```python
# -*- coding: utf-8 -*-
"""
created on: 2024-03-26
@author:    Jasper Heuer
use:        1) caclulate ratio of no data pixels
            2) crop imagery to glacier extent
            3) classify glacier surface into snow/ice/bedrock based on simple threshold values
            4) detect border pixels between classes
"""

# import packages ==============================================================

import os
import glob
import time
import rasterio
import numpy as np
from osgeo import gdal
from rasterio.plot import show

# define variables ============================================================

base_path = "C:/Jasper/Master/Thesis/Data/"
os.chdir(base_path)

start_time = time.time() # set start time

# create new directories:
path_reclassified = "./Landsat/Reclassified/"
path_cropped = "./Landsat/Cropped/"
path_borders = "./Landsat/Borders/"

if not os.path.exists(path_reclassified):
    os.makedirs(path_reclassified)
if not os.path.exists(path_cropped):
    os.makedirs(path_cropped)
if not os.path.exists(path_borders):
    os.makedirs(path_borders)

# define file names:
fn_mask = "./Masks/mittivakkat_outline.shp"
fn_raster_mask = "./Masks/mask.tif"
study_area_out = "./Masks/no_data_ratio.tif"

# define meta data:
dst_crs = "EPSG:32624"
res = 30

# calculate number of no data pixels ===========================================

# create raster where glacier area pixels = 1 and other pixels = 0
study_area = gdal.Open(fn_raster_mask)
study_crop = gdal.Warp(study_area_out, study_area,
                       cutlineDSName=fn_mask,
                       cropToCutline=True,
                       dstNodata=0)

# read data:
no_data_ds = rasterio.open(study_area_out)
no_data_full = no_data_ds.read()

# count 0- and 1-pixels:
no_data_classes, no_data_counts = np.unique(no_data_full, return_counts=True)
no_data_dict = dict(zip(no_data_classes, no_data_counts))
no_data_sum = sum(no_data_counts)

# calculate ratio between 0-pixels and total number of pixels:
no_data_base_ratio = no_data_dict[0] / no_data_sum

# batch crop to Mittivakkat mask ===============================================

```

```
file_list = glob.glob("./Landsat/Resampled/" + "*.tif", recursive=True)

# create list of dates:
date_list = []

for i in range(0, np.size(file_list)):
    date_list.append(file_list[i].split("\\")[1][0:13])

# batch crop imagery:
for i in range(0, np.size(file_list)):
    img_in = file_list[i]
    img_out = "./Landsat/Cropped/" + str(date_list[i]) + "_cropped.tif"

    img_ds = gdal.Open(img_in)

    # crop imagery to study area:
    img_crop = gdal.Warp(img_out, img_ds,
                         cutlineDSName=fn_mask,
                         cropToCutline=True,
                         dstNodata=np.nan)

    img_ds = None
    img_crop = None
    img_out = None

    print("Cropped: " + str(date_list[i]))

# batch classify surface ═══════════════════════════════════════════════════════

file_list2 = glob.glob("./Landsat/Cropped/" + "*.tif", recursive=True)

# create list of dates:
date_list2 = []
for i in range(0, np.size(file_list2)):
    date_list2.append(file_list2[i].split("\\")[1][0:13])

# batch reclassify imagery/detect pixels:
for i in range(0, np.size(file_list2)):
    if date_list2[i][0:4] == "LC08" or date_list2[i][0:4] == "LC09":
        img_ds = rasterio.open(file_list2[i])
        img_full = img_ds.read()
        band_01 = img_ds.read(2)
    else:
        img_ds = rasterio.open(file_list2[i])
        img_full = img_ds.read()
        band_01 = img_ds.read(1)

    # relassify surface based on pixel threshold:
    reclass = np.where((band_01 > 0.55), 3,
                       np.where((band_01 ⩾ 0.075), 2,
                                np.where((band_01 > 0), 1,
                                         np.where((band_01 == np.nan), 0, 0))))

    # check ratio of no data pixels to number of pixels
    classes, counts = np.unique(reclass, return_counts=True)
    classes_dict = dict(zip(classes, counts))
    number_of_pixels = sum(counts)

    no_data_ratio = classes_dict[0] / number_of_pixels


    # filter by amount of no data in study area:
    if no_data_ratio > (no_data_base_ratio + ((1 - no_data_base_ratio) * 0.2)):
        print("Skipped image: " + str(file_list2[i]).split("\\")[1][0:13])
        pass # if more than 20% of the image contains no data, skip image

    else:
        show(reclass) # plot reclass array

        # detect pixels:
        borders = np.zeros_like(band_01, dtype=int)
        x_size, y_size = np.shape(borders)[0], np.shape(borders)[1]

```

```
        # reclassify pixels based on border-type:
        for k in range(x_size):
            for j in range(y_size):
                # select class 1 (bedrock):
                if reclass[k,j] == 1:
                    slice = reclass[k-1:k+2,j-1:j+2] # get k1 neighborhood
                    if np.any(slice == 2):
                        borders[k,j] = 4 # update border pixel value
                    if np.any(slice == 3):
                        borders[k,j] = 5
                # select class 2 (ice):
                if reclass[k,j] == 2:
                    slice = reclass[k-1:k+2, j-1:j+2]
                    if np.any(slice == 3):
                        borders[k,j] = 6
                    if np.any(slice == 1):
                        borders[k,j] = 7
                # select class 3 (snow):
                if reclass[k,j] == 3:
                    slice = reclass[k-1:k+2, j-1:j+2]
                    if np.any(slice == 2):
                        borders[k,j] = 8
                    if np.any(slice == 1):
                        borders[k,j] = 9

        # show(borders) # plot border array

        # save reclass array as GeoTiff:
        with rasterio.open(
            "./Landsat/Reclassified/" + str(date_list[i]) + "_reclass.tif",
            mode="w",
            driver="GTiff",
            height=img_full.shape[1],
            width=img_full.shape[2],
            count=img_full.shape[0],
            dtype=img_full.dtype,
            crs=img_ds.crs,
            transform=img_ds.transform
            ) as dst:
                dst.write(reclass, 1)

        # save border array as GeoTiff:
        with rasterio.open(
            "./Landsat/Borders/" + str(date_list[i]) + "_borders.tif",
            mode="w",
            driver="GTiff",
            height=img_full.shape[1],
            width=img_full.shape[2],
            count=img_full.shape[0],
            dtype=img_full.dtype,
            crs=img_ds.crs,
            transform=img_ds.transform
            ) as dst:
                dst.write(borders, 1)

        print("Reclassified: " + str(date_list[i]))

# print duration:
print(f"Duration: {time.time() - start_time} seconds")

```