Team HydrAA

TRACK 2: Data Science and the Seven Seas: Collision Avoidance

HACKtheMACHINE SEATTLE

September 21-23, 2018

Overall Approach

- Diverse Team
- Divide and Conquer
- Software, Applications, Methodology

Challenge 1: Identifying Interactions Between Ships

- Filtering (Eric)
  - Data is small enough to load into memory using Pandas
  - Data filtered (based on mentor advice) also to quickly identifier higher risk situations
    - Speeds must be greater than or equal to 4 knots
    - No tugs
    - Anchored and Moored ship's removed
    - Geospatial filtering would have been added with more time
- Time Window Analysis
  - Data split into 30 minute windows with 15 minute overlaps
  - All position data within those windows are calculated using pairwise euclidians differences
  - Further filtered to 4nmi interactions and then analyzed using...
- Right of Way Categorization (from challenge2)
  - Responsibilities Between Vessels
    - Associated hierarchy, mapping to VesselTypes

Challenge 1

In [6]:

```
time_start = df['BaseDateTime'].min()
time_delta = (df['BaseDateTime'].max()-df['BaseDateTime'].min()).total_seconds()/60
time_window = 30
time_step = 15
interactions_list = []
interaction_number = 0

for t in tqdm(np.arange(0, time_delta, time_step)):
    time_step_idx = np.all(np.vstack([
        df['BaseDateTime'] > time_start + pd.Timedelta(minutes=t),
        df['BaseDateTime'] < time_start + pd.Timedelta(minutes=t+time_window)
        ]
       ), axis=0)
    df_sub = df[time_step_idx]
    #Spatial distances are calculated
    distances = np.triu(spatial.distance.squareform(spatial.distance.pdist(df_sub.iloc[:, 2:4])))
    distances[distances > 0.067] = 0
    pairs = np.nonzero(distances)
    #Remove distances of ship relative to self
    MMSI_pairs = (df_sub['MMSI'].iloc[pairs[0]], df_sub['MMSI'].iloc[pairs[1]])
    ship_non_self_idx = np.argwhere(MMSI_pairs[0].values!=MMSI_pairs[1].values)
    ship_pairs = np.array([pairs[0][ship_non_self_idx], pairs[1][ship_non_self_idx]]).T[0]
    #Find each unique interaction
    mmsi_ship_pairs = df_sub['MMSI'].values[ship_pairs]
    if len(mmsi_ship_pairs) == 0:
        continue
    interactions = np.unique(mmsi_ship_pairs, axis=0)
    for pair in interactions:
        ship1 = df_sub[df_sub['MMSI'] == pair[0]]
        ship2 = df_sub[df_sub['MMSI'] == pair[1]]

        out = [ship1['MMSI'].values[0],
               ship2['MMSI'].values[0],
               np.mean([ship1['LAT'].values[0], ship2['LAT'].values[0]]),
               np.mean([ship1['LON'].values[0], ship2['LON'].values[0]]),
               ship1['BaseDateTime'].values[0],
               detect_interaction(ship1, ship2)
              ]
        interactions_list.append(out)
```

Challenge 2: Behavioral Model for Ships' Interactions

- Using AIS Data Elements to Fill In Missing
  - Inference and relationship between Status, VesselType
  - Data aggregation via addition of VesselGroup data element for high-level consumption and analysis
  - Hierarchical behavior – deviation from what ships were *supposed* to do

- Categorizing Encounters
  - Crossing, Head-On, Overtaking
    - Classification based on direction of approach and projected motion

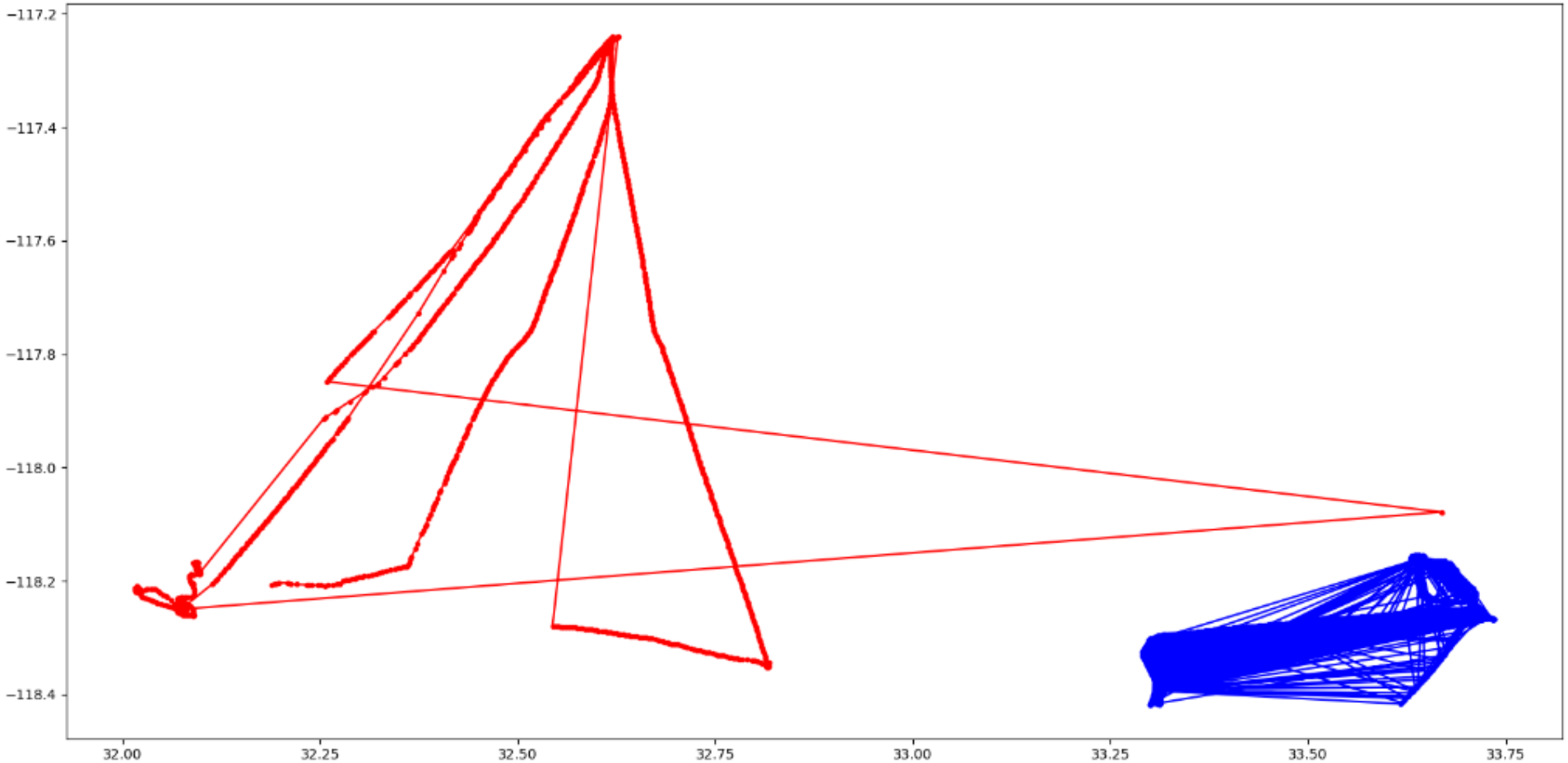
Challenge 2: Behavioral Model for Ships' Interactions

In [5]:

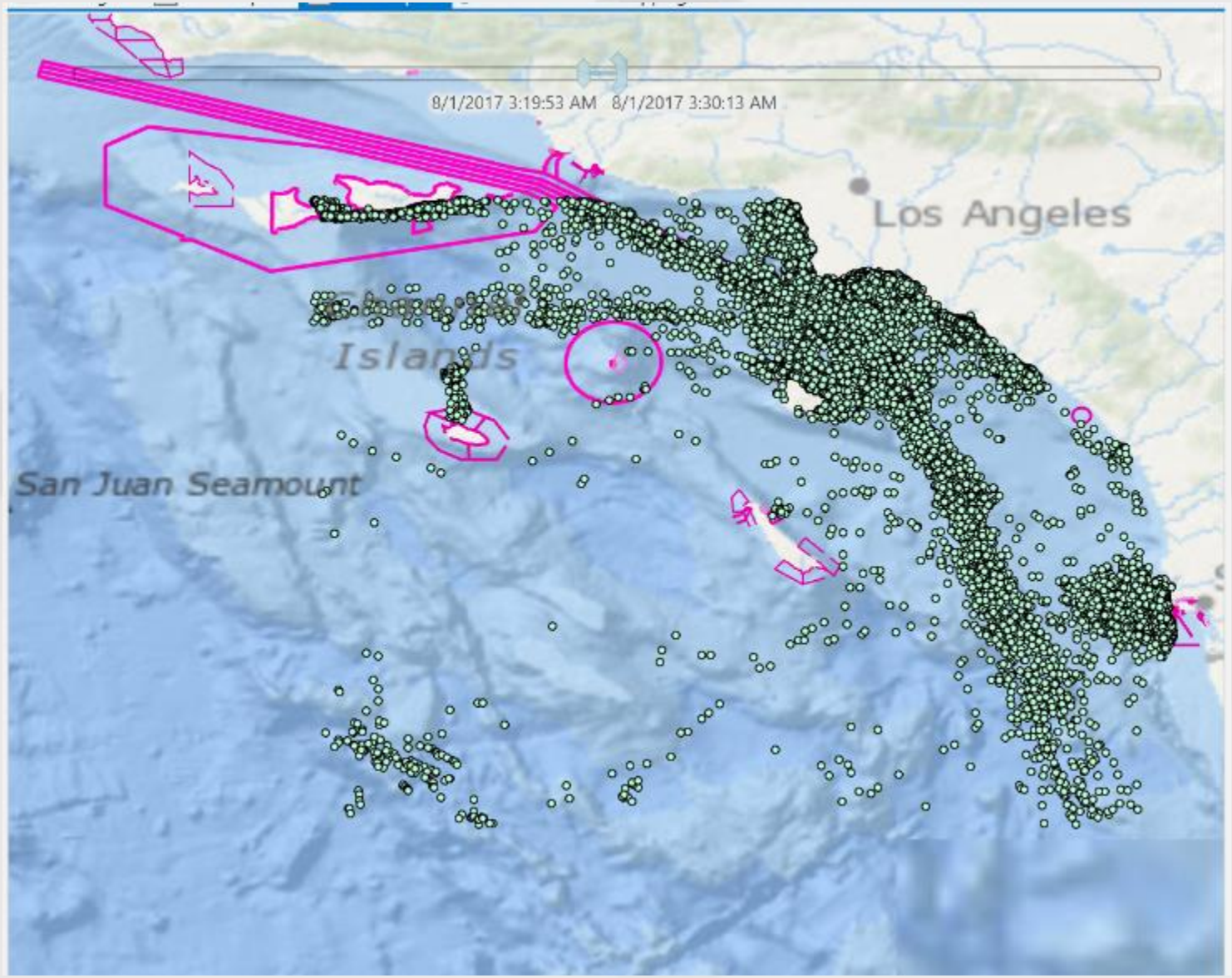
```
def detect_interaction(ship1, ship2):
    def norm_angle(ang):
        if ang < 0:
            ang += 360
        if ang > 360:
            ang -= 360
        return ang

    init1 = (ship1['LAT'].values[0], ship1['LON'].values[0])
    init2 = (ship2['LAT'].values[0], ship2['LON'].values[0])
    bear1 = ship1['COG'].values[0]
    bear2 = ship2['COG'].values[0]
    ship1_behind = norm_angle(bear1-180)
    ship2_behind = norm_angle(bear2-180)
    x1, y1, _, _ = utm.from_latlon(init1[0], init1[1])
    x2, y2, _, _ = utm.from_latlon(init2[0], init2[1])
    dx = x2 - x1
    dy = y2 - y1
    angle1 = 180*np.arctan2(dy, dx)/np.pi*-1+90
    angle2 = 180*np.arctan2(-dy, -dx)/np.pi*-1+90
    if (angle1 < ship1_behind+67.5 and angle1 > ship1_behind-67.5) or (angle2 < ship2_behind+67.5 and angl
e2 > ship2_behind-67.5):
        return 'Overtaking'
    elif (angle1 < bear1+10 and angle1 > bear1-10) or (angle2 < bear2+10 and angle2 > bear2-10):
        return 'Head-On'
    else:
        return 'Crossing'
```

Outliers


−117.2
−117.4
−117.6
−117.8
−118.0
−118.2
−118.4
32.00
32.25
32.50
32.75
33.00
33.25
33.50
33.75

Results


8/1/2017 3:19:53 AM
8/1/2017 3:30:13 AM
Los Angeles
Islands
San Juan Seamount