Step 2.4 Computing GLCM features

Diane ESPEL

2025-10-31

Contents

1 Objectives

This script is designed to extract textural features from multispectral satellite imagery by computing Haralick indices based on the **Gray Level Co-occurrence Matrix (GLCM)** method. The goal is to characterize the spatial patterns and texture of spectral indices (e.g., NDVI) within defined moving windows, enhancing habitat mapping by capturing structural information beyond spectral values alone.

The script automates:

- loading spectral index rasters,
- defining calculation windows based on spatial resolution and plot size,
- computing multiple GLCM texture metrics,
- and saving each resulting texture layer for further analysis.

GLCM (Gray Level Co-occurrence Matrix) texture features quantify the spatial relationships of pixel values in an image, providing insight into texture complexity, uniformity, and variation. This script calculates six Haralick indices:

- **Homogeneity:** Measures similarity among neighboring pixel pairs; higher values indicate more uniform texture.
- **Contrast:** Captures intensity differences between adjacent pixels; higher values reflect greater texture variation.
- **Dissimilarity:** Quantifies how different pixel pairs are; a higher score means more variation.

- **Entropy:** Represents randomness or complexity in the texture; higher entropy indicates more unpredictable patterns.
- **Angular Second Moment (ASM):** Indicates texture uniformity and energy; higher values correspond to more ordered patterns.
- **Correlation:** Measures linear dependency between pixel pairs; higher correlation means stronger spatial relationships.

2 Script explanation

2.1 Clean environment and graphics

```r
rm(list = ls()) # Clear all objects from the R environment to start fresh
graphics.off()   # Close all graphics devices (if any plots are open)
```

2.2 Load required packages

```r
library(GLCMTextures)  # For computing Gray Level Co-occurrence Matrix (GLCM) texture features
↪  (Haralick indices)
library(terra)         # For raster data manipulation and spatial analysis
```

2.3 Define global variables

Note: A global variable is a variable defined outside of any function. This means the variable is accessible from any part of the code, including inside functions. A global variable retains its value throughout the execution of the R script unless it is explicitly modified in the code.

It is important to define at a minimum:

- the "District": the archipelago of interest (e.g. "CRO" for Crozet archipelago)
- the "Island": the island within the archipelago of interest (e.g. "POS" for Possession island)
- the "Satellite1": the name of satellite providing multispectral imagery
- the "Res1": the resolution of the multispectral imagery

```r
District='CRO' # 3-letter code for archipelago (e.g. Crozet)
Island='POS'   # 3-letter code for island (e.g. Possession)
Satellite1="Pleiades" # satellite name of multispectral imagery
Res1 = "50cm"  # spatial resolution of multispectral imagery
```

2.4 Set working directory

To optimize memory, you must define one general root directory ("localHOME") that serves as the base path for your input and output data, respectively. This directory should point to the local environment where:

- input spectral index is located under "data/raster/Cut_image"
- outputs glcm features will be saved under "data/raster/Cut_image"

```r
# Base local path (customize to your local environment)
#localHOME = paste0("your_local_path/")
localHOME=paste0("/home/genouest/cnrs_umr6553/despel/CARTOVEGE/")

# Path to open input index raster
open_cut_raster_path=paste0(localHOME,"data/raster/Cut_image")

# Path to save texture rasters
save_cut_raster_path=paste0(localHOME,"data/raster/Cut_image")
```

2.5 Compute GLCM features for available rasters

The function `glcm_textures()` computes texture features from a raster image based on the Gray Level Co-occurrence Matrix (GLCM) approach, which captures spatial relationships between pixel intensities to describe image texture.

- Input raster: The function takes a single-layer raster (spectral_raster[[1]]), typically a spectral index like NDVI. If the raster has multiple bands, you extract just one layer for the calculation.
- Window size (`w`): This defines the size of the moving square window (e.g., 3x3, 5x5 pixels) over which the GLCM is computed. The texture is analyzed locally within this window, capturing spatial patterns at that scale.
- Number of grey levels (`n_levels`): The raster pixel values are discretized into a specified number of grey levels (here, 16). This reduces the range of pixel values into bins, allowing for manageable computation of co-occurrence probabilities. A higher number of levels preserves more detail but increases computational cost.
- Quantization method (`quantization`): Determines how pixel values are binned into grey levels. The "equal prob" method distributes pixel values so that each grey level contains roughly an equal number of pixels, balancing the representation of intensity values and enhancing texture discrimination. Other options include "uniform" (equal width bins) or "linear".
- Shift directions (`shift`): Specifies the pixel offsets to consider when forming pairs for the GLCM. Each pair consists of a pixel and its neighbor offset by these vectors:
  - (1, 0) vertical neighbors (90°),
  - (1, 1) diagonal neighbors (45°),
  - (0, 1) horizontal neighbors (0°),
  - (-1, 1) diagonal neighbors (135°). The default textures are calculated using a 45 degree shift (c(1,1)); the default is to return directionally/rotationally invariant textures that are averaged across all 4 directions

Calculating texture across multiple directions and averaging results produces rotationally invariant texture measures, meaning the features are consistent regardless of image orientation.

```r
# List of years and months
all_years <- c("2025","2024","2023","2022","2021","2020","2015","2011")
all_months <- c("01","02","03","04","05","06","07","08","09","10","11","12")

# Define selected raster
selected_raster="ndvi"

# Run the loop on available rasters
```

```r
for (Year1 in all_years){

  print(paste0("Year: ", Year1))

  for (Month1 in all_months) {
    print(paste0("Month: ", Month1))

    # define raster path
    raster_path<- file.path(open_cut_raster_path, paste0(District,"_",Island,"_",Satellite1,
 ↪  "_",Year1,"_",Month1,"_",Res1, "_",selected_raster,"_cut.TIF"))

    if (!file.exists(raster_path)) {
      message("  -> Raster not found: ", raster_path, "  (skipping)")
      next
    }

    # open availlable raster
    spectral_raster <- rast(raster_path)

    # Compute textural features----------------------------------------------------------

    # Define window size for GLCM
    print("Defining calculation window; must be odd and larger than training plot size")

    radius=2 # radius of learning plots = 2 m
    res_pleiades=res(spectral_raster)[1] # # Spatial resolution of raster (assumed 0.5m)
    nb_pixels_radius=radius/res_pleiades # Convert radius from meters to number of pixels
    npix <- round(nb_pixels_radius + 1) # Number of pixels in window dimension (rounded and
 ↪  incremented)
    window_size=c(npix,npix) # Window size as number of rows and columns (must be square)

    # computing GLCM matrix
    print("6 Haralick indices")

    # apply glmc_textures() function
    GLCM_textures <- GLCMTextures::glcm_textures(spectral_raster[[1]], # SpatRaster (in case
 ↪  nlyr(spectral_raster)>1,spectral_raster[[1]] ; if not, write only spectral_raster
                                                 w = window_size,  # The size of the window used to
                                                  ↪  compute pairs of pixels for the GLCM (Gray
                                                  ↪  Level Co-occurrence Matrix)
                                                 n_levels = 16, # The number of grey levels used to
                                                  ↪  discretize the image. A higher number means
                                                  ↪  more detail in texture representation.
                                                 quantization = "equal prob", # Method of
                                                  ↪  quantizing the pixel values. "equal prob"
                                                  ↪  assigns equal probability to each grey level
                                                  ↪  during quantization.
                                                 # This simplifies the texture analysis, but other
                                                  ↪  methods like "uniform" or "linear" could also
                                                  ↪  be used for different quantization schemes.
                                                 shift = list(c(1, 0), # (1,0) corresponds to a
                                                  ↪  vertical direction (90° shift)
                                                              c(1, 1), # (1,1) corresponds to a
                                                               ↪  diagonal direction (45° shift)
                                                              c(0, 1), # (0,1) corresponds to a
                                                               ↪  horizontal direction (0° shift)
                                                              c(-1, 1) # (-1,1) corresponds to a
                                                               ↪  direction of 135° shift
                                                 ),  #The default textures are calculated using a
 ↪  45 degree shift (c(1,1)); the default is to return directionally/rotationally invariant
 ↪  textures that are averaged across all 4 directions
```

```
                                               metrics = c("glcm_homogeneity", # Measure of the
                                                ↪  similarity of pixel pairs in the GLCM; higher
                                                ↪  values indicate more uniformity in the
                                                ↪  texture.
                                                           "glcm_contrast",     # Measure of the
                                                            ↪  contrast or difference between the
                                                            ↪  values of neighboring pixels;
                                                            ↪  higher values indicate more
                                                            ↪  variation in texture.
                                                           "glcm_dissimilarity", # Measure of how
                                                            ↪  different two pixel values are;
                                                            ↪  higher values indicate more
                                                            ↪  dissimilarity.
                                                           "glcm_entropy",      # Measure of the
                                                            ↪  randomness or complexity in the
                                                            ↪  texture; higher values indicate
                                                            ↪  more complex or unpredictable
                                                            ↪  patterns.
                                                           "glcm_ASM",          # Angular Second
                                                            ↪  Moment; a measure of texture
                                                            ↪  uniformity.
                                                           "glcm_correlation"   # Measure of the
                                                            ↪  linear dependency between pixel
                                                            ↪  pairs; higher values indicate more
                                                            ↪  correlation in texture.
                                               )
   )
   # Rename output layers for clarity

    ↪  names(GLCM_textures)=c("Homogeneity","Contrast","Dissimilarity","Entropy","Second_moment","Correlation")



   #  Save computed Haralick texture
    ↪  layers---------------------------------------------------------


   # Extract each texture layer by looping through the names in the GLCM_textures list
   for (name_texture in names(GLCM_textures)) {

     print(paste0("Saving texture index: ", name_texture))  # Print the texture name that is
      ↪  currently being processed
     texture_layer <- GLCM_textures[[name_texture]]  # Extract the texture layer using the name
↪  (from GLCM_textures list)

     # Save the texture layer to a file
     FILE2 <- paste0(save_cut_raster_path, "/", District, "_", Island, "_", Satellite1, "_",
↪  Year1, "_", Month1, "_", Res1, "_", name_texture, "_cut.TIF")
     writeRaster(texture_layer, FILE2, overwrite = TRUE)  # Write the texture layer to disk as a
      ↪  GeoTIFF file

   }


   print("All texture indices have been successfully saved.")



 }  # end of month loop
} # end of year loop
```

This code block iterates through all texture layers stored in the GLCM_textures list and saves each one as a separate raster file. For every texture index, it prints the name of the texture being processed, retrieves the corresponding raster layer, and writes it to disk as a GeoTIFF file.