9 Extra Focus Image

While tracking, take an image with ComCam and check that the header is containing the right telemetry.

[18]:
```
exp_extra = await comcam.take_object(15)
print(f"Target 1 exposure: {exp_extra}")
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Target 1 exposure: [2022030800002]
```

1.10 Go Back to Nominal Focus Position

Put the hexapod back to 0mm.

[19]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

[19]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f4c23b01c40>
```

1.11 Stop Tracking

If using MTMount Simulator and CCW Following Mode Disabled, stop tracking to prevent the Rotator to hit the limit switches.

[20]:
```
await mtcs.stop_tracking()
```

```
<IPython.core.display.HTML object>
```

1.12 Get Zernike Coefficients

Use the MTAOS Wavefront Estimator Pipeline to calculate the required Zernike Coefficients that represent the Wavefront data.

[25]:
```
import yaml
```

[27]:
```
wep_config = yaml.safe_dump(
    dict(
        tasks=dict(
            isr=dict(
                config=dict(
                    doOverscan=False,
                    doApplyGains=False,
```

```
                )
            ),
            generateDonutCatalogWcsTask=dict(
                config={
                    "filterName": "phot_g_mean",
                    "connections.refCatalogs": "gaia_dr2_20200414",
                    "donutSelector.sourceLimit": 10,
                    "donutSelector.fluxField": "phot_g_mean_flux"
                }
            )
        )
    )
)
```

[44]:
```
await mtcs.set_state(salobj.State.ENABLED, components=["mtaos"],␣
 ↪settings=dict(mtaos="comcam"))
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

[45]:
```
await mtcs.rem.mtaos.cmd_setLogLevel.set_start(
    level=logging.DEBUG,
    timeout=5
)
```

[45]:
```
<ddsutil.MTAOS_ackcmd_fd03e870 at 0x7f4c23af8dc0>
```

[46]:
```
print(f"Intra focus exposure: {exp_intra[0]}\n"
      f"Extra focus exposure: {exp_extra[0]}\n")
```

```
Intra focus exposure: 2022030800001
Extra focus exposure: 2022030800002
```

[47]:
```
await mtcs.rem.mtaos.cmd_runWEP.set_start(visitId=exp_intra[0] - 2021111900000,
                                          extraId=exp_extra[0] - 2021111900000,
                                          config=wep_config)
```

[47]:
```
<ddsutil.MTAOS_ackcmd_fd03e870 at 0x7f4c89d8e670>
```

1.13 Get Corrections

Use the MTAOS Optical Feedback Controller to retrieve the corrections that should be applied to m1m3, m2, camera hexapod, and m2 hexapod.

[48]:
```
await mtcs.rem.mtaos.cmd_runOFC.start(timeout=60.)
```

[48]:
```
<ddsutil.MTAOS_ackcmd_fd03e870 at 0x7f4ceca4c520>
```

1.14 Issue the corrections

Issue the corrections found by the MTAOS OFC to m1m3, m2, camera hexapod, and m2 hexapod.

[49]:
```
await mtcs.rem.mtaos.cmd_issueCorrection.start(timeout=60.)
```

[49]:
```
<ddsutil.MTAOS_ackcmd_fd03e870 at 0x7f4cec53d6d0>
```

1.15 Verify ISR Data

Make sure that the Instrument Signature Removal ran on the intra- and extra-focus data and that this data is accessible via Butler.

[50]:
```
if summit:
    butler = dafButler.Butler("/repo/LSSTComCam/")
else:
    butler = dafButler.Butler("/repo/main/")
```

[51]:
```
registry = butler.registry

collections = [collection for collection in registry.queryCollections()
               if collection.startswith('mtaos_wep')]
```

[52]:
```
exp_intra_id = {'instrument': 'LSSTComCam',
                'detector': 0,
                'exposure': exp_intra[0]}

raw_intra = butler.get('postISRCCD', dataId=exp_intra_id,
                       collections=collections)

print(raw_intra.getMetadata())
```

```
<IPython.core.display.HTML object>
```

```
COMMENT = [ "= '        '", "---- Date, night and basic image information ----",
"= '        '", "---- Telescope info, location, observer ----", "= '        '",
"---- Pointing info, etc. ----", "= '        '", "---- Image-identifying used to
build OBS-ID ----", "= '        '", "---- Image sequence numbers", "= '
'", "---- Additional Keys Information from Camera ----", "= '        '", "----
Test Stand information ----", "= '        '", "---- Information from Camera
(Common block) ----", "= '        '", "---- Filter/grating information ----", "=
```

```
'        '", "---- Exposure-related information ----", "= '        '", "----
Header information ----", "= '        '", "---- Hierarch information for CSC
Simulatiom Mode ----", "---- Information from Camera per sensor ----", "= '
'", "---- Geometry from Camera ----", "= '        '", "---- Checksums ----" ]
ORIGIN = "LSST DM Header Service"
// FITS file originator
DATE = "2022-03-08T16:03:21.793"
// Creation Date and Time of File
DATE-BEG = "2022-03-08T16:03:38.114"
// Time at the start of integration
DATE-END = "2022-03-08T16:03:56.170"
// end date of the observation
MJD = 59646.669002237
// Modified Julian Date that the file was written
MJD-BEG = 59646.669191134
// Modified Julian Date derived from DATE-BEG
MJD-END = 59646.669400113
// Modified Julian Date derived from DATE-END
OBSID = "CC_O_20220308_000001"
// ImageName from Camera StartIntergration
GROUPID = "2022-03-08T16:03:37.572"
IMGTYPE = "OBJECT"
// BIAS, DARK, FLAT, OBJECT
BUNIT = "adu"
// Brightness units for pixel array
FACILITY = "Vera C. Rubin Observatory"
// Facility name
TELESCOP = "Simonyi Survey Telescope"
// Telescope name
INSTRUME = "ComCam"
// Instrument used to obtain these data
OBSERVER = "LSST"
// Observer name(s)
OBSGEO-X = 1818938.9400000
// [m] X-axis Geocentric coordinate
OBSGEO-Y = -5208470.9500000
// [m] Y-axis Geocentric coordinate
OBSGEO-Z = -3195172.0800000
// [m] Z-axis Geocentric coordinate
RA = 307.07808333333
// RA commanded from pointing component
DEC = -87.472194444444
// DEC commanded from pointing component
RASTART = 307.07808856575
// RA of telescope from AZSTART and ELSTART
DECSTART = -87.472194452775
// DEC of telescope from AZSTART and ELSTART
RAEND = 307.07809751277
```

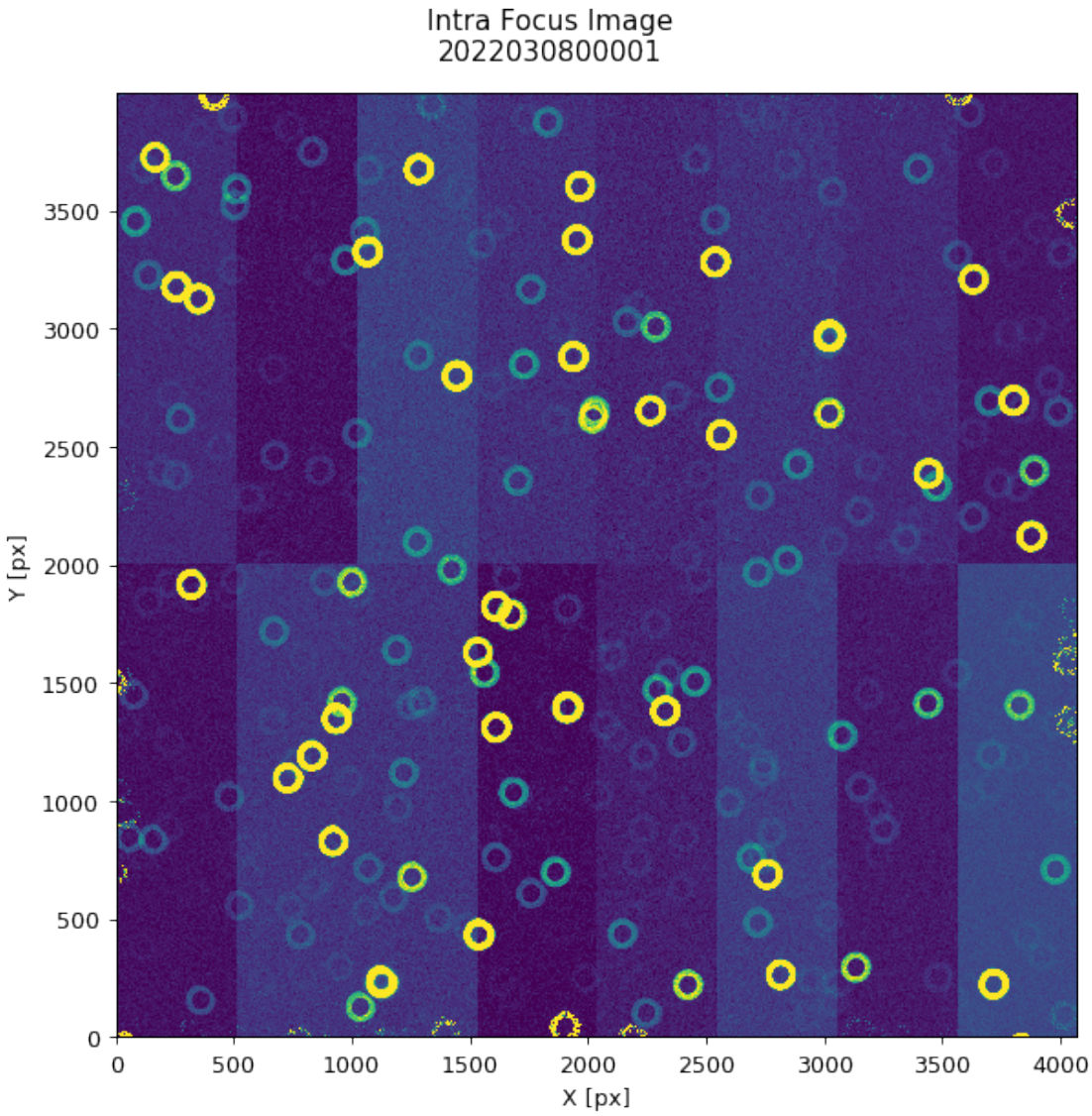
```
// RA of telescope from AZEND and ELEND
DECEND = -87.472194467023
// DEC of telescope from AZEND and ELEND
ROTPA = 8.4879831658374e-314
// Rotation angle relative to the sky (deg)
ROTCOORD = "sky"
// Telescope Rotation Coordinates
HASTART = 1.7945748113027
// [HH:MM:SS] Telescope hour angle at start
ELSTART = 32.579073337134
// [deg] Telescope zenith distance at start
AZSTART = 181.39892105080
// [deg] Telescope azimuth angle at start
AMSTART = 1.8529594844458
// Airmass at start
HAEND = 1.8003060161957
// [HH:MM:SS] Telescope hour angle at end
ELEND = 32.577247268406
// [deg] Telescope zenith distance at end
AZEND = 181.40305225664
// [deg] Telescope azimuth angle at end
AMEND = 1.8530507480393
// Airmass at end
TRACKSYS = "RADEC"
// Tracking system RADEC, AZEL, PLANET, EPHEM
RADESYS = "ICRS"
// Equatorial coordinate system FK5 or ICRS
FOCUSZ = 1000.0000000000
// Focus Z position
OBJECT = "slew_icrs"
// Name of the observed object
CAMCODE = "CC"
// The code for the camera (AT/CC/MC)
CONTRLLR = "O"
// The controller (e.g. O for OCS, C for (A)CCS)
DAYOBS = "20220308"
// The observation day as defined by image name
SEQNUM = 1
// The sequence number from the image name
TESTTYPE = "OBJECT"
// Test Type: BIAS:DARK:FLAT:OBS:PPUMP:QE:SFLAT
EMUIMAGE = "CC_H_20211231_006001"
// Image being emulated
CURINDEX = 0
// Index number for exposure within the sequence
MAXINDEX = 1
// Number of requested images in sequence
PROGRAM = <Unknown>
```

```
// Name of the program
REASON = <Unknown>
// Reason for observation
TSTAND = "EOCCv2_SUM"
// The Test Stand used
IMAGETAG = "716b11ddfc2ea180"
// DAQ Image id
OBSANNOT = ""
// Observing annotation
FILTBAND = "g"
// Name of the filter band
FILTER = "g_07"
// Name of the physical filter
FILTSLOT = 3
// Filter home slot
FILTPOS = 4423.0000000000
// Filter measured position of slide
SHUTTIME = 15.000000000000
// Shutter exposure time in seconds
FILENAME = "CC_O_20220308_000001.fits"
// Original file name
HEADVER = "2.9.5"
// Version of header
SIMULATE MTMOUNT = 1
// MTMount Simulation Mode (False=0)
SIMULATE MTM1M3 = <Unknown>
// MTM1M3 Simulation Mode (False=0)
SIMULATE MTM2 = 1
// MTM2 Simulation Mode (False=0)
SIMULATE CAMHEXAPOD = 0
// CAMHexapod Simulation Mode (False=0)
SIMULATE M2HEXAPOD = 1
// M2Hexapod Simulation Mode (False=0)
SIMULATE MTROTATOR = 0
// MTRotator Simulation Mode (False=0)
SIMULATE MTDOME = 1
// MTDome Simulation Mode (False=0)
SIMULATE MTDOMETRAJECTORY = 0
// MTDomeTrajectory Simulation Mode (False
CCD_MANU = "ITL"
// CCD Manufacturer
CCD_TYPE = 2
// CCD Model Number
CCD_SERN = "23166"
// Manufacturers CCD Serial Number
LSST_NUM = "ITL-3800C-229"
// LSST Assigned CCD Number
SEQCKSUM = "2552520002"
```

```
// Checksum of Sequencer
SEQNAME = "FP_ITL_2s_ir2_v25.seq"
// SequenceName from Camera
REBNAME = "LCA-13574-061"
// Name of the REB
CONTNUM = "-1"
// CCD Controller (WREB) Serial Number
TEMP_SET = -100.00000000000
// Temperature set point (deg C)
CCDTEMP = <Unknown>
// Measured temperature (deg C)
CCDSLOT = "S00"
// Name of the CCD Slot (SXx)
RAFTBAY = "R22"
// Name of the RAFT Bay (Rnn)
OVERH = 64
// Over-scan pixels
OVERV = 48
// Vert-overscan pix
PREH = 0
// Pre-scan pixels
INHERIT = 1
// Inherits global header
EXTNAME = "Segment10"
DETSEC = "[509:1,1:2000]"
DETSIZE = "[1:4072,1:4000]"
DTV1 = 510
// detector transformation vector
DTV2 = 0
// detector transformation vector
DTM1_1 = -1.0000000000000
// detector transformation matrix
DTM2_2 = 1.0000000000000
// detector transformation matrix
DTM1_2 = 0
// detector transformation matrix
DTM2_1 = 0
// detector transformation matrix
ASTRO METADATA FIX MODIFIED = 0
ASTRO METADATA FIX DATE = "2022-03-08T16:44:13.217987"


<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

[53]:
```
%matplotlib inline
fig, ax = plt.subplots(num="Intra Focus Image", figsize=(7,7), dpi=90)
```

```
vmin = np.percentile(raw_intra.image.array, 2)
vmax = np.percentile(raw_intra.image.array, 98)

ax.imshow(raw_intra.image.array,
          origin='lower',
          interpolation='nearest',
          vmin=vmin,
          vmax=vmax)
ax.set_xlabel("X [px]")
ax.set_ylabel("Y [px]")

fig.suptitle(f"Intra Focus Image\n{exp_intra_id['exposure']}")
fig.tight_layout()

plt.show()
```



[54]:
```
exp_extra_id = {'instrument': 'LSSTComCam',
                'detector': 0,
                'exposure': exp_extra[0]}

exp_extra = butler.get('postISRCCD', dataId=exp_extra_id,
                       collections=collections)

print(exp_extra.getMetadata())
```

```
<IPython.core.display.HTML object>
```

```
COMMENT = [ "= '         '", "---- Date, night and basic image information ----",
"= '         '", "---- Telescope info, location, observer ----", "= '         '",
```

"---- Pointing info, etc. ----", "= '   '", "---- Image-identifying used to build OBS-ID ----", "= '   '", "---- Image sequence numbers", "= '   '", "---- Additional Keys Information from Camera ----", "= '   '", "---- Test Stand information ----", "= '   '", "---- Information from Camera (Common block) ----", "= '   '", "---- Filter/grating information ----", "= '   '", "---- Exposure-related information ----", "= '   '", "---- Header information ----", "= '   '", "---- Hierarch information for CSC Simulatiom Mode ----", "---- Information from Camera per sensor ----", "= '   '", "---- Geometry from Camera ----", "= '   '", "---- Checksums ----"]
ORIGIN = "LSST DM Header Service"
// FITS file originator
DATE = "2022-03-08T16:04:39.403"
// Creation Date and Time of File
DATE-BEG = "2022-03-08T16:04:56.042"
// Time at the start of integration
DATE-END = "2022-03-08T16:05:14.094"
// end date of the observation
MJD = 59646.669900501
// Modified Julian Date that the file was written
MJD-BEG = 59646.670093084
// Modified Julian Date derived from DATE-BEG
MJD-END = 59646.670302019
// Modified Julian Date derived from DATE-END
OBSID = "CC_O_20220308_000002"
// ImageName from Camera StartIntergration
GROUPID = "2022-03-08T16:04:55.874"
IMGTYPE = "OBJECT"
// BIAS, DARK, FLAT, OBJECT
BUNIT = "adu"
// Brightness units for pixel array
FACILITY = "Vera C. Rubin Observatory"
// Facility name
TELESCOP = "Simonyi Survey Telescope"
// Telescope name
INSTRUME = "ComCam"
// Instrument used to obtain these data
OBSERVER = "LSST"
// Observer name(s)
OBSGEO-X = 1818938.9400000
// [m] X-axis Geocentric coordinate
OBSGEO-Y = -5208470.9500000
// [m] Y-axis Geocentric coordinate
OBSGEO-Z = -3195172.0800000
// [m] Z-axis Geocentric coordinate
RA = 307.07808333333
// RA commanded from pointing component
DEC = -87.472194444444
// DEC commanded from pointing component

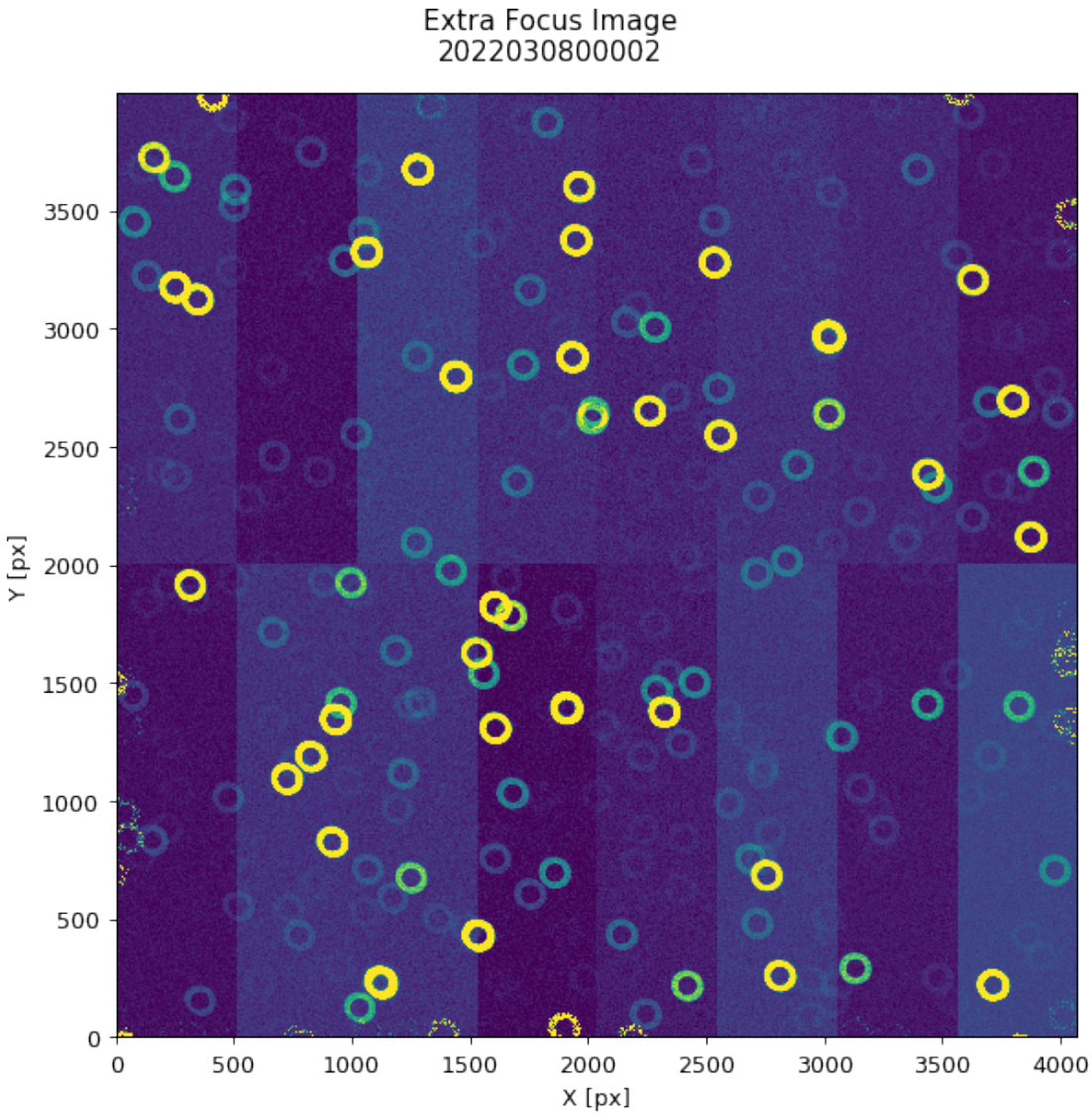
```
RASTART = 307.07813575193
// RA of telescope from AZSTART and ELSTART
DECSTART = -87.472194527913
// DEC of telescope from AZSTART and ELSTART
RAEND = 307.07809680630
// RA of telescope from AZEND and ELEND
DECEND = -87.472194465896
// DEC of telescope from AZEND and ELEND
ROTPA = 8.4879831658374e-314
// Rotation angle relative to the sky (deg)
ROTCOORD = "sky"
// Telescope Rotation Coordinates
HASTART = 1.8163002738333
// [HH:MM:SS] Telescope hour angle at start
ELSTART = 32.572168653481
// [deg] Telescope zenith distance at start
AZSTART = 181.41446054246
// [deg] Telescope azimuth angle at start
AMSTART = 1.8533069105704
// Airmass at start
HAEND = 1.8219339200019
// [HH:MM:SS] Telescope hour angle at end
ELEND = 32.570359257530
// [deg] Telescope zenith distance at end
AZEND = 181.41849654770
// [deg] Telescope azimuth angle at end
AMEND = 1.8533976538736
// Airmass at end
TRACKSYS = "RADEC"
// Tracking system RADEC, AZEL, PLANET, EPHEM
RADESYS = "ICRS"
// Equatorial coordinate system FK5 or ICRS
FOCUSZ = -1000.0000000000
// Focus Z position
OBJECT = "slew_icrs"
// Name of the observed object
CAMCODE = "CC"
// The code for the camera (AT/CC/MC)
CONTRLLR = "O"
// The controller (e.g. O for OCS, C for (A)CCS)
DAYOBS = "20220308"
// The observation day as defined by image name
SEQNUM = 2
// The sequence number from the image name
TESTTYPE = "OBJECT"
// Test Type: BIAS:DARK:FLAT:OBS:PPUMP:QE:SFLAT
EMUIMAGE = "CC_H_20211231_006002"
// Image being emulated
```

```
CURINDEX = 0
// Index number for exposure within the sequence
MAXINDEX = 1
// Number of requested images in sequence
PROGRAM = <Unknown>
// Name of the program
REASON = <Unknown>
// Reason for observation
TSTAND = "EOCCv2_SUM"
// The Test Stand used
IMAGETAG = "6af46f8cfe7e3625"
// DAQ Image id
OBSANNOT = ""
// Observing annotation
FILTBAND = "g"
// Name of the filter band
FILTER = "g_07"
// Name of the physical filter
FILTSLOT = 3
// Filter home slot
FILTPOS = 4423.0000000000
// Filter measured position of slide
SHUTTIME = 15.000000000000
// Shutter exposure time in seconds
FILENAME = "CC_O_20220308_000002.fits"
// Original file name
HEADVER = "2.9.5"
// Version of header
SIMULATE MTMOUNT = 1
// MTMount Simulation Mode (False=0)
SIMULATE MTM1M3 = <Unknown>
// MTM1M3 Simulation Mode (False=0)
SIMULATE MTM2 = 1
// MTM2 Simulation Mode (False=0)
SIMULATE CAMHEXAPOD = 0
// CAMHexapod Simulation Mode (False=0)
SIMULATE M2HEXAPOD = 1
// M2Hexapod Simulation Mode (False=0)
SIMULATE MTROTATOR = 0
// MTRotator Simulation Mode (False=0)
SIMULATE MTDOME = 1
// MTDome Simulation Mode (False=0)
SIMULATE MTDOMETRAJECTORY = 0
// MTDomeTrajectory Simulation Mode (False
CCD_MANU = "ITL"
// CCD Manufacturer
CCD_TYPE = 2
// CCD Model Number
```

```
CCD_SERN = "23166"
// Manufacturers CCD Serial Number
LSST_NUM = "ITL-3800C-229"
// LSST Assigned CCD Number
SEQCKSUM = "2552520002"
// Checksum of Sequencer
SEQNAME = "FP_ITL_2s_ir2_v25.seq"
// SequenceName from Camera
REBNAME = "LCA-13574-061"
// Name of the REB
CONTNUM = "-1"
// CCD Controller (WREB) Serial Number
TEMP_SET = -100.00000000000
// Temperature set point (deg C)
CCDTEMP = 519.48052978516
// Measured temperature (deg C)
CCDSLOT = "S00"
// Name of the CCD Slot (SXx)
RAFTBAY = "R22"
// Name of the RAFT Bay (Rnn)
OVERH = 64
// Over-scan pixels
OVERV = 48
// Vert-overscan pix
PREH = 0
// Pre-scan pixels
INHERIT = 1
// Inherits global header
EXTNAME = "Segment10"
DETSEC = "[509:1,1:2000]"
DETSIZE = "[1:4072,1:4000]"
DTV1 = 510
// detector transformation vector
DTV2 = 0
// detector transformation vector
DTM1_1 = -1.0000000000000
// detector transformation matrix
DTM2_2 = 1.0000000000000
// detector transformation matrix
DTM1_2 = 0
// detector transformation matrix
DTM2_1 = 0
// detector transformation matrix
ASTRO METADATA FIX MODIFIED = 0
ASTRO METADATA FIX DATE = "2022-03-08T16:44:02.835918"


<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

[55]:
```
%matplotlib inline
fig, ax = plt.subplots(num="Extra Focus Image", figsize=(7, 7), dpi=90)

vmin = np.percentile(exp_extra.image.array, 2)
vmax = np.percentile(exp_extra.image.array, 98)

ax.imshow(exp_extra.image.array,
          origin='lower',
          interpolation='nearest',
          vmin=vmin,
          vmax=vmax)

ax.set_xlabel("X [px]")
ax.set_ylabel("Y [px]")

fig.suptitle(f"Extra Focus Image\n{exp_extra_id['exposure']}")
fig.tight_layout()

plt.show()
```



1.16 Slew and Track Second Target

Now, slew to a second target. The coordinates for this targets are **TBD** and depend on new simulated data. You will probably not run this for now until we have new simulated data. We will leave the notebook simply to have the structure pre-define.

Slew to **RA TBD** and **DEC TBD** with **rot_type=RotType.Physical** and **Rotator Angle of 0º**.

RotType Physical Ensures that the Rotator will not move. This is necessary because the CCW is not running (MTmount in simulation mode).

[ ]:
```
await mtcs.slew_icrs(ra=???, dec=???, rot_type=RotType.Physical, rot=0)
```

1.17 Take in-focus image 2

Once the different components are ready (M1M3, M2, rotator and CCW, hexapods) and tracking, take an image using the take_image command in playback mode. This second image should be the one that uses the correction calculated with the first slew.

[ ]:
```
exp_focus2 = await comcam.take_object(15)
print(f"Target exposure: {exp_focus2}")
```

1.18 Intra Focus Position 2

Using the Camera Hexapod, piston ComCam +1mm.

[ ]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

1.19 Intre Focus Image 2

While tracking, take an image and check that the header is containing the right telemetry.

[ ]:
```
exp_intra2 = await comcam.take_object(15)
print(f"Target 1 exposure: {exp_intra2}")
```

1.20 Extra Focus Position 2

Apply an offset of -2000 um to the Camera Hexapod, to bring it down to -1 mm.

[ ]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=-2000.)
```

1.21 Extra Focus Image 2

While tracking, take an image and check that the header is containing the right telemetry

[ ]:
```
exp_extra2 = await comcam.take_object(15)
print(f"Target 1 exposure: {exp_extra2}")
```

1.22 Go back to focus position 2

Send the hexapod back to 0 mm by applying an offset of 1000 um in Z.

```
[ ]: await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

1.23 Stop tracking 2

If using MTMount Simulator and CCW Following Mode Disabled, stop tracking to prevent the Rotator to hit the limit switches.

```
[ ]: await mtcs.stop_tracking()
```

1.24 Get Zernikes Coefficients 2

Use the MTAOS to calculate the required offsets to be sent to M1M3, M2, and the hexapods.

When we run the command in the example code below, if it does not raise the **TBD** error, then we know that the MTAOS WEP could find and retrieve the calibration files.

```
[ ]: await mtcs.rem.mtaos.cmd_runWEP.set_start(visitId=exp_intra2[0],
                                               extraId=exp_extra2[0])
```

1.25 Get Corrections 2

Apply the resulting offsets to the M1M3, M2 and the hexapods

```
[ ]: await mtcs.rem.mtaos.cmd_runOFC.start(timeout=60.)
```

1.25.1 Issue the corrections 2

Issue (apply) the corrections found by the MTAOS OFC to m1m3, m2, camera hexapod, and m2 hexapod.

```
[ ]: await mtcs.rem.mtaos.cmd_issueCorrection.start(timeout=60.)
```

Verify Offsets TBD

Verify that the offsets are the expected one by plotting: - m1m3 actuator 101 z force - m2 actuator B1 force - camera hex y position - m2 hex y position - What about others?

1.26 Wrap Up and Shut Down

This section is intended for shutting down the system and should not be run as part of the regular testing procedure. Only run the following cells if you are done with the system and don't plan on executing any further tests.

[ ]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtaos"])
```

[ ]:
```
await mtcs.lower_m1m3()
```

[ ]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtm1m3"])
```

[ ]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtm2"])
```

[ ]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mthexapod_1"])
```

[ ]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mthexapod_2"])
```

[ ]:
```
await mtcs.standby()
```

[ ]:
```
await comcam.standby()
```