""" estimate footprint values (at the 34-kt radius) for the ocean skill metrics -

var=OHC

Z26

MLD\_temp\_crit

MLT\_temp\_crit

MLD\_dens\_crit

MLT\_dens\_crit

T100

PEA

and

returns a set of plots of a var.

Libraries for the metrics:

from Upper\_ocean\_metrics import OHC\_from\_profile,

MLD\_temp\_crit,

MLD\_dens\_crit,

T100,

Potential\_energy\_anomaly100

--------

history:

--------

edited by hsk 5/9/2021 """

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

#%% Frist, user inputs for a storm of interest, tcID, and cycle

storm='isaias'

tcid='09l'

var='OHC'

i2pickl = True

cycle='2020080112'

# Load Python modules and libraries

import xarray as xr

import numpy as np

import netCDF4

from datetime import datetime, timedelta

import seawater as sw

from utils4HWRF import readTrack6hrly, Rwinds6hr, btRwinds

from geoutils import haversine

import os, sys, glob

import pandas as pd

import matplotlib.pylab as plt

from Upper\_ocean\_metrics import footprint\_OHC

from utils import coast180, hsk\_Spectral\_r

from pathlib import Path

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

# Find BT coincide with a cycle of interest

# Bdeck

graphdir='/work/noaa/hwrf/save/hskim/Isaias/scripts/footprint/output/GOFS3p1/'+cycle

if not os.path.isdir(graphdir):

p=Path(graphdir)

p.mkdir(parents=True)

abdeck='/work/noaa/hwrf/noscrub/input/abdeck'

btk=os.path.join(abdeck,'btk')+'/bal'+tcid[:2]+'2020.dat'

bdn,bln,blt,bpmn,bvmx=readTrack6hrly(btk)

bln=-1\*bln+360

cx,cy=coast180()

cx=cx+360

#%% 3-hourly 1/12-degree global ocean anal GOFS3.1

# NCODA url and download dataset

url\_anal = 'http://tds.hycom.org/thredds/dodsC/GLBy0.08/expt\_93.0/ts3z'

anal = xr.open\_dataset(url\_anal,decode\_times=False)

analx,analy = np.meshgrid(anal.lon.data,anal.lat.data)

adum=np.ones(analx.shape)\*np.nan

ttm = anal.time

tm = netCDF4.num2date(ttm[:],ttm.units)

#---------- building dataframe for each footprint

#names=['YMDH','lon','lat','SST','SSS','OHC','Z26','MLDTc','MLTTc','MLDdc','MLTdc','PEA']

#c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13=[],[],[],[],[],[],[],[],[],[],[],[],[]

ac1,ac2,ac3,ac4,ac5,ac6,ac7=[],[],[],[],[],[],[]

names=['YMDH','lon','lat','SST','SSS','OHC']

DF=pd.DataFrame()

knt=0

plt.close('all')

ymdh=datetime.strptime(cycle,'%Y%m%d%H')

fi=np.where(bdn==ymdh)

for N in np.arange(fi[0],len(bdn),1):

dfc=pd.DataFrame()

D=bdn[N]

dR=haversine(analx,analy,bln[N],blt[N])/1000.

ymdh=datetime.strftime(D,'%Y%m%d%H')

if knt==0:

cycle=ymdh

# find a set of radii for the 34-, 50- and 64-kt wind, and apply the 34-kt wind radius as for a footprint

brads=btRwinds(btk,ymdh)

if brads[0] != 0:

dR[dR>brads[0]]=np.nan

# Find the lat and lon index in the GOFS grid

ii=np.where(~np.isnan(dR))

i0=np.unique(ii[0])

i1=np.unique(ii[1])

ni0=len(i0)

ni1=len(i1)

# Footprint (x,y)

lns,lts=np.meshgrid(analx[0,i1],analy[i0,0])

one=dR/dR

oneq=np.nan\*np.ones([ni0,ni1])

for r,R in enumerate(i0):

for c,C in enumerate(i1):

oneq[r,c]=one[R,C]

#------ each forecast lead time @6 hourly

# Find the time index

oktimem = np.where(tm == D)[0]

time\_anal = tm[oktimem[0]]

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

#%% estimate (OHC.Z26,MLD\_temp\_cr,MLT\_temp\_cr,MLD\_dens\_cr,MLT\_dens\_cr,T100,PEA)

# at each grid cell on a footprint

# mixed depth by a temperature criteria

dtemp = 0.2

ref\_depth = 10 # meters

# mixed depth by a density criteria

drho = 0.125

ref\_depth = 10 # meters

Ta=np.squeeze(anal.water\_temp[oktimem[0],:,i0,i1].data)

Sa=np.squeeze(anal.salinity[oktimem[0],:,i0,i1].data)

Za=np.tile(anal.depth.data,[ni0,ni1,1])

Za=Za.transpose(2,0,1)

Da=sw.dens(Sa,Ta,Za)

o=footprint\_OHC(Za,Ta,Da)

OHC=o\*oneq

ac1.extend(np.tile(D,[ni0\*ni1,1]).flatten())

ac2.extend(lns.flatten())

ac3.extend(lts.flatten())

ac4.extend(np.squeeze(Ta[0,:,:]).flatten()) # SST

ac5.extend(np.squeeze(Sa[0,:,:]).flatten()) # SST

ac6.extend(OHC.flatten()) # OHC

if np.mod(knt,9)==0:

plt.figure(figsize=(13,10))

pn=plt.gcf().number

plt.subplot(3,3,1)

plt.contourf(lns,lts,OHC\*oneq,levels=np.arange(0,160,10),extend='max',cmap=hsk\_Spectral\_r()); plt.colorbar()

plt.title('(1) '+ymdh)

plt.plot(bln,blt,'-ko',markersize=3); plt.plot(cx,cy,'gray',alpha=0.6)

plt.axis([lns.min(),lns.max(),lts.min(),lts.max()])

plt.text(lns.mean()+(lns.max()-lns.min())/4,lts.min()+0.08,"%8.2f"%(np.nanmean(OHC)),fontsize=10)

else:

plt.figure(pn)

h=np.mod(knt,9)+1

print("knt & h="+str(knt)+', '+str(h))

plt.subplot(3,3,h)

plt.contourf(lns,lts,OHC\*oneq,levels=np.arange(0,160,10),extend='max',cmap=hsk\_Spectral\_r()); plt.colorbar()

plt.title('('+str(h)+') '+ymdh)

plt.plot(bln,blt,'-ko',markersize=3); plt.plot(cx,cy,'gray',alpha=0.6)

plt.axis([lns.min(),lns.max(),lts.min(),lts.max()])

plt.text(lns.mean()+(lns.max()-lns.min())/4,lts.min()+0.08,"%8.2f"%(np.nanmean(OHC)),fontsize=10)

plt.suptitle('GOFS3.1: \n'+storm.upper()+' OHC [kJcm$^{-2}$]',fontsize=13)

# dataFrame

dict={'YMDH':ac1, 'lon [E]':ac2, 'lat [N]':ac3, 'SST [C]':ac4, 'SSS [psu]':ac5, 'OHC [kJ/cm^2]':ac6}

dfc = pd.DataFrame.from\_dict(dict, orient='columns')

DF=DF.append(dfc)

knt+=1

print('OHC estimates from GOFS3.1 on a footprint done for forecast hour '+str(N\*6))

for p in np.arange(1,pn,1):

plt.figure(num=p)

plt.savefig(graphdir+'/GOFS3p1\_'+storm+tcid+'\_'+cycle+'\_OHC\_fig'+str(p)+'.png',bbox\_inches='tight')

# --- option to save

if i2pickl:

dfc.to\_pickle(graphdir+'/'+storm.upper()+tcid.upper()+'\_OHC.pkl')

print('save data to '+graphdir)

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