LVV-T2229

March 2, 2022

1 Closed Loop ComCam Image Ingestion and Application of Correction - Version 2.0

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.

[1]:
```
__executed_by__ = "Bruno Quint"  # Put your name here
__executed_on__ = "2022-03-02"  # Put the date of execution here

summit = True  # This is used later to define where Butler stores the images
```

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_53716/1665379685.py:2: DeprecationWarning: Call to deprecated
function (or staticmethod) get_node. (Please use lsst.rsp.get_node())
  nb.utils.get_node()
```

[3]:
```
'yagan07'
```

[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
```

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

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

[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>
```

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

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

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

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

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

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

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

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

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

[11]:
```
comcam.set_rem_loglevel(40)
```

[12]:
```
await comcam.start_task
```

[12]:
```
[None, None, None]
```

[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>
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Input In [14], in <module>
```

```
----> 1 await mtcs.slew_icrs(ra="20:28:18.74", dec="-87:28:19.9",␣
 ↪rot_type=RotType.Physical, rot=0)

File ~/auto-op-env-packages/ts_observatory_control/python/lsst/ts/observatory/
 ↪control/base_tcs.py:589, in BaseTCS.slew_icrs(self, ra, dec, rot, rot_type,␣
 ↪target_name, dra, ddec, offset_x, offset_y, az_wrap_strategy, time_on_target,␣
 ↪slew_timeout, stop_before_slew, wait_settle)
    584     valid_rottypes = ", ".join(repr(rt) for rt in RotType)
    585     raise RuntimeError(
    586         f"Unrecognized rottype {rot_type}. Should be one of␣
 ↪{valid_rottypes}"
    587     )
--> 589 await self.slew(
    590     radec_icrs.ra.hour,
    591     radec_icrs.dec.deg,
    592     rotPA=rot_angle.deg,
    593     target_name=target_name,
    594     frame=self.CoordFrame.ICRS,
    595     epoch=2000,
    596     equinox=2000,
    597     parallax=0,
    598     pmRA=0,
    599     pmDec=0,
    600     rv=0,
    601     dRA=dra,
    602     dDec=ddec,
    603     rot_frame=rot_frame,
    604     rot_track_frame=rot_track_frame,
    605     az_wrap_strategy=az_wrap_strategy,
    606     time_on_target=time_on_target,
    607     rot_mode=self.RotMode.FIELD,
    608     slew_timeout=slew_timeout,
    609     stop_before_slew=stop_before_slew,
    610     wait_settle=wait_settle,
    611     offset_x=offset_x,
    612     offset_y=offset_y,
    613 )
    615 return radec_icrs, rot_angle

File ~/auto-op-env-packages/ts_observatory_control/python/lsst/ts/observatory/
 ↪control/base_tcs.py:761, in BaseTCS.slew(self, ra, dec, rotPA, target_name,␣
 ↪frame, epoch, equinox, parallax, pmRA, pmDec, rv, dRA, dDec, rot_frame,␣
 ↪rot_track_frame, rot_mode, az_wrap_strategy, time_on_target, slew_timeout,␣
 ↪stop_before_slew, wait_settle, offset_x, offset_y)
    754 getattr(self.rem, self.ptg_name).cmd_poriginOffset.set(
    755     dx=offset_x * self.plate_scale,
    756     dy=offset_y * self.plate_scale,
    757     num=0,
    758 )
```

```
    760 try:
--> 761     await self._slew_to(
    762         getattr(self.rem, self.ptg_name).cmd_raDecTarget,
    763         slew_timeout=slew_timeout,
    764         offset_cmd=getattr(self.rem, self.ptg_name).cmd_poriginOffset,
    765         stop_before_slew=stop_before_slew,
    766         wait_settle=wait_settle,
    767     )
    768 except salobj.AckError as ack_err:
    769     self.log.error(
    770         f"Command to track target {target_name} rejected: {ack_err.
 ↪ackcmd.result}"
    771     )

File ~/auto-op-env-packages/ts_observatory_control/python/lsst/ts/observatory/
 ↪control/maintel/mtcs.py:292, in MTCS._slew_to(self, slew_cmd, slew_timeout,␣
 ↪offset_cmd, stop_before_slew, wait_settle, check)
    287         getattr(self.rem, comp).evt_summaryState.flush()
    288         self.scheduled_coro.append(
    289             asyncio.create_task(self.check_component_state(comp))
    290         )
--> 292 await self.process_as_completed(self.scheduled_coro)

File ~/auto-op-env-packages/ts_observatory_control/python/lsst/ts/observatory/
 ↪control/remote_group.py:1157, in RemoteGroup.process_as_completed(self, tasks
   1155 except Exception as e:
   1156     await self.cancel_not_done(tasks)
-> 1157     raise e
   1158 else:
   1159     await self.cancel_not_done(tasks)

File ~/auto-op-env-packages/ts_observatory_control/python/lsst/ts/observatory/
 ↪control/remote_group.py:1154, in RemoteGroup.process_as_completed(self, tasks
   1152 for res in asyncio.as_completed(tasks):
   1153     try:
-> 1154         ret_val = await res
   1155     except Exception as e:
   1156         await self.cancel_not_done(tasks)

File /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-2.0
 ↪0/lib/python3.8/asyncio/tasks.py:619, in as_completed.<locals>._wait_for_one(
    616 if f is None:
    617     # Dummy value from _on_timeout().
    618     raise exceptions.TimeoutError
--> 619 return f.result()
```

```
File ~/auto-op-env-packages/ts_observatory_control/python/lsst/ts/observatory/
 ↪control/remote_group.py:495, in RemoteGroup.check_component_state(self,␣
 ↪component, desired_state)
    493 if state != desired_state:
    494     self.log.warning(f"{component} not in {desired_state!r}: {state!r}"
--> 495     raise RuntimeError(
    496         f"{component} state is {state!r}, expected {desired_state!r}"
    497     )
    498 else:
    499     self.log.debug(f"{component}: {state!r}")

RuntimeError: mtrotator state is <State.FAULT: 3>, expected <State.ENABLED: 2>
```

[15]:
```
await mtcs.set_state(salobj.State.ENABLED, components=["mtptg"])
```

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

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

[18]:
```
await mtcs.set_state(salobj.State.ENABLED, components=["mtrotator"])
```

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

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

[19]:
```
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>

[19]:
```
(<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.

[20]:
```
exp_focus = await comcam.take_object(15, reason="LVV-T2229")
print(f"Target exposure: {exp_focus}")
```

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

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Target exposure: [2022030200005]
```

1.6 Intra Focus Position

Using the Camera Hexapod, piston ComCam +1mm

[21]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

[21]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f8ea4388970>
```

1.7 Intra Focus Image

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

[22]:
```
exp_intra = await comcam.take_object(15, reason="LVV-T2229")
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: [2022030200006]
```

1.8 Extra Focus Position

Using the Camera Hexapod, piston ComCam to -1mm

[23]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=-2000.)
```

[23]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f8ea43a1970>
```

1.9 Extra Focus Image

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

[25]:
```
exp_extra = await comcam.take_object(15, reason="LVV-T2229")
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: [2022030200008]
```

1.10 Go Back to Focus Position

Put the hexapod back to 0mm.

[26]:
```
await mtcs.rem.mthexapod_1.cmd_offset.set_start(z=1000.)
```

[26]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f8ebdc1c160>
```

1.11 Stop Tracking

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

[27]:
```
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.

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

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.

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

1.14 Issue the corrections

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

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

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.

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

[ ]:
```
registry = butler.registry

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

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

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

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

[ ]:
```
%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 cell is not currently included as part of the test execution, but included here as needed to shutdown the systems

[ ]:
```
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()
```