220204_LVV-T2229

February 11, 2022

1 Closed Loop ComCam Image Ingestion and Application of Correction

This notebook is used for the level 3 integration tests from test plan LVV-P81 (https://jira.lsstcorp.org/secure/Tests.jspa#/testPlan/LVV-P81) as part of test cylce LVV-C176 (https://jira.lsstcorp.org/secure/Tests.jspa#/testCycle/LVV-C176). The following tests are currently run as part of this notebook:

- LVV-T2229 (https://jira.lsstcorp.org/secure/Tests.jspa#/testCase/LVV-T2229)

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.

Last executed by B. Quint

Run the setup.ipnyb notebook to bring all components up and in their enabled position. Check Chronograph.

Bring ComCom online and tranistion it to EnabledState. Check Chronograph.

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

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

```
/tmp/ipykernel_11186/1665379685.py:2: DeprecationWarning: Call to deprecated
function (or staticmethod) get_node. (Please use lsst.rsp.get_node())
  nb.utils.get_node()
```

[2]:
```
'yagan05'
```

[23]:
```
import os
import sys
import asyncio
import logging
```

```
import pandas as pd
import numpy as np

from matplotlib import pyplot as plt

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

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

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

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

[7]:
```
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>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
```

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

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

[9]:
```
comcam = ComCam(domain=domain, log=log)
```

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

[10]:
```
comcam.set_rem_loglevel(40)
```

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

[11]: [None, None, None]

[12]:
```
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>
```

Find a target around az = 120º and el = 60º and rotator angle at PhysicalSky and 1.8º.

At this position, the rotator stays within a couple of degrees of its initial position. This is because the CCW is not running (MTmount in simulation mode).

target -> az = 120^o, el = 60^o

[13]:
```
target = await mtcs.find_target(az=120, el=60, mag_limit=8)

print(f"Target: {target}")
```

```
Target: HD 213827
```

Slew to target:

```
[14]: await mtcs.slew_object(target, rot_type=RotType.PhysicalSky, rot=1.9)
```

```
<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>
```

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.

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

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

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Target exposure: [2022020400005]
```

Using the Camera Hexapod, piston ComCam +1mm

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

[16]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f6d02733e20>
```

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

[17]:
```
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: [2022020400006]
```

Using the Camera Hexapod, piston ComCam to -1mm

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

[18]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f6cc7050d90>
```

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

[19]:
```
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: [2022020400007]
```

Put the hexapod back to 0mm.

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

[20]:
```
<ddsutil.MTHexapod_ackcmd_c4d6958b at 0x7f6cc7067130>
```

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

[21]:
```
await mtcs.stop_tracking()
```

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

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

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

await mtcs.rem.mtaos.cmd_runOFC.start(timeout=60.)

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

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

[ ]:
```
await mtcs.set_state(
    state=salobj.State.ENABLED,
    settings=dict(mtaos="impg"),
    components=["mtaos"]
    )
```

Process wavefront data

[24]:
```
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"
                }
            )
        )
    )
)
```

[30]:
```
await mtcs.rem.mtaos.cmd_runWEP.set_start(visitId=exp_intra[0] - 2021111900000,
                                          extraId=exp_extra[0] - 2021111900000,
                                         config=wep_config)
```

[30]:
```
<ddsutil.MTAOS_ackcmd_fd03e870 at 0x7f6ce0d4d6d0>
```

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

[31]:
```
await mtcs.rem.mtaos.cmd_runOFC.start(timeout=60.)
```

[31]: <ddsutil.MTAOS_ackcmd_fd03e870 at 0x7f6d027de820>

Issue the corrections

[32]:
```
await mtcs.rem.mtaos.cmd_issueCorrection.start(timeout=60.)
```

[32]: <ddsutil.MTAOS_ackcmd_fd03e870 at 0x7f6d02913b80>

<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>

Query the butler to verify that the images are there and check the metadata. This step must be verified using a separate noteboook.

1.1 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

[33]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtaos"])
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

[34]:
```
await mtcs.lower_m1m3()
```

<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>

[35]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtm1m3"])
```

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

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

[36]:
```
await mtcs.set_state(salobj.State.OFFLINE, components=["mtm1m3"])
```

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

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Input In [36], in <module>
----> 1 await mtcs.set_state(salobj.State.OFFLINE, components=["mtm1m3"])

File /opt/lsst/src/ts_observatory_control/python/lsst/ts/observatory/control/
 ↪remote_group.py:794, in RemoteGroup.set_state(self, state, settings,␣
 ↪components)
    791         self.log.debug(f"[{comp}]::{ret_val[i]!r}")
    793 if error_flag:
--> 794     raise RuntimeError(
    795         f"Failed to transition {failed_components} to "
    796         f"{salobj.State(state)!r}."
    797     )
    798 else:
    799     self.log.info(f"All components in {salobj.State(state)!r}.")

RuntimeError: Failed to transition ['mtm1m3'] to <State.OFFLINE: 4>.
```

[37]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtm2"])
```

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

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

[38]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mthexapod_1"])
```

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

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

[39]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mthexapod_2"])
```

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

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Input In [39], in <module>
----> 1 await mtcs.set_state(salobj.State.STANDBY, components=["mthexapod_2"])
```

```
File /opt/lsst/src/ts_observatory_control/python/lsst/ts/observatory/control/
 ↪remote_group.py:794, in RemoteGroup.set_state(self, state, settings,␣
 ↪components)
    791         self.log.debug(f"[{comp}]::{ret_val[i]!r}")
    793 if error_flag:
--> 794     raise RuntimeError(
    795         f"Failed to transition {failed_components} to "
    796         f"{salobj.State(state)!r}."
    797     )
    798 else:
    799     self.log.info(f"All components in {salobj.State(state)!r}.")

RuntimeError: Failed to transition ['mthexapod_2'] to <State.STANDBY: 5>.
```

[42]:
```
await mtcs.standby()
```

```
<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 [42], in <module>
----> 1 await mtcs.standby()

File /opt/lsst/src/ts_observatory_control/python/lsst/ts/observatory/control/
 ↪remote_group.py:905, in RemoteGroup.standby(self)
    902 async def standby(self):
    903     """Put all CSCs in standby."""
--> 905     await self.set_state(salobj.State.STANDBY)

File /opt/lsst/src/ts_observatory_control/python/lsst/ts/observatory/control/
 ↪remote_group.py:794, in RemoteGroup.set_state(self, state, settings,␣
 ↪components)
    791         self.log.debug(f"[{comp}]::{ret_val[i]!r}")
```

```
    793 if error_flag:
--> 794     raise RuntimeError(
    795         f"Failed to transition {failed_components} to "
    796         f"{salobj.State(state)!r}."
    797     )
    798 else:
    799     self.log.info(f"All components in {salobj.State(state)!r}.")

RuntimeError: Failed to transition ['mtptg', 'mtm1m3', 'mthexapod_2', 'mtdome',
 ↪'mtdometrajectory'] to <State.STANDBY: 5>.
```

[43]:
```
await comcam.standby()
```

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

[40]:
```
await mtcs.set_state(salobj.State.STANDBY, components=["mtrotator"])
```

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

[ ]:
```

```