SORTS_demo1

December 16, 2020

1 Live interaction demo of Space Object Radar Tracking Simulator (SORTS)

1.1 D. Kastinen, J. Vierinen, T. Grydeland, J. Kero

- We just heard about the filtering process
- How valuable radar time is and how many objects are out there
- Now we come to the question of "what schedule?" and "what data do we expect to get?"

[1]:

```
#Setup widgets
from __future__ import print_function
import ipywidgets as widgets
from IPython.display import display

#setup basic plotting
#%matplotlib inline
%matplotlib widget
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d

import numpy as np

#Load SORTS
import sorts

print(f'Using SORTS: {sorts.__version__}')
```

```
Using SORTS: 4.0.0-rc.1
```

2 Preliminaries

2.1 Space Objects

- What Tom just said!
- Its in all our interest to have sustainable usage of space!
- It could also pay for operation at E3D: the tracking data is a valuable comodity

2.2 E3D

For example: * The "module cutoff filter" will create data gaps at First Stage Beamforming * As Tom explained, we will know where objects are going to be… * Its all about planning! We can plan to avoid big targets in the beam (better ionosphere) * .. or plan to hit them (better multi-usage)!

3 SORTS

A toolbox for simulating radars, their schedule, objects in space and resulting measurements from those radars

- The idea behind this presentation: show how scheduling can be simulated with SORTS
- Took me about 1d to figure out Jupyter
- Took me about 1h to code the simulation using SORTS

SORTS on github * To get the code / contribute / suggest

SORTS examples (and docs) * For your inspiration

SORTS in peer-review! * Cool application!

SORTS and ESA projects * 10,000 - 20,000 debris passes per hour over E3D

[2]:

```
#Import our simulation into the notebook
from scheduler_simulation import calc_observation, lims, e3d_sched,␣
 ↪recalc_track_data
```

[6]:

```
output = widgets.Output()
with output:
    fig = plt.figure(figsize=(8,8))
    ax = fig.add_subplot(111, projection='3d')
    ax2 = fig.add_axes([.7, .1, .2, .2])

def fun(**kw):
    calc_observation(ax, ax2, **kw)

grid = widgets.GridspecLayout(8,6)

grid[:,:4] = output

###
# CREATE ALL THE SLIDERS AND UI.... AAAAAARGH!
###
t = widgets.FloatSlider(min=lims['t'][0], max=lims['t'][1], step=lims['t'][2],␣
 ↪description='Time [s]'); grid[1,4] = t
t_cnt = widgets.FloatText(min=lims['t'][0], max=lims['t'][1],␣
 ↪step=lims['t'][2], description='Time [s]'); grid[0,5] = t_cnt
t_span = widgets.FloatSlider(min=lims['t_span'][0], max=lims['t_span'][1],␣
 ↪step=lims['t_span'][2], description='Time span [s]'); grid[2,4] = t_span
view_range = widgets.FloatSlider(min=lims['view_range'][0],␣
 ↪max=lims['view_range'][1], step=lims['view_range'][2], description='Zoom␣
 ↪[km]'); grid[1,5] = view_range
d_min = widgets.FloatSlider(min=lims['d_min'][0], max=lims['d_min'][1],␣
 ↪step=lims['d_min'][2], value=5.0, description='Min diameter [m]'); grid[2,5]␣
 ↪= d_min
L = widgets.FloatSlider(min=lims['L'][0], max=lims['L'][1], step=lims['L'][2],␣
 ↪description='Side-length [km]'); grid[3,4] = L
N = widgets.IntSlider(min=lims['N'][0], max=lims['N'][1], step=lims['N'][2],␣
 ↪description='Samples/side [1]'); grid[3,5] = N
```

```
dwell = widgets.FloatSlider(min=lims['dwell'][0], max=lims['dwell'][1],␣
 ↪step=lims['dwell'][2], description='Dwell [s]'); grid[4,4] = dwell
h = widgets.FloatSlider(min=lims['h'][0], max=lims['h'][1], step=lims['h'][2],␣
 ↪description='Altitude [km]'); grid[4,5] = h
h_min = widgets.FloatSlider(min=lims['h_min'][0], max=lims['h_min'][1],␣
 ↪step=lims['h_min'][2], description='Min altitude [km]'); grid[5,4] = h_min
beams = widgets.IntSlider(min=lims['beams'][0], max=lims['beams'][1],␣
 ↪step=lims['beams'][2], description='Rx-beams [1]'); grid[5,5] = beams
plot_rx = widgets.ToggleButton(value=True, description='Plot rx beams');␣
 ↪grid[6,4] = plot_rx
include_objects = widgets.ToggleButton(value=False, description='Plot objects');
 ↪ grid[6,5] = include_objects
sim_snr = widgets.ToggleButton(value=False, description='Simulate observation');
 ↪ grid[7,4] = sim_snr
recalc = widgets.Button(description='Run Observation simulation',␣
 ↪button_style=''); grid[7,5] = recalc
track = widgets.Dropdown(
    options=['None'] + e3d_sched.inds,
    value = 1268,
    description='Track object:',
)
grid[0,4:5] = track


#Relations
def h_on_value_change(change):
    new_h = change['new']
    h_min.max = new_h
h.observe(h_on_value_change, names='value')

def on_button_clicked(b):
    recalc_track_data()
recalc.on_click(on_button_clicked)

tlink = widgets.jslink((t, 'value'), (t_cnt, 'value'))

#Cache interactive to start with
out = widgets.interactive_output(
    fun,
    dict(
        t = t,
        t_span = t_span,
        d_min = d_min,
        L = L,
        N = N,
        dwell = dwell,
        h = h,
        h_min = h_min,
        beams = beams,
        plot_rx = plot_rx,
        include_objects = include_objects,
        view_range = view_range,
        track_ind = track,
        track_snr = sim_snr,
    ),
)

#Display control
display(grid, out)
```

```
GridspecLayout(children=(Output(layout=Layout(grid_area='widget001')),␣
↪FloatSlider(value=0.0, description='Tim…
```

```
Output()
```

4 Discussion

4.1 Questions

- Thoughts about scheduling for your own future experiments? Lets discuss!
- Does this framework seem useful to your research? how? Poke me!
- Could it be useful with modification/additions? what modifications? Open an issue on Github!