Wave field and rip current dynamics in a laboratory setting

Christine Baker and Emma Nuss

Abstract

The alongshore length scales of breaking waves along a beach in a laboratory wave basin are estimated. The FFT analysis is applied to transects of pixel intensity in images and the sea-surface elevation from stereo reconstructions and model output of the sea-surface. The spectrum are filtered to remove high wavenumber variance and averaged over a trial to compute a robust spectra representing the breaking wave field characteristics.

1. Introduction

Wave evolution and breaking drives dynamics occur in the nearshore, the region of water along the coast consisting of the surf zone (the edge of the beach to the seaward extent of wave breaking) and the inner shelf (the offshore edge of the surf zone to about 1.5 km offshore). Wave field characteristics can define current circulation patterns, nearshore sediment transport, and run up on beaches. Irregular wave fields are commonly analyzed in spectral space, which can be used to estimate bulk wave characteristics. Although, the Fast Fourier Transform (FFT) is a well established method for estimating bulk wave characteristics such as wave height, period, angle, and spread, the FFT routine can also be used to describe other wave and current properties such as fluctuations in surfzone currents, low-frequency wave properties (*i.e.,* infragravity waves), and the length scale of breaking wave crests.

This project focuses on the estimation of the alongshore (distance parallel to the beach) length scales of breaking waves in a laboratory wave basin by using data collected during the experiment and model output. During wave breaking, vorticity is injected at the edges of wave breaking crests. The wave-breaking generated eddies can coalesce within the surf zone (similar to 2D turbulence) and eventually lead to a rip current (fast moving offshore-directed jet of water) ejection. Due to the reliance of rip current formation on eddies formed during wave breaking, it is important to establish the length scales of the breaking crests.

Spectral analysis can be performed on alongshore transects of the sea-surface elevation to estimate the distribution of the length scales of breaking waves in the surf zone. To obtain a robust estimate of the wave alongshore length scales, a FFT routine is used to compute a spectrum at each data sampling time step, the small lengthscales are removed by applying a filter, and the spectra computed over the span of a trial are averaged. The directional properties of the waves are related to the alongshore length scales of the breaking crests (*e.g.,* higher wave directional spread can lead to shorter alongshore length scales of the wave crests, and vice versa is true for unidirectional waves). The theoretical background for FFT analysis and filtering is presented in Section 2 and the algorithm implementation and development for laboratory data and model output are detailed in Section 3. The computational results are discussed in Section 4 and conclusions are summarized in Section 5. The python/MATLAB functions used and the full analysis codes are presented in Appendix A and B, respectively.

2. Theoretical Background

Fourier introduced a notion that a function, $f(x)$, can be represented by a trigonometric series of sines and consines, such that for $-\pi < x \leq \pi$:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx) \tag{2.1}$$

which produces 2π-periodic functions. The Fourier Transform defined over an entire line $-\infty \leq x \leq \infty$ is defined as:

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \tag{2.2}$$

and the inverse Fourier Transform is defined as:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{2.3}$$

The FFT, a routine developed to perform forward and backward Fourier transforms, requires $O(N \log N)$ operation count to solve a system (Kutz, 2013). The FFT routine has excellent accuracy properties and transforms over a finite interval interval $-L \leq x \leq L$ with periodic boundary conditions and are discretized into 2^n points. High-, low-, and band-pass filtering, noise attenuation via frequency filtering, can be applied to data by applying a filter to Fourier transformed data in frequency-space. Step-wise filters (*i.e.,* box-car filters) are simple but generate ringing in frequency-space due to their sharp edges, whereas smoothed filters can reduce ringing. Filtering can significantly remove noise and improve the detection of a signal (Kutz, 2013).

Another technique to remove white-noise in a time series is to compute the FFT of small windows of data and average the spectra. If the signal is present throughout a data record, this will average out the white-noise in a signal and retain signals that are not noise when appropriate data lengths are chosen. There are drawbacks to this approach such as the requirement to have a long enough data set to have multiple realizations of the signal of interest, *i.e.,* low-frequency signals in the data may be missed if the subsection length of the data are not long enough to resolve these frequencies.

3. Algorithm Implementation and Development

The alongshore wavenumber spectra representing the alongshore lengthscales of breaking wave crests is computed for stereo reconstructions and imagery collected during laboratory experiments and wave-resolved model output from simulations with a configuration representing the laboratory experiments. These spectral estimates are computed as follows:

1. For a snapshot in time, alongshore transect of the sea-surface elevation from the wave breaking region is selected from the data/model output
2. The transect is demeaned and detrended
3. The wavenumbers are defined and rescaled by $2\pi/L$, because the FFT assumes a 2π periodic signal.
4. A FFT is used to compute the wavenumber spectrum of the data.
5. The spectrum is filtered to remove small length scale variance that is not of interest
6. The spectra computed at each time step for the trial is averaged

3.1. Laboratory Experiment Data Analysis

Laboratory experiments to study rip currents were ran in a large-scale wave basin in the O.H. Hindsdale Wave Research Laboratory at Oregon State University. Data collected during these laboratory experiments included in situ and remotely sensed measurements, including camera imagery and LiDAR surveys. For this project, the alongshore wavenumber spectra of the sea-surface elevation is computed from pixel intensity from a camera mounted to the ceiling of the tank (Figure 1a) and stereo reconstructions of the sea-surface (digital elevation models (DEMs) produced from matching points in three camera images, Figure 1b)

Directionally spread (short alongshore length scale, Figure 2a,b) and unidirectional (long alongshore length scale, Figure 2c,d) waves were produced to investigate how these wave characteristics impact the formation and evolution of rip currents in the laboratory environment. The alongshore transects from the stereo reconstructions are of the sea-surface elevation, which directly provides an estimate of the variance of the sea-surface at different length scales. For the camera imagery, the transects of pixels is a proxy for the breaking crest length as the breaking crests appear white (high values of pixel intensity) and the regions where waves are not breaking appears darker (low values of pixel intensity).

The camera images were originally taken at an oblique angle. To extract a specific cross-shore location (x), the images were rectified into the wave basin coordinate system. The rectified image was re-sampled at a constant resolution (cross- and alongshore resolution of 0.01 m). The stereo reconstructions were computed using a software called Metashape and the DEMs were output at the same resolution as the re-sampled images. The alongshore transects from the images and stereo reconstructions (derived from 3 camera images) are selected at 8 Hz (camera sampling rate) for all alongshore locations at the cross-shore position of $x = 30$ m (near the breaking region). The camera image transect is turned into black and white, detrended, and normalized by 256 (maximum pixel intensity).

The spectrum is computed using FFT methods at this sampling rate, and the wavenumbers (k) are computed and re-scaled. A filter with a cutoff of $k/2\pi$ of 1 1/m (*i.e.,* length scales of 1 m) with three transitional band samples (0.7059, 0.2363, and 0.0225, Elgar (1988)) is multiplied by each spectrum to remove any small-scale variance that smaller in length (higher k) than expected for the laboratory waves. The spectra computed over 10 min of data (4800 spectra) are averaged to produce an wave-field representative estimate spectra of the wave length scales. This analysis was performed on three wave conditions with the same wave height, period, and water level, and differing wave directional spread ($\sigma = 0, 20, 40°$).

3.2. Model

A phase-resolving numerical wave model, FUNWAVE-TVD, was setup to simulate similar conditions to the laboratory experiments described above. FUNWAVE-TVD is a numerical model that solves the nonlinear Boussinesq equations and has an internal wavemaker that can produce complex wave fields (Shi et al., 2012). To produce similar conditions to the laboratory experiments, the model was setup with an analogous cross-shore bathymetry profile and was run with 0.2 meter waves with a directional spread of 20° ($\sigma = 20°$).

The model was run for 1 hour of simulated time, allowing for sufficient time for the wave field to develop. Model outputs included horizontal velocity and sea surface height at every grid point with spacing of 5 centimeters in the cross-shore and 10 centimeters in the alongshore output every 0.2 seconds. Model states were output into text files, these text files were compiled into three 4-D arrays of horizontal velocities (u,v) and sea surface elevation.

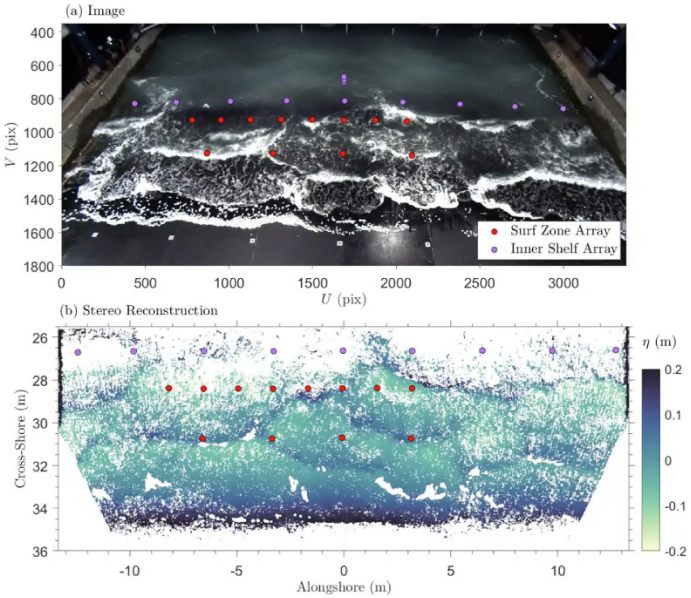

Figure 1: A snapshot of an (a) image collected during the laboratory experiment and (b) the stereo reconstruct of the sea-surface elevation (color contours). The experiments included of two arrays of in situ instruments measuring the pressure and velocity in the water column (magenta and red circles).

To compute a wavenumber spectrum from the model output, sea surface elevation from a single cross-shore location (x) was concatenated into a 2-D data array for each time step of model output (Figure 3). This process was done for cross-shore positions of, 1) on the offshore edge of the surfzone, 2) mid-way through the surfzone, and 3) near the shoreline in the peak breaking zone. For each time record, a wavenumber spectrum of the alongshore transect of sea surface height was computed using the FFT method described above. These spectra (17500 records) were averaged together to produce a single wavenumber spectrum for the three cross-shore positions (Figure 5).

4. Computational Results

4.1. Laboratory Experiment Data

The alongshore length-scales of the breaking wave crests can be inferred from the alongshore spectra of the sea-surface elevation (Figure 4a) and the pixel intensity (Figure 4b) for three wave conditions with differing wave directional spread ($\sigma = 0, 20, 40°$).

For the alongshore wavenumber spectra of the sea-surface elevation computed from the stereo reconstructions and images, the variance for $\sigma = 0°$ is much lower at all length scales than the cases with more directional spread. This is expected since, theoretically, waves that are unidirectional was be uniform in length in the alongshore. The alongshore transect is demean and detrended before applying the FFT, and thus, for a uniform wave, there would be no variance at any length scales. However, there is *some* alongshore variability in the wave field (see Figure 2d) which may be due to a tank side wall effect or imperfect wave production at the wave maker.

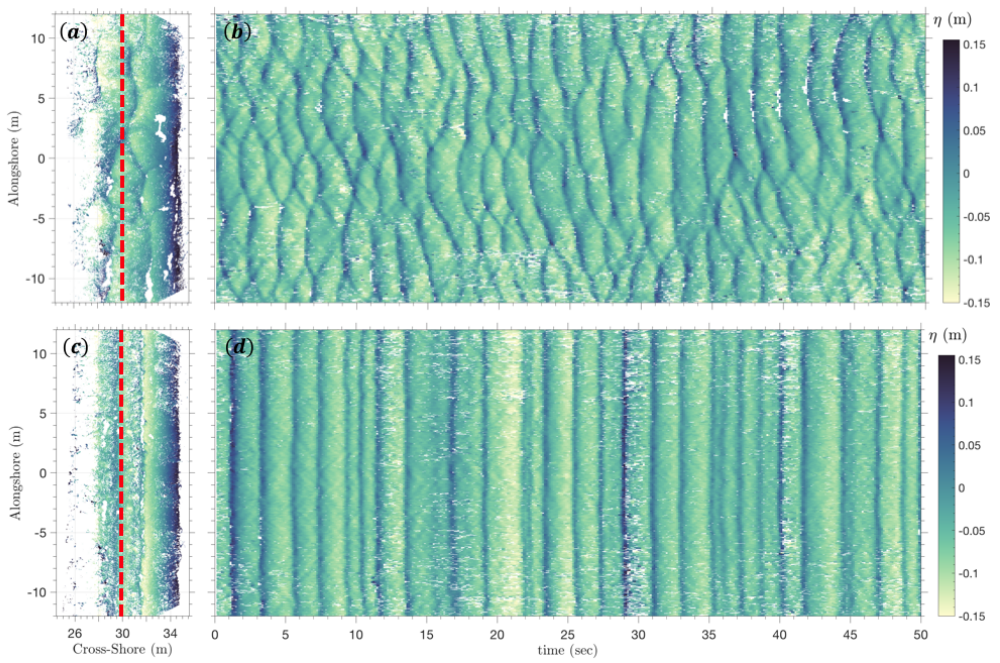

Figure 2: The stereo reconstructed sea-surface elevation (color contours) from an experiment with (a,b) directionally spread waves and (c,d) unidirectional waves. Snapshots of the stereo reconstruction from both trails show the location where the alongshore transect is taken (cross-shore coordinate, $x = 30$ m) (a,c red dashed line). The reconstructed sea-surface at the alongshore transect is shown in time (b,d) for 50 sec of the experiment.

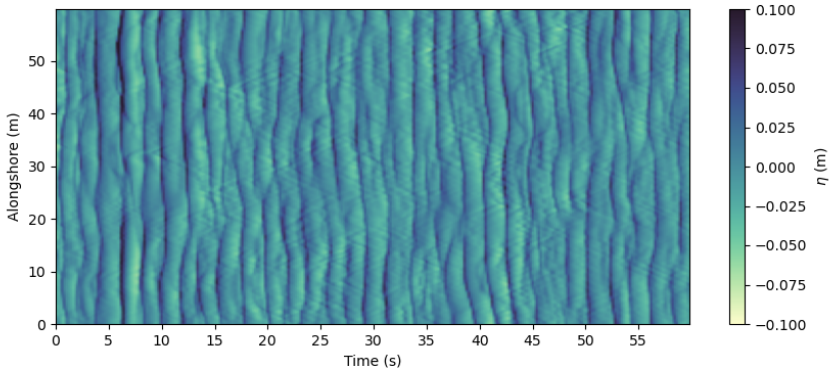

Figure 3: Sea-surface elevation from a numerical simulation taken at a cross-shore transect in the surfzone for 60 seconds of the simulation period.

Additionally, the stereo reconstructions are imperfect and have some matched points that are not physical. However, these issues with the stereo will likely impact the high wavenumbers (small length scales) that are removed by the filter. The reasoning for the irregular peaks in the image spectra for $\sigma = 0°$ is unknown.

The two trials with non-zero directional spread ($\sigma = 20, 40°$) have similar variance magnitudes with a small shift in the peak length scale from 20 m for the $\sigma = 20°$ waves to 13 m for the $\sigma = 40°$ waves (Figure 4a, peak in solid line is at higher wavenumbers than the dashed line). Theoretically, as the directional spread increases, the alongshore length scales of the breaking waves should decrease, which is supported by the results from this analysis.

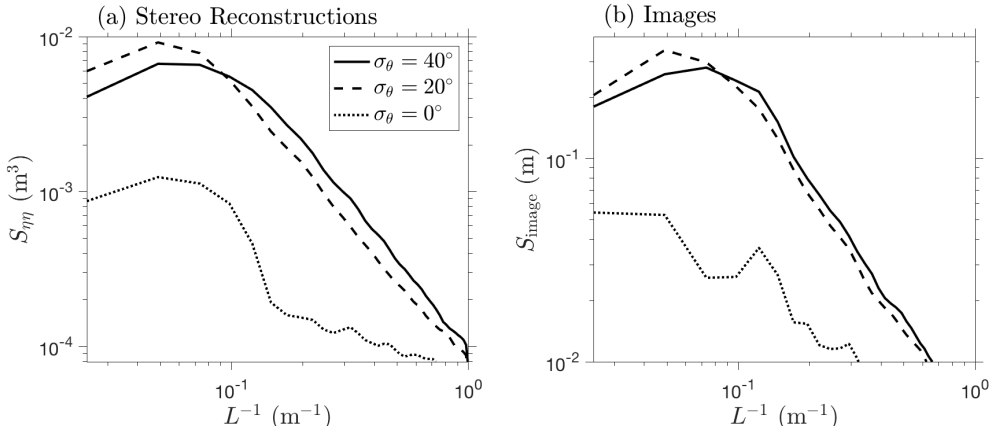

Figure 4: The average alongshore wavenumber (a) sea-surface elevation spectra ($S_{\eta\eta}$) and (b) normalized pixel intensity spectra (S_{im}) as a function of inverse length scale ($L^{-1} = k/2\pi$) for wave conditions with a directional spread (σ) of 0° (solid), 20° (dashed), and 40° (dotted).

4.2. Model

The alongshore length scales of the wave crests at three cross-shore locations within the surfzone are inferred from the alongshore wavenumber spectra of sea surface elevation (Figure 5). The wavenumber spectra can be compared to the laboratory spectra to understand how similar the wave conditions are between both environments. Model spectra are similar, but not consistent with laboratory spectra. This discrepancy suggests that model simulations do not exactly capture all of the dynamics observed in the laboratory setting. Likely model parameters including wave height, bottom friction, and others can be tuned to be similar to the observed wave conditions.

All three spectra show approximately the same energy profile at small wavenumbers, but differ at larger wavenumbers. This pattern shows that at each cross-shore location there is a similar amount of variance for longer length-scales, but not for shorter length-scales. For shorter length-scales (larger wavenumbers), there is higher variance in the breaking region than the mid-region or offshore edge. These spectra suggest that as a directionally-spread wave-field shoals, wave breaking and wave-breaking generated eddies act to move energy from larger scales to smaller scales.

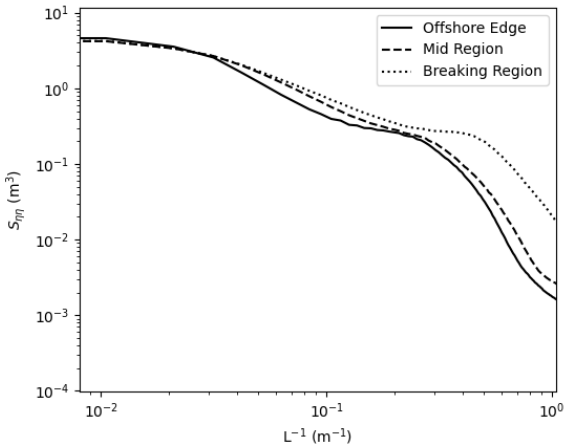

Figure 5: The average alongshore wavenumber sea-surface elevation spectra ($S_{\eta\eta}$) as a function of inverse length scale ($L^{-1} = k/2\pi$) for wave conditions with a directional spread (σ) of 20° in the offshore edge of the surfzone (solid), middle of the surfzone (dashed), and peak breaking zone (dotted).

5. Summary and Conclusions

Fourier transforms are an important data analysis and mathematical tool for converting data between the time or spatial domain and the frequency or wavenumber domain. For laboratory and modeling experiments aiming to understand wave evolution and surfzone dynamics, converting spatial information into wavenumber space is a valuable tool. In this project we apply the theory of Fourier transforms to compute wavenumber spectra from alongshore transects of sea-surface elevation data from a laboratory setting and a simulated numerical setting. The wavenumber spectra provides information about the distribution of length-scales present in a series of laboratory experiemnts, as well as a numerical simulation. Wavenumber spectra were computed from sea-surface elevation data from the laboratory data for three wave cases ($\sigma = 0°, 20°, 40°$) and from pixel intensity. Spectra show that higher variance is seen in larger wavenumbers (smaller wave crests) for greater directional spread (σ). From the model output, wavenumber spectra were computed from sea-surface elevation data for three cross-shore positions in the surfzone. Spectra show that variance increases at smaller scales (larger wavenumbers) moving from offshore to onshore. Comparison between laboratory and model spectra provide insight into model simulation accuracy and ability to reproduce a similar wave field. Overall, Fourier transforms provide a useful tool for understanding the distribution of energy in wavenumber space and aiding in investigation of surfzone dynamics.

References

J. N. Kutz, Data-driven modeling & scientific computation: methods for complex systems & big data, Oxford University Press, 2013.

S. Elgar, Comment on "fourier transform filtering: A cautionary note" by a.m. g. forbes, Journal of Geophysical Research 93 (1988) 15,755–15,756.

F. Shi, J. T. Kirby, J. C. Harris, J. D. Geiman, S. T. Grilli, A high-order adaptive time-stepping tvd solver for boussinesq modeling of breaking waves and coastal inundation, Ocean Modelling 43 (2012) 36–51.

Appendix A Functions

A.1 Python

Relevant Python functions:

- `Dspec = np.fft.fft(D)`: The Fourier transform of D
- `k = np.fft.fftshift(k)`: Shifts the zero-frequency component to the center of the spectrum
- `x = np.arange(0, n)`: Creates a 1D array of integers starting at 0 up to `n-1`
- `z = np.append(x, y)`: Appends array y to the end of array x and stores it in variable z
- `ind = np.argwhere(D == condition) or np.where(D == condition)`: Finds the indices of an array where a logical condition is satisfied
- `np.asarray(D)`: Takes a list or some non-array and converts it to an array type
- `L = [i for i in range(n)]`: (List comprehension) Creates a list of `0` to `n-1`, can be used for high dimensions of lists and with if conditional statements
- `os.path.join(path, filename)`: Creates a path object based on the `path` and `filename` given

A.2 MATLAB

Relevant MATLAB functions:

- `axis`: indicate the limits for the current plotted axes
- `abs`: compute absolute value
- `eval` evaluation sentence
- `detrend`: detrend the data
- `figure`: open new figure
- `find`: find values meeting some threshold
- `fft`: fast fourier transform
- `fftshift`: shift zero frequency component
- `double`: change value to type double
- `grid`: add grid lines to a figure
- `imageRectifier`: rectify image into real world coordiantes
- `imshow`: plot image
- `imread`: read image

- `interp1`: 1d interpolation of data
- `*label`: label the x, y, z axes of plots
- `length`: find the length of the largest array dimension
- `load`: load data from a .m file into workspace
- `log`: compute log
- `min`: calculate the minimum
- `nanmean`: calculate the mean
- `num2str`: convert number to string
- `pcolor`: method to plot surfaces in 2d
- `print`: export and save figure
- `repmat`: created matrix with repeated values
- `reshape`: reshape data
- `rgb2gray`: convert rgb image to gray
- `set`: manipulate a figure
- `size`: grab size of a matrix
- `squeeze`: extract data from multi-dimensional matrix
- $\texttt{TRC}_{c}am_{i}nfo$: $functiontocreatefilestructuretoretrievedata$`text` : $definetextforafigure$
- `zeros`: create a matrix of zeros

Appendix B Code

B.1 Python

Code can be found at: https://github.com/emmashie/amath-582

B.1.1 Loading in model output and concatenating data

```
import os
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import glob
import pandas as pd
import xarray as xr

rundir = 'jonswap_test'
fdir = os.path.join('/data2','enuss','funwave_lab_setup',rundir,'output')
savedir = os.path.join('/data2', 'enuss', 'funwave_lab_setup', rundir, 'postprocessing', 'compiled_output')
if not os.path.exists(savedir):
    os.makedirs(savedir)

depFile = os.path.join(fdir,'dep.out')
dep = np.loadtxt(depFile)
[n,m] = dep.shape

# discretization
dx = 0.05
dy = 0.1

# x and y field vectors
x = np.arange(0,m*dx,dx)
y = np.arange(0,n*dy,dy)

def funwave_to_netcdf(flist, x, y, time, fpath, name):
        var = np.zeros((fnum,len(y),len(x)))
        for i in range(fnum):
                var_i = pd.read_csv(os.path.join(fdir,flist[i]), header=None, delim_whitespace=True)
                var_i = np.asarray(var_i)
                var[i,:,:] = var_i
                del var_i
        dim = ["time", "y", "x"]
        coords = [time, y, x]
        dat = xr.DataArray(var, coords=coords, dims=dim, name=name)
        dat.to_netcdf(fpath)

##### ETA #####
flist = [file for file in glob.glob(os.path.join(fdir,'eta_*'))]
fnum = len(flist)
time = np.arange(0,fnum)
fpath = os.path.join(savedir, 'eta.nc')
funwave_to_netcdf(flist, x, y, time, fpath, 'eta')

##### U #####
flist = [file for file in glob.glob(os.path.join(fdir, 'u_*'))]
fnum = len(flist)
time = np.arange(0,fnum)
fpath = os.path.join(savedir, 'u.nc')
funwave_to_netcdf(flist, x, y, time, fpath, 'u')
```

```
##### V #####
flist = [file for file in glob.glob(os.path.join(fdir, 'v_*'))]
fnum = len(flist)
time = np.arange(0,fnum)
fpath = os.path.join(savedir, 'v.nc')
funwave_to_netcdf(flist, x, y, time, fpath, 'v')

##### MASK #####
flist = [file for file in glob.glob(os.path.join(fdir, 'mask_*'))]
fnum = len(flist)
time = np.arange(0,fnum)
fpath = os.path.join(savedir, 'mask.nc')
funwave_to_netcdf(flist, x, y, time, fpath, 'mask')

#### computing wave spectra ####
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import xarray as xr
import numpy.ma as ma
import cmocean.cm as cmo

plt.ion()
#plt.style.use('ggplot')

fdir = os.path.join('/data2','enuss','funwave_lab_setup','jonswap_test','postprocessing')
eta_df = xr.open_dataset(os.path.join(fdir, 'compiled_output', 'eta.nc'))
t = 500
eta = eta_df['eta'].values[t:,:,:]
time = eta_df['time'].values[t:]*0.2 # time in seconds

depFile = os.path.join(fdir,'compiled_output', 'dep.out')
dep = np.loadtxt(depFile)
[n,m] = dep.shape

# discretization
dx = 0.05
dy = 0.1

# x and y field vectors
x = np.arange(0,m*dx,dx)
y = np.arange(0,n*dy,dy)

n = np.max(y)
T = 0.2*len(time)
k_pos = np.linspace(0, n/2, int(len(y)/2))
k_neg = np.linspace(-n/2, 0, int(len(y)/2))
k = (2*np.pi/n)*np.append(k_pos, k_neg)
ks = np.fft.fftshift(k)

[tt, yy] = np.meshgrid(y, time)

j = 1540
j = j+100
fig, ax = plt.subplots(figsize=(10,4))
p = ax.pcolormesh(yy[:300,:], tt[:300,:], eta[:300,:,j], cmap=cmo.deep, shading='gouraud')
ax.set_xticks(time[:300][::25])
ax.set_xticklabels(np.arange(0,60,1)[::5])
```

```
ax.set_ylabel('Alongshore (m)')
ax.set_xlabel('Time (s)')
fig.colorbar(p, ax=ax, label='$\eta$ (m)')
p.set_clim((-0.1,0.1))
fig.savefig('plots/eta_hov_brk.png')

t = 0
spec_avg = np.mean(np.abs(np.fft.fftshift(np.fft.fft(eta[t:-1,:,j], axis=1))), axis=0)
spec_avg2 = np.mean(np.abs(np.fft.fftshift(np.fft.fft(eta[t:-1,:,j+60], axis=1))), axis=0)
spec_avg3 = np.mean(np.abs(np.fft.fftshift(np.fft.fft(eta[t:-1,:,j+100], axis=1))), axis=0)


fig, ax = plt.subplots()
ax.loglog(ks, spec_avg, color='k', label='Offshore Edge')
ax.loglog(ks, spec_avg2, '--', color='k', label='Mid Region')
ax.loglog(ks, spec_avg3, linestyle='dotted', color='k', label='Breaking Region')
ax.set_ylabel('$S_{\eta \eta}$ (m$^3$)')
ax.set_xlabel('L$^{-1}$ (m$^{-1}$)')
ax.legend(loc='best')
fig.savefig('plots/wavenumber_spec3.png')
```

B.2 MATLAB

Code can be found at: https://github.com/cmbaker94/Baker_AMATH582

B.2.1 Compute Spectra Stereo Reconstructions

```
% Quantify the short-crestedness of waves from stereo reconstructions and
% LiDAR imagery. This code will pick a cross-shore location to compute the
% alongshore spectra of sea-surface elevation at each data timesample and
% average for the data span.

% Set up paths and clear workspace
clear all
close all
clc
addpath(genpath('C:\Users\cmbaker9\Documents\MATLAB\MTOOLS'))
addpath(genpath('E:\code\cameras'))

%%%%%%%%%%%%%%%%% USER MANIPULATED SECTION BELOW %%%%%%%%%%%%%%%%%%%%%%%%%

% % WC comparison
calc_spread = [0; 20; 40]; % 20 needs to be seperate?
calc_xshore = [30];

%%%%%%%%%%%%%%%%% USER MANIPULATED SECTION ABOVE %%%%%%%%%%%%%%%%%%%%%%%%%

for cs = 1:length(calc_spread)
    spread = calc_spread(cs);
for cx = 1:length(calc_xshore)
    xshore = calc_xshore(cx);

    %% STEP 0: Given user input find file information

    if spread == 0
        % Lidar (LI)
        lidarfile = '2018-09-01-22-13-48_Velodyne-HDL-32-Data_gridded';
        % Stereo Reconstructions (SR)
        Tcam.cam.tstart         = '09-01-2018-2214UTC';      % time starting collection based on spreadsheet
```

```
    Tcam.cam.tdate          = '09-01-2018-2155UTC';      % trial date and time - format ex: 09-01-2018-2155UTC
elseif spread == 20
    % Lidar (LI)
    lidarfile = '2018-08-30-22-22-39_Velodyne-HDL-32-Data_gridded';
    % Stereo Reconstructions (SR)
    Tcam.cam.tstart  = '08-30-2018-2222UTC';                  % time starting collection based on spreadsheet
    Tcam.cam.tdate   = '08-30-2018-2216UTC';      % trial date and time - format ex: 09-01-2018-2155UTC
elseif spread == 30
    % Lidar (LI)
    lidarfile = '';
    % Stereo Reconstructions (SR)
    Tcam.cam.tstart  = '08-29-2018-2255UTC';                  % time starting collection based on spreadsheet
    Tcam.cam.tdate   = '08-29-2018-2236UTC';      % trial date and time - format ex: 09-01-2018-2155UTC
elseif spread == 40
    % Lidar (LI)
    lidarfile = '2018-08-30-21-29-26_Velodyne-HDL-32-Data_gridded';
    % Stereo Reconstructions (SR)
    Tcam.cam.tstart  = '08-30-2018-2129UTC';                  % time starting collection based on spreadsheet
    Tcam.cam.tdate   = '08-30-2018-2119UTC';      % trial date and time - format ex: 09-01-2018-2155UTC
end

%% STEP 1: Create paths and load files

% general path and names
datapath    = 'E:/';

% Stereo Reconstructions
Tcam        = TRC_camera_info(Tcam);
transect    = load([Tcam.datafolder,'dem_transect_x',num2str(xshore),'m_xavg5cm.mat']);

%% STEP 2: Create figure folders

% figure folder
fssubfolder = datestr(date,'yy-mm-dd');
figfolder   = [datapath,'figures/meas_comp/',Tcam.trialname,'/',Tcam.trimname,fssubfolder,'/'];

% make figure folders
eval(['!mkdir ',datapath,'figures/meas_comp/',Tcam.trialname]);
eval(['!mkdir ',datapath,'figures/meas_comp/',Tcam.trialname,'/',Tcam.trimname]);
eval(['!mkdir ',figfolder])

%% STEP 4: Choose cross-shore location

% Let's choose the location based on the bathymetry and the location with
% maximum points


%% STEP 5: Choose cross-shore location and extract sea-surface elevation

eval(['dy          = transect.y',num2str(xshore),'(2,1)-transect.y',num2str(xshore),'(1,1);'])
eval(['z = squeeze(transect.z',num2str(xshore),'(:,:));'])
eval(['x = nanmean(transect.x',num2str(xshore),',1)'';'])

%% STEP 6: Compute spectra

order = 3;
framelen = 39;

dx = 0;
```

```
count = 0
for i = 6:size(z,2)-5
    count = 1+count;
    ztemp = squeeze(z(:,i-dx:i+dx));
    ztemp = double(nanmean(ztemp,2));
    ztemp(ztemp>1.35)=NaN;
    ztemp(ztemp<0.95)=NaN;
    eval(['ytemp = transect.y',num2str(xshore),'(:,',num2str(i),');'])
    ztemp(ytemp>13)=NaN;
    ztemp(ytemp<-13)=NaN;

    zfittemp = movmean(ztemp,11,'omitnan','EndPoints','fill');
    filt = [nanmean(zfittemp)-(2.5*nanstd(zfittemp)) nanmean(zfittemp)+(4*nanstd(zfittemp))];
    zfittemp(zfittemp<filt(1))=NaN;
    zfittemp(zfittemp>filt(2))=NaN;
    zfit(:,count)=zfittemp;
end

%%
figure('units','inches','position',[1 1 7 7],'Color','w');
count = 1;
for i = 1:size(zfit,2)
    count = count+1;
    ztemp = zfit(:,i);
    % remove nans and interpolate
    nanz            = isnan(ztemp);              % find location of NaNs
    nant            = [1:numel(ztemp)]';        % make strings for interp
    nanratio        = sum(nanz)/length(nant); % find ratio of NaNs
    ztemp(nanz)     = interp1(nant(~nanz),ztemp(~nanz),nant(nanz)); % interpolate between the NaNs
    zcutnan         = ztemp(~isnan(ztemp)); % if nans still at the beginning or end of list, just remove these p
    eta             = zcutnan-nanmean(zcutnan);
    eta             = detrend(eta,1);% detrending

    if length(eta)>2350
%         WL = length(eta);
%         OL = 0;
        L = 13; % spatial domain
        n = length(eta); % Fourier modes
        ktemp = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; ks = fftshift(k);
        Stemp = fftshift(fft(eta));
%         [Stemp,ktemp,Sc(count,:,:)]   = pwelch(eta,WL,OL,[],1/dy,'ConfidenceLevel',0.95); % compute spectra
        [val,id] = min(abs(ktemp-1));
        filter = zeros(size(ktemp));
        filter(1:id-1) = 1;
        filter(id:id+2) = [0.7089 0.2363 0.0225];
        S(count,:) = Stemp.*filter;
        k(count,:) = ktemp;
    else
        S(count,:) = NaN(size(S(count-1,:)));
        k(count,:) = NaN(size(k(count-1,:)));
        Sc(count,:,:) = NaN(size(Sc(count-1,:,:)));
    end

    if i<20
        clf
        semilogy(k(count,:),S(count,:),'k','LineWidth',2)
        xlabel('$L^{-1}$ (m$^{-1}$)','interpreter','latex','fontsize',20);
        ylabel('$S_{\eta\eta}$ (m$^2$/Hz)','interpreter','latex','fontsize',20);
```

```
            h1=gca;
            set(h1, 'YScale', 'log')
            set(h1,'tickdir','in','xminortick','on','yminortick','on');
            set(h1,'ticklength',1*get(h1,'ticklength'));
            set(h1,'fontsize',15);
            ylim([10^-6 10^-3])
            title(['x = ',num2str(xshore),'m , t = ',num2str(i/Tcam.Hz)],'interpreter','latex','fontsize',20);
            pause(0.2)
        end
    end


    %% Prep data
    Savg=nanmean(S,1);
    Scavg = nanmean(Sc,1);

    y = ytemp;
    x = nanmean(x);


    subname = '';%'onewindow_11movmean_detrended';
    psname = [Tcam.datafolder,'AMATH/dem_transect_wavecrest_length_x',num2str(xshore),'_',subname,'.mat'];
    eval(['save -v7.3 ',psname,' S',' k',' Sc',' Savg',' Scavg',' y',' x',' zfit']);

close all
clear zfit
end
end
```

B.2.2 Compute Spectra from Images

```
% This code will plot rectified images using code from
% D_gridGenExpampleRect.m in the CIRN-Quantitative-Coastal-Imaging-Toolbox


% Set up paths and clear workspace
clear all
close all
clc
addpath(genpath('C:\Users\cmbaker9\Documents\MATLAB\MTOOLS'))
addpath(genpath('E:\code\cameras'))
addpath(genpath('E:\code\insitu'))
addpath(genpath('E:\code\CIRN-Quantitative-Coastal-Imaging-Toolbox\X_CoreFunctions/'))
%%%%%%%%%%%%%%%%%%%% USER MANIPULATED SECTION %%%%%%%%%%%%%%%%%%%%%%%%%

    % Trial info
    Tinfo.Hs = 0.30;
    Tinfo.Tp = 2;
    Tinfo.tide = 1.07;
    Tinfo.spread = 20;

if Tinfo.spread == 0
    % In situ
    sz.Tdate = '09-01-2018-2213UTC';
    is.Tdate = '09-06-2018-1559UTC';
    % Lidar (LI)
    lidarfile = '2018-09-01-22-13-48_Velodyne-HDL-32-Data_gridded';
    % Stereo Reconstructions (SR)
    Tcam.tstart         = '09-01-2018-2214UTC';                  % time starting collection based on spreadsheet
```

```
    Tcam.tdate          = '09-01-2018-2155UTC';      % trial date and time - format ex: 09-01-2018-2155UTC
    offsets             = [-487;-69];% [-588; -70]; % index offset relative to camera [in situ, lidar]
elseif Tinfo.spread == 20
    % In situ
    sz.Tdate = '08-30-2018-2222UTC';
    is.Tdate = '09-06-2018-1655UTC';
    % Lidar (LI)
    lidarfile = '2018-08-30-22-22-39_Velodyne-HDL-32-Data_gridded';
    % Stereo Reconstructions (SR)
    Tcam.tstart  = '08-30-2018-2226UTC';                  % time starting collection based on spreadsheet
    Tcam.tdate   = '08-30-2018-2216UTC';      % trial date and time - format ex: 09-01-2018-2155UTC
elseif Tinfo.spread == 40
    % In situ
    sz.Tdate = '08-30-2018-2129UTC'; % 2119+20 min
    is.Tdate = '09-06-2018-1841UTC';
    % Lidar (LI)
    lidarfile = '2018-08-30-21-29-26_Velodyne-HDL-32-Data_gridded';
    % Stereo Reconstructions (SR)
    Tcam.tstart  = '08-30-2018-2129UTC';                  % time starting collection based on spreadsheet
    Tcam.tdate   = '08-30-2018-2119UTC';      % trial date and time - format ex: 09-01-2018-2155UTC
end

%% STEP 1: Create paths, files and naming

% general path and names
datapath    = 'E:\';

% Stereo Reconstructions
Tcam.camerasys   = 'TRM';                     % camera setup - e.g., TRM (offshore) or TRC (onshore)
Tcam.scene       = '1';                       % scene number of trial - typ 1
Tcam.imagestart  = 0;                      % images number of first frame on file
Tcam.dx          = 0.05;
Tcam.dy          = 0.05;
Tcam.regx        = [25:Tcam.dx:37];
Tcam.regy        = [-13:Tcam.dy:13];
Tcam.Hz           = 8;

imagepath = ['D:\TRC_Fall_Experiment\',Tcam.camerasys,'-',Tcam.tdate,'\',Tcam.camerasys,'-',Tcam.tdate,'_Scene1_JPEG
Tcam.numframes   = length(dir([imagepath, '*.jpg']))-1;   % number of frames processed

Tcam.trialname   = [Tcam.camerasys,'-',Tcam.tdate];
Tcam.imagerange  = [num2str(Tcam.imagestart,'%05.f'),'-',num2str(Tcam.imagestart+(Tcam.numframes-1),'%05.f')];
Tcam.trimname    = ['frames_',Tcam.imagerange,'\'];
Tcam.datafolder  = [datapath,'data\processed\cameras\',Tcam.trialname,'\',Tcam.trimname];


%% STEP 2: Create figure folders

% figure folder
fssubfolder = datestr(date,'yy-mm-dd');
figfolder   = [datapath,'figures\cameras\images\',Tcam.trialname,'\',Tcam.trimname,fssubfolder,'\'];

% make figure folders
eval(['!mkdir ',datapath,'figures\cameras\images\',Tcam.trialname]);
eval(['!mkdir ',datapath,'figures\cameras\images\',Tcam.trialname,'\',Tcam.trimname]);
eval(['!mkdir ',figfolder])

%% Establish timeseries range:
```

```
starttemp       = datenum(Tcam.tstart(1:end-3),'mm-dd-yyyy-HHMM')+datenum(0,0,0,0,0,Tcam.imagestart/Tcam.Hz);
endtemp         = datenum(Tcam.tstart(1:end-3),'mm-dd-yyyy-HHMM')+datenum(0,0,0,0,0,(Tcam.imagestart+Tcam.numframes-
camera.time        = starttemp:datenum(0,0,0,0,0,1/Tcam.Hz):endtemp;
clear *temp

%% Insitu

load([datapath,'data/processed/cameras/c2_intrinsics_extrinsics.mat']);

% extrinsics = [39.35 0.02 11.03 267.7*pi/180 32.82*pi/180 0.13*pi/180];
extrinsics(1) = extrinsics(1)+0.75;

%% Create matrix to rectify images
% see code: D_gridGenExampleRect.m

localOrigin = [0, 0]; % [ x y]
localAngle =[0]; % Degrees +CCW from Original World X
localFlagInput=1;

ixlim=[19 36];
iylim=[-14.5 14.5];
idxdy=0.01;

iz=0;

%  World Extrinsics, need to make into sell
Extrinsics{1}=extrinsics;
Intrinsics{1}=intrinsics;

% %  Local Extrinsics
% localExtrinsics{k} = localTransformExtrinsics(localOrigin,localAngle,1,xtrinsics{1});

%  Create Equidistant Input Grid
[iX iY]=meshgrid([ixlim(1):idxdy:ixlim(2)],[iylim(1):idxdy:iylim(2)]);

%  Make Elevation Input Grid
iZ=iX*0+iz;

% If entered as Local
if localFlagInput==1
    % Assign local Grid as Input Grid
    localX=iX;
    localY=iY;
    localZ=iZ;

    % Assign world Grid as Rotated local Grid
    [ X Y]=localTransformEquiGrid(localOrigin,localAngle,0,iX,iY);
    Z=X*.0+iz;
end

teachingMode = 0;

%% extract x = 28

%% plot
imageno =  7200:1:11999;

% figure('units','inches','position',[1 1 12 8],'color','w')
for i = 1:length(imageno)
```

```
    %read image
    imagefile = [imagepath,getfield(dir([imagepath,'Movie1_Scene1_c2_',sprintf('%05d',imageno(i)),'_*.jpg']),'name')
    IM{1} = imread(imagefile);

    % World Rectification
    [Ir]= imageRectifier(IM,Intrinsics,Extrinsics,X,Y,Z,teachingMode);

    [val,id] = min(abs(X(1,:)-28));
    Irtemp = double(rgb2gray(Ir));
    IMtran(i,:) = Irtemp(:,id)';
    IMy(i,:) = Y(:,1);
    dy = Y(2,1)-Y(1,1);

    WL = length(IMy(i,:));
    OL = 0;

    speccompute = (IMtran(i,:)-nanmean(IMtran(i,:)))/256;
    speccompute             = detrend(speccompute,1);% detrending
    %     [Stemp,ktemp]   = pwelch(speccompute,WL,OL,[],1/dy,'ConfidenceLevel',0.95); % compute spectra
    L = 13; % spatial domain
    n = length(IMtran(i,:)); % Fourier modes
    ktemp = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; ks = fftshift(k);
    Stemp = fftshift(fft(eta));
    [val,id] = min(abs(ktemp-1));
    filter = zeros(size(ktemp));
    filter(1:id-1) = 1;
    filter(id:id+2) = [0.7089 0.2363 0.0225];
    S(i,:) = Stemp.*filter;
    k(i,:) = ktemp;

end

%%
Savg=nanmean(S,1);
y = nanmean(IMy,1);
x = 30;

xshore = 30;
subname = '';%'onewindow_11movmean_detrended';
eval(['!mkdir ',Tcam.datafolder,'AMATH'])
psname = [Tcam.datafolder,'AMATH/image_values_transect_wavecrest_length_x',num2str(xshore),'_',subname,'.mat'];
eval(['save -v7.3 ',psname,' IMtran',' IMy',' x']);
psname = [Tcam.datafolder,'AMATH/image_transect_wavecrest_length_x',num2str(xshore),'_',subname,'.mat'];
eval(['save -v7.3 ',psname,' S',' k',' Savg',' y',' x']);
```

B.2.3 Plot Spectra

```
% Set up paths and clear workspace
clear all
close all
clc

%%%%%%%%%%%%%%%% USER MANIPULATED SECTION BELOW %%%%%%%%%%%%%%%%%%%%%%%%

% WC comparison
xshore = 30;
sprd = [0 20 40];
```

```
%%%%%%%%%%%%%%%% USER MANIPULATED SECTION ABOVE %%%%%%%%%%%%%%%%%%%%%%%%

%% STEP 1: Create paths and load files

% general path and names
datapath    = '/Users/cmbaker9/Documents/UW_Classes/AMATH_Data_Analysis/project/';

subname = '_';
for i = 1:length(sprd)
    eval(['spec',num2str(sprd(i)),'.image    = load([datapath,num2str(sprd(i)),''deg/image_transect_wavecrest_length
    eval(['spec',num2str(sprd(i)),'.stereo    = load([datapath,num2str(sprd(i)),''deg/dem_transect_wavecrest_length_
end

figfolder   = '/Users/cmbaker9/Documents/UW_Classes/AMATH_Data_Analysis/project/';

%% Plot figure

figure('units','inches','position',[1 1 16 5],'Color','w');
ax1 = axes('Position',[0.075 0.15 0.275 0.75])
plot(spec40.stereo.k(20,:),spec40.stereo.Savg,'k','LineWidth',2,'LineStyle','-')
hold on
plot(spec20.stereo.k(20,:),spec20.stereo.Savg,'k','LineWidth',2,'LineStyle','--')
plot(spec0.stereo.k(20,:),spec0.stereo.Savg,'k','LineWidth',2,'LineStyle',':')
text(10^-1.57,10^-1.88,'(a) Stereo Reconstructions','interpreter','latex','fontsize',20);
box on
h1=gca;
set(h1, 'YScale', 'log')
set(h1, 'XScale', 'log')
set(h1,'tickdir','out','xminortick','on','yminortick','on');
set(h1,'ticklength',2*get(h1,'ticklength'));
set(h1,'ytick',[10^(-4) 10^(-3) 10^(-2) 10^(-1)],'yticklabel',{'10^{-4}' '10^{-3}' '10^{-2}' '10^{-1}'});
set(h1,'xtick',[10^(-2) 10^(-1) 10^(0) 10^(1)],'xticklabel',{'10^{-2}' '10^{-1}' '10^{0}' '10^{1}'});
set(h1,'fontsize',15);
xlabel('$L^{-1}$ (m$^{-1}$)','interpreter','latex','fontsize',20);
ylabel('$S_{\eta\eta}$ (m$^3$)','interpreter','latex','fontsize',20);
ylim([10^-4.1 10^-2])
xlim([nanmin(spec40.stereo.k(20,2)) 1])
h2 = legend('$\sigma_{\theta} = 40^{\circ}$','$\sigma_{\theta} = 20^{\circ}$','$\sigma_{\theta} = 0^{\circ}$')
set(h2,'interpreter','latex','fontsize',18,'orientation','vertical','Location','northeast');


ax2 = axes('Position',[0.440 0.15 0.275 0.75])
plot(spec40.image.k(20,:),spec40.image.Savg,'k','LineWidth',2,'LineStyle','-')
hold on
plot(spec20.image.k(20,:),spec20.image.Savg,'k','LineWidth',2,'LineStyle','--')
plot(spec0.image.k(20,:),spec0.image.Savg,'k','LineWidth',2,'LineStyle',':')
box on
h1=gca;
set(h1, 'YScale', 'log')
set(h1, 'XScale', 'log')
set(h1,'tickdir','out','xminortick','on','yminortick','on');
set(h1,'ticklength',2*get(h1,'ticklength'));
set(h1,'ytick',[10^(-4) 10^(-3) 10^(-2) 10^(-1) 10^(0)],'yticklabel',{'10^{-4}' '10^{-3}' '10^{-2}' '10^{-1}' '10^{0
set(h1,'xtick',[10^(-2) 10^(-1) 10^(0) 10^(1)],'xticklabel',{'10^{-2}' '10^{-1}' '10^{0}' '10^{1}'});
set(h1,'fontsize',15);
ylabel('$S_{\mathrm{image}}$ (m)','interpreter','latex','fontsize',20);
xlabel('$L^{-1}$ (m$^{-1}$)','interpreter','latex','fontsize',20);
text(10^-1.57,10^-0.3,'(b) Images','interpreter','latex','fontsize',20);
```

```
ylim([10^-2 10^-0.4])
xlim([nanmin(spec40.image.k(20,2)) 1])

extra = '';
Sname1 = [figfolder,'comparison_sprd0and40_x',num2str(xshore(1)),'-',num2str(xshore(end)),'m',extra];
print(Sname1,'-dpng')
```