# 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.

In [ ]:
```
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}")
```

---

## 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.

---

## 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**.

# 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

In [1]:
```
%load_ext autoreload
%autoreload 2
```

In [2]:
```
import rubin_jupyter_utils.lab.notebook as nb
nb.utils.get_node()
```

```
/tmp/ipykernel_26104/1665379685.py:2: DeprecationWarning: Call to deprecated
function (or staticmethod) get_node. (Please use lsst.rsp.get_node())
  nb.utils.get_node()
```

Out[2]:
```
'yagan04'
```

In [3]:
```
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
```

```
lsst.ts.utils.tai INFO: Update leap second table
lsst.ts.utils.tai INFO: current_tai uses the system TAI clock
```

In [4]:
```
logging.basicConfig(format="%(name)s:%(message)s", level=logging.DEBUG)
```

In [5]:
```
log = logging.getLogger("setup")
log.level = logging.DEBUG
```

In [6]:
```
domain = salobj.Domain()
```

In [7]:
```
mtcs = MTCS(domain=domain, log=log)
mtcs.set_rem_loglevel(40)
```

```
setup.MTCS DEBUG: mtmount: Adding all resources.
setup.MTCS DEBUG: mtptg: Adding all resources.
setup.MTCS DEBUG: mtaos: Adding all resources.
setup.MTCS DEBUG: mtm1m3: Adding all resources.
setup.MTCS DEBUG: mtm2: Adding all resources.
setup.MTCS DEBUG: mthexapod_1: Adding all resources.
setup.MTCS DEBUG: mthexapod_2: Adding all resources.
setup.MTCS DEBUG: mtrotator: Adding all resources.
setup.MTCS DEBUG: mtdome: Adding all resources.
setup.MTCS DEBUG: mtdometrajectory: Adding all resources.
MTHexapod INFO: Read historical data in 0.03 sec
MTHexapod INFO: Read historical data in 0.05 sec
```

In [8]:
```
await mtcs.start_task
```

Out[8]:
```
[None, None, None, None, None, None, None, None, None, None]
```

In [9]:
```
comcam = ComCam(domain=domain, log=log)
comcam.set_rem_loglevel(40)
```

```
setup.ComCam DEBUG: cccamera: Adding all resources.
setup.ComCam DEBUG: ccheaderservice: Adding all resources.
setup.ComCam DEBUG: ccarchiver: Adding all resources.
```

In [10]:
```
await comcam.start_task
```

Out[10]:
```
[None, None, None]
```

In [11]:
```
await comcam.enable()
```

```
setup.ComCam INFO: Enabling all components
setup.ComCam DEBUG: Gathering settings.
setup.ComCam DEBUG: Couldn't get settingVersions event. Using empty sett
ings.
setup.ComCam DEBUG: Couldn't get settingVersions event. Using empty sett
ings.
setup.ComCam DEBUG: Complete settings for cccamera.
setup.ComCam DEBUG: Complete settings for ccheaderservice.
setup.ComCam DEBUG: Complete settings for ccarchiver.
setup.ComCam DEBUG: Settings versions: {'cccamera': '', 'ccheaderservice
': '', 'ccarchiver': ''}
setup.ComCam DEBUG: [cccamera]::[<State.ENABLED: 2>]
setup.ComCam DEBUG: [ccheaderservice]::[<State.ENABLED: 2>]
setup.ComCam DEBUG: [ccarchiver]::[<State.ENABLED: 2>]
setup.ComCam INFO: All components in <State.ENABLED: 2>.
```

---

# 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 0°**. 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:

In [12]:
```
await mtcs.slew_icrs(ra="20:28:18.74", dec="-87:28:19.9", rot_type=RotType.P
```

```
setup.MTCS DEBUG: Setting rotator to physical fixed position 0.0 deg. Ro
tator will not track.
setup.MTCS WARNING: Camera cable wrap following disabled in MTMount.
setup.MTCS DEBUG: Wait 5.0s for rotator to settle down.
setup.MTCS DEBUG: Workaround for rotator trajectory problem. Moving rota
tor to its current position: 1.66
setup.MTCS DEBUG: Wait for MTRotator in position event.
setup.MTCS DEBUG: MTRotator in position: False.
setup.MTCS INFO: MTRotator in position: True.
setup.MTCS DEBUG: MTRotator in position True. Waiting settle time 5.0s
setup.MTCS DEBUG: Sending slew command.
setup.MTCS DEBUG: Scheduling check coroutines
setup.MTCS DEBUG: process as completed...
setup.MTCS DEBUG: Monitor position started.
setup.MTCS DEBUG: Waiting for Target event from mtmount.
setup.MTCS DEBUG: mtmount: <State.ENABLED: 2>
setup.MTCS DEBUG: mtptg: <State.ENABLED: 2>
setup.MTCS DEBUG: mtaos: <State.ENABLED: 2>
setup.MTCS DEBUG: mtm1m3: <State.ENABLED: 2>
setup.MTCS DEBUG: mtm2: <State.ENABLED: 2>
setup.MTCS DEBUG: mthexapod_1: <State.ENABLED: 2>
setup.MTCS DEBUG: mthexapod_2: <State.ENABLED: 2>
setup.MTCS DEBUG: mtrotator: <State.ENABLED: 2>
setup.MTCS DEBUG: mtdome: <State.ENABLED: 2>
setup.MTCS DEBUG: mtdometrajectory: <State.ENABLED: 2>
setup.MTCS DEBUG: Wait for mtmount in position events.
setup.MTCS DEBUG: Wait for dome in position event.
setup.MTCS DEBUG: Wait for MTRotator in position event.
setup.MTCS DEBUG: MTRotator in position: True.
setup.MTCS DEBUG: MTRotator already in position. Handling potential race
condition.
setup.MTCS DEBUG: Wait for MTMount elevation in position event.
setup.MTCS DEBUG: MTMount elevation in position: False.
setup.MTCS DEBUG: Wait for MTMount azimuth in position event.
setup.MTCS DEBUG: MTMount azimuth in position: False.
setup.MTCS DEBUG: Mount target: private_revCode: bdcb00ba, private_sndSt
amp: 1647976873.2471752, private_rcvStamp: 1647976873.247737, private_se
qNum: 8932, private_identity: MTMount, private_origin: 11345, elevation:
30.22061612731386, elevationVelocity: -0.00018974111014833205, azimuth:
183.0157543334537, azimuthVelocity: -6.684477842736761e-06, taiTime: 164
7976873.3055394, trackId: 1, tracksys: SIDEREAL, radesys: ICRS, priority
: 0
setup.MTCS DEBUG: [Tel]: Az = +120.833[ +62.2]; El = +060.413[ -30.2] [R
ot]: +001.660[  -0.0] [Dome] Az = +000.000; El = +000.000
setup.MTCS DEBUG: Dome azimuth in position.
```

```
setup.MTCS DEBUG: Dome elevation in position.
setup.MTCS INFO: MTRotator in position: False.
setup.MTCS INFO: MTRotator in position: True.
setup.MTCS DEBUG: MTRotator in position True. Waiting settle time 3.0s
setup.MTCS DEBUG: [Tel]: Az = +158.777[ +24.2]; El = +041.456[ -11.2] [Rot]: +000.000[  -0.0] [Dome] Az = +000.000; El = +000.000
setup.MTCS INFO: MTMount elevation in position: True.
setup.MTCS DEBUG: MTMount elevation in position True. Waiting settle time 3.0s
setup.MTCS INFO: MTMount azimuth in position: True.
setup.MTCS DEBUG: MTMount azimuth in position True. Waiting settle time 3.0s
setup.MTCS DEBUG: [Tel]: Az = +183.016[  +0.0]; El = +030.218[  +0.0] [Rot]: +000.000[  +0.0] [Dome] Az = +000.000; El = +000.000
```

Out[12]:
```
(<ICRS Coordinate: (ra, dec) in deg
     (307.07808333, -87.47219444)>,
 <Angle 0. deg>)
```

---

# 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.

In [13]:
```
exp_focus = await comcam.take_object(15)
print(f"Target exposure: {exp_focus}")
```

```
setup.ComCam DEBUG: Generating group_id
setup.ComCam DEBUG: imagetype: OBJECT, TCS synchronization not configured.
setup.ComCam DEBUG: OBJECT 0001 - 0001
Target exposure: [2022032200005]
```

---

# Intra Focus Position

Using the Camera Hexapod, piston ComCam +1mm

In [14]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

Out[14]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7fbc24149100>
```

---

## Intra Focus Image

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

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

```
setup.ComCam DEBUG: Generating group_id
setup.ComCam DEBUG: imagetype: OBJECT, TCS synchronization not configure
d.
setup.ComCam DEBUG: OBJECT 0001 - 0001
Target 1 exposure: [2022032200006]
```

---

## Extra Focus Position

Using the Camera Hexapod, piston ComCam to -1mm

In [16]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=-2000.)
```

Out[16]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7fbc2cf634c0>
```

---

## Extra Focus Image

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

In [17]:
```
exp_extra = await comcam.take_object(15)
print(f"Target 1 exposure: {exp_extra}")
```

```
setup.ComCam DEBUG: Generating group_id
setup.ComCam DEBUG: imagetype: OBJECT, TCS synchronization not configure
d.
setup.ComCam DEBUG: OBJECT 0001 - 0001
Target 1 exposure: [2022032200007]
```

## Go Back to Focus Position

Put the hexapod back to 0mm.

In [18]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

Out[18]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7fbc2cbef5e0>
```

## Stop Tracking

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

In [19]:
```
await mtcs.stop_tracking()
```

```
setup.MTCS DEBUG: Stop tracking.
```

## Get Zernike Coefficients

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

In [ ]:
```
await mtcs.rem.mtaos.cmd_runWEP.set_start(visitId=exp_intra[0] - 20211119000
                                          extraId=exp_extra[0] - 20211119000
```

## Get Corrections

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

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

---

## Issue the corrections

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

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

---

## 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.

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

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

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

In [ ]:
```
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())
```

In [ ]:
```
%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()
```

In [ ]:
```
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())
```

In [ ]:
```
%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()
```

---

## 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 0°**.

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

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

---

## 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.

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

---

## Intra Focus Position 2

Using the Camera Hexapod, piston ComCam +1mm.

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

## Intre Focus Image 2

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

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

## Extra Focus Position 2

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

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

## Extra Focus Image 2

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

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

## Go back to focus position 2

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

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

---

## Stop tracking 2

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

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

---

## 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.

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

---

## Get Corrections 2

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

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

---

## Issue the corrections 2

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

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

---

## Verify Offsets TBD</h2>

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?

---

## 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.

In [20]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtaos"])
```

```
setup.MTCS DEBUG: [mtaos]::[<State.ENABLED: 2>, <State.DISABLED: 1>, <St
ate.STANDBY: 5>]
setup.MTCS INFO: All components in <State.STANDBY: 5>.
```

In [21]:
```
await mtcs.lower_m1m3()
```

```
setup.MTCS DEBUG: M1M3 current detailed state {<DetailedState.ACTIVEENGI
NEERING: 11>, <DetailedState.ACTIVE: 7>}, executing command...
setup.MTCS DEBUG: process as completed...
setup.MTCS DEBUG: M1M3 detailed state 8
setup.MTCS DEBUG: mtm1m3: <State.ENABLED: 2>
setup.MTCS DEBUG: mtm1m3: <State.ENABLED: 2>
setup.MTCS DEBUG: M1M3 detailed state 5
```

In [22]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtm1m3"])
```

```
setup.MTCS DEBUG: [mtm1m3]::[<State.STANDBY: 5>, <State.OFFLINE: 4>]
setup.MTCS INFO: All components in <State.OFFLINE: 4>.
```

In [23]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtm2"])
```

```
setup.MTCS DEBUG: [mtm2]::[<State.ENABLED: 2>, <State.DISABLED: 1>, <Sta
te.STANDBY: 5>]
setup.MTCS INFO: All components in <State.STANDBY: 5>.
```

In [24]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mthexapod_1"])
```

```
setup.MTCS DEBUG: [mthexapod_1]::[<State.ENABLED: 2>, <State.DISABLED: 1
>, <State.STANDBY: 5>]
setup.MTCS INFO: All components in <State.STANDBY: 5>.
```

In [25]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mthexapod_2"])
```

```
setup.MTCS DEBUG: [mthexapod_2]::[<State.ENABLED: 2>, <State.DISABLED: 1
>, <State.STANDBY: 5>]
setup.MTCS INFO: All components in <State.STANDBY: 5>.
```

In [26]:
```
await mtcs.standby()
```

```
setup.MTCS DEBUG: [mtmount]::[<State.ENABLED: 2>, <State.DISABLED: 1>, <
State.STANDBY: 5>]
setup.MTCS DEBUG: [mtptg]::[<State.ENABLED: 2>, <State.DISABLED: 1>, <St
ate.STANDBY: 5>]
setup.MTCS DEBUG: [mtaos]::[<State.STANDBY: 5>]
setup.MTCS DEBUG: [mtm1m3]::[<State.STANDBY: 5>]
setup.MTCS DEBUG: [mtm2]::[<State.STANDBY: 5>]
setup.MTCS DEBUG: [mthexapod_1]::[<State.STANDBY: 5>]
setup.MTCS DEBUG: [mthexapod_2]::[<State.STANDBY: 5>]
setup.MTCS DEBUG: [mtrotator]::[<State.ENABLED: 2>, <State.DISABLED: 1>,
<State.STANDBY: 5>]
setup.MTCS DEBUG: [mtdome]::[<State.ENABLED: 2>, <State.DISABLED: 1>, <S
tate.STANDBY: 5>]
setup.MTCS DEBUG: [mtdometrajectory]::[<State.ENABLED: 2>, <State.DISABL
ED: 1>, <State.STANDBY: 5>]
setup.MTCS INFO: All components in <State.STANDBY: 5>.
```

In [27]:
```
await comcam.standby()
```

```
setup.ComCam DEBUG: [cccamera]::[<State.ENABLED: 2>, <State.DISABLED: 1>
, <State.STANDBY: 5>]
setup.ComCam DEBUG: [ccheaderservice]::[<State.ENABLED: 2>, <State.DISAB
LED: 1>, <State.STANDBY: 5>]
setup.ComCam DEBUG: [ccarchiver]::[<State.ENABLED: 2>, <State.DISABLED:
1>, <State.STANDBY: 5>]
setup.ComCam INFO: All components in <State.STANDBY: 5>.
```

In [29]:
```
await mtcs.set_state(salobj.State.OFFLINE, components=["mtmount"])
```

```
setup.MTCS DEBUG: [mtmount]::[<State.STANDBY: 5>, <State.OFFLINE: 4>]
```

```
setup.MTCS INFO: All components in <State.OFFLINE: 4>.
```

In [ ]: