[SV30 프로젝트] 선박 GIS 데이터 전처리 파이프라인 개발 및 최적 항적 추출

2023.10.15

사용한 데이터셋 및 라이브러리

- LTEM_MAP_2022062811.csv
- LTEM_MAP_2022062812.csv
- LTEM_MAP_2022062813.csv
- LTEM_MAP_2022062814.csv
- LTEM_MAP_2022062815.csv
- LTEM_MAP_2022062816.csv
- LTEM_MAP_2022062817.csv
- LTEM_MAP_2022062818.csv
- LTEM_MAP_2022062819.csv
- LTEM_MAP_2022062820.csv
- LTEM_MAP_2022062821.csv
- LTEM_MAP_2022062822.csv
- LTEM_MAP_2022062823.csv
- LTEM_MAP_2022062900.csv
- LTEM_MAP_2022062901.csv
- LTEM_MAP_2022062902.csv
- LTEM_MAP_2022062903.csv
- LTEM_MAP_2022062904.csv
- LTEM_MAP_2022062905.csv
- LTEM_MAP_2022062906.csv
- LTEM_MAP_2022062907.csv
- LTEM_MAP_2022062908.csv
- LTEM_MAP_2022062909.csv
- LTEM_MAP_2022062910.csv
- LTEM_MAP_2022062911.csv

```
all_csv = glob(  해당 경로는 회사 내부 보안 문제로 공개 불가 합니다.  *.csv')
all_csv = sorted(all_csv)
selected_csv = all_csv[0:25]

total = pd.DataFrame()

for sel in selected_csv:
    temp = pd.read_csv(sel, sep = ',', encoding='utf-8')
    total = pd.concat([total, temp])
```

해당 경로에 있는
2022/06/28/11 ~
2022/06/29/11

24시간치 데이터 사용
약 1600만 row

```
total
```

	szMsgSendDT	SHIP_CODE	dSOG	dCOG	dLat	dLon
0	20220628110654000	BE010da2	6.331898	149.0	36.965206	126.828316
1	20220628110654000	AB110b5d	10.100000	272.0	37.874233	129.008636
2	20220628110654000	AB110b8e	0.000000	227.0	38.499546	128.425110
3	20220628110654000	AB0908a0	0.000000	0.0	34.556713	127.675613
4	20220628110654000	BD010d54	0.000000	275.0	35.978321	126.622986
...	...	...	...	...	...	...
738051	20220629115959000	AB02044d	9.479563	291.0	36.348801	129.429718
738052	20220629115959000	AB08064b	0.000000	333.0	36.249516	126.536850
738053	20220629115959000	BE020e3f	7.910589	173.0	35.364128	125.724831
738054	20220629115959000	AB010107	0.000000	256.0	36.675259	126.128418
738055	20220629115959000	AB0203fd	6.647376	60.0	36.263489	129.402298

16020592 rows × 6 columns

```
import pandas as pd
import folium as f
import os
import glob
import matplotlib.pyplot as plt
from glob import glob
import seaborn as sns
import datetime
import random
import numpy as np
from haversine import haversine
from sklearn.cluster import DBSCAN
import itertools
```

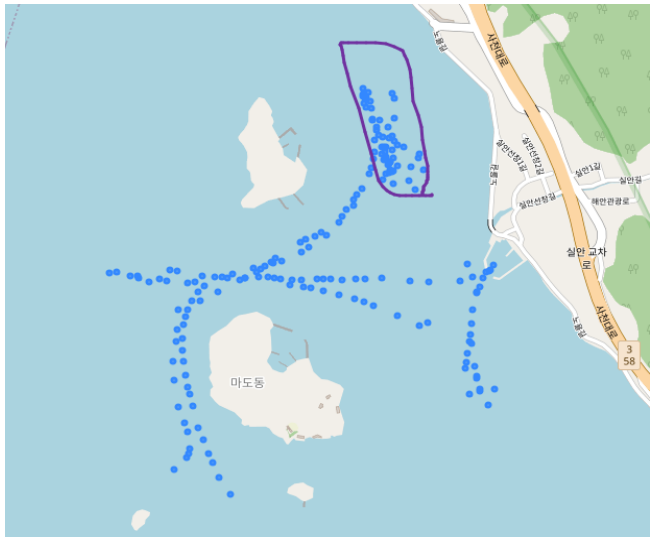
SV30 데이터의 문제점

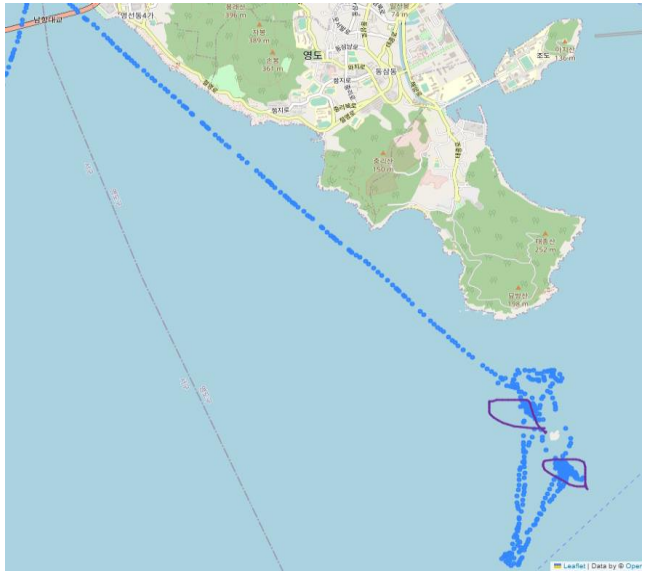
일반적이지 않은 동선이 매우 많다

1. 움직임이 전혀 없는 정적인 데이터

2. 해류와 같은 자연현상으로 인하여 미세한 움직임이 존재하는 데이터

3. 동적이지만 고기 잡이를 위해 일반적이지 않은 동선을 가지는 데이터





데이터 전처리 1

[가설 1] SOG < 2 인 선박은 움직임이 없을 것이다

해당 가설은 도메인 전문가의 의견 입니다

* SOG 는 선박의 속도를 의미 합니다

SOG < 2 인 데이터를 필터링 한 결과

1. 항구에 주차하고 있는 선박
2. 바다 한가운데 정적인 위치 데이터

제거 성공

735783	20220629115949000	AB08068d	0.000000	0.0	34.599522	127.742950
736571	20220629115952000	AB08068d	0.000000	0.0	34.599522	127.742950

12456 rows × 6 columns

696017	20220628145659000	AB08068d	4.641284	54.0	34.599586	127.742584
696647	20220628145703000	AB08068d	4.064362	58.0	34.599640	127.742668
697553	20220628145709000	AB08068d	3.163625	76.0	34.599670	127.742737

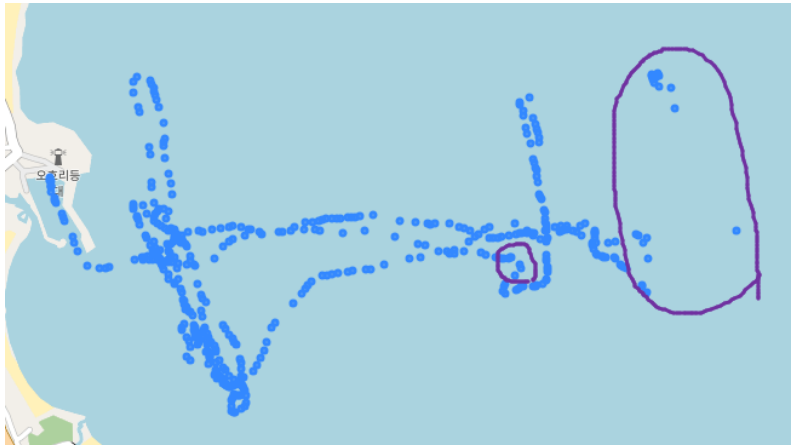
838 rows × 6 columns

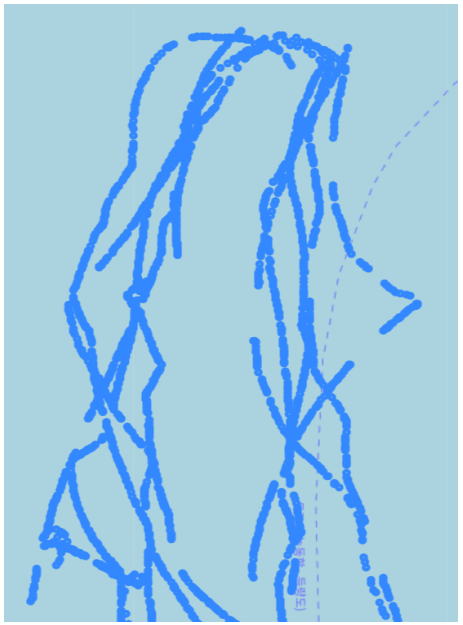
대부분의 데이터셋 에서 90% 이상이
움직임이 없는 데이터임을 확인

데이터 전처리 1

[가설 1] SOG < 2 인 선박은 움직임이 없을 것이다

가설 1의 한계점 :
해류와 같은 자연현상으로 인한 미세한 움직임
+ 고기 잡이로 인한 일반적이지 않은 동선들은
여전히 존재



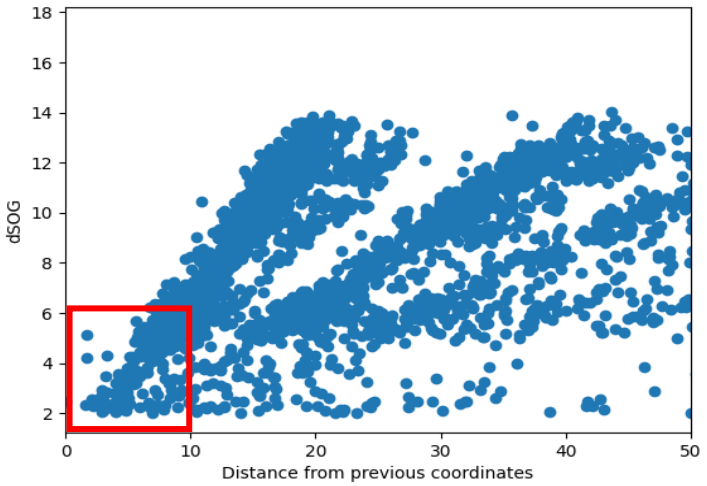


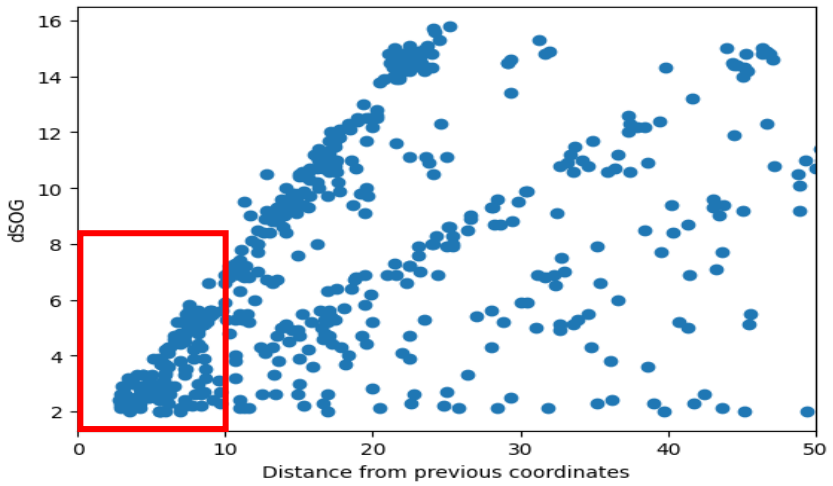
데이터 전처리 2

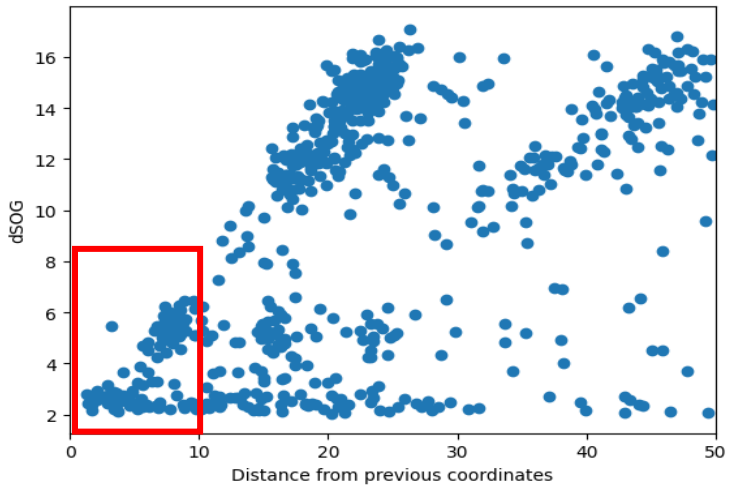
[가설 2] 움직임이 적은 동선은 직전 좌표와의 거리가 10m 이하일 것이다.

전처리 기준 값을 위한 EDA
X : 전 좌표와의 거리 (haversine)
Y: SOG

* Haversine 은 두 위경도 좌표 사이의 거리를 구할 때 사용하는 공식 입니다.
지구가 원인 것을 감안하여 만든 두 좌표 사이 직선 거리 입니다.







- 가설 설정 및 EDA 결과

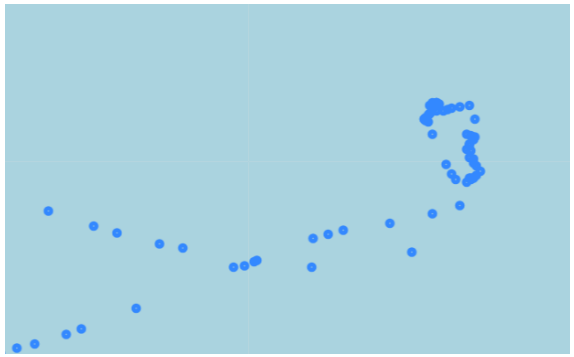
직선과 가까운 일반적인 동선은
SOG도 높을 것이고 이전 좌표와의 거리도 상대적으로 클 것으로 예상
하지만 고기잡이 주변 동선은 상대적으로 SOG도 낮을 것이고
직전 좌표와의 거리도 작을 것

EDA 결과 -> 상대적으로 속도도 낮고 거리도 낮은 데이터들의 분포가 distance 10m 이하에 분포 되어 있음
해당 데이터들이 움직임이 적은 쓰레기 동선 (뭉텅이 데이터)이라고 판단하여 이를 전처리 하고자 함 (일부 원본 손실은 피할 수 없음)

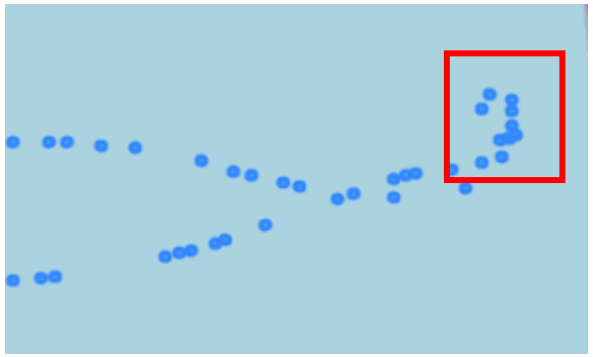
데이터 전처리 2

[가설 2] 움직임이 적은 동선은 직전 좌표와의 거리가 10m 이하일 것이다.

가설 2의 한계점 :
미세한 움직임 (뭉터기 등)은 대부분 제거 되지만,
고기 잡이로 인한 일반적이지 않은 동선들은
여전히 존재



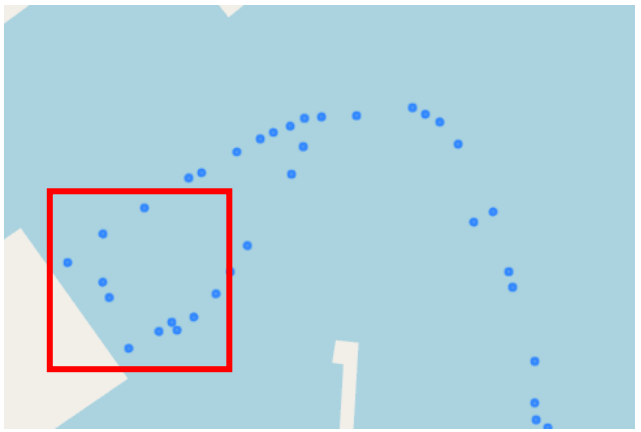
< SOG > 2>



< SOG > 2 + Distance > 10 >

<원본>

<SOG > 2 >



< SOG > 2 + Distance > 10 >

선박 : AB0102c0

데이터 전처리 3

[가설 3] 동선 중 가장 긴 직선 3가지 동선이 가장 정상적인 동선일 것이다

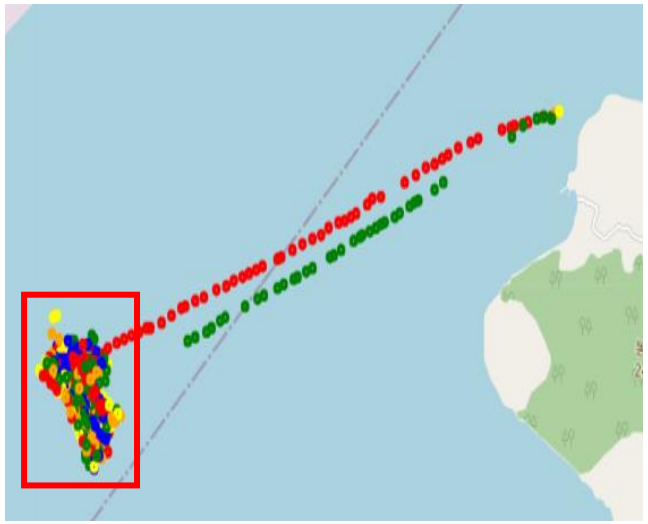
* 가설 설정 및 EDA

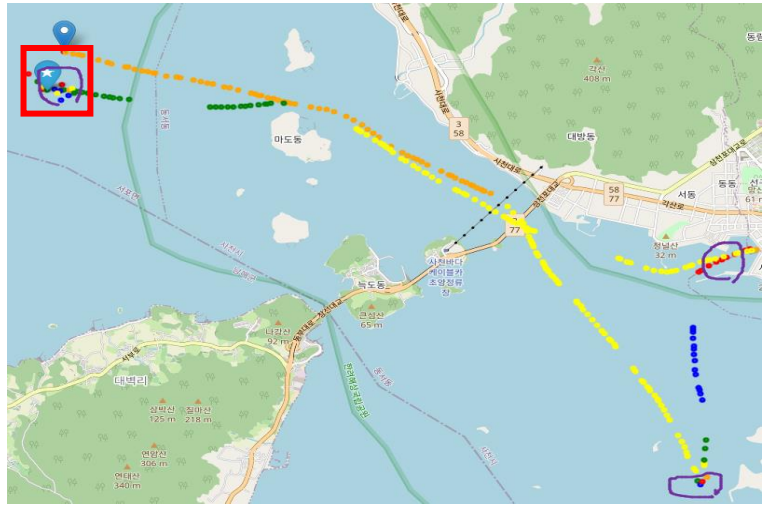
동선을 folium 으로 시각화 하면서 tooltip 으로 cog 변화량을 체크 해 본 결과,
정상적인 동선에서는 전 좌표와의 cog 변화 값이 매우 작았지만, 고기잡이와 같은 비정상 동선에 대해서는 한 공간 내에서 cog 변화량이 큰 경향을 보이는 것을 확인했다.

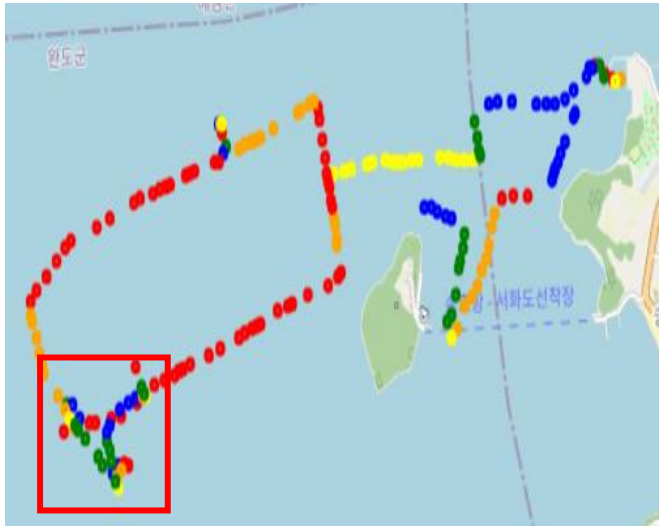
따라서, cog 차이가 20보다 큰 경우, 점의 색을 바꾸어서 시각화를 해보았고 그림에서 보다시피 직선에 가까운 일반적인 동선들은 같은 색으로 점이 찍히는 것을 확인 할 수 있다.

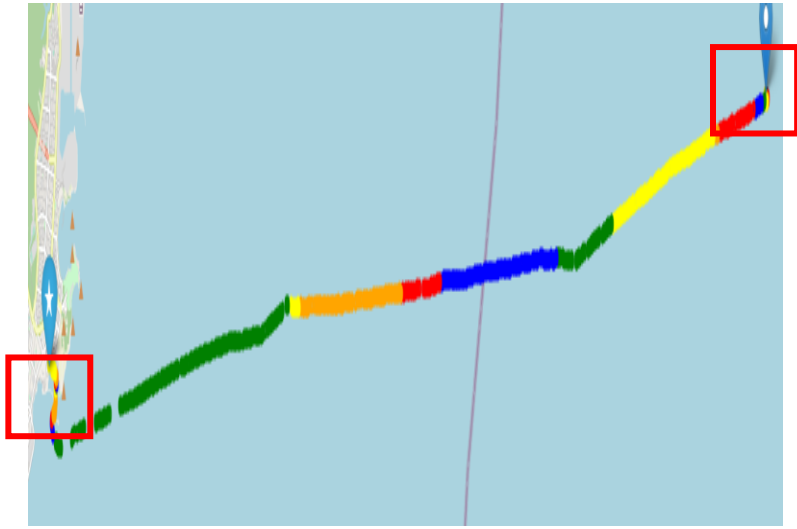
cog 변화량으로 색을 바꾸어 시각화 하기 위해서 전 좌표와의 cog 값의 차이의 절댓값을 dcog_diff 라는 feature 를 파생변수로 생성 하였다.

색이 같은 가장 긴 동선 3가지를 추출 하는 이유는 목적지 까지 가는길 + 돌아오는 길 + 정상 동선의 손실 방지 이다.









데이터 전처리 3

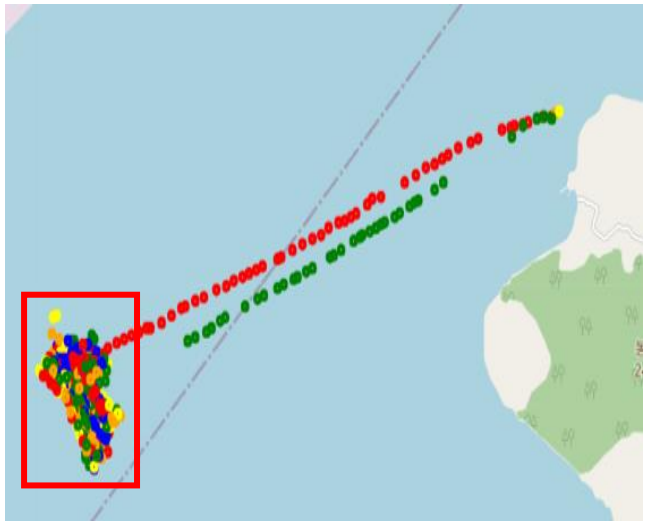
[가설 3] 동선 중 가장 긴 직선 3가지 동선이 가장 정상적인 동선일 것이다

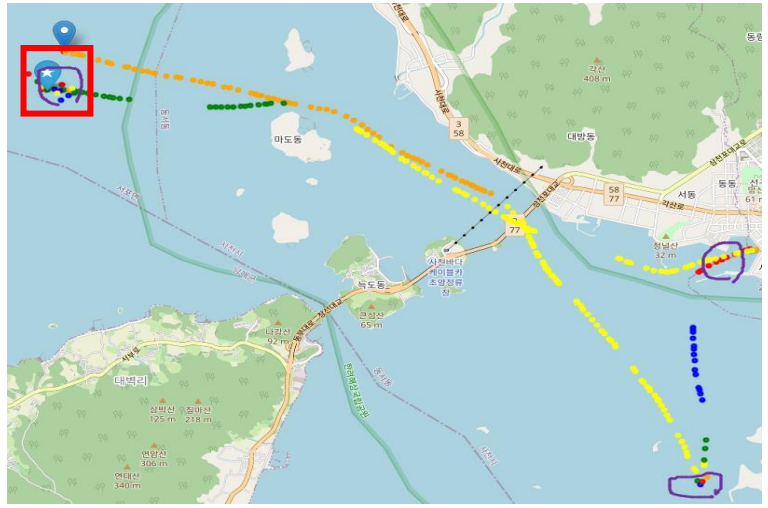
개발 알고리즘 :

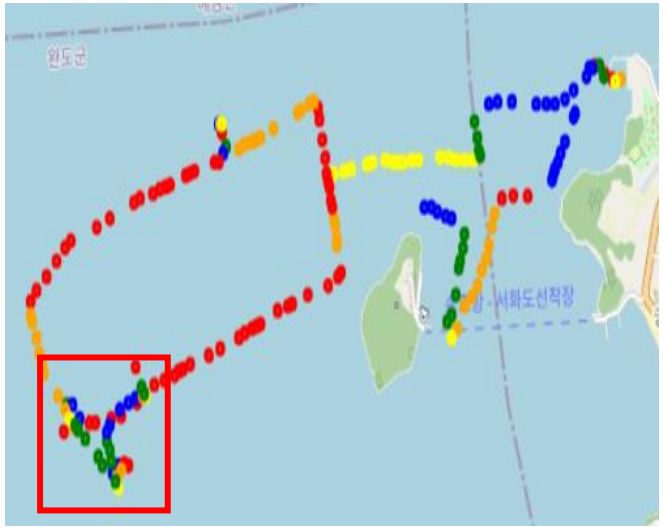
전 위치 좌표와 COG 차이가 20보다 큰 경우,
점의 색깔을 다르게 하여서 시각화

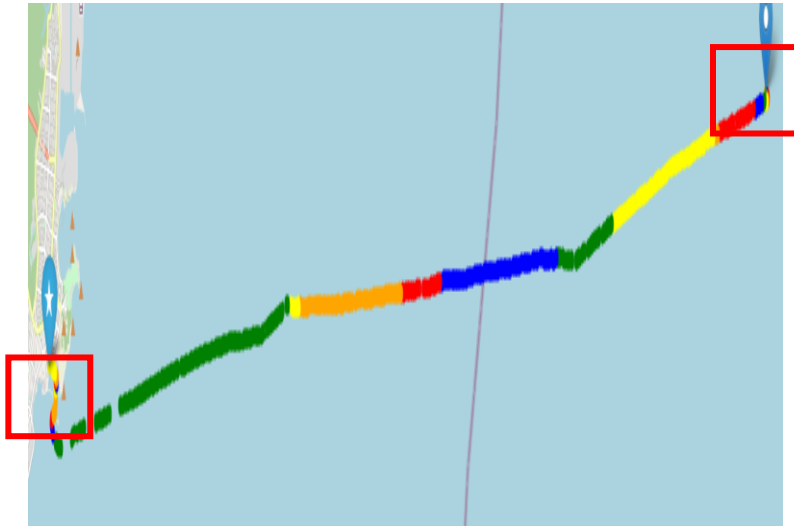
직선에 가까운 일반적인 동선이라면
색이 같은 점이 많을 것이라고 판단,

일반적이지 않은 동선이라면 일부 공간 내에
많은 색이 있을 것이라고 생각









	szMsgSendDT	SHIP_CODE	dSOG	dCOG	dLat	dLon	dist	dCOG_diff
8649	2022-06-28-11-07-31	AB110b60	4.5	102.0	36.169014	126.270401	23.973732	NaN
17432	2022-06-28-11-08-07	AB110b60	4.5	81.0	36.169037	126.271317	21.234785	21.0
19995	2022-06-28-11-08-20	AB110b60	4.9	95.0	36.169056	126.271667	31.574699	14.0
28173	2022-06-28-11-08-52	AB110b60	5.2	83.0	36.169113	126.272499	35.773811	12.0
31504	2022-06-28-11-10-56	AB110b60	4.9	76.0	36.169277	126.275650	283.432595	7.0
...	...	...	...	...	...	...	...	...
709157	2022-06-29-11-57-49	AB110b60	4.0	237.0	36.146439	126.253632	29.374765	17.0
712301	2022-06-29-11-58-02	AB110b60	4.0	241.0	36.146317	126.253380	26.368735	4.0
716594	2022-06-29-11-58-21	AB110b60	3.8	228.0	36.146114	126.253067	29.047385	13.0
725432	2022-06-29-11-59-03	AB110b60	3.5	224.0	36.145649	126.252296	38.641352	4.0
734063	2022-06-29-11-59-42	AB110b60	3.7	230.0	36.145222	126.251633	47.710484	6.0

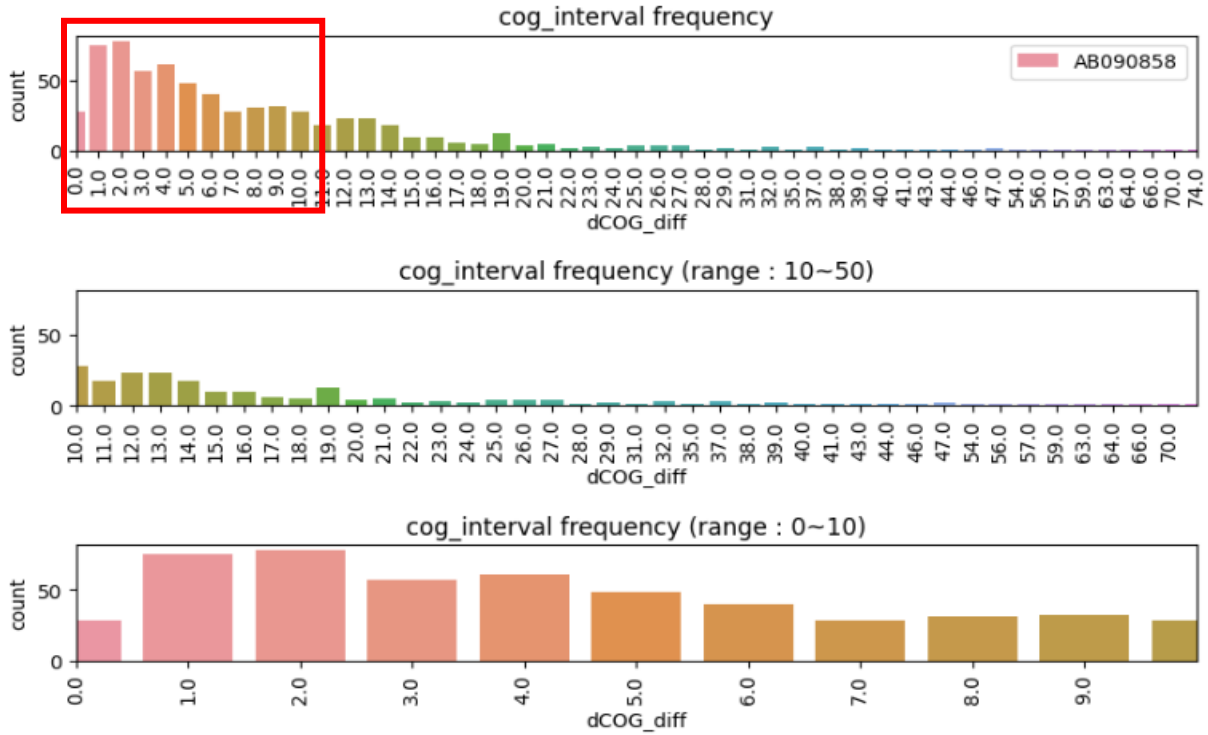
1796 rows × 8 columns

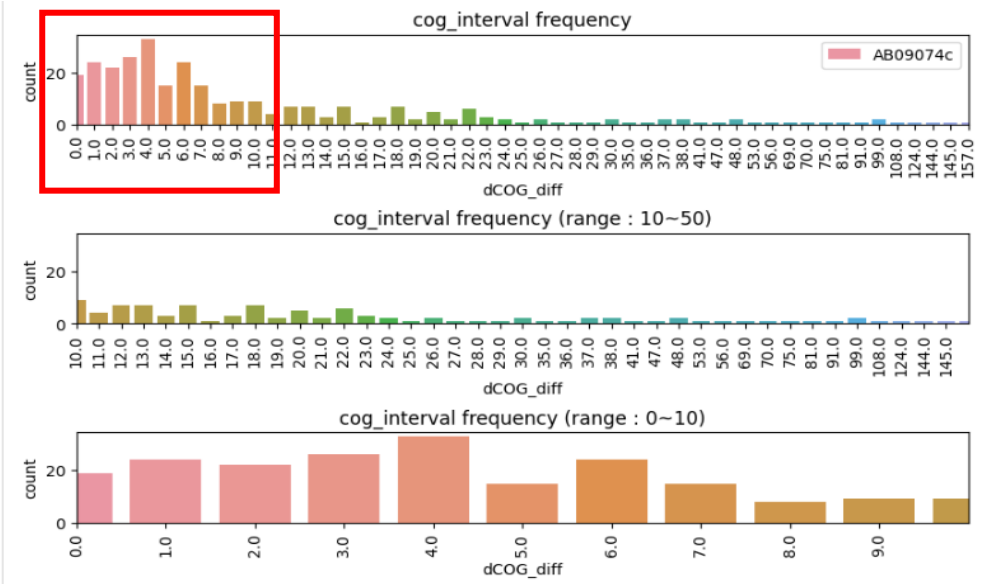
데이터 전처리 3

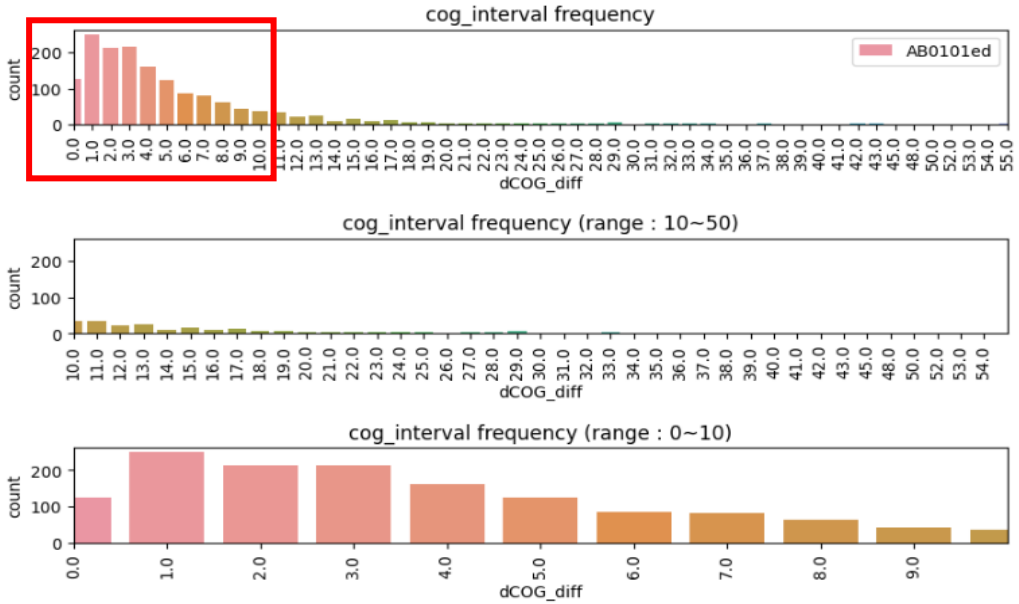
Why COG > 20 ?

대부분의 선박에서
cog_interval의 수치가 0~10에 몰려있음을 확인
보통 cog_interval 의 값이 10 이상이거나 20 이상 일 때,
해당 수치에서의 빈도수가 크게 줄어드는 것을 확인

COG= 10 을 기준 값으로 정할 시, 동선의 색 변화가 민감하게 반응할 것이라 판단, COG = 20 을 기준 값으로 정함



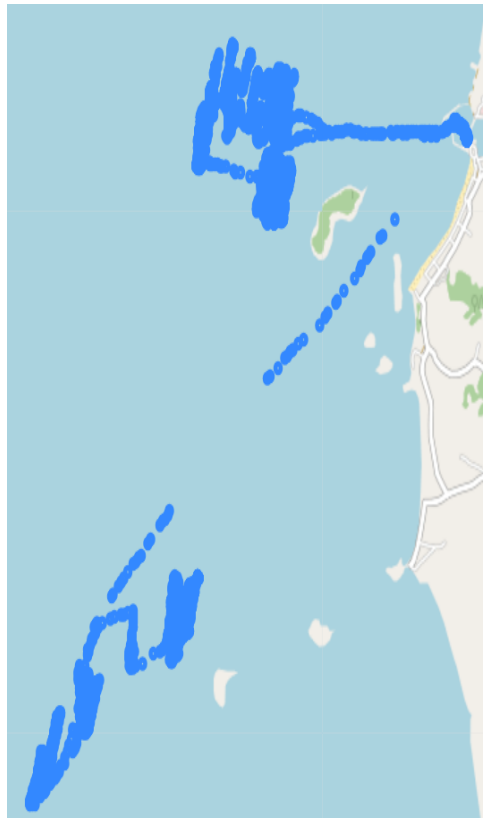




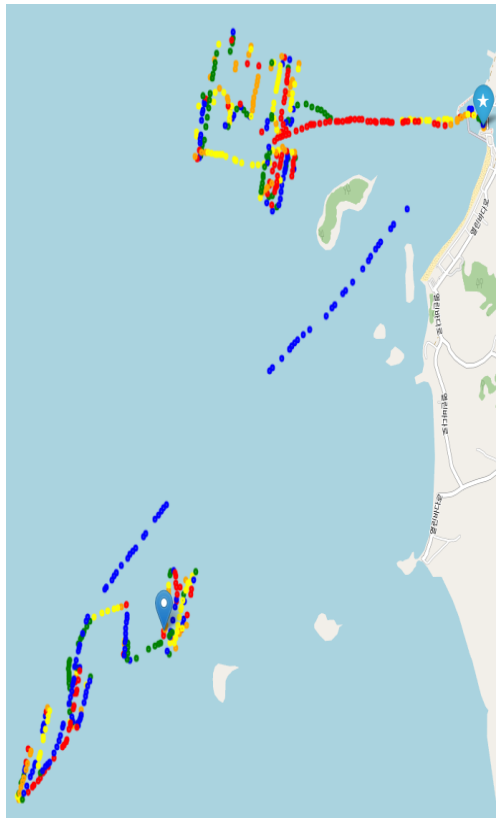
데이터 전처리 3

[결과 1]

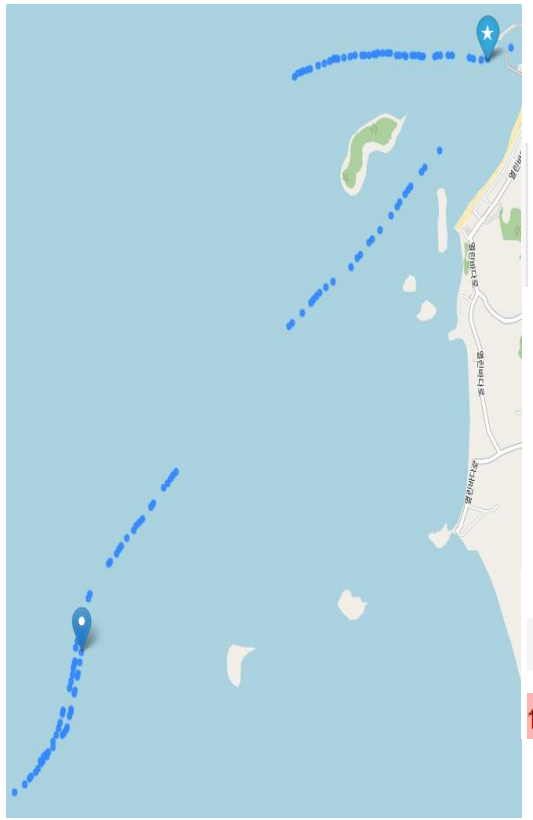
선박 : AB08064b



<원본>



<전처리 1+2>



<전처리3>

735218	20220629115947000	AB08064b	0.0	333.0	36.249516	126.536850	
738052	20220629115959000	AB08064b	0.0	333.0	36.249516	126.536850	

12191 rows × 6 columns

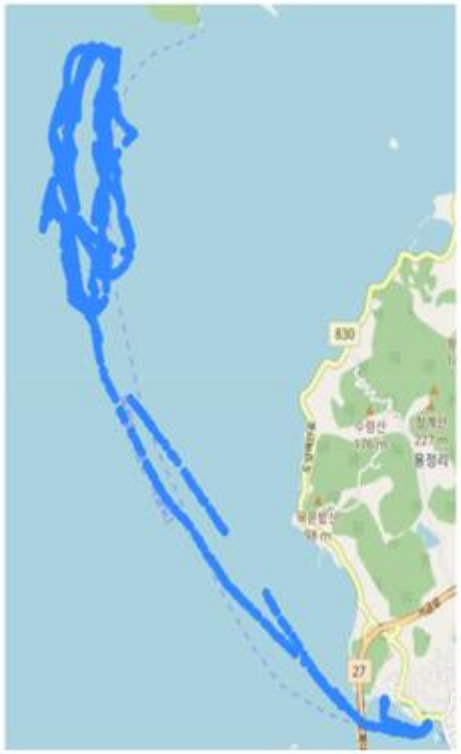
851	2022-06-29-09-13-05	AB08064b	13.8	95.0	36.249329	126.532448	76.243736	5.0
852	2022-06-29-09-13-14	AB08064b	13.8	73.0	36.249374	126.533180	65.875718	22.0

154 rows × 8 columns

데이터 전처리 3

[결과 2]

선박 : AB0100bb



<원본>

<전처리 1+2>

<전처리 3>

736447	20220629115952000	AB0100bb	0.0	290.0	34.526737	127.132584	
737640	20220629115958000	AB0100bb	0.0	290.0	34.526737	127.132584	

12186 rows × 6 columns

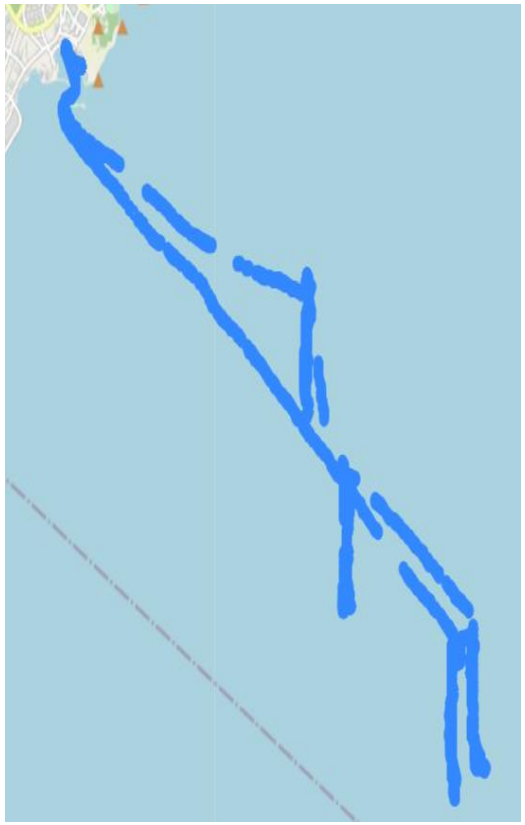
1192	2022-06-29-07-52-30	AB0100bb	10.9	136.0	34.528580	127.119919	16.799694
1193	2022-06-29-07-54-52	AB0100bb	9.8	95.0	34.526146	127.127419	738.415523

288 rows × 8 columns

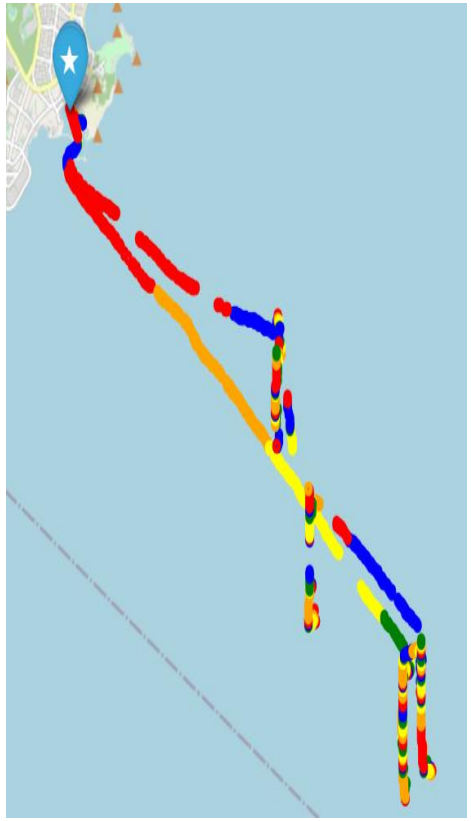
데이터 전처리 3

[결과 3]

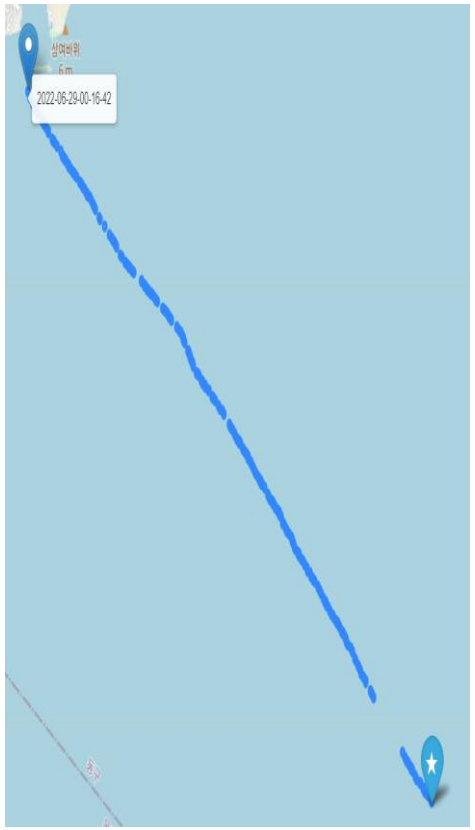
선박 : AB0805d3



<원본>



<전처리 1+2>



<전처리3>

734911	20220629115945000	AB0805d3	0.0	6.0	35.484928	129.427032	
736572	20220629115952000	AB0805d3	0.0	6.0	35.484928	129.427032	

10772 rows × 6 columns

342	2022-06-29-00-52-38	AB0805d3	9.6	115.0	35.418381	129.514404	32.807439	6.0
343	2022-06-29-00-52-50	AB0805d3	9.4	142.0	35.418026	129.514923	61.370463	27.0

282 rows × 8 columns

데이터 전처리 3

[결과 4]

선박 : AB08064f

<원본>

<전처리 1+2>

<전처리3>

390925	20220629083745000	AB08064f	0.0	220.0	34.762962	127.946236
392241	20220629083751000	AB08064f	0.0	220.0	34.762962	127.946236

7103 rows × 6 columns

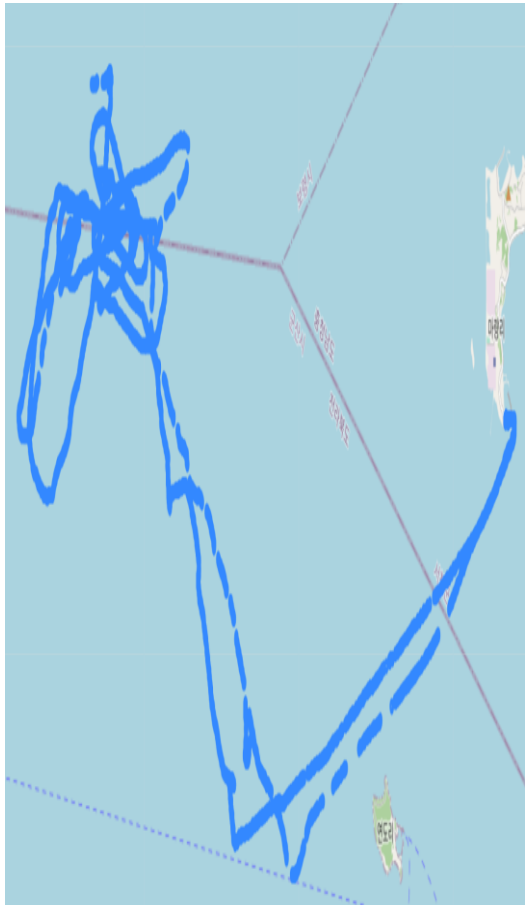
92	2022-06-28-16-51-14	AB08064f	9.2	17.0	34.746235	127.943581	33.890675	5.0
93	2022-06-28-16-51-26	AB08064f	9.4	4.0	34.746723	127.943787	43.442819	13.0
94	2022-06-28-16-51-52	AB08064f	9.4	347.0	34.747784	127.943550	61.743529	343.0

81 rows × 8 columns

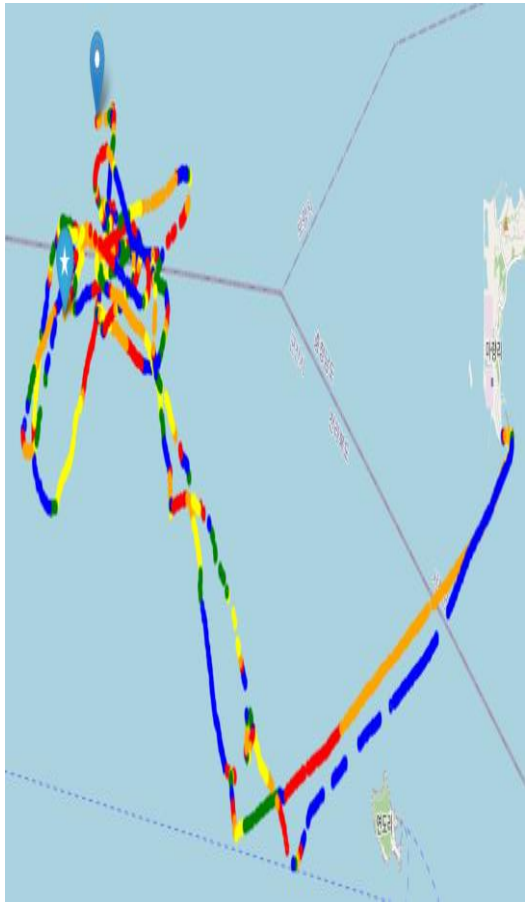
데이터 전처리 3

[결과 5]

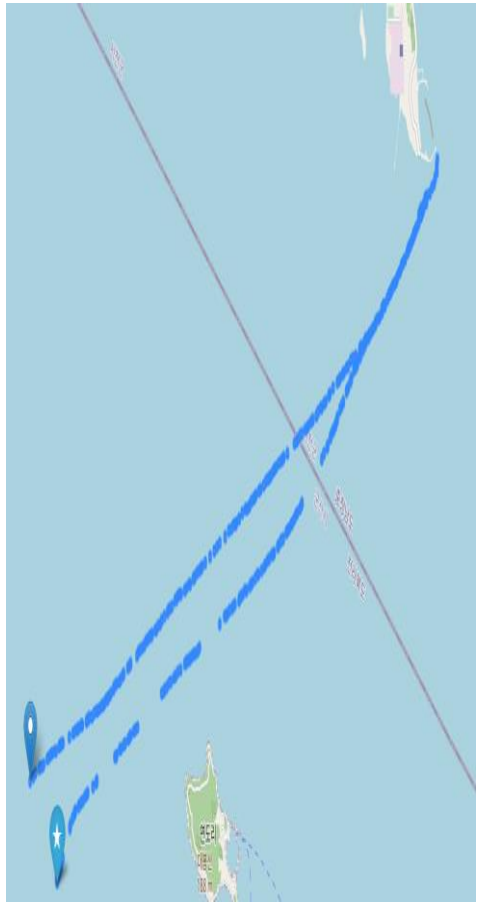
선박 : AB110b60



<원본>



<전처리 1+2>



<전처리3>

735445	20220629115948000	AB110b60	3.8	229.0	36.145153	126.251534	
737136	20220629115955000	AB110b60	3.7	226.0	36.145073	126.251419	

12242 rows × 6 columns

2540	2022-06-29-06-00-43	AB110b60	10.2	221.0	36.078522	126.388168	30.213225	7.0
2541	2022-06-29-06-00-55	AB110b60	7.1	241.0	36.078243	126.387703	52.039498	20.0

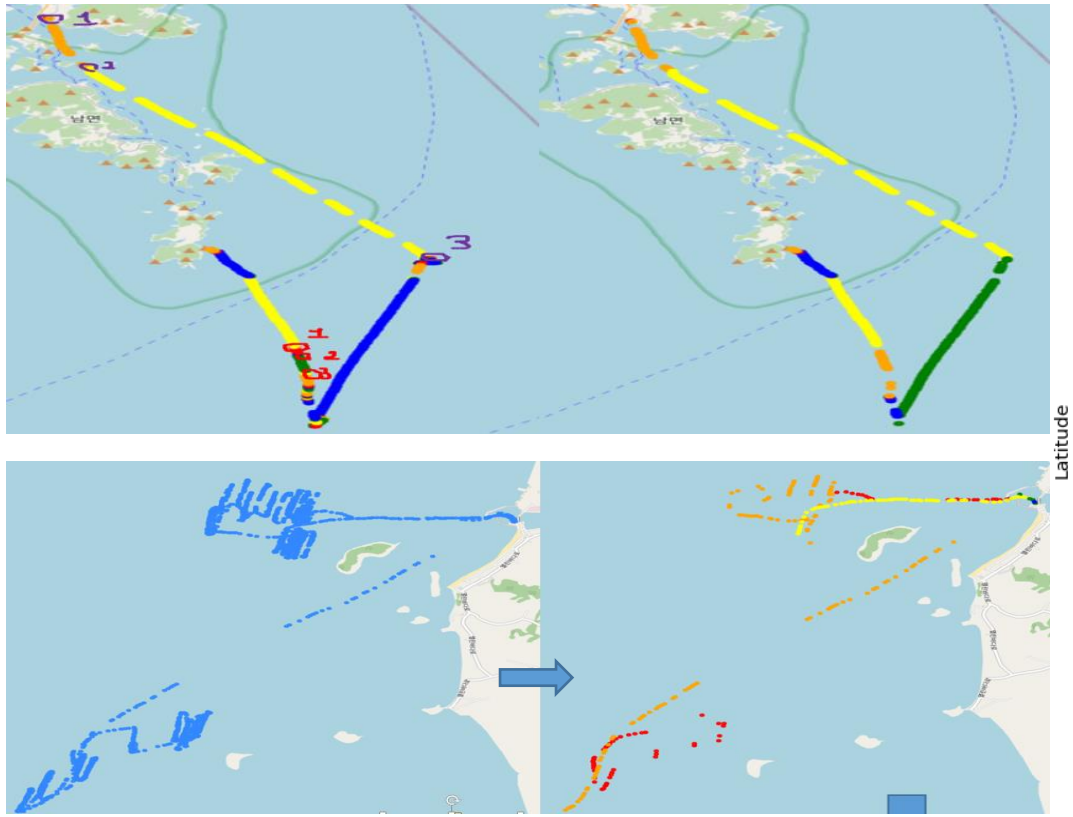
700 rows × 8 columns

비정상인 동선들은 많이 제거가 됐지만,
직선에 가까운 3개의 동선만 추출 하다 보니
정상적인 동선들도 꽤 많이 전처리가 되어버린다.

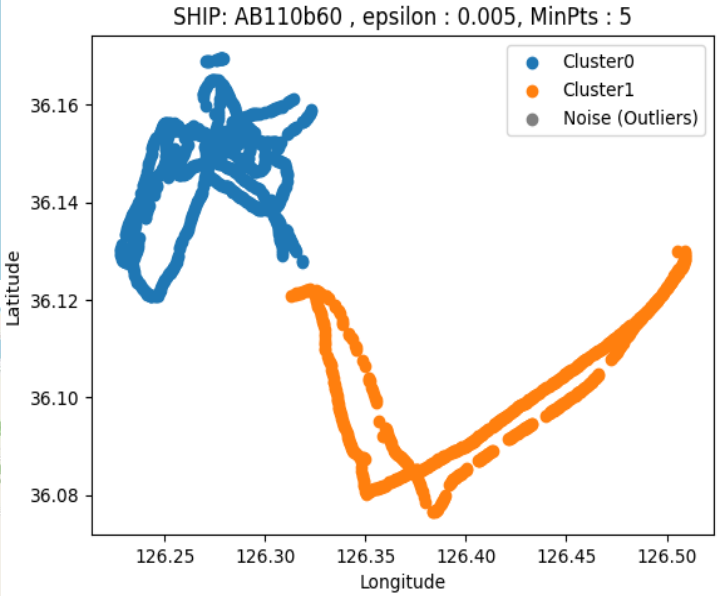
하지만, 정상적인 동선들의 손실을 감안 하더라도
양식 (고기잡이)와 같은 비정상 동선을 완벽히 제거하는 것에 더 큰 의의를 두었다.

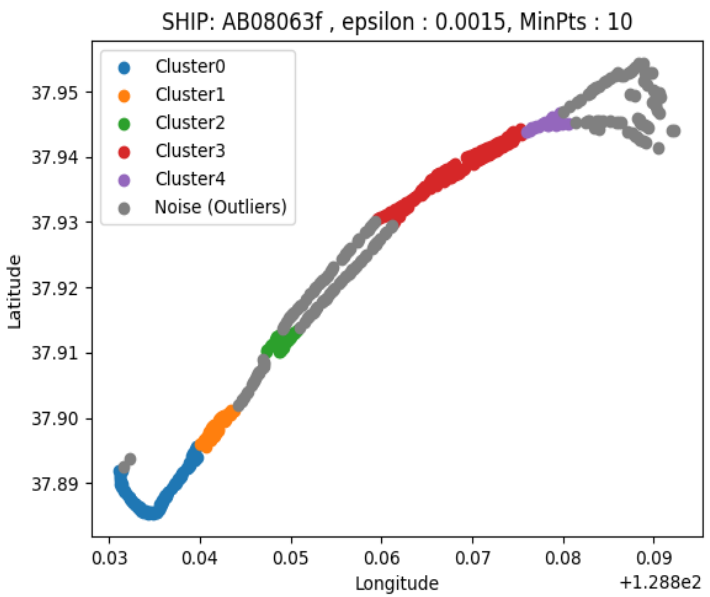
다른 전처리 기법들

[COG 변화량 + haversine]



[DBSCAN 동선 군집화]





최대한 정상적인 동선을 남겨보려 하기 위해 cog 변화 + haversine 을 이용하여 알고리즘을 구현했으나, 항상 위 사진과 같은 예외 사항이 몇 개 존재 했다.

DBSCAN으로 동선을 군집화 하여서
가장 큰 군집을 제거하거나, noise 로 분류된 데이터들을 제거하는 방향도 구상해보았지만
각 선박의 데이터셋 마다 parameter 들은 모두 달랐고 이를 일반화 할 수 없다고 판단 하였다.

결론

[24시간치 모든 A 선박]
16020592 row -> 365590 row
약 97.7 % 데이터 전처리

	szMsgSendDT	SHIP_CODE	dSOG	dCOG	dLat	dLon
0	20220628110654000	BE010da2	6.331898	149.0	36.965206	126.828316
1	20220628110654000	AB110b5d	10.100000	272.0	37.874233	129.008636
2	20220628110654000	AB110b8e	0.000000	227.0	38.499546	128.425110
3	20220628110654000	AB0908a0	0.000000	0.0	34.556713	127.675613
4	20220628110654000	BD010d54	0.000000	275.0	35.978321	126.622986
...	...	...	...	...	...	...
738051	20220629115959000	AB02044d	9.479563	291.0	36.348801	129.429718
738052	20220629115959000	AB08064b	0.000000	333.0	36.249516	126.536850
738053	20220629115959000	BE020e3f	7.910589	173.0	35.364128	125.724831
738054	20220629115959000	AB010107	0.000000	256.0	36.675259	126.128418
738055	20220629115959000	AB0203fd	6.647376	60.0	36.263489	129.402298

16020592 rows × 6 columns

	szMsgSendDT	SHIP_CODE	dSOG	dCOG	dLat	dLon	dist	dCOG_diff
722	2022-06-29-04-27-27	AB110b5d	9.900000	41.0	37.918968	128.953674	55.238613	24.0
723	2022-06-29-04-27-30	AB110b5d	10.600000	34.0	37.919079	128.953796	16.308807	7.0
724	2022-06-29-04-27-42	AB110b5d	10.900000	40.0	37.919556	128.954269	67.327630	6.0
725	2022-06-29-04-28-09	AB110b5d	10.500000	52.0	37.920368	128.955643	150.580022	12.0
726	2022-06-29-04-28-12	AB110b5d	10.700000	55.0	37.920456	128.955795	16.562920	3.0
...	...	...	...	...	...	...	...	...
230	2022-06-29-11-52-37	AB090842	10.400000	70.0	35.977085	126.527885	15.938311	5.0
231	2022-06-29-11-52-40	AB090842	10.100000	71.0	35.977135	126.528053	16.078794	1.0
232	2022-06-29-11-52-49	AB090842	10.200000	93.0	35.977180	126.528580	47.643206	22.0
1	2022-06-29-11-48-37	AB09094c	3.554099	20.0	34.469666	127.294792	11.661909	70.0
2	2022-06-29-11-48-51	AB09094c	4.032332	56.0	34.469799	127.295036	26.857239	36.0

365590 rows × 8 columns

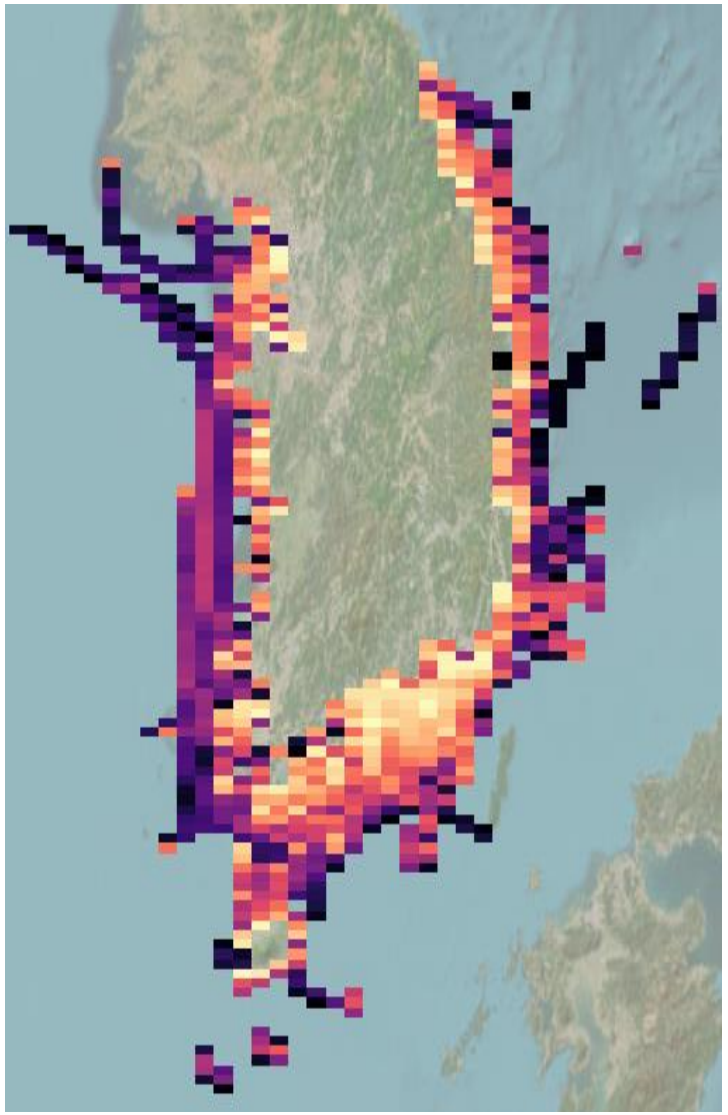
```
전처리 전과정 수행 시간 : 20.882212423952296 sec
--------------------------
upper_2_dataframe 함수 실행 시간 : 0.10616427403874695 sec
add_distance_df 함수 실행 시간 : 0.8612683080136776 sec
dist_upper_10_df 함수 실행 시간 : 0.07999231992289424 sec
reset_time_series 함수 실행 시간 : 4.2651349569205195 sec
add_cog_interval 함수 실행 시간 : 0.11661809799261391 sec
similar_straight_cog 함수 실행 시간 : 15.453034467063844 sec
```

가설 설정한 아이디어를 데이터로 확인 해보며, 이러한 가설을 알고리즘으로 개발
DEMO로 24시간치 데이터로 전처리 모듈을 개발 하였음

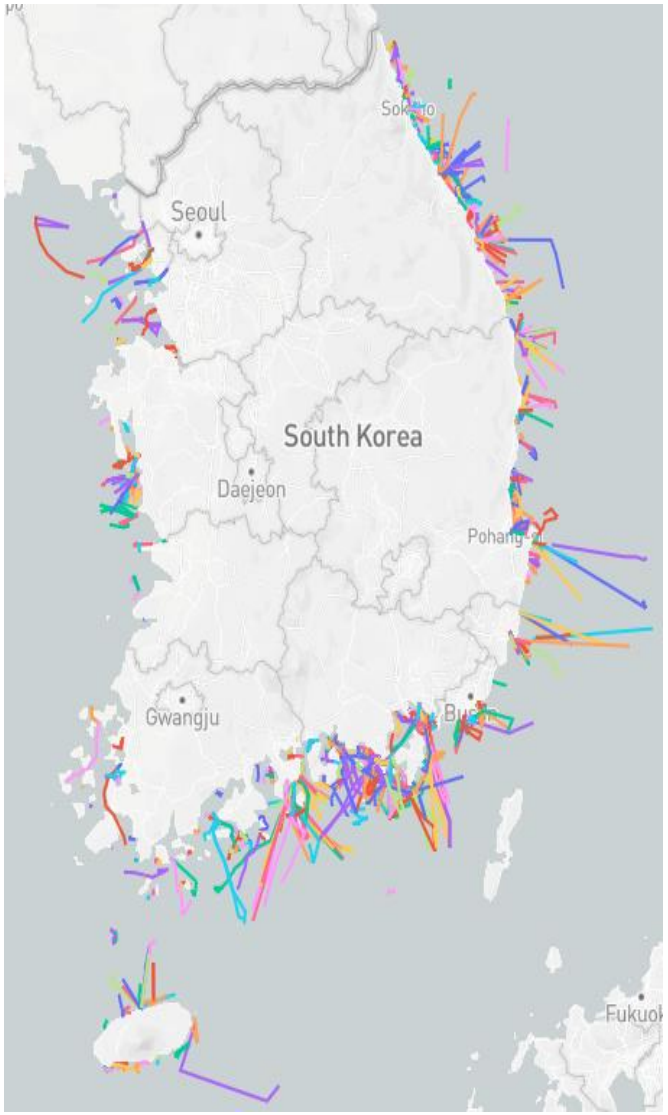
또한 모든 알고리즘을 파이프라인 구조로 설계 하였으며 해당 전처리 알고리즘을 메서드로 만들어 클래스를 만들고 이를 모듈화 하였음

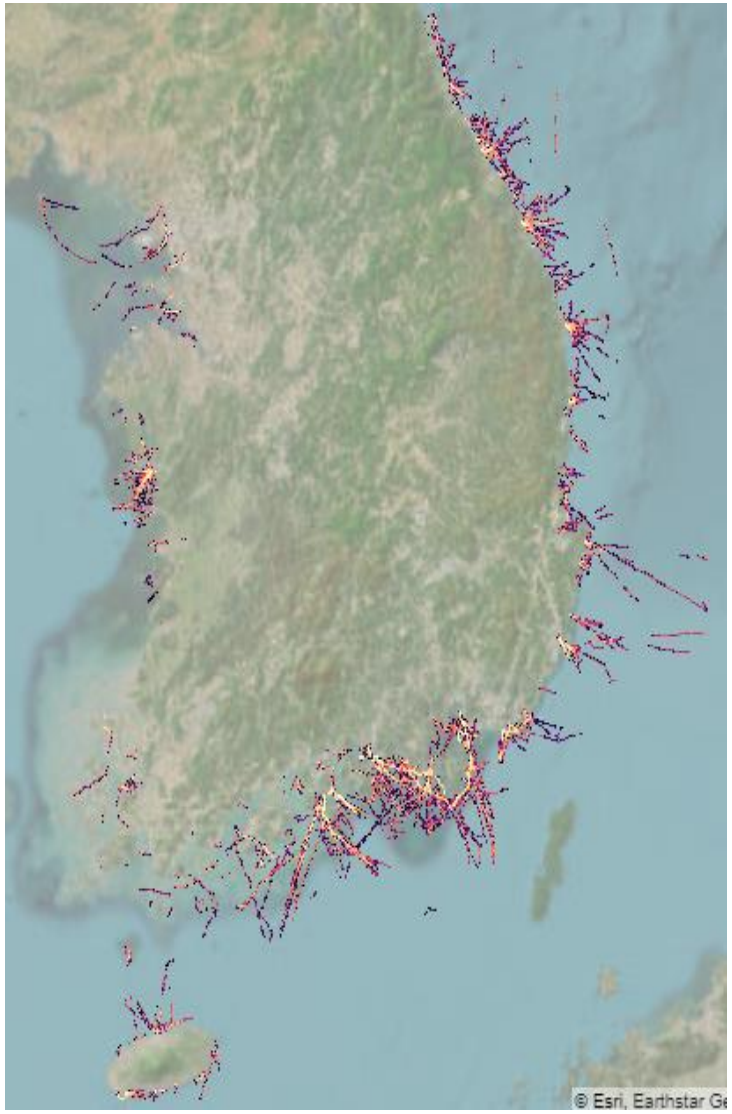
전처리 전/후 항적 시각화

Before



After



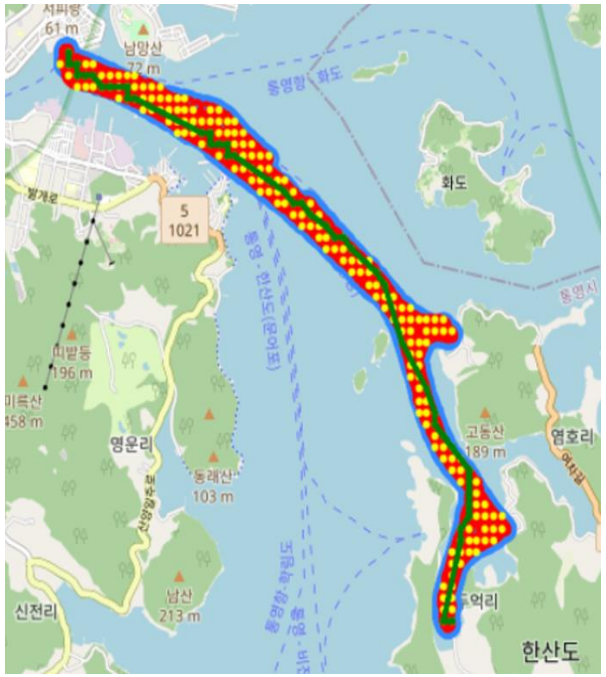


최적 안전 항로 추출 기법 개발

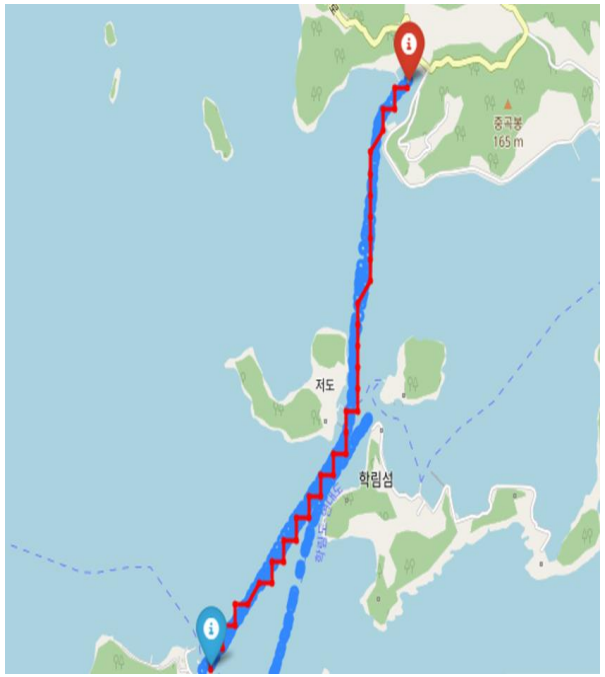
전처리 된 항적을 추렸으므로, 출발지와 목적지 사이에 비정상 동선이 filtering 되고 남은 정상 동선만 존재.

바다를 100x100 격자 구조로 쪼개어 위 경도 좌표를 parameter 로 넣으면 격자 번호로 반환 하는 클래스 개발

이렇게, 출발지와 도착지 사이 격자 번호를 가지는 항적을 최종적으로 return 하는 방향으로 최적 안전 항로 추출 기법 제안 중



초록선 : 최적 안전 항로



빨간선 : 최적 안전 항로

한국 지도 격자 구조