OpNav tasks (FA21)

These tasks relate to data generated by the OpNav simulator (currently found on Box under Subsystems/Op-Nav/Simulation).

# Run simulation

Assignee: Everyone

Set up a Python environment with numpy, OpenCV, and pyquaternion. Download the simulator code. Download a trajectory file.

1. Create directory “out”. Modify “main.py” to run “test\_main()” instead of “main()”. Run “main.py”. Look at generated images in “out” and read JSON output.
2. Create directory “images”. Modify “main.py” to run “main()” again, and modify “main()” to read your chosen trajectory file. Run “main.py”. Look at generated images in “images”.
3. Modify “main.py” to set “illuminator” to “None” and to comment out “draw\_stereographic\_detections()”. Rerun “main.py”. Look at generated images in “images”.
4. Save JSON output to a file and clean it up to make it valid JSON (put “cameras” and “observations” in same object, fix commas).

For subsequent tasks, you may be able to use some of the outputs uploaded to Box (see “run\_index.md”), but you may need to modify parameters of the simulation (enabling or disabling Sun illumination, for example). You should be comfortable generating custom datasets to meet your testing needs.

# Finding and measuring circles

Owner: TBD

Reviewer: TBD

Process images referenced by “image\_stereographic” for frames in the dataset. Demonstrate detection of each body listed under “detections” for the frame. Measure center and size of each circle and compare to “center\_st” and “radius\_st” for the detection (after scaling by the camera’s stereographic\_scale and shifting by (width-1)/2 or (height-1)/2 as appropriate).

1. Do this for fully illuminated (“illuminator=None”) bodies.
2. Do this for Sun-illuminated bodies. Note any spurious detections.

# Reprojecting camera images

Owner: TBD

Reviewer: TBD

Process images referenced by “image\_gnomonic” from frames in the dataset. Using the spacecraft’s body rates for the observation, apply rolling shutter correction and reproject into stereographic coordinates. Compare resulting images to the corresponding “image\_stereographic” images. Determine appropriate coordinate scaling.

1. Do this using “omega\_cam” from the frame.
2. Independently compute “omega\_cam” from “omega\_body” from the observation and “q\_body2cam” for the camera. Compare against “omega\_cam” from the frame.
3. Measure circle centers and sizes and compare to “center\_st” and “radius\_st” with appropriate scaling (see “Finding and measuring circles” task).

Note: some bodies will appear in the “image\_stereographic” images but not in the “image\_gnomonic” images. This is expected, and you should not expect to find these bodies in your reprojected images.

# Interpreting body measurements

Owner: TBD

Reviewer: TBD

Given “center\_st” and “radius”st” from detections for frames in the dataset, use the formulas in “st\_circle\_inv()” and “stereographic\_inv()” to recover the unit vector pointing to the center of the object (in camera frame) as well as its angular size. Compare the unit vector to “direction\_cam” in the detection (note: the simulator’s convention for “camera frame” may not match other conventions in use). Compare the angular size to “angular\_size” for the observed body in the observation (note: “angular\_size” is an angular diameter, not an angular radius).

Then, using the “time” of the frame, the “omega\_body” of the spacecraft, and “q\_body2cam” for the camera, rotate the unit vector to the spacecraft’s body frame at the start of the observation. Compare with “direction\_body” for the observed body.

# Kalman filtering

Owner: TBD

Reviewer: TBD

TODO: Spacecraft velocity data is required to verify the Kalman filter.

Convert “direction\_body” and “angular\_size” for the observed bodies in a sequence of observations into the inputs required by the Kalman filter(s) (e.g. angular separation between bodies, etc.). For each observation, initialize the Kalman filter to the spacecraft state, provide the ephemeris from the bodies, ingest the observation data, and ensure that the Kalman filter does not wander away from the true state.

Then initialize the Kalman filter to the initial spacecraft state and run the Kalman filter, with propagation, on the sequence of ephemeris and observation data and ensure that the states follow the true states.

1. Do this for the translational filter only
2. Do this for the attitude filter, using “omega\_body” as the gyro measurements