STS Cargo Transfer

What do you think about the provided data set?

The provided data set is a sample of vessel positions and status over two days. It does not include ground truth regarding whether STS takes place or not, as it does not seem to be public information. So there is no way to apply classical machine learning prediction here. However, the sample is large enough to give an idea of how things work. I wrote a Python script to help visualization and try to detect STS with a simple method (see cargo.py).

Which variable seems to be useful or useless to you and why?

Variables I consider useful are:

- latitude, longitude and received_time_utc: they can be used directly to find if two vessels are in the same neighborhood at the same moment
- speed, course, heading: STS would happen at reduced speed or without moving, and we can expect the vessels to be aligned.
- draught: this is directly related to the load of the vessel. Change in draught is very likely to be due to a change in load. However this is *manually* set by the crew, so we can expect this variable to have low reliability (e.g. the crew forgets to set it, or human mistakes)
- new_navigational_status: the status could be an indicator of STS. If some status are unrelated to STS (e.g. engaged in fishing), it is less clear how to use without domain knowledge (and ground truth for STS).

Variables I would discard for the analysis are:

- provider_id, added_at, added_by, updated_at, updated_by, point: these are related to database operations and unlikely to be useful for STS detection.

What kind of parameter could we use to describe or discriminate a STS from other shipping operations?

First, STS involves two ships that operate side by side. It is necessary for the vessels to stay together for long enough (e.g. at least 1h). Speed and orientation of the vessels should also be the same. But it is not sufficient. In my Python script I use such rule to detect *potential* STS transfers.

Apart from position and time parameters, draught indicates the load of ship and would be a useful indicator of STS transfer.

Finally, if we trace the complete trajectory of two vessels with potential STS, we can expect that one would go to the terminal while the other would not. We can then validate a potential STS transfer.

How would you build the model for STS detection:

- **How would you represent a STS (as either a mathematical or programming object)**
- **Could you build a prototype algorithm for the STS detection**

I would represent a STS as a pair of vessel ids along with start/end timestamps in indicating the duration of STS, and the positions and status of the vessels. Additional information could be origin and destination of a ship (to help discriminating true/false positives using the fact that STS serves the purpose of transferring from/to a terminal).

I built a prototype of STS detection algorithm in a Python script. I coded an interactive display of vessel position and trajectories with potential STS detection. STS detection is based on continuous neighboring detection: I used a k-d tree data structure to find closest pairs of vessels within a certain range efficiently, and if they constantly remain neighbors they are classified as potential STS. Additional data is printed in the console while running the script so we can review the detected STS (looking at variables such as status and draught to refine the classification).

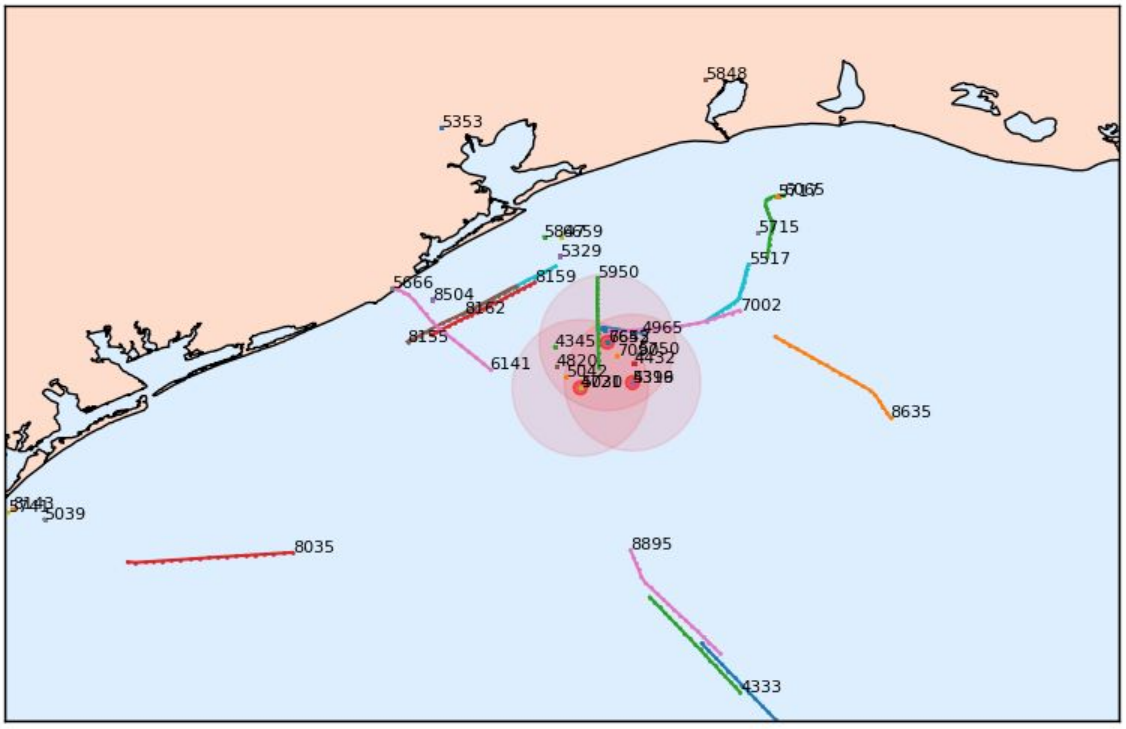
For technical purposes, data is resampled every 10 minutes and trajectories are interpolated linearly. Potential STS are displayed as red circles on the map, and we can zoom in to see exactly the vessels involved.

Is there any vessel doing STS in the given data set, if yes with which vessel

My script detects most of potential STS in a well defined area in the sea at the South of what appears to be a terminal. Some false positive occur for ships being at terminals. But these are easy to eliminate using geographical data (these ships are not at sea). Removing these obvious false positives, the potential STS detected by my algorithm are:

- 4345 - 5847 from 5:00 to 11:30 on 25/11/2017
- 6065 - 7090 from 10:00 to 19:00 on 25/11/2017 (draught varies for 7090)
- 4432 - 5517 from 9:00 on 25/11/2017 to 4:00 on 26/11/2017
- 4316 - 5715 from 9:00 on 25/11/2017 to 0:30 on 26/11/2017
- 4965 - 7002 from 5:30 to 8:00 on 26/11/2017
- 6653 - 7642 from 6:30 to 23:59 on 26/11/2017 (draught varies for 6653 and 7642)
- 4721 - 5030 from 7:30 to 23:59 on 26/11/2017 (draught varies for 4721)
- 4316 - 5399 from 6:00 to 22:00 on 26/11/2017

Please find below a screenshot of the tool I coded to detect STS:



from 09:10 (26/11/2017) to 12:10 (26/11/2017)

time 217.00

Oil Product Storage Pipeline Architecture

The purpose is to estimate storage variations from observation of petrol tanks with moving roofs. Raw data we receive are satellite images of the sites. I would divide the tasks into two subtasks:

1. Detecting tanks on the image
2. Estimating level for each tank

Regarding the first step, even though we monitor a few sites with an architecture unlikely to change quickly (i.e. it takes some time to build a new tank), image from the satellite may not be guaranteed to be taken from the same angle every time. So finding the tanks on the image is not straightforward (I assume that the only data we receive are raw images without precise geo-localization of objects, if geographical coordinates are available on the image, then step 1 is much easier to achieve: we just need to mark tanks once for all).

To achieve step 1, we can use techniques from image recognition, i.e. segment the image with a grid a find if a tank is on a patch or not (like face detection algorithms). Because light conditions change, an simple model can easily be fooled, and neural networks are well adapted to these tasks: I would train a convolutional neural network for this (using Tensorflow and GPU hardware). It would require manual annotation of image patches, but given we are trying to detect only tanks, it would not be too costly (however we need to sample at different times and weather conditions to get a representative dataset). An improvement of step 1 could be to align detected objects on a map to improve accuracy (once aligned, we can remove false positives and add false negatives).

Step 2 consists in estimating the level of each detected tank. The idea is to produce the estimation by measuring the shadow of the edge of the tank on the roof: the longer the shadow is the lower the floating roof is. However this depends on the position of the sun, hence the time of the day and day of the year. And weather conditions may vary and make it difficult to do an automated measurement of this shadow. Again, we could rely on machine learning models to predict roughly the level of oil in the tank (e.g. 0% 25% 50% 75% 100%). Still using neural networks, this would require further manual annotations to build a dataset.

Finally having estimated the level of oil for each tank, and knowing the volume of a tank (e.g. via direct measurement), the total amount of stored oil can be estimated.