```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
These are extra mesh generating tools
Author: Alexandra Christensen
Created: Wed Oct 14 10:25:58 2020
Updated: 12/01/2022

"""


import rasterio
from osgeo import gdal
import os
import sys
import numpy as np
import matplotlib.pyplot as plt
import geopandas as gpd
import matplotlib

from orinoco import (get_distance_in_channel,
                     get_distance_segments)

import warnings
warnings.filterwarnings('ignore', '.*do not.*', )
warnings.warn('ShapelyDeprecationWarning')
warnings.warn('UserWarning')

from skimage.segmentation import felzenszwalb, slic
from skimage.color import label2rgb

if os.getcwd().split('/')[1] == 'Users':
    code_path = '/users/alchrist/documents/github/ANUGA/processing/code/'
else:
    code_path = '/projects/loac_hydro/alchrist/processing/code'
sys.path.insert(1, code_path)

from cleaning_tools import make_distance
from polygon_tools import findconnectedwater, get_nearest,
nearest_neighbor, getpolygonpoints, removenearbypoints, delete_holes



def
make_SWORD_networks(folders,ref,delta,parameters,pixel_step,watermaskname
,skip=False):

print('\n\n\n############################################################
##################################')
    print('###############################[Step
4][Make_SWORD_Networks]##################################')

print('##################################################################
############################\n')
```

```
    save_profile = ref.profile
    save_profile['compress'] = 'deflate'


    xres,yres = int(save_profile['transform'][0]),-
int(save_profile['transform'][4])

#########################################################################
##
    ###################### Import Config Parameters
###########################

#########################################################################
##
    parameters.iloc[0]
    ulx  = parameters['ulx'][0]
# ULX coordinate
    lry  = parameters['lry'][0]
# LRY coordinate
    lrx  = parameters['lrx'][0]
# LRX coordinate
    uly  = parameters['uly'][0]
# ULY coordinate
    #os.system('gdal_rasterize -burn 1 -tr %s %s -te %s %s %s %s -tap
%s%s_oceanextended.shp %s%s_oceanextended.tif'
%(xres,yres,ulx,lry,lrx,uly,folders[6],delta,folders[1],delta))
    #os.system('gdal_rasterize -burn 1 -tr %s %s -te %s %s %s %s -tap
%s%s_riverextended.shp %s%s_riverextended.tif'
%(xres,yres,ulx,lry,lrx,uly,folders[6],delta,folders[1],delta))

    try: os.mkdir(folders[1])
    except:''


#########################################################################
#
    ## RASTERIZE WATER/LAND DELIENATI SHAPEFILES

#########################################################################
#
    print('\n')
    #print('[Step 4][Rasterizing ocean polygons with %s m resolution]'
%(xres))

    watermask = rasterio.open('%s%s_watermask_%s.tif'
%(folders[8],delta,xres))#.read(1)
    # with
rasterio.open('/users/alchrist/documents/currentprojects/anuga/globaldelt
as/SWORD_watermasks/%s.tif' %(watermaskname),'w', **save_profile) as dst:
    #     dst.write_band(1,watermask.astype('float64'))
    try: oceanmask = rasterio.open('%s%s_fulloceans_%s.tif'
%(folders[8],delta,xres))
```

```
    except:
        os.system('gdalwarp -overwrite -tr %s %s -te %s %s %s %s -
srcnodata -9999 -dstnodata -9999 %s%s_fulloceans_30.tif
%s%s_fulloceans_%s.tif -co COMPRESS=DEFLATE'
%(xres,yres,ulx,lry,lrx,uly,folders[8],delta,folders[8],delta,xres))
        oceanmask = rasterio.open('%s%s_fulloceans_%s.tif'
%(folders[8],delta,xres))
    #rivermask3 = ndimage.binary_closing(rivermask,
np.ones([3,3]),iterations=2).astype(rivermask.dtype)
    # watermask = skimage.morphology.diameter_closing(watermask,
diameter_threshold=50, connectivity=1, parent=None, tree_traverser=None)
    #watermask = ndimage.binary_dilation(watermask,
np.ones([2,2]),iterations=1).astype(watermask.dtype)
    # structure1 = np.array([[1., 0.], [0., 1.]])
    # watermask2 = ndimage.morphology.binary_hit_or_miss(watermask,
structure1 =
structure1.astype(watermask.dtype),structure2=np.ones([2,2]))
    # watermask3 = np.where(watermask2==True,1,0)
    # watermask4 = watermask3[:,1:]
    # watermask5 = np.append(watermask4,
np.zeros([watermask4.shape[0],1]), axis=1)
    # watermask6 = watermask + watermask5



    #test_features = skimage.segmentation.felzenszwalb(rivermask,
scale=1, sigma=0.8, min_size=20)
    #footprint = skimage.morphology.disk(50)
    #res = skimage.morphology.white_tophat(rivermask,footprint)

#########################################################################
#####
    ####################### Get River Widths and Depths
###########################

#########################################################################
#####
    ## Use Orinoco code (Charlie) to get distance and widths of rivers
    print('[Step 4][Build river network with Orinoco] .......')
    distance =
make_distance(watermask,oceanmask,folders,delta,pixel_step,xres)

    return distance




#########################################################################
###

#########################################################################
###
```

```
#########################################################################
###
    ## PART VI
    ## OUTPUT: segments for SWOT simulator

#########################################################################
#
def
make_segments_for_swot(folders,delta,ref,parameters,skip):#,min_size,scal
e):
    step = 6

print('\n################################################################
##############################')
    print('#############################[Step
%s][Make_Segments_For_SWOT]################################'%(step))

print('##################################################################
############################\n')
    if skip == False:
        save_profile = ref.profile
        save_profile['compress'] = 'deflate'
        save_profile['nodata'] = None
        save_profile['dtype'] = 'int32'
        xres,yres = int(save_profile['transform'][0]),-
int(save_profile['transform'][4])

        segments_dir = folders[7] + 'segments_%s/' %(xres)
        try: os.mkdir(segments_dir)
        except:''
        print('Segments will be saved to %s' %(segments_dir))


        parameters.iloc[0]
        EPSG = parameters['EPSG'][0]
# Coordinate System must be UTM
        ulx  = parameters['ulx'][0]
# ULX coordinate
        lry  = parameters['lry'][0]
# LRY coordinate
        lrx  = parameters['lrx'][0]
# LRX coordinate
        uly  = parameters['uly'][0]
# ULY coordinate

        river = rasterio.open('%s%s_rivers_%s.tif'
%(folders[8],delta,xres))

        ocean = rasterio.open('%s%s_fulloceans_%s.tif'
%(folders[8],delta,(xres)))
        transform = ocean.transform
        dx, dy = transform.a, -transform.e
        oceanmask = np.where(ocean.read(1).astype('int')==1,1,0)
        #oceanmask = np.where((ocean.read(1)==1)&(dem<0)&(dem>-
100),True,False)

        dem = ref.read(1).astype('float32')
#rasterio.open('%s%s_GEBCO_%s.tif' %(folders[7],delta,xres*3)).read(1)
        dem_masked = np.where(oceanmask==1,dem,np.nan)

        watermask = np.where((river.read(1)==1) | (oceanmask==1),1,0)


        print('[Step %s][Making ocean segments using skimage
segmentation] ......'%(step))
        pixel_size = ocean.profile['transform'][0]

        total_pixels = np.sum(np.sum(oceanmask))
        print('There are %s water (non river) pixels' %(total_pixels))
        print('Therefore, the total water (non river) area is approx.
%sm2' %(total_pixels*dx*dy))
        #print('##################### Superpixel area will be %sm'
%(superpixel_area))
        #n_slic_segments =
int(np.round(total_pixels*pixel_size*pixel_size)/superpixel_area)#211600)
))
        #print('##################### # Slic segments will be %s'
%(n_slic_segments))

        cmap = matplotlib.colors.ListedColormap ( np.random.rand (
256,3))
        ix = [0.0001,0.005,1]#compactness
        seg_size = [50000,100000,1000000,10000000]
        jx = [int(dx*dx*total_pixels/x) for x in seg_size]#to make
segments of size 0.5ha, 1ha, 10ha, and 100ha
        #jx = [10,100,1000,10000]#number of segments

        fig,axes =
plt.subplots(ncols=len(jx),nrows=len(ix),figsize=(20,20))
        print('Slic Segmentation:')
        print('Compactness: %s' %(ix))
        print('Number of Segments: %s' %(jx))
        for i in range(0,len(ix)):
            for j in range(0,len(jx)):
                compactness = ix[i]
                nsegments = jx[j]
                if
os.path.isfile('%s%s_oceandem_superpixels_%sm2_compact%s_segs%s_slic.shp'
%(segments_dir,delta,seg_size[j],compactness,nsegments))==False:
                    print('Compactness: %s     # Segments: %s'
%(compactness, nsegments))
                    superpixel_labels_s = slic(dem, n_segments=nsegments,
compactness=compactness,mask=oceanmask)
                    superpixel_labels_s =
np.where((oceanmask==1),superpixel_labels_s,0)
```

```
                    with rasterio.open(segments_dir+
'%s_oceandem_superpixels_%sm2_compact%s_segs%s_slic.tif'
%(delta,seg_size[j],compactness,nsegments), 'w', **save_profile) as ds:
                        ds.write(superpixel_labels_s.astype(np.int32), 1)

try:os.remove('%s%s_oceandem_superpixels_%sm2_compact%s_segs%s_slic.shp'%
(segments_dir,delta,seg_size[j],compactness,nsegments))
                    except:''
                    os.system('gdal_polygonize.py -q
%s%s_oceandem_superpixels_%sm2_compact%s_segs%s_slic.tif '\

'%s%s_oceandem_superpixels_%sm2_compact%s_segs%s_slic.shp '\

%(segments_dir,delta,seg_size[j],compactness,nsegments,segments_dir,delta
,seg_size[j],compactness,nsegments))
                else:
                    superpixel_labels_s = rasterio.open(segments_dir+
'%s_oceandem_superpixels_%sm2_compact%s_segs%s_slic.tif'
%(delta,seg_size[j],compactness,nsegments)).read(1)
                axes[i,j].imshow(superpixel_labels_s,cmap=cmap)

        plt.tight_layout()
        plt.savefig('%s%s_Segments_Ocean_slic.png' %(segments_dir,delta))

        ix = [0.1,1000]#[0.1,10,1000] #scale = Free parameter. Higher
means larger clusters.
        jx = [0.1,0.95] #sigma = Width (standard deviation) of Gaussian
kernel used in preprocessing.
        seg_size = [1000,10000,100000,1000000] # to make segments of
minimum size 0.1ha, 1ha, 10ha, 100ha
        zx = [int(x/(dx*dy)) for x in seg_size]
        #zx = [1,10,100] #minsize = Minimum component size. Enforced
using postprocessing.
        print('Felzenswalb segmentation:')
        print('Scale: %s' %(ix))
        print('Sigma: %s' %(jx))
        print('Minimum size: %s' %(zx))

        for i in range(0,len(ix)):
            fig,axes =
plt.subplots(ncols=len(zx),nrows=len(jx),figsize=(20,20))
            for j in range(0,len(jx)):
                for z in range(0,len(zx)):
                        scale = ix[i]
                        sigma = jx[j]
                        min_size = zx[z]
                        if
os.path.isfile('%s%s_oceandem_superpixels_%sm2_scale%s_sigma%s_min%s_felz
.shp' %(segments_dir,delta,seg_size[z],scale,sigma,min_size))==False:
                            print('Scale: %s     Sigma: %s     Min Size:
%s' %(scale, sigma, min_size))
                            superpixel_labels_f =
felzenszwalb(dem_masked.astype('float32'), scale=scale, sigma=sigma,
min_size=min_size)
```

```
                            superpixel_labels_f =
np.where((oceanmask==1),superpixel_labels_f,0)
                            with rasterio.open(segments_dir +
'%s_oceandem_superpixels_%sm2_scale%s_sigma%s_min%s_felz.tif'
%(delta,seg_size[z],scale,sigma,min_size), 'w', **save_profile) as ds:

ds.write(superpixel_labels_f.astype(np.int32), 1)
                            try:

os.remove('%s%s_oceandem_superpixels_%sm2_scale%s_sigma%s_min%s_felz.shp'
\

%(segments_dir,delta,seg_size[z],scale,sigma,min_size))
                            except:''
                            os.system('gdal_polygonize.py
%s%s_oceandem_superpixels_%sm2_scale%s_sigma%s_min%s_felz.tif '\

'%s%s_oceandem_superpixels_%sm2_scale%s_sigma%s_min%s_felz.shp -q'\

%(segments_dir,delta,seg_size[z],scale,sigma,min_size,segments_dir,delta,
seg_size[z],scale,sigma,min_size))
                        else:
                            superpixel_labels_f =
rasterio.open(segments_dir +
'%s_oceandem_superpixels_%sm2_scale%s_sigma%s_min%s_felz.tif'
%(delta,seg_size[z],scale,sigma,min_size)).read(1)
                        axes[j,z].imshow(superpixel_labels_f,cmap=cmap)

            plt.tight_layout()
            plt.savefig('%s%s_Segments_Ocean_felz_%s.png'
%(segments_dir,delta,sigma))

        print('[Step %s][Making river segments using Orinoco]
......'%(step))
        px = [2,4,6,8]
        print('Resolution is %sm '%(dx))
        print('Orinoco with pixel steps: %s' %(px))
        fig,axes = plt.subplots(ncols=len(px),figsize=(20,10))
        print('\n######[Make_Channel_Networks][Orinoco -->
get_distance_in_channel] .......\n')
        dist = get_distance_in_channel(watermask,
                                        oceanmask,
                                        dx=dx,
                                        dy=dy,
                                        min_rel_area=0) #removes areas
with less than 2.5% of total size

        print('\n######[Make_Channel_Networks][Orinoco -->
get_distance_segments] .......\n')
        for p in range(len(px)):
            pixel_step = px[p]
            print('Pixel Step: %s' %(pixel_step))
            if os.path.isfile('%s%s_river_superpixels_%sx%s.shp -q'
%(segments_dir,delta,xres,pixel_step)) == False:
```

```
                # Build distance raster with the scikit-fmm distance
function (phi)

                # Build segment raster according to phi(x)/D where D is
threshold defined by pixel_step * res
                # Connectivity set as 8, edges or corners connectedness
                # Interface adjacent segments are IDs of segments at
river/ocean interface
                # dist = np.where(np.isnan(dist),0,dist)
                segments, interface_adj_segments =
get_distance_segments(dist,

pixel_step,

dx=dx,

dy=dy,

connectivity=8,

min_size=None)

                #segments = np.where(segments==0,np.nan,segments)
                with rasterio.open('%s%s_river_superpixels_%sx%s.tif'
%(segments_dir,delta,xres,pixel_step), 'w', **save_profile) as ds:
                    ds.write(segments.astype(np.int32), 1)

                if os.path.isfile('%s%s_river_superpixels_%sx%s.shp'
%(segments_dir,delta,xres,pixel_step)):
                    os.remove('%s%s_river_superpixels_%sx%s.shp'
%(segments_dir,delta,xres,pixel_step))

                ## Polygonize the segments
                print('\n######[Make_Channel_Networks][Orinoco -->
segment raster to shapefile] .......\n')
                os.system('gdal_polygonize.py
%s%s_river_superpixels_%sx%s.tif %s%s_river_superpixels_%sx%s.shp -q'
%(segments_dir,delta,xres,pixel_step,segments_dir,delta,xres,pixel_step))
                #segments =
gpd.read_file('%s%s_segments_%sx%s.shp'%(folders[7],delta,xres,pixel_step
))
            else:
                segments =
rasterio.open('%s%s_river_superpixels_%sx%s.tif'
%(segments_dir,delta,xres,pixel_step)).read(1)
            axes[p].imshow(segments,cmap=cmap)


        plt.tight_layout()
        plt.savefig('%s%s_Segments_River.png' %(segments_dir,delta))

        print('[Step 5][Segments saved] ......')
        print('[Step 5][Finished] .......')
    else:
        print('[Step 5][SKIP].......')


#########################################################################
###
#########################################################################
###
#########################################################################
###
## PART VI
## OUTPUT: ANUGA mesh
#########################################################################
#
def
make_mesh_polygons(folders,delta,res,parameters,med_width,cellsperwidth,o
utdir,skip):
    step=6

print('\n################################################################
##############################')
    print('################################[Step
6][Make_Mesh_Polygons]##################################')

print('##################################################################
############################\n')
    if skip == False:

#########################################################################
##
        ###################### Import Config Parameters
###########################

#########################################################################
##
        xres,yres = res,res
        parameters.iloc[0]
        EPSG = parameters['EPSG'][0]
# Coordinate System must be UTM
        ulx  = parameters['ulx'][0]
# ULX coordinate
        lry  = parameters['lry'][0]
# LRY coordinate
        lrx  = parameters['lrx'][0]
# LRX coordinate
        uly  = parameters['uly'][0]
# ULY coordinate

        model_domain = gpd.read_file('%s%s_modeldomain.shp'
%(folders[7],delta))
        print('\n[Step 6][Make_Mesh_Polygons][Get all water bodies]
.......\n')

        allwater = gpd.read_file('%s%s_water_connected_and_lakes_%s.shp'
%(folders[7],delta,xres))
```

```
        allwater = gpd.overlay(allwater,model_domain,how='intersection')
        water_connected_not_lakes =
gpd.read_file("%s%s_water_connected_%s.shp" %(folders[7],delta,xres))
        water_connected_not_lakes =
gpd.overlay(water_connected_not_lakes,model_domain,how='intersection')

        rivers = gpd.read_file('%s%s_rivers_%s.shp'
%(folders[7],delta,xres))
        rivers = gpd.overlay(rivers,model_domain,how='intersection')
        oceans = gpd.read_file('%s%s_fulloceans_%s.shp'
%(folders[7],delta,xres))
        oceans = gpd.overlay(oceans,model_domain,how='intersection')

        lands = gpd.read_file('%s%s_lands_%s.shp'
%(folders[7],delta,xres))
        lands = gpd.overlay(lands,model_domain,how='intersection')
        try:
            lakes = gpd.read_file('%s%s_lakes_%s.shp'
%(folders[7],delta,xres))
            lakes = gpd.overlay(lakes,model_domain,how='intersection')
        except: lakes = []

        print('\n[Step %s][Make_Polygons][3000m buffer coastal zone]
.......\n'%(step))
        try:nearshore_oceans = gpd.read_file("%s%s_nearshore_%s.shp"
%(folders[7],delta,xres))
        except:
            lands.geometry = lands.buffer(100,resolution=2)
            lands.geometry = lands.buffer(500,resolution=2)
            lands['dissolve'] = 1
            lands = lands.dissolve(by='dissolve').reset_index(drop=True)
            lands.geometry = lands.buffer(2400,resolution=2)
            lands['dissolve'] = 1
            lands = lands.dissolve(by='dissolve').reset_index(drop=True)
            nearshore_oceans =
gpd.overlay(oceans,lands,how='intersection')
            nearshore_oceans = nearshore_oceans.rename(columns={"FID_1":
"FID"})
            #nearshore_oceans = nearshore_oceans.drop(columns='FID_2')
            nearshore_oceans =
gpd.overlay(nearshore_oceans,model_domain,how='intersection')
            nearshore_oceans['dissolve'] = 1
            nearshore_oceans =
nearshore_oceans.dissolve(by='dissolve').reset_index(drop=True)

            nearshore_oceans.to_file("%s%s_nearshore_%s.shp"
%(folders[7],delta,xres))
            print('#################### Nearshore polygons saved to
%s%s_nearshore_%s.shp' %(folders[7].split(delta)[-1],delta,xres))
        os.system('gdal_rasterize -burn 1 -tr %s %s -te %s %s %s %s
%s%s_nearshore_%s.shp %s%s_nearshore_%s.tif -co COMPRESS=DEFLATE'
%(xres,yres,ulx,lry,lrx,uly,folders[7],delta,xres,folders[8],delta,xres))
```

```
        print('\n[Step %s][Make_Polygons][Non-coastal ocean zone]
.......\n'%(step))
        try:farocean = gpd.read_file("%s%s_farocean_%s.shp"
%(folders[7],delta,xres))
        except:
            farocean =
gpd.overlay(oceans,nearshore_oceans,how='difference')
            farocean.to_file("%s%s_farocean_%s.shp"
%(folders[7],delta,xres))
            farocean =
gpd.overlay(farocean,model_domain,how='intersection')

            farocean.to_file("%s%s_farocean_%s.shp"
%(folders[7],delta,xres))
            print('#################### Farshore polygons saved to
%s%s_farshore_%s.shp' %(folders[7].split(delta)[-1],delta,xres))
        os.system('gdal_rasterize -burn 1 -tr %s %s -te %s %s %s %s
%s%s_farocean_%s.shp %s%s_farocean_%s.tif -co COMPRESS=DEFLATE'
%(xres,yres,ulx,lry,lrx,uly,folders[7],delta,xres,folders[8],delta,xres))

        print('\n[Step %s][Make_Polygons][Add lakes to connected water]
.......\n'%(step))
        try:
            water_connected_10km =
gpd.read_file("%s%s_water_connected_10km_%s.shp"
%(folders[7],delta,xres))
        except:
            water_connected_10km =
water_connected_not_lakes.copy(deep=True).reset_index(drop=True)
            water_connected_10km.geometry =
water_connected_10km.buffer(100,resolution=2)
            water_connected_10km.geometry =
water_connected_10km.buffer(500,resolution=2)
            water_connected_10km['dissolve'] = 1
            water_connected_10km =
water_connected_10km.dissolve(by='dissolve').reset_index(drop=True)
            water_connected_10km.geometry =
water_connected_10km.buffer(7400,resolution=2)
            water_connected_10km['dissolve'] = 1
            water_connected_10km =
water_connected_10km.dissolve(by='dissolve').reset_index(drop=True)
            water_connected_10km =
gpd.overlay(water_connected_10km,model_domain,how='intersection')


water_connected_10km.to_file("%s%s_water_connected_10km_%s.shp"
%(folders[7],delta,xres))
            print('#################### 10km water connected to ocean
polygons saved to %s%s_water_connected_10km_%s.shp'
%(folders[7].split(delta)[-1],delta,xres))
```

```
######################################################################
        ## FILL IN WATER BODIES FOR MESH GENERATION

######################################################################
        print('\n[Step 6][Make_Mesh_Polygons][Get points along all water
body polygons] .......\n')
        try: points_w=gpd.read_file("%s%s_water_points_%s.shp"
%(outdir,delta,xres))
        except:
            water_filled, points_w =
getpolygonpoints(allwater,ulx,uly,lrx,lry,100)
            water_filled =
water_filled.explode(index_parts=True).reset_index(drop=True)
            points_w.to_file("%s%s_water_points_%s.shp"
%(outdir,delta,xres))
            water_filled.to_file("%s%s_waterfilled_%s.shp"
%(outdir,delta,xres))
        print('\n[Step 6][Make_Mesh_Polygons][Delete any holes within
polygons] .......\n')

        try: water_filled=gpd.read_file("%s%s_waterfilled_%s.shp"
%(outdir,delta,xres))
        except:
            water_no_holes,holes = delete_holes(water_filled)
            water_filled =
gpd.overlay(water_filled,water_no_holes,how='union')
            water_filled.to_file("%s%s_waterfilled_%s.shp"
%(outdir,delta,xres))

        try: river_no_holes=gpd.read_file("%s%s_river_mesh_%s.shp"
%(outdir,delta,xres))
        except:
            rivers = rivers[rivers['DN']!=0]
            river_no_holes, holes = delete_holes(rivers)
            river_no_holes.geometry =
river_no_holes.buffer(round(med_width/cellsperwidth))
            river_no_holes =
gpd.overlay(river_no_holes,model_domain,how='intersection')
            river_no_holes.to_file("%s%s_river_mesh_%s.shp"
%(outdir,delta,xres))

        if len(lakes)>1:
            try:lake_no_holes=gpd.read_file("%s%s_lake_mesh_%s.shp"
%(outdir,delta,xres))
            except:
                lakes_no_holes = lakes_no_holes.reset_index(drop=True)
                lake_no_holes, holes = delete_holes(lakes)
                lake_no_holes =
gpd.overlay(lake_no_holes,river_no_holes,how='difference')
                lake_no_holes =
gpd.overlay(lake_no_holes,model_domain,how='intersection')
                lake_no_holes.to_file("%s%s_lake_mesh_%s.shp"
%(outdir,delta,xres))
```

```
        try: ocean_no_holes=gpd.read_file("%s%s_fullocean_mesh_%s.shp"
%(outdir,delta,xres))
        except:
            ocean_no_holes, holes = delete_holes(oceans)
            ocean_no_holes =
gpd.overlay(ocean_no_holes,river_no_holes,how='difference')
            ocean_no_holes = ocean_no_holes[ocean_no_holes.area>12000]
            ocean_no_holes =
gpd.overlay(ocean_no_holes,model_domain,how='intersection')
            ocean_no_holes.to_file("%s%s_fullocean_mesh_%s.shp"
%(outdir,delta,xres))

        nearshore_oceans.geometry = nearshore_oceans.buffer(-10*xres)
        try:
nearshore_no_holes=gpd.read_file("%s%s_nearshore_mesh_%s.shp"
%(outdir,delta,xres))
        except:
            nearshore_no_holes, holes = delete_holes(nearshore_oceans)
            nearshore_no_holes =
gpd.overlay(nearshore_no_holes,river_no_holes,how='difference')
            nearshore_no_holes =
gpd.overlay(nearshore_no_holes,model_domain,how='intersection')
            nearshore_no_holes =
nearshore_no_holes.drop(columns=['FID_1','FID_2'])
            nearshore_no_holes.to_file("%s%s_nearshore_mesh_%s.shp"
%(outdir,delta,xres))

        try: farocean_no_holes=gpd.read_file("%s%s_farocean_mesh_%s.shp"
%(outdir,delta,xres))
        except:
            farocean_no_holes, holes = delete_holes(farocean)
            farocean_no_holes =
gpd.overlay(farocean_no_holes,river_no_holes,how='difference')
            farocean_no_holes =
gpd.overlay(farocean_no_holes,model_domain,how='intersection')
            farocean_no_holes =
farocean_no_holes.drop(columns=['FID_1','FID_2'])
            farocean_no_holes.to_file("%s%s_farocean_mesh_%s.shp"
%(outdir,delta,xres))

        try:
fulloceans_no_holes=gpd.read_file("%s%s_fulloceans_mesh_%s.shp"
%(outdir,delta,xres))
        except:
            fulloceans_no_holes, holes = delete_holes(oceans)
            fulloceans_no_holes =
gpd.overlay(fulloceans_no_holes,river_no_holes,how='difference')
            fulloceans_no_holes =
gpd.overlay(fulloceans_no_holes,model_domain,how='intersection')
            fulloceans_no_holes =
fulloceans_no_holes.drop(columns=['FID_1','FID_2'])
            fulloceans_no_holes.to_file("%s%s_faroceans_mesh_%s.shp"
%(outdir,delta,xres))
```

```
        try: land_no_holes=gpd.read_file("%s%s_land_mesh_%s.shp"
%(outdir,delta,xres))
        except:
            land_no_holes, holes = delete_holes(lands)
            land_no_holes =
gpd.overlay(land_no_holes,river_no_holes,how='difference')
            land_no_holes =
gpd.overlay(land_no_holes,model_domain,how='intersection')
            land_no_holes.to_file( "%s%s_land_mesh_%s.shp"
%(outdir,delta,xres))


        meshes = ['river','ocean','land','lake']

        print('##################### Ocean Mesh Polygons saved to
%s%s_ocean_mesh.shp' %(outdir,delta))
        print('##################### Land Mesh Polygons saved to
%s%s_land_mesh.shp' %(outdir,delta))
        print('##################### River Mesh Polygons saved to
%s%s_river_mesh.shp' %(outdir,delta))
        print('##################### Lake Mesh Polygons saved to
%s%s_lake_mesh.shp' %(outdir,delta))
        print('\n[Step 6][Make_Mesh_Polygons] Finished .......\n')
    else:
        print('\n[Step 6][Make_Mesh_Polygons] SKIP .......\n')
        meshes = ['river','ocean','land','lake']

    return meshes

def
determine_riverscale(folders,delta,ref,parameters,pixel_step,skip=False):

    '''

    Parameters
    ----------
    parameters : pd.dataframe
        Configuration parameters for model setup.
    skip : boolean
        If True, do not build water polygons.

    Returns
    -------
    distance : np.array
        Distances from ocean along river channels.
    widths : np.array
        Widths of river cross sections, approximately every 150m
    riverscale : float
        Maximum triangle area for generating mesh in rivers.
    med_width : float
        median width of river channels.
    cellsperwidth : float
        minimum number of cells per channel width.
    watermask : np.array
        Mask of water/land.

    '''


print('\n\n\n###########################################################
#################################')
    print('######################################## [River Scale]
######################################')

print('#################################################################
############################\n')
    save_profile = ref.profile
    save_profile['compress'] = 'deflate'

    xres,yres = int(save_profile['transform'][0]),-
int(save_profile['transform'][4])

########################################################################
##
    ###################### Import Config Parameters
###########################

########################################################################
##

    parameters.iloc[0]
    EPSG = parameters['EPSG'][0]
# Coordinate System must be UTM
    ulx  = parameters['ulx'][0]
# ULX coordinate
    lry  = parameters['lry'][0]
# LRY coordinate
    lrx  = parameters['lrx'][0]
# LRX coordinate
    uly  = parameters['uly'][0]
# ULY coordinate
    try: os.mkdir(folders[1])
    except:''


########################################################################
#
    ## RASTERIZE WATER/LAND DELIENATI SHAPEFILES

########################################################################
#

    river = rasterio.open('%s%s_rivers_%s.tif' %(folders[8],delta,xres))
    rivermask = np.where(river.read(1) ==1,1.,0.)

    widths = np.where((rivermask==1),gdal.Open('%s%s_widths_%sx%s.tif'
%(folders[8],delta,xres,pixel_step)).ReadAsArray(),np.nan)
```

```
    os.system('gdal_rasterize -burn 1 -tr %s %s -te %s %s %s %s
%s%s_river_centerline_%sx%s.shp %s%s_river_centerline_%sx%s.tif -co
COMPRESS=DEFLATE'
%(xres,yres,ulx,lry,lrx,uly,folders[7],delta,xres,pixel_step,folders[8],d
elta,xres,pixel_step))

    print('\n[Get Riverscale][Get Median Width of Rivers] .......\n')
    avg_width = np.nanmean(widths)
    med_width = np.nanmedian(widths)
    print('##################### Median river width is %sm' %(med_width))

    print('\n[Get Riverscale][How many cells per width] .......\n')
    if med_width >= 1500.:
        cellsperwidth = 10
    else: cellsperwidth = 5
    print('##################### Cells per river width is %s'
%(cellsperwidth))

    print('\n[Get Riverscale][Estimate maximum triangle size to maintain
%s per median %sm river width] .......\n' %(cellsperwidth, med_width))
    riverscale = int((med_width /cellsperwidth)**2/2)
    print('##################### Therefore, max triangle scale is %sm2'
%(riverscale))

    return med_width, riverscale, cellsperwidth
```