EOSC 584 Project
Fraser River Plume Visualization with Remote Sensing Data

Shumin Li
M.Sc. in Oceanography
shuminli@eoas.ubc.ca

A study on the plume shape, area and variation in the Strait of Georgia

March 30, 2021

Contents

1 Introduction

This project aims to visualize the structure, shape and variation of Fraser River Plume using public available remote sensing data, which include optical frequency channels of Sentinel-2 Level-2A Product, derived SPM (suspended particulate matters) images from MODIS Auqa/Terra missions, and HF (high-frequency) radar derived surface currents in the strait of Georgia. Relationship between the plume area with wind, tide and river discharge are explored, and the intrinsic bias of SPM is demonstrated and discussed. All codes, figures together with this report are kept in Github for peer review at https://github.com/shumin-li/EOSC-584-Project.

Figures and analysis ideas in this project are inspired by Pawlowicz et al. (2017), in which seasonality and wind/river influence on the Fraser River plume area/patterns are analyzed using MODIS derived SPM data from 2003 to 2015. Some differences between this project and Pawlowicz et al. (2017) are as follows: this project 1) added the analysis of plume area and patterns over a tidal cycle, 2) looked at the surface circulation patterns for the same analysis conditions we gave for SPM images, 3) bias of the SPM images are given a closer attention, 4) we used a different standard to pick up "good days" for SPM images, which allow more images for analysis, but may also included some "bad days", 5) analysis period extended from 13 years to 17 years, 6) Sentinel-2 image was used in this study as a demonstration example, 7) added boxchart representations of plume area variation in a yearly cycle and tidal cycle, 8) projections, grid boxes, legend designs and some other rendering setups are different.

2 Data

2.1 Remote Sensing Data

2.1.1 Sentinel-2 Level-2A Product

This product is described as Bottom-Of-Atmosphere reflectance in cartographic geometry, and it's data volume for a single shot is about 800 MB (at most 10m resolution for 100x100 km^2 area). The data and figures can be downloaded at (https://scihub.copernicus.eu/dhus/#/home). To use the data in Matlab, firstly, we need to use a software called SNAP to export the raw data into a .nc file. Then, we can import the .nc file into Matlab, extract the data from the bands with optical frequencies (red, green and blue), clean the data and finally plot it onto a mapping grid (codes in Sec. 5.3 and figure in Figure. 1).

One tricky part for plotting this Sentinel-2 image is that, it does not have a regular grid, which is required by m_image functions. There are several way to work around it, and in this project, we used a simple way to pick up a row of longitude and a column of latitude at the center of plume, and use those to make a "regular grid" for it. Since we mainly focus on the plume region, this way of plotting gives a very accurate representation in the plume center, and the mis-display at the corners of the image is almost invisible. The second way is to plot it in black-white color using m_pcolor. The third and most accurate way is to plot all the data into UTM projection, since the Sentinel-2 grid is mapped in UTM projection.

2.1.2 MODIS SPM Image

MODIS L1A images can be downloaded form the NASA Ocean Color website (https://oceancolor.gsfc.nasa.gov/). Data from both Aqua and Terra missions from year 2003 to 2019 are used in this project. Earlier studies (Pawlowicz et al., 2017) found a semi-empirical relationship between Suspended Particulate Matters (SPM) concentration and MODIS high-resolution 645 nm channel as:

$$S_W = \frac{A^\rho \rho_w}{1 - \rho_w / C^\rho} + B^\rho \tag{1}$$

where the parameters A^ρ = 258.85, B^ρ = 0, and C^ρ = 0.1641. ρ_w is the subsurface reflectance, and S_W is the SPM concentration we will use in this project. The data processing from MODIS images to each individual .mat file of SPM data was done by Research Assistant Katia Stankov, and I was only using the derived SPM data files for this project.

There are about 11, 259 individual SPM images from year 2003 to 2019, and most of them are either empty or taken in cloudy days. I set up a standard that automatically picks up 2, 731 good images (Sec. 6.1) and carried out the following analyses in the project. However, by the end of the day, reviewing all individual images and picking up all good images manually will give us a better result.

2.1.3 HF Radar (CODAR) Surface Currents

The Ocean Networks Canada (ONC) (https://data.oceannetworks.ca/home) Victoria Network Under the Sea (VENUS) ocean observatory maintains four CODAR Ocean Sensors Inc. SeaSonde units in the southern Strait of Georgia at the Westshore coal port (deployed in Dec 2011), Iona outfall (deployed Aug 2012), Georgina Point (deployed May 2016), and Point Atkinson (deployed January 2017). These direction-finding HF radar systems are sensitive to currents in the upper 50 cm of the water column, and they operate at a nominal frequency of 25 MHz. All stations produce hourly radial current maps by combining seven 10-minute averages that are gridded to range bins of 0.5 km and bearing bins of 5°. For this project, I was using the CODAR surface currents data struct processed by Research Assistant Katia Stankov, and carrying out the further analysis.

2.2 Other Data

Fraser River discharge data is obtained from the gauging station at Hope, which is about 30 m above sea level and around 120 km inland. These data can be acquired from Water Survey of Canada (https://wateroffice.ec.gc.ca/report/real_time_e.html?stn=08MF005).

Wind data was got from Sand Heads lighthouse station, Environment Canada Climate ID # 1107010, which is positioned at the end of the jetty extending from Steveston to the edge of the mudflats. Two-minute averaged hourly wind direction and speed data at 11 m can be downloaded from Environment Canada (https://climate.weather.gc.ca/historical_data/search_historic_data_e.html).

3 Brief Results and Discussion

From Satellite images, Fraser River Plume can be easily captured and visualized, especially during summer times when the river discharge is high. Three distinct dynamical regions can be observed as: a highly-turbid near-field jet, a recirculating bulge, and a far-field tail. Plume fronts can move at a speed of about 3 km/h as tide flooding in and out (see yellow and red curves in Figure. 1). For the locations with a sharp front boundary, HF radar derived surface currents are usually found as convergence or sheer.

3.1 Seasonality Analysis

Seasonal variations are found to be obvious for the plume shape and area. It has a great extent in summer months (especially in May and June), while it is mostly trapped onto the shallow mudflat region in winter months (Figure. 2). For the same time when SPM images are taken, mean surface currents are showing the propagation and dynamics of the river plume. Currents are pointing out from the river mouth into the Strait, and some convergence features can be found at the plume boundary (see April and July in Figure. 3). However, some radar current plots seem to be chaotic, that is because the number of the data used here is limited to the same time when SPM images are taken. For the figures of all available CODAR data, see https://github.com/shumin-li/EOSC-584-Project/blob/main/figures/codar_by_month.png.

Fraser River has a high discharge from May to August and low discharge in winter months (Figure. 5). Plume area is also bigger in summer and relatively smaller in winter (Figure. 4). However, the area in November, January and February seems to show a second peak. There are a few reasons to cause this difference: 1): our method of selecting "good days" of SPM image did not work well enough in winter, which may accidentally count the clouds as plume . 2): The river flow data was from the gauge station at Hope, which can not capture high river discharge events caused by massive precipitations during winter. 3), in winter, as the strong salt wedge mixing and water entrainment inside the Fraser River mouth, density difference between the plume water and ambient ocean water is much smaller, which may result in different propagation speed and dynamics in winter.

For the wind records in this 17-year period, southeasterly winds are more often observed than northwesterly winds, especially in winter (Figure. 6). However, for the time when MODIS images are taken, the number for northwesterly winds is bigger than that of southeasterly winds in every month (Figure. 7). This means that the SPM images have a strong bias towards the preference of northwesterly wind, most likely because northwesterly winds usually brings sunny weather.

3.2 Wind and River Discharge Influence

To avoid the bias of SPM images and look at the wind and river influence of the plume separately, we can classify the images into different groups based wind directions and river flow conditions. Plume shape and structures are nicely shown in Figure. 8 and corresponding surface currents are shown in Figure. 9. To some degree, the plume area is linear correlated with the Fraser River Discharge (Figure. 10).

3.3 Tidal Influence

With the similar concept, we can average the SPM images into a tidal cycle. The patterns are indicating a strong plume pushing process during the big ebb of the day (see from h = -7 to h = 6 in Figure. 11), while a much weaker tidal variation for the small ebb of the day (h = 7 to h = 18 in Figure. 11). Unfortunately, this exciting finding is strongly biased as well. The SPM images are taken at about 13:00 p.m. of each day. In summer, the big ebb appears in the day time, and in winter the big ebb is always at night. This difference can be better demonstrated in a histogram (Figure. 13), in which images from h = -1 to h = 7 are all taken in springs and summers, while from h = 12 to h = -5, all images are taken in falls and winters.

For the plume area, we may see the pattern of bigger plume area during the big ebb (Figure. 14), and this conclusion may have big bias as well. Since the same boxchart of river flow for the time when SPM images are taken are also showing the peak at the big ebb of the day, so it is hard to say if tide has a influence on the plume area or not. However, that is not to say tide does not influence the plume shape at all, since a great number of cases showing the plume front propagation along the tidal currents (e.g. Figure. 1), just MODIS derived SPM data might not be the best to show this influence.

4 Figures

4.1 Comparison of Sentinel-2, MODIS SPM Images, and HF Radar Surface Currents

(Function codes see Sec. 5.3)

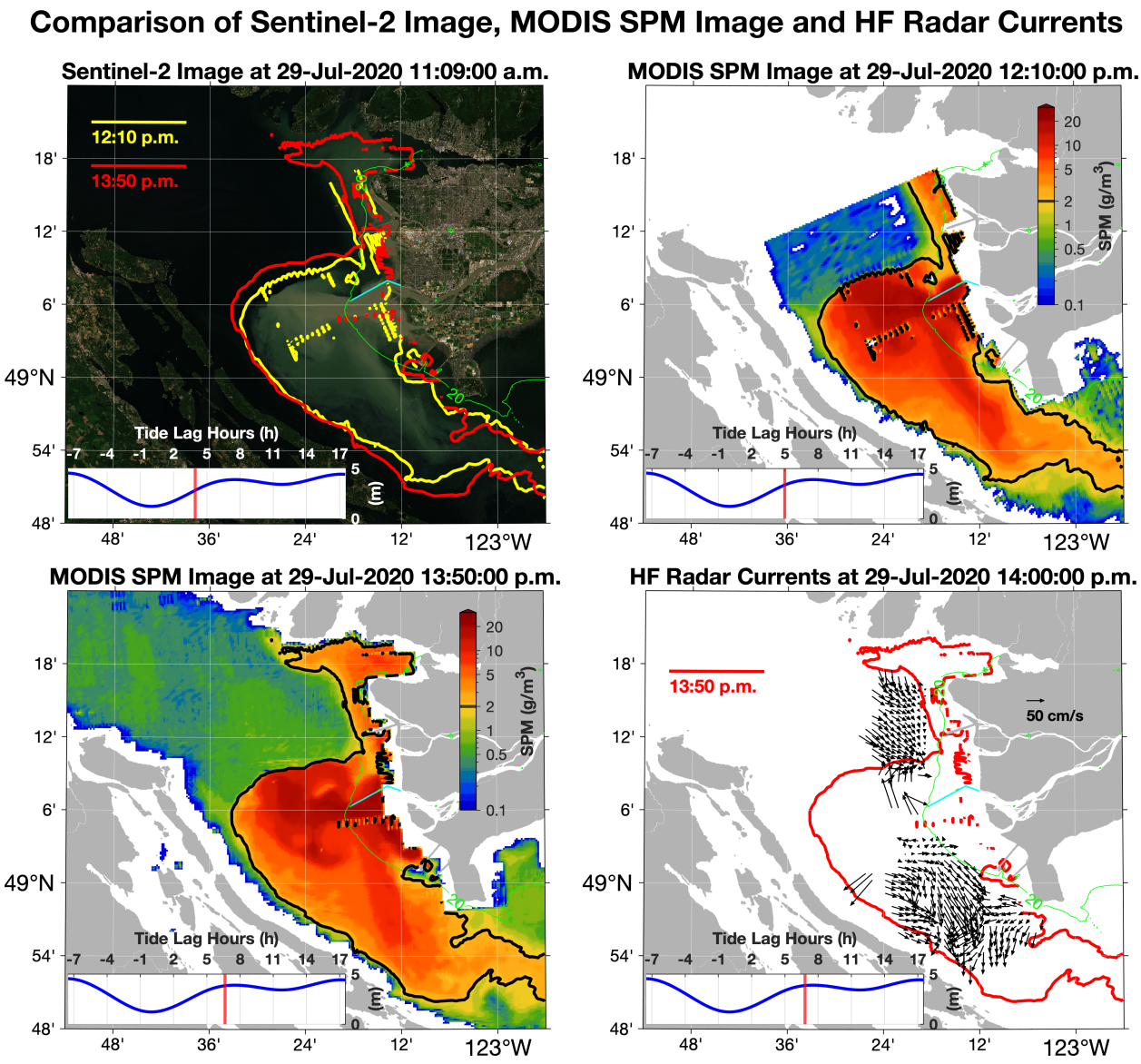

Figure 1: Comparison of plume structures from Sentinel-2 image (upper left) at 11:09 a.m., MODIS derived SPM images (upper right and lower left) at 12:10 p.m and 13:50 p.m., and HF radar surface currents at 14:00 p.m. (All taken on July 29th, 2020). Relative tide lag hours and tidal elevations are shown at the lower left corner of each subplot. Contours of SPM = 2 g/m^3, which indicates the boundary of the plume region, are plotted on both SPM maps, and also shown in Sentinel-2 image (yellow for 12:10 p.m. and red for 13:50 p.m.) and HF-radar currents map (red for 13:50 p.m.) Green contours are showing the isobath of 20 m and cyan color curve shows the channel of Fraser River outreach at Sand Heads.

4.2 Seasonality Analysis

4.2.1 Plume Patterns in Each Month

(Function codes see Sec. 5.4, with `'option'-> 'spm', 'type'-> 'season'`)

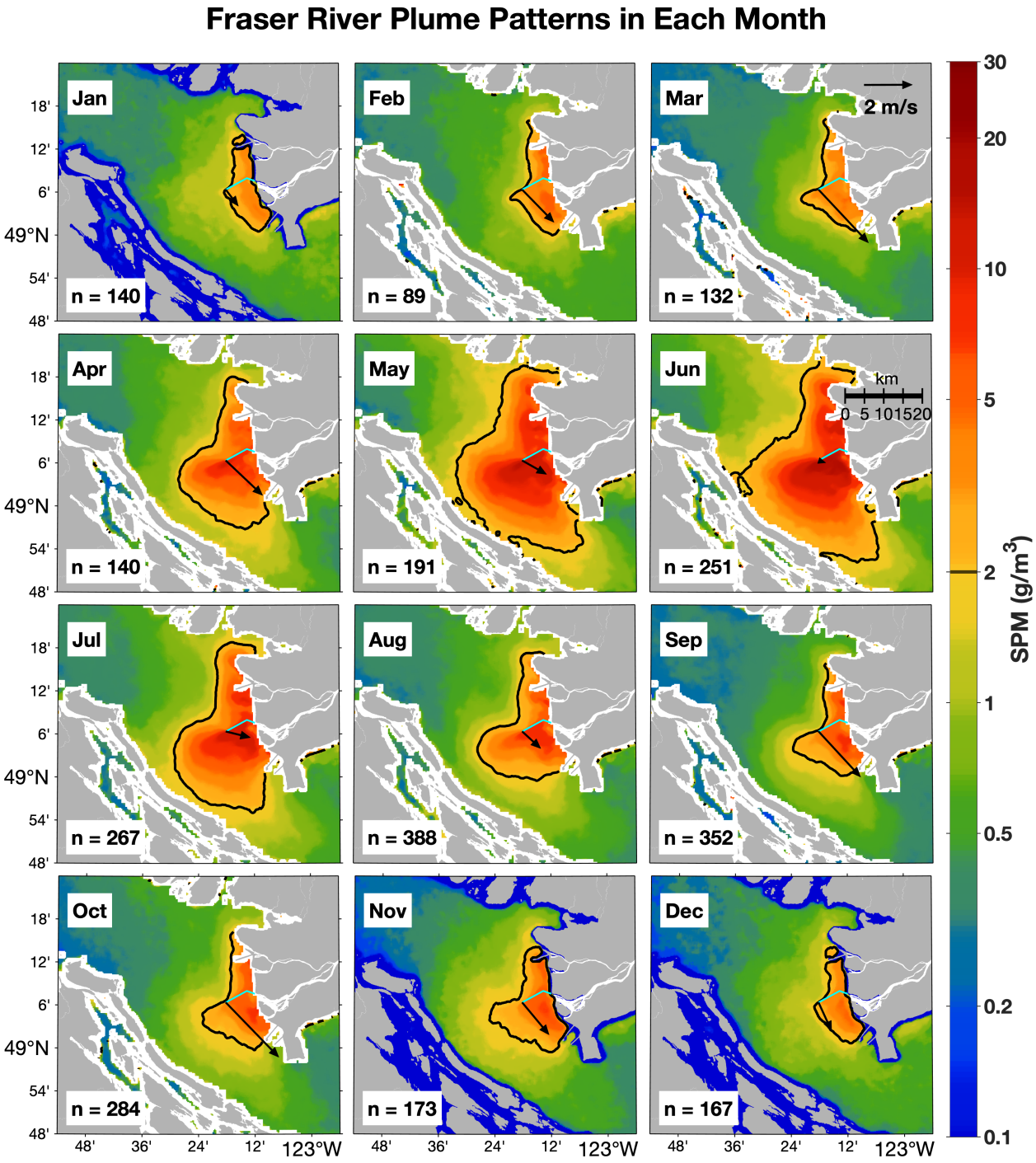

Figure 2: Fraser River Plume distribution patterns in each month. Black contours in the figures are showing SPM = 2 g/m^3, which indicates the boundary of the plume area. Number of binned images (n) is shown at the lower left corner of each subplots. Length scale is shown in the subplot of June. Mean wind vectors are added at Sand Heads station and wind length scales are added onto the subplot fro March.

4.2.2 Surface Currents in Each Month (t = SPM)

(Function codes see Sec. 5.4, with `'option'-> 'codar_at_spm', 'type'-> 'season'`)

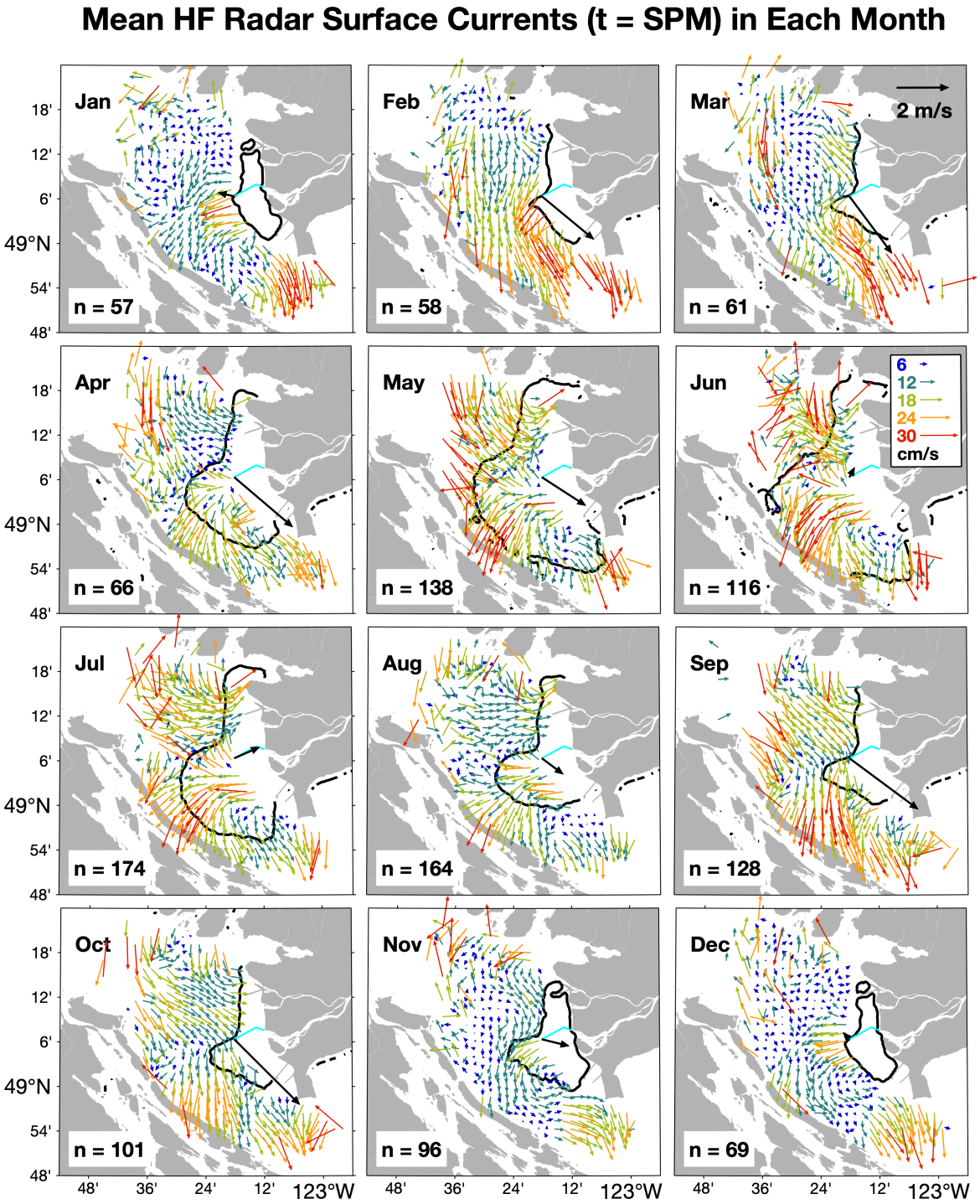

Figure 3: HF Radar derived surface currents in the Strait of Georgia in each month (for the time when MODIS images are taken). Black contours in the figures are showing SPM = 2 g/m^3 for each month (see Figure 2). Numbers of averaged currents (n) are shown at the lower left corner of each subplot. Arrow scales and corresponding colors are shown in the subplot of June. Wind vectors and their length scale are the same as Figure 2

4.2.3 Yearly Cycle of Plume Area

(Function codes see Sec. 5.5, with `'option'-> 'season'`)

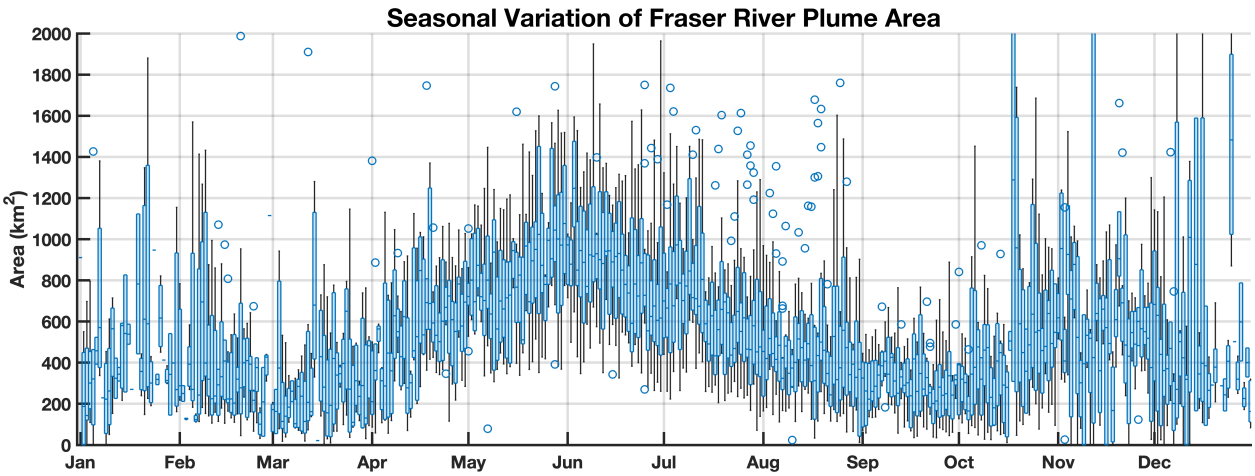

Figure 4: Boxchart of the Fraser River Plume area variation in an annual cycle. Boxes are showing the daily median and quartiles from 2003 to 2019.

4.2.4 Seasonality of Fraser River Discharge

(Function codes see Sec. 5.6)

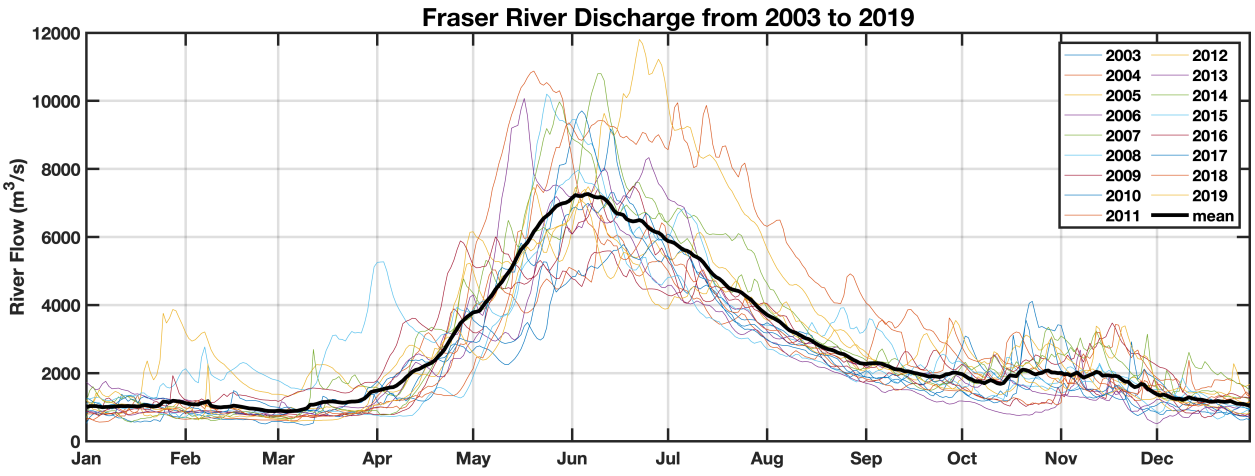

Figure 5: Fraser River discharge from 2003 to 2019. Mean flow of the 17 years are shown in thick black line

4.2.5 Seasonality of Wind at Sand Heads

(Function codes see Sec. 5.7)

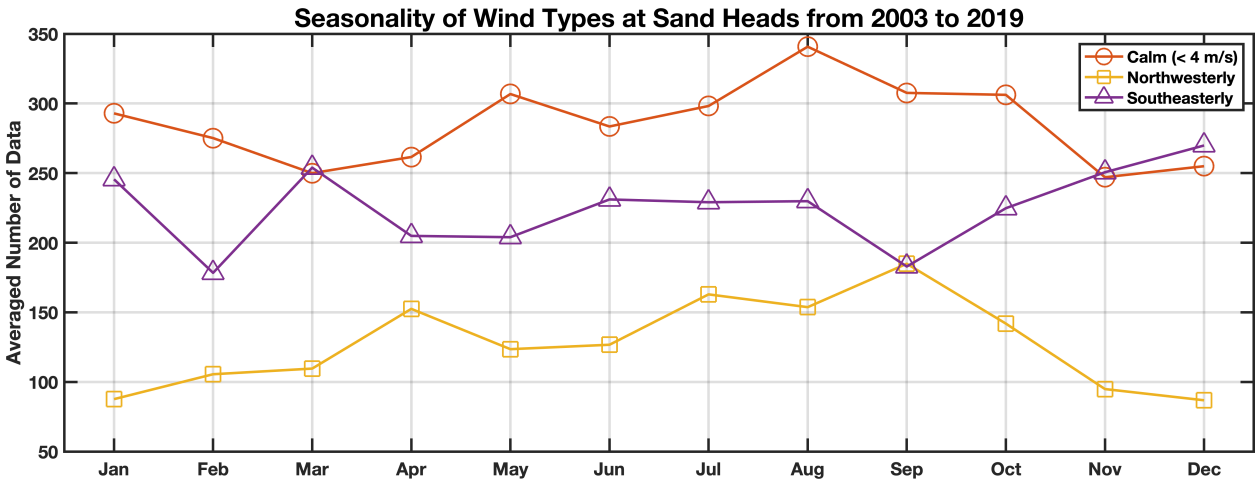

Figure 6: Averaged number of hours for each type of wind by month. (for example, among all the Januaries from 2003 to 2019, on average, there would be 293 hours recorded as calm winds ($<$ 4 m/s), 246 hours recorded as southeasterly winds and 88 hours recorded as northwesterly winds)

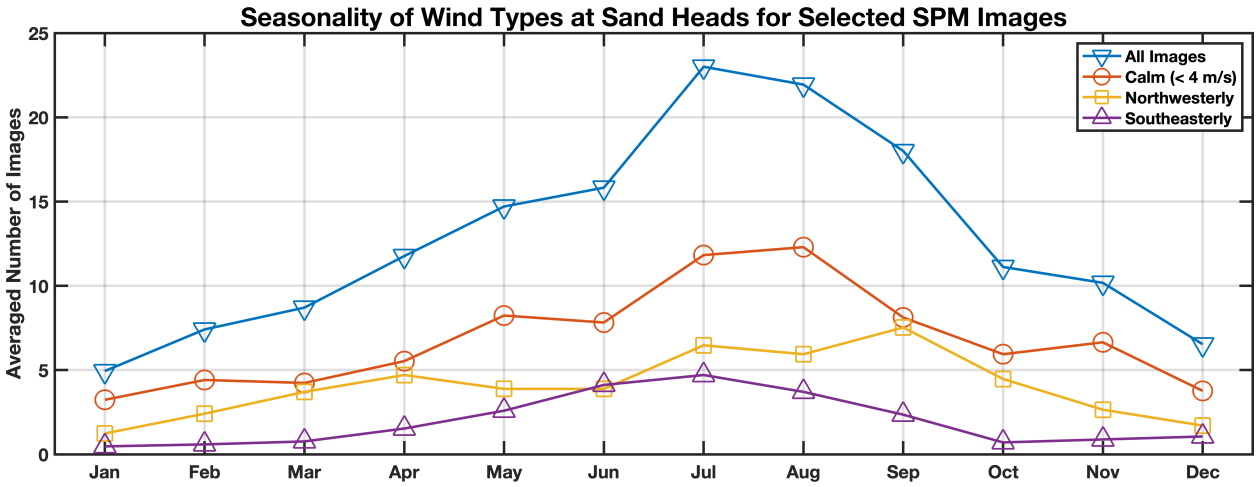

Figure 7: Averaged number SPM images for each type of wind by month. (Similar to Figure 6, but y-axis are showing the number of SPM images). This figure shows the bias of the SPM images: much less SPM images are taken under southeasterly wind conditions, which may also indicates that northwesterly wind are more likely to bring sunny weather.

4.3 Wind and River Influence on the Plume Shape and Area

4.3.1 Plume Patterns under Various Wind and River Flow Conditions

(Function codes see Sec. 5.4, with `'option'-> 'spm', 'type'-> 'wind_rf'`)

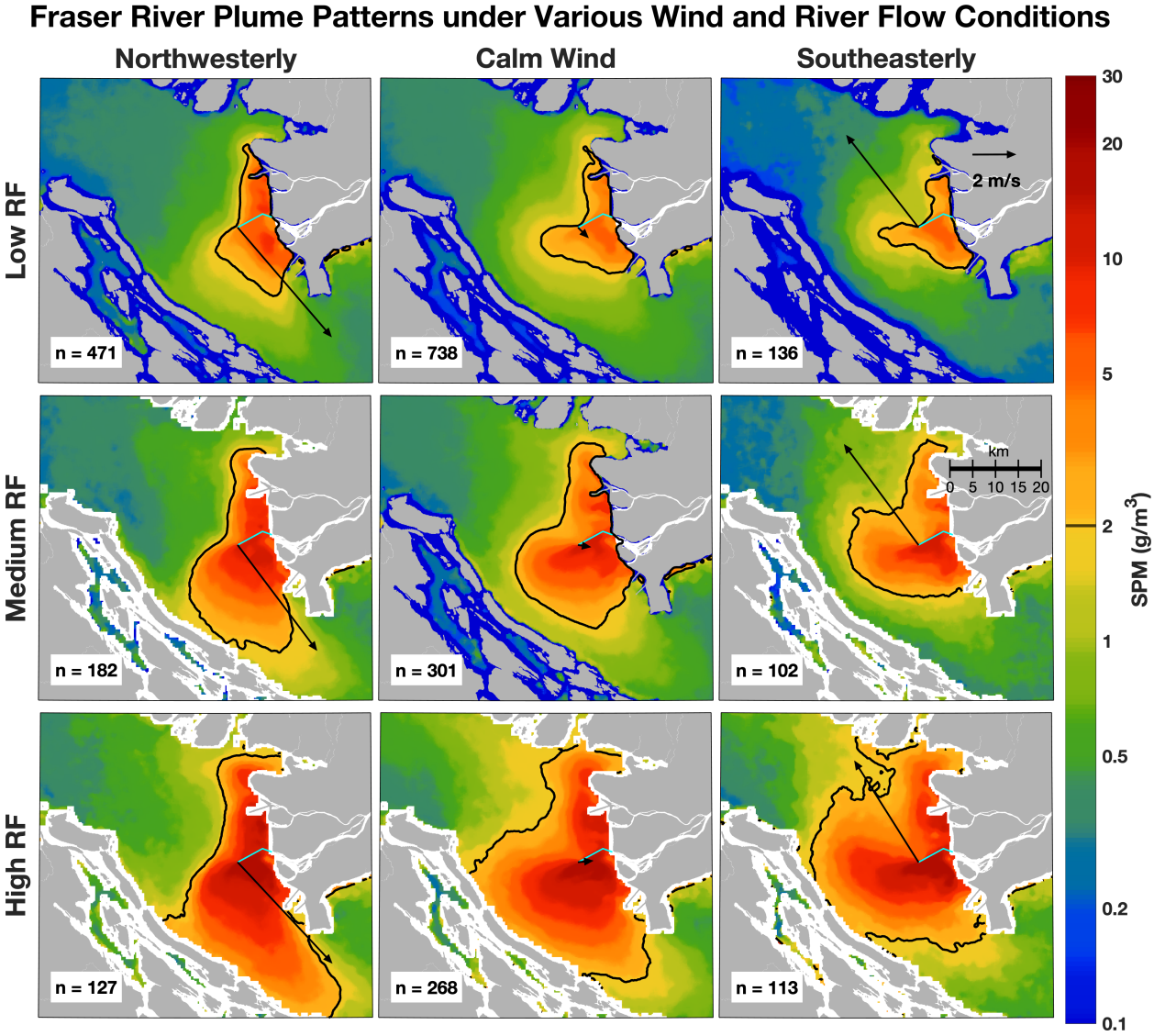

Figure 8: Fraser River Plume patterns under various wind and river flow conditions. (Colormaps, wind vectors and plume contours are similar to Figure 2)

4.3.2 Surface Currents under Various Wind and River Flow Conditions (t = SPM)

(Function codes see Sec. 5.4, with `'option'-> 'codar_at_spm', 'type'-> 'wind_rf'`)

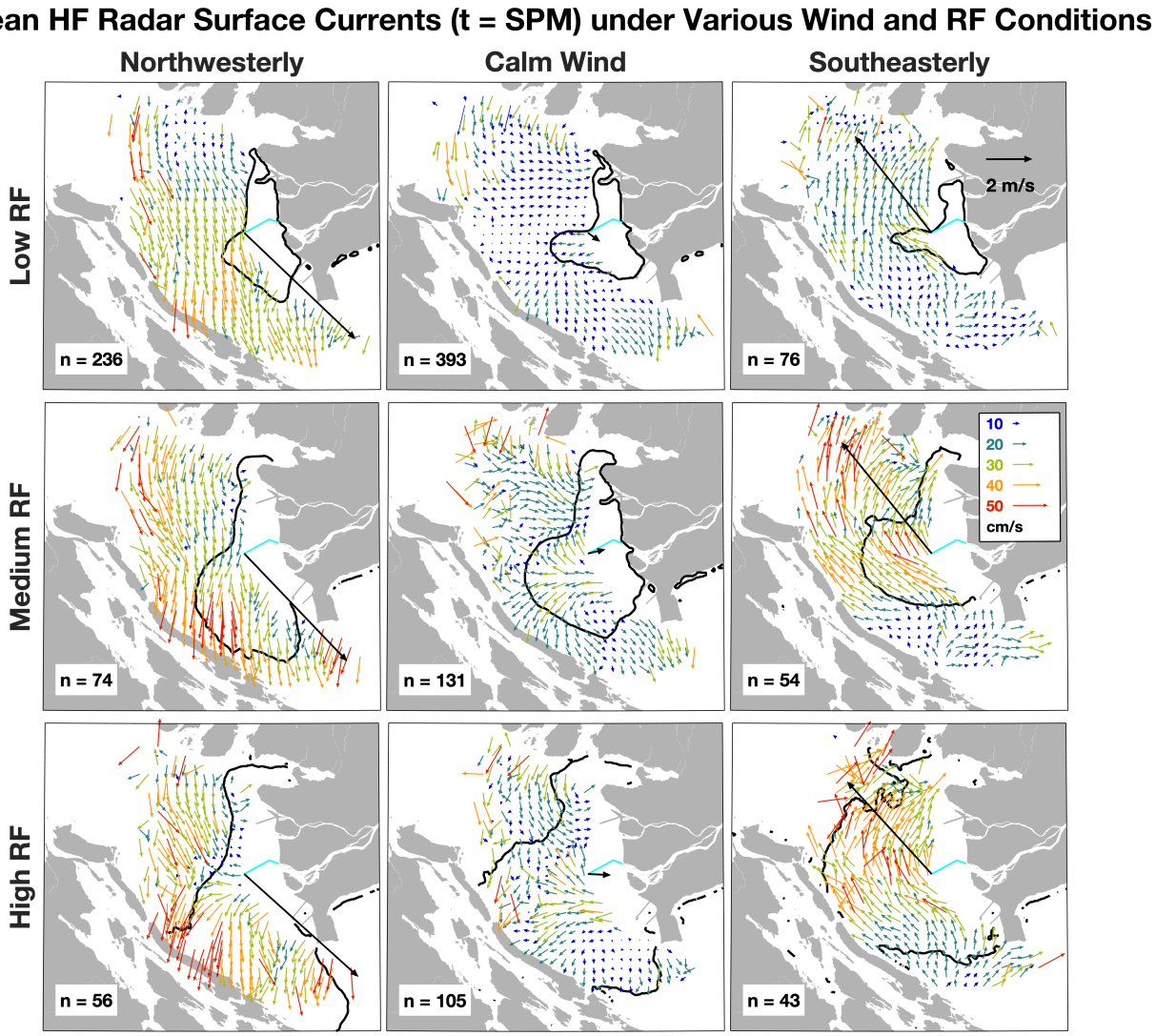

Figure 9: Surface currents in the Strait of Georgia under various wind and river flow conditions (for the time when MODIS derived SPM images are taken). (Current colors, wind vectors and plume contours are similar to Figure 3)

4.3.3 Correlations between Plume Area and River Discharge

(Function codes see Sec. 5.8)

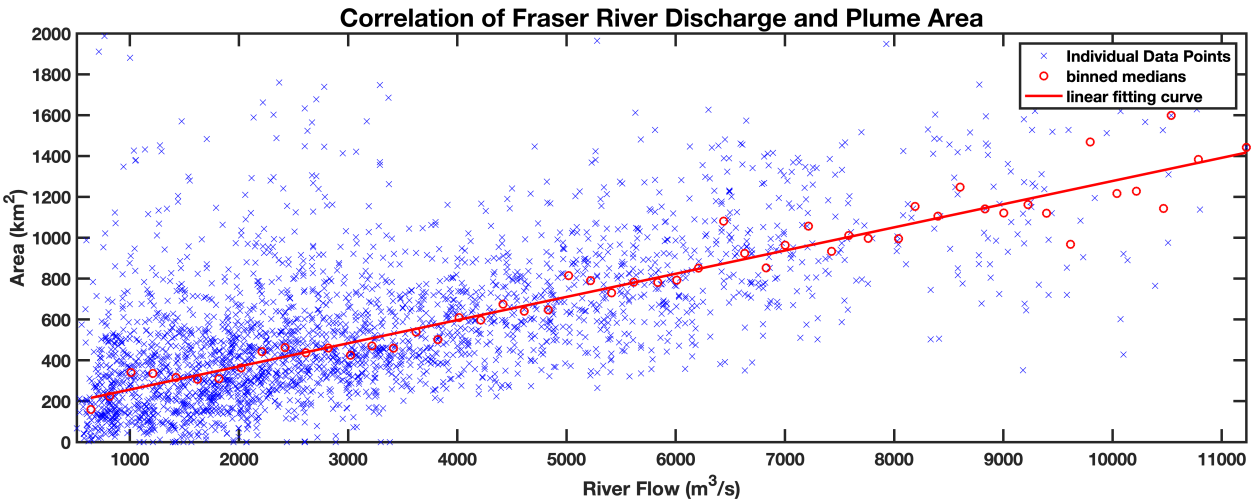

Figure 10: Correlations between Fraser River discharge and its plume area. Red circles are showing medians of binned data with a step of 200 m^3/s, and red curve is showing a linear fitting.

4.4 Tidal Influence on Fraser River Plume

(Function codes see Sec. 5.4, with `'option'-> 'spm', 'type'-> 'tide'`)

4.4.1 Plume Patterns over a Tidal Cycle

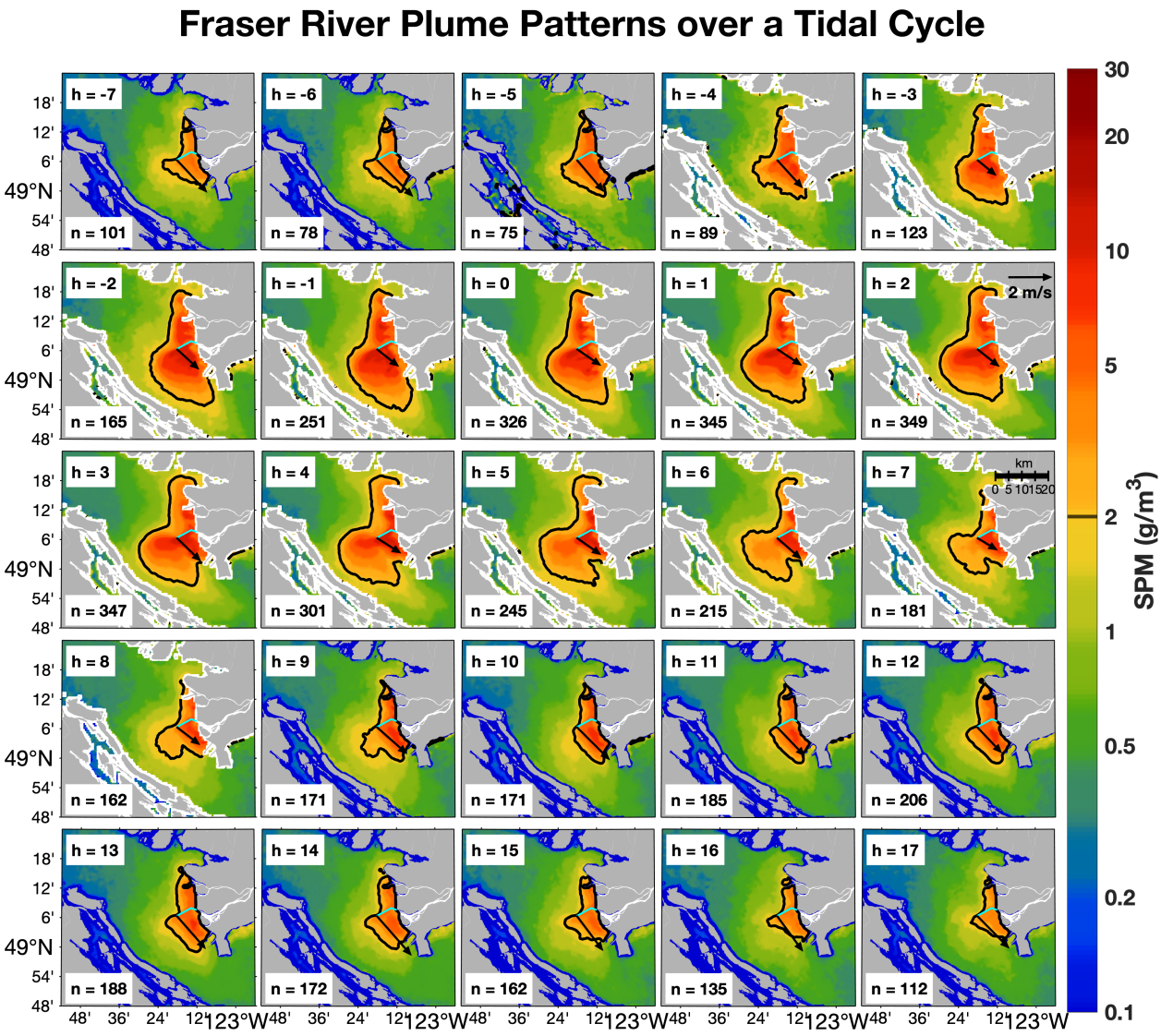

Figure 11: Fraser River Plume patterns over a tidal cycle. (Similar to Figure 2, but SPM images are binned to tide lag hours, in which h = 0 is the time for the lower low tide of the day (see Figure 1 for a better visualization of the elevation change within a tidal cycle)). Length scale is shown in the middle right subplot where h = 7 and wind vector scale is shown in the subplot where h = 2.

4.4.2 Surface Currents over a Tidal Cycle (t = SPM)

(Function codes see Sec. 5.4, with `'option'-> 'codar', 'type'-> 'tide'`)

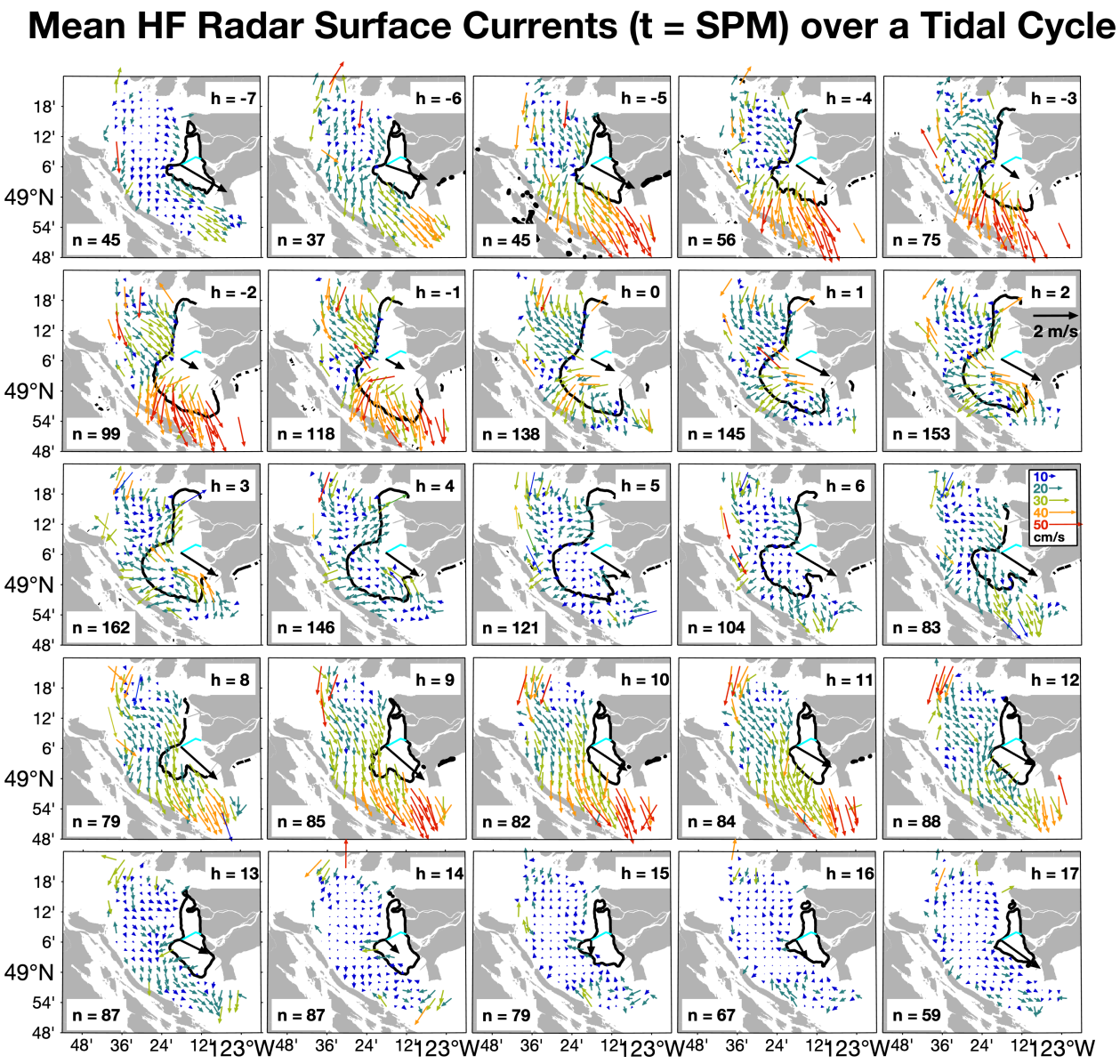

Figure 12: Surface currents in the Strait of Georgia over a tidal cycle (for the time when SPM images are taken). (Similar to Figure 3, but current vectors are binned to tide lag hours, in which h = 0 is the time for the lower low tide of the day). Vector scales and colors are shown in the middle right subplot where h = 7, wind vector scale is shown in subplot where h = 2.

4.4.3 Histogram of SPM Image Numbers over a Tidal Cycle

(Function codes see Sec. 5.9)

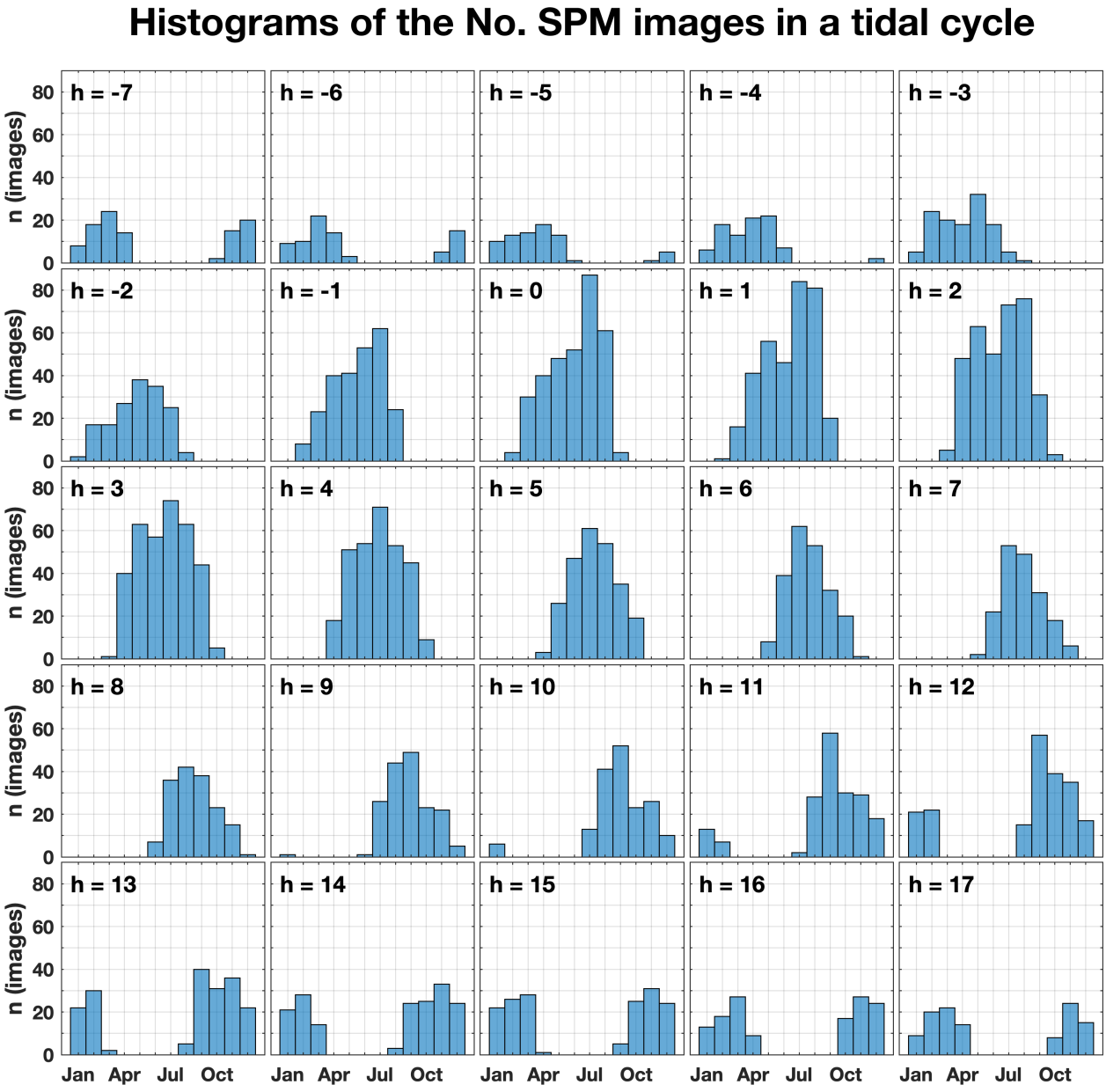

Figure 13: Numbers of valid SPM image distribution in each month over a tidal cycle. This figure is demonstrating the bias for analyzing the plume patterns over a tidal cycle.

4.4.4 Plume Area Change with Tide

(Function codes see Sec. 5.5, with `'option'-> 'tide'`)

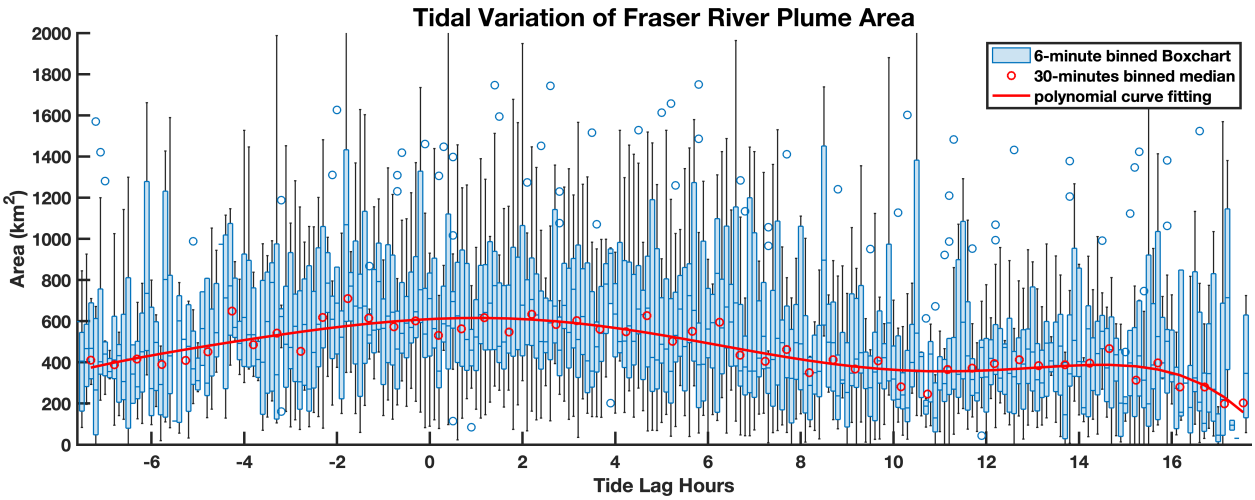

Figure 14: Fraser River Plume area variation within a tidal cycle. Figure shows the 6-minute binned boxchart with median and quartiles, 30-minute binned median in red circle and a sextic polynomial fitting function onto the red circles.

4.4.5 River Discharge Variation with Tide (t = SPM)

(Function codes see Sec. 5.5, with `'option'-> 'river_tide_at_spm'`)

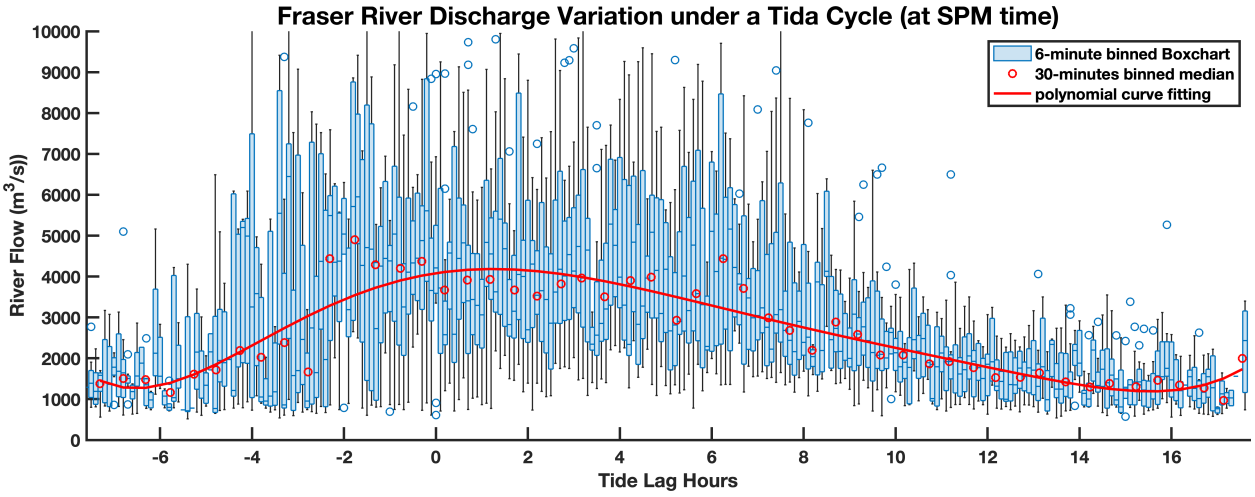

Figure 15: Boxchart of Fraser River Discharge (for the time when SPM images are taken) over a tidal cycle. (This can also show the bias when analyzing the SPM images under a tidal cycle.)

5 Codes and Annotations

5.1 project_main.m

```
%% Project Main

% This is the main fuction for the project used for Course EOSC 584.
% Running this function in order will load and process the required data,
% and plot all the fiugres in the project report
%
% Shumin Li, March 2021
% -------------------------------------------------------------------------

% loading data
% detailed description of the this function see project_dataload.m
[SPMGD, COD_GRID, coast_dir, river_dir, fraser, ...
    sandheads, bx, gridLon, gridLat, bath_lat, bath_lon, bath_z, ...
    sh, COD] = project_dataload;

% add the path of the project, in which all the functions are kept
addpath /Users/shuminli/Nextcloud/study_prjs/EOSC584_Project;

save_str = 'yes';
%% Comparison Plot
% plot comparison figures of sentinel-2, MODIS SPM images,
% and CODAR surface currents
comp_figure(sandheads, gridLon, coast_dir, gridLat, river_dir,...
    bath_lat, bath_z, bath_lon, COD)

%% Seasonality analysis

% Seasonality analysis plots of plume shape, surface currents, plume area,
% river discharge and wind speed/directions

% Plot the figures of plume pattern Seasonality
% using plume_subplots function, with 'option'->'spm' and 'type'->'season'
plume_subplots(sandheads, gridLon, gridLat, river_dir, ...
    coast_dir, bx, SPMGD, COD, fraser, sh,  'option','spm',...
    'type','season','save_fig',save_str);

% Plot of the figurers of surface currents seasonality
% using plume_subplots function, with 'option'->'codar' and 'type'->'season'
plume_subplots(sandheads, gridLon, gridLat, river_dir, ...
    coast_dir, bx, SPMGD, COD, fraser, sh,  'option','codar',...
    'type','season','save_fig',save_str);

% Boxchart of plume area seasonality
area_boxchart(SPMGD,'option','season','save',save_str)

% Fraser River Discharge Seasonality
river_seasonality(fraser,'save',save_str)

% Wind type seasonality for all wind data, and only for the time when SPM
% images are taken
wind_seasonality(sh,SPMGD,'save',save_str)

%% Wind and river influence analysis

% Plot the figures of plume patterns by different wind and river flow
% conditions, using plume_subplots function, with 'option'->'spm'
% and 'type'->'wind_rf'
plume_subplots(sandheads, gridLon, gridLat, river_dir, ...
    coast_dir, bx, SPMGD, COD, fraser, sh,  'option','spm',...
    'type','wind_rf','save_fig',save_str);

% Plot of the figurers of surface currents variation with wind and river
```

```
% flow, using plume_subplots function, with 'option'->'codar'
% and 'type'->'wind_rf'
plume_subplots(sandheads, gridLon, gridLat, river_dir, ...
    coast_dir, bx, SPMGD, COD, fraser, sh,  'option','codar',...
    'type','wind_rf','save_fig',save_str);

% Plot the correlation between plume area and river discharge
area_rf_relation(SPMGD, 'save',save_str)


%% Tidal influence analysis

% Plot the figures of plume patterns under a tidal cycle
% conditions, using plume_subplots function, with 'option'->'spm'
% and 'type'->'tide'
plume_subplots(sandheads, gridLon, gridLat, river_dir, ...
    coast_dir, bx, SPMGD, COD, fraser, sh,  'option','spm',...
    'type','tide','save_fig',save_str);

% Plot of the figurers of surface currents variation with wind and river
% flow, using plume_subplots function, with 'option'->'codar'
% and 'type'->'tide'
plume_subplots(sandheads, gridLon, gridLat, river_dir, ...
    coast_dir, bx, SPMGD, COD, fraser, sh,  'option','codar',...
    'type','tide','save_fig',save_str);

% Boxchart of plume area over a tidal cycle
area_boxchart(SPMGD,'option','tide','save',save_str)
```

5.2 project_dataload.m

```
function [SPMGD, COD_GRID, coast_dir, river_dir, fraser, ...
    sandheads, bx, gridLon, gridLat, bath_lat, bath_lon, bath_z, ...
    sh, COD] = project_dataload

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function loads the data, directories and other information which
% will be used in Shumin Li's course project for EOSC 584:
% Fraser River Plume Observation with Satellite Remote Sensing Data
%
% SPMGD: data struct of MODIS derived SPM (suspend particulate matters)
% COD_GRID: CODAR lon/lat grid and time
% coast_dir: directory of BC coast
% river_dir: directory of BC river and lake coastlines
% fraser: fraser river discharge data
% sh: surface winds from Sandheads lighthouse
% sandheads: locations of the Fraser River outreach at Sandheads
% bx: grid box used for plotting maps
% gridLon, gridLat: Longitude and Latitude for MODIS SPM grids
% bath_lat, bath_lon, bath_z: latitude, longitude and depthe of a fine
% scaled bathymetry in the Strait of Georgia

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%









% Load the processed MODIS SPM data, only the good days (2731 images from 2003 to 2019)
% are included in the struct. Some import fileds in SPMGD are
% val: [668  7292731   double], 668*792 pixels each image, 2731 images total
% tiimePDT: [1 2731  double], time in PDT
% wspd: [1 2731  double], wind speed (m/s)
% wdir: [1 2731  double], wind direction in degrees
```

```
% RF: [1 2731  double], Fraser River Discharge (m^3/s)
% lag_min: [1 2731  double], tide lag time in minutes relative to the lower
%          low tide of the day. (value between -7.5 and 17.5 hours)
% lag_h: [1 2731  double], tide lag time in hours relative to the lower
%          low tide of the day. (value are -7:17 hours)
% percent_valid: [1 2731  double]. Percetage of pixels with valid data in
%          the Strait of Georgia. (over 50% are considered to be good)
% center_valid: [1 2731  double]. Percetage of pixels with valid data in
%          the center of plume region. (over 60% are good)
% area: [1 2731  double]. area with SPM ≥ 2 g/m^3, which is considered as
%          the plume area
% plume_ave: [1 2731  double]. averaged SPM value in the plume area. (g/m^3)
load('/users/shuminli/Nextcloud/study_prjs/front_propagation/SPMGD.mat');

% Load COD_GRID.mat, which include the longitude, latitude and time
% information of the CODAR derived surface currents
load /users/shuminli/Nextcloud/data/CODAR/COD_GRID.mat;

% directory of a fine scale BC coast data
coast_dir = '/users/shuminli/Nextcloud/data/others/BCcoast.mat';

% directory of a fine scaled river and lake contours
river_dir = '/users/shuminli/Nextcloud/data/others/PNWrivers.mat';

% Load daily fraser river discharge data
load /users/shuminli/Nextcloud/data/river/Rich/FRASER.mat;

% Load wind data recorded from Sandhead Lighthouse
sh=load('/users/shuminli/Nextcloud/data/wind/sand.mat');

% locations of the Fraser River outreach at Sandheads
sandheads=[-123-[18.2 13.7 12]/60 ;...
            49+[6.4 8  7.6]/60 ];

% grid box used for plotting maps
bx=[-123.9 -122.9 48.8 49.4];

% generate the grids for MODIS SPM images,
latMin=48.5;
latMax=50;
lonMin=-124.5;
lonMax=-122.;
[gridLon,gridLat] = meshgrid([lonMin:0.00343:lonMax], [latMin:0.002246:latMax]);

% Extract the find-scale bathymetry information from
% british_columbia_3_msl_2013.nc, mask out the unnecessary information for
% the area that we do not care, and leave a clean data for plotting the 20m
% bathymetry in the future maps
fname = '/users/shuminli/Nextcloud/data/others/british_columbia_3_msl_2013.nc';
load /users/shuminli/Nextcloud/study_prjs/front_propagation/spm_bath_mask_for20m.mat;

lat_lim = [48.8 49.4]; lon_lim = [-123.8 -122.9];
lat = ncread(fname,'lat');
lon = ncread(fname,'lon');

ilon = lon≥ lon_lim(1) & lon≤ lon_lim(2);
ilat = lat≥ lat_lim(1) & lat≤ lat_lim(2);
Z = ncread(fname,'Band1',[find((ilon),1,'first') find((ilat),1,'first')],...
    [sum(ilon) sum(ilat)], [1,1]);

bath_lat = repmat(flipud(lat(ilat)),[1,sum(ilon)]);
bath_lon = repmat(lon(ilon)',[sum(ilat),1]);
bath_z = flipud(Z'*(-1));
% bath_in: index of bathymetry data within a plygon around the river plume
bath_in = inpolygon(bath_lon, bath_lat, spm_bath_mask_for20m(:,1),spm_bath_mask_for20m(:,2));
bath_z(¬bath_in) = nan;
```

```
% Load the CODAR data and name it as COD. Some important fields in COD are:
% mtime: [40221 1  double], time in UTC
% lon/lat: [74 93  double], longitudes and latitudes
% u/v: [74  9340221   double], zonal and meridional velocities in cm/s
CODa=load('/users/shuminli/Nextcloud/data/CODAR/SoG_radar_totals.mat');
COD=CODa.v2;
clear CODa
end
```

5.3 comp_figure.m

See Figure. 1

```
function comp_figure(sandheads, gridLon, coast_dir, gridLat, river_dir, ...
    bath_lat, bath_z, bath_lon, COD)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function makes 4 comprison plots between Sentinel-2 Image, MODIS
% Aqua derived SPM (Suspended Particulte Matters) images, and HF Radar
% derived surface currents on July 29th, 2020
%
% Sentinel-2 image is taken at 29-July-2020 11:09 a.m. (PDT)
%
% MODIS took 2 images on the same day at:
% 29-July-2020 12:10 p.m. and 29-July-2020 13:50 p.m. (pdt)
%
% HF Radar (CODAR) currents are chose at 29-July-2020 14:00 p.m. (pdt)
%
% Shumin Li, 2021, March
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Prepare data and address

% Address for the .nc file of Sentinel-2 data
address = strcat('/users/shuminli/Nextcloud/data/satellite/sentinel-2/',...
    'S2A_MSIL2A_20200729T190921_N0214_R056_T10UDV_20200729T233117_resampled.nc');
ncid = netcdf.open(address);

% longitude and latitude infor mation of sentinel-2 data
sentinel_lon = ncread(address,'lon');
sentinel_lat = ncread(address,'lat');

% extract bands with optical frequecy of red, gree and blue. and rescale it
% to the range of 0 to 1;
R = transpose(im2double(netcdf.getVar(ncid,4)));    % Red channel
G = transpose(im2double(netcdf.getVar(ncid,3)));    % Green channel
B = transpose(im2double(netcdf.getVar(ncid,2)));    % Blue channel
R = rescale(R,0,1);
G = rescale(G,0,1);
B = rescale(B,0,1);


% Remove the overexposure data (wich is usually found to be > 0.18),
% concatenate R, G and B into a [1098x1098x3 double] sized RGB array, and
% rescle it to [0 255];
R(abs(R)>0.18) = NaN;
G(abs(G)>0.18) = NaN;
B(abs(B)>0.18) = NaN;
RGB = cat(3, R, G, B);
RGB = uint8(rescale(RGB, 0, 255));

% addpath to use function tide_hrs, and the t_tide toolbox
addpath /Users/shuminli/Nextcloud/study_prjs/front_propagation;
addpath /Users/shuminli/Nextcloud/codes/t_tide
```

```
53
54
55  % 2 addresses for the SPM data files in 2020 July 29th
56  spm_add1 = '/Users/shuminli/Nextcloud/data/SPM/SPM_new_download/A2020211201000.mat';
57  spm_add2 = '/Users/shuminli/Nextcloud/data/SPM/SPM_new_download/A2020211215000.mat';
58
59  % extract spm data from the two files
60  spm_struct_1 = load(spm_add1,'data');
61  spm_data_1 = spm_struct_1.data.gridSPM;
62  spm_data_1(spm_data_1 ≤ 0) = nan;
63  spm_struct_2 = load(spm_add2,'data');
64  spm_data_2 = spm_struct_2.data.gridSPM;
65  spm_data_2(spm_data_2 ≤ 0) = nan;
66
67  % UTC times for the 2 SPM data and 1 Sentinel-2 data
68  t1 = datenum(2020,7,29,12,10,00) + 8/24;
69  t2 = datenum(2020,7,29,13,50,00) + 8/24;
70  t_sentinel = datenum(2020, 7, 29, 11, 9, 0) + 8/24;
71
72  %% Start Ploting the figure
73
74
75  figure('Position',[100 0 800 900],'color','w')
76  % making 2x2 subplot
77  ha = tight_subplot(2,2,[0.06 0.03],[ 0.04 0.11],[0.05 0.02]);
78
79  % a title above all subplots
80  sgtitle('Comparison of Sentinel-2 Image, MODIS SPM Image and HF Radar Currents',...
81      'FontSize',20,'FontWeight','bold')
82
83  % grid box for m_map
84  bx=[-123.9 -122.9 48.8 49.4];
85
86  %% Plot SPM Images
87  % plot the two spm image in axes ha(2) and ha(3)
88  for i = 2:3
89
90      %  assign data address and time for the 2 SPM data
91      switch i
92          case 2
93              spm_data = spm_data_1;
94              t = t1;
95          case 3
96              spm_data = spm_data_2;
97              t = t2;
98      end
99
100
101     axes(ha(i));
102
103     % Plot spm data with a "m_jet" colormap, and with a log colorscale
104     % within the range of [0.1 30] SPM (g/m^3)
105     m_proj('lambert','lon',bx(1:2),'lat',bx(3:4));
106     m_pcolor(gridLon, gridLat, spm_data);
107     caxis([0.1,30]);
108     colormap(m_colmap('jet','step',15));
109     set(gca,'colorscale','log');
110
111     % adding the line of river channel outreach at Sand Heads
112     m_line(sandheads(1,:),sandheads(2,:),'linewi',1,'color','cyan');
113
114     % adding coastlines, river lines, and grids
115     m_usercoast(coast_dir,'patch',[.7 .7 .7],'edgecolor','none');
116     m_usercoast(river_dir,'patch',[.7 .7 .7],'edgecolor','none');
117     m_grid('linestyle','-','gridcolor','w','tickdir','out','box','on','FontSize',15);
118
119     % title for the subplots
120     title(['MODIS SPM Image at ',datestr(t - 8/24) ,' p.m.'],'fontsize',15)
```

```
121
122     % Adding a log-scale colorbar inside the map using m_contfbar, draw a
123     % line at SPM = 2 g/m^3, which will be the same value as the contour
124     % in the map
125     [ax,¬]=m_contfbar(.8,[.5 .95],[0.1 30],(0.1:0.03:30),...
126         'edgecolor','none','endpiece','yes',...
127         'colorscale','log','YScale','log');
128     ax.YAxisLocation = 'right';
129     ax.YTick = [0.1 0.5 1 2 5 10 20 30];
130     ax.YLabel.String = 'SPM (g/m^3)';
131     ax.YLabel.FontWeight = 'bold';
132     ax.FontSize = 10;
133     ax.TickLength = [0.02 0.02];
134     yline(ax, 2,'linewi',2,'color','k')
135
136     % Adding the 20-m isobath line, which indicated the boundary of
137     % mudflats
138     hold on
139     [c2,h2] = m_contour(bath_lon,bath_lat,bath_z,[20 20],...
140         'linestyle','-','color','g',...
141         'ShowText','on','LabelSpacing',300);
142     clabel(c2,h2,'color','g')
143
144     % Adding a contour line with SPM = 2 g/m^3, which indicated the
145     % boundary of plume region
146     hold on
147     v = [2, 2];
148     m_contour(gridLon, gridLat,spm_data,v,'color','k','linewi',2)
149
150
151     sub_position = ha(i).Position;
152
153     % Adding a new axis at the lower left corner to show the tidal
154     % elevation within a tidal cycle and the current tidal elevation
155     ax_tide = axes ('box', 'off' );
156     ax_tide.Position = sub_position + [0.028 0.005 -0.22, -0.35];
157     % create a hourly time array of +/- 5 days since the give time
158     hour_time = t-5:1/24:t+5; % UTC time
159
160     % using t_xide function (from t_tide toolbox) to get the tidal
161     % elevations of the given time array
162     hour_tide = t_xtide('Point Atkinson (2)',hour_time-8/24); % PDT
163
164     % getting tide lag hours (which is define as the time relative to the
165     % lower low tide of the day) in minutes. tideval is an array for
166     % plotting a smooth tidal cycle plot later. Look atAppendix of more
167     % information of tide_hrs function
168     [lag_min, ¬, ¬, tideval] = ...
169         tide_hrs(hour_tide, hour_time - 8/24,t - 8/24,'no');
170
171     % plot the tidal elevation figure at lower left corner.
172     minilag = -7.5:1/60:17.5;
173     plot(minilag, tideval, 'b','Linewi',2);
174     hold on
175     xline(lag_min,'color','r','linewi',2)
176     xlim([-7.5 17.5]);
177     ylim([0 5]);
178     set(ax_tide, 'box','off','XAxisLocation','top','YAxisLocation',...
179         'right','FontWeight', 'bold','FontSize',10)
180     xticks(-7:3:17);
181     xlabel('Tide Lag Hours (h)');
182     ylabel('(m)');
183     grid on
184 end
185
186
187 %% Plot the Sentinel-2 Image
188
```

```
% Start plotting Setinel-2 image in the left subplot
axes(ha(1));
m_proj('lambert','lon',bx(1:2),'lat',bx(3:4));

% plot the RGB color onto the map using m_image function
m_image(sentinel_lon(:,6000), sentinel_lat(7500,:),RGB);
m_grid('linestyle','-','gridcolor',[1 1 1],'tickdir','out','box','on','FontSize',15);
hold on

% Adding the the contours where SPM = 2 g/m^3 in the SPM plot on to the
% sentinel plot as a comparison. yellow for 12:10 p.m. and red for 13:50 p.m.
v = [2, 2];
m_contour(gridLon, gridLat,spm_data_1,v,'color','y','linewi',2,'linest','-')
hold on
m_contour(gridLon, gridLat,spm_data_2,v,'color','r','linewi',2,'linest','-')


% adding Sand Heads locations
hold on
m_line(sandheads(1,:),sandheads(2,:),'linewi',1,'color','cyan');

% adding 20-m isobath contour
hold on
[c2,h2] = m_contour(bath_lon,bath_lat,bath_z,[20 20],...
    'linestyle','-','color','g',...
    'ShowText','on','LabelSpacing',300);
clabel(c2,h2,'color','g')

% title of the subplot
t = t_sentinel;
title(['Sentinel-2 Image at ',datestr(t - 8/24),' a.m.'],'fontsize',15)

% make some legend labels at the upper left corner
hold on
m_line([-123.85, -123.65],[49.35, 49.35],'linewi',2,'color','y')
m_text(-123.85, 49.33,'12:10 p.m.','color','y',...
    'Fontsize',12,'fontweight','bold')

m_line([-123.85, -123.65],[49.29, 49.29],'linewi',2,'color','r')
m_text(-123.85, 49.27,'13:50 p.m.','color','r',...
    'Fontsize',12,'fontweight','bold')


sub_position = ha(1).Position;
% adding a similar tidal elevation plot at the lower left corner for
% the time when Sentinel-2 image is taken
ax_tide = axes ('box', 'off' );
ax_tide.Position = sub_position + [0.028 0.005 -0.22, -0.35];

hour_time = t-5:1/24:t+5;
hour_tide = t_xtide('Point Atkinson (2)',hour_time-8/24); % PDT
[lag_min, ¬, ¬, tideval] = ...
    tide_hrs(hour_tide, hour_time - 8/24,t - 8/24,'no');
minilag = -7.5:1/60:17.5;
plot(minilag, tideval, 'b','Linewi',2);
hold on
xline(lag_min,'color','r','linewi',2)
xlim([-7.5 17.5]);
ylim([0 5]);
grid on
xticks(-7:3:17);
xlabel('Tide Lag Hours (h)','color','w');
ylabel('(m)','color','w');
set(ax_tide, 'box','off','XAxisLocation','top','YAxisLocation',...
    'right','FontWeight', 'bold','FontSize',10,'XColor',...
    'w','YColor','w','GridColor',[.15 .15 .15])


```

```
%% Plot HF Radar (CODAR) surface currents

axes(ha(4));
m_proj('lambert','lon',bx(1:2),'lat',bx(3:4));
% finding the index of the codar data which is closest to the time of the
% second SPM image
[¬,cod_idx] = min(abs(COD.mtime - t2));

% adding the plume contour from the second SPM image
v = [2, 2];
hold on
m_contour(gridLon, gridLat,spm_data_2,v,'color','r','linewi',2,'linest','-')

% coastlines and grids
m_usercoast(coast_dir,'patch',[.7 .7 .7],'edgecolor','none');
m_usercoast(river_dir,'patch',[.7 .7 .7],'edgecolor','none');
m_grid('linestyle','-','gridcolor','w','tickdir','out','box','on','FontSize',15);

% surface current arrows
m_vec(300,COD_GRID.lon,COD_GRID.lat,COD.u(:,:,cod_idx),...
    COD.v(:,:,cod_idx),'headlength',3,'shaftwidth',0.5,'facecolor','k');

% put a vector length scale legend at upper right of the subplot
m_vec(300, -123.1, 49.25, 50, 0, 'headlength',3,'shaftwidth',0.5)
m_text(-123.1, 49.23,'50 cm/s', 'FontSize',10, 'FontWeight','bold')

% put a legend of the plume region contour at upper left
m_line([-123.85, -123.65],[49.29, 49.29],'linewi',2,'color','r')
m_text(-123.85, 49.27,'13:50 p.m.','color','r',...
    'Fontsize',12,'fontweight','bold')

% adding sand heads locaitons
hold on
m_line(sandheads(1,:),sandheads(2,:),'linewi',1,'color','cyan');

% adding 20-m isobath contour
hold on
[c2,h2] = m_contour(bath_lon,bath_lat,bath_z,[20 20],...
    'linestyle','-','color','g',...
    'ShowText','on','LabelSpacing',300);
clabel(c2,h2,'color','g')

% title of the subplot
title(['HF Radar Currents at ',...
    datestr(COD.mtime(cod_idx) - 8/24), ' p.m.'],...
    'FontSize',15,'FontWeight','bold');

% put a similar tidal elevation axes at lower left corner
sub_position = ha(4).Position;
t = COD.mtime(cod_idx);
ax_tide = axes ('box', 'off' );
ax_tide.Position = sub_position + [0.028 0.005 -0.22, -0.35];

% getting tide lag hours
[lag_min, ¬, ¬, tideval] = ...
    tide_hrs(hour_tide, hour_time - 8/24, t - 8/24,'no');

% plot the tidal elevation figure at lower left corner.
minilag = -7.5:1/60:17.5;
plot(minilag, tideval, 'b','Linewi',2);
hold on
xline(lag_min,'color','r','linewi',2)
xlim([-7.5 17.5]);
ylim([0 5]);
set(ax_tide, 'box','off','XAxisLocation','top','YAxisLocation',...
    'right','FontWeight', 'bold','FontSize',10)
xticks(-7:3:17);
xlabel('Tide Lag Hours (h)');
```

```
ylabel('(m)');
grid on

% export the high resolution figure using fuction export_fig
save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
save_fname = 'comp_fig';
export_fig([save_dir, save_fname],'-png','-r400');

end
```

5.4 plume_subplots.m

9 figures could be plotted base on this function, with the following `'option'` and `'type'` arguments:

`'option'->'spm', 'type'->'season'`, See Figure. 2
`'option'->'codar', 'type'->'season'`, (Not shown in this report)
`'option'->'codar_at_spm', 'type'->'season'`, See Figure. 3

`'option'->'spm', 'type'->'wind_rf'`, See Figure. 8
`'option'->'codar', 'type'->'wind_rf'`, (Not shown in this report)
`'option'->'codar_at_spm', 'type'->'wind_rf'`, See Figure. 9

`'option'->'spm', 'type'->'tide'`, See Figure. 11
`'option'->'codar', 'type'->'tide'`, (Not shown in this report)
`'option'->'codar_at_spm', 'type'->'tide'`, See Figure. 12

For the figures that are not shown in this report, go to my Github repository https://github.com/shumin-li/EOSC-584-Project/tree/main/figures for more information.

```
function plume_subplots(sandheads, gridLon, gridLat, river_dir, ...
    coast_dir, bx, SPMGD, COD, fraser, sh,  varargin)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This fuction makes a figure with subplots of Fraser River plume patterns
% or HF radar derived surface currents with the following options and
% inputs
%
% Options:
% - 'option'
%   - 'spm': making plots of plume pattern based MODIS derived SPM Images
%   - 'codar': making plots of HF radar derived surface currents
%   - 'codar_at_spm': similar plots as the option codar, but only
%   averaging for the time when spm images are taken (i.e. it will have the
%   same bais that spm images would have)
%
% - 'type'
%   - 'season': devide into each month and make a 4*3 subplot
%   - 'wind_rf': devide into different wind (northwesterly, calm and
%   southeasterly) and river discharge (low, medium and high) conditions,
%   and make a 3*3 subplot
%   - 'tide': devide into tide lag hours relative to the lower low tide of
%   the day. 5*5 subplots
%
% - 'save_fig'
%   - 'yes': export the figure into the project directory
%   - 'no' (or any other string): do not save the figure
%
% Other input variables in order:
% - sandheads: locations of Sand Heads lighthouse, which indicates the
% river channel outreach.
% - gridLon/gridLat: longitude and latitude grids of SPM images
```

```
% - river_dir: data directory for a fine-scale river coastlines information
% - coast_dir: directory for fine-scale BC coastline
% - bx: box size used in m_proj function
% - SPMGD: data struct of selected good SPM images, for more information
% see the annotation in project_dataload fuction
% - COD: data struct for HF radar surface currents
% - fraser: daily fraser river discharge
% - sh: data struct for wind records from Sand Heads lighthouse station
%
% Shumin Li, 2021 March
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% initialize the data and prarameters

% this is a test input
% varargin = {'option','codar_at_spm','type','wind_rf','save_fig','no'};

% extract the option inputs from varargin (variable argument input)
k=1;
while k≤length(varargin)
    switch lower(varargin{k}(1:3))
        case 'opt'
            opt_str = varargin{k+1};
        case 'typ'
            typ_str = varargin{k+1};
        case 'sav'
            save_str = varargin{k+1};
    end
    k = k+2;
end


% since there are only two legal oprtions (spm / codar), we can call
% opt_bool as option boolean value. if opt_bool = true, plot the figure of
% spm image, otherwize, plot surface currents. report error for illegal
% input
% ------------------------------------------------------------------------
% (update on March 28th: add an option of 'codar_at_spm', however, since
% the option for 'codar' and 'codar_at_spm' are plotting very similar
% figures, we will keep the earlier binary separation with some minor
% adaptions. (if more options are added in the future, here should be
% changed to a 'switch' condition instead of 'if' condition)
if strcmpi(opt_str, 'spm')
    opt_bool = true;
elseif strcmpi(opt_str(1:5), 'codar')
    opt_bool = false;
else
    error("'option' has to be 'spm', 'codar' or 'codar_at_spm'")
end


% report error for illegal 'type' input
if ¬ any(strcmpi(typ_str,{'season','wind_rf','tide'}))
    error("'type' has to be either 'season', 'wind_rf' or 'tide'")
end


% assign different initial values for spm plots and codar plots
if opt_bool
    % color of the plume boundary contour (SPM = 2g/m^3)
    contour_col = 'k';
    % string in the big title above all subplots, part 1
    sg_title_1 = 'Fraser River Plume Patterns';
    % string as the filename for saving the fiugre, part 1
    save_str_1 = 'spm';
    % margin width of the subplots, (leave some extra space on the righ to
    % put the colorbar.
```

```
    marg_w = [0.06 0.1];
else
    % the contour color (for plume boundaries)
    contour_col = 'k';
    % margin width for the subplots
    marg_w = [0.08 0.08];

    % based on two different conditions
    % 1. 'codar' (for all codar data) and
    % 2. 'codar_at_spm' (for codar data at the time when spm images are taken)
    %
    % put in different data indexes, title names and filenames when
    % saving the file.
    switch opt_str
        case 'codar'

            sg_title_1 = 'Mean HF Radar Surface Currents';
            save_str_1 = 'codar';

            % find the wind vectors at Sand Heads at the same time when
            % CODAR data are recorded. (initialized with nan)
            codar_wu = nan(size(COD.mtime));
            codar_wv = nan(size(COD.mtime));

            % looping through the time for CODAR data, when there is a wind
            % data redord within one hour difference of the CODAR time, put
            % the data into codar_wu (u component) and codar_wv (v
            % component)
            for  k = 1:numel(COD.mtime)
                [sh_idx1, sh_idx2] = min(abs(sh.wtime - COD.mtime(k)));
                if sh_idx1 < 1/24
                    codar_wu(k) = sh.wu(sh_idx2);
                    codar_wv(k) = sh.wv(sh_idx2);
                end

            end

        case 'codar_at_spm'

            % in this case, we are trying to find the data indices for
            % CODAR data at which time SPM images are taken, and then find
            % the samilar data indices when wind data are valid.
            sg_title_1 = 'Mean HF Radar Surface Currents (t = SPM)';
            save_str_1 = 'codar_at_spm';

            codar_wu = nan(size(COD.mtime));
            codar_wv = nan(size(COD.mtime));

            % build the boolean array for codar data, initialized with
            % false, and write it as true when a valid SPM image time is
            % found to be falling in 1 hour from the CODAR time.
            codar_at_spm_idx = false(size(COD.mtime));

            % better_index are the index we used to define a 'good day' of
            % SPM image (coverage in the SoG is > 40% and the coverage in
            % the core plume region is > 50%)
            better_index = SPMGD.percent_valid > 0.4 & SPMGD.center_valid > 0.5;

            for k = 1:numel(COD.mtime)
                [spm_idx1, spm_idx2] = min(abs(SPMGD.timeUTC - COD.mtime(k)));

                if spm_idx1 < 1/24 && better_index(spm_idx2)
                codar_at_spm_idx(k) = true;

                % here, since SPMGD only have wind speed and directions, so
                % we need to do some calculations to get the u and v
                % component of the wind.
                codar_wu(k) = SPMGD.wspd(spm_idx2)*cosd(SPMGD.wdir(spm_idx2));
```

```
                codar_wv(k) = SPMGD.wspd(spm_idx2)*sind(SPMGD.wdir(spm_idx2));
                end

            end
    end
end


% assign variables for each type of the plot
switch typ_str

    % subplots in each month
    case 'season'
        % legend_idx: which subplot to put the legend (length/vector scale)
        legend_idx = 6;
        % save_str_2: part 2 of the filename string
        save_str_2 = '_by_month';
        % fig_pos: position of the figure
        fig_pos = [100 100 630 900];
        % row: number of rows of the subplots
        row = 4;
        % col: number of columns of the subplots
        col = 3;
        % gap: [gap_h gap_w] gaps in height and width between subplots
        gap = [0.01 0];
        % marg_h: [lower upper] lower and upper margins for subplots
        marg_h = [0.05 0.09];
        % sg_title_2: part 2 of the string in the sg_title
        sg_title_2 = ' in Each Month';
        % sgt_FS: sg_title FrontSize
        sgt_FS = 18;
        % cbar_pos: colorbar position
        cbar_pos = [0.91 0.048 0.025 0.863];
        % cod_thr: surface currents threshold, above which arrows are not
        % shown
        cod_thr = 30;
        % vec_scale: scale factor of m_vec funtion to plot arrows
        vec_scale = 250;
        % cod_gap: every 3 data points in row and colum of the data grid is
        % shown
        cod_gap = 2;
        % ruler_y: the Y Position of the m_rule, which indicates the length
        % scale in the subplot No. legend_idx
        ruler_y = 0.76;
        % ruler_FS: FontSize of the ruler above
        ruler_FS = 10;
        % vec_leg_FS: arrow length scale FontSize in the subplot No.
        % legend_idx
        vec_leg_FS = 10;

        % wind_legend_idx: the index of the subplot where we are going to
        % put a legend length of a wind vector example of 2 m/s
        wind_legend_idx = 3;
        % wind_leg_FS: FontSize of this legend
        wind_leg_FS = 12;

        % assign differnt initial values of 'scale', 'headlength' and
        % 'shaftwidth' parameter for spm images and codar images when using
        % m_vec funcion.
        if opt_bool
            % 'scale' of wind vector
            wvec_scale = 5;
            % 'headlength' of wind vector arrow
            wvec_hl = 6;
            % 'shaftwidth' of wind vector arrow
            wvec_sw = 1;

```

```
237         else
238             wvec_scale = 12;
239             wvec_hl = 2;
240             wvec_sw = 0.4;
241         end
242
243
244
245     % subplots in different wind and river flow conditions. (descriptions
246     % of parameters see case 'season' above)
247     case 'wind_rf'
248         legend_idx = 6;
249         save_str_2 = '_by_wind_rf';
250         fig_pos = [100 100 800 800];
251         row = 3;
252         col = 3;
253         gap = [0.005 0.005];
254         marg_h = [0.05 0.105];
255         sg_title_2 = ' under Various Wind and RF Conditions';
256         sgt_FS = 20;
257         cbar_pos = [0.91 0.048 0.025 0.846];
258         cod_thr = 50;
259         vec_scale = 400;
260         cod_gap = 2;
261         ruler_y = 0.76;
262         ruler_FS = 10;
263         vec_leg_FS = 10;
264
265         wind_legend_idx = 3;
266         wind_leg_FS = 12;
267
268         if opt_bool
269             wvec_scale = 5;
270             wvec_hl = 6;
271             wvec_sw = 1;
272
273         else
274             wvec_scale = 12;
275             wvec_hl = 2;
276             wvec_sw = 0.4;
277         end
278
279
280         % Some specific data index calculation for type 'wind_rf'
281         %
282         % Finding the SPM indices of calm wind (< 4 m/s), strong
283         % northwesterlies, and strong southeasterlies
284         w_spd = SPMGD.wspd;
285         w_dir = SPMGD.wdir;
286
287         calm_idx = w_spd < 4;
288         strong_nw_idx = w_dir ≥ 270 & w_dir ≤ 360 & ¬calm_idx;
289         strong_se_idx = w_dir ≥ 90 & w_dir ≤ 180 & ¬calm_idx;
290
291         % Finding the SPM indices of calm low river flow (< 3000 m^3/s),
292         % median river flow (3000 < SPMGD.RF < 5000), and high river flow
293         rf_low = SPMGD.RF < 3000;
294         rf_med = SPMGD.RF ≥ 3000 & SPMGD.RF ≤ 5000;
295         rf_high = SPMGD.RF > 5000;
296
297         % cell of the indices for wind and river conditions
298         wind_idx = {strong_nw_idx, calm_idx, strong_se_idx};
299         river_idx = {rf_low, rf_med, rf_high};
300
301         % if option is codar, then we need data indices for COD
302         if ¬opt_bool
303             cod_rf_low(size(COD.mtime)) = false;
304             cod_rf_med(size(COD.mtime)) = false;
```

```
            cod_rf_high(size(COD.mtime)) = false;

            cod_sh_calm(size(COD.mtime)) = false;
            cod_sh_nw(size(COD.mtime)) = false;
            cod_sh_se(size(COD.mtime)) = false;


            for k = 1: numel(COD.mtime)
                % finding the indices of difference river flow conditions
                % of COD data
                [fraser_idx1, fraser_idx2] = ...
                    min(abs(fraser.mtime - COD.mtime(k)));

                if fraser_idx1 < 1
                    if fraser.flow(fraser_idx2) < 3000
                        cod_rf_low(k) = true;
                    elseif fraser.flow(fraser_idx2) ≥ 3000 && ...
                            fraser.flow(fraser_idx2) ≤ 5000
                        cod_rf_med(k) = true;
                    else
                        cod_rf_high(k) = true;
                    end
                end

                % finding the indices of difference wind conditions of
                % COD data
                [sh_idx1, sh_idx2] = min(abs(sh.wtime - COD.mtime(k)));

                if sh_idx1 < 1/24
                    if sh.wspd(sh_idx2) < 4
                        cod_sh_calm(k) = true;
                    elseif sh.wspd(sh_idx2) ≥ 4 && ...
                            sh.wdir(sh_idx2) ≥ 270 && ...
                            sh.wdir(sh_idx2) ≤ 360
                        cod_sh_nw(k) = true;
                    elseif sh.wspd(sh_idx2) ≥ 4 && ...
                            sh.wdir(sh_idx2) ≥ 90 && ...
                            sh.wdir(sh_idx2) ≤ 180
                        cod_sh_se(k) = true;
                    end
                end
            end

            cod_rf_idx = {cod_rf_low, cod_rf_med, cod_rf_high};
            cod_sh_idx = {cod_sh_nw, cod_sh_calm, cod_sh_se};

        end

        % strings used for xlabels and ylabels
        wind_str = {'Northwesterly','Calm Wind','Southeasterly'};
        river_str = {'Low RF','Medium RF', 'High RF'};

        % indices used later in the loop to tell which data indices should
        % a subplot draw
        rf_loop = [1 1 1 2 2 2 3 3 3];
        wind_loop = [1 2 3 1 2 3 1 2 3];


    % subplots under a tidal cycle
    case 'tide'
        legend_idx = 15;
        save_str_2 = '_by_tide';
        fig_pos = [100 100 650 600];
        row = 5;
        col = 5;
        gap = [0.005 0.005];
        marg_h = [0.05 0.09];
```

```
373         sg_title_2 = ' over a Tidal Cycle';
374         sgt_FS = 20;
375         cbar_pos = [0.91 0.048 0.025 0.863];
376         cod_thr = 50;
377         vec_scale = 500;
378         cod_gap = 3;
379         ruler_y = 0.86;
380         ruler_FS = 7;
381         vec_leg_FS = 7;
382
383         wind_legend_idx = 10;
384         wind_leg_FS = 9;
385
386         if opt_bool
387             wvec_scale = 6;
388             wvec_hl = 5;
389             wvec_sw = 1;
390
391         else
392             wvec_scale = 15;
393             wvec_hl = 2;
394             wvec_sw = 0.4;
395         end
396
397
398         % Here is the function for generating tide lag hours of each
399         % recorded HF radar time. it took about 6 seconds to calculate. So
400         % here, in order to save time, we will save the data in advance and
401         % use it when we need it. (tide_hrs is a function I wrote, it will
402         % be fully explained somewhere else in the documents (see Appendix
403         % in the report), and t_xtide was from t_tide toolbox)
404         %    |
405         %    |
406         %    v
407         %
408         %     tidet = COD.mtime(1)-2:1/24:COD.mtime(end)+2;
409         %     tideh = t_xtide('Point Atkinson (2)',tidet - 8/24);
410         % for k = 1: numel(COD.mtime)
411         %     [m, l, ¬,¬] = ...
412         %         tide_hrs(tideh, tidet - 8/24, COD.mtime(k) - 8/24, 'no');
413         %     cod_tide_l(k) = l;
414         %     cod_tide_min(k) = m;
415         %
416         % end
417         % save('/Users/shuminli/Nextcloud/study_prjs/EOSC584_Project/cod_lag_data.mat',...
418         %             'cod_tide_l','cod_tide_min')
419
420         load('/Users/shuminli/Nextcloud/study_prjs/EOSC584_Project/cod_lag_data.mat')
421 end
422
423 month_str_short = {'Jan','Feb','Mar','Apr','May','Jun',...
424     'Jul','Aug','Sep','Oct','Nov','Dec'};
425
426
427 %% start making the plot
428
429 % clf
430
431 % the colorspecs of CODAR surface current arrows
432 vec_color = m_colmap('jet',6);
433 % separate the arrows into 6 steps, and use a different color representing
434 % each magnitude of the currents.
435 vec_step = round(linspace(0, cod_thr, 6));
436
437 % initialize figure size, subplot numbers and margins, and m_map projections
438 figure('color','w','Position',fig_pos);
439 ha = tight_subplot(row,col,gap,marg_h,marg_w);
440 m_proj('lambert','lon',bx(1:2),'lat',bx(3:4));
```

```
441
442 % Loop for row*col subplots
443 for i = 1:row*col
444
445     % making data index (finding the indices to average the spm image or
446     % codar surface currents)
447
448     % better_index is trying to find the images with over 40% coverage in
449     % the Strait of Georgia and over 50% valid data in the plume region.
450     % (to some degree, this is a simple quality control index)
451     better_index = SPMGD.percent_valid > 0.4 & SPMGD.center_valid > 0.5;
452
453
454     switch typ_str
455
456         case 'season'
457             % finding the month for each spm image time
458             [¬,spm_month,¬,¬,¬,¬] = datevec(SPMGD.timePDT);
459             % find the indices of the month in looping (i.e. i)
460             month_idx = spm_month == i;
461
462             % put the above two indices together
463             data_idx =  month_idx & better_index;
464
465             % finding the data indices for codar data
466             [¬,cod_month,¬,¬,¬,¬] = datevec(COD.mtime);
467             cod_idx = cod_month == i;
468
469         case 'wind_rf'
470             % finding the indices for corresponded wind and river flow
471             % conditions
472             data_idx =  river_idx{rf_loop(i)} & ...
473                 wind_idx{wind_loop(i)} & better_index;
474             if ¬opt_bool
475                 cod_idx =  cod_rf_idx{rf_loop(i)} & cod_sh_idx{wind_loop(i)};
476             end
477
478         case 'tide'
479
480             % tide_idx and cod_tide_idx are trying to filter out the days
481             % of weak tidal variations (if the lower low tide of the day is
482             % even higher than 2.1 m in Pt. Atkinson, there we would
483             % consider that day does not have a "big ebb")
484             tide_idx = SPMGD.LowTide < 2.1;
485             cod_tide_idx = cod_tide_l < 2.1;
486
487             % find the images which are taken +/- 1 hour within the given
488             % tide lag hour
489             j = i - 8;
490             lag_2h =  SPMGD.lag_min > j-1 & SPMGD.lag_min < j+1;
491             cod_lag_2h = cod_tide_min > j-1 & cod_tide_min < j+1;
492
493             % Count tide lag hours from 17h to 18h into "-7h" segment,
494             % and also count the -8h to -7h period as part of "17h" segment
495             if j == -7
496                 lag_2h =  SPMGD.lag_min > 17 | SPMGD.lag_min < -6;
497                 cod_lag_2h = cod_tide_min > 17 | cod_tide_min < -6;
498             elseif j == 17
499                 lag_2h =  SPMGD.lag_min > 16 | SPMGD.lag_min < -7;
500                 cod_lag_2h =  cod_tide_min > 16 | cod_tide_min < -7;
501
502             end
503
504             % put the earlier creterion together
505             data_idx =  lag_2h & better_index & tide_idx;
506             cod_idx = cod_lag_2h & cod_tide_idx;
507
508     end
```

```
509
510     % for the case when 'option' input is 'codar_at_spm', we will merge the
511     % cod_idx with codar_at_spm_idx. (sometimes, these two array are the
512     % tranposed size to each other, so to avoid that error, we need to put
513     % them as arrays with same size.
514     if strcmpi(opt_str,'codar_at_spm')
515         if size(cod_idx,1) == size(codar_at_spm_idx,1)
516             cod_idx = cod_idx & codar_at_spm_idx;
517         elseif size(cod_idx,1) == size(codar_at_spm_idx,2)
518             cod_idx = cod_idx & codar_at_spm_idx';
519         end
520     end
521
522
523     % extract zonal and meridional velocity vectors of given indices
524     spm_u = COD.u(:,:,cod_idx);
525     spm_v = COD.v(:,:,cod_idx);
526
527     % calculate the mean value of each component
528     u_data = nanmean(spm_u,3);
529     v_data = nanmean(spm_v,3);
530
531     % eliminate the outliers (speed greater than the threshold -- cod_thr)
532     cod_spd = sqrt(u_data.^2 + v_data.^2);
533     u_data(cod_spd > cod_thr) = nan;
534     v_data(cod_spd > cod_thr) = nan;
535
536
537     % find the pixel medians for images of the data indices
538     spm_median = nanmedian(SPMGD.val(:,:,data_idx),3);
539
540     % apply a 2-D median filter with 5x5 window size to make the plot look
541     % smoother while maintaining the plume boundary correctly
542     spm_data = medfilt2(spm_median,[5,5]);
543
544
545     % get into each subplot within the loop
546     axes(ha(i));
547
548
549     if opt_bool
550
551         % plot the spm data with log colorscale using m_pcolor and a m_jet
552         % colormap with 15 'step'
553         m_pcolor(gridLon, gridLat, spm_data);
554         caxis([0.1,30]);
555         colormap(m_colmap('jet','step',15));
556         set(gca,'colorscale','log');
557
558
559     end
560
561     % adding coastline and river coast
562     m_usercoast(coast_dir,'patch',[.7 .7 .7],'edgecolor','none');
563     m_usercoast(river_dir,'patch',[.7 .7 .7],'edgecolor','none');
564
565     % add the contour of SPM = 2 g/m^3
566     hold on
567     v = [2, 2];
568     m_contour(gridLon, gridLat,spm_data,v,'color',contour_col,'linewi',1.5);
569
570
571
572     % calculate the number of images/currents used in this subplot, and
573     % transfer it from number into string, for the use of next step
574     if opt_bool
575         str_n = num2str(sum(data_idx));
576     else
```

```
577         str_n = num2str(size(spm_u,3));
578     end
579
580
581     switch typ_str
582         case 'season'
583             % put the month string onto the upper left corner
584             m_text(-123.85, 49.32, month_str_short{i},'FontSize',12,...
585                 'FontWeight','bold','edgecolor','none','backgroundcolor','w');
586
587             % adding a text box at the lower left corner showing how many
588             % images/currents data are used in this subplot (str_n)
589             m_text(-123.85, 48.86,['n = ',str_n],'FontSize',12,...
590                 'FontWeight','bold','edgecolor','none','backgroundcolor','w');
591
592         case 'wind_rf'
593             % adding a text box at the lower left corner showing how many
594             % images/currents data are used in this subplot (str_n)
595             m_text(-123.85, 48.86,['n = ',str_n],'FontSize',12,...
596                 'FontWeight','bold','edgecolor','none','backgroundcolor','w');
597
598         case 'tide'
599             if opt_bool
600                 % put tide lag hour onto upper left of each subplot for spm
601                 % plot
602                 m_text(-123.85, 49.33, ['h = ', num2str(i-8)],...
603                     'FontSize',9,'FontWeight','bold',...
604                     'edgecolor','none','backgroundcolor','w');
605             else
606                 % same number but different locations for codar plot
607                 m_text(-123.15, 49.33, ['h = ', num2str(i-8)],...
608                     'FontSize',9,'FontWeight','bold',...
609                     'edgecolor','none','backgroundcolor','w');
610             end
611
612             % put a text about the number of used data at lower left corner
613             m_text(-123.85, 48.86,['n = ',str_n],'FontSize',...
614                 9,'FontWeight','bold',...
615                 'edgecolor','none','backgroundcolor','w');
616     end
617
618
619
620
621
622
623     if opt_bool
624
625         % if plotting spm image, put a length scale using m_ruler onto
626         % subplot in the middle right (No. legend_idx)
627         if i == legend_idx
628             m_ruler([0.69, 0.97],ruler_y,'FontSize',ruler_FS);
629         end
630
631     else
632         % sometimes, it will be too crowded to put all the current vectors
633         % onto a small subput, to better visualize, we will only represent
634         % the every two or three (cod_gap) grid in rows and columns.
635         %
636         % here are the longitude, latitude, u, v, and speed information for
637         % the extracted grid
638         lon = COD.lon(1:cod_gap:end, 1:cod_gap:end);
639         lat = COD.lat(1:cod_gap:end, 1:cod_gap:end);
640         u = u_data(1:cod_gap:end, 1:cod_gap:end);
641         v = v_data(1:cod_gap:end, 1:cod_gap:end);
642         spd = sqrt(u.^2 + v.^2);
643
644         % using different colors (in vec_color) to represent differnt
```

```
645         % magnitudes of vector arrows
646         for j = 1:numel(vec_step)-1
647             uv_idx = spd > vec_step(j) & spd ≤ vec_step(j+1);
648             m_vec(vec_scale,lon(uv_idx),lat(uv_idx),...
649                 u(uv_idx),v(uv_idx), vec_color(j,:),'headlength',1,...
650                 'shaftwidth',0.2,'headangle',40);
651         end
652
653
654         % plot a legend box showing the correspondance of color spectrum
655         % and arrow magnitudes
656         if i == legend_idx
657
658             % boundary of legend box
659             bndry_lon=[-123.16 -123.16 -122.92 -122.92 -123.16];
660             bndry_lat=[49.13   49.38   49.38   49.13   49.13];
661
662             % plot a legend box with white background color
663             m_line(bndry_lon, bndry_lat, 'linewi',1, 'color','k');
664             m_patch(bndry_lon, bndry_lat,'w');
665
666             % add the legend arrows and their speed
667             for j = 1:5
668             m_vec(vec_scale, -123.06, 49.4 - 0.04*j, vec_step(j+1), 0, ...
669                 vec_color(j,:), 'headlength',1,...
670                 'shaftwidth',0.2,'headangle',40)
671             m_text(-123.14, 49.4 - 0.04*j, num2str(vec_step(j+1)), ...
672                 'color',vec_color(j,:),'FontSize',...
673                 vec_leg_FS, 'FontWeight','bold')
674             end
675
676             % add the unit at bottom in black color
677             m_text(-123.14, 49.4 - 0.04*(j+1), 'cm/s', ...
678                 'color','k','FontSize',...
679                 vec_leg_FS, 'FontWeight','bold')
680         end
681
682     end
683
684
685     % adding the line showing sandheads (river outreach)
686     hold on
687     m_line(sandheads(1,:),sandheads(2,:),'linewi',1,'color','cyan');
688
689
690
691     % add the wind vector onto the sandheads
692     if opt_bool
693         data_wu = nanmean(SPMGD.wspd(data_idx).*cosd(SPMGD.wdir(data_idx)));
694         data_wv = nanmean(SPMGD.wspd(data_idx).*sind(SPMGD.wdir(data_idx)));
695     else
696         data_wu = nanmean(codar_wu(cod_idx));
697         data_wv = nanmean(codar_wv(cod_idx));
698     end
699      m_vec(wvec_scale, sandheads(1,1),sandheads(2,1),data_wu, data_wv,'k', ...
700         'headlength', wvec_hl, 'shaftwidth', wvec_sw,'headangle',40)
701
702
703
704     % location to put the legend for wind vector of 2 m/s
705     wvec_lon = -123.14;
706     wvec_lat = 49.35;
707
708     % for some figures, we want to put this legend to somewhere a little
709     % bit lower
710     if (¬ opt_bool && strcmpi(typ_str,'tide')) || strcmpi(typ_str,'wind_rf')
711         wvec_lat = 49.25;
712     end
```

```
713
714     % add the legend for wind vectors
715     if i == wind_legend_idx
716         m_vec(wvec_scale, wvec_lon, wvec_lat, 2, 0, 'k', 'headlength', wvec_hl, ...
717             'shaftwidth', wvec_sw,'headangle',40)
718         m_text(wvec_lon, wvec_lat - 0.05,{'2 m/s'},'FontSize',wind_leg_FS,...
719             'FontWeight','bold','color','k')
720     end
721
722
723
724     % adjust the grid tick labels
725     switch typ_str
726         case {'season','tide'}
727             % only show the logitude and latitude TickLabels in the left
728             % column and bottom row
729
730             % both ytick and xtick for the lower-left one are kept
731             if i == (row-1)*col +1
732                 m_grid('linestyle','none','tickdir','out','box','on',...
733                     'FontSize',12);
734
735             % delete xticks for the leftmost column (except the lower-left one)
736             elseif mod(i,col) == 1
737                 m_grid('linestyle','none','tickdir','out','box','on',...
738                     'FontSize',12, 'xtick',[]);
739
740             % delete yticks for the bottom row
741             elseif i > (row-1)*col +1
742                 m_grid('linestyle','none','tickdir','out','box','on',...
743                     'FontSize',12, 'ytick',[]);
744
745             % delete both xticks and yticks for all other subplots
746             else
747                 m_grid('linestyle','none','tickdir','out','box','on',...
748                     'FontSize',12, 'ytick',[],'xtick',[]);
749             end
750
751         case 'wind_rf'
752
753             % put ylabels for the leftmost column
754             if mod(i,col) == 1
755                 ylabel(river_str{rf_loop(i)},'FontSize',18,...
756                     'FontWeight','bold');
757                 ha(i).YLabel.Position(1:2) = [-0.0060 0];
758
759             end
760
761             % put xlabels on top for the top row
762             if i ≤ col
763                 xlabel(wind_str{wind_loop(i)},'FontSize',18,...
764                     'FontWeight','bold');
765                 set(ha(i),'XAxisLocation','top');
766                 ha(i).XLabel.Position(1:2) = [0 0.0054];
767             end
768
769             % delete all xticks and yticks for the subplots
770             m_grid('linestyle','none','tickdir','out','box','on',...
771                 'FontSize',12, 'ytick',[],'xtick',[]);
772
773     end
774
775
776     % A title above all subplots
777     sgtitle([sg_title_1,sg_title_2],'FontSize',sgt_FS,'FontWeight','bold');
778
779 end
780
```

```
781
782 % Making a colorbar at the right side of the figure if the 'option' is
783 % 'spm'
784 if opt_bool
785     % set position and make axes
786     cbar = axes('Position',cbar_pos,'box','on');
787     % cy: y values for filling out the colors in colorbar. (a log-scale
788     %   spaced array)
789     cy = 10.^(linspace(-1,log10(30),200));
790     cy = flipud(cy');
791     % cx: values for the x axis (it could be anything)
792     cx = [0,1];
793     % Y: a meshgrid data for cx and cy
794     [¬, Y] = meshgrid(cx, cy);
795     % make a pcolor plot into the axes box as the colorbar
796     pcolor(cx, cy, Y); shading flat;
797     % reset some properties of the colorbar
798     set(cbar,'colorscale','log','yscale','log','YAxisLocation','right');
799     % make a same colormap as we did in the subplots
800     colormap(m_colmap('jet','step',15));
801     % only show some certain YTick labels
802     cbar.YTick = [0.1 0.2 0.5 1 2 5 10 20 30];
803     cbar.TickLength = [0.028 0.02];
804     % delete XTick labels
805     cbar.XTick = [];
806     cbar.FontSize = 12;
807     ylim([0.1 30]);
808     cbar.FontWeight = 'bold';
809     hold on
810     % draw a horizonal line at SPM = 2 g/m^3
811     yline(2,'linewi',2,'color','k')
812     ylabel('SPM (g/m^3)','FontSize',14,'FontWeight','bold',...
813         'Position',[1.8 1.7321 -1])
814 end
815
816 % directory and filenames for saving the gigure
817 save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
818 save_fname = strcat(save_str_1, save_str_2);
819
820 % if 'save_fig' input is 'yes', then use function export_fig to export
821 % a nice high-resolution figure
822 if strcmpi(save_str,'yes')
823     export_fig([save_dir, save_fname],'-png','-r400');
824 end
825
826
827 end
```

5.5 area_boxchart.m

2 figures could be plotted base on this function, with the following `'option'` argument:

`'option'->'season'`, See Figure. 4
`'option'->'tide'`, See Figure. 14
`'option'->'river_tide_at_spm'`, See Figure. 15

```
1 function area_boxchart(SPMGD,varargin)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % This function makes a nice boxchart plot of plume area variation (or some
5 % other data (e.g. river discharge) according to seasonality or tidal cycle
6 % for the given options:
7 %
```

```
% - 'option'
%   - 'season': plume area binned into a yearly cycle (default)
%   - 'tide': plume area binned into a tidal cycle
%   - 'river_tide_at_spm': Fraser River discharge (when SPM images are
%         taken) are binned into a tidal cycle
% - 'save'
%   - 'yes': save the figure into current directory
%   -  'no' or other string: don't save (default)
%
% Shumin Li, March 2021
%
%
% Update March 28th, 2021, added an option for 'river_tide_at_spm'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Extrat option information from vargarin

k=1;
while k≤length(varargin)
    switch lower(varargin{k}(1:3))
        case 'opt'
            opt_str = varargin{k+1};
        case 'sav'
            sav_str = varargin{k+1};
    end
    k = k+2;
end


if ¬any(opt_str)
    disp("default plotting option: 'season'")
    disp("to plot tidal cycle, add 'option','tide' ")
    opt_str = 'season';
end

if ¬any(sav_str)
    disp("default: not saving the figure")
    disp("to save to plot, add 'save','yes' ")
    sav_str = 'no';
end

%% plot the seasonal variation of the plume area

if strcmpi(opt_str,'season')

    % get the new time series with only the number of the day in the year
    [¬,month,day,¬,¬,¬] = datevec(SPMGD.timeUTC);
    time_day = datenum(0,month,day,0,0,0);

    % make the boxchart figure
    figure('Position',[100 100 1200 400],'color','w')
    boxchart(time_day,SPMGD.area,'BoxWidth',1)


    % set limits for x and y axes
    xlim([0 366])
    ylim([0 2000])

    % set the xticklabels to be month strings using function axdate
    axdate(12);
    ax = gca;
    ax.XTickLabel{1} = 'Jan';
    set(ax, 'FontSize',14,'FontWeight','bold','linewi',2)
    grid on

    % add the title and ylabels
    ylabel('Area (km^2)')
```

```
    title('Seasonal Variation of Fraser River Plume Area',...
        'FontSize',20,'Fontwei','bold')

    % save the figure if sav_str is 'yes';
    if strcmpi(sav_str, 'yes')
        save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
        save_fname = 'area_seasonality';
        export_fig([save_dir, save_fname],'-png','-r400');
    end

end

%% area boxchart in a tidal cycle
% similar to the boxchart for the seasonal variation above

if any(strcmpi(opt_str,{'tide','river_tide_at_spm'}))

    switch opt_str

        case 'tide'
            data = SPMGD.area;
            y_lim = [0 2000];
            y_label = 'Area (km^2)';
            title_str = 'Tidal Variation of Fraser River Plume Area';
            save_fname = 'area_tidal';
        case 'river_tide_at_spm'
            data = SPMGD.RF;
            y_lim = [0 10000];
            y_label = 'River Flow (m^3/s))';
            title_str = ...
                'Fraser River Discharge Variation under a Tida Cycle (at SPM time)';
            save_fname = 'river_tide_at_spm';
    end

    figure('Position',[100 100 1200 400],'color','w')

    % devide tide lag minutes into 10-minute groups
    lag_new = round(SPMGD.lag_min * 10)/10;

    % making the boxchart and set y axis limits
    boxchart(lag_new, data,'boxwidth',0.1)
    ylim(y_lim);

    % find the meadian values for every 30 minutes in the tidal cycle, and
    % plot them as red circles
    [xg, yg] = consolidator(SPMGD.lag_min, data, @median, 0.5);
    hold on
    plot(xg, yg,'linestyle','none','marker','o','color','r','linewi',2)

    % calculate a polynomial fit of the red circles and plot it onto the
    % boxchart
    hold on
    p = polyfit(xg,yg,6);

    % if we want a more tide-like fit (starting point and ending point at
    % almost the same elevation, then we can copy some data at the end of
    % the cycle and put it before the beginning of the cycle, and vise
    % versa to add beging part of data on to the end.
    %
    %     p = polyfit(cat(1,xg(end-5:end)- 25.36,xg,xg(1:5) + 25.36), ...
    %         cat(1,yg(end-5:end),yg,yg(1:5)), 6);

    f = polyval(p, xg);
    plot(xg, f, '-','color','r','linewi',2);

    % add legend
    legend('6-minute binned Boxchart','30-minutes binned median',...
        'polynomial curve fitting')
```

```
144
145     % set axis parameters, add labels and titles
146     xlim([-7.6 17.7])
147     ax = gca;
148     ax.XTick = linspace(-6,16,12);
149     set(ax, 'FontSize',14,'FontWeight','bold','linewi',2)
150     ylabel(y_label)
151     xlabel('Tide Lag Hours')
152     title(title_str,...
153         'FontSize',20,'Fontwei','bold')
154
155     % save the figure if sav_str is 'yes';
156     if strcmpi(sav_str, 'yes')
157         save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
158         export_fig([save_dir, save_fname],'-png','-r400');
159     end
160
161 end
162
163 end
```

5.6 river_seasonality.m

See Figure. 5

```
1  function river_seasonality(fraser,varargin)
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % This function makes a plot of fraser river discharge seasonality from
4  % 2003 to 2019, and a mean river flow pattern was plot above all individual
5  % year line
6  %
7  % - 'save'
8  %   - 'yes': save the figure into the current directory
9  %   - 'no' or other string: do not save the figure
10 %
11 % Shumin Li, March 2021
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13
14 if strcmpi(varargin{1}, 'save')
15     sav_str = varargin{2};
16 end
17
18 % start making the plot
19 figure('Position', [100 100 1200 400],'color','w')
20
21 % Getting the year, month and day values of the river flow data
22 time_idx = fraser.mtime ≥ datenum(2003,1,1) & ...
23     fraser.mtime ≤ datenum(2019,12,31);
24 rf_time = fraser.mtime(time_idx);
25 rf_flow = fraser.flow(time_idx);
26 [year, month, day,¬,¬,¬] = datevec(rf_time);
27
28 % make a matrix to store all river flow data from 2003 to 2019
29 year_unique = unique(year);
30 rf_all = nan(numel(year_unique),365);
31 for i = 1: numel(year_unique)
32
33     year_idx = year == year_unique(i);
34     time_fake = datenum(0,month(year_idx), day(year_idx));
35     year_flow = rf_flow(year_idx);
36
37     plot(time_fake, year_flow)
38     hold on
39
40     rf_all(i,:) = year_flow(1:365);
```

```
    legend_str{i} = num2str(year_unique(i));
end

%% put the mean river flow pattern in a year

hold on
plot(1:365, nanmean(rf_all,1),'linewi',3,'color','k');

% set axis parameters
xlim([1 365]);
axdate(12);
ax = gca;
ax.XTickLabel{1} = 'Jan';
set(ax, 'FontSize',14,'FontWeight','bold','linewi',2)

% add the legend
legend_str{numel(year_unique)+1} = 'mean';
legend(legend_str,'Location','northeast','NumColumns',2)

grid on

% add ylabel and titles
ylabel('River Flow (m^3/s)')
title('Fraser River Discharge from 2003 to 2019',...
    'FontSize',20,'Fontwei','bold')

% save the figure if option of 'save' given as 'yes'
if strcmp(sav_str, 'yes')
    save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
    save_fname = 'river_seasonality';
    export_fig([save_dir, save_fname],'-png','-r400');
end

end
```

5.7 wind_seasonality.m

2 figures are plotted in this function, see Figure. 6 and Figure. 7

```
function wind_seasonality(sh,SPMGD,varargin)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function will make two figures of seasonality of wind type from Sand
% Heads (calm, northwesterly, southeasterly), one for all possible wind
% data from 2003 to 2019, and the second only for the times when SPM images
% are taken among those years
%
% - 'save'
%   - 'yes': save the figure into the current directory
%   - 'no' or other string: do not save the figure
%
% Shumin Li, March 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% making plots
month_str_short = {'Jan','Feb','Mar','Apr','May','Jun',...
    'Jul','Aug','Sep','Oct','Nov','Dec'};

if strcmpi(varargin{1}, 'save')
    sav_str = varargin{2};
end

% k = 1: making the first figure of wind seasonality of all wind data
% k = 2: making the second figure of wind seasonality for only SPM data
for k = 1:2

```

```
    figure('Position', [100 100 1200 400],'color','w')

    % find the data indices for calm wind, northwesterly and southeasterly

    if k == 1
        time_idx = sh.wtime ≥ datenum(2003,1,1) & ...
            sh.wtime ≤ datenum(2019,12,31);
        w_time = sh.wtime(time_idx);
        w_spd = sh.wspd(time_idx);
        w_dir = sh.wdir(time_idx);

    elseif k == 2
        w_time = SPMGD.timeUTC;
        w_spd = SPMGD.wspd;
        w_dir = SPMGD.wdir;

    end

    calm_idx = w_spd < 4;
    strong_nw_idx = w_dir ≥ 270 & w_dir ≤ 360 & ¬calm_idx;
    strong_se_idx = w_dir ≥ 90 & w_dir ≤ 180 & ¬calm_idx;


    % get the month of each recorded data time
    [¬, month, ¬, ¬ ,¬,¬] = datevec(w_time);
    time_month = datenum(0, month, 0);


    clear calm_n nw_n se_n

    % get the number of each type of wind in each month
    time_unique = unique(time_month);
    for i = 1:numel(time_unique)

        loop_idx = time_month == time_unique(i);
        calm_n(i) = sum (loop_idx & calm_idx);
        nw_n(i) = sum (loop_idx & strong_nw_idx);
        se_n(i) = sum (loop_idx & strong_se_idx);

    end

    % plot the lines with markers for each type of wind
    if k == 2
        plot(1:numel(time_unique), (calm_n + nw_n +se_n)/17,...
            'Marker','v',...
            'color','#0072BD','linesty','-','linewi',2,'MarkerSize',15)
        hold on
    end
    plot(1:numel(time_unique), calm_n/17,'Marker','o',...
        'color','#D95319','linesty','-','linewi',2,'MarkerSize',15)
    hold on
    plot(1:numel(time_unique), nw_n/17,'Marker','s',...
        'color','#EDB120','linesty','-','linewi',2,'MarkerSize',15)
    hold on
    plot(1:numel(time_unique), se_n/17,'Marker','^',...
        'color','#7E2F8E','linesty','-','linewi',2,'MarkerSize',15)

    % set the axis parameters
    xlim([0.5 12.5]);
    ax = gca;
    ax.XTick = (1:12);
    ax.XTickLabel = month_str_short;
    set(ax, 'FontSize',14,'FontWeight','bold','linewi',2)

    grid on


    % add labels, legends and titles
```

```
    if k == 1
        ylabel('Averaged Number of Data')
        legend('Calm (< 4 m/s)','Northwesterly','Southeasterly')
        title('Seasonality of Wind Types at Sand Heads from 2003 to 2019',...
            'FontSize',20,'FontWeight','bold')
        save_fname = 'wind_seasonality';

    elseif k == 2
        ylabel('Averaged Number of Images')
        legend('All Images','Calm (< 4 m/s)','Northwesterly','Southeasterly')
        title('Seasonality of Wind Types at Sand Heads for Selected SPM Images',...
            'FontSize',20,'FontWeight','bold')
        save_fname = 'wind_seasonality_spm';
    end

    if strcmpi(sav_str, 'yes')
        save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
        export_fig([save_dir, save_fname],'-png','-r400');
    end

end


end
```

5.8 area_rf_relation.m

See Figure. 10

```
function area_rf_relation(SPMGD, varargin)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function simply makes a figure showing the correlation of river
% discharge and plume area. with the option to save the figure or not
%
% - 'save'
%   - 'yes': save the figure into the current directory
%   - 'no' or other string: do not save the figure
%
% Shumin Li, March 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% plot the figure

varargin = {'save','no'};
if strcmpi(varargin{1}, 'save')
    sav_str = varargin{2};
end


figure('Position', [100 100 1200 400],'color','w')
plot(SPMGD.RF, SPMGD.area, 'bx')
ylim([0 2000]);

% make a consolidated data with the torarence of 200 and function of median
[xg, yg] = consolidator(SPMGD.RF, SPMGD.area, @median, 200);

% plot the group medians in red circle
hold on
plot(xg, yg,'linestyle','none','marker','o','color','r','linewi',2)

% add a linear fit curve
hold on
p = polyfit(xg, yg, 1);
f = polyval(p, xg);
plot(xg, f, '-','color','r','linewi',2);
```

```

% add legend
legend('Individual Data Points','binned medians',...
    'linear fitting curve')

% set axes parameters
xlim([min(SPMGD.RF) max(SPMGD.RF)])
ax = gca;
set(ax, 'FontSize',14,'FontWeight','bold','linewi',2)

% add xlabel, ylabel and titles
xlabel('River Flow (m^3/s)')
ylabel('Area (km^2)')
title('Correlation of Fraser River Discharge and Plume Area',...
    'FontSize',20,'Fontwei','bold')

% save the figure if input option 'save' gives 'yes'
if strcmpi(sav_str,'yes')
    save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
    save_fname = 'area_rf_correlation';
    export_fig([save_dir, save_fname],'-png','-r400');
end

end
```

5.9 tide_spm_histogram.m

See Figure. 13

```
function tide_spm_histogram(SPMGD, varargin)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This fuction simply makes a figure of 5*5 subplots showing the valid SPM
% image distributed in each month under a tidal cycle. mainly demenstrating
% some bias in analyzing the MODIS SPM images.
%
% - 'save'
%   - 'yes': export the figure into the project directory
%   - 'no' (or any other string): do not save the figure
%
% Shumin Li, March 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if strcmpi(varargin{1}, 'save')
    sav_str = varargin{2};
end

%%
marg_w = [0.08 0.08];
fig_pos = [100 100 650 600];
row = 5;
col = 5;
gap = [0.005 0.005];
marg_h = [0.05 0.09];
sgt_FS = 20;

figure('color','w','Position',fig_pos);
ha = tight_subplot(row,col,gap,marg_h,marg_w);

[~,spm_month,~,~,~,~] = datevec(SPMGD.timePDT);
better_index = SPMGD.percent_valid > 0.4 & SPMGD.center_valid > 0.5;
sgtitle('Histograms of the No. SPM images in a tidal cycle',...
    'FontSize',sgt_FS,'FontWeight','bold');


```

```

for i = 1:row*col

axes(ha(i));

tide_idx = SPMGD.LowTide < 2.1;

j = i - 8;
lag_2h =  SPMGD.lag_min > j-1 & SPMGD.lag_min < j+1;

% Count tide lag hours from 17h to 18h into "-7h" segment,
% and also count the -8h to -7h period as part of "17h" segment
if j == -7
    lag_2h =  SPMGD.lag_min > 17 | SPMGD.lag_min < -6;
elseif j == 17
    lag_2h =  SPMGD.lag_min > 16 | SPMGD.lag_min < -7;
end

% put the earlier creterion together
data_idx =  lag_2h & better_index & tide_idx;

hist_edges = linspace(0.5, 12.5, 13);
histogram(spm_month(data_idx),hist_edges)

ylim([0 90]);

ax = gca;
ax.XTick = (1:12);
ax.XTickLabel = {'Jan','','','Apr','','','Jul','','','Oct','',''};
ax.YTick = (0:10:90);
ax.YTickLabel = {'0','','20','','40','','60','','80',''};
grid on
text(0.5, 80,['h = ', num2str(j)],'FontSize',12,'FontWeight',"bold")

if i ≤ (row-1)*col
    set(gca,'XTickLabel',[])
end

if ¬(mod(i,col) == 1)
    set(gca,'YtickLabels',[])
else
    ylabel('n (images)','FontSize',12,'FontWeight',"bold")
end

set(gca, 'FontWeight',"bold")
end

if strcmpi(sav_str, 'yes')
    save_dir = '/users/shuminli/Nextcloud/study_prjs/EOSC584_Project/';
    save_fname = 'SPM_histogram';
    export_fig([save_dir, save_fname],'-png','-r400');
end


end
```

6 Appendix

This section contains some codes which are not directly related to plotting, but important for the data processing.

6.1 build_SPMindex.m

```
function build_SPMindex(varargin)
%% build the struct of SPMGD.mat data struct from scratch
% 'save': save the SPMGD.mat into the given folder
%        - 'no': don't save
%        - a directory with file names
%             e.g. '/users/shuminli/Nextcloud/study_prjs/front_propagation/SPMGD2.mat'
%             then the SPMGD variable will be saved as SPMGD2.mat in the
%             given directory.
%
% Shumin Li, February 2021

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% loading necessary data and directories

% load wind data from sand heads
sh=load('/users/shuminli/Nextcloud/data/wind/sand.mat');

% load a reprocessed wind data in the Strait of Georgia using EOF
sh_re = load('/users/shuminli/Nextcloud/data/wind/SoGwinds.mat');

% load fraser river discharge data
load /users/shuminli/Nextcloud/data/river/river_new/fraser.mat;

% spm_plume_ginput.mat conatians two matrix:
% spm_plume_input and spm_center_input:
%
% - spm_plume_input: a polygon inside the strait of georgia, (where the plume
%   is likely to appear)
% - spm_center_input: a ploygon near the river mouth, where the core of the
%   plume is most like to be.
%
% Then we can use these two box as a standard to automatically picking up
% the possible good spm images (e.g. the coverage in the strait is over 50%
% or the coverage in the plume center is over 60%)
%
% And spm_in and spm_center_in are the indices inside each polygon
load('/users/shuminli/Nextcloud/study_prjs/front_propagation/spm_plume_ginput.mat');
spm_in = inpolygon(gridLon,gridLat,spm_plume_ginput(:,1),spm_plume_ginput(:,2));
spm_centerin = inpolygon(gridLon,gridLat,spm_center_ginput(:,1),spm_center_ginput(:,2));

% Making the long/lat grid of SPM images
latMin=48.5;
latMax=50;
lonMin=-124.5;
lonMax=-122.;
[gridLon,gridLat] = meshgrid([lonMin:0.00343:lonMax], [latMin:0.002246:latMax]);

%% begin to build the struct file

% This folder contains 11,000+ individual .mat file for daily spm pictures
% (twice a day from 2003 to 2019). All filenames are like
% 'A2018144204500.mat', where 'A' mean MODIS Aqua Satellite, '2018' is
% year, '144' is the day of the year, '204500' are the hour, minutes and
% seconds in UTC
SPM_folder = '/users/shuminli/Nextcloud/data/SPM/SPM_mat/';
SPM_dir = dir([SPM_folder, '*.mat']);

% extract the time information from the filenames
```

```
SPM_name_num = nan(size(SPM_dir));
for i = 1:numel(SPM_name_num)
    SPM_name_num(i) = str2num(SPM_dir(i).name(4:12));
end


% build the date number and date strings
SPM_datenum = nan(size(SPM_dir));
SPM_datestr = cell(size(SPM_dir));
% 3 seconds for this loop
for i = 1:length(SPM_name_num)
    yy = str2num(SPM_dir(i).name(2:5));
    % ddd2mmdd is a simple fuction that converts 3-digit day to month and
    % day, the function of ddd2mmdd is as follows:
    %-------------------------------------------------
    % function [mm,dd] = ddd2mmdd(year, ddd)
    % v = datevec(datenum(year, ones(size(year)), ddd));
    % mm = v(:,2); dd = v(:,3);
    %-------------------------------------------------
    [mm,dd] = ddd2mmdd(yy,str2num(SPM_dir(i).name(6:8)));
    if mm < 10 && dd<10
        SPM_datestr{i} = [num2str(yy),'-0',num2str(mm),'-0',num2str(dd),' ...
            ',SPM_dir(i).name(9:10),':',SPM_dir(i).name(11:12),':',SPM_dir(i).name(13:14)];
    elseif mm < 10 && dd ≥ 10
        SPM_datestr{i} = [num2str(yy),'-0',num2str(mm),'-',num2str(dd),' ...
            ',SPM_dir(i).name(9:10),':',SPM_dir(i).name(11:12),':',SPM_dir(i).name(13:14)];
    elseif mm ≥ 10 && dd < 10
        SPM_datestr{i} = [num2str(yy),'-',num2str(mm),'-0',num2str(dd),' ...
            ',SPM_dir(i).name(9:10),':',SPM_dir(i).name(11:12),':',SPM_dir(i).name(13:14)];
    else
        SPM_datestr{i} = [num2str(yy),'-',num2str(mm),'-',num2str(dd),' ...
            ',SPM_dir(i).name(9:10),':',SPM_dir(i).name(11:12),':',SPM_dir(i).name(13:14)];
    end
    formatIn = 'yyyy-mm-dd HH:MM:SS';
    SPM_datenum(i) = datenum(SPM_datestr{i},formatIn);
end

%% making the SPM struct for good days from 03 to 19

% generate the hourly tidal elevation from 2003 to 2019 from t_xtide
% function (about 1 sencond to run)
hour_time = SPM_datenum(1)-2:1/24:max(SPM_datenum(:))+2; % UTC time
hour_tide = t_xtide('Point Atkinson (2)',hour_time-8/24);


% building the SPMGD struct (about 3 minutes to run the loop)
SPMGD.filename = {};
j = 1;
for i = 1:numel(SPM_dir)
    name = SPM_dir(i).name;

    % 'A2018144204500.mat' is a bad data which cases some unknown error, so
    % we just skip it
    if strcmp(name,'A2018144204500.mat')
        continue
    end

    % load data
    data = load([SPM_folder,SPM_dir(i).name]);
    data_mat = data.data.gridSPM;

    % calculate the percentage of valida data within the strait and within
    % the center of the plume region
    percent_valid = sum(isfinite(data_mat(spm_in)),'all')/sum(spm_in(:));
    center_valid = sum(isfinite(data_mat(spm_centerin)),'all')/sum(spm_centerin(:));
    clear data


```

```
    if percent_valid ≥ 0.5 || center_valid ≥ 0.6
        % filenames, values, time in UTC and PDT, and timestring in PDT
        SPMGD.filename{j} = name;
        SPMGD.val(:,:,j) = data_mat;
        SPMGD.timeUTC(j) = SPM_datenum(i);
        SPMGD.timePDT(j) = SPM_datenum(i) - 8/24;
        SPMGD.timePDTstr{j} = datestr(SPM_datenum(i) - 8/24);

        % find the index of corresponding wind data and river flow data
        [w1, n1] = min(abs(SPM_datenum(i) - sh_re.it));
        [w2, n2] = min(abs(SPM_datenum(i) - sh.wtime));
        [ff, rr] = min(abs(SPM_datenum(i) - fraser.mtime));



        % fill in the wind speed and direction data. (Primarily use the
        % origional data from sandheads (i.e. sh), and use the EOF
        % reprocessed data (i.e. sh_re) as a compensate.
        if w1 < 1/24
            SPMGD.wspd(j) = abs(sh_re.TS(n1,1));
            wd = atan2d(imag(sh_re.TS(n1,1)),real(sh_re.TS(n1,1)));
            if wd< 0
                wd = wd + 360;
            end
            SPMGD.wdir(j) = wd;

        elseif w2 < 1/24
            SPMGD.wspd(j) = sh.wspd(n2);
            SPMGD.wdir(j) = sh.wdir(n2);
        else
            SPMGD.wspd(j) = nan;
            SPMGD.wdir(j) = nan;
        end

        % fill in the river discharge data
        if ff < 1
            SPMGD.RF(j) = fraser.flow(rr);
        else
            SPMGD.RF(j) = nan;
        end

        % calcute and record the tidal lag hours and other tidal
        % information using function tide_hrs
        [SPMGD.lag_min(j), SPMGD.LowTide(j),...
            SPMGD.LowTideTimePDT(j),SPMGD.TideVal(j,:)] = ...
            tide_hrs(hour_tide, hour_time - 8/24,SPM_datenum(i) - 8/24,'no');
        SPMGD.lag_h(j) = round(SPMGD.lag_min(j));
        SPMGD.percent_valid(j) = percent_valid;
        SPMGD.center_valid(j) = center_valid;
        j = j+1;

    end
end

SPMGD.lag_h(SPMGD.lag_h == -8) = -7;
SPMGD.lag_h(SPMGD.lag_h == 18) = 17;


% calculating the approximate area of a pixel at the center plume:
% longitude -123.3 at gridLon(:,351);
% latitude 49.1 at gridLat(272,:);
A = [272, 351];

% 4 edge points of the grid in the central plume area
B(1) = m_idist(gridLon(A(1),A(2)), gridLat(A(1),A(2)),...
    gridLon(A(1)+1,A(2)),gridLat(A(1)+1,A(2)));
B(2) = m_idist(gridLon(A(1),A(2)), gridLat(A(1),A(2)),...
    gridLon(A(1),A(2)+1),gridLat(A(1),A(2)+1));
```

```
B(3) = m_idist(gridLon(A(1)+1,A(2)), gridLat(A(1)+1,A(2)),...
    gridLon(A(1)+1,A(2)+1),gridLat(A(1)+1,A(2)+1));
B(4) = m_idist(gridLon(A(1),A(2)+1), gridLat(A(1),A(2)+1),...
    gridLon(A(1)+1,A(2)+1),gridLat(A(1)+1,A(2)+1));

% the four side length are almost identical, so it is almost a perfect
% square, so we will calculate the area of the pixel (parea) as a square
C = nanmean(B);
parea = C*C;

% calculate area of the plume region, the averaged spm level inside this
% region and returen the new SPMGD struct.
for i = 1: numel(SPMGD.filename)
    spm_data = SPMGD.val(:,:,i);
    npixel = sum(spm_data > 2, 'all');
    SPMGD.area(i) = parea * npixel / 1000000;
    SPMGD.plume_ave(i) = nanmean(spm_data(spm_data > 2));
end

%% save the .mat file depending on input

save_idx=find(strcmpi('save',varargin),1);
if isempty(save_idx) || strncmpi(varargin{save_idx+1},'no',1)

else
    save_dir = varargin{save_idx+1};
    save(save_dir,'SPMGD','-v7.3');
end

end
```

6.2 tide_hrs.m

This function is used in Sec. 5.3, 5.4,6.1

```
function  [H, L, T, V]= tide_hrs(tideh, tidet, tt, is_fig)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function mainly help us to find the tdial lag hours relative to the
% lower low tide of the day
% it will export the following variables:
%
% - H: the thide lag hours
% - L: the low tide height
% - T: the low tide time
% - V: the array of the tidal elevation for 2 days before/after the given
% time
%
% it requires the following input:
% - tideh: tidal elevation vector for a certain amount of time
% - tidet: the times corresponding to tideh
% - tt: the time you putin to get [H, L, T, V].
% - is_fig: 'yes' for ploting an example figure, 'no' for not plotting
%
% Shumin Li, February 2021
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for k = 1:numel(tt)
    t = tt(k);

    timest = t - 1.5 ;
    timeed = t + 1.5 ;

    timei = find(tidet ≥ timest & tidet ≤ timeed);

    % interp the input hourly data into per-minite data
```

```
    timen = timest:1/24/60:timeed;
    tiden = interp1(tidet(timei),tideh(timei),timen,'spline');

    % finding the regional minimum points in the curve
    n = length(timen);

    c = [];
    d = [];

    for ii = 2:n-1
        if tiden(ii) - tiden(ii-1) < 0 && tiden(ii) - tiden(ii+1) < 0
            c = [c; ii];
            d = [d;tiden(ii)];
        end

    end

    % sort these regional minimum points from low to high, as long as
    % lowest point falls in the range of [-7.6 17.6] hrs, the choose it as
    % the lower-low tide point of the day, and output its tide lag time in
    % minutes, height, time, an array of it daily tidal elevation for plot

    [B,I] = sort(d);
    time_diff = 24*(t - timen(c));

    for i = 1:numel(I)
        if -7.6 < time_diff(I(i)) && time_diff(I(i)) < 17.6
            H(k) = time_diff(I(i));
            L(k) = B(i);
            T(k) = timen(c(I(i)));
            [¬,bb] = min(abs(timen - T(k)));
            if numel(tt) > 1
                V(k,:) = tiden(bb - 7.5*60: bb+17.5*60);
            else
                V = tiden(bb - 7.5*60: bb+17.5*60);
            end

            break
        end
    end

end

% if the last input from the function is 'yes', then plot an example figure
if strcmp(is_fig,'yes') && numel(tt) == 1
    figure('Position',[100 100 800 200])

    clf
    plot(timen, tiden,'-r','linewi',2)
    xline(t,'linewi',2,'color','k')
    hold on
    xline(T,'color','k','linewi',2,'linest','--')
    axdate(12);
    hold on
    set(gca, 'FontSize',14)
    plot(timen(c),d, 'go','linewi',2)
    title(['Tide Lag = ',num2str(H(1)),' (h); Low Tide Height = ', num2str(L(1)),' (m); ', ...
        'Low Tide Time = ', datestr(T(1))])
end

end
```

7 Acknowledgements

Finishing this project, I would like to thank Professor Phillippe Tertell for introducing this great course and many useful Matlab functions, thank the help from Professor Rich Pawlowich for his help and his valuable toolboxes: m_map toolbox, t_tide toolbox, axdate function. Thank Research Assistant Katia Stankov for processing the raw MODIS and CODAR data. Thank Florian Knorn for the mcode.sty so that I can put Matlab codes nicely and easily into the Latex file.

References

Pawlowicz, R., R., D. C., Halverson, M., Devred, E., and Johannessen, S. (2017). Advection, surface area, and sediment load of the fraser river plume under variable wind and river forcing. *Atmosphere-Ocean*, 55(4-5):293–313.