IM(G): One time ComCam Image Ingestion and MTAOS Correction

This notebook is used to execute the LVV-T2228 (1.0) test script during System Spread Integration Tests on Level 3.
It is part of the plan LVV-P81 and of the test cycle 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.

In summary, you slew to a target and start tracking. Then you find the Wavefront Error as Zernike Coefficients,
convert them to corrections to be applied to M1M3, M2, Camera Hexapod and M2 Hexapod. Finally you stop tracking.

In [27]:
```
from lsst.ts import utils
import yaml


# 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}")
```

```
Executed by isotuela on 2022-05-13T16:53:08.234.
  At the summit? True
```

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

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

```
/tmp/ipykernel_16623/1665379685.py:2: DeprecationWarning: Call to deprecate
d function (or staticmethod) get_node. (Please use lsst.rsp.get_node())
  nb.utils.get_node()
```

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

In [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
```

```
WARNING: version mismatch between CFITSIO header (v4.000999999999999) and l
inked library (v4.01).


WARNING: version mismatch between CFITSIO header (v4.000999999999999) and l
inked library (v4.01).


WARNING: version mismatch between CFITSIO header (v4.000999999999999) and l
inked library (v4.01).
```

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

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

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

In [8]:
```
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.
```

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

```
MTHexapod INFO: Read historical data in 0.01 sec
MTHexapod INFO: Read historical data in 0.03 sec
```

Out[9]:
```
[None, None, None, None, None, None, None, None, None, None]
```

In [10]:
```
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: ccoods: Adding all resources.
```

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

```
MTHexapod.electrical WARNING: tel_electrical DDS read queue is filling:
11 of 100 elements
```

Out[11]:
```
[None, None, None]
```

```
MTHexapod.application WARNING: tel_application DDS read queue is fillin
g: 13 of 100 elements
MTHexapod.application WARNING: tel_application DDS read queue is fillin
g: 28 of 100 elements
MTHexapod.actuators WARNING: tel_actuators DDS read queue is filling: 13
of 100 elements
MTHexapod.actuators WARNING: tel_actuators DDS read queue is filling: 28
of 100 elements
```

In [12]:
```
await comcam.enable()
```

```
setup.ComCam INFO: Enabling all components
setup.ComCam DEBUG: Expand overrides None
setup.ComCam DEBUG: Complete overrides: {'cccamera': '', 'ccheaderservic
e': '', 'ccoods': ''}
setup.ComCam DEBUG: [cccamera]::[<State.ENABLED: 2>]
setup.ComCam DEBUG: [ccheaderservice]::[<State.ENABLED: 2>]
setup.ComCam DEBUG: [ccoods]::[<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 [14]:
```
await mtcs.slew_icrs(ra="20:28:18.74", dec="-87:28:19.9", rot_type=RotType.S
```

```
setup.MTCS DEBUG: RotSky = 0.0 deg, RotPhys = -73.91093506913535 deg.
```

```
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: 18.96
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: True.
setup.MTCS DEBUG: MTMount elevation already in position. Handling potent
ial race condition.
setup.MTCS DEBUG: Wait for MTMount azimuth in position event.
setup.MTCS DEBUG: MTMount azimuth in position: True.
setup.MTCS DEBUG: MTMount azimuth already in position. Handling potentia
l race condition.
setup.MTCS DEBUG: Mount target: private_revCode: bdcb00ba, private_sndSt
amp: 1652460314.2204893, private_rcvStamp: 1652460314.220718, private_se
qNum: 3271, private_identity: MTMount, private_origin: 35669, elevation:
29.679449211414607, elevationVelocity: -0.00018452426112961995, azimuth:
182.93287815426234, azimuthVelocity: -5.1905256766679586e-05, taiTime: 1
652460314.2793567, trackId: 2, tracksys: SIDEREAL, radesys: ICRS, priori
ty: 0
setup.MTCS INFO: MTMount elevation in position: False.
setup.MTCS INFO: MTMount azimuth in position: False.
setup.MTCS INFO: MTMount elevation in position: True.
setup.MTCS DEBUG: MTMount elevation in position True. Waiting settle tim
e 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 INFO: MTRotator in position: False.
setup.MTCS DEBUG: [Tel]: Az = +182.932[  +0.0]; El = +029.674[  +0.0] [R
ot]: +018.960[  +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 DEBUG: [Tel]: Az = +182.933[  -0.0]; El = +029.678[  -0.0] [R
ot]: +033.367[  -0.0] [Dome] Az = +000.000; El = +000.000
setup.MTCS DEBUG: [Tel]: Az = +182.932[  -0.0]; El = +029.677[  -0.0] [R
ot]: +054.519[  +0.0] [Dome] Az = +000.000; El = +000.000
setup.MTCS DEBUG: [Tel]: Az = +182.932[  -0.0]; El = +029.676[  -0.0] [R
ot]: +073.388[  +0.0] [Dome] Az = +000.000; El = +000.000
setup.MTCS INFO: MTRotator in position: True.
setup.MTCS DEBUG: MTRotator in position True. Waiting settle time 3.0s
```

Out[14]:
```
(<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 [15]:
```
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 configure
d.
Target exposure: [2022051300005]
```

Intra Focus Position

Using the Camera Hexapod, piston ComCam +1mm

In [16]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

Out[16]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7fab95a039a0>
```

Intra Focus Image

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

In [17]:
```
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.
Target 1 exposure: [2022051300006]
```

Extra Focus Position

Using the Camera Hexapod, piston ComCam to -1mm

In [18]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=-2000.)
```

Out[18]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7fabf482beb0>
```

Extra Focus Image

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

In [19]:
```
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.
CCHeaderService.logevent_logMessage ERROR: evt_logMessage DDS read queue
is full (100 elements); data may be lost
Target 1 exposure: [2022051300007]
```

Go Back to Focus Position

Put the hexapod back to 0mm.

In [20]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

Out[20]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7fab739829a0>
```

Stop Tracking

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

In [21]:
```
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 [28]:
```
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"
                }
            )
        )
    )
)
```

In [30]:
```
await mtcs.rem.mtaos.cmd_runWEP.set_start(visitId=exp_intra[0],
                                          extraId=exp_extra[0],
                                         config = wep_config)
```

```
---------------------------------------------------------------------------
AckError                                  Traceback (most recent call last)
Input In [30], in <cell line: 1>()
----> 1 await mtcs.rem.mtaos.cmd_runWEP.set_start(visitId=exp_intra[0],
      2                                           extraId=exp_extra[0],
      3                                          config = wep_config)

File /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe
-3.0.0/lib/python3.8/site-packages/lsst/ts/salobj/topics/remote_command.py:
418, in RemoteCommand.set_start(self, timeout, wait_done, **kwargs)
    377 """Create a new ``self.data``, set zero or more fields,
    378 and start the command.
    379
   (...)
    415     If ``data`` is not None and not an instance of `DataType`.
    416 """
    417 self.set(**kwargs)
--> 418 return await self.start(timeout=timeout, wait_done=wait_done)

File /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe
-3.0.0/lib/python3.8/site-packages/lsst/ts/salobj/topics/remote_command.py:
485, in RemoteCommand.start(self, data, timeout, wait_done)
    481 cmd_info = CommandInfo(
    482     remote_command=self, seq_num=seq_num, wait_done=wait_done
    483 )
    484 self.salinfo._running_cmds[seq_num] = cmd_info
--> 485 return await cmd_info.next_ackcmd(timeout=timeout)

File /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe
-3.0.0/lib/python3.8/site-packages/lsst/ts/salobj/topics/remote_command.py:
195, in CommandInfo.next_ackcmd(self, timeout)
    193     ackcmd = await self._wait_task
    194     if ackcmd.ack in self.failed_ack_codes:
--> 195         raise base.AckError(msg="Command failed", ackcmd=ackcmd)
    196     return ackcmd
    197 except asyncio.TimeoutError:

AckError: msg='Command failed', ackcmd=(ackcmd private_seqNum=1459207769, a
ck=<SalRetCode.CMD_FAILED: -302>, error=1, result='Failed: Error running pi
peline task: ')
```

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()
```

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

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

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

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

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

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

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

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