```
import numpy as np
from astropy.io import fits
import glob
from reproject.mosaicking import find_optimal_celestial_wcs
#from drizzlepac.astrodrizzle import Drizzle

from drizzle import drizzle
import threading
import time
from copy import copy
from astropy.wcs import WCS
import os
home = os.getenv('HOME')
import sep

## Understand the Code, not sure what this is meaning...

# This is the only thing that needs to switch depending on if you're working at the lab
# or on your own computer
onedrive = r"C:\\Users\\aka188\OneDrive - Simon Fraser University (1sfu)"

# Use the "M37-calibrated" directory that you created in Assignment3 (or use my
# calibrated data in the shared folder under Assignment 4)
#data_dir = onedrive+ r'\PHYS391-Spring-2025-private\Assignment3\Solutions'

#calibrated_dir = data_dir+ r'\M37-calibrated'
calibrated_dir = r"C:\Users\aka188\OneDrive - Simon Fraser University (1sfu)\PHYS391-Spring-2025-
# This is your private directory.  You should save this python script there too.
working_dir = onedrive + '/PHYS391-Spring-2025-private/Assignment5'

#-------------------------------------------------------------------------------------
crop = slice(160,4340),slice(120,5750) # How I cropped the calibrated files


weightfile = working_dir + r'\weight_BIN2.fits'  # the file you created in badpixels.py
weight = fits.open(weightfile)[0].data # get the data
weight = weight[crop] # crop the weights so to the same area as the calibrated data

# Find the optimal size of an image that will fit all the images (follow Lecture 6)
filelist = glob.glob(calibrated_dir + "/*.fit") ## Gets all Fits file
wcs_out, shape_out = find_optimal_celestial_wcs(filelist, auto_rotate = True)
wcs_out.array_shape = shape_out
print('Dimensions of new canvas:',shape_out) # should be bigger than the original shape

bands = ['PhotB','PhotV','PhotR'] # This is the list of the three filter names
F,B = 5,512  ## Your favourite values for F and B from Assignment 4

def alignCombine(band):

    # Get the list of the files that were exposed in the current band
    # This list SHOULD NOT contain ALL the files.  Only the ones in the band you want.
    filelist = glob.glob(calibrated_dir+'\*'+band+'*') # Use *'s as wild cards

    filelist.sort()
    driz = drizzle.Drizzle(outwcs = wcs_out) # create a Drizzle object
    sky_means = [] # a list to collect the mean values of the sky that we subtract
    for f in filelist:# When testing, just use filelist[0:3] to run on the first 2
        hdu = fits.open(f) # Open the current file
        orig_wcs = WCS(hdu[0].header) # Get the WCS of this image
        texp = hdu[0].header["EXPTIME"] # Get exposure time
        binning = hdu[0].header["XBINNING"] # Get the binning in one dimension
        image = hdu[0].data # get the data

        image_data = image.byteswap(inplace=False).newbyteorder()
        image = image_data # Turn it in ADU/s (Drizzle requires this)
        ## Later told not to convert to ADU/s. Keep ADU

        # determine the sky level as a function of position
        sky = sep.Background(image)
        sky_data = sky.back()
        image_nosky = image - sky_data

        # Since the sky is pretty uniform on all the images, let's just save the mean
        # values of the sky to be used for later for noise estimates, instead of
        # drizzling the sky to the new frame (which takes a long time)
        sky_mean = np.mean(sky_data)
        sky_means.append(sky_mean)

        # Run Drizzle's add_image() (follow Lecture 6)
        driz.add_image(image_nosky, orig_wcs , inwht = weight, expin = texp )

    # Get the mean of the sky means list
    sky_mean = np.mean(sky_means)

    # Create a fits header object that contains the exposure time and skymean
    hdu = fits.PrimaryHDU()
    hdu.header['EXPINDIV'] = (texp,'Exposure time of individual exposures')
    hdu.header['SKYMEAN'] = (sky_mean,'Mean sky value across all exposures')
    hdu.header['BINNING'] = (binning,'Binning of the pixels for the original frames')

    # Write the drizzled image to file, using the fits header we created above
    driz.write(f'M37_{band}.fit',outheader=hdu.header)


t1 = time.time()
# If threading is giving you trouble, just uncomment this:
# for band in bands: alignCombine(band) # 44 min on lab computers
#
# And comment the next 3 lines
threads = [threading.Thread(target=alignCombine,args=(band,)) for band in bands]
for t in threads: t.start()
for t in threads: t.join() # this means wait until the thread finishes before running
                           # the next line (ie, the next item in the for loop.)
                           # 31 min on lab computers

print(f'alignCombine() took {(time.time()-t1)/60.0} minutes')
```