Hit Detection Notebook

This notebook demonstrates how to find the vessels and times that a particular satellite was able to see.

Pre-requisites

This notebook presumes that data processing and precompute has already been completed. Please see the **Data Preparation** notebooks.

Required libraries for this notebook (not the pre-computation one):

- Numpy 1.19.2
- Pandas v1.1.5
- Numba 0.51.2
- Holoviews 1.14
- Bokeh 2.2.3
- Datashader 0.11.1
- PyTables 3.6.1
- Jupyter (core=4.7, client=6.1.7)

These should be installed via `conda`, from the default Anaconda repository (not Conda-Forge).

The `skyfield` library should be installed via `pip`, because it is very important that `skyfield` version is 1.35, and the `sgp4` library that it installs is v2.14; `conda` installs sgp4=2.10 as of 1/2021.

Data and Locations

The required data variables are described below:

Variable	Description	S3 location
`AIS_DIR`	Location of ais_????.h5 files or ais_????.interp.h5 files	`s3://anaconda-hit-finder-prod/AIS` or `s3://anaconda-hit-finder-prod/AIS_interp`
`SAT_DIR`	Location of precomputed satellite tracks.	`s3://anaconda-hit-finder-prod/satellites_active` or `s3://anaconda-hit-finder-prod/satellites_all`

The default values of `AIS_DIR` are set below.

A comment on hardware

This algorithm is heavily parallelized and can take advantage of all cores on the machine.

Development was done on a Macbook Pro with 4 cores and 16 GB of memory, and Amazon AWS EC2 instances of type `m5zn.6xlarge` and `t3.2xlarge`. The target deployment environment is a workstation-grade 8 or 16 core machine with 16GB of memory.

In [1]:
```
# Initial setup
# Make the notebook wider
from IPython.display import display, HTML

display(HTML(data="""
<style>
    div#notebook-container    { width: 95%; }
    div#menubar-container     { width: 65%; }
    div#maintoolbar-container { width: 99%; }
</style>
"""))
```

In [2]:
```
import os
import pandas as pd
```

In [3]:
```
# Set some configuration variables
# In general, these should be explicit paths with no variables or homedir (~)
AIS_DIR = "data/vessel data/Cleaned AIS"
SAT_DIR = "data/satellite data/index_active"

if not os.path.isdir(AIS_DIR) or not os.path.isdir(SAT_DIR):
    raise IOError("Invalid source data directory")
```

Step 0. Configure the input parameters

In [4]:
```
# The satellite we're interested in
norad_id = 25544  # The International Space Station

# The start and end times we're interested in.  For the sake of simplicity in
# this notebook, we are restricting to just one year.  The Python script
# is able to query multiple years.
start_time = pd.Timestamp("2014-12-31T00:00:01")
end_time = pd.Timestamp("2015-02-01T00:00:00")

# Based on the year of interest, also define the AIS file to look at
AIS_FILENAME = "ais_2015.h5"
```

Step 1. Load the satellite data

In [5]:
```
from scripts.sathelpers import SatelliteDataStore
satdata = SatelliteDataStore(SAT_DIR)

(times, lats, lons, alts) = satdata.get_precomputed_tracks(norad_id, start=start_time,
        end=end_time)
# The longitudes in the pre-computed satellite tracks range from 0-360,
# but we need them in (-180,180) format.
mask = lons > 180.0
lons[mask] -= 360

# Now convert to a dict that can by passed in to the intersection calculation
sat = pd.DataFrame({"date_time": times.astype("<M8[s]"),
       "lat": lats, "lon": lons, "alt": alts})
```

Step 2. Load the AIS data

Since the example in this notebook is from the period of time of 2015, we just need to load its AIS tracks.

In [6]:
```
ais = pd.read_hdf(os.path.join(AIS_DIR, AIS_FILENAME))
ais.sort_values(by="date_time", inplace=True)
ais.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3187260 entries, 309064 to 3186559
Data columns (total 4 columns):
 #   Column     Dtype
---  ------     -----
 0   mmsi_id    int64
 1   date_time  datetime64[ns]
 2   lat        float32
 3   lon        float32
dtypes: datetime64[ns](1), float32(2), int64(1)
memory usage: 97.3 MB
```

Step 3. Compute the visible points

In [7]:
```
from scripts import intersect; intersect.PRINT_INFO=True
```

In [8]:
```
hits = intersect.compute_hits(sat, ais, start_time="2015-01-01", end_time="2015-01-17", workers=4)
```

```
Time clipping truncated from 3187260 to 1275193 points.
Using 4 chunks
  compute time: 0.033619         extract time: 0.019459

/home/jbednar/vault/scripts/intersect.py:266: RuntimeWarning: invalid value encountered in _compute
  np.reshape(hit_mask[:totalsize], (-1, chunksize)))
/home/jbednar/vault/scripts/intersect.py:272: RuntimeWarning: invalid value encountered in _compute
  hit_mask[totalsize:])
```

In [9]:
```
len(hits)
```

Out[9]:
```
984645
```

In [10]:
```
hits2 = intersect.compute_hits(sat, ais, start_time="2015-01-01", end_time="2015-01-17",
                               workers=4, assume_half_earth=True)
```

```
Time clipping truncated from 3187260 to 1275193 points.
Using 4 chunks
  compute time: 0.030689         extract time: 0.019425
```

In [11]:
```
len(hits2)
```

Out[11]:
```
987514
```

Step 4. Visualize the results

In [17]:
```
import panel as pn
import datetime as dt
import holoviews as hv
from holoviews.util.transform import lon_lat_to_easting_northing as ll2en
from holoviews.operation.datashader import rasterize
hv.extension('bokeh')
```

In [18]:
```
from scripts import plot_helpers
```

In [25]:
```
t = plot_helpers.plot_points(hits)
#t2 = plot_helpers.plot_points(hits2)

x_range, y_range = ll2en([-205,-135], [10,60])
bounds = dict(x=tuple(x_range), y=tuple(y_range))
points = rasterize(t).redim.range(**bounds)
```

In [26]:
```
pn.Row(points).servable()
```

Out[26]: