Example Code and Workflow

Duffy et al. (2017) Spatial assessment of intertidal seagrass meadows using optical imaging systems and a lightweight drone. *Estuarine, Coastal and Shelf Science.*

06/11/17

TODO

- Add crop and store to functions folder and source in relevant scripts to avoid repeating?
- Adapt statistics script/section from count pixels in quadrats script.

Introduction

This project contains a set of example scripts and associated folder structure to run some of the processing and analysis code associated with the above paper. Data was collected with a consumer grade camera mounted on a drone platform, the photos were processed and stitched using Agisoft Photoscan. Three classification procedures were undertaken and presented in the paper. Details of each of these processes are found within the scripts in this repository.

A few files are required as inputs:

- An RGB orthomosaic (normally created in Photogrammetry software, e.g. GeoTiff). In this example the file is `Angle_Ricoh_BNG_PS2_Ortho_Crop.tif`.
- A vector polygon with the boundary of the orthomosaic (e.g. shapefile). In this example the file is `Angle_Ricoh_Mask.shp`.
- A vector set of polygons with the boundaries of ground based quadrats (e.g. shapefile). In this example the file is `Angle_Quadrats_Poly_Crop.shp`.
- A .csv file with the recorded seagrass coverage values from the ground based surveys. In this examples the file is `Angle_Quadrat_Data.csv`. Columns describing the quadrat ID and the % cover of *Zostera noltii* are required in this file. Here they are labelled as `Quad_ID` and `X..Cover`.

For the `R` part of the processing analysis, the following packages are required: `dplyr`, `glcm`, `raster`, `rgdal`, `RStoolbox`, `tools`.

Due to the size of geographic data used in this study, only the code will be uploaded to this repository.

The following is a guide to the different components and scripts used in the processing and analysis for the paper.

Part 1: Creating Texture Bands

Image texture bands were created using the `glcm` package. The `Creating_Texture.R` script contains the code for performing this on the green band of the input orthomosaic. Bands can be outputted as either individual rasters or as a stack. This is a simple processing step, and all available texture bands in `glcm::glcm` were considered.

Part 2a: Unsupervised Classifications of RGB bands

This part of the processing workflow tests multiple combinations of input bands and classes for unsupervised classification using `RStoolbox::unsuperClass`. Example code is contained within `Unsupervised_Classifications_RGB.R`.

The `crop_and_store` function is defined at the start of the script. This is used to pull out data from the classification results map that relate to quadrat areas. It also formats the data into more easy to understand data frames.

Classification with between 2 and 5 classes are then performed on the three bands and the pixel counts for each class in each quadrat written out as text files.

Part 2b: Unsupervised Classifications of RGB and texture bands

This part of the processing workflow tests multiple combinations of input bands and classes for unsupervised classification using `RStoolbox::unsuperClass`. Example code is contained within `Unsupervised_Classifications_RGBTex.R`. This script is similar to `Unsupervised_Classifications_RGB.R` except that it includes all of the texture bands created in Part 1.

Additional parts have been added to this script:

Data is read in and an an object `combos` created. This is a list containing all possible combinations of the texture bands contained in the input raster. In the example, all texture bands are created with `glcm::glcm` and therefore there are 8 bands.

The final part of the script contains some nested loops which query the `combos` list for combinations of bands. Once a combination is selected and a stack of associated rasters created, an unsupervised classification is performed with between 2 and 5 classes. Upon each iteration of the classes loop an output data frame is created from the `crop_and_store` function. Once successfully completed, one will have 100s of text files each named iteratively containing the counts of pixels within each quadrat for each band combination and varying numbers of classes.

It is recommended that the combo loop is split among multiple `R` sessions as it takes a long time to run on big orthomosaics such as the data presented in the paper.

Part 3: Collating Classification Results & Comparing with Ground Data

In this next part of the workflow pixel counts from the unsupervised classifications are compared to ground based estimates of seagrass coverage. Example code is found within `Collating_Classification_Results_RGB.R` and `Collating_Classification_Results_RGBTex.R`.

Firstly, the `RMSD_calc` function is defined. This is used later in the script to assess the performance of different classifications in relation to ground based seagrass estimates within quadrat areas.

Next, the .csv containing Quadrat IDs and coverage estimates from the ground is read in. For this workflow, the .csv file looks as follows:

Quad_ID	X..Cover
A	45
B	32
etc.	etc.

Also, the `files` object is created, storing a list of the output text files created with `Unsupervised_Classifications_RGB.R` and `Unsupervised_Classifications_RGBTex.R`. These files are then iteratively processed in the proceeding loop. The data is converted to proportions and compared to ground estimates with the `RMSD_calc` function. These results are stored in the `results` object and the proportional data is written out on each iteration.

Finally the results data frame is formatted to display the id of the classifications with the lowest RMSD scores. From this, one can determine which classification works best by revisiting the `combos` object to determine which layers were used in that particular classifier.

At this stage, the combinations of bands/layers and classes with the lowest RMSD scores are chosen as the candidate classifiers to further analyse. One for chosen for each RGB, RGB & Texture and OBIA.

Part 4: Statistics: Calculating Uncertainty